INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2010; **64**:1336–1362
Published online 5 August 2010 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/fld.2376

A performance comparison of nodal discontinuous Galerkin methods on triangles and quadrilaterals

D. Wirasaet[1, *, †], S. Tanaka[1], E. J. Kubatko[2], J. J. Westerink[1] and C. Dawson[3]

[1]*Environmental Fluid Dynamics Laboratories, Department of Civil Engineering and Geological Sciences, University of Notre Dame, Notre Dame, IN 46556, U.S.A.*
[2]*Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, Columbus, OH 43210, U.S.A.*
[3]*Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712, U.S.A.*

SUMMARY

This work presents a study on the performance of nodal bases on triangles and on quadrilaterals for discontinuous Galerkin solutions of hyperbolic conservation laws. A nodal basis on triangles and two tensor product nodal bases on quadrilaterals are considered. The quadrilateral element bases are constructed from the Lagrange interpolating polynomials associated with the Legendre–Gauss–Lobatto points and from those associated with the classical Legendre–Gauss points. Settings of interest concern the situation in which a mesh of triangular elements is obtained by dividing each quadrilateral element into two triangular elements or *vice versa*, the mesh of quadrilateral elements is obtained by merging two adjacent triangular elements. To assess performance, we use a linear advecting rotating plume transport problem as a test case. For cases where the order of the basis is low to moderate, the computing time used to reach a given final time for the quadrilateral elements is shorter than that for the triangular elements. The numerical results also show that the quadrilateral elements yield higher computational efficiency in terms of cost to achieve similar accuracy. Copyright © 2010 John Wiley & Sons, Ltd.

Received 16 November 2009; Revised 23 April 2010; Accepted 10 May 2010

KEY WORDS: finite element; discontinuous Galerkin methods; quadrilateral elements; triangular elements; computational cost; transport equation

1. INTRODUCTION

Discontinuous Galerkin (DG) finite element methods are particularly well-suited for the solution of propagation- and convection-dominated problems with or without sharp gradients in the forcing functions and/or flows. DG methods, conceptually similar to finite volume methods, inherently

*Correspondence to: D. Wirasaet, Computational Hydraulics Laboratory, Department of Civil Engineering and Geological Sciences, University of Notre Dame, Notre Dame, IN 46556, U.S.A.
†E-mail: dwirasae@nd.edu

preserve mass perfectly on the elemental level, which make them ideal for coupling flow and transport models. Additional advantages of DG methods are their high order of accuracy and their high scalability for parallel implementation when explicit time integration schemes are employed. As DG methods use discontinuous approximations, they allow the use of an arbitrary order of polynomial approximation in each element and can accommodate non-conformal meshes without a continuity requirement on the trial functions along element boundaries (see [1, 2] for reviews of the DG methods). These features render them ideally suited for an adaptive discretization with h (mesh) and p (order of polynomial) refinements.

One of the many areas in which DG methods have made significant impact is the solution of the Shallow Water Equations (SWE) [3–7], which are used in modeling many advection and propagation flow processes in coastal regions, such as hurricane induced flooding, debris flows, tsunami waves, and many others. Simulations of environmental fluid flows in coastal regions normally involves large, geometrically complicated domains and integration over long periods of time. These situations necessitate an accurate and efficient solution of the SWE. While DG methods have a number of advantages, one disadvantage, in comparison with Continuous Galerkin (CG) methods on a given mesh, is the larger number of degrees of freedom (DOFs), which consequently translate into greater computational costs. The preliminary comparative study in [5] of the CG and DG method for SWE finds that, on serial machines, the cost per time step of the DG approach is four to five times more expensive than that of the CG approach when comparing similar order of interpolation on identical meshes. At the same time, the study in [8] finds that the DG approach is significantly more efficient on an accuracy per DOF basis and on large-scale parallel machines. It is noted, in these studies, that triangular meshes are used.

The motivation behind this work stems from the concept that, for roughly the same element size, a mesh of quadrilateral elements would consist of roughly half as many elements as a mesh of triangular elements (a quadrilateral element may be obtained from merging two adjacent triangular elements). Furthermore, the number of edges in the quadrilateral mesh is roughly two-thirds that of the triangular mesh. We note that, for DG methods, evaluating the edge integral is one of the major computational costs. From these observations, one might expect that the use of quadrilateral elements may improve the computational efficiency of DG schemes. To gain more insight into these ideas, we conduct a study of the performance of DG methods on triangles and quadrilaterals. We consider a solution of hyperbolic conservation laws using the so-called nodal DG methods [9, 10] in this investigation. The nodal DG method utilizes a nodal Lagrange interpolation basis as the approximating basis functions, which provides a simple and generic means to treat a (nonlinear) flux term appearing in the hyperbolic conservation laws. For the quadrilateral element, the nodal bases used are obtained by the tensor product of a one-dimensional (1D) Lagrange basis instead of working with a space of higher dimensional polynomials in a strict sense. The numerical experiments of a two-dimensional (2D) linear transport problem show an optimistic outcome wherein for low to moderate order bases, the computing time used to reach the final time with the quadrilateral elements is shorter than that of the triangular elements. Furthermore, the quadrilateral elements are more computationally efficient in term of the cost to achieve a specific error level.

In this paper, a nodal DG formulation for hyperbolic conservation laws is first summarized. Subsequently, we describe the nodal bases on triangles and on quadrilaterals. For quadrilateral elements, we also discuss the strategy we use to efficiently evaluate the elemental matrices required in the DG scheme. Lastly, we present the results of numerical experiments on a rotating Gaussian plume problem in Section 3, and draw some conclusions of the study in Section 4.

2. METHODOLOGY

2.1. *Discontinuous Galerkin method for Hyperbolic conservation laws*

We consider the numerical solution of a 2D scalar conservation law of the form,

$$\frac{\partial u}{\partial t}+\nabla\cdot\boldsymbol{f}(u,\boldsymbol{x},t)=0,\quad \boldsymbol{x}\in\Omega\in\mathbb{R}^2,\ t\in\mathbb{R}_0^+, \tag{1}$$

where $\boldsymbol{f}=(f_x, f_y)$ is a nonlinear flux. The equation is augmented with appropriate boundary and initial conditions. For the spatial discretization using DG, the solution u is replaced by a discontinuous approximate solution u_h, which, in each element K, belongs to a finite dimensional approximation space $V(K)$. The approximate solution on the element K is then obtained by requiring that,

$$\int_K \frac{\partial u_h}{\partial t} v_h\,\mathrm{d}\boldsymbol{x}-\int_K \boldsymbol{f}(u_h)\cdot\nabla v_h\,\mathrm{d}\boldsymbol{x}+\int_{\partial K}\widehat{\boldsymbol{f}}\cdot\mathbf{n}v_h\,\mathrm{d}s=0, \tag{2}$$

for all $v_h\in V(K)$, where $\mathbf{n}$ represents the outward-pointing unit normal vector. The so-called numerical flux $\widehat{\boldsymbol{f}}$ (also known as the Riemann solver) resolves the flux $\boldsymbol{f}(u_h)$ being multiply defined on the element interface arising from the fact that the approximation is not continuous across the element boundary. The numerical flux, which depends on the traces from both sides of the element interface, is essential for the stability, convergence, and efficiency of the DG method (see examples of different numerical fluxes in [11, 12]).

In this work, we consider the simple Lax–Friedrichs flux for the numerical flux. To define the flux, consider two adjacent elements K^- and K^+ and let e be their common edge, which is not necessarily an entire edge. The local Lax–Friedrichs flux for $\boldsymbol{x}\in e$ is given by

$$\widehat{\boldsymbol{f}}=\frac{\boldsymbol{f}(u_h^-)+\boldsymbol{f}(u_h^+)}{2}+\frac{C}{2}\mathbf{n}^{\mp}(u_h^{\mp}-u_h^{\pm}), \tag{3}$$

where (u_h^-, u_h^+) are the solution value at $\boldsymbol{x}$ of the element K^- and of the element K^+, $\mathbf{n}^-=-\mathbf{n}^+$, respectively, and C corresponds to the largest value, on the edge e, of the local maximum flux Jacobian

$$\max_{u\in[u_h^-,u_h^+]}\left|\mathbf{n}\cdot\frac{\partial \boldsymbol{f}}{\partial u}\right|. \tag{4}$$

It is noted that the boundary conditions are enforced weakly through the numerical flux (by properly specifying an exterior state in the numerical flux at the physical boundaries such that the desirable conditions are achieved in a weak sense).

Here, we use a nodal basis of $\mathscr{P}^p(K)$, a space of polynomials of degree of at most p, to approximate the solution in each element K. Let $\{\phi_i\}_{i=1,\ldots,N_p}$ be a nodal basis associated with the interpolation points $\{\boldsymbol{x}_i\}_{i=1,\ldots,N_p}$, $\boldsymbol{x}_i\in K$. The nodal basis functions possess the so-called interpolation property,

$$\phi_i(\boldsymbol{x}_j)=\delta_{ij}. \tag{5}$$

The details of the nodal basis functions used in this work are discussed in the subsequent sections. The approximate solution in terms of the basis functions is defined by

$$u_h = \sum_{i=1}^{N_p} \tilde{u}_i(t)\phi_i(\boldsymbol{x}), \quad \boldsymbol{x} \in K, \tag{6}$$

where $\tilde{u}_i(t)$ are the time-dependent expansion coefficients. Owing to the interpolation property, the value of $\tilde{u}_i(t)$ corresponds simply to that of the approximate solution at its associated interpolation point, i.e. $\tilde{u}_i(t) = u_h(\boldsymbol{x}_i, t)$. Instead of treating the nonlinear flux term in a conventional manner, the term is approximated by the nodal representation as follows:

$$f_x(u_h, \boldsymbol{x}, t) \approx (I_N f_x)(\boldsymbol{x}) \equiv \sum_{i=1}^{N_p} \tilde{f}_{x,i}(t)\phi_i(\boldsymbol{x}), \tag{7}$$

where the coefficient is defined by $\tilde{f}_{x,i}(t) = f_x(\tilde{u}_i, \boldsymbol{x}_i, t)$ (the nodal representation of the y-directed flux is defined analogously). With (5) and (7) at hand, the statement (2) is equivalent to the following system of ordinary differential equations (ODEs):

$$\sum_{j=1}^{N_p} \left(m_{i,j} \frac{\mathrm{d}\tilde{u}_j}{\mathrm{d}t} - s^x_{i,j} \tilde{f}_{x,j} - s^y_{i,j} \tilde{f}_{y,j} \right) + \int_{\partial K} \widehat{\boldsymbol{f}}_h \cdot \mathbf{n}\phi_i \,\mathrm{d}s = 0, \quad i = 1, \ldots, N_p, \tag{8}$$

where entries of the elemental mass matrix and of the elemental stiffness matrices are given, respectively, by

$$m_{i,j} = \int_K \phi_i \phi_j \,\mathrm{d}\boldsymbol{x} \tag{9}$$

$$s^x_{i,j} = \int_K \frac{\partial \phi_i}{\partial x} \phi_j \,\mathrm{d}\boldsymbol{x}, \quad s^y_{i,j} = \int_K \frac{\partial \phi_i}{\partial y} \phi_j \,\mathrm{d}\boldsymbol{x}. \tag{10}$$

The solution procedure consists of integrating these ODEs for the expansion coefficients and subsequently substituting the resulting coefficients into (6) to obtain the approximate solution. In (8), the expansion coefficients from different elements appear solely in the numerical flux term; therefore, the global mass matrix of the DG method is block diagonal and, consequently, can be inverted in an elementwise fashion. Moreover, the elemental mass matrix, its inverse, and the elemental stiffness matrices can be pre-computed and stored at the initial stage of the simulation. The situation is far simpler for triangular elements and rectangular elements than for curvilinear elements. Since, for these elements, there is a geometric transformation with a constant Jacobian mapping between the physical element and the reference element (an example of a typical geometry of a triangular reference element is $\{(\xi, \eta) | \xi, \eta \geqslant -1$ and $\xi + \eta \leqslant 0\}$ and of a quadrilateral reference element is $\{(\xi, \eta) | (\xi, \eta) \in [-1, 1]^2\}$), the calculation of elemental matrices amounts simply to appropriately scaling the elemental matrices associated with the reference element.

