Discontinuous Galerkin Methods with Nodal and Hybrid Modal/Nodal Triangular, Quadrilateral, and Polygonal Elements for Nonlinear Shallow Water Flow

D. Wirasaet[a,*], E. J. Kubatko[b], C. E. Michoski[c], S. Tanaka[a], J. J. Westerink[a], C. Dawson[c]

[a] *Computational Hydraulics Laboratory, Department of Civil Engineering and Geological Sciences, University of Notre Dame, Notre Dame, IN, 46556, U.S.A.*
[b] *Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, Columbus, OH, 43210, U.S.A.*
[c] *Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX, 78712, U.S.A.*

Abstract

We present a comprehensive assessment of nodal and hybrid modal/nodal discontinuous Galerkin (DG) finite element solutions on a range of unstructured meshes for nonlinear shallow water flow. The nodal DG methods on triangles and a tensor-product nodal basis on quadrilaterals are considered. The hybrid modal/nodal DG methods utilize two different polynomial bases on polygons in realizing the DG discretization; orthogonal basis functions constructed by the Gram-Schmidt process are used as trial and test functions in a DG weak formulation and a nodal basis is used as an efficient means for area integration. The performance of these methods in terms of accuracy and computational cost is demonstrated using h and p convergence studies on a two-dimensional nonlinear shallow water problem with a manufactured solution. To assess the performance of quadrilateral and polygonal elements in comparison to triangular elements, we consider a setting in which a quadrilateral mesh, a mixed triangular-quadrilateral mesh, and polygonal mesh are derived from a given triangular mesh and *vice versa*. The tests conducted reveal the merit of using the quadrilateral elements in terms of computational cost per accuracy and

*Corresponding author
Email address: `dwirasae@nd.edu` (D. Wirasaet)

computing time. The numerical results clearly show that high order schemes significantly improve the cost performance for a given level of accuracy, with cubic or bi-cubic interpolants particularly achieving dramatic improvement in accuracy as compared to linear interpolants, with diminishing benefit as $p > 3$.
Keywords: Discontinuous Galerkin, Finite elements, Nodal, Modal, Triangular element, Quadrilateral element, Polygonal element, Computational cost, Manufactured solution, Shallow water equations

1. Introduction

The shallow water equations (SWE) are used extensively in modeling many important physical phenomena, such as hurricane induced flooding, tides, riverine flows, tsunami waves, dam breaks, and many others. The equations when coupled with the transport equation can also be used to model problems such as salinity, heat, and contaminant movement. Simulations of such environmental fluid flow frequently involve large, geometrically complicated domains and integration over long periods of times. An accurate and efficient solution of the SWE is therefore crucial in numerical simulations. While relatively young in comparison with more conventional approaches, discontinuous Galerkin (DG) finite element methods (see [1, 2] for reviews of DG methods) have become a powerful alternative for solving the SWE [3, 4, 5, 6, 7, 8, 9, 10, 11]. Conceptually similar to finite volume methods, DG methods inherently have the property of being conservative on the element level, making them ideal for coupling flow and transport models. Additional notable advantages of DG methods include the ease of constructing high order approximations on unstructured meshes and high scalability for parallel implementation when used in conjunction with explicit time integration schemes. Since DG methods use a discontinuous approximation, they are able to accommodate non-conformal meshes and the use of different bases in each element, thus rendering them naturally well suited for a discretization with adaptive h (mesh) and p (polynomial order) refinements. While DG methods possess a number of favorable properties, one major drawback in comparison to continuous Galerkin (CG) methods on a given mesh is the larger number of degrees of freedom, which consequently translates into greater computational costs. The preliminary comparison study in [12] of CG and DG methods for the SWE shows that, when using linear interpolation on identical meshes, the cost per time step of the DG approach on serial machines is approximately four to five times more expensive than the CG approach. The subsequent study in [6] finds that the DG approach is significantly more efficient in terms of achieving a specified error level for a given computational cost and in terms of scalability on large-scale parallel machines. Note that triangular meshes are used in these studies.

The motivation behind the current work stems partly from an observation that a quadrilateral element may be obtained by merging two adjacent triangular elements and *vice versa*, two triangular elements from bisecting a quadrilateral element. Hence, by considering this mesh setting, a mesh of quadrilateral elements would consist of approximately half as many elements as a mesh of triangular elements. Furthermore, the number of edges in the quadrilateral mesh would be approximately two-thirds that of the triangular mesh. Note that evaluating area integrals and edge integrals represents the major computational cost in DG methods. From this observation, the use of quadrilateral elements appears to be an appealing means to potentially improve the computational efficiency of DG schemes. In our previous work [13], we conducted a study of the performance of the numerical solution of a linear two-dimensional transport problem using DG methods based on Lagrange nodal bases on triangles and tensor-product nodal bases on quadrilaterals. Indeed, numerical results shown there clearly demonstrated the benefit of using the nodal tensor-product bases on quadrilaterals wherein, for low to moderate interpolation order, the computing time used in the simulation is shorter than that of the simulation using the nodal bases on triangles. In addition, the nodal tensor-product DG scheme yields more computational efficiency in terms of the cost to achieve a specified maximum error level at the nodes. In this current work, we present a comprehensive performance assessment of an application of such nodal DG schemes to nonlinear shallow water flow. In addition, we also present an application of the hybrid modal/nodal DG method, devised by Gassner *el al.* [14], to the SWE. Such a DG method is based on a pair of the so-called polymorphic nodal bases on a polygon which consists of an orthogonal modal basis and its nodal basis counterpart. The former is utilized in realizing the DG discretization and the latter is employed in constructing a quadrature free approach for evaluating integrals over a polygon. Such a hybrid DG scheme furnishes a flexible and, owing to its quadrature free nature, efficient means for solving nonlinear problems. For the hybrid elements, in addition to triangular and quadrilateral meshes, we also consider meshes of polygonal elements in the performance study. Note that a common practice in assessing numerical schemes for the SWE is to apply them to the problems with analytic solutions proposed by Lynch and Gray [15]. These problems serve as invaluable tools for shallow water flow solvers; however, they are governed by the linear SWE and their analytic solution is effectively dependent on only one spatial coordinate. Here, we resort to a so-called manufactured-solution of the nonlinear SWE to generalize the investigation. Note that a problem with a manufactured solution is formulated by substituting a smooth solution function into the SWE and solving the equations with the corresponding forcing term and boundary conditions given by the manufactured solution [16]. Here, the performance of nodal and hybrid modal/nodal DG methods, both in terms of accuracy and computational cost, is systematically assessed through h and p convergence studies. We also examine the benefit of using high-order methods (here a high-order method refers to a scheme with interpolation order $p > 1$) from the perspective of computational cost for a given level of accuracy through the convergence studies.

This paper is organized as follows. In section 2, a general framework of the DG method employed in this work is summarized. Subsequently, we summarize two different bases to use with the DG method, namely, the so-called polymorphic nodal bases and the nodal bases. In section 3, we present a description of the two-dimensional nonlinear shallow water equations with a manufactured solution. Subsequently, by means of h and p convergence studies, we present a comprehensive assessment of the hybrid modal/nodal DG schemes and the nodal DG scheme.

2. Methodology

2.1. Discontinuous Galerkin methods for hyperbolic conservation laws

We first describe a framework of the specific DG formulation employed in this study. For simplicity of presentation, we consider the numerical solution of two-dimensional scalar conservation laws of the form,

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

where $u(\boldsymbol{x},t)$ is a conserved variable and $\boldsymbol{f} = (f_x(u), f_y(u))$ is a nonlinear flux with f_x and f_y denoting a flux function in the x- and y-direction, respectively. The equation is augmented with appropriate boundary and initial conditions. To discretize the equation using DG, the domain Ω is triangulated into a set of finite non-overlapping elements. Let $\mathcal{T}_h$ denote such a set of elements from triangulation. The solution u is then replaced by a discontinuous approximate solution u_h which, in each element $K \in \mathcal{T}_h$, belongs to a finite dimensional space $V(K)$. The approximate solution on the element K is determined by requiring that,

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

for all $v \in V(K)$, 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 $\boldsymbol{f}(u_h)$ which is multiply-defined on the element boundary since the approximation is not continuous across the element interface. 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 *e.g.* [17, 18]). Note that the coupling between the approximate solution in K and in its immediate neighbors enters the weak formula only through the edge integral term.

Suppose here that a finite dimensional space $V(K)$ (with desirable properties) is chosen for each element K and that $\{\widetilde{\phi}_m^K(\boldsymbol{x})\}_{m=1,\ldots,N_p}$ forms a basis of $V(K)$, where N_p denotes the dimension of the space $V(K)$. The approximate solution, when restricted to K, is then defined by

$$u_h\big|_K = \sum_{m=1}^{N_p} \widetilde{u}_m^K(t)\widetilde{\phi}_m^K(\boldsymbol{x}), \quad \boldsymbol{x} \in K, \tag{3}$$

where $\widetilde{u}_m^K(t)$ represent the time-dependent expansion coordinates. The global approximate solution corresponds simply to a direct sum of (3) over all elements. By adopting this basis, the local statement (2) for the element K reduces to the following system of ordinary differential equations (ODEs):

$$\begin{aligned} &\sum_{n=1}^{N_p} \mathcal{M}_{m,n}^K \frac{d\widetilde{u}_n^K}{dt} + \int\limits_K f_x(u_h)\frac{\partial \widetilde{\phi}_m^K}{\partial x} d\boldsymbol{x} + \\ &\qquad \int\limits_K f_y(u_h)\frac{\partial \widetilde{\phi}_m^K}{\partial y} d\boldsymbol{x} + \int\limits_{\partial K} \widehat{\boldsymbol{f}}_h \cdot \mathbf{n}\widetilde{\phi}_m^K ds = 0, \quad m = 1, \ldots, N_p, \end{aligned} \tag{4}$$

where $\mathcal{M}_{m,n}$, an entry of the elemental mass matrix, is defined by

$$\mathcal{M}_{m,n}^K = \int\limits_K \widetilde{\phi}_m^K(\boldsymbol{x})\widetilde{\phi}_n^K(\boldsymbol{x})d\boldsymbol{x}. \tag{5}$$

Note that the superscript K is used to indicate an affiliation of the basis functions and their expansion coordinates with the element K. Hereafter, this superscript is dropped for notational simplicity.

