Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 194 (2004) 588–610
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Locally divergence-free discontinuous Galerkin methods for the Maxwell equations

Bernardo Cockburn [a,1], Fengyan Li [b,2], Chi-Wang Shu [b,*,2]

[a] School of Mathematics, University of Minnesota, Minneapolis, MN 55455, USA
[b] Division of Applied Mathematics, Brown University, Box F, Providence, RI 02912, USA

Received 22 July 2003; received in revised form 8 September 2003; accepted 9 September 2003

Abstract

In this paper, we develop the locally divergence-free discontinuous Galerkin method for numerically solving the Maxwell equations. The distinctive feature of the method is the use of approximate solutions that are exactly divergence-free inside each element. As a consequence, this method has a smaller computational cost than that of the discontinuous Galerkin method with standard piecewise polynomial spaces. We show that, in spite of this fact, it produces approximations of the same accuracy. We also show that this method is more efficient than the discontinuous Galerkin method using globally divergence-free piecewise polynomial bases. Finally, a post-processing technique is used to recover $(2k+1)$th order of accuracy when piecewise polynomials of degree k are used.

Keywords: Discontinuous Galerkin method; Divergence-free solutions; Maxwell equations

1. Introduction

There are many partial differential equations with solutions which are divergence-free. Examples include the incompressible Euler and Navier–Stokes equations, the magnetohydrodynamics equations, and the Maxwell equations. For some of the problems, such as the incompressible Euler and Navier–Stokes equations, the divergence-free condition is an explicit part of the equations. For some others, such as the magnetohydrodynamics equations and the Maxwell equations, the solutions of the PDE should automatically satisfy the divergence-free condition if the initial data is divergence-free, but it is usually a challenge to have the numerical solutions also satisfy this divergence-free condition (exactly or very

* Corresponding author. Tel.: +401-863-2549; fax: +401-863-1355.
E-mail addresses: cockburn@math.umn.edu (B. Cockburn), fengyan-l@dam.brown.edu (F. Li), shu@dam.brown.edu (C.-W. Shu).
[1] Research supported in part by NSF grant DMS-0107609.
[2] Research supported by ARO grant DAAD19-00-1-0405, NSF grant DMS-0207451, NASA Langley grant NCC1-01035 and AFOSR grant F49620-02-1-0113.

0021-9991/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2003.09.007

accurately). There is an extensive literature on designing such numerical methods, for example, we could refer to [18] for the incompressible Euler and Navier–Stokes equations, [3] for the magnetohydrodynamics equations, and [19] for the Maxwell equations, among many others. It is well known that negligence in dealing with the divergence-free condition numerically can lead to serious defects, see, e.g. [16,20].

In this paper, we develop the locally divergence-free discontinuous Galerkin methods equipped with TVD Runge–Kutta time discretization (RKDG) [8,11,13] for solving two-dimensional Maxwell equations, as our starting point to explore the effective treatment of the divergence-free condition with discontinuous Galerkin methods. The same technique can be generalized to three dimensions.

The discontinuous Galerkin method is a class of finite element methods using completely discontinuous piecewise polynomial space for the numerical solution and the test functions. One would need to use more degrees of freedom for the same order of accuracy, comparing with continuous finite element methods, however, the discontinuities at the element interfaces allow the design of suitable inter-element boundary treatments (the so-called numerical fluxes) to obtain highly accurate and stable methods in many difficult situations. The discontinuous Galerkin method has become very popular in recent years for solving convection dominated problems. It has several distinct advantages. We refer to the lecture notes [7], the survey paper [9] and other papers in that Springer volume, and the review paper [13] for details and history of the discontinuous Galerkin method.

Electromagnetism is one application area of the discontinuous Galerkin method, among many other areas. In [15], the discontinuous Galerkin method with the standard piecewise polynomial spaces is used to solve Maxwell equations on unstructured meshes.The divergence-free condition is not explicitly enforced and is left to the accuracy of the solver. One can expect, and does observe, global divergence errors which are kth order small when piecewise polynomials of degree k are used.

Attempts have been made in the literature to enforce explicitly the divergence-free condition. The staggered mesh magnetic field transport algorithm was first proposed by Yee [22] for the transport of electromagnetic fields, with the idea of applying a staggered grid to maintain the divergence-free condition. Another approach is to modify the PDE by using Lagrange multipliers. In [19], Munz et al. established the generalized Lagrange multiplier approach. They rewrote the constrained formulation of the Maxwell equations by adding a coupling term into Gauss's law that resulted in a purely hyperbolic model system.

Classical finite element methods for solving Maxwell equations can be found in, e.g. [1,16]. Baker and co-workers [2,18] introduced a discontinuous Galerkin method for solving the Stokes equations and the stationary Navier–Stokes equations. They used an interior penalty method with locally divergence-free approximate solutions. Optimal error estimates were proven.

We follow the approach of Baker and co-workers [2,18] and use the locally divergence-free polynomials as trial space in the discontinuous Galerkin method to solve Maxwell equations. Note that although the resulting approximations are divergence-free inside each element, they are not globally divergence-free since their normal component across element interfaces are not necessarily continuous. We measure such divergence errors at the final time for a time dependent calculation and demonstrate numerically that they are kth order small when piecewise polynomials of degree k are used. We then show that if we project the approximate solution at the final time into the space of globally divergence-free piecewise polynomials, the result remains $(k+1)$th order accurate in the L^2- and L^∞-norms; in fact it is even more accurate than before the projection. Finally, we show that the discontinuous Galerkin method using locally divergence-free bases is in general no worse than the more costly discontinuous Galerkin method with globally divergence-free piecewise polynomial bases.

A post-processing technique [10,21] is also applied to the numerical solutions of discontinuous Galerkin methods using locally divergence-free polynomial bases, with or without the projection at the end, as well as the one using the globally divergence-free polynomial bases. In all these cases the accuracy is enhanced from $(k+1)$th order to $(2k+1)$th order when P^k elements are used, indicating that a higher order of accuracy in negative-order norms is retained by all these methods.

The paper is organized as follows. In Section 2, we introduce the locally divergence-free space and the numerical formulation of the algorithm. In the same section, a way to measure the divergence of a piecewise smooth function is also described, and the L^2-projection is defined from the space of locally divergence-free discontinuous piecewise polynomials to a subspace which contains the globally divergence-free piecewise polynomials. We also present a theoretical result of L^2-stability as well as error estimates in Section 2. Section 3 contains numerical results to demonstrate the accuracy of the algorithm and the projection. Concluding remarks are given in Section 4. In Appendix A we collect some details related to the implementation of the L^2-projection from the locally divergence-free piecewise polynomial spaces to the globally divergence-free piecewise polynomial spaces, introduced in Section 2.

2. Locally divergence-free discontinuous Galerkin method

The following two-dimensional linear Maxwell equations will be considered:

$$\frac{\partial H_x}{\partial t} = -\frac{\partial E_z}{\partial y}, \quad \frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\partial x}, \quad \frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}. \tag{2.1}$$

This is a hyperbolic system with a divergence-free solution (H_x, H_y) for all time if initially it is divergence-free. The standard RKDG method for solving (2.1) would start with a triangulation $\mathscr{T}_h$ of the domain Ω, with the element being denoted by K, the edge by e, $h = \min_K\{\text{the radius of the largest circle within } K\}$, and the outward unit normal by $\mathbf{n} = (n_1, n_2)$. The collections of all the elements and edges are denoted by $\mathscr{K}$ and $\mathscr{E}$, respectively. The solution space, which is the same as the test space, is given by

$$\bar{\mathbf{V}}_h = \bar{\mathbf{V}}_h^k = \{\mathbf{v} : \mathbf{v}|_K \in \mathbf{P}^k(K),\ K \in \mathscr{K}\} = \bar{\mathbf{V}}_{h,0}^k \oplus \{v : v|_K \in P^k(K),\ K \in \mathscr{K}\}, \tag{2.2}$$

where $\mathbf{P}^k(K) = (P^k(K))^3$, and $P^k(K)$ denotes the space of polynomials in K of degree at most k. We write the approximation space $\bar{\mathbf{V}}_{h,0}^k$ for (H_x, H_y) separately from that for E_z as we would like to replace the space $\bar{\mathbf{V}}_h$ in (2.2) by

$$\mathbf{V}_h = \mathbf{V}_h^k = \left\{\mathbf{v} \in \bar{\mathbf{V}}_h^k : \left(\frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y}\right)\Big|_K = 0\right\} = \mathbf{V}_{h,0}^k \oplus \{v : v|_K \in P^k(K),\ K \in \mathscr{K}\}. \tag{2.3}$$

That is, the vector (v_1, v_2) is a locally divergence-free polynomial vector. Notice that the dimension of $\mathbf{V}_{h,0}^k|_K$ is $(k+1)(k+4)/2$, only about half as that of the dimension of $\bar{\mathbf{V}}_{h,0}^k|_K$ which is $(k+1)(k+2)$. We can thus save a lot of computational cost by using the locally divergence-free space (2.3) instead of the standard piecewise polynomial space (2.2). It is very easy to write out a local basis for $\mathbf{V}_{h,0}^k|_K$. For example, if K is a rectangle, with center (x_i, y_j) and width $\Delta x_i, \Delta y_j$, if we denote

$$\bar{X} = \frac{x - x_i}{\Delta x_i}, \quad \bar{Y} = \frac{y - y_j}{\Delta y_j},$$

one set of bases of $\mathbf{V}_{h,0}^k|_K$ would be, when $k = 1$,

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} \Delta x_i \bar{X} \\ -\Delta y_j \bar{Y} \end{pmatrix}, \quad \begin{pmatrix} \bar{Y} \\ 0 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ \bar{X} \end{pmatrix}.$$

For $k = 2$, we need to add

$$\begin{pmatrix} \Delta x_i(12\bar{X}^2 - 1) \\ -24\Delta y_j \bar{X}\bar{Y} \end{pmatrix}, \quad \begin{pmatrix} -24\Delta x_i \bar{X}\bar{Y} \\ \Delta y_j(12\bar{Y}^2 - 1) \end{pmatrix}, \quad \begin{pmatrix} 12\bar{Y}^2 - 1 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 12\bar{X}^2 - 1 \end{pmatrix}.$$

And, by adding the following five more polynomial pairs, we get the bases for $k = 3$

$$\begin{pmatrix} \Delta x_i(4\bar{X}^3 - \bar{X}) \\ -\Delta y_j(12\bar{X}^2 - 1)\bar{Y} \end{pmatrix}, \quad \begin{pmatrix} \Delta x_i(12\bar{X}^2 - 1)\bar{Y} \\ -\Delta y_j\bar{X}(12\bar{Y}^2 - 1) \end{pmatrix}, \quad \begin{pmatrix} -\Delta x_i\bar{X}(12\bar{Y}^2 - 1) \\ \Delta y_j(4\bar{Y}^3 - \bar{Y}) \end{pmatrix},$$

$$\begin{pmatrix} 20\bar{Y}^3 - 3\bar{Y} \\ 0 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 20\bar{X}^3 - 3\bar{X} \end{pmatrix}.$$

In general, we can obtain bases for $\mathbf{V}_{h,0}^k|_K$ by taking the *curl* of bases of $P^{k+1}(K)$.