2.2. *Triangular nodal element*

For the triangular elements, we use the nodal basis on a triangle with a set of interpolation points described in [9, 10, 13]. These nodal bases on the reference triangle $I = \{\boldsymbol{\xi} = (\xi, \eta) | \xi, \eta \geqslant -1$ and $\xi + \eta \leqslant 0\}$ are constructed as follows (note that an arbitrary triangular domain K is related to the reference triangle I by an appropriate mapping $\boldsymbol{x}_K(\xi): I \rightarrow K$, such as an affine mapping [14]). Let $\{\psi_i(\xi)\}_{i=1,\ldots,N_p}$ be the Dubiner basis of $\mathscr{P}^p(I)$ [15] (in general, $\{\psi_i(\xi)\}$ can be an arbitrary basis of $\mathscr{P}^p(I)$ [14]). The Dubiner basis is orthonormal over the reference triangle and the basis functions are polynomials. They are the products of Legendre and Jacobi polynomials and thus can be evaluated in an efficient manner. The number of basis functions is

$$N_p = \frac{(p+2)(p+1)}{2} \tag{11}$$

for a given order $p \in \mathbb{N}$. For a given set of interpolation points, $\{\xi_i\}_{i=1,\ldots,N_p}$, the nodal basis functions $\{\phi_i\}_{i=1,\ldots,N_p}$ are constructed by requiring that,

$$f(\xi) = \sum_{i=1}^{N_p} \tilde{f}_i \phi_i(\xi) = \sum_{i=1}^{N_p} \hat{f}_i \psi_i(\xi), \tag{12}$$

and that the nodal functions ϕ_i satisfy the interpolation condition (5), i.e. $\phi_i(\xi_j) = \delta_{ij}$. As a result, the transformations between two representations can be determined by

$$\tilde{f}_i = \sum_{j=1}^{N_p} \mathscr{V}_{i,j} \hat{f}_j, \quad i = 1, \ldots, N_p, \tag{13}$$

$$\psi_i(\xi) = \sum_{j=1}^{N_p} \mathscr{V}_{j,i} \phi_j(\xi), \quad i = 1, \ldots, N_p, \tag{14}$$

where the generalized Vandermonde matrix $\mathscr{V}$ is defined by

$$\mathscr{V} = (\mathscr{V}_{i,j}), \quad \mathscr{V}_{i,j} = \psi_j(\xi_i). \tag{15}$$

The nodal basis function can, therefore, be defined as a combination of the Dubiner basis functions as follows, given that $\mathscr{V}$ is invertible,

$$\phi_i(\xi) = \sum_{j=1}^{N_p} (\mathscr{V}^{-1})_{j,i} \psi_j(\xi), \quad i = 1, \ldots, N_p. \tag{16}$$

In practice, the explicit form of the nodal basis functions is rarely needed. Interpolated values at desired points (other than that of the interpolation points) are obtained by first calculating the modal coefficients $\{\hat{f}_i\}$ from the nodal coefficients $\{\tilde{f}_i\}$ by means of inverting (13) and subsequently computing the interpolated values through the modal representation.

The generalized Vandermonde matrix that is well-conditioned leads to a well-behaved interpolation. This can be obtained by the use of a set of interpolation points with certain distributions. Here, we use the set of interpolation points, described by Hesthaven [10, 13], which yields this desirable effect. Figure 1 shows, as examples, the set of interpolation points for $p = 2$, 4, 6, and 8 on I. The interpolation points have Legendre–Gauss–Lobatto (LGL) node sets along the edges and dense interpolation points adjacent to the boundary. The so-called Fekete points [14] are another example that produces a well-conditioned generalized Vandermonde matrix.

2.3. Quadrilateral nodal element

For quadrilateral elements, instead of working with $\mathscr{P}^p(Q)$, the approximation space on the reference square $Q = [-1, 1]^2$ is selected as $\mathscr{P}^p([-1, 1]) \times \mathscr{P}^p([-1, 1])$, the tensor products of

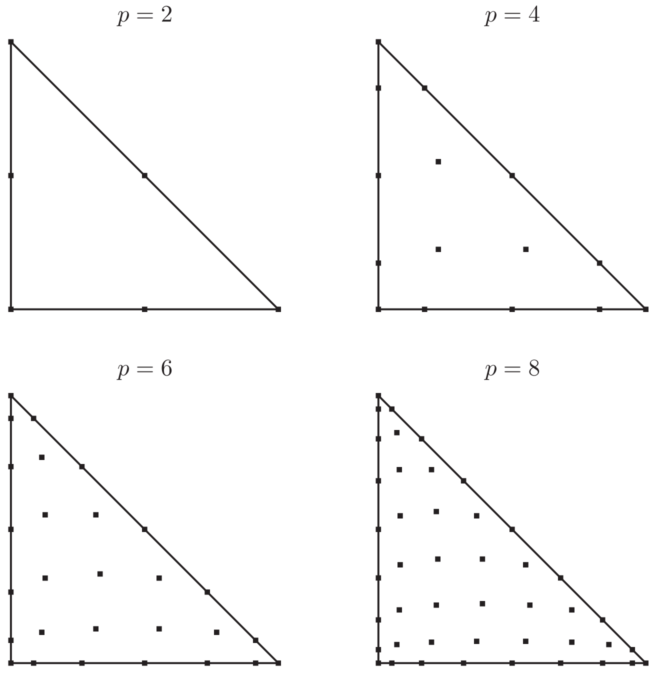


Figure 1. Distribution of interpolation points for the Lagrange polynomials of degree $p=2$, 4, and 8 on the reference triangle.

$\mathscr{P}^p([-1,1])$, a space of 1D polynomials of degree of at most p. Let $\{P_i(x)\}_{i=0,\ldots,p}$ be the normalized Legendre polynomials on $[-1,1]$, a 2D orthonormal basis on Q is defined by

$$\psi_{(p+1)j+i+1}(\boldsymbol{\xi}) \equiv P_i(\xi)P_j(\eta), \quad 0 \leqslant i, j \leqslant p. \tag{17}$$

The number of basis functions in this case is

$$N_p = (p+1)^2. \tag{18}$$

A nodal basis $\{\phi_i(\boldsymbol{\xi})\}_{i=1,\ldots,N_p}$ is then constructed in an identical way described previously for a given set of interpolation points $\{\boldsymbol{\xi}_i\}_{i=1,\ldots,N_p}$. Here, we consider the following two different sets of interpolation points with (i) an LGL distribution and (ii) a Legendre–Gauss (LG) distribution. They can be described generically as

$$\boldsymbol{\xi}_{(p+1)j+i+1} = (x_i, x_j), \quad 0 \leqslant i, j \leqslant p,$$

where $\{x_i\}_{i=0,\ldots,p}$ are the zeros of $(1-x^2)\,\mathrm{d}(P_{p-1}(x))/\,\mathrm{d}x$ for the LGL nodal set and the classical Gauss points, which are the zeros of the Legendre polynomial $P_p(x)$, for the LG nodal set. Figure 2 depicts the nodal sets for $p=4$. It is noted that all points of the LG nodal set reside in the interior part of Q. The nodal basis functions associated with the LG nodal set also form an orthogonal basis. This can be verified by using the fact that the 2D Gauss quadrature,

$$q_{(p+1)i+j+1} \equiv w_i w_j, \quad 0 \leqslant i, j \leqslant p, \tag{19}$$

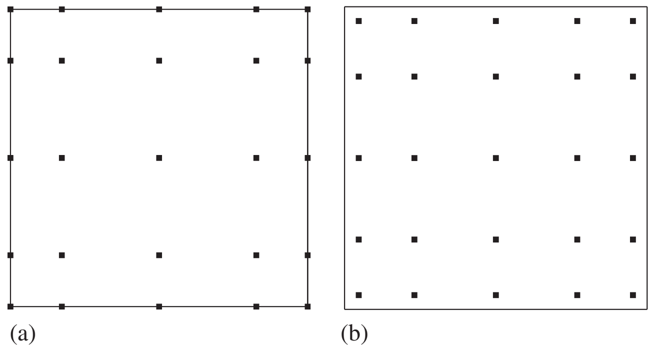


Figure 2. Distribution of interpolation points for the Lagrange polynomials of $p=4$ on the reference square: (a) Legendre–Gauss–Lobatto points and (b) Legendre–Gauss points.

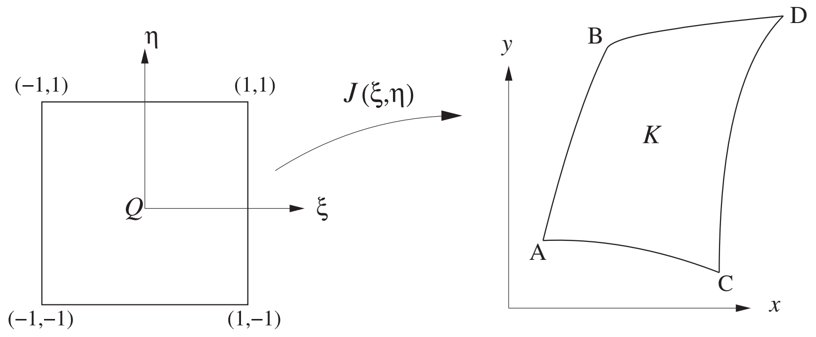


Figure 3. Mapping of a curvilinear element.

which is a tensor product of the 1D classical Gauss weights $\{w_i\}_{i=0,\ldots,p}$ of the Gauss points, integrates exactly polynomials in $\mathscr{P}^{2p+1}([-1,1])\times\mathscr{P}^{2p+1}([-1,1])$. Since the LG nodal set is identical to the 2D Gauss points, the following orthogonal relation holds

$$\int_Q \phi_i\phi_j\,\mathrm{d}\boldsymbol{\xi}=\sum_{n=1}^{N_p} q_n\phi_i(\boldsymbol{\xi}_n)\phi_j(\boldsymbol{\xi}_n)=q_n\delta_{in}\delta_{jn}=q_i\delta_{ij}, \tag{20}$$

for the nodal basis functions defined on this type of nodal set. Note that the nodal basis functions defined on the LGL nodal set do not form an orthogonal basis.

The connection between Q and a possibly curve-edged quadrilateral domain K can be established through an appropriate mapping $\boldsymbol{x}_K(\boldsymbol{\xi}):Q\to K$ (see Figure 3), such as a transfinite mapping [16] or an isoparametric mapping [17]. Assume that such a mapping and its inverse exist. For notational simplicity, the mapping is denoted by $x=x(\xi,\eta)$ and $y=y(\xi,\eta)$. The elemental mass matrix is equivalent to

$$\int_K \phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}=\int_Q J(\xi,\eta)\phi_i(\boldsymbol{\xi})\phi_j(\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi}, \tag{21}$$

where $J(\xi,\eta)=|\partial(x,y)/\partial(\xi,\eta)|$ is the Jacobian of the geometry transformation. Except for rectangular elements K (with a bilinear interpolation for $\boldsymbol{x}_K(\boldsymbol{\xi})$), the value of the Jacobian $J(\xi,\eta)$ is not constant.