The integrals involving the nonlinear terms, namely, the second and third term in (4), are conventionally evaluated by using a quadrature rule of a certain order depending on the form of the integrand. In this work, we adopt a technique frequently used in spectral methods and in nodal DG methods [19, 20, 21, 14] to evaluate these integrals. This technique relies on the so-called nodal basis, another basis for $V(K)$, to construct a simple but efficient means in treating the nonlinear terms. Here, let $\{\phi_m \in V(K)\}_{m=1,\ldots,M_p}$ with $M_p \geq N_p$ be a nodal basis associated with the interpolation points $\{\boldsymbol{x}_m \in K\}_{m=1,\dots,M_p}$. The nodal basis functions possess the so-called interpolation property, namely,

$$\phi_m(\boldsymbol{x}_n) = \delta_{m,n} = \begin{cases} 1 & \text{for} \quad m = n \\ 0 & \text{for} \quad m \neq n \end{cases}. \tag{6}$$

It is noted that the nodal basis functions can be systematically defined through the basis $\{\widetilde{\phi}_m\}$. The nodal basis functions as well as its associated $\{\widetilde{\phi}_m\}$ used in this work are discussed in the subsequent sections. Notice that we allow the number of nodal basis functions M_p to be greater than the number of trial basis functions N_p. In the case where $M_p > N_p$, the property (6) of the nodal basis functions holds in an approximate sense only, *i.e.* the value of the nodal basis functions are close to unity at their associated node and close to zero at the other nodes. With the nodal basis functions at hand, the nonlinear flux term is approximated as an interpolant as follows

$$f_x(u_h(\boldsymbol{x})) \approx (If_x)(\boldsymbol{x}) \equiv \sum_{m=1}^{M_p} f_{x,m}\phi_m(\boldsymbol{x}) = \boldsymbol{\phi}^T \boldsymbol{f}_x \tag{7}$$

where $\boldsymbol{f}_x = \{f_{x,1}, \dots, f_{x,M_p}\}^T$ and $\boldsymbol{\phi} = \{\phi_1, \dots, \phi_{M_p}\}^T$ (the nodal representation of the y-directed flux $(If_y)(\boldsymbol{x})$ is defined in an analogous fashion). Here, the nodal coordinates are simply defined by $f_{x,m} = f_x(u_h(\boldsymbol{x}_m))$. By adopting the nodal representation (7), formula (4) becomes the following system of ODEs,

$$\begin{aligned} \sum_{n=1}^{N_p} \Bigg(\mathcal{M}_{m,n} \frac{d\widetilde{u}_n}{dt} &- \mathcal{S}_{x,(m,n)} f_{x,n} - \\ &\mathcal{S}_{y,(m,n)} f_{y,n} \Bigg) + \int_{\partial K} \widehat{\boldsymbol{f}}_h \cdot \mathbf{n} \widetilde{\phi}_m ds = 0, \; m = 1, \dots, N_p. \end{aligned} \tag{8}$$

where $\mathcal{S}_{x,(m,n)}$ denotes an entry of the so-called general stiffness matrix $\boldsymbol{\mathcal{S}}_x$ of dimension $N_p \times M_p$ [14] defined as follows

$$\boldsymbol{\mathcal{S}}_x \equiv \int_K \frac{\partial \widetilde{\boldsymbol{\phi}}}{\partial x} \boldsymbol{\phi}^T d\boldsymbol{x}. \tag{9}$$

where $\boldsymbol{\phi} = \{\phi_1, \ldots, \phi_{M_p}\}^T$ and $\widetilde{\boldsymbol{\phi}} = \{\widetilde{\phi}_1, \ldots, \widetilde{\phi}_{N_p}\}^T$ (simply replace x by y in (9) for an expression of the y-directed general stiffness matrix $\boldsymbol{\mathcal{S}}_y$).

The solution procedure consists of integrating ODEs (8) for the expansion coordinates $\{\widetilde{u}_m^K\}$, $K \in \mathcal{T}_h$ and subsequently substituting the resulting expansion coordinates into (3) to obtain the approximate solution. In (8), the expansion coordinates from different elements appear solely in the numerical flux term; therefore, the global mass matrix of the DG scheme is decoupled-block diagonal and thus can be inverted in an elementwise fashion. The inversion of the global mass matrix can be made even simpler by choosing $\{\widetilde{\phi}_m\}$ to form an orthogonal basis since, in this case, the global mass matrix is simply decoupled diagonal. Note that the elemental mass matrix, its inverse, and the general stiffness matrices $\boldsymbol{\mathcal{S}}_x$ and $\boldsymbol{\mathcal{S}}_y$ for each element can be pre-computed and stored at the initial stage of simulations. This method has a computational advantage in the sense that the number of operations required in the evaluation of the non-linear flux is proportional to the number of nodes regardless of the form of the non-linear flux. The disadvantage of this approach is that there is error introduced through the interpolation process. Such an error, known as an alias error, may induce an instability for marginally resolved computations [19, 21]. In this case, an instability can still be effectively controlled by employing a de-aliasing strategy [19, 21].

The remaining tasks in defining a DG scheme concern choosing the finite dimensional approximation space, its associated basis functions, a time discretization scheme, and a Riemann solver. The next two subsections describe two particular sets of polynomial bases, namely the polymorphic nodal bases and the nodal bases, to be used in the framework described above. Thereafter, we discuss in brief a time integration scheme employed in this study.

2.2. Polymorphic nodal elements: modal and nodal basis

Below, we summarize the construction of the so-called polymorphic nodal bases for a convex polygon devised by Gassner *et al.*[14]. Such bases consist of an orthogonal polynomial basis to be used as a set of trial and test functions and its associated nodal basis counterpart to be used in treating nonlinear terms.

For a given convex polygon K, we introduce a coordinate transformation $\boldsymbol{\xi}_K : K \to \mathcal{K}$, connecting an element K with a so-called reference element $\mathcal{K}$, as follows

$$\boldsymbol{\xi} = \boldsymbol{\xi}_K(\boldsymbol{x}) = \frac{\boldsymbol{x} - \boldsymbol{x}_c}{\Delta X}, \quad \boldsymbol{x} \in K, \tag{10}$$

where $\boldsymbol{\xi} = (\xi, \eta)$, $\boldsymbol{x} = (x, y)$, $\boldsymbol{x}_c$ denotes the centroid of K, and $\Delta X = \max(x_{\max} - x_{\min}, y_{\max} - y_{\min})$ is a scaling factor (see Figure 1 for a depiction). The reference element $\mathcal{K}$ is the range of the coordinate transformation. Note that the transformation (10) amounts simply to a rigid-body translation and linear scaling of the element K. Let $\{\pi_m\}_{m=1,\dots,N_p}$ be the monomial basis

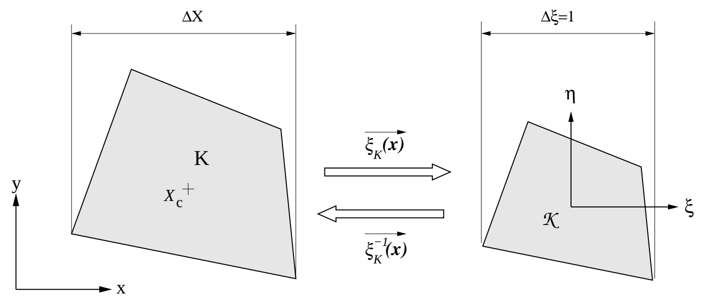


Figure 1: Schematic diagram of coordinate transformation.

of $P^p(\mathcal{K})$, the space of polynomials with degree of at most p, namely

$$\pi_m(\boldsymbol{\xi}) = \xi^i \eta^j, \quad i, j \geq 0, i + j \leq p, \tag{11}$$

$$m = \frac{1}{2}(i + j + 1)(i + j + 2) - i \tag{12}$$

The number of basis functions N_p and the order p are related through

$$N_p = \frac{(p + 1)(p + 2)}{2}. \tag{13}$$

An orthonormal basis $\{\widetilde{\phi}_m(\boldsymbol{\xi})\}_{m=1,\dots,N_p}$ is subsequently constructed by applying a modified Gram-Schmidt process [22] with the usual L_2 inner product to the monomial basis (see Algorithm 1 for the description of the orthonormalization procedure). Consequently, the basis functions $\widetilde{\phi}_m(\boldsymbol{\xi})$ possess the orthonormal

Algorithm 1 A modified Gram-Schmidt process

```
for $m = 1$ to $N_p$ do
    $\widetilde{\phi}_m \leftarrow \pi_m$
    for $j = 1$ to $m - 1$ do
        $\widetilde{\phi}_m \leftarrow \widetilde{\phi}_m - \frac{(\widetilde{\phi}_m, \widetilde{\phi}_j)}{(\widetilde{\phi}_j, \widetilde{\phi}_j)}\widetilde{\phi}_j$        $\triangleright\ (f, g) = \int_{\mathcal{K}} fg d\boldsymbol{\xi}$
    end for
    $\widetilde{\phi}_m \leftarrow \frac{\widetilde{\phi}_m}{\sqrt{(\widetilde{\phi}_m, \widetilde{\phi}_m)}}$
end for
```

property, more precisely,

$$\int_{\mathcal{K}} \widetilde{\phi}_m(\boldsymbol{\xi})\widetilde{\phi}_n(\boldsymbol{\xi})d\boldsymbol{\xi} = \delta_{m,n}.$$

The basis functions on the physical element K can then be defined as follows

$$\widetilde{\phi}_m(\boldsymbol{x}) = (\widetilde{\phi}_m \circ \boldsymbol{\xi}_K)(\boldsymbol{x}), \quad m = 1, \dots, N_p. \tag{14}$$

Here, for notational simplicity, we use an identical notation for the basis functions on physical element K and on the reference element $\mathcal{K}$. Since the transformation (10) is affine, the space spanned by $\{\widetilde{\phi}_m(\boldsymbol{x})\}$ is therefore a space of polynomials with degree of at most p. In addition, $\{\widetilde{\phi}_m(\boldsymbol{x})\}$ forms an orthogonal basis over K owing to the geometric transformation (10) having a constant Jacobian. Figure 2 illustrates, as an example, the orthogonal basis functions for $p = 2$ on a pentagonal element.