We rewrite Eq. (2.1) in the conservative form

$$\mathbf{u}_t + \nabla \cdot \mathbf{f}(\mathbf{u}) = 0, \tag{2.4}$$

where $\mathbf{u} = (H_x, H_y, E_z)^{\mathrm{T}}$. Following the usual definition of discontinuous Galerkin methods for conservation laws, e.g. [8,11], we obtain the RKDG formulation for (2.4): find $\mathbf{u_h} \in \mathbf{V_h}$, such that

$$\int_K \mathbf{u}_h t \cdot \mathbf{v}\,\mathrm{d}\mathbf{x} + \sum_{e\in\partial K} \int_e \mathbf{h}(\mathbf{u}_h^{\mathrm{int}(K)}, \mathbf{u}_h^{\mathrm{ext}(K)}, \mathbf{n}) \cdot \mathbf{v}\,\mathrm{d}s - \int_K \mathbf{f}(\mathbf{u_h}) \cdot \nabla \mathbf{v}\,\mathrm{d}\mathbf{x} = 0 \quad \forall \mathbf{v} \in \mathbf{V_h}, \tag{2.5}$$

where $\mathbf{h}(\mathbf{u}^{\mathrm{int}(K)}, \mathbf{u}^{\mathrm{ext}(K)}, \mathbf{n})$ is taken as the upwinding flux consistent with $\mathbf{f}(\mathbf{u}) \cdot \mathbf{n}$

$$\mathbf{h}(\mathbf{u}^{\mathrm{int}(K)}, \mathbf{u}^{\mathrm{ext}(K)}, \mathbf{n}) = \begin{pmatrix} n_2(\bar{E}_z - \frac{n_2}{2}[H_x] + \frac{n_1}{2}[H_y]) \\ -n_1(\bar{E}_z - \frac{n_2}{2}[H_x] + \frac{n_1}{2}[H_y]) \\ n_2\bar{H}_x - n_1\bar{H}_y - \frac{[E_z]}{2} \end{pmatrix}. \tag{2.6}$$

Here

$$\bar{v} = \frac{v^{\mathrm{int}(K)} + v^{\mathrm{ext}(K)}}{2}, \quad [v] = v^{\mathrm{ext}(K)} - v^{\mathrm{int}(K)}$$

and $v^{\mathrm{int}(K)}$, $v^{\mathrm{ext}(K)}$ are the limits of v at interface e from the interior and exterior of K respectively. The upwinding flux is obtained by diagonalizing the system in the normal direction of the cell boundary, taking the upwinding flux in each characteristic variables, and then coming back to the original variables, which is a standard technique in the computation of hyperbolic systems. An alternative choice of the numerical flux is the Lax–Friedrichs flux

$$\mathbf{h}(\mathbf{u}^{\mathrm{int}(K)}, \mathbf{u}^{\mathrm{ext}(K)}, \mathbf{n}) = \begin{pmatrix} n_2(\bar{E}_z - \frac{n_2}{2}[H_x] + \frac{n_1}{2}[H_y]) - \frac{1}{2}(n_1^2[H_x] + n_1n_2[H_y]) \\ -n_1(\bar{E}_z - \frac{n_2}{2}[H_x] + \frac{n_1}{2}[H_y]) - \frac{1}{2}(n_1n_2[H_x] + n_2^2[H_y]) \\ n_2\bar{H}_x - n_1\bar{H}_y - \frac{[E_z]}{2} \end{pmatrix}. \tag{2.7}$$

This flux has more control on the jumps of the normal velocity across element interfaces, see Proposition 2.1, and hence may show an advantage in some calculations, see, for example, the control on numerical divergence errors for the example with a singular solution in Section 3.5.

2.1. A way to measure the divergence

Although a locally divergence-free space is used, the approximation to (H_x, H_y) does have errors in its divergence given by the jumps in the normal direction across element interfaces. Thus a computable measurement of the global divergence needs to be defined.

One way is to use the H^{-1} norm of the divergence of the function. Unfortunately, it is very difficult to compute the H^{-1} norm of a function. Measurements equivalent to the H^{-1} norm can be computed, for example in [4], where a procedure to compute such a measurement is described through solving a Laplace equation with the multi-grid method.

Here, we introduce a simpler measurement. For a vector function $\mathbf{u}$ which is smooth within each element $K \in \mathscr{T}_h$, the following semi-norm is defined:

$$||\mathbf{u}||_{*,h} = \sum_{e \in \mathscr{E}} \int_e |[\mathbf{u} \cdot \mathbf{n}]| \, \mathrm{d}s + \sum_{K \in \mathscr{K}} \int_K |\nabla \cdot \mathbf{u}| \, \mathrm{d}\mathbf{x}.$$

We argue that $||\cdot||_{*,h}$ is a norm of the divergence of $\mathbf{u}$. Notice that, for a given piecewise smooth function, if its divergence in each element is zero, then it is globally divergence-free if and only if the normal component of the function across each interface e is continuous. Hence in order to measure the global divergence of a function, the jump of the normal component of the function across those edges needs to be considered.

We can easily check that the two terms in the definition of $||\cdot||_{*,h}$ are on the same scale. If we write out the definition of the H^{-1} norm of the divergence of a function,

$$\sup_{\phi} \frac{(\mathbf{u}, \nabla\phi)}{||\phi||_1},$$

where ϕ goes through all the smooth functions on Ω with compact support and replace the H^1 norm $||\cdot||_1$ used in denominator by the L^∞ norm, we will get

$$\sup_{\phi} \frac{(\mathbf{u}, \nabla\phi)}{||\phi||_\infty} = \sup_{\phi} \frac{1}{||\phi||_\infty} \left\{ \sum_{K \in \mathscr{K}} \left(\int_{\partial K} \mathbf{u} \cdot \mathbf{n} \phi \, \mathrm{d}s - \int_K \nabla \cdot \mathbf{u} \phi \, \mathrm{d}\mathbf{x} \right) \right\} \leqslant ||\mathbf{u}||_{*,h}.$$

One can prove that it is actually an equality by using in particular a sequence $\{\phi_m\}$ which approximates

$$\phi(x) = \begin{cases} \operatorname{sign}(\nabla \cdot \mathbf{u}(x)) & \forall x \in K, \\ \operatorname{sign}(\mathbf{u}(x') \cdot \mathbf{n}_{\mathbf{K}}|_{\substack{x' \in K \\ x' \to x}} + \mathbf{u}(\mathbf{x}') \cdot \mathbf{n}_{\mathbf{K}'}|_{\substack{x' \in K' \\ x' \to x}} & \forall x \in e = K \cap K', \end{cases}$$

where $\mathbf{n}_K$ is the outward unit normal on edge e for element K. We thus end up with our definition of $||\cdot||_{*,h}$.

2.2. An L^2-projection

Even if we use the locally divergence-free space, the numerical solution is still not globally divergence-free because of the discontinuities of the normal component across element interfaces. Here, we introduce the L^2-projection from the locally divergence-free piecewise polynomial space to its globally divergence-free subspace.

The original idea of the projection is borrowed from [5], only the bases and definition are modified to meet our need. To simplify the discussion, we consider only rectangular elements. Arbitrary triangular elements can also be treated in a similar fashion. Similar to [5], we first augment $(P^k(K))^2$, the polynomial space with degree at most k, by $\operatorname{span}(\operatorname{curl}(x^{k+1}y), \operatorname{curl}(xy^{k+1}))$, a two-dimensional subspace of $(P^{k+1}(K))^2$.

We introduce the notations

$$\mathbf{W}_{h,0} = \mathbf{W}_{h,0}^k = \{\mathbf{u} = (u, v) : \mathbf{u}|_K \in \mathbf{V}_{h,0}^k(K) \cup \{\Phi_k, \Psi_k\}\},$$

$$\mathbf{W}_h = \mathbf{W}_h^k = \mathbf{W}_{h,0}^k \oplus \{v : v|_K \in P^k(K),\ K \in \mathscr{K}\},$$

$$\widetilde{\mathbf{W}}_h = \widetilde{\mathbf{W}}_h^k = \{\mathbf{u} \in \mathbf{W}_{h,0}^k : \nabla \cdot \mathbf{u} \in L^2\},$$

where $\Phi_k, \Psi_k \in \mathbf{V}_{h,0}^{k+1}(K) \setminus \mathbf{V}_{h,0}^k(K)$ and $\Phi_{k,1}$ contains the term $\bar{X}^{k+1}$, $\Psi_{k,2}$ contains the term $\bar{Y}^{k+1}$. For example,

$$\Phi_1 = \begin{pmatrix} \Delta x_i(12\bar{X}^2 - 1) \\ -24\Delta y_j \bar{X}\bar{Y} \end{pmatrix}, \quad \Psi_1 = \begin{pmatrix} -24\Delta x_i \bar{X}\bar{Y} \\ \Delta y_j(12\bar{Y}^2 - 1) \end{pmatrix},$$

$$\Phi_2 = \begin{pmatrix} \Delta x_i(4\bar{X}^3 - \bar{X}) \\ -\Delta y_j(12\bar{X}^2 - 1)\bar{Y} \end{pmatrix}, \quad \Psi_2 = \begin{pmatrix} -\Delta x_i \bar{X}(12\bar{Y}^2 - 1) \\ \Delta y_j(4\bar{Y}^3 - \bar{Y}) \end{pmatrix},$$

and

$$\Phi_3 = \begin{pmatrix} \Delta x_i(80\bar{X}^4 - 24\bar{X}^2 + 1) \\ -16\Delta y_j(20\bar{X}^3 - 3\bar{X})\bar{Y} \end{pmatrix}, \quad \Psi_3 = \begin{pmatrix} -16\Delta x_i(20\bar{Y}^3 - 3\bar{Y})\bar{X} \\ \Delta y_j(80\bar{Y}^4 - 24\bar{Y}^2 + 1) \end{pmatrix}.$$

Notice that a function $\mathbf{u} \in \mathbf{W}_{h,0}$ also belongs to $\widetilde{\mathbf{W}}_h$ if and only if $\mathbf{u} \cdot \mathbf{n}$, the normal component of $\mathbf{u}$, is continuous across the interfaces e of $\mathscr{T}_h$.

The projection $\Pi_h = \Pi_h^k$ we will use here is the L^2-projection onto $\widetilde{\mathbf{W}}_h^k$. It is not surprising to see this is a global projection, since the divergence-free condition is a global property.

In Appendix A we will collect some details related to the implementation of this projection. Basically, we choose the normal component along each e as degrees of freedom to guarantee its continuity.

2.3. L^2-stability and an error estimate

We present in this section the theoretical results of the L^2-stability and an error estimate.