In general, to compute such an integral accurately, the Gauss quadrature of a certain order, depending on the form of $J(\xi,\eta)$, is employed. Here, we opt for a less accurate approach; however, the elemental matrices can be calculated more efficiently. We first describe this approach for the nodal basis with the LG nodal set. In this case, we approximate the integral using the LG nodal set and its weight (19) as a quadrature, i.e.

$$\int_K \phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \approx m^a_{i,j} \equiv \sum_{n=1}^{N_p} J(\xi,\eta)|_{\xi_n} q_n \phi_i(\boldsymbol{\xi}_n)\phi_j(\boldsymbol{\xi}_n) = J(\xi,\eta)|_{\xi_i} q_i \delta_{ij}. \tag{22}$$

This approach yields a diagonal lumped mass matrix. The stiffness matrices are computed with the following formula:

$$\int_K \frac{\partial\phi_i(\boldsymbol{x})}{\partial x}\phi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \approx s^{a,x}_{i,j} \equiv \sum_{n=1}^{N_p} D^{\xi}_{n,i} \left.\frac{\partial\xi}{\partial x}\right|_{\xi_n} m^a_{n,j} + \sum_{n=1}^{N_p} D^{\eta}_{n,i} \left.\frac{\partial\eta}{\partial x}\right|_{\xi_n} m^a_{n,j}, \tag{23}$$

where $D^{\xi}_{n,i}=(\partial\phi_i/\partial\xi)|_{\xi_n}$, $D^{\eta}_{n,i}=(\partial\phi_i/\partial\eta)|_{\xi_n}$ represent the derivative matrices and $\partial\xi/\partial x|_{\xi_n}$, $\partial\eta/\partial x|_{\xi_n}$ are the partial derivatives of the inverse mapping $\boldsymbol{x}_K^{-1}$, which are typically obtained from,

$$\begin{bmatrix} \dfrac{\partial\xi}{\partial x} & \dfrac{\partial\xi}{\partial y} \\ \dfrac{\partial\eta}{\partial x} & \dfrac{\partial\eta}{\partial y} \end{bmatrix} = \frac{1}{J(\xi,\eta)} \begin{bmatrix} \dfrac{\partial y}{\partial\eta} & -\dfrac{\partial x}{\partial\eta} \\ -\dfrac{\partial y}{\partial\xi} & \dfrac{\partial x}{\partial\xi} \end{bmatrix}, \tag{24}$$

given that the Jacobian is non-zero. Note that a formula for the approximation of $s^{y}_{i,j}$ corresponds to the result of replacing x in (23) by y. In this approach, the derivative matrices of the reference element can be pre-computed beforehand once and for all (since there are only two derivative matrices of dimension $N_p \times N_p$, this would require a small amount of memory for storage). It is worth noting that for straight-edged quadrilateral elements, Equation (22) yields the exact mass matrix, provided that $\boldsymbol{x}_K(\boldsymbol{\xi})$ is a bi-linear interpolation of the four vertices. Furthermore, with the bi-linear interpolation for $\boldsymbol{x}_K(\boldsymbol{\xi})$, Equations (22) and (23) are exact for rectangular elements.

For the nodal basis with the LGL nodal set, the natural quadrature weights associated with this nodal set are that of the tensor products of the 1D Gauss–Lobatto weights [18]. The use of the 2D Gauss–Lobatto quadrature leads to the approximate elemental matrices having forms that are identical to (22) and (23) (where $\{\boldsymbol{\xi}_i\}$ and $\{q_i\}$ are understood as the LGL nodal set and the 2D Gauss–Lobatto weights, respectively). However, the 2D Gauss–Lobatto quadrature, which integrates exactly polynomials in $\mathscr{P}^{2p-1}([-1,1])\times\mathscr{P}^{2p-1}([-1,1]))$, is less accurate than the 2D Gauss quadrature. To obtain more accurate elemental matrices, we employ instead the 2D Gauss quadrature (i.e. the LG nodal set and its associated weight) for the integration. This strategy yields the following approximate mass matrix,

$$\int_K \phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \approx m^a_{i,j} \equiv \sum_{n=1}^{N_p} \tilde{\mathscr{V}}_{n,i} \sum_{l=1}^{N_p} m^{a,\mathrm{LG}}_{n,l} \tilde{\mathscr{V}}_{l,j}, \tag{25}$$

and the following stiffness matrix

$$\int_K \frac{\phi_i(\boldsymbol{x})}{\partial x}\phi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \approx s^{a,x}_{i,j} \equiv \sum_{n=1}^{N_p} \tilde{\mathscr{V}}_{n,i} \sum_{l=1}^{N_p} s^{a,x,\mathrm{LG}}_{n,l} \tilde{\mathscr{V}}_{l,j}. \tag{26}$$

In the above equations, $m^{a,\mathrm{LG}}_{i,j}$ and $s^{a,x,\mathrm{LG}}_{i,j}$ are the approximate mass matrix and the approximate stiffness matrix associated with the LG nodal basis discussed above. The matrix $\tilde{\mathscr{V}}_{i,j}$, defined by,

$$\widetilde{\mathscr{V}}_{i,j} = \sum_{l=1}^{N_p} (\mathscr{V}_{\mathrm{LG}})_{i,l} (\mathscr{V}^{-1}_{\mathrm{LGL}})_{l,j}, \tag{27}$$

interpolates the function values on the LG nodal set from the function values on the LGL nodal set. Here, $\mathscr{V}_{\mathrm{LGL}}$ and $\mathscr{V}_{\mathrm{LG}}$ are the generalized Vandermonde matrices associated with the LGL nodal set and the LG nodal set, respectively. Note that the matrix $\widetilde{\mathscr{V}}_{i,j}$ and its inverse can be pre-computed and stored at the initial phase of the simulation.

In practice, especially in explicit time integration, it is the result of the matrix–vector multiplication of the elemental matrices and a given vector that is required. The matrix–vector multiplication of these matrices with a given vector can be determined by a succession of matrix–vector multiplications described by the formulae given above (which involve the matrices defined on the reference element and the diagonal matrices involved with the coordinate transformation). Therefore, it is not necessary to explicitly form the elemental matrices, thus reducing the storage required.

3. NUMERICAL EXPERIMENTS

We use a linear transport problem in order to assess the performance of the nodal DG method on triangular elements and on quadrilateral elements. Such a problem is governed by (1) with the following flux term:

$$\boldsymbol{f} = \boldsymbol{a}(\boldsymbol{x})u = (a_1(\boldsymbol{x})u, a_2(\boldsymbol{x})u). \tag{28}$$

A test case involves the advection of a scalar field of a plume driven by a rotating velocity field in a square computational domain. More specifically, the advection velocity considered is given by

$$(a_1, a_2) = (-\omega y, \omega x), \tag{29}$$

where ω is a constant angular velocity. The initial condition of the scalar field is the Gaussian plume centered at $\boldsymbol{x}_c$,

$$u(\boldsymbol{x}, t=0) = \exp(-\sigma|\boldsymbol{x}-\boldsymbol{x}_c|^2), \quad \sigma > 0. \tag{30}$$

The computational domain is square $[-L, L]^2$, where L is sufficiently large.

The numerical experiments are undertaken with the following parameters: $\omega = 5\pi/6$, $\boldsymbol{x}_c = (0, 3/5)$, $\sigma = 125\times 1000/33^2$, and $L=1$. A computer code employed in the numerical computations is mainly written in Fortran 90 and compiled using an Intel® Fortran compiler version 11. The fortran Basic Linear Algebra Subprograms (BLAS) level 2 [19] is used for matrix–vector multiplications on the elemental level. All computations are conducted on a workstation with two dual-core AMD Opteron® 2220 processors, 4 GB RAM, and a 32-bit Linux operating system (note that the operating system is able to use up to 3 GB of RAM). Computing times reported are an average of two identical simulations.

Here, we assess the performance of the DG discretization using the nodal bases on different element types described in the previous section, i.e. the nodal basis on triangles, the nodal basis of the LGL and of the LG nodal set on rectangles and on skewed rectangles (which refer to convex quadrilaterals). To fulfill this purpose, we consider the nodal bases of order ranging from $p=1$ up to $p=8$ and consider three mesh configurations. The first configuration consists of triangular elements which are devised by bi-secting square elements. The next configuration is a mesh of square elements. For the last configuration, a mesh of skewed-rectangular elements is employed. In each configuration, the problem is solved on three different resolutions, namely, from coarsest to finest, h, $h/2$, and $h/3$. Figure 4 illustrates the coarsest mesh of each configuration. The meshes of different resolutions for the first and second configurations are built from the uniform grid of $(M+1)\times(M+1)$ with $M=20$, 40, and 60, respectively. For the third configuration, the coarsest mesh (of resolution h) is built by slightly relocating the interior vertices of the uniform grid of 21×21. The relocation is carried out in a random fashion and the distances shifted vary from 0 to 0.02 in each direction. The two finer meshes are then obtained by dividing each skewed rectangle into 2×2 and 3×3 skewed rectangles. Note that the meshes of resolution h, $h/2$, and $h/3$ have element sizes (approximately) $2/20$, $2/40$, and $2/60$, respectively. The coarsest quadrilateral meshes consist of 20^2 elements. The number of elements of the two finer meshes are four and nine times that of the coarsest mesh. For the triangular meshes, the numbers of elements are twofold that of the quadrilateral meshes of the same (so-called) resolution. Note that the total number of nodal points, denoted by N_t, corresponds to the summation of the number of nodal points of each element.

The time-dependent system of ODEs arising from the DG spatial discretization is numerically integrated by an explicit fourth–fifth order Runge–Kutta–Fehlberg (RKF45) method (see a detailed account in [20]). RKF45 has a mechanism to automatically select the step size Δt used in the integration to control the accuracy of the solution. The integrator utilizes the fourth-order scheme and the fifth-order scheme that uses all values of the sub-steps of the fourth-order scheme. The difference of approximate solutions from the two schemes gives an estimate of the errors, which can then be used to adjust the step size. In this work, we use the RKF45 subroutine written by Shampine *et al.* [21]. In this subroutine, the accuracy of the solution is controlled by the parameter named `relerr` and `abserr` (`relerr > abserr`), denoted here as ε_r and ε_a. The values of these parameters are set to sufficiently small values in order to keep an error from the temporal discretization small when compared with the spatial error. For the h-meshes, we use $(\varepsilon_r, \varepsilon_a)=(10^{-5}, 10^{-8})$ for all $p=1$–8. These values are also employed in the test cases with the $h/2$- and $h/3$-meshes except in those where the order of basis p is high. In these cases, smaller values of $(\varepsilon_r, \varepsilon_a)$ are used to keep the temporal error sufficiently small. More specifically, the value of these parameters are set to $(10^{-7}, 10^{-11})$ for the test cases with $p=8$ for the $h/2$-meshes and with $p=7$ and 8 for the $h/3$-meshes.

3.1. Accuracy

Figure 5 shows a snapshot at various times of the approximate solution computed on the coarsest triangular mesh using the DG method with $p=1$. The approximate solution computed on the coarsest skewed-rectangular mesh using the DG method with the LGL nodal set of $p=1$ is shown in Figure 6. Note the discontinuities of the approximate solution at the element edges. It can be observed in these figures that the approximate solution exhibits a significant decay in height

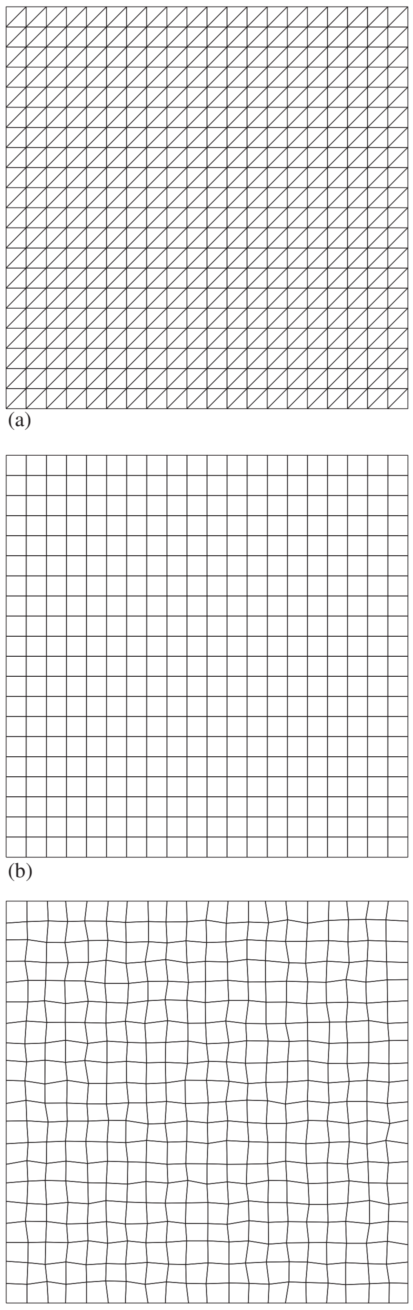

(a)
(b)
(c)

Figure 4. Three configurations of computational meshes of resolution 20×20: (a) triangular elements; (b) square elements; and (c) skewed-rectangular elements.