It is noted that the Gram-Schmidt process may be applied directly to the monomial basis defined on K. The reason we define the basis functions on K through the basis functions on the reference element $\mathcal{K}$ stems primarily from the practical aspect of implementing the Gram-Schmidt process on a finite precision

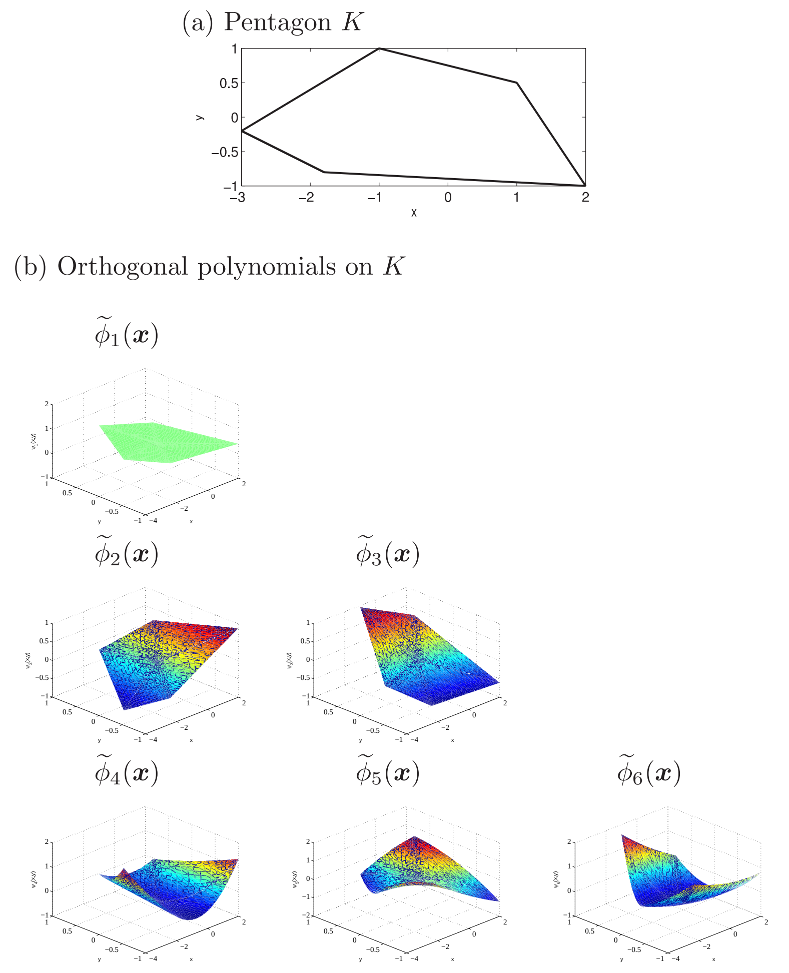


Figure 2: Orthogonal basis functions for $p = 2$ on a pentagon constructed from using a modified Gram-Schmidt process.

machine. The reference element $\mathcal{K}$ is centered around the origin and is of order one in size, in other words, it is well normalized. Consequently, an evaluation of monomial basis functions does not involve a point with exceedingly large and or small numeric values, a scenario possibly encountered in the case of working directly in the physical element when the element size is very large or very small. We find that implementing the Gram-Schmidt process on the transformed element yields more favorable results especially for large order p.

For a given set of nodal points $\Omega_I(p) = \{\boldsymbol{x}_m \in K\}_{m=1,\dots,M_p}$, a so-called nodal basis $\{\phi_m(\boldsymbol{x})\}_{m=1,\dots,M_p}$ and it associated coordinate $\{u_m\}_{m=1,\dots,M_p}$ are constructed from considering the following conditions, for $u(\boldsymbol{x}) \in P^p(K)$,

$$\phi_m(\boldsymbol{x}_n) = \delta_{m,n}, \quad u(\boldsymbol{x}) = \sum_{m=1}^{N_p} \widetilde{u}_m \widetilde{\phi}_m(\boldsymbol{x}) \doteq \sum_{m=1}^{M_p} u_m \phi_m(\boldsymbol{x}). \tag{15}$$

As a result, the transformations between two representations are determined by

$$\boldsymbol{u} = \boldsymbol{V}\widetilde{\boldsymbol{u}} \quad \text{and} \quad \widetilde{\boldsymbol{\phi}} = \boldsymbol{V}^T \boldsymbol{\phi} \tag{16}$$

where $\boldsymbol{u} = \{u_1, \dots, u_{M_p}\}^T$, $\widetilde{\boldsymbol{u}} = \{\widetilde{u}_1, \dots, \widetilde{u}_{N_p}\}^T$, $\boldsymbol{\phi} = \{\phi_1, \dots, \phi_{M_p}\}^T$, $\widetilde{\boldsymbol{\phi}} = \{\widetilde{\phi}_1, \dots, \widetilde{\phi}_{N_p}\}^T$ and $\boldsymbol{V}$ is a generalized Vandermonde matrix whose entries are given by

$$V_{m,n} = \widetilde{\phi}_n(\boldsymbol{x}_m), \quad m = 1, \dots, N_p, \; n = 1, \dots, M_p. \tag{17}$$

The remaining task in defining the nodal basis involves choosing a nodal set. The distribution of the nodal points has a crucial implication on the quality of an interpolant. It is known that a high quality interpolant, indicated by a small value of the Lebesgue constant [23], can be achieved with node sets having nodal points clustered in the vicinity of the boundaries of the element. Here, we use the specific framework devised by Gassner *et. al.* [14] to generate a nodal set yielding such a desirable effect. Such a nodal set consists of a set of nodes on the element boundary and a set of nodes in the interior. The interior nodes are generated by nesting a set of the scaled-down boundary nodes in a way that the nodes are denser near the boundaries. More precisely, a nodal set on a given polygon of N_{gon} sides is constructed from the following formula,

$$\Omega_I(p) = \bigcup_{r=0}^{r_{\max}} \mathcal{M}_r(\Omega_I^S(p - r(N_{\mathrm{gon}} - p_d))). \tag{18}$$

Here, $\Omega_I^S(q)$ denotes a set of boundary nodes which has $q+1$ nodes with the Gauss-Lobatto node distribution on each edge of the considered polygon. Note that $\Omega_I^S(0) = \{\boldsymbol{x}_c\}$ where $\boldsymbol{x}_c$ denotes the centroid of the polygon. The mapping $\mathcal{M}_r$ generates the interior nodes by means of scaling down, with a certain factor depending on the nesting step r, the boundary point set Ω_I^S (note that $\mathcal{M}_0$ is an identity mapping). See Gassner *et. al.* [14] for the detailed account of the mapping $\mathcal{M}_r$. Figure 3 shows, as an example, a nodal set for $p = 5$ on a quadrilateral, triangular, and pentagonal element. In (18), the parameter p_d with $0 \le p_d < N_{\mathrm{gon}}$ is used in adjusting the number of interior nodes. See Figure 3(a) and (b) for a comparison of the node sets with different values p_d. Note that including more interior points by increasing the value of p_d improves the accuracy of an interpolant [14], however, at the expense of computational efficiency in terms of the number of operations required. For a given p and p_d, the maximum number of interior nesting steps $r_{\max}$ in (18) is determined by

$$r_{\max} = \mathrm{floor}\left(\frac{p}{N_{\mathrm{gon}} - p_d}\right). \tag{19}$$

where floor(x) corresponds to the largest integer not greater than x and the number of nodes M_p is given by

$$M_p = N_{\mathrm{gon}}(r_{\max} + 1)\left\{p - \frac{1}{2} r_{\max}(N_{\mathrm{gon}} - p_d)\right\} + \delta_{0,p-r_{\max}(N_{\mathrm{gon}}-p_d)}. \tag{20}$$

It can be verified that the number of nodal points M_p from this construction is in general greater than N_p (except for a triangle element where $M_p = N_p$). For $M_p \neq N_p$, an inverse of the Vandermonde matrix is not uniquely defined. To circumvent this issue, a pseudo-inverse Vandermonde matrix defined in the

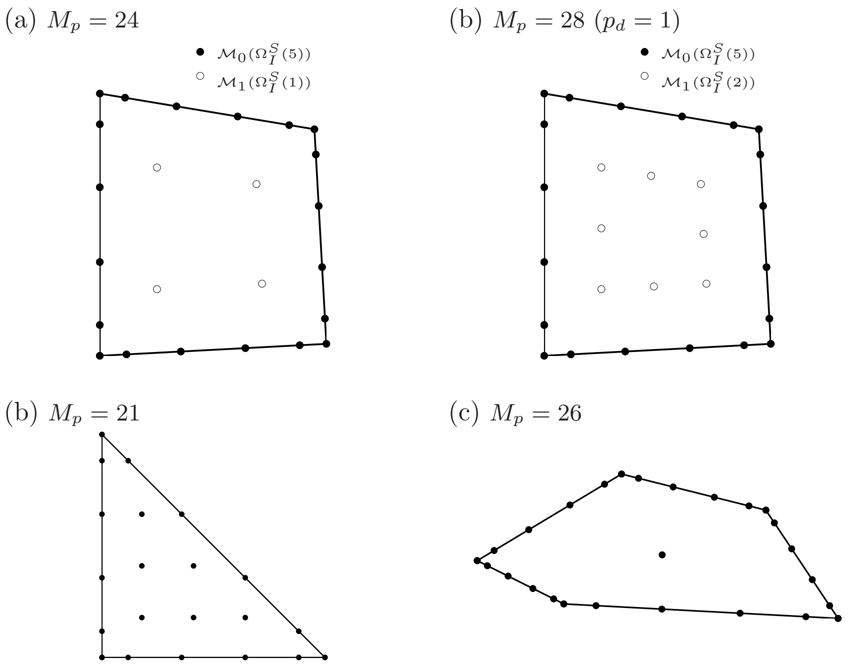

(a) $M_p = 24$
● $\mathcal{M}_0(\Omega_I^S(5))$
○ $\mathcal{M}_1(\Omega_I^S(1))$
(b) $M_p = 28$ ($p_d = 1$)
● $\mathcal{M}_0(\Omega_I^S(5))$
○ $\mathcal{M}_1(\Omega_I^S(2))$
(b) $M_p = 21$
(c) $M_p = 26$