Proposition 2.1 (L^2-stability). *Let* $\mathbf{u}_h = (H_{x,h}, H_{y,h}, E_{z,h})$ *be the solution of* (2.5) *with the upwinding flux* (2.6). *Then,*

$$\int_\Omega (H_{x,h}(\mathbf{x}, t)^2 + H_{y,h}(\mathbf{x}, t)^2 + E_{z,h}(\mathbf{x}, t)^2)\,\mathrm{d}\mathbf{x} + \int_0^t \left\{ \sum_e \int_e (n_2[H_{x,h}] - n_1[H_{y,h}])^2 + [E_{z,h}]^2\,\mathrm{d}\mathbf{s} \right\} \mathrm{d}t$$
$$= \int_\Omega (H_{x,h}(\mathbf{x}, 0)^2 + H_{y,h}(\mathbf{x}, 0)^2 + E_{z,h}(\mathbf{x}, 0)^2)\,\mathrm{d}\mathbf{x}.$$

For the solution $\mathbf{u}_h = (H_{x,h}, H_{y,h}, E_{z,h})$ *of* (2.5) *with the Lax–Friedrichs flux* (2.7), *we have*

$$\int_\Omega (H_{x,h}(\mathbf{x},t)^2 + H_{y,h}(\mathbf{x},t)^2 + E_{z,h}(\mathbf{x},t)^2)\,\mathrm{d}\mathbf{x} + \int_0^t \left\{ \sum_e \int_e (n_2[H_{x,h}] - n_1[H_{y,h}])^2 + (n_1[H_{x,h}] + n_2[H_{y,h}])^2 + [E_{z,h}]^2\,\mathrm{d}\mathbf{s} \right\} \mathrm{d}t = \int_\Omega (H_{x,h}(\mathbf{x},0)^2 + H_{y,h}(\mathbf{x},0)^2 + E_{z,h}(\mathbf{x},0)^2)\,\mathrm{d}\mathbf{x}.$$

Proof. The techniques used are now standard for proving the L^2-stability of semidiscrete DG, see, e.g. [12,17]. In particular, the proof of this proposition follows the same lines as that for proving the cell entropy inequality in [17], by taking the test functions $\mathbf{v} = \mathbf{u}_h$ in (2.5), working through the integrals, and grouping the boundary terms using the specific forms of the upwinding or the Lax–Friedrichs fluxes. We omit the details. □

Notice that, when the upwinding flux is used, we have a control only on the jumps of the tangential velocity across element interfaces, but we have a control on the jumps of both the tangential and the normal velocity across element interfaces when the Lax–Friedrichs flux is used. Since the only numerical divergence errors are contained in the jumps of the normal velocity across element interfaces, such a stronger control when the Lax–Friedrichs flux is used may be beneficiary in certain situations, such as the situation when local divergence is concentrated near the singularity of the solution, see the example in Section 3.5.

Proposition 2.2 (Error estimate). *Let* $\mathbf{u} = (H_x, H_y, E_z)$ *be the smooth exact solution of* (2.1)*, and* $\mathbf{u}_h = (H_{x,h}, H_{y,h}, E_{z,h})$ *the solution of* (2.5) *and* (2.6) *or* (2.5)–(2.7) *in* $\mathbf{V_h^k}$ *or* $\mathbf{W_h^k}$*. Then,*

$$||\mathbf{u} - \mathbf{u}_h||_0 \leqslant C||\mathbf{u}||_{k+1} h^{k+1/2}.$$

Proof. Again, the techniques used are now standard for proving error estimates of semidiscrete DG, given a cell entropy inequality or an L^2-stability, see, e.g. [12]. In particular, the proof of this proposition follows the same lines as that for proving Theorem 2.2 in [12]. We omit the details but mention that for the error estimate, we also need the approximation results from the following Lemma 2.3. □

Lemma 2.3 (Approximation results). *Let* v *be a* H^{k+1} *function defined on* K *satisfying* $\nabla \cdot v = 0$*, and let* v_h *be its* L^2*-projection into* $\mathbf{V}_{h,0}^k|_K$ (*or* $\mathbf{W}_{h,0}^k|_K$). Then

$$||v - v_h||_{0,K} \leqslant Ch^{k+1}|v|_{k+1,K},$$

$$||v - v_h||_{0,\partial K} \leqslant Ch^{k+1/2}|v|_{k+1,K}.$$

The first part in Lemma 2.3 is a direct consequence of the approximation result in [2]: under the same condition on v, there is a function $w_h \in \mathbf{V}_{h,0}^k|_K$, such that

$$||v - w_h||_{l,K} \leqslant Ch^{k+1-l}|v|_{k+1,K}, \quad 0 \leqslant l \leqslant k+1.$$

Certainly, the L^2-error for the L^2-projection v_h will not be larger than that for w_h. The second inequality is a simple application of the Bramble–Hilbert lemma (see [6]).

Proposition 2.4 (Error estimate when the global projection is used at the end of computation). *If* $\mathbf{u} = (H_x, H_y, E_z) = (\mathbf{u}_0, E_z)$ *is the smooth exact solution,* $\mathbf{u}_h = (H_{x,h}, H_{y,h}, E_{z,h}) = (\mathbf{u}_{h,0}, E_{z,h})$ *is the solution of* (2.5) *and* (2.6) *or* (2.5)–(2.7) *with the underlying space* $\mathbf{V}_h = \mathbf{V}_h^k$ *or* $\mathbf{W}_h = \mathbf{W}_h^k$, *and* $\mathbf{v}_h = (\Pi_h(H_{x,h}, H_{y,h}), E_{z,h}) = (\Pi_h(\mathbf{u}_{h,0}), E_{z,h}) =: \mathbf{P}_h(\mathbf{u}_h)$, *where* $\Pi_h = \Pi_h^k$ *is the* L^2*-projection onto the globally divergence-free subspace, then we have*

$$||\mathbf{u} - \mathbf{v}_h||_0 \leqslant C||\mathbf{u}||_{k+1} h^{k+1/2}.$$

Proof. First,

$$||\mathbf{u} - \mathbf{v}_h||_0 = ||\mathbf{u} - \mathbf{P}_h(\mathbf{u}_h)||_0 \leqslant ||\mathbf{u} - \mathbf{P}_h(\mathbf{u})||_0 + ||\mathbf{P}_h(\mathbf{u}) - \mathbf{P}_h(\mathbf{u}_h)||_0 \leqslant ||\mathbf{u} - \mathbf{P}_h(\mathbf{u})||_0 + ||\mathbf{u} - \mathbf{u}_h||_0,$$

the last inequality is because $\mathbf{P}_h$ is an L^2-projection, hence it would not increase the L^2-norm. By Proposition 2.2, we have

$$||\mathbf{u} - \mathbf{u}_h||_0 \leqslant C||\mathbf{u}||_{k+1} h^{k+1/2}.$$

On the other hand, we claim that

$$||\mathbf{u} - \mathbf{P}_h\mathbf{u}||_0 \leqslant C||\mathbf{u}||_{k+1} h^{k+1},$$

this is because we could use the projection introduced in [5], which already yields an error bound of h^{k+1}, and the L^2-projection $\mathbf{P}_h$ would have a smaller error than *any* other projection. □

3. Numerical examples

Notice that the Maxwell equation (2.1) admit the following exact solution:

$$\begin{pmatrix} H_x \\ H_y \\ E_z \end{pmatrix} = \begin{pmatrix} -\beta \\ \alpha \\ 1 \end{pmatrix} f(\cos\omega(t + \alpha x + \beta y)), \tag{3.1}$$

where f is an arbitrary function, $\alpha,\ \beta,\ \omega$ are constants, and $\alpha^2 + \beta^2 = 1$.

In this section, we show representative examples of the computational results obtained with the following two functions f in (3.1):

$$f(w) = e^w, \tag{3.2}$$

which is smooth, and

$$f(w) = \begin{cases} w\log|w| & \text{if } w \neq 0, \\ 0 & \text{if } w = 0, \end{cases} \tag{3.3}$$

which has a singularity at $w = 0$ (but is still continuous there). We will call them the smooth solution and the singular solution respectively in the following examples. The computational domain is always taken as

$$\Omega = \left[0, \frac{2\pi}{|\omega\alpha|}\right] \times \left[0, \frac{2\pi}{|\omega\beta|}\right],$$

and the periodic boundary condition is used. We take $\omega = 1$, $\alpha = \cos(0.3\pi)$ and $\beta = \sin(0.3\pi)$ as well as the final time $t = 14$ in the numerical experiments, unless otherwise stated.

We use both the uniform and non-uniform (randomly perturbed from a uniform mesh up to 10%) rectangular meshes and the linear strong stability preserving (SSP) Runge–Kutta time discretization [14] of order comparable to the spatial accuracy. In most of the examples we will use the upwinding flux (2.6). The results using the Lax–Friedrichs flux (2.7) are similar for most cases. In the singular solution case, Section 3.5, we will show results using both fluxes.

3.1. Comparison of RKDG methods using locally divergence-free P^k and standard P^k bases

We first compare the results obtained with the P^k RKDG method using the locally divergence-free bases with those using the usual polynomial bases. Note that the number of degrees of freedom per element for the first method is $((k+1)(k+2)/2)+((k+1)(k+4)/2)$ whereas for the second is $((k+1)(k+2)/2)+(k+1)(k+2)$. This means that, as k increases, the ratio of the complexity of these methods per time step and per element, namely,

$$\mathrm{rc}(k) := \frac{\left(\frac{3(k+1)(k+2)}{2}\right)^2}{\left(\frac{(k+1)(k+2)}{2}+\frac{(k+1)(k+4)}{2}\right)^2},$$

tends to 9/4. This means that as k increases, the method using locally divergence-free basis is more than twice faster for the same mesh. In Table 1, we display, for several values of the polynomial degree k, the ratio of the complexity of the methods, $\mathrm{rc}(k)$. We remark, however, that this is based solely on the reduction of degrees of freedom and does not take into account special structures of the bases, e.g., tensor product type bases for the p-version, for which the actual savings in cost may be less.

The results for the smooth solution on non-uniform meshes are shown in Tables 2 and 3. From Table 2 we can see that the standard P^k and locally divergence-free P^k both have the same convergence order and almost the same magnitudes of L^2 and L^∞ errors on the same mesh. The errors in E_z are not listed since both test spaces give almost the same results.

Besides the saving of computational cost, another advantage of using the locally divergence-free bases can be observed in the reduction of global divergence errors in Table 3. Although the global divergence errors are still of order k, the magnitude is reduced in all cases comparing with the results using the usual polynomial spaces.

3.2. The effect of the projection to globally divergence-free subspaces at the end of the calculation

In this section, we would like to see the effect on the accuracy of the numerical solution of a single application of the projection into the subspace of globally divergence-free functions at the very end of the calculation. This can be considered to be a cosmetic post-processing step whose purpose is to provide a globally divergence-free numerical solution which is sometimes desired in applications.

We look again at the case of the smooth solution on non-uniform meshes. We first use $\mathbf{V}_h^k$ as the trial space for the RKDG, then at the very end of the computation, the $\mathbf{H}$ component of the numerical solution is projected to the augmented space $\mathbf{W}_h^k$.