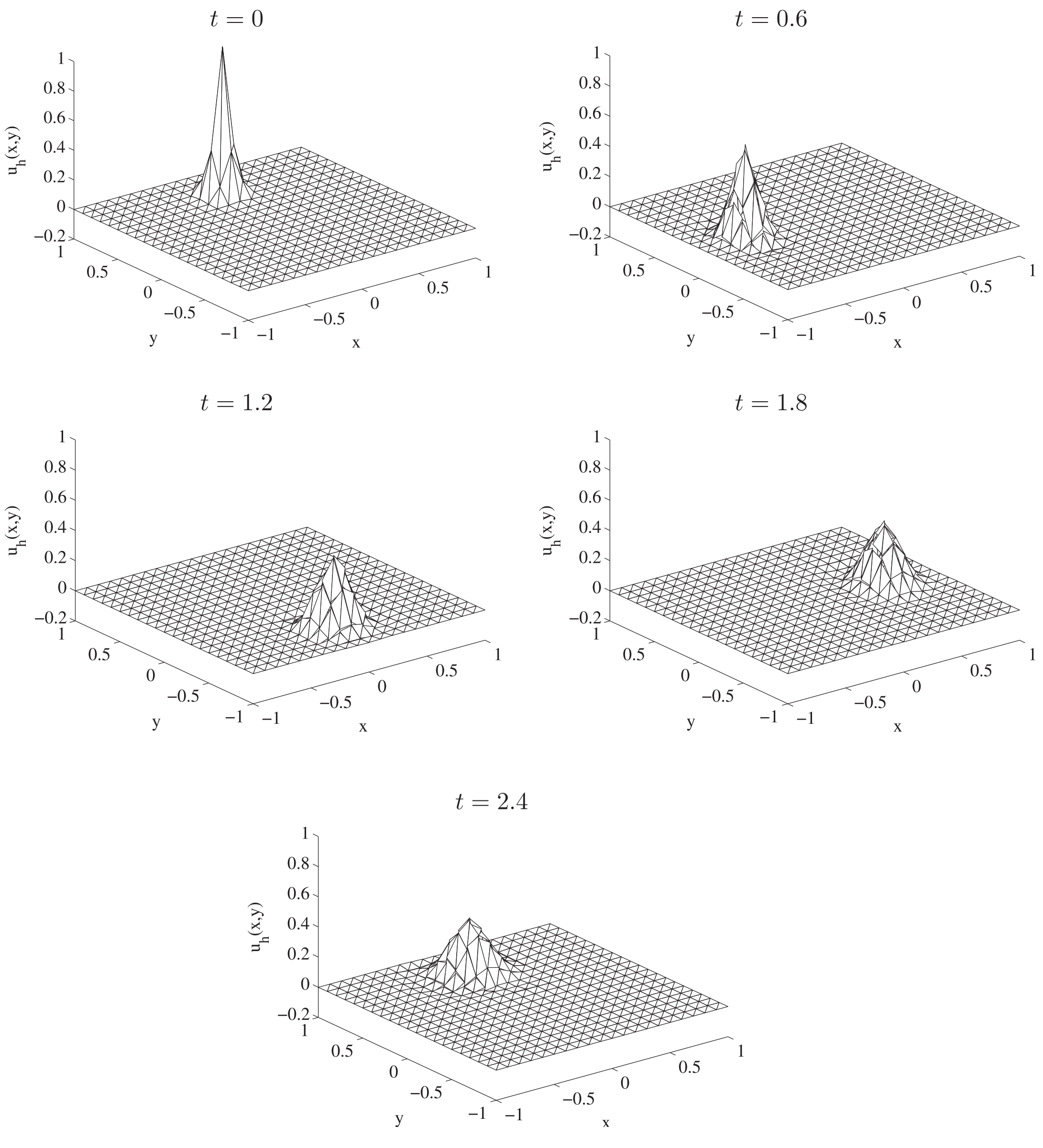


Figure 5. Numerical solution at various times obtained with Lax–Friedrichs nodal DG method on triangles with $p=1$ and a mesh of 2×20^2 elements of size $h=2/20$.

(as an example, in Figure 6, the height of the approximate plume at $t=2.4$, the time at which the exact plume makes one complete loop, reduces approximately 70%) and a broadening of the plume. This indicates the numerically diffusive nature of the solution for the selected size of the elements and p. The effect of numerical diffusion reduces as computational resolution increases. The resolution can be increased by increasing the order of the basis functions p or decreasing the

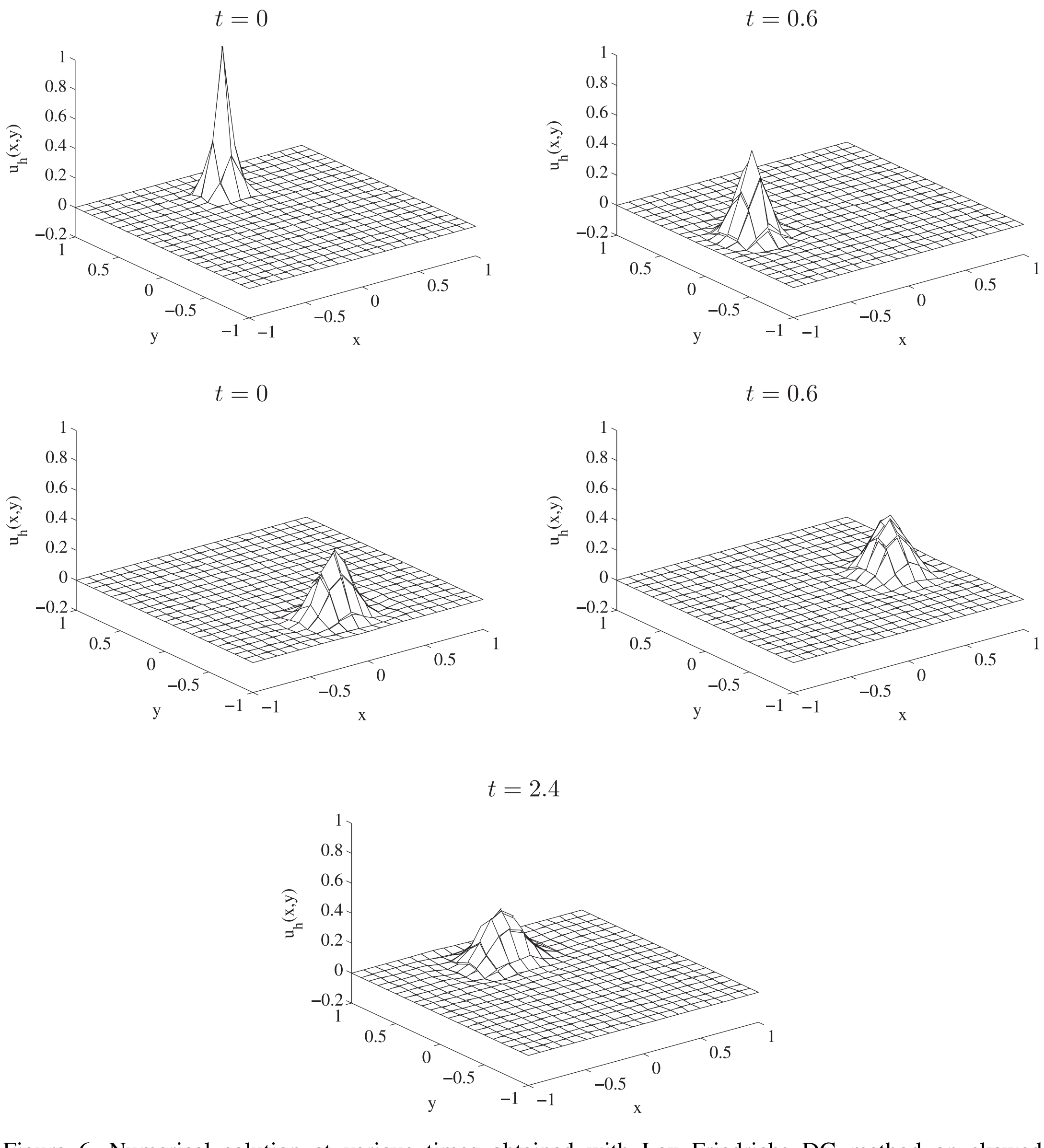
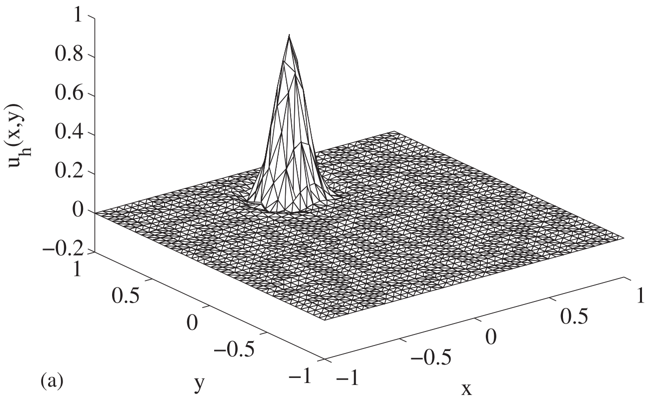


Figure 6. Numerical solution at various times obtained with Lax–Friedrichs DG method on skewed rectangles with the LGL nodal set of $p=1$ and a mesh of 20^2 elements of size $h=2/20$.

size of the elements. Figures 7 and 8 show the approximate solutions at $t=2.4$ obtained using the LGL nodal set on the skewed-rectangular elements when a higher resolution than that used in Figure 6 is employed. Figures 7(a), (b) show the approximate solutions using $p=2$ and $p=3$ when the skewed-rectangular mesh of h-resolution is used (the triangles shown in this figure are used only for plotting purposes so that the solution at the interior points of the elements can be



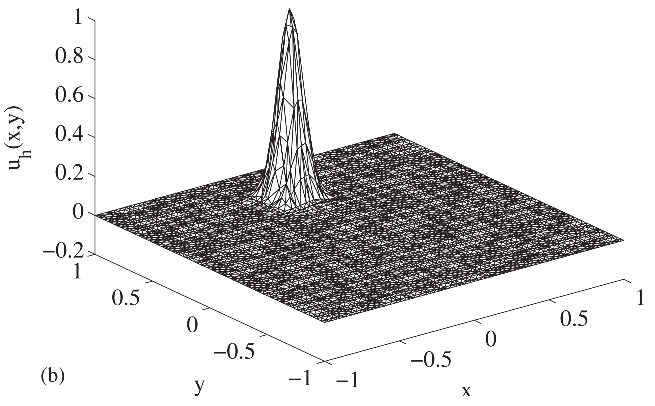


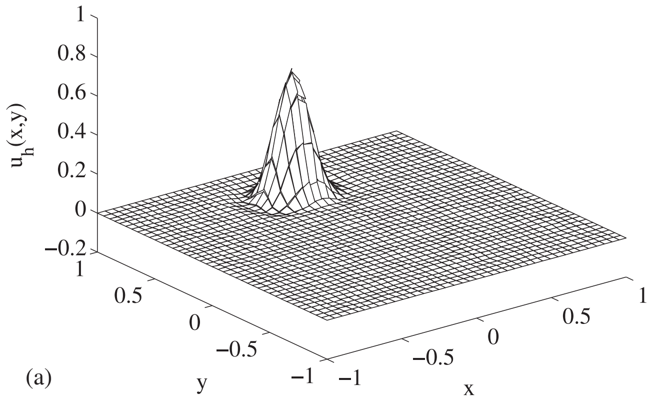
Figure 7. Numerical solution at $t=2.4$ obtained from Lax–Friedrichs DG method using the nodal basis with LGL nodal set on skewed-rectangular elements: (a) $p=2$, $h=2/20$, $N_t=3600$ and (b) $p=3$, $h=2/20$, $N_t=6400$.

visualized). The solution depicted in Figures 8(a), (b) is solved on the skewed-rectangular mesh of resolution h and $h/2$ while the order of basis functions p is held fixed at unity. Note the considerable reduction of the effect of the numerical diffusion when compared with the approximate solution shown in Figure 6, which is computed using the lower resolution setting (note that, for example, the height of the approximate plume plotted in Figure 7(b) reduces approximately 4%).