Figure 3: Nodal distribution for $p = 5$ ($N_p = 21$): (a) quadrilateral element with $p_d = 0$ (b) quadrilateral element with $p_d = 1$ (c) triangular element with $p_d = 0$, and (d) pentagonal element $p_d = 0$

least squares sense, more specifically,

$$\boldsymbol{V}^{-1} \equiv \overline{\boldsymbol{V}}^{-1}\boldsymbol{V}^T, \quad \overline{\boldsymbol{V}} = \boldsymbol{V}^T\boldsymbol{V}, \tag{21}$$

is utilized in defining the inverse transformations

$$\widetilde{\boldsymbol{u}} = \boldsymbol{V}^{-1}\boldsymbol{u} \quad \text{and} \quad \boldsymbol{\phi} = (\boldsymbol{V}^{-1})^T\widetilde{\boldsymbol{\phi}}. \tag{22}$$

Note that for $M_p > N_p$, the set $\{\phi_m\}$ defined as above, although it spans every polynomial in $P^p(K)$, is not a basis since it is a linearly dependent set. However, for simplicity, we still call such a set the nodal basis and its members, nodal basis functions.

In practice, an explicit form for the nodal basis functions is rarely required. Instead, interpolated values at given points are obtained by first calculating the modal coordinates $\widetilde{\boldsymbol{f}} = \{\widetilde{f}_1, \ldots, \widetilde{f}_{N_p}\}^T$ from the nodal coordinates $\boldsymbol{f} = \{f_1, \ldots, f_{M_p}\}^T$ by means of an inverse transformation and subsequently calculating the interpolated values through the modal representation. As a consequence of using the pseudo-inverse Vandermonde matrix (21) in defining the nodal basis functions, except for the case where $N_p = M_p$, the nodal basis function is close but not identical to unity at its associated node and is close but not identical to zero at the other nodes, *i.e.* $\phi_m(x_n) \neq \delta_{m,n}$. Thus, a function value at the nodal points of the polynomial approximation of a function $f(\boldsymbol{x})$ defined by

$$(I_p f)(\boldsymbol{x}) = \sum_{m=1}^{M_p} f(\boldsymbol{x}_m)\phi_m(\boldsymbol{x}) = \boldsymbol{\phi}^T \boldsymbol{f}$$

is in general not identical to the value of the nodal coordinate, *i.e.* $(I_p f)(\boldsymbol{x}_i) \neq f(\boldsymbol{x}_i)$. Furthermore, the nodal representation is in general not identical to the modal representation, *i.e.*

$$(I_p f)(\boldsymbol{x}) \neq (P_p f)(\boldsymbol{x}) \equiv \widetilde{\boldsymbol{\phi}}^T \widetilde{\boldsymbol{f}}, \quad \widetilde{\boldsymbol{f}} = \boldsymbol{V}^{-1}\boldsymbol{f}.$$

Note that the scheme based on the polymorphic nodal bases utilizes the modal basis functions as the trial and test functions in the DG formulation. Owing to the orthogonality of the modal basis, the global mass matrix of this scheme is decoupled diagonal which can be trivially inverted. With the change-of-bases transformations (16) and (22) at hand, the general stiffness matrix (9) can be evaluated by considering

$$\boldsymbol{\mathcal{S}}_x = \boldsymbol{V}^T \boldsymbol{S}_x, \quad \text{where} \quad \boldsymbol{S}_x \equiv \int_K \frac{\partial \boldsymbol{\phi}}{\partial x} \boldsymbol{\phi}^T d\boldsymbol{x}. \tag{23}$$

The calculation of the general stiffness amounts to transforming a stiffness matrix $\boldsymbol{S}_x$ via the generalized Vandermonde matrix. We use a technique similar to that devised by Hesthaven and Warburton [21] in evaluating the stiffness matrix $\boldsymbol{S}_x$. This technique, which does not require Gaussian integration, is given in Appendix A.

2.3. Nodal elements: nodal bases on triangles and quadrilaterals

The DG scheme based on the nodal bases uses a nodal basis not only as an efficient means for treating nonlinear flux terms but also as trial and test functions in the DG formulation, in other words, in this scheme, $\widetilde{\phi}_m(\boldsymbol{x})$ corresponds simply to $\phi_m(\boldsymbol{x})$. Here, a nodal basis on triangles and tensor-product nodal basis on quadrilaterals are considered.

For triangular elements, although the nodal basis on triangles constructed as in the previous subsection represents an excellent candidate, we consider a nodal basis with a set of interpolation points described in [20, 24, 21]. Unlike the nodal basis described in the last subsection which is constructed in an element-by-element fashion, such a nodal basis is defined in a more conventional way, *i.e.* through a set of nodal basis functions on a single master triangle. More precisely, the nodal basis $\{\phi_m(\boldsymbol{\xi}) \in P^p(I_t)\}_{m=1,\dots,N_p}$ associated with a given nodal set $\{\boldsymbol{\xi}_m \in I_t\}_{m=1,\dots,N_p}$ on the master element $I_t = \{\boldsymbol{\xi} = (\xi, \eta) \mid \xi, \eta \geq -1 \text{ and } \xi + \eta \leq 0\}$ is first constructed using the approach described in the last subsection. Subsequently, nodal basis functions on the physical triangular element K are defined as $\phi_m(\boldsymbol{x}) = (\phi_m \circ \boldsymbol{x}_K^{-1})(\boldsymbol{x})$ where $\boldsymbol{x}_K^{-1}$ is an inverse mapping of the affine mapping $\boldsymbol{x}_K : I_t \to K$:

$$\boldsymbol{x}_K(\boldsymbol{\xi}) = \sum_{i=1}^{3} L_{t,i} \boldsymbol{x}_i^K \tag{24}$$

where $\boldsymbol{x}_i^K$ denotes a coordinate of the i^{th}-vertex of the element (the vertices are numbered in a counter clockwise manner) and the functions $L_{t,i}$ are defined by

$$L_{t,1} = -(\xi + \eta)/2, \quad L_{t,2} = (\xi + \eta)/2, \quad \text{and} \quad L_{t,3} = (1 + \eta)/2. \tag{25}$$

Defining the nodal basis in this way presents an advantage in that elemental matrices, *i.e.* mass and stiffness matrices, can be simply obtained by appropriately scaling the elemental matrices associated with the master elements owing to the mapping having a constant Jacobian. Consequently, the amount of computer memory required and also computational costs in evaluating the elemental matrices are lower than a scheme using the nodal basis constructed directly on the physical elements. It is noted that we use the set of nodal points on the master element given by [24] (as an example, see Figure 4(a) for the nodal set for $p = 5$). Such a nodal set, which has slight differences in the nodal distribution in comparison to that determined by the approach in [14], yields an optimal result in terms of achieving the lowest value of the Lebesgue constant (which is also reflected in the Vandermonde matrix having a small condition number).

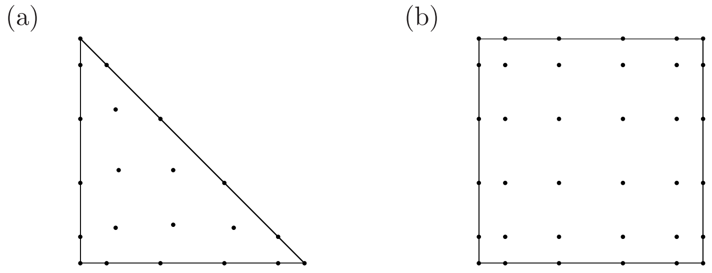


Figure 4: Distribution of interpolation points on the master elements with $p = 5$: (a) triangular element, and (b) rectangular element.

For nodal quadrilateral elements, instead of working with P^p, the approximation space on the master element $I_q = [-1,1]^2$ is selected as $Q^p(I_q) = P^p([-1,1]) \times P^p([-1,1])$, the tensor products of $P^p([-1,1])$, a space of one-dimensional polynomials of degree at most p. Let $\{P_i(x)\}_{i=0,\ldots,p}$ be the normalized Legendre polynomial basis on $[-1,1]$, a two-dimensional orthonormal basis on I_q can then be defined by

$$\psi_{(p+1)j+i+1}(\boldsymbol{\xi}) \equiv P_i(\xi)P_j(\eta), \quad 0 \leq i, j \leq p. \tag{26}$$

The number of basis functions and the order p in this case is related through

$$N_p = (p+1)^2. \tag{27}$$

Note the higher number of degrees of freedom in comparison to elements of P^p-type for a given interpolation order p. A nodal basis $\{\phi_m\}_{m=1,\ldots,N_p}$ is then constructed in an identical way described in the last subsection, provided that a set of nodal points $\{\boldsymbol{\xi}_m \in I_t\}_{m=1,\ldots,N_p}$ is given. Here, we consider the set of interpolation points with a Legendre-Gauss-Lobatto distribution, which is given by

$$\boldsymbol{\xi}_{(p+1)j+i+1} = (x_i, x_j), \quad 0 \leq i, j \leq p, \tag{28}$$

where $\{x_i\}_{i=0,\ldots,p}$ are the zeros of the function $(1-x)^2 d(P_{p-1}(x))/dx$ (a nodal set with the classical two-dimensional Legendre-Gauss distribution was also considered in [13]). Figure 4 depicts the nodal set for $p = 5$. The nodal basis functions on the physical (convex) quadrilateral element K are then defined as $\phi_m(\boldsymbol{x}) = (\phi_m \circ \boldsymbol{x}_K^{-1})(\boldsymbol{x})$ with a bi-linear mapping $\boldsymbol{x}_K : I_q \to K$:

$$\boldsymbol{x}_K = \sum_{i=1}^{4} \boldsymbol{x}_i^K L_{q,i}(\boldsymbol{\xi}) \tag{29}$$

where $\boldsymbol{x}_i^K$ denotes a coordinate of the i^{th}-vertex of K (the vertices are numbered in a counter clockwise manner) and

$$L_{q,1} = (1-\xi)(1-\eta)/4, \; L_{q,2} = (1+\xi)(1-\eta)/4,$$
$$L_{q,3} = (1+\xi)(1+\eta)/4, \text{ and } L_{q,4} = (1-\xi)(1+\eta)/4.$$

Note that except for rectangular and four-sided parallelogram elements, the Jacobian of the mapping (29) is not a constant; as a consequence, the elemental matrices (*i.e.* elemental mass and stiffness matrices) of each element can no longer be obtained by scaling the elemental matrices associated with the master element. While they can be computed accurately and subsequently stored element-by-element, we adopt a less accurate but more memory-economical approach in approximating such matrices [13]. Such an approach, owing to the use of a (fixed order) classical two-dimensional Gauss quadrature, defines the approximate elemental matrices as a multiplication of the matrices defined on the master element and the matrices involved with the coordinate mapping. Although the latter vary element-by-element, they are diagonal matrices, thus significantly reducing the storage required.