Table 1
Ratio of the complexities per time step per element of the RKDG methods

k	1	2	3	4	5	6	7	8	9	10
$\mathrm{rc}(k)$	1.27	1.44	1.56	1.65	1.72	1.77	1.82	1.86	1.89	1.92

Table 2
L^2 and L^∞ errors in $\mathbf{H}$

Mesh	$\mathbf{P}^k$				Locally divergence-free $\mathbf{P}^k$			
	L^2 error	Order	L^∞ error	Order	L^2-error	Order	L^∞ error	Order
P^1								
20×20	7.41E − 02		1.85E − 01		7.41E − 02		1.81E − 01	
40×40	1.63E − 02	2.19	4.54E − 02	2.02	1.63E − 02	2.19	4.48E − 02	2.01
80×80	3.46E − 03	2.23	1.07E − 02	2.09	3.46E − 03	2.23	1.07E − 02	2.07
160×160	8.12E − 04	2.09	2.72E − 03	1.98	8.12E − 04	2.09	2.70E − 03	1.98
320×320	1.98E − 04	2.04	6.98E − 04	1.96	1.98E − 04	2.04	6.92E − 04	1.96
P^2								
10×10	3.75E − 02		9.40E − 02		3.87E − 02		9.32E − 02	
20×20	3.15E − 03	3.58	1.53E − 02	2.62	3.18E − 03	3.61	1.90E − 02	2.29
40×40	2.54E − 04	3.63	2.42E − 03	2.66	2.47E − 04	3.69	2.94E − 03	2.69
80×80	2.85E − 05	3.16	2.96E − 04	3.03	2.69E − 05	3.20	3.75E − 04	2.97
160×160	3.58E − 06	2.99	3.75E − 05	2.98	3.27E − 06	3.04	4.66E − 05	3.01
P^3								
10×10	3.94E − 03		2.81E − 02		4.03E − 03		3.08E − 02	
20×20	1.45E − 04	4.77	2.67E − 03	3.40	1.40E − 04	4.85	2.99E − 03	3.37
40×40	8.83E − 06	4.04	1.89E − 04	3.82	8.43E − 06	4.05	2.17E − 04	3.78
80×80	5.58E − 07	3.98	1.31E − 05	3.85	5.29E − 07	4.00	1.51E − 05	3.84
160×160	3.54E − 08	3.98	8.01E − 07	4.03	3.31E − 08	4.00	9.40E − 07	4.01

Table 3
Errors in the divergence of $\mathbf{H}$, $\nabla \cdot \mathbf{H}$

Mesh	$\mathbf{P}^k$		Locally divergence-free $\mathbf{P}^k$	
	$\|\|\cdot\|\|_{*,h}$ error	Order	$\|\|\cdot\|\|_{*,h}$ error	Order
P^1				
20×20	8.92E − 00		7.65E − 00	
40×40	4.76E − 00	0.91	4.04E − 00	0.92
80×80	2.42E − 00	0.98	2.07E − 00	0.96
160×160	1.28E − 00	0.92	1.10E − 00	0.91
320×320	6.85E − 01	0.90	5.86E − 01	0.91
P^2				
10×10	4.78E − 00		2.20E − 00	
20×20	1.55E − 00	1.63	6.50E − 01	1.76
40×40	4.12E − 01	1.91	1.65E − 01	1.97
80×80	1.08E − 01	1.94	4.14E − 02	2.00
160×160	2.86E − 02	1.91	1.03E − 02	2.00
P^3				
10×10	8.88E − 01		3.57E − 01	
20×20	1.38E − 01	2.69	4.56E − 02	2.97
40×40	1.81E − 02	2.93	5.30E − 03	3.10
80×80	2.38E − 03	2.93	6.40E − 04	3.05
160×160	3.25E − 04	2.87	7.92E − 05	3.01

It turns out that the projection is a very good "post-processor", see Table 4, in the sense that it improves the numerical solution in two ways: it eliminates the divergence error in the numerical solution, which is expected as the projection is designed for this purpose, *and* it reduces the L^2 and L^∞ errors for

Table 4
L^2 and L^∞ errors in $\mathbf{H}$

Mesh	Locally divergence-free				Projection onto augmented space			
	L^2 error	Order	L^∞ error	Order	L^2 error	Order	L^∞ error	Order
P^1								
10×10	2.28E − 01		5.20E − 01		2.26E − 01		5.02E − 01	
20×20	7.41E − 02	1.62	1.81E − 01	1.52	7.32E − 02	1.62	1.69E − 01	1.57
40×40	1.63E − 02	2.19	4.48E − 02	2.01	1.59E − 02	2.20	3.96E − 02	2.09
80×80	3.46E − 03	2.23	1.07E − 02	2.07	3.34E − 03	2.25	8.13E − 03	2.28
160×160	8.12E − 04	2.09	2.70E − 03	1.98	7.77E − 04	2.11	1.93E − 03	2.08
P^2								
10×10	3.87E − 02		9.32E − 02		3.83E − 02		9.17E − 02	
20×20	3.18E − 03	3.61	1.90E − 02	2.29	2.94E − 03	3.70	9.39E − 03	3.29
40×40	2.47E − 04	3.69	2.94E − 03	2.69	1.82E − 04	4.01	1.11E − 03	3.08
80×80	2.69E − 05	3.20	3.75E − 04	2.97	1.65E − 05	3.46	1.34E − 04	3.04
160×160	3.27E − 06	3.04	4.66E − 05	3.01	1.90E − 06	3.12	1.73E − 05	2.96
P^3								
10×10	4.03E − 03		3.08E − 02		4.01E − 03		1.86E − 02	
20×20	1.40E − 04	4.85	2.99E − 03	3.37	1.31E − 04	4.94	1.40E − 03	3.73
40×40	8.43E − 06	4.05	2.17E − 04	3.78	8.08E − 06	4.02	1.02E − 04	3.78
80×80	5.29E − 07	4.00	1.51E − 05	3.84	5.14E − 07	3.97	6.79E − 06	3.91
160×160	3.31E − 08	4.00	9.40E − 07	4.01	3.23E − 08	3.99	4.28E − 07	3.99

the numerical solution, which is *not* necessarily expected. Indeed, since we are projecting into a subspace, a degradation of the quality of the approximation can be expected. However, the fact that this projection eliminates the error in the divergence and simultaneously reduces the L^2 and L^∞ errors indicates that the original numerical solution by the RKDG method with locally divergence-free polynomials is already highly accurate both in the usual L^2 and L^∞ errors and in the divergence error. The projection cannot render the solution more accurate if the information is not already there hidden in the numerical solution.

3.3. Comparison of locally and globally divergence-free RKDG methods

In this section, we compare the results obtained from three RKDG methods, namely, the one using the locally divergence-free polynomial bases, the one using the locally divergence-free polynomial bases with a projection into the globally divergence-free space at the end of the computations, and the one using the globally divergence-free polynomial bases.

The last method is implemented by starting from a globally divergence-free numerical initial condition, advancing in an inner stage of the Runge–Kutta method by the DG procedure, then projecting back to the globally divergence-free space. This is clearly equivalent to working on the RKDG method using the globally divergence-free piecewise polynomial space, which is in $H(\mathrm{div})$. Note that to carry out the projection into the globally divergence-free space, we must numerically solve a large sparse linear system. For this reason, this method is computationally quite costly. We do not advocate it as a practical numerical method; we include it here mainly for the purpose of comparison.

Again we use the case of the smooth solution on non-uniform meshes. The trial space in the RKDG is now taken as $\mathbf{W}_h^k$ in order to be able to implement the global projection at every Runge–Kutta inner stage.

From Tables 5 and 6, we can see that if we use the projection at the end of the RKDG, then both the L^2 and L^∞ errors of $\mathbf{H}$ are reduced from the one before the projection. This is consistent with what we have

Table 5
L^2-errors in **H** ("LDF + No Pro, LDF + Final Pro, GDF" stand for: RKDG using locally divergence-free polynomial bases, using locally divergence-free polynomial bases with projection at the end, and using globally divergence-free polynomial bases)

Mesh	LDF + No Pro		LDF + Final Pro		GDF	
	L^2 error	Order	L^2 error	Order	L^2 error	Order
P^1						
10×10	2.33E − 01		2.32E − 01		2.29E − 01	
20×20	7.21E − 02	1.69	7.19E − 02	1.69	7.03E − 02	1.70
40×40	1.52E − 02	2.24	1.52E − 02	2.25	1.51E − 02	2.22
80×80	3.27E − 03	2.22	3.25E − 03	2.22	3.25E − 03	2.21
160×160	7.74E − 04	2.08	7.69E − 04	2.08	7.68E − 04	2.08
P^2						
10×10	3.35E − 02		3.32E − 02		3.94E − 02	
20×20	2.77E − 03	3.60	2.54E − 03	3.71	2.19E − 03	4.17
40×40	2.28E − 04	3.61	1.69E − 04	3.91	1.46E − 04	3.91
80×80	2.54E − 05	3.17	1.62E − 05	3.38	1.55E − 05	3.23
P^3						
10×10	3.94E − 03		3.78E − 03		5.35E − 03	
20×20	1.68E − 04	4.55	1.32E − 04	4.84	1.23E − 04	5.45
40×40	1.05E − 05	4.00	8.18E − 06	4.01	7.89E − 06	3.96
80×80	6.62E − 07	3.99	5.16E − 07	3.99	5.11E − 07	3.95

Table 6
L^∞-errors in **H**

Mesh	LDF + No Pro		LDF + Final Pro		GDF	
	L^∞ error	Order	L^∞ error	Order	L^∞ error	Order
P^1						
10×10	4.97E − 01		4.88E − 01		4.79E − 01	
20×20	1.66E − 01	1.58	1.59E − 01	1.62	1.57E − 01	1.61
40×40	4.02E − 02	2.05	3.68E − 02	2.11	3.66E − 02	2.10
80×80	8.57E − 03	2.23	7.76E − 03	2.25	7.76E − 03	2.24
160×160	2.12E − 03	2.02	1.91E − 03	2.02	1.91E − 03	2.02
P^2						
10×10	8.49E − 02		7.96E − 02		9.75E − 02	
20×20	1.41E − 02	2.59	8.73E − 03	3.19	9.32E − 03	3.39
40×40	2.14E − 03	2.72	1.04E − 03	3.07	1.01E − 03	3.20
80×80	2.69E − 04	2.99	1.28E − 04	3.02	1.33E − 04	2.93
P^3						
10×10	3.01E − 02		1.90E − 02		2.28E − 02	
20×20	2.92E − 03	3.36	1.44E − 03	3.73	1.27E − 03	4.17
40×40	2.08E − 04	3.81	1.02E − 04	3.82	9.45E − 05	3.75
80×80	1.39E − 05	3.90	6.67E − 06	3.93	6.59E − 06	3.84

observed in the previous subsection, in fact the only difference here is that the trial space for the RKDG is slightly larger. If we compare the results using the projection at the end of computation and results using the globally divergence-free polynomial space (remember both of them give globally divergence-free

Table 7
L^2 and L^∞ errors in E_z

Mesh	LDF + No Pro/Final Pro				GDF			
	L^2error	Order	L^∞ error	Order	L^2error	Order	L^∞ error	Order
P^1								
10×10	2.41E − 01		5.17E − 01		2.38E − 01		5.08E − 01	
20×20	7.31E − 02	1.72	1.76E − 01	1.55	7.16E − 02	1.73	1.74E − 01	1.54
40×40	1.55E − 02	2.24	4.18E − 02	2.08	1.54E − 02	2.22	4.17E − 02	2.06
80×80	3.32E − 03	2.22	9.72E − 03	2.11	3.32E − 03	2.21	9.69E − 03	2.11
160×160	7.84E − 04	2.08	2.72E − 03	1.84	7.83E − 04	2.08	2.72E − 03	1.83
P^2								
10×10	3.27E − 02		8.01E − 02		3.97E − 02		9.92E − 02	
20×20	2.57E − 03	3.67	1.13E − 02	2.83	2.28E − 03	4.12	1.25E − 02	2.99
40×40	1.83E − 04	3.81	1.72E − 03	2.72	1.68E − 04	3.76	1.89E − 03	2.72
80×80	1.88E − 05	3.28	2.15E − 04	3.00	1.89E − 05	3.15	2.46E − 04	2.95
P^3								
10×10	3.53E − 03		1.90E − 02		5.44E − 03		2.60E − 02	
20×20	9.19E − 05	5.27	1.76E − 03	3.43	1.39E − 04	5.29	2.06E − 03	3.66
40×40	5.12E − 06	4.17	1.24E − 04	3.82	1.00E − 05	3.79	1.66E − 04	3.63
80×80	3.18E − 07	4.01	8.78E − 06	3.82	6.89E − 07	3.85	1.18E − 05	3.81