In Figure 9, we plot the errors in the approximate solutions at $t=0.8$ as a result of varying the orders of the nodal basis p (where the computational mesh is held unchanged). Figures 9(a)–(c) report the results when the problem is solved on the meshes of resolution h, $h/2$, and $h/3$, respectively. Here, the maximum norm over the nodal points,

$$\|g\|_{\mathcal{N},\infty} = \max_{i\in\{1,\ldots,N_{\mathrm{el}}\}} \left(\max_{\boldsymbol{x}\in \boldsymbol{X}_{K_i}} |g(\boldsymbol{x})| \right) \tag{31}$$

where N_{el} denotes the number of elements in a given mesh, $\boldsymbol{X}_{K_i}$ the set of nodal points of the element K_i, is used in measuring the error. The results clearly indicate the exponential convergence



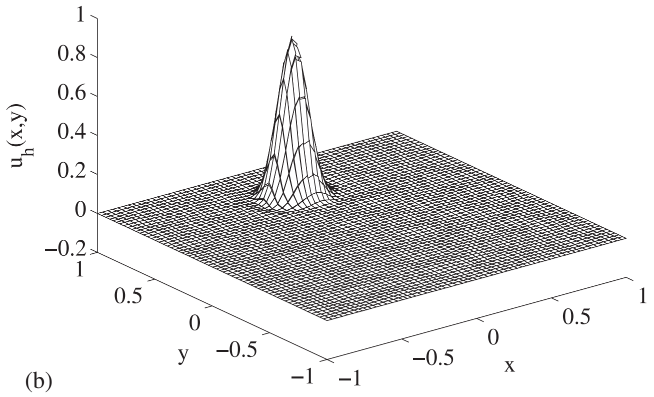


Figure 8. Numerical solution at $t = 2.4$ obtained from Lax–Friedrichs DG method using the nodal basis with LGL nodal set on skewed-rectangular elements: (a) $p = 1$, $h = 2/40$, $N_t = 6400$ and (b) $p = 1$, $h = 2/60$, $N_t = 14400$.

of the DG methods with the nodal basis on triangles and the nodal basis on quadrilaterals with the LGL and LG nodal set. It can be observed from this figure that, for the same order of p and the mesh of the same resolution, the nodal basis on quadrilaterals (both squares and skewed rectangles) yields, for most cases, more accurate solutions than the basis on triangles. For the quadrilateral elements, the approximate solutions computed using the nodal basis with the LG nodal set, for most cases, are more accurate than that of using the LGL nodal set. Furthermore, it can be noticed that, for the same type of nodal set, the curves representing the errors in the approximate solution from employing the skewed-rectangular elements and from employing the square elements are almost indistinguishable. This implies that the use of (mildly) skewed-rectangular elements does not deteriorate the accuracy of the approximate solution. As the exact solution of this particular problem is smooth, an increase in resolution by raising the order of basis p dramatically improves the accuracy of the approximate solution at the expense of a small increase in the number of DOFs (which corresponds to the number of nodal points N_t). Note that raising the order of the basis p by one order increases the DOFs for triangle elements by $1+2/(p+1)$ times and for quadrilateral elements $1+2/(p+1)+1/(p+1)^2$ while the accuracy improves at an

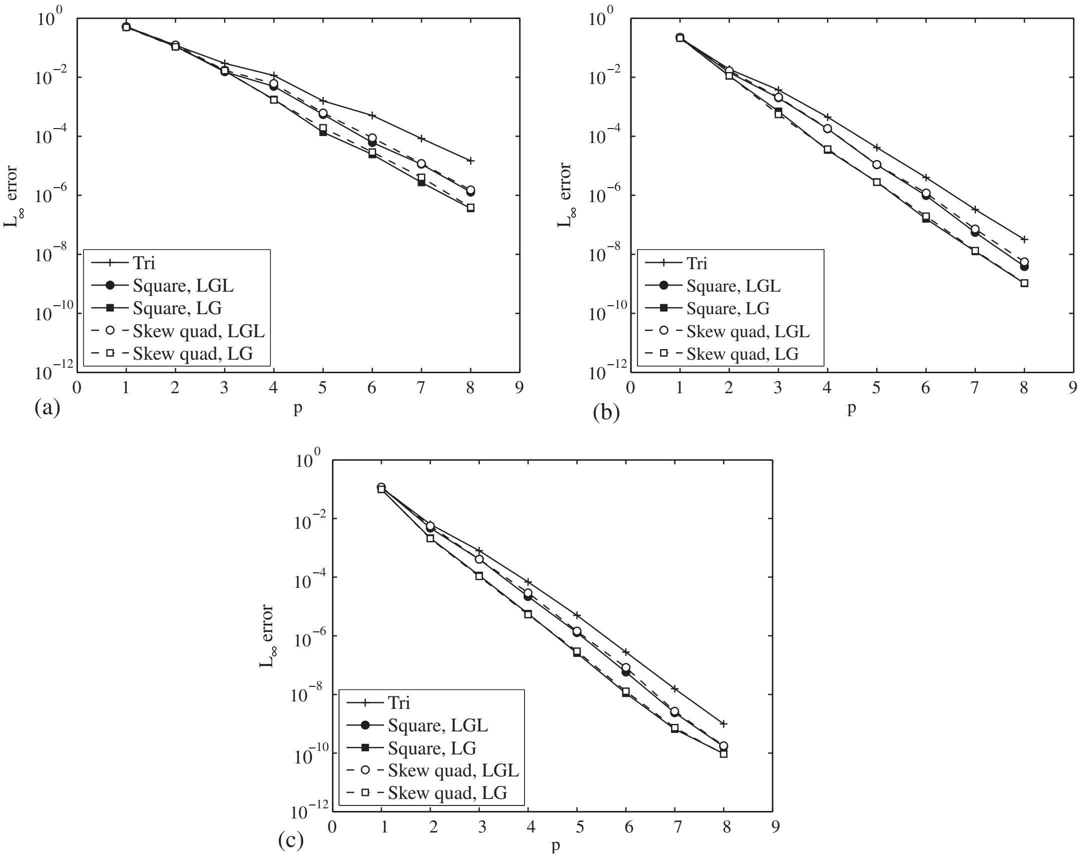


Figure 9. $\|u-u_h\|_{\mathcal{N},\infty}$ at $t=0.8$ as function of orders of nodal bases p. The approximate solutions are computed on the meshes of resolution h, $h/2$, and $h/3$: (a) h-resolution; (b) $h/2$-resolution; and (c) $h/3$-resolution.

exponential-like rate. Figure 10 plots, on a log–log scale, the errors in the approximate solutions against the element sizes (in these plots $\sqrt{N_t}$ is used as a measure of element size). Figures 10(a)–(c) show the results from the DG solution on triangles, on squares with the LGL nodal set, and on squares with the LG nodal set, respectively. Since the errors in the methods with the skewed rectangles differ just slightly from that with squares, we exclude them from the figure. It can be seen that, for all order p, the error as a function of element size $h \simeq \sqrt{N_t}$ appears as a straight line in the log–log plot. This indicates that the error in the approximate solution $\|u-u_h\|_{\mathcal{N},\infty}$ behaves like $O(h^S)$, where S, the rate of convergence, is the slope of the straight line. Note that the value of the slope is higher when the higher order p is used. The rate of convergence for each order p is typically close to $p+1$, regardless of the element type being used. In other words, for this linear problem, the numerical solution from the DG solutions described converges like $O(h^{p+1})$. This optimal behavior on the convergence of the numerical solution can be expected according to theoretical estimates (see [22–24]). Note that for a general nonlinear flux $\boldsymbol{f}$, the estimated error of $O(h^{p+1/2})$ is expected for a DG method with the use of the Lax–Friedrichs flux [24].

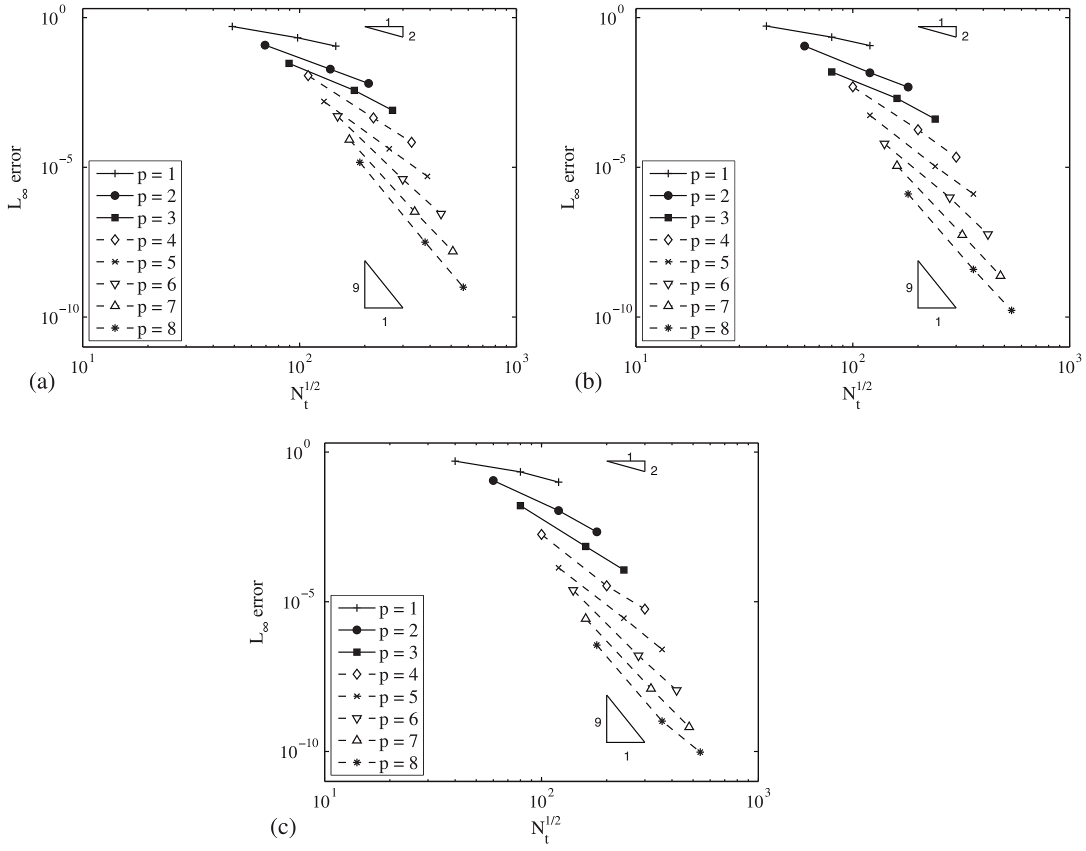
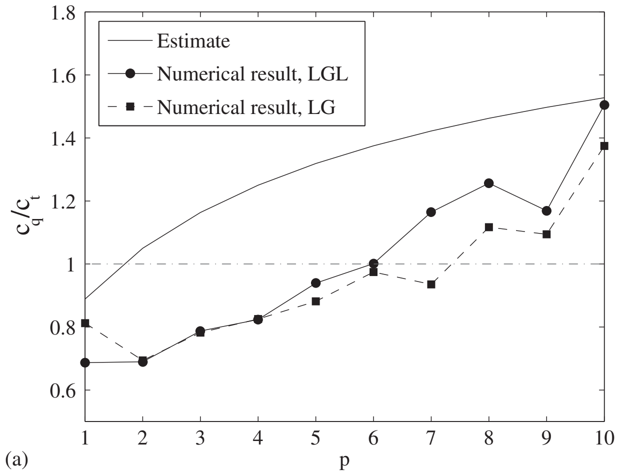


Figure 10. $\|u-u_h\|_{\mathcal{N},\infty}$ as a function of $h \simeq \sqrt{N_t}$ for different orders of p: (a) triangular elements; (b) square elements with the LGL nodal set; and (c) square elements with the LG nodal set.