2.4. Temporal discretization

The system of ODEs governing the time evolution of the discrete solution for all elements can be written as

$$\mathbf{M}\frac{d\widetilde{\mathbf{u}}_h}{dt} = \mathbf{r}(\widetilde{\mathbf{u}}_h, t) \tag{30}$$

where $\mathbf{M}$ represents the global mass matrix, $\widetilde{\mathbf{u}}_h$ denotes the global vector of the expansion coordinates (modal coordinates for the schemes based on polymorphic bases and nodal coordinates for nodal bases), and $\mathbf{r}(\widetilde{\mathbf{u}}_h, t)$ is the right-hand-side vector which is a contribution of the volume integrals involving the flux terms and the edge integrals involving the numerical flux terms.

The time-dependent system (30) is numerically integrated using an explicit fourth-fifth order Runge-Kutta-Fehlberg (RKF45) method (see a detailed account of this scheme in *e.g.* [25]). RKF45 has a mechanism to automatically select the step size Δt used in the integration to control accuracy of the solution. Concisely, the integrator utilizes the fourth-order Runge-Kutta scheme and the fifth-order Runge-Kutta scheme that uses all values of substages of the fourth-order scheme. It accepts the solution from the fifth order subscheme and adjusts the step size to control the truncation error of the fourth order subscheme. Here, we use the RKF45 subroutine written by Shampine *et al.* [26]. In this subroutine, the accuracy of the solution is controlled by the parameters `relerr` and `abserr`, denoted here as ε_r and ε_a ($\varepsilon_r > \varepsilon_a$). Since we focus on assessing an accuracy of the spatial discretization, the values of these parameter are set to sufficiently small values in order to keep errors from the temporal discretization negligible when compared with the spatial errors.

3. Numerical Experiments on Shallow Water Flow

In this section, we present an application of the polymorphic nodal DG method (PNDG) (or the hybrid modal/nodal DG) and nodal DG method (NDG) to problems governed by nonlinear SWE. We investigate performances of these methods on various element types in terms of accuracy and computational cost. As mentioned earlier, one of our main questions concerns a possibility of using elements of shapes other than triangles to improve the performance of DG schemes. To gain more insight into this aspect, we consider a scenario in which computational meshes are constructed from a given triangular mesh or *vice versa*. Here, we use test problems with an analytic solution in order to facilitate the investigation. More precisely, we use a manufactured smooth solution with a tide-like behavior. Note that a problem with a manufactured solution is obtained by substituting a smooth function into the SWE and solving the equations with the resulting forcing terms and boundary conditions produced by the manufactured solution. Note that the computer code employed in the numerical simulations is mainly written in Fortran 90 and compiled using the Intel® Fortran version 11.1 compiler. We make use of the fortran basic linear algebra subprogram (BLAS) level 2 [27] for matrix-vector multiplications. Numerical computations are conducted on dual six-core 2.4 GHz AMD Opteron model 2431 64 bit 12 GB RAM nodes available at the Center for Research Computing at the University of Notre Dame.

Before proceeding, it is noted that, in the following subsection, we use a broken L_2 norm

$$\|f(\boldsymbol{x})\|_{\Omega_h} = \left(\sum_{K \in \mathcal{T}_h} \int_K f(\boldsymbol{x})^2 d\boldsymbol{x} \right)^{1/2} \tag{31}$$

in measuring the error in the approximate solution. Computing times reported are an average of at least two identical simulations. When required, the total degrees of freedom (DOFs) denoted by N_t is the sum over all elements of the DOFs of each element, *i.e.*

$$N_t = \sum_{K \in \mathcal{T}_h} N_p^K \tag{32}$$

where $\mathcal{T}_h$ denotes a given mesh and N_p^K represents the degrees of freedom of element K for a given interpolation order p (for a system of equations, the above formula is the DOFs for one solution variable). In PNDG, N_p corresponds to the number of modes and in NDG to the number of nodal points. Moreover, in PNDG, the other number determining the computational work is the total number of nodal coordinates which is given by

$$M_t = \sum_{K \in \mathcal{T}_h} M_p^K \tag{33}$$

where M_p^K denotes the number of nodal points of element K used in defining the nodal bases counterpart. Note that the value of M_p depends on p, p_d, and the shape of elements (M_p can be computed using (20); see Table 1 for M_p for the triangular and quadrilateral elements for p up to 6).

Table 1: Degrees of freedom N_p and the number of nodal points M_p per element in different element types.

Order p	PNDG & NDG (Tri)	PNDG (Quad)			LGL-NDG(Quad)
			M_p		
	$M_p = N_p$	N_p	$p_d = 0$	$p_d = 1$	N_p
1	3	3	4	4	4
2	6	6	8	8	9
3	10	10	12	13	16
4	15	15	17	20	25
5	21	21	24	28	36
6	28	28	32	37	49

3.1. Nonlinear Shallow Water problems

In this section, we consider the two-dimensional nonlinear SWE which consist of the depth-average continuity, x-, and y- momentum equations written in conservative form as follows,

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) = \mathbf{s}(\mathbf{q}, \boldsymbol{x}, t) \tag{34}$$

where the vector of the conserved variables $\mathbf{q}$ is

$$\mathbf{q} = \begin{pmatrix} H \\ uH \\ vH \end{pmatrix}, \tag{35}$$

the shallow water flux $\mathbf{F} = (\boldsymbol{f}(\boldsymbol{q}), \boldsymbol{g}(\boldsymbol{q}))$

$$\boldsymbol{f} = \begin{pmatrix} uH \\ u^2 H + \frac{1}{2} g H^2 \\ uvH \end{pmatrix}, \quad \boldsymbol{g} = \begin{pmatrix} vH \\ uvH \\ v^2 H + \frac{1}{2} g H^2 \end{pmatrix}, \tag{36}$$

and the vector of forcing terms $\mathbf{s}(\mathbf{q}, \boldsymbol{x}, t)$

$$\mathbf{s} = \begin{pmatrix} 0 \\ -gH \frac{\partial z_b}{\partial x} + F_x \\ -gH \frac{\partial z_b}{\partial y} + F_y \end{pmatrix}. \tag{37}$$

Here, $H(\boldsymbol{x}, t)$ denotes the total water column height, u and v represent the depth-averaged velocity in the x- and y- directions respectively, g is the magnitude of the gravitational acceleration, $z_b(\boldsymbol{x})$ represents the bottom level measured from some horizontal reference (see figure 5 for an illustration). F_x and F_y in (37) are used in denoting forcing terms in the momentum equations which may be present, *e.g.* Coriolis force, friction losses, surface stresses. Note that, in this study, we consider an effect of the diffusion of momentum from turbulence negligible and the terms describing such an effect are excluded from the equations.

The DG discretization of the system (34) can be obtained by a straightforward generalization of the procedure outlined in section 2.1 for the scalar hyperbolic conservation laws. As in the scalar case, the numerical flux must be provided in order to complete the discretization. Here, we consider the local Lax-Friedrichs (LxF) flux. To define the flux, consider two adjacent elements

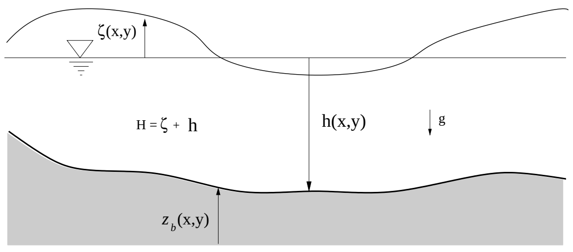


Figure 5: Schematic diagram of the free surface and bathymetry

K^- and K^+ and let e be their common edge. The local Lax-Friedrichs is defined as follows, for $\boldsymbol{x} \in e$

$$\widehat{\mathbf{F}} = \frac{\mathbf{F}(\mathbf{q}_h^-) + \mathbf{F}(\mathbf{q}_h^+)}{2} + \frac{C}{2}\boldsymbol{n}^{\mp}(\mathbf{q}_h^{\mp} - \mathbf{q}_h^{\pm}) \tag{38}$$

where $\mathbf{q}_h^-$ and $\mathbf{q}_h^+$ are respectively the solution value at $\boldsymbol{x}$ of the element K^- and K^+, $\mathbf{n}^- = -\mathbf{n}^+$, and the constant C corresponds to the largest value, along the edge e, of the absolute maximum eigenvalue of the normal flux Jacobian matrix,

$$\max_{s\in[\mathbf{q}_h^-, \mathbf{q}_h^+]} \left| \lambda \left(n_x \frac{\partial \boldsymbol{f}}{\partial \mathbf{q}}\bigg|_s + n_y \frac{\partial \boldsymbol{g}}{\partial \mathbf{q}}\bigg|_s \right) \right| = \max_{s\in[\mathbf{q}_h^+, \mathbf{q}_h^-]} \left[\left(|\boldsymbol{n} \cdot \boldsymbol{u}| + |\sqrt{gH}| \right) \bigg|_s \right] \tag{39}$$

where $\lambda(\cdot)$ denotes the eigenvalue of the matrix. Note that the boundary conditions are enforced weakly by properly specifying an exterior state in the numerical flux along the physical boundaries such that the desirable conditions are obtained in a weak sense.