Table 8
L^2 and L^∞ errors in $\mathbf{H}$ with LDF + No Pro

Mesh	Before post-processing				After post-processing			
	L^2 error	Order	L^∞ error	Order	L^2 error	Order	L^∞ error	Order
P^1								
10×10	2.03E − 01		4.71E − 01		2.02E − 01		4.41E − 01	
20×20	5.74E − 02	1.82	1.49E − 01	1.66	5.69E − 02	1.83	1.31E − 01	1.75
40×40	9.83E − 03	2.55	3.01E − 02	2.30	9.55E − 03	2.58	2.38E − 02	2.45
80×80	1.43E − 03	2.79	5.13E − 03	2.55	1.28E − 03	2.90	3.30E − 03	2.85
160×160	2.28E − 04	2.64	9.09E − 04	2.50	1.62E − 04	2.98	4.20E − 04	2.97
P^2								
10×10	3.01E − 02		7.32E − 02		3.06E − 02		6.89E − 02	
20×20	2.31E − 03	3.71	1.28E − 02	2.52	1.84E − 03	4.06	4.73E − 03	3.86
40×40	1.99E − 04	3.53	1.77E − 03	2.85	6.70E − 05	4.78	1.80E − 04	4.71
80×80	2.38E − 05	3.07	2.25E − 04	2.97	2.17E − 06	4.95	5.89E − 06	4.94
160×160	2.96E − 06	3.00	2.82E − 05	3.00	6.85E − 08	4.99	1.86E − 07	4.98
P^3								
10×10	3.96E − 03		2.42E − 02		5.93E − 03		1.36E − 02	
20×20	1.79E − 04	4.47	2.43E − 03	3.32	6.05E − 05	6.62	1.64E − 04	6.37
40×40	1.12E − 05	4.00	1.60E − 04	3.92	3.75E − 07	7.33	1.05E − 06	7.29
80×80	7.01E − 07	3.99	9.95E − 06	4.01	3.02E − 09	6.96	7.68E − 09	7.10
160×160	4.38E − 08	4.00	6.21E − 07	4.00	5.05E − 11	5.90	1.12E − 10	6.10

solution $\mathbf{H}$), they have the same order of accuracy in L^2 and L^∞, and the latter one, which is much more computationally expensive, does not give much better results than the former one, especially for the component E_z (see Table 7).

Table 9
L^2 and L^∞ errors in **H** with LDF + Final Pro

Mesh	Before post-processing				After post-processing			
	L^2 error	Order	L^∞ error	Order	L^2 error	Order	L^∞ error	Order
P^1								
10×10	2.03E − 01		4.62E − 01		2.02E − 01		4.41E − 01	
20×20	5.73E − 02	1.82	1.42E − 01	1.70	5.69E − 02	1.83	1.31E − 01	1.75
40×40	9.77E − 03	2.55	2.75E − 02	2.37	9.55E − 03	2.58	2.38E − 02	2.45
80×80	1.39E − 03	2.81	4.37E − 03	2.65	1.28E − 03	2.90	3.30E − 03	2.85
160×160	2.12E − 04	2.71	6.99E − 04	2.64	1.62E − 04	2.98	4.20E − 04	2.97
P^2								
10×10	2.96E − 02		6.89E − 02		3.07E − 02		6.92E − 02	
20×20	1.99E − 03	3.89	7.21E − 03	3.25	1.84E − 03	4.06	4.74E − 03	3.87
40×40	1.18E − 04	4.07	7.78E − 04	3.21	6.70E − 05	4.78	1.81E − 04	4.71
80×80	1.20E − 05	3.30	9.22E − 05	3.08	2.17E − 06	4.95	5.89E − 06	4.94
160×160	1.46E − 06	3.04	1.14E − 05	3.02	6.85E − 08	4.98	1.86E − 07	4.98
P^3								
10×10	3.73E − 03		1.66E − 02		5.94E − 03		1.37E − 02	
20×20	1.34E − 04	4.80	1.16E − 03	3.84	6.05E − 05	6.62	1.65E − 04	6.38
40×40	8.28E − 06	4.01	7.28E − 05	3.99	3.73E − 07	7.34	1.05E − 06	7.29
80×80	5.23E − 07	3.99	4.60E − 06	3.99	2.65E − 09	7.13	7.47E − 09	7.14
160×160	3.28E − 08	4.00	2.89E − 07	3.99	3.05E − 11	6.44	1.01E − 10	6.21

Table 10
L^2 and L^∞ errors in **H** with GDF

Mesh	Before post-processing				After post-processing			
	L^2 error	Order	L^∞ error	Order	L^2 error	Order	L^∞ error	Order
P^1								
10×10	1.94E − 01		4.46E − 01		1.94E − 01		4.23E − 01	
20×20	5.48E − 02	1.82	1.37E − 01	1.70	5.45E − 02	1.83	1.25E − 01	1.75
40×40	9.58E − 03	2.52	2.70E − 02	2.34	9.36E − 03	2.54	2.33E − 02	2.43
80×80	1.38E − 03	2.79	4.35E − 03	2.64	1.27E − 03	2.88	3.28E − 03	2.83
160×160	2.12E − 04	2.70	6.99E − 04	2.64	1.62E − 04	2.97	4.20E − 04	2.97
P^2								
10×10	3.74E − 02		8.80E − 02		3.79E − 02		7.88E − 02	
20×20	1.76E − 03	4.41	6.61E − 03	3.73	1.60E − 03	4.57	4.12E − 03	4.26
40×40	1.03E − 04	4.09	7.37E − 04	3.16	4.03E − 05	5.31	1.11E − 04	5.22
80×80	1.18E − 05	3.13	9.12E − 05	3.01	1.18E − 06	5.10	3.21E − 06	5.11
P^3								
10×10	5.25E − 03		1.81E − 02		7.20E − 03		1.62E − 02	
20×20	1.23E − 04	5.42	1.01E − 03	4.17	6.21E − 05	6.86	1.72E − 04	6.55
40×40	7.96E − 06	3.95	6.99E − 05	3.85	4.82E − 07	7.01	1.34E − 06	7.01
80×80	5.17E − 07	3.94	4.54E − 06	3.94	4.60E − 09	6.71	1.42E − 08	6.56

3.4. Post-processing to enhance accuracy

In this section, we would like to see the accuracy of the three RKDG methods: using the locally divergence-free polynomial bases, using the locally divergence-free polynomial bases with a projection to the

Table 11
L^2 and L^∞ errors in E_z with LDF + No Pro/+Final Pro

Mesh	Before post-processing				After post-processing			
	L^2 error	Order	L^∞ error	Order	L^2 error	Order	L^∞ error	Order
P^1								
10×10	2.06E − 01		4.82E − 01		2.05E − 01		4.45E − 01	
20×20	5.77E − 02	1.84	1.47E − 01	1.71	5.68E − 02	1.85	1.31E − 01	1.77
40×40	9.99E − 03	2.53	3.01E − 02	2.29	9.54E − 03	2.57	2.39E − 02	2.45
80×80	1.49E − 03	2.74	5.02E − 03	2.58	1.28E − 03	2.90	3.30E − 03	2.85
160×160	2.52E − 04	2.57	1.50E − 03	1.74	1.62E − 04	2.98	4.20E − 04	2.97
P^2								
10×10	2.93E − 02		7.00E − 02		3.04E − 02		6.87E − 02	
20×20	2.04E − 03	3.84	9.57E − 03	2.87	1.84E − 03	4.05	4.73E − 03	3.86
40×40	1.40E − 04	3.87	1.29E − 03	2.89	6.71E − 05	4.78	1.80E − 04	4.71
80×80	1.55E − 05	3.17	1.63E − 04	2.98	2.18E − 06	4.95	5.89E − 06	4.94
160×160	1.92E − 06	3.01	2.03E − 05	3.00	6.86E − 08	4.99	1.86E − 07	4.98
P^3								
10×10	3.49E − 03		1.62E − 02		5.93E − 03		1.36E − 02	
20×20	9.14E − 05	5.25	1.22E − 03	3.73	6.03E − 05	6.62	1.64E − 04	6.38
40×40	4.99E − 06	4.20	7.03E − 05	4.11	3.71E − 07	7.34	1.05E − 06	7.29
80×80	3.08E − 07	4.02	4.31E − 06	4.03	2.65E − 09	7.13	7.45E − 09	7.14
160×160	1.92E − 08	4.00	2.69E − 07	4.00	2.03E − 11	7.03	5.53E − 11	7.07

Table 12
L^2 and L^∞ errors in E_z with GDF

Mesh	Before post-processing				After post-processing			
	L^2 error	Order	L^∞ error	Order	L^2 error	Order	L^∞ error	Order
P^1								
10×10	1.98E − 01		4.67E − 01		1.96E − 01		4.27E − 01	
20×20	5.53E − 02	1.84	1.43E − 01	1.71	5.44E − 02	1.85	1.26E − 01	1.76
40×40	9.82E − 03	2.49	2.96E − 02	2.27	9.35E − 03	2.54	2.34E − 02	2.43
80×80	1.49E − 03	2.72	5.00E − 03	2.57	1.27E − 03	2.88	3.28E − 03	2.83
160×160	2.52E − 04	2.56	1.51E − 03	1.72	1.62E − 04	2.97	4.20E − 04	2.97
P^2								
10×10	3.78E − 02		9.00E − 02		3.79E − 02		7.90E − 02	
20×20	1.87E − 03	4.34	9.09E − 03	3.31	1.60E − 03	4.57	4.12E − 03	4.26
40×40	1.30E − 04	3.84	1.29E − 03	2.82	4.03E − 05	5.31	1.10E − 04	5.22
80×80	1.54E − 05	3.08	1.63E − 04	2.99	1.18E − 06	5.10	3.21E − 06	5.11
P^3								
10×10	5.35E − 03		2.31E − 02		7.20E − 03		1.62E − 02	
20×20	1.38E − 04	5.28	1.64E − 03	3.81	6.21E − 05	6.86	1.72E − 04	6.55
40×40	9.84E − 06	3.81	1.24E − 04	3.72	4.82E − 07	7.01	1.34E − 06	7.01
80×80	6.79E − 07	3.86	8.67E − 06	3.84	4.46E − 09	6.76	1.37E − 08	6.61

globally divergence-free space at the end, and using the globally divergence-free polynomial bases, on the known higher-order convergence rates in negative-order norms. We use uniform meshes in this section for the smooth solution and perform a local post-processing technique [10,21] to the numerical solutions of the three RKDG methods mentioned above, to see if the accuracy can be enhanced from (k+1)th order to