3.2. *Computational efficiency*

We first make a comparison of the computational cost to evaluate the right-hand side of (8) of the nodal basis on triangles and the nodal basis on quadrilaterals. Let us first discuss a crude comparison estimate. For this estimate, it is assumed that the majority of computing operations are related to the evaluations of the area integrals and the edge integrals. In addition, these integrals are computed from the matrix–vector multiplications of the pre-computed matrices and given vector (note that the implementation of our code follows this line of concept) and the cost of matrix–vector multiplication of the $M\times N$ matrix with an $N\times 1$ vector is of $O(MN)$. Furthermore, the calculation is carried out in an element-wise fashion. We consider a scenario in which the number of elements in a triangular mesh is twice that of a quadrilateral mesh. Let us first outline an estimated cost for triangular elements. In this case, the cost of the area integrals for one element for a given p is proportional to

$$\sim N_p \times N_p = \frac{(p+1)^2(p+2)^2}{4}. \tag{32}$$



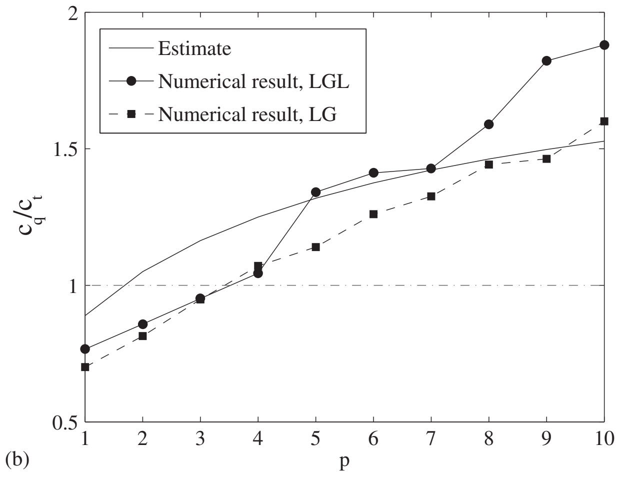


Figure 11. The ratio of c_q to c_t as a function of order p when the problem is solved on the meshes of resolution h. c_q and c_t denotes the computing time used to evaluate one right-hand side term of the discretization with quadrilateral elements and with triangle elements, respectively: (a) with compiler optimization and (b) without compiler optimization.

Provided that an edge integral is done via a $p+1$ quadrature rule, the operations required on a single edge is proportional to $N_p \times (p+1)$ and thus for one element, the cost of the edge integrals is proportional to

$$\sim 3 \times N_p \times (p+1) = \frac{3(p+1)^2(p+2)}{2}. \tag{33}$$

By assuming that the values of constants associated with (32) and with (33) are roughly the same, the cost of evaluating the right-hand side for a triangular mesh consisting of N_{el} elements, denoted

Table I. Computing time (in s) used in the evaluation of the right-hand side of DG discretizations on the meshes of resolution h.

	Optimized code*			Non-optimized code		
p	Tri.[†]	Sq., LGL[‡]	Sq., LG[§]	Tri.	Sq., LGL	Sq., LG
1	0.0108	0.0074	0.0088	0.0179	0.0137	0.0125
2	0.0150	0.0104	0.0104	0.0263	0.0225	0.0214
3	0.0162	0.0128	0.0127	0.0386	0.0367	0.0366
4	0.0208	0.0171	0.0171	0.0564	0.0589	0.0604
5	0.0255	0.0239	0.0225	0.0761	0.1021	0.0868
6	0.0333	0.0333	0.0324	0.0999	0.1410	0.1259
7	0.0449	0.0523	0.0420	0.1467	0.2094	0.1945
8	0.0542	0.0681	0.0606	0.1920	0.3053	0.2768
9	0.0745	0.0871	0.0815	0.2577	0.4694	0.3770
10	0.0846	0.1273	0.1163	0.3210	0.6035	0.5137

*A compiler optimization turned on.

[†]Triangular elements.

[‡]Square elements with the LGL points.

[§]Square elements with the LG points.

as c_t, is roughly proportional to

$$c_t \sim N_{\mathrm{el}} \frac{(p+1)^2(p+2)}{2}\left(\frac{p+2}{2}+3\right). \tag{34}$$

By following the same argument as above, the cost of evaluating the right-hand side of a quadrilateral mesh with $N_{\mathrm{el}}/2$ elements, denoted as c_q, is proportional to

$$c_q \sim \frac{N_{\mathrm{el}}}{2}(p+1)^3(p+5). \tag{35}$$

By comparing (34) and (35), it is found from this estimate that $c_q > c_t$ for $p > 1$ (that is the use of rectangular elements is more costly than the use of triangular elements when the order p is greater than one). It must be emphasized that this estimate is crude and the actual algorithm is in fact more complicated than the setting used in the estimate. The ratio of c_q and c_t from the numerical experiments (along with our estimate) on the meshes of resolution h with various p is plotted in Figure 11. Here, the values of c_q and c_t from the experiments correspond to a ratio of the total time spent on a routine calculating the right-hand side term divided by the total number of times that this routine being called. Note that c_q reported are the results from the calculations using the mesh with square elements. Figure 11(a) shows the results with the compiler optimization turned on and Figure 11(b) shows the results with the compiler optimization turned off. The numeric values of the computing time used in the evaluation of one right-hand side vector are reported in Table I. Although not shown here, the results on the meshes of higher resolution show a similar trend. From Figure 11(b), it can be observed that the cost for the quadrilateral elements is greater than that for the triangular elements when $p \geqslant 6$. With the compiler optimization being omitted, the point at which the quadrilateral elements are more costly starts at $p > 3$. Regardless of the compiler optimization being on or off, the numerical results show that the order p, at which the calculation of the right-hand side associated with the quadrilateral elements is more expensive than that of the

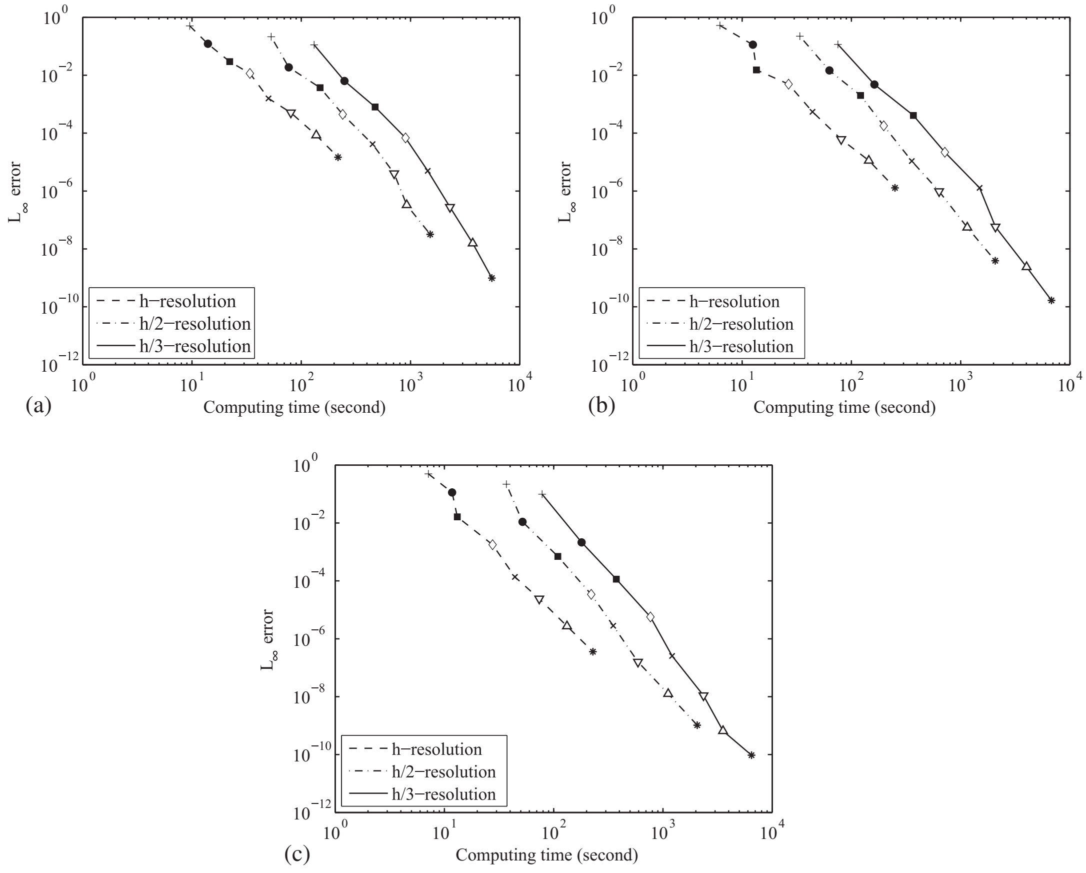
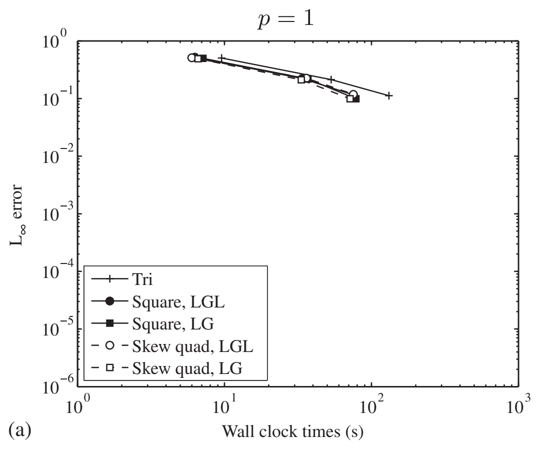


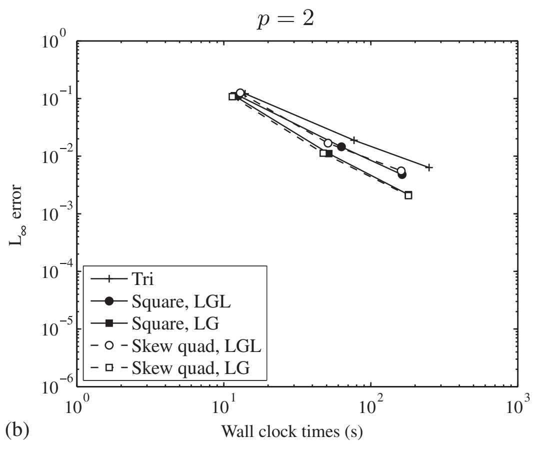
Figure 12. $\|u-u_h\|_{\mathcal{N},\infty}$ at $t=0.8$ versus computing time (in s) used for meshes of three different resolution: (a) triangular elements; (b) square elements with the LGL nodal set; and (c) square elements with the LG nodal set. $+$, $p=1$; $\bullet$, $p=2$; $\blacksquare$, $p=3$; $\diamond$, $p=4$; $\times$, $p=5$; ∇, $p=6$; $\triangle$, $p=7$; and $*$, $p=8$.

triangular elements, is higher than that predicted by the estimate discussed above. Note that we speculate that the disparity of c_q/c_t between the estimate and the numerical results stems partly from the efficiency of memory traffic and cache management.