3.2. Manufactured solutions

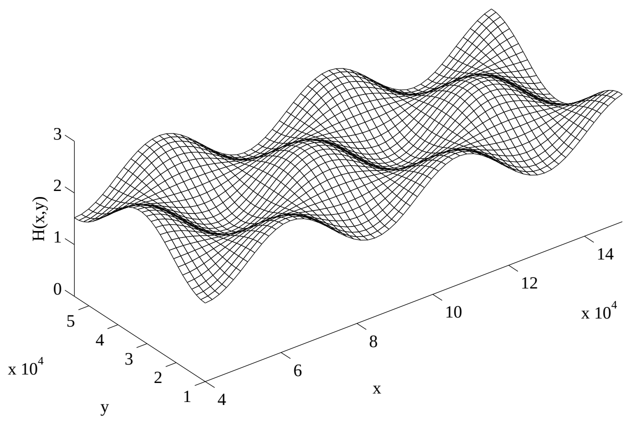
Here, a problem with an exact solution is used as a verification tool for assessing the DG schemes. Specifically, we consider the problem in which the vector of source terms $\mathbf{s}(\mathbf{q}, \boldsymbol{x}, t)$ corresponds to a vector of terms arising from substituting the smooth functions below

$$
\begin{aligned}
H &= 2\xi_0 \frac{\cos(\omega(x-x_1))\cos(\omega(y-y_1))}{\cos(\omega(x_2-x_1))\cos(\omega(y_2-y_1))}\cos(\omega(t+\tau)) + H_0 \\
uH &= \xi_0 \frac{\sin(\omega(x-x_1))\cos(\omega(y-y_1))}{\cos(\omega(x_2-x_1))\cos(\omega(y_2-y_1))}\sin(\omega(t+\tau)) \qquad (40) \\
vH &= \xi_0 \frac{\cos(\omega(x-x_1))\sin(\omega(y-y_1))}{\cos(\omega(x_2-x_1))\cos(\omega(y_2-y_1))}\sin(\omega(t+\tau))
\end{aligned}
$$

into the left hand side of (34). In (40), ω, τ, x_1, x_2, y_1, y_2, ξ_0, and H_0 are positive constants (the value of H_0 is selected sufficiently large so that H is positive everywhere). The exact solution is used to prescribe the initial condition and the boundary conditions. Note that this solution has a vanishing forcing term for the depth-averaged continuity equation.

In all numerical calculations reported below, the values of the parameters appearing in (40) are set to $\omega = 0.0001405$, $\tau = 3456$, $x_1 = 40\times10^3$, $x_2 = 150\times10^3$, $y_1 = 10 \times 10^3$, $y_2 = 55 \times 10^3$, $\xi_0 = 0.25$, and $H_0 = 2$. The simulations are performed through $t_f = 172800$s (a period of the solution is approximately 44720). Figure 6 illustrates the exact solution of the total water column height H and one of the x-directed momentum flow components at this particular time and with the selected parameters. Here, we carry out an h and p convergence study in order to assess the performance of the PNDG and NDG method in terms of accuracy and computational cost. The h convergence study examines how an approximate solution behaves as mesh resolution h is varied while holding the approximation order p unchanged and *vice versa* for the p convergence study. We employ different mesh configurations in order to gain more insight on the performance of the modal bases and nodal bases on different element shapes and how they compare. Here, non-triangular meshes considered are constructed from a given triangular mesh. A construction is carried out in a such way that a non-triangle element and a triangular element occupying a given point in the computational domain have a similar edge length.

(a) H



(a) uH

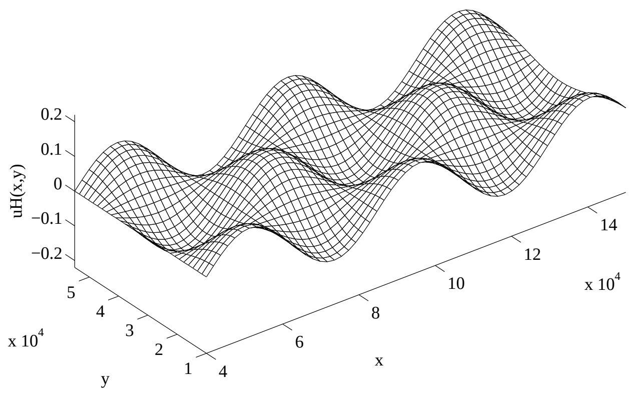


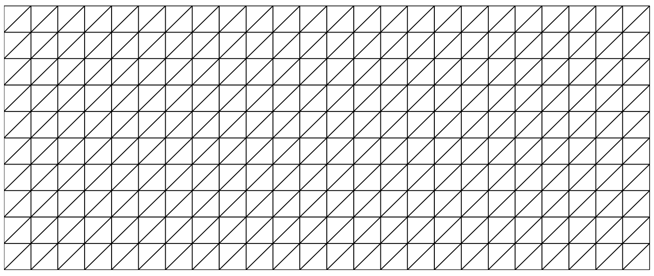
Figure 6: Manufactured solution (40), plotted on 60×30 grid, at $t = 172800$ (2 days); (a) H and (b) uH.

Below, we first present numerical results computed on a so-called regular mesh and subsequently results as computed on an unstructured mesh. In each case, we consider meshes with a range of resolutions and we consider the DG schemes with p ranging from 1 to 5. Here, for the PNDG scheme on quadrilaterals, we use the polymorphic elements with $p_d = 0$ for $p = 1$ and 2 and with $p_d = 1$ for $p = 3$ to 5; for the triangular elements, we use $p_d = 0$ regardless of the order p considered. This choice of p_d stems directly from accuracy and required computational operations consideration of the polymorphic bases. For the NDG scheme on quadrilaterals, we consider the scheme derived using tensor products of 1-D Lagrange polynomial basis defined on the Legendre-Gauss-Lobatto points while using the classical tensor-product Gauss quadrature for the area integration (for more detail see [13]). In all numerical simulations, we set the values of the parameters controlling temporal error in the RKF45 to $(\varepsilon_r, \varepsilon_a) = (5 \times 10^{-7}, 5 \times 10^{-9})$. Note that RKF45 automatically uses varying Δt that is, to some extent, optimal for controlling errors in an approximate solution of ODE(s) being considered, more specifically, in our case, ODEs arising from DG spatial discretization of the SWE.

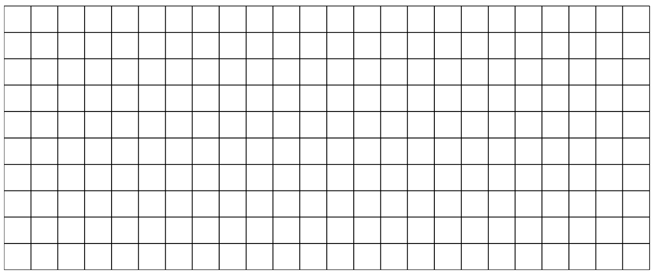
3.3. Solution computed on regular meshes

We first consider three so-called regular mesh configurations, namely, a regular triangular mesh, a rectangular mesh, and a skewed-rectangular mesh. The last configuration refers to a mesh with convex quadrilaterals. In each configuration, four nested meshes are employed in order to examine the h convergence property. In all configurations, the meshes, from the coarsest to the finest resolution, are denoted as h, $h/2$, $h/4$, and $h/8$ respectively. Figure 7 shows the coarsest mesh of each configuration. The coarsest mesh of the first and second configurations are built from a uniform grid of 25×11 points with the node-to-node distance in the x-direction and in the y-direction being $\Delta x = 4583.33$ and $\Delta y = 4500$, respectively. Note that the regular triangular mesh can be viewed as a result of bisecting rectangular elements in the rectangular mesh or *vice versa*, the rectangular mesh can be viewed as a result of merging two adjacent

(a) Triangular mesh, $N_{el} = 480$.



(b) Rectangular mesh, $N_{el} = 240$.



(c) Skewed-rectangular mesh, $N_{el} = 240$.

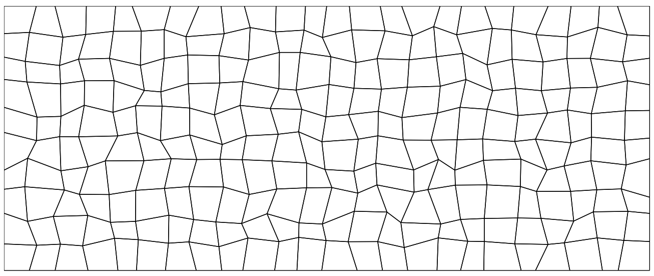


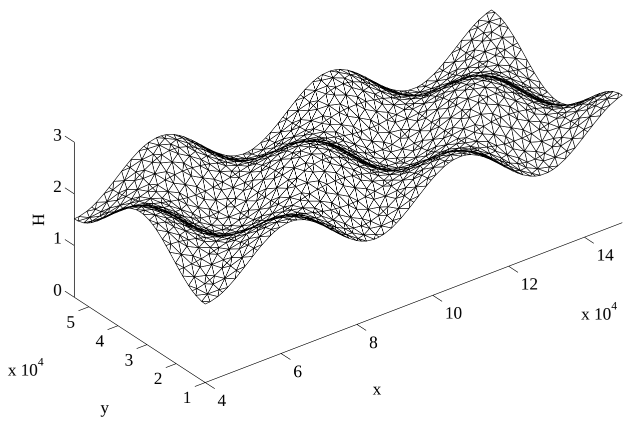
Figure 7: Coarsest regular mesh used in the SWE with a manufactured solution; (a) triangular mesh, (b) rectangular mesh, and (c) skewed-rectangular (quadrilateral) mesh.

triangular elements. For the third configuration, the coarsest mesh is obtained by relocating interior points of the uniform 25×11 grid. Each interior point is relocated in either direction from its original location with a distance varying randomly from 0 to 25% of the node-to-node distance of the uniform grid. The coarsest triangular mesh consists of 480 elements and the coarsest quadrilateral meshes consist of 240 elements. The three finer meshes are obtained by applying successive uniform refinements to the coarsest mesh. The uniform refinement divides each triangle into four similar subtriangles and uniformly divides each rectangle into four subrectangles. The finest triangular and rectangular meshes therefore consist of 30720 elements and 15360 elements, respectively. Note also that for the same resolution, the number of elements in the rectangular mesh is half that of the triangular mesh.