Table 13
L^2 and L^∞ errors in **H** with LDF + No Pro

Mesh	Before post-processing				After post-processing			
	L^2 error	Order	L^∞ error	Order	L^2 error	Order	L^∞ error	Order
P^1								
10×10	3.64E − 02		1.46E − 01		3.43E − 02		5.97E − 02	
20×20	6.64E − 03	2.46	2.08E − 02	2.81	7.64E − 04	5.49	4.37E − 03	3.77
40×40	1.69E − 03	1.97	4.55E − 03	2.19	2.96E − 04	1.37	7.92E − 04	2.46
80×80	4.34E − 04	1.96	1.19E − 03	1.93	9.69E − 06	4.93	1.04E − 04	2.93
160×160	1.08E − 04	2.01	3.03E − 04	1.98	1.58E − 06	2.62	8.56E − 06	3.60
P^2								
10×10	1.38E − 02		7.18E − 02		2.24E − 02		4.76E − 02	
20×20	1.54E − 03	3.16	6.71E − 03	3.42	1.11E − 03	4.34	4.67E − 03	3.35
40×40	1.85E − 04	3.06	1.53E − 03	2.13	1.03E − 04	3.43	4.27E − 04	3.45
80×80	1.32E − 05	3.80	8.67E − 05	4.15	3.74E − 06	4.78	1.89E − 05	4.50
160×160	1.65E − 06	3.06	1.20E − 05	2.85	3.40E − 08	6.78	4.87E − 07	5.28
P^3								
10×10	8.05E − 03		5.71E − 02		2.60E − 02		4.86E − 02	
20×20	3.61E − 04	4.48	2.17E − 03	4.72	1.13E − 03	4.52	6.13E − 03	2.99
40×40	2.83E − 05	3.68	2.49E − 04	3.13	2.38E − 05	5.57	9.20E − 05	6.06
80×80	1.15E − 06	4.62	5.57E − 06	5.48	9.46E − 07	4.65	5.19E − 06	4.15
160×160	4.50E − 08	4.68	3.31E − 07	4.07	8.59E − 09	6.78	1.21E − 07	5.42

Table 14
L^2 and L^∞ errors in **H** with LDF + Final Pro

Mesh	Before post-processing				After post-processing			
	L^2 error	Order	L^∞ error	Order	L^2 error	Order	L^∞ error	Order
P^1								
10×10	2.78E − 02		1.00E − 01		3.41E − 02		5.67E − 02	
20×20	5.36E − 03	2.37	1.33E − 02	2.91	9.01E − 04	5.24	4.82E − 03	3.56
40×40	1.55E − 03	1.79	4.10E − 03	1.70	2.99E − 04	1.59	8.05E − 04	2.58
80×80	4.16E − 04	1.90	1.07E − 03	1.94	9.62E − 06	4.96	1.02E − 04	2.98
160×160	1.04E − 04	2.00	2.70E − 04	1.98	1.57E − 06	2.61	8.61E − 06	3.57
P^2								
10×10	2.34E − 02		9.19E − 02		2.46E − 02		4.97E − 02	
20×20	1.40E − 03	4.06	4.17E − 03	4.46	1.17E − 03	4.40	4.91E − 03	3.34
40×40	1.45E − 04	3.27	1.05E − 03	1.99	1.06E − 04	3.46	4.34E − 04	3.50
80×80	6.98E − 06	4.38	3.79E − 05	4.79	4.09E − 06	4.70	2.36E − 05	4.20
160×160	7.89E − 07	3.14	5.02E − 06	2.91	3.75E − 08	6.77	5.45E − 07	5.44
P^3								
10×10	6.23E − 03		2.68E − 02		2.61E − 02		4.84E − 02	
20×20	3.67E − 04	4.08	1.64E − 03	4.03	1.18E − 03	4.47	6.27E − 03	2.95
40×40	2.58E − 05	3.83	1.22E − 04	3.74	2.32E − 05	5.67	8.72E − 05	6.17
80×80	1.05E − 06	4.62	5.35E − 06	4.52	9.36E − 07	4.63	5.09E − 06	4.10
160×160	3.37E − 08	4.96	1.71E − 07	4.96	9.60E − 09	6.61	1.27E − 07	5.33

$(2k+1)$th order when P^k elements are used. We clearly see from Tables 8–12 that $(2k+1)$th order accuracy is achieved after the post-processing for all three cases. A careful inspection of Tables 8 and 9 reveals that RKDG using locally divergence-free polynomial space, with or without projection at the end, gives almost

Table 15
L^2 and L^∞ errors in **H** with GDF

Mesh	Before post-processing				After post-processing			
	L^2 error	Order	L^∞ error	Order	L^2 error	Order	L^∞ error	Order
P^1								
10×10	2.63E − 02		8.31E − 02		3.36E − 02		5.44E − 02	
20×20	5.59E − 03	2.23	1.34E − 02	2.63	1.79E − 03	4.23	4.21E − 03	3.69
40×40	1.56E − 03	1.84	4.12E − 03	1.70	2.23E − 04	3.00	4.97E − 04	3.08
80×80	4.16E − 04	1.91	1.07E − 03	1.94	1.31E − 05	4.09	1.37E − 04	1.86
160×160	1.04E − 04	2.00	2.70E − 04	1.99	1.51E − 06	3.12	8.36E − 06	4.03
P^2								
10×10	2.47E − 02		8.97E − 02		2.37E − 02		4.88E − 02	
20×20	3.13E − 03	2.98	1.66E − 02	2.43	1.63E − 03	3.86	5.18E − 03	3.24
40×40	3.09E − 04	3.34	2.22E − 03	2.91	1.64E − 04	3.32	7.45E − 04	2.80
80×80	7.46E − 06	5.37	5.14E − 05	5.43	1.86E − 06	6.46	5.97E − 06	6.96
160×160	7.84E − 07	3.25	5.11E − 06	3.33	6.08E − 09	8.26	3.41E − 08	7.45
P^3								
10×10	1.48E − 02		5.15E − 02		2.59E − 02		4.86E − 02	
20×20	3.22E − 03	2.20	9.21E − 03	2.48	1.52E − 03	4.09	6.61E − 03	2.88
40×40	3.41E − 04	3.24	1.40E − 03	2.72	9.23E − 05	4.04	3.05E − 04	4.44
80×80	3.10E − 05	3.46	2.12E − 04	2.72	1.04E − 05	3.15	5.52E − 05	2.47
160×160	2.01E − 06	3.95	1.626E − 05	3.71	6.10E − 07	4.10	4.92E − 06	3.49

the same L^2 and L^∞ error for **H** after post-processing, which in some cases, is smaller than the one from the RKDG using globally divergence-free polynomial bases after post-processing. This seems also to be the case for E_z.

Since this post-processing technique is based on the assumption of a $(2k+1)$th order convergence rate in the negative-order k norm of the numerical solution, a successful enhancement after the post-processing to $(2k+1)$th order accuracy is an indication that the global projection to the divergence-free subspace applied

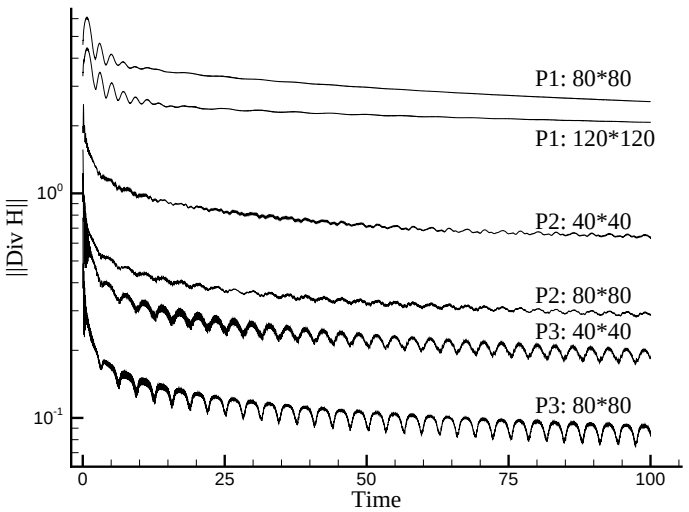

Fig. 1. The divergence of **H** against time t for the singular solution.

at the end of computation does not affect the high-order convergence rate in the negative-order norm. This is not surprising since the projection involved is an L^2-projection, which should not affect negative-order norms.

The computational results for the smooth solution (3.1) and (3.2) can be summarized as follows:

1. The RKDG methods using the regular piecewise polynomial bases and the locally divergence-free piecewise polynomial bases give comparable L^2 and L^∞ errors, although the latter uses much fewer degrees of freedom and gives smaller global divergence errors.
2. With the RKDG method using the locally divergence-free polynomial bases and the projection into the globally divergence-free subspace at the end, we obtain *smaller* L^2 and L^∞ errors than before this projection. These errors are no worse than those obtained with the RKDG method using the globally divergence-free polynomial space, which is much more computational expensive.
3. A post-processing of [10,21] after the global projection still recovers $(2k+1)$th order accuracy, indicating that the removal of the error in the divergence from the final numerical solution, neither increases (it even decreases) the L^2 and L^∞ errors, nor affects the faster convergence in the negative-order k norm.

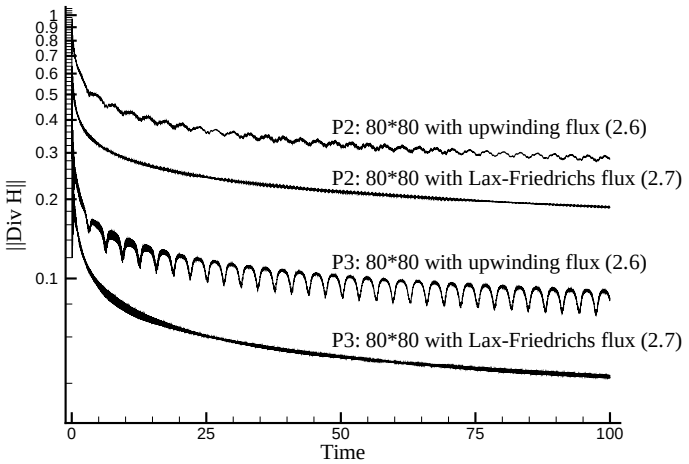

Fig. 2. Comparison of the time evolution of divergence using the upwinding flux (2.6) and using the Lax–Friedrichs flux (2.7).

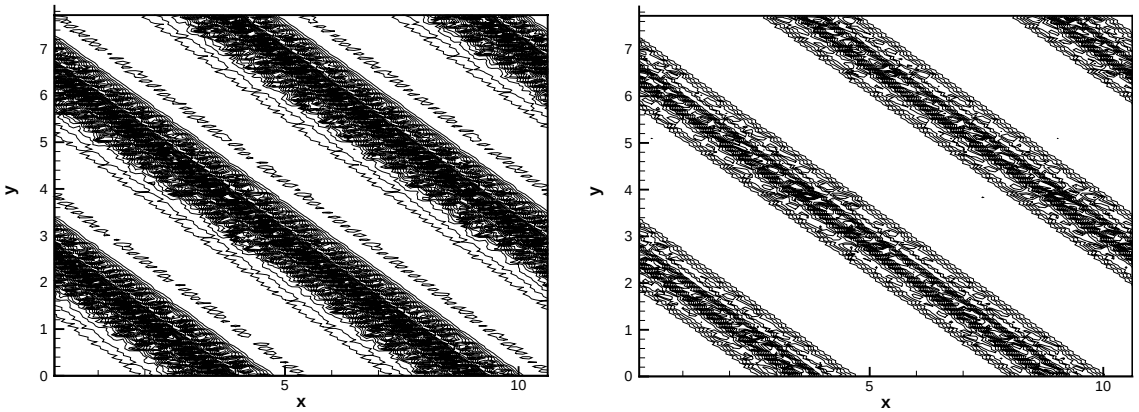

Fig. 3. The edgewise divergence at $t = 100$ with $\mathbf{V}_h^2$ on a $80 * 80$ mesh. Twenty contours from 0 to 0.00014. Left: using the upwinding flux (2.6); right: using the Lax–Friedrichs flux (2.7).