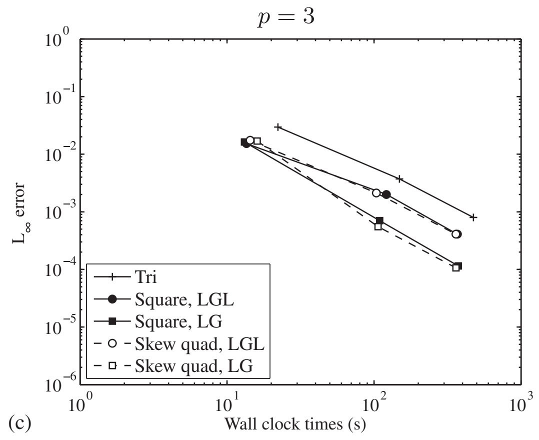
Next we consider the relationship between the accuracy of the approximate solution using different element types and the total computing times used. The computing times reported in what follows are the results from the computer code compiled with the optimization turned on. Figure 12 shows the maximum absolute error in the solution at $t=0.8$ versus the computing time required to integrate the problem in time. Figure 12(a) reports the results where the triangular meshes are used. The numerical results obtained on the square meshes with the LGL nodal set are plotted in Figure 12(b) and with the LG nodal set in Figure 12(c). Note that each curve is the result of using different levels of resolution. The symbols on each line denote the error for different orders of basis p employed. It can be observed that, as the order p increases, as expected, the computing time required is longer in order to achieve spectral accuracy. The longer run times are a consequence of the use of smaller Δt in the RKF45 (note here we let the RKF45 select



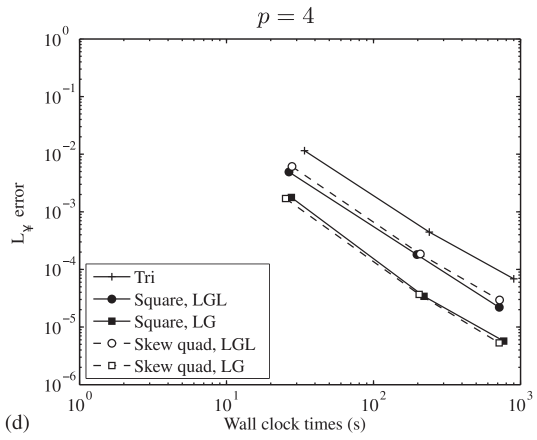
(a)



(b)



(c)



(d)

Figure 13. $\|u-u_h\|_{\mathcal{N};\infty}$ at $t=0.8$ versus computing time (in s) used in the methods of different element types of fixed p: (a) $p=1$; (b) $p=2$; (c) $p=3$; and (d) $p=4$.

the size of Δt automatically). In addition to obtaining a solution of a certain specified accuracy, the computing time required is shorter for a coarser mesh with a higher order p. In other words, it is more efficient to use a coarser mesh with a higher order p. To see more clearly how the methods compare on the meshes of different resolutions, we plot the error on the mesh of different resolutions when the order p is held fixed against the computing times in Figure 13 for $p=1$–4 and in Figure 14 for $p=5$–8. Note that each line corresponds to a result of a different element type. The symbols on each line represent the three different resolutions. It can be observed that, for the quadrilateral elements, to obtain a certain accuracy, the use of the LG nodal set takes less computing time than that of the LGL nodal set for most cases. In addition, regarding the computing cost to achieve a similar accuracy, the DG solutions on the quadrilateral elements (both square and skewed-rectangular elements) are more efficient than that on the triangular elements. To gain a better idea quantitatively of how these methods compare, we fit, in a least square sense, the data (in log–log scale) with a linear function and then calculate from the fitting result a computing time for a required accuracy, denoted as ε. Table II tabulates the computing times corresponding to the

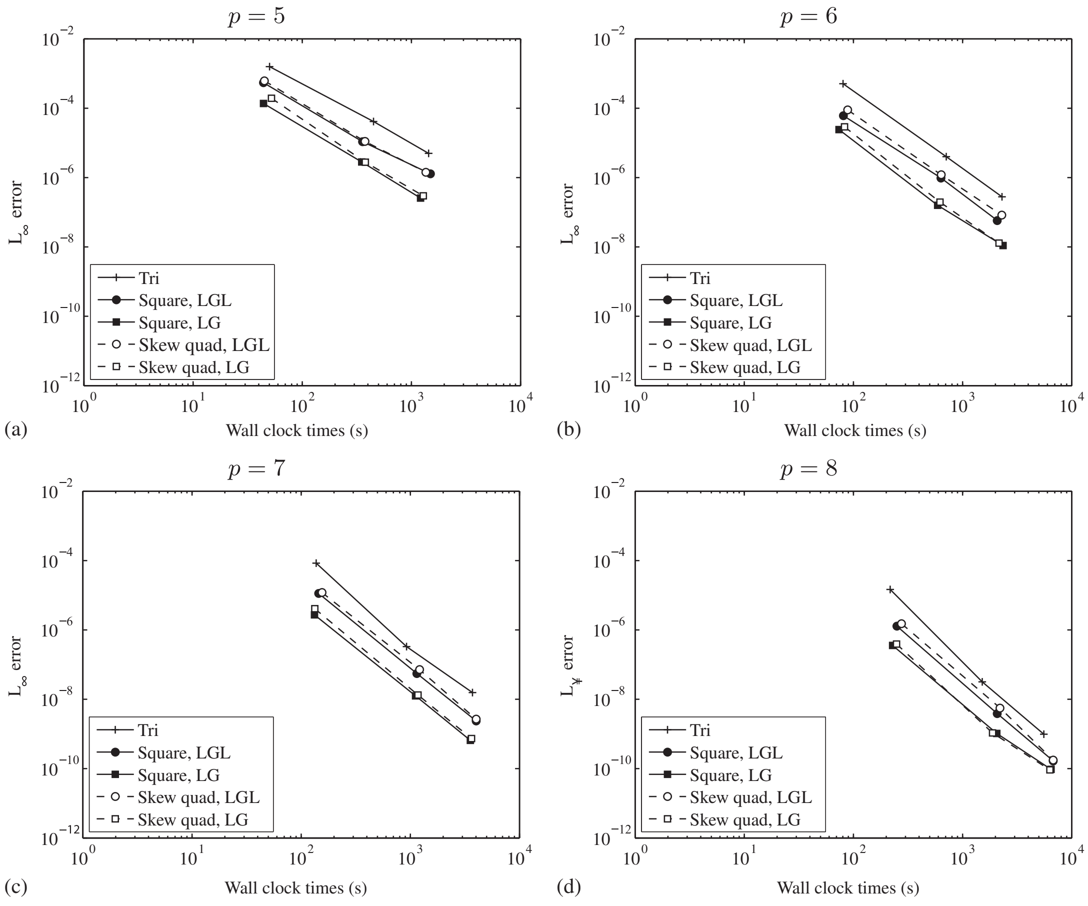


Figure 14. $\|u - u_h\|_{\mathcal{N},\infty}$ at $t = 0.8$ versus computing time (in s) used in the methods of different element types of fixed p: (a) $p = 5$; (b) $p = 6$; (c) $p = 7$; and (d) $p = 8$.

given values of ε for different types of elements and the different values of p. Note that the values of ε are selected such that they are well in the range of data considered. In this table, a numeric value inside a parenthesis denotes the ratio of the time of the considered method to that of the triangular elements with the same value of order p. From this table, it can be seen that the nodal DG methods using the quadrilateral elements are approximately 1.4 to 5.4 times more efficient than those using the triangular elements for a required accuracy. The quadrilateral elements employing the LG nodal set are typically two times more efficient than those using the LGL nodal set.

Next we consider DG solutions of the problem on two meshes shown in Figure 15. Figure 15(a) shows the unstructured-triangular mesh and Figure 15(b) the mixed mesh built by naively merging adjacent triangles in the original triangular mesh into quadrilaterals. The triangular mesh consists of 5344 elements and the mixed mesh consists of 738 triangles and 2308 skewed rectangles. The merging process is carried out in a way that a resulting quadrilateral element has a determinate Jacobian. In other words, we do not merge two triangles forming a quadrilateral with reflex interior

Table II. Computing times corresponding to a given value of error ε of different types of elements.

	Order of basis p							
Method	1 $\varepsilon=$ 2.0E−1	2 $\varepsilon=$ 1.0E−2	3 $\varepsilon=$ 1.0E−3	4 $\varepsilon=$ 2.0E−4	5 $\varepsilon=$ 2.0E−5	6 $\varepsilon=$ 5.0E−6	7 $\varepsilon=$ 1.0E−6	8 $\varepsilon=$ 1.0E−7
Tri. elements*	58.01	153.37	415.19	433.69	657.62	635.83	699.61	1121.40
Sq. elements, LGL†	38.54 (1.5)	87.74 (1.7)	185.56 (2.2)	186.05 (2.3)	283.57 (2.3)	270. 74 (2.3)	373.36 (1.8)	638.15 (1.8)
Sq. elements, LG‡	40.18 (1.4)	59.98 (2.6)	86.39 (4.8)	91.80 (4.7)	122.76 (5.4)	142.52 (4.5)	196.93 (3.6)	367.58 (3.1)
Skewed elements, LGL§	41.33 (1.4)	91.58 (1.7)	177.90 (2.3)	214.43 (2.0)	294.89 (2.2)	336.31 (1.9)	414.67 (1.7)	742.28 (1.5)
Skewed elements, LG¶	35.21 (1.6)	56.97 (2.7)	85.02 (4.9)	83.85 (5.2)	155.12 (4.2)	169.49 (3.8)	227.57 (3.1)	393.41 (2.9)

*Triangular elements.

†Square elements with the LGL nodal set.

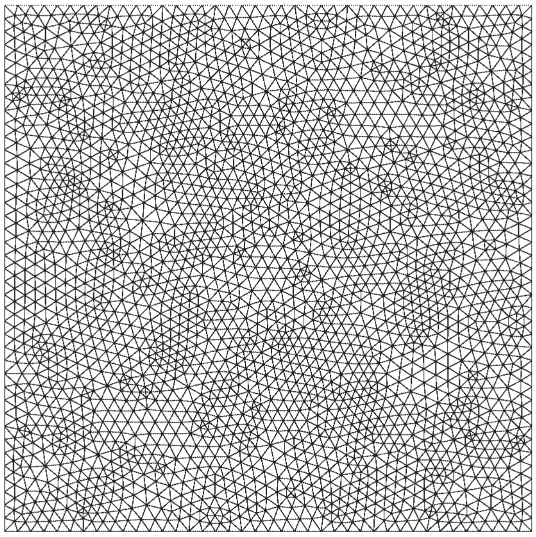
‡Square elements with the LG nodal set.

§Skewed-rectangular elements with the LGL nodal set.

¶Skewed-rectangular elements with the LG nodal set.

angles ($\geqslant 180^\circ$). Note that the majority of the triangles in the original mesh have interior angles of approximately 60° and consequently quadrilateral elements in the mixed mesh are mostly diamond shaped with interior angles of approximately 60° and 120°. The problem is solved with the parameters in RKF45 set to $\varepsilon_r = 10^{-5}$ and $\varepsilon_a = 10^{-10}$. Table III reports the resulting computing time required to integrate the problem to time $t = 0.8$ for various orders of p. Errors in the solutions, $\|u - u_h\|_{\mathcal{N};\infty}$, are tabulated in Table IV. Figure 16 plots the error in the solution versus the computing time. It can be observed from Table III that, for the orders of the basis p up to 5, the computing times required in the integration for the mixed mesh are shorter than for the triangular mesh (a ratio of run times is closer to unity as p increases) and roughly the same at $p = 6$. For $p \geqslant 7$, the use of the mixed mesh requires longer computing time than that of the triangular mesh. This behavior is consistent with the behavior observed in the previous tests (when meshes used are all quadrilaterals). From Figure 16, it can be observed that, DG solutions with the mixed mesh have slightly higher efficiency in term of computing cost to achieve a similar accuracy (at the nodal points). For the mixed mesh case, the use of quadrilateral elements with the LG nodal set yields similar computational efficiency as with the LGL nodal set.