3.3.1. Accuracy

As an example, Figure 8 shows, without smoothing, the approximate total water column height and x-directed water flow at the final simulation time $t_f = 172800$ as computed on the h-rectangular mesh using the PNDG method with $p = 3$ and $p_d = 1$ (the triangles shown in this figure are drawn for plotting purpose only so that the solution at the interior nodes can be visualized). Note the qualitative agreement with the exact solution depicted in Figure 6. Table 2 tabulates the accuracy of the DG schemes through $|\Omega|^{-1/2}\|H - H_h\|_{\Omega_h}$, the normalized L_2 errors of the approximate total water column height H_h. In this table, within every approximation order p, we highlight the error of the scheme yielding the most accurate overall solution and we tabulate the errors of the other schemes as the ratio of the error for a specific scheme relative to the error of the most accurate on the same so-called mesh resolution (the higher the ratio, the less accurate the solution in comparison to that of the scheme highlighted). Evidently, all the DG schemes considered are convergent; the error levels in the approximate solution become smaller either as the order of basis functions increases or as the size of the elements decreases, *i.e.* as discretization resolution increases. It can be observed from the table that, overall, within

(a) $(H)_h$



(b) $(uH)_h$

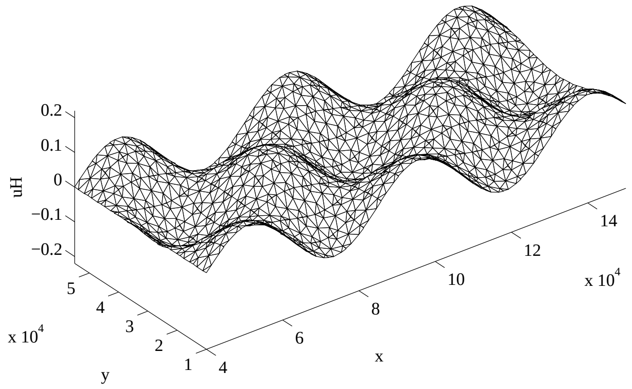


Figure 8: Approximate solution of the SWEs with the manufactured solution at $t_f = 172800$ obtained by using PNDG with $p = 3$ and $p_d = 1$ on the rectangular mesh of resolution h; (a) H_h and (b) $(Hu)_h$.

Table 2: Normalized L_2 errors in the approximate total water column height $\mathcal{E}_H \equiv |\Omega|^{-1/2}\|H-(H)_h\|_{\Omega_h}$ of the overall most accurate scheme for a given interpolation order p and error ratios (specific scheme relative to the most accurate scheme for that p), as computed on regular meshes, and rate of h-convergence. A code [ppmn] denotes the DG method preceding it.

DG Bases & Mesh	p	$\frac{\mathcal{E}_H \text{ in [p}p\text{m}n]}{\mathcal{E}_H \text{ in [p}p\text{m1]}}$ on $\frac{h}{2^q}$-mesh				h-conv. rate
		h	$h/2$	$h/4$	$h/8$	
NDG quad [p1m1]	1	1.61e-02	4.14e-03	1.04e-03	2.61e-04	1.98
		1.00	1.00	1.00	1.00	1.98
NDG skewed-quad [p1m2]	1	1.12	1.12	1.12	1.12	1.99
PNDG quad [p1m3]	1	1.16	1.13	1.12	1.12	2.00
PNDG tri [p1m4]	1	1.14	1.14	1.14	1.14	1.98
NDG tri [p1m5]	1	1.14	1.14	1.14	1.14	1.98
PNDG skewed-quad [p1m6]	1	1.29	1.26	1.25	1.25	2.00
NDG quad [p2m1]	2	3.29e-04	3.96e-05	4.91e-06	6.16e-07	3.02
		1.00	1.00	1.00	1.00	3.02
NDG skewed-quad [p2m2]	2	1.49	1.46	1.44	1.43	3.04
PNDG tri [p2m3]	2	2.12	2.01	1.97	1.97	3.05
NDG tri [p2m4]	2	2.12	2.01	1.97	1.97	3.05
PNDG quad [p2m5]	2	3.52	3.38	3.32	3.28	3.05
PNDG skewed-quad [p2m6]	2	4.59	4.33	4.20	4.13	3.07
NDG quad [p3m1]	3	1.73e-05	1.03e-06	6.43e-08	3.94e-09	4.03
		1.00	1.00	1.00	1.00	4.03
NDG skewed-quad [p3m2]	3	1.72	1.75	1.75	1.74	4.02
PNDG tri [p3m3]	3	3.05	3.26	3.23	3.28	4.00
NDG tri [p3m4]	3	3.05	3.26	3.23	3.28	4.00
PNDG quad [p3m5]	3	6.09	6.50	6.50	6.64	3.99
PNDG skewed-quad [p3m6]	3	7.98	8.99	9.12	9.35	3.96
NDG quad [p4m1]	4	8.14e-07	2.56e-08	8.32e-10	2.77e-11	4.95
		1.00	1.00	1.00	1.00	4.95
NDG skewed-quad [p4m2]	4	2.22	2.14	2.10	2.07	4.98
PNDG tri [p4m3]	4	4.69	4.73	4.62	4.43	4.98
NDG tri [p4m4]	4	4.75	4.84	4.73	4.54	4.97
PNDG quad [p4m5]	4	10.74	10.68	10.33	9.70	5.00
PNDG skewed-quad [p4m6]	4	15.00	17.53	17.61	16.64	4.90
NDG quad [p5m1]	5	4.08e-08	6.37e-10	1.01e-11	9.34e-12*	5.99
		1.00	1.00	1.00	1.00*	5.99
NDG skewed-quad [p5m2]	5	2.63	2.67	2.60	0.93*	6.00
NDG tri [p5m3]	5	7.97	7.92	7.75	0.89*	6.01
PNDG tri [p5m4]	5	8.12	8.03	7.96	0.43*	6.00
PNDG quad [p5m5]	5	23.48	24.31	24.60	1.46*	5.95
PNDG skewed-quad [p5m6]	5	43.48	49.74	50.72	0.93	5.88

* indicates a case in which the temporal errors are not negligible in comparison to the spatial errors.

almost every order p and similar mesh resolution, the approximate solution H_h, ordering from more to less accurate, is that obtained from the following schemes: NDG on rectangles, NDG on skewed rectangles, NDG and PNDG on triangles, PNDG on rectangles, and PNDG on skewed rectangles. Moreover, the error ratios of less accurate schemes to the most accurate scheme are higher as the approximation order p increases, for instance, the error ratio of PNDG on rectangles to NDG on rectangles increases from approximately 1.1 times for $p = 1$ to roughly 24 times for $p = 5$. Notice also that, for an identical triangular mesh and order p, the error levels in the approximate solution H_h obtained from PNDG and NDG are virtually indistinguishable. This can be expected due to the fact that both NDG and PNDG on triangular elements use P^p-type bases defined on triangles. It is evident that, for the same order p, NDG on a rectangular mesh yields a more accurate solution H_h than the PNDG solution on the same rectangular mesh. The same can be said of NDG and PNDG on the same skewed-rectangular mesh. Such a gain in accuracy is attributed mainly to, for a given interpolation order p, the tensor-product basis employed in the NDG scheme on rectangles being able to span additional cross polynomial terms not belonging to the span of polynomial basis employed in PNDG. Furthermore, at the same mesh resolution, NDG on rectangles yields smaller error levels in H_h than the schemes with triangles even though a rectangular element used has an area that is twice that of a triangular element (however, note that both elements have similar edge lengths). This demonstrates to some extent the benefit of the tensor product bases in terms of accuracy. It can be noticed that NDG on skewed rectangles, as expected, degrades the accuracy of H_h when compared to the corresponding NDG solution on rectangles (nonetheless in this case it still produces substantially more accurate solutions than the schemes with triangles). This implies that the milder the skewed rectangles used, the more benefit would be gained from considering the scheme based on the tensor-product bases.

It can be noticed from Table 2 that, for a given order p, the values of error ratios associated with each DG scheme in relation to the most accurate scheme are fairly uniform across all the meshes used. This implies that the error levels from all the schemes reduce at roughly the same rate (dependent on the interpolation order p) when the mesh size is reduced. Indeed, this observation is reflected in the numerical order of convergence of H_h. The order, which refers to the value s from fitting ch^s with c being constant to $|\Omega|^{-1/2}\|H - H_h\|_{\Omega_h}$, is reported in the last column of Table 2. We note that all the DG schemes, regardless of bases or element shapes, exhibit the convergence rate of approximately $O(h^{p+1})$ for the total water column height, in other words $|\Omega|^{-1/2}\|H - H_h\|_{\Omega_h} \sim O(h^{p+1})$ (however note that each scheme has a different value for the constant c). Note that the degradation in the order of convergence for most schemes with $p = 5$ and $h/8$ meshes is due to the fact that the temporal errors from the RKF45, with the specific error tolerance employed, are no longer negligible in comparison to the spatial errors. The convergence rate of the approximate solution is higher than that of the theoretical estimate $O(h^{p+1/2})$ expected for a Lax-Friedrichs DG solution to a problem with nonlinear fluxes [28]. To examine the p-convergence properties, the error levels solved for each mesh resolution are plotted against the order p used on the semi-log scale (error levels on a log scale and p on a linear scale). Figure 9 shows examples of such plots for the h- and $h/2$-meshes. As the curves for all DG schemes appear approximately as straight lines, it can be noted that all DG schemes considered exhibit, with respect to the interpolation order p of the DG bases, the expected exponential convergence rate in the total water column height.