3.5. The example with the singular solution

The singularity at $t = 0$ propagates along two characteristics. We will look at the errors in the smooth region not polluted by the crossing of characteristics from the singularity. The data here are computed at $t = 0.4$, in the unpolluted area $\{(x, y) : |\cos(\omega(\alpha x + \beta y + t))| > 0.9\}$. See Tables 13–15. We use uniform meshes here and look at the errors both before and after the post-processing. The results are very similar to those obtained before for globally smooth solutions, when the global projection is not used or when it is used only once at the end, indicating that the global projection to the divergence-free subspace, when used at the end, has not polluted the errors globally. However, when the globally divergence-free polynomial space is used for RKDG, we do observe some pollution, especially in the higher-order P^3 case. This indicates that it might not be a good idea to work in the globally divergence-free space for problem with singular solutions, as functions in it are too global.

It is of particular interest to monitor the size of global divergence of the RKDG solution using the locally divergence-free bases, before the projection is performed. If this error is under control, then the projection has a chance of removing it without affecting accuracy otherwise. In Fig. 1, we plot the global divergence error $||\nabla \cdot \mathbf{H}||_{*,h}$ against time t when $\mathbf{V}_h$ is the solution space. We can see there is a decay trend of the magnitude of the divergence, which is very nice for such singular solutions. We note that the Lax–Friedrichs flux (2.7) has a better control on the jumps of the normal velocity across element interfaces, which are the only source of numerical divergence error in our method. Thus, we compare in Fig. 2 the global divergence error $||\nabla \cdot \mathbf{H}||_{*,h}$ against time t when the upwinding flux (2.6) and the Lax–Friedrichs flux (2.7) are used, respectively. We can see that the method with the Lax–Friedrichs flux has smaller numerical divergence errors for the same mesh, but both methods have a decaying numerical divergence. Finally, we plot in Fig. 3 the "edgewise" divergence $\int_e |[\mathbf{H} \cdot \mathbf{n}]| \, ds$ at $t = 100$ for the methods with the upwinding flux (2.6) (left) and with the Lax–Friedrichs flux (2.7) (right). We can see that the numerical divergence are concentrated along the region where the solution is singular, and the results with the Lax–Friedrichs flux have smaller numerical divergence and narrower numerical divergence regions.

4. Concluding remarks

Discontinuous Galerkin methods using a locally divergence-free polynomial bases seems to be very effective for solving the Maxwell equations. After a final projection into a globally divergence-free subspace, the numerical solution maintains $(k + 1)$th order accuracy in the L^2- and L^∞-norms and $(2k + 1)$th order accuracy after post-processing, and it is no worse than using the globally divergence-free polynomial bases. This implies that the original solution without the final projection is already very accurate both in the L^2- and L^∞-norms and in divergence error, for the removal of the latter does not increase the former.

Appendix A

In this section we describe a few key details related to the implementation of the L^2-projection into the globally divergence-free piecewise polynomial spaces introduced in Section 2. The basic idea is to choose the normal components along the edges as degrees of freedom to make it easy to guarantee its continuity.

A.1. The P^1 case

We first look at one rectangular element K with center (x_i, y_j), edge lengths $\Delta x_i, \Delta y_j$, and edges e_1, e_2, e_3, e_4 starting from the bottom and counting counter-clockwisely.

A function $\mathbf{u} \in \widetilde{\mathbf{W}}_h^1$ restricted on this element K can be written as

$$\mathbf{u} = \sum_{k=1}^{7} u_k \Theta_k = u_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + u_2 \begin{pmatrix} \Delta x_i \bar{X} \\ -\Delta y_j \bar{Y} \end{pmatrix} + u_3 \begin{pmatrix} \bar{Y} \\ 0 \end{pmatrix} + u_4 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + u_5 \begin{pmatrix} 0 \\ \bar{X} \end{pmatrix} + u_6 \begin{pmatrix} \Delta x_i (12\bar{X}^2 - 1) \\ -24\Delta y_j \bar{X}\bar{Y} \end{pmatrix} + u_7 \begin{pmatrix} -24\Delta x_i \bar{X}\bar{Y} \\ \Delta y_j (12\bar{Y}^2 - 1) \end{pmatrix},$$

and the normal components along $e_k,\ k = 1, \ldots, 4$ are $a_1 + b_1\bar{X}$, $a_2 + b_2\bar{Y}$, $a_3 + b_3\bar{X}$, and $a_4 + b_4\bar{Y}$. Then, the following relations:

$$u_1 = \frac{a_2 + a_4}{2} - 2u_6\Delta x_i, \quad u_2 = \frac{a_2 - a_4}{\Delta x_i}, \quad u_4 = \frac{a_3 + a_1}{2} - 2u_7\Delta y_j,$$
$$u_3 = \frac{b_2 + b_4}{2}, \quad u_7 = \frac{b_4 - b_2}{24\Delta x_i}, \quad u_5 = \frac{b_1 + b_3}{2}, \quad u_6 = \frac{b_1 - b_3}{24\Delta y_j}$$

as well as the constraint

$$(a_2 - a_4)\Delta y_j + (a_3 - a_1)\Delta x_i = 0,$$

which comes from the fact that $\mathbf{u}$ is divergence-free in this cell, need to be satisfied. Thus, in this particular cell, we can choose $a_k, b_l,\ k, l = 1, \ldots, 4$ as the degrees of freedom but with one constraint on a_k.

We now consider the whole domain Ω and assume periodic boundary conditions for simplicity (this assumption is not essential). For each edge $e \in \mathscr{E}$, we have one independent degree of freedom for 'b', called b_e, so in total there are $2 \times m \times n$ (assuming there are $m \times n$ rectangular elements) degrees of freedom corresponding to 'b'.

As for 'a', the situation is a little bit more complicated. For each $e \in \mathscr{E}$, first an a_e can be assigned. However, in each element, there is one constraint for the four a_es related to this cell, so it finally ends up that there are $m \times n + 1$ degrees of freedom for 'a' for partition $\mathscr{T}_h$ (notice it seems there should be $2 \times m \times n - m \times n$ degrees of freedom, yet notice there are actually only $m \times n - 1$ independent constraints since the function is divergence-free in each cell). In principle, we can arbitrarily pick up a way to determine on which $m \times n + 1$ edges there are degrees of freedom for 'a'.

Once the degrees of freedom are chosen, we can define the bases for $\widetilde{\mathbf{W}}_h^1$ simply by letting each degree of freedom as 1 and the rest as 0. Notice a basis function related to each b_e just has a support of two-elements, yet the one related to 'a' is quite global; this reflects the global nature of the divergence-free condition.

For those basis functions related to 'b', we can simply store the indices of the two supporting elements. As for the basis functions related to 'a', since they might have global support, there is a problem on how to store the information efficiently. We use the sparse matrix storage technique to record the support and the corresponding coefficient of each basis function. The same technique is also used to store the projection matrix due to its sparseness and large size. The projection matrix is of course symmetric positive definite, which enables us to use the preconditioned conjugated gradient (CG) method to carry out the projection. These techniques are also used for the P^k cases described below with $k > 1$.

A.2. The P^2 case

The P^2 case has basically the same setting as the P^1 case.

Assuming $\mathbf{u} \in \widetilde{\mathbf{W}}_h^2$ has the following form on an element K:

$$\mathbf{u} = \sum_{k=1}^{11} u_k \Theta_k = u_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + u_2 \begin{pmatrix} \Delta x_i \bar{X} \\ -\Delta y_j \bar{Y} \end{pmatrix} + u_3 \begin{pmatrix} \bar{Y} \\ 0 \end{pmatrix} + u_4 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + u_5 \begin{pmatrix} 0 \\ \bar{X} \end{pmatrix}$$
$$+ u_6 \begin{pmatrix} \Delta x_i (12\bar{X}^2 - 1) \\ -24 \Delta y_j \bar{X} \bar{Y} \end{pmatrix} + u_7 \begin{pmatrix} -24 \Delta x_i \bar{X} \bar{Y} \\ \Delta y_j (12\bar{Y}^2 - 1) \end{pmatrix} + u_8 \begin{pmatrix} 12\bar{Y}^2 - 1 \\ 0 \end{pmatrix} + u_9 \begin{pmatrix} 0 \\ 12\bar{X}^2 - 1 \end{pmatrix}$$
$$+ u_{10} \begin{pmatrix} \Delta x_i (4\bar{X}^3 - \bar{X}) \\ -\Delta y_j (12\bar{X}^2 - 1) \bar{Y} \end{pmatrix} + u_{11} \begin{pmatrix} -\Delta x_i \bar{X} (12\bar{Y}^2 - 1) \\ \Delta y_j (4\bar{Y}^3 - \bar{Y}) \end{pmatrix},$$

and the normal components along $e_k,\ k = 1, \ldots, 4$ are $a_1 + b_1 \bar{X} + c_1 (12\bar{X}^2 - 1)$, $a_2 + b_2 \bar{Y} + c_2 (12\bar{Y}^2 - 1)$, $a_3 + b_3 \bar{X} + c_3 (12\bar{X}^2 - 1)$, and $a_4 + b_4 \bar{Y} + c_4 (12\bar{Y}^2 - 1)$, then the following relations need to be satisfied

$$u_1 = \frac{a_2 + a_4}{2} - 2u_6 \Delta x_i, \quad u_2 = \frac{a_2 - a_4}{\Delta x_i}, \quad u_4 = \frac{a_3 + a_1}{2} - 2u_7 \Delta y_j$$
$$u_3 = \frac{b_2 + b_4}{2}, \quad u_7 = \frac{b_4 - b_2}{24 \Delta x_i}, \quad u_5 = \frac{b_1 + b_3}{2}, \quad u_6 = \frac{b_1 - b_3}{24 \Delta y_j}$$
$$u_8 = \frac{c_2 + c_4}{2}, \quad u_{11} = \frac{c_4 - c_2}{\Delta x_i}, \quad u_9 = \frac{c_1 + c_3}{2}, \quad u_{10} = \frac{c_1 - c_3}{\Delta y_j}$$

as well as the same constraint as in the P^1 case

$$(a_2 - a_4) \Delta y_j + (a_3 - a_1) \Delta x_i = 0.$$

Thus, we can choose a_i, b_j, c_k, $i, j, k = 1, \ldots, 4$ as the degrees of freedom but with one constraint on a_k.

Now, there are three groups of degrees of freedom, corresponding to '*a*', '*b*', '*c*'. The basis functions related to '*a*' still represent the global nature of the divergence-free condition and might have very wide support; those related to '*b*' and '*c*' are local, and each of them has a two element support. The dimension of $\widetilde{\mathbf{W}}_h^2$ is $5 \times m \times n + 1$.