The numerical results discussed above provide clear evidence that there is a benefit in using quadrilateral elements, especially, with the LG nodal set. For low-order p, the methods using the quadrilateral elements would be faster or as fast as the methods using the triangular elements (given that the quadrilateral mesh consists roughly of half as many elements as the triangular mesh) to reach the final time of a simulation. It is noted here that in computing the action of the stiffness matrices (23) and (26) on a given vector, the derivative matrix $D^{\xi}_{i,j}$ and $D^{\eta}_{i,j}$ are kept in a full format. For quadrilateral elements, the number of nonzero entries in each row of such matrices is $(p+1)$ instead of $(p+1)^2$ owing to the tensor product nature of the basis functions. Therefore, the calculation of this action of these matrices on the vector can be speeded up by using a sparse matrix–vector multiplication algorithm since this scales like $O((p+1)^3)$ instead of

(a)


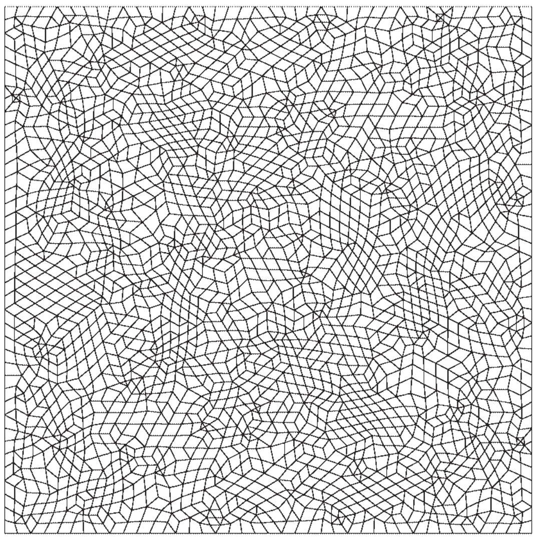
(b)


Figure 15. Two computational meshes: (a) triangular mesh (5344 elements) and (b) mixed mesh (738 triangles, 2308 skewed rectangles), obtained by merging adjacent triangles into quadrilaterals.

$O((p+1)^4)$. Although not tested here, we believe that, by exploiting the sparseness of $D^{\xi}_{i,j}$ and $D^{\eta}_{i,j}$, the computing time required in the calculations using the quadrilateral elements would be lower in comparison to those reported here. Quadrilateral elements also benefit from the fact that the nodal basis on a quadrilateral can represent more cross terms of polynomials than the nodal basis on a triangle, thus it can be expected in general that the approximate solutions from the quadrilateral elements would have better accuracy or, at worst, approximately the same accuracy as those from the triangular elements (which unfortunately we do not know *a priori* for a general

Table III. Computing time (in s) for triangular and mixed meshes required to reach $t=0.8$.

	Triangular mesh	Mixed mesh (728 Tri, 2308 Quad.)	
Order p	($N_{\mathrm{el}}=5344$)	Tri+LGL elem.*	Tri+LG elem.†
1	1.07e+02	6.87e+01	6.77e+01
2	1.68e+02	1.26e+02	1.32e+02
3	3.18e+02	2.64e+02	2.77e+02
4	6.41e+02	5.49e+02	5.66e+02
5	9.76e+02	9.27e+02	9.30e+02
6	1.80e+03	1.66e+03	1.78e+03
7	2.71e+03	2.86e+03	2.75e+03
8	4.22e+03	5.34e+03	5.02e+03

*Triangular elements and quadrilateral LGL elements.
†Triangular elements and quadrilateral LG elements.

Table IV. Error in solution, $\|u-u_h\|_{\mathcal{N},\infty}$, at $t=0.8$.

	Triangular mesh	Mixed mesh (728 Tri, 2308 Quad.)	
Order p	($N_{\mathrm{el}}=5344$)	Tri+LGL elem.*	Tri+LG elem.†
1	1.10e−01	1.54e−01	1.74e−01
2	1.58e−02	1.33e−02	1.33e−02
3	1.74e−03	1.67e−03	1.12e−03
4	1.65e−04	1.04e−04	1.03e−04
5	1.60e−05	9.06e−06	7.46e−06
6	1.15e−06	6.19e−07	6.18e−07
7	7.54e−08	4.24e−08	2.90e−08
8	5.38e−09	2.75e−09	2.79e−09

*Triangular elements and quadrilateral LGL elements.
†Triangular elements and quadrilateral LG elements.

problem). This expectation makes the use of the quadrilateral elements particularly appealing for the low to moderate order p since this implies that the methods with the quadrilateral elements would likely be more efficient to obtain a required level of accuracy. If the former scenario is the case, the use of quadrilateral elements with higher order p is still appealing due to their higher accuracy per cost as shown here.

4. CONCLUSIONS

In this work, we have conducted a study of DG solutions on triangles and quadrilaterals. The main motivation is to gain more insight into whether the quadrilateral element would improve the efficiency of the DG methods in a setting in which one quadrilateral element may be formed by combining two (adjacent) triangles. For a quadrilateral element, we consider two tensor product

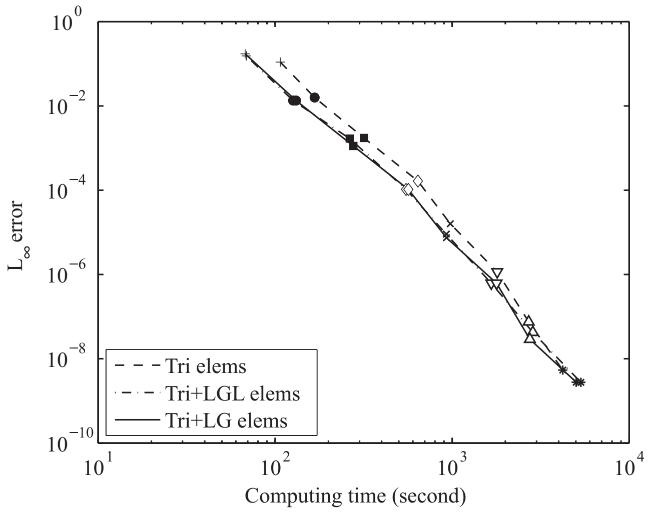


Figure 16. $\|u-u_h\|_{\mathcal{N};\infty}$ at $t=0.8$ versus computing time (in s) used in the triangular mesh and in the mixed mesh. $+$, $p=1$; $\bullet$, $p=2$; $\blacksquare$, $p=3$; $\diamond$, $p=4$; $\times$, $p=5$; $\triangledown$, $p=6$; $\triangle$, $p=7$; $*$, $p=8$.

nodal bases: one is associated with the LGL nodal points and the other is associated with the LG nodal points. An efficient means to evaluate the elemental matrices in these bases was described. To assess the performance of these methods we consider the nodal bases of various order p and different configurations with various mesh resolutions. The linear transport problem is used as a test problem. Here, the integration in time is carried out using RKF45, which has a mechanism to automatically adjust the time step sizes used in the integration. The numerical solutions show that all the methods considered exhibit spectral convergence when the resolution of the computational meshes is held fixed and the order of basis p is varied. The maximum absolute error at the nodes in the approximate solutions behaves like $O(h^{p+1})$ when p is held fixed and the mesh size is changed. From an efficiency standpoint, it is more efficient to use a coarser mesh with high-order p. The results from numerical experiments demonstrate that, for the cases where the order of the basis is low to moderate (up to $p=5$–6), the quadrilateral elements require less computing time than the triangular elements in order to reach the given final time (although the crude estimate on the cost of evaluating the right hand side predicts that this may not be the case for $p\geqslant 1$). For the tests conducted, the quadrilateral elements yield between 1.4 to 5.4 times higher computational efficiency in term of cost to achieve a similar accuracy (at the nodes). The use of quadrilateral elements with the LG nodal set yields the best computational efficiency among the methods tested. The numerical results provide evidence that there may be a substantial benefit in using the quadrilateral elements, in particular, when using low to moderate order p.

ACKNOWLEDGEMENTS

This work was supported by National Science Foundation grants DMS-0620696, DMS-0620697, OCI-0746232, OCI-0749015, and by the Office of Naval Research, Award Number N00014-06-10285.

REFERENCES

1. Cockburn B, Shu C. Runge–Kutta discontinuous Galerkin methods for convection-dominated problems. *Journal of Scientific Computing* 2001; **16**:173–261.
2. Cockburn B. Discontinuous Galerkin methods. *ZAMM Zeitschrift für Angewandte Mathematik und Mechanik* 2003; **83**:731–754.
3. Aizinger V, Dawson C. A discontinuous Galerkin method for two-dimensional flow and transport in shallow water. *Advances in Water Resources* 2002; **25**:67–84.
4. Eskilsson C, Sherwin SJ. A triangular spectral/hp discontinuous Galerkin method for modelling 2D shallow water equations. *International Journal for Numerical Methods in Fluids* 2004; **45**:605–623.
5. Dawson C, Westerink J, Feyen J, Pothina D. Continuous, discontinuous, and coupled discontinuous–continuous Galerkin finite elements methods for the shallow water equations. *International Journal for Numerical Methods in Fluids* 2006; **52**:63–88.
6. Kubatko E, Westerink J, Dawson C. HP discontinuous Galerkin methods for advection dominated problems in shallow water flow. *Computer Methods in Applied Mechanics and Engineering* 2006; **196**:437–451.
7. Bunya S, Kubatko E, Westerink J, Dawson C. A wetting and drying treatment for the Runge–Kutta discontinuous Galerkin solution to the shallow water equations. *Computer Methods in Applied Mechanics and Engineering* 2009; **198**:1548–1562.
8. Kubatko E, Bunya S, Dawson C, Westerink J, Mirabito C. A performance comparison of continuous and discontinuous finite element shallow water models. *Journal of Scientific Computing* 2009; **40**:315–339.
9. Hesthaven J, Warburton T. Nodal high-order methods on unstructured grids I. Time-domain solution of Maxwell's equations. *Journal of Computational Physics* 2002; **181**:186–221.
10. Hesthaven J, Warburton T. *Nodal Discontinuous Galerkin Methods*: *Algorithms*, *Analysis*, *and Application*. Springer Science+Business Media, LLC: Berlin, 2008.
11. LeVeque RJ. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press: Cambridge, 2002.
12. Qiu J, Khoo BC, Shu C. A numerical study of the performance of the Runge–Kutta discontinuous Galerkin method based on different numerical fluxes. *Journal of Computational Physics* 2006; **212**:540–565.
13. Hesthaven J. From electrostatics to almost optimal nodal sets for polynomial in simplex. *SIAM Journal on Numerical Analysis* 1998; **35**:655–676.
14. Solín P. *Partial Differential Equations and the Finite Element Method*. Wiley: New Jersey, 2006.
15. Dubiner M. Spectral methods on triangle and other domains. *Journal of Scientific Computing* 1991; **6**:345–390.
16. Gordon WJ, Hall CA. Construction of curvilinear coordinate systems and application to mesh generation. *International Journal for Numerical Methods in Engineering* 1973; **7**:461–477.
17. Zienkiewicz OC, Taylor RL, Zhu JZ. *The Finite Element Method*: *its Basis and Fundamentals* (6th edn). Elsevier: Amsterdam, 2005.
18. Canuto C, Hussaini MY, Quarteroni A, Zang TA. *Spectral Methods in Fluid Dynamics*. Spring Series in Computational Physics. Springer: Berlin, 1988.
19. Dongarra J, Croz J, Hammarling S, Hanson R. An extended set of fortran basic linear algebra subprograms. *Technical Memorandum 41*, Argonne National Laboratory, Argonne, IL, 1998. Available from: http://www.netlib.org/blas.
20. Butcher J. *Numerical Methods for Ordinary Differential Equations*. Wiley: New York, 2003.
21. Shampine L, Watts H, Davenport S. Solving nonstiff ordinary differential equations—state of art. *SIAM Review* 1976; **18**:376–411. Available at: http://www.netlib.org/fmm/rkf45.f.
22. Richter G. An optimal-order error estimate for the discontinuous Galerkin method. *Mathematics of Computation* 1998; **50**:75–88.
23. Peterson T. A note on the convergence of the discontinuous Galerkin method for a scalar hyperbolic equation. *SIAM Journal on Numerical Analysis* 1991; **28**:133–140.
24. Zhang Q, Shu C. Error estimates to smooth solutions of Runge–Kutta discontinuous Galerkin methods for scalar conservative laws. *SIAM Journal on Numerical Analysis* 2004; **42**:641–666.