Notice that in P^2, the '*c*' group basis functions have the same pattern of support as the '*b*' group, so basically we just need to store the support of the '*a*' group and that of the '*b*' group.

A.3. The P^3 case

Assuming $\mathbf{u} \in \widetilde{\mathbf{W}}_h^3$ has the following form on an element K:

$$\mathbf{u} = \sum_{k=1}^{16} u_k \Theta_k = u_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + u_2 \begin{pmatrix} \Delta x_i \bar{X} \\ -\Delta y_j \bar{Y} \end{pmatrix} + u_3 \begin{pmatrix} \bar{Y} \\ 0 \end{pmatrix} + u_4 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + u_5 \begin{pmatrix} 0 \\ \bar{X} \end{pmatrix} + u_6 \begin{pmatrix} \Delta x_i (12\bar{X}^2 - 1) \\ -24 \Delta y_j \bar{X} \bar{Y} \end{pmatrix}$$
$$+ u_7 \begin{pmatrix} -24 \Delta x_i \bar{X} \bar{Y} \\ \Delta y_j (12\bar{Y}^2 - 1) \end{pmatrix} + u_8 \begin{pmatrix} 12\bar{Y}^2 - 1 \\ 0 \end{pmatrix} + u_9 \begin{pmatrix} 0 \\ 12\bar{X}^2 - 1 \end{pmatrix} + u_{10} \begin{pmatrix} \Delta x_i (4\bar{X}^3 - \bar{X}) \\ -\Delta y_j (12\bar{X}^2 - 1) \bar{Y} \end{pmatrix}$$
$$+ u_{11} \begin{pmatrix} \Delta x_i (12\bar{X}^2 - 1) \bar{Y} \\ -\Delta y_j \bar{X} (12\bar{Y}^2 - 1) \end{pmatrix} + u_{12} \begin{pmatrix} -\Delta x_i \bar{X} (12\bar{Y}^2 - 1) \\ \Delta y_j (4\bar{Y}^3 - \bar{Y}) \end{pmatrix} + u_{13} \begin{pmatrix} 20\bar{Y}^3 - 3\bar{Y} \\ 0 \end{pmatrix} + u_{14} \begin{pmatrix} 0 \\ 20\bar{X}^3 - 3\bar{X} \end{pmatrix}$$
$$+ u_{15} \begin{pmatrix} \Delta x_i (80\bar{X}^4 - 24\bar{X}^2 + 1) \\ -16 \Delta y_j (20\bar{X}^3 - 3\bar{X}) \bar{Y} \end{pmatrix} + u_{16} \begin{pmatrix} -16 \Delta x_i (20\bar{Y}^3 - 3\bar{Y}) \bar{X} \\ \Delta y_j (80\bar{Y}^4 - 24\bar{Y}^2 + 1) \end{pmatrix},$$

and the normal components along $e_k, k = 1, \ldots, 4$ are:

$$a_1 + b_1\bar{X} + c_1(12\bar{X}^2 - 1) + d_1(20\bar{X}^3 - 3\bar{X}),$$
$$a_2 + b_2\bar{Y} + c_2(12\bar{Y}^2 - 1) + d_2(20\bar{Y}^3 - 3\bar{Y}),$$
$$a_3 + b_3\bar{X} + c_3(12\bar{X}^2 - 1) + d_3(20\bar{X}^3 - 3\bar{X}),$$
$$a_4 + b_4\bar{Y} + c_4(12\bar{Y}^2 - 1) + d_4(20\bar{Y}^3 - 3\bar{Y}),$$

we then have the following relations:

$$u_1 = \frac{a_2 + a_4}{2} - 2u_6\Delta x_i, \quad u_2 = \frac{a_2 - a_4}{\Delta x_i}, \quad u_4 = \frac{a_3 + a_1}{2} - 2u_7\Delta y_j,$$
$$u_3 = \frac{b_2 + b_4}{2} - 2\Delta x_i u_{11}, \quad u_7 = \frac{b_4 - b_2}{24\Delta x_i}, \quad u_5 = \frac{b_1 + b_3}{2} + 2\Delta y_j u_{11}, \quad u_6 = \frac{b_1 - b_3}{24\Delta y_j},$$
$$u_8 = \frac{c_2 + c_4}{2}, \quad u_{12} = \frac{c_4 - c_2}{\Delta x_i}, \quad u_9 = \frac{c_1 + c_3}{2}, \quad u_{10} = \frac{c_1 - c_3}{\Delta y_j},$$
$$u_{13} = \frac{d_2 + d_4}{2}, \quad u_{16} = \frac{d_4 - d_2}{16\Delta x_i}, \quad u_{14} = \frac{d_1 + d_3}{2}, \quad u_{15} = \frac{d_1 - d_3}{16\Delta y_j},$$

and also the same constraint as in the P^1 case

$$(a_2 - a_4)\Delta y_j + (a_3 - a_1)\Delta x_i = 0.$$

It seems that we can now define four groups of basis functions related to 'a', 'b', 'c' and 'd', just like in the P^1, P^2 cases. However, if we let all those a_i, b_j, c_k, d_l, $i, j, k, l = 1, \ldots, 4$ be zeros, the following will be left:

$$u_3 = -2\Delta x_i u_{11}, \quad u_5 = 2\Delta y_j u_{11}.$$

That is, we miss another 'bubble' basis function whose support is just a single element, and this basis function does not come from the continuity of the normal component of the function $\mathbf{u}$. This basis function can be written as

$$-2\Delta x_i \begin{pmatrix} \bar{Y} \\ 0 \end{pmatrix} + 2\Delta y_j \begin{pmatrix} 0 \\ \bar{X} \end{pmatrix} + \begin{pmatrix} \Delta x_i(12\bar{X}^2 - 1)\bar{Y} \\ -\Delta y_j\bar{X}(12\bar{Y}^2 - 1) \end{pmatrix} = \begin{pmatrix} \Delta x_i(12\bar{X}^2 - 3)\bar{Y} \\ -\Delta y_j\bar{X}(12\bar{Y}^2 - 3) \end{pmatrix}.$$

The rest is the same as in the P^1, P^2 cases. The dimension of $\widetilde{\mathbf{W}}_h^3$ is $8 \times m \times n + 1$.

As for even larger k in P^k, similar processes can be designed. We would, however, like to remark on two points: when k is getting larger, the number of bubble basis functions will increase rapidly and functions in $\widetilde{\mathbf{W}}_h^k$ will have much richer local structure within each element. Another thing we need pay attention to, especially from the computational point of view, is that when k becomes larger, in order to let 'a' be the only group which might have the global support, we need to figure out the right way to define the degrees of freedom along each edge. In fact, in the rectangular partition, we can simply write the normal component of a function as $\sum_{i=0}^{k} w_i\eta_i$, where $\{\eta_i\}_{i=0}^{k}$ are the orthogonal bases in one-dimensional P^k. Then, $\{w_i\}_{i=0}^{k}$ are the right degrees of freedom we are looking for.

References

[1] F. Assous, P. Degond, E. Heintze, P.A. Raviart, J. Segre, On a finite element method for solving the three dimensional Maxwell equations, Journal of Computational Physics 109 (1993) 222–237.

[2] G.A. Baker, W.N. Jureidini, O.A. Karakashian, Piecewise solenoidal vector fields and the Stokes problem, SIAM Journal on Numerical Analysis 27 (1990) 1466–1485.
[3] D.S. Balsara, D.S. Spicer, A staggered mesh algorithm using high order Godunov fluxes to ensure solenoidal magnetic fields in magnetohydrodynamic simulations, Journal of Computational Physics 149 (1999) 270–292.
[4] J. Bramble, T. Sun, A negative-norm least squares method for Reissner–Mindlin plates, Mathematics of Computation 67 (1998) 901–916.
[5] F. Brezzi, J. Douglas Jr., L.D. Marini, Two families of mixed finite elements for second order elliptic problems, Numerische Mathematik 47 (1985) 217–235.
[6] P. Ciarlet, The Finite Element Methods for Elliptic Problems, North-Holland, Amsterdam, 1975.
[7] B. Cockburn, Discontinuous Galerkin methods for convection-dominated problems, in: T.J. Barth, H. Deconinck (Eds.), High-Order Methods for Computational Physics, Lecture Notes in Computational Science and Engineering, vol. 9, Springer, Berlin, 1999, pp. 69–224.
[8] B. Cockburn, S. Hou, C.-W. Shu, The Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case, Mathematics of Computation 54 (1990) 545–581.
[9] B. Cockburn, G. Karniadakis, C.-W. Shu, The development of discontinuous Galerkin methods, in: B Cockburn, G. Karniadakis, C.-W. Shu (Eds.), Discontinuous Galerkin Methods: Theory, Computation and Applications, Lecture Notes in Computational Science and Engineering, vol. 11, Springer, Berlin, 2000, pp. 3–50, Part I: Overview.
[10] B. Cockburn, M. Luskin, C.-W. Shu, E. Süli, Enhanced accuracy by post-processing for finite element methods for hyperbolic equations, Mathematics of Computation 72 (2003) 577–606.
[11] B. Cockburn, C.-W. Shu, The Runge–Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems, Journal of Computational Physics 141 (1998) 199–224.
[12] B. Cockburn, C.-W. Shu, The local discontinuous Galerkin method for time-dependent convection–diffusion systems, SIAM Journal on Numerical Analysis 35 (1998) 2440–2463.
[13] B. Cockburn, C.-W. Shu, Runge–Kutta Discontinuous Galerkin methods for convection-dominated problems, Journal of Scientific Computing 16 (2001) 173–261.
[14] S. Gottlieb, C.-W. Shu, E. Tadmor, Strong stability preserving high order time discretization methods, SIAM Review 43 (2001) 89–112.
[15] J.S. Hesthaven, T. Warburton, Nodal high-order methods on unstructured grids. I. time-domain solution of Maxwell's equations, Journal of Computational Physics 181 (2002) 186–221.
[16] B.-N. Jiang, J. Wu, L.A. Povinelli, The origin of spurious solutions in computational electromagnetics, Journal of Computational Physics 125 (1996) 104–123.
[17] G.-S. Jiang, C.-W. Shu, On cell entropy inequality for discontinuous Galerkin methods, Mathematics of Computation 62 (1994) 531–538.
[18] O.A. Karakashian, W.N. Jureidini, A nonconforming finite element method for the stationary Navier–Stokes equations, SIAM Journal on Numerical Analysis 35 (1998) 93–120.
[19] C.-D. Munz, P. Omnes, R. Schneider, E. Sonnendrücker, U. Voß, Divergence correction techniques for Maxwell solvers based on a hyperbolic model, Journal of Computational Physics 161 (2000) 484–511.
[20] R.A. Nicolaides, D.-Q. Wang, Convergence analysis of a covolume scheme for Maxwell's equations in three dimensions, Mathematics of Computation 67 (1998) 947–963.
[21] J. Ryan, C.-W. Shu, H. Atkins, Extension of a post-processing technique for the discontinuous Galerkin method for hyperbolic equations with application to an aeroacoustic problem, SIAM Journal on Scientific Computing (submitted).
[22] K.S. Yee, Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, IEEE Transactions on Antenna Propagation AP 14 (1966) 302–307.