Note that the results of the approximate momentum components uH_h and vH_h are qualitatively similar to that observed in the errors of the total water column height. Overall, the schemes, ranked in order, from more to less accurate with respect to the momentum components, are as follows: NDG on rectangles, NDG on skewed-rectangles, NDG and PNDG on triangles, PNDG on rectangles, and PNDG on skewed rectangles. The convergence rate of uH and vH are between $O(h^p)$ and $O(h^{p+1})$. The convergence of the schemes on rectangles and regular triangles behave somewhat irregularly; the convergence rates of these schemes are typically close to the expected rate $O(h^{p+1/2})$ for even p and close to $O(h^{p+1})$ for odd p. This somewhat irregular behavior appears less pronounced

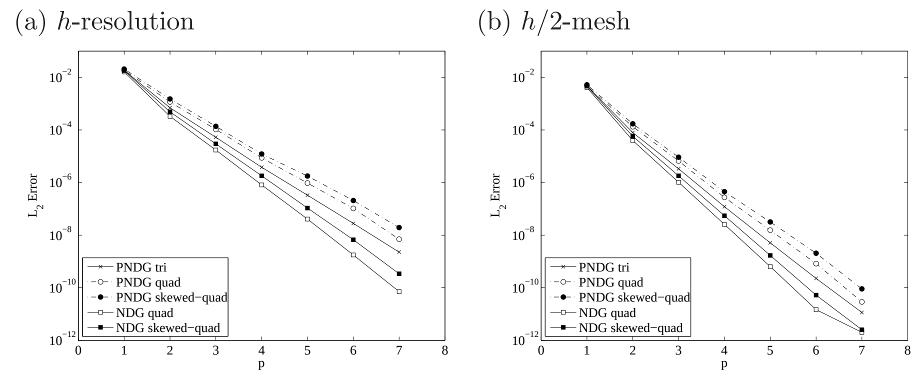


Figure 9: Interpolation order p used in DG schemes versus their normalized L_2-errors $|\Omega|^{-1/2}\|H - H_h\|_{\Omega_h}$ at $t_f = 172800$ as computed on their corresponding (a) h-meshes and (b) $h/2$-meshes.

in the schemes on skewed rectangles and the numerical order of convergence for all the schemes on skewed rectangles is typically close to $p+1$ for both odd and even p.

3.3.2. Computing times

Table 3 tabulates computing times in seconds, denoted as T_c, for the RKF45/ LxF-DG schemes to reach the final simulation time $t_f = 172800$. Note that the computing times reported are an average of three identical simulations (except for the $p = 5$ and $h/8$ combination where they are the result of at least two runs). In this table, within every interpolation order p, the values of the computing times for the scheme requiring the least computing time are listed first and are highlighted in gray boxes. The computing times of the other schemes are subsequently tabulated as the ratio of the computing time for a specific scheme relative to the computing time of the fastest scheme; the higher the ratios, the longer the computing time relative to that of the fastest scheme. Notice that, in every scheme, while holding the mesh resolution unchanged, the computing time required is higher as the interpolation order p of the scheme increases. These increases in computing times stem primarily from the following two reasons. First, the degrees of freedom per element increase as the interpolation order p increases. Table 4, which tabulates a normalized computational cost

Table 3: Computing times T_c (in seconds) of the overall fastest DG scheme for a given interpolation order p and time ratios (specific scheme relative to the fastest scheme for that p), as computed on regular meshes through $t_f = 172800$. A code [ppmn] denotes the DG method preceding it. s denotes the rate of computing times as function of h, *i.e.* $T_c \sim O(h^s)$.

DG Bases& Mesh	p	$\frac{T_c \text{ of } [\text{p}p\text{m}n]}{T_c \text{ of } [\text{p}p\text{m}1]}$ on $\frac{h}{2^q}$-mesh				s-rate
		h	$h/2$	$h/4$	$h/8$	
PNDG quad [p1m1]	1	124	827	6068	41151	-2.80
		1.00	1.00	1.00	1.00	-2.80
PNDG skewed-quad [p1m2]	1	1.02	1.07	0.97	1.01	-2.78
NDG skewed-quad [p1m3]	1	1.43	1.40	1.30	1.32	-2.75
NDG quad [p1m4]	1	1.35	1.43	1.32	1.36	-2.79
PNDG tri [p1m5]	1	2.28	2.79	2.36	2.28	-2.77
NDG tri [p1m6]	1	2.91	2.83	2.62	2.53	-2.73
PNDG skewed-quad [p2m1]	2	238	1755	11550	76385	-2.77
		1.00	1.00	1.00	1.00	-2.77
PNDG quad [p2m2]	2	1.02	0.95	1.06	1.00	-2.78
NDG skewed-quad [p2m3]	2	1.73	1.54	1.48	1.51	-2.70
NDG quad [p2m4]	2	1.78	1.62	1.54	1.53	-2.70
PNDG tri [p2m5]	2	2.00	2.20	2.17	2.10	-2.79
NDG tri [p2m6]	2	3.03	2.65	2.70	2.71	-2.72
PNDG quad [p3m1]	3	411	2727	19791	126896	-2.77
		1.00	1.00	1.00	1.00	-2.77
PNDG skewed-quad [p3m2]	3	0.99	1.06	0.96	1.04	-2.78
NDG skewed-quad [p3m3]	3	1.88	1.81	1.67	1.81	-2.74
NDG quad [p3m4]	3	1.97	1.95	1.79	1.78	-2.71
PNDG tri [p3m5]	3	1.84	2.19	1.96	2.01	-2.79
NDG tri [p3m6]	3	2.80	2.70	2.48	2.58	-2.72
PNDG quad [p4m1]	4	688	4424	28808	202288	-2.73
		1.00	1.00	1.00	1.00	-2.73
PNDG skewed-quad [p4m2]	4	1.03	1.08	1.07	1.24	-2.81
PNDG tri [p4m3]	4	1.71	2.10	2.08	1.93	-2.78
NDG skewed-quad [p4m4]	4	2.09	2.08	2.02	2.22	-2.75
NDG quad [p4m5]	4	2.17	2.17	2.20	1.91	-2.68
NDG tri [p4m6]	4	2.92	2.98	3.06	2.76	-2.71
PNDG quad [p5m1]	5	1070	6905	43926	319869	-2.68
		1.00	1.00	1.00	1.00	-2.68
PNDG skewed-quad [p5m2]	5	1.03	1.09	1.14	1.47	-2.75
PNDG tri [p5m3]	5	1.66	2.00	1.75	2.08	-2.72
NDG skewed-quad [p5m4]	5	2.29	2.32	2.28	2.93	-2.67
NDG quad [p5m5]	5	2.40	2.41	2.42	2.10	-2.69
NDG tri [p5m6]	5	2.75	2.72	2.66	2.52	-2.66

Table 4: Normalized computing times in milliseconds (ms) per one right hand side call per element, $T_r \equiv T_c/(\mathcal{N}_{RHS}N_{el})$, evaluated from the simulations on regular h meshes. Note that $2T_r$ are reported for schemes with triangular elements.

Order p	DG bases & Mesh			
	PNDG tri ($2T_r$)	NDG tri ($2T_r$)	PNDG quad (T_r)	NDG quad (T_r)
1	0.074	0.079	0.045	0.051
2	0.092	0.099	0.059	0.069
3	0.116	0.121	0.076	0.092
4	0.153	0.157	0.107	0.130
5	0.202	0.197	0.143	0.180
6	0.260	0.213	0.177	0.224
7	0.308	0.241	0.215	0.271
8	0.412	0.321	0.289	0.388

per element, clearly indicates an increase in the computational cost per element as the interpolation order p increases. Here, the cost per element is normalized by reporting twice the computing time per element for the schemes with triangular meshes to reflect the mesh setting of interest, namely, the number of elements in a triangular mesh is twice that of its quadrilateral counterpart. Second, the time step size Δt used is smaller as p increases in order to keep the temporal accuracy sufficiently small and, as an explicit time scheme is used, to maintain numerical stability. This is reflected in Table 5 which tabulates, as an example, the number of calls made within the RKF45 integrator to a subroutine calculating the right hand side, denoted as $\mathcal{N}_{RHS}$, of the ODEs (30) in the DG schemes with triangular meshes and rectangular meshes. Note that $\mathcal{N}_{RHS}$ in the PNDG schemes with rectangular meshes are slightly less than that of PNDG schemes with their triangular-mesh counterparts. The same can be said of $\mathcal{N}_{RHS}$ in NDG schemes with rectangular meshes and NDG schemes with their triangular-mesh counterparts. Likewise, while fixing the interpolation order p, the computing times required increases as the mesh resolution increases, *i.e.* as element size decreases. The increasing computing time is the direct consequence of the fact that the number of elements (consequently the total number of degrees of freedom) increases as the mesh is refined. Furthermore, as the mesh size decreases, the time step size Δt used is smaller in order to maintain sufficient temporal accuracy and to ensure the stability (rows in Table 5 reflect this aspect). Computing times T_c, for a fixed p and varying h, behaves like ch^s, where c and s are constant. The numerical rates s are tabulated in the last column of Table 3. We note that the computing times as a function of the mesh size for all interpolation orders p can be approximated by

$$T_c \sim O(h^s), \quad \text{with } s \approx -2.7 \text{ to } -2.8$$

Notice that the differences between the rates s are relatively small and the rates appear to be independent of the interpolation order p. The values of the constant c, as can be expected from the results, vary for the different DG schemes as well as the interpolation order p.

Table 5: The number of calls to a subroutine computing RHS vector, $\mathcal{N}_{RHS}$, required in RKF45/PNDG and RKF45/NDG schemes on regular meshes.

	PNDG tri				NDG tri			
p	h	$h/2$	$h/4$	$h/8$	h	$h/2$	$h/4$	$h/8$
1	15853	25117	39325	61759	19144	30079	47743	76507
2	21565	34807	54193	84075	30475	48547	75679	118123
3	27067	42457	66109	104439	39733	61609	95995	156349
4	32053	50293	78765	130015	53581	84673	132133	207202
5	36823	57937	93001	171510	62455	96691	151249	248853

	PNDG quad				NDG quad			
p	h	$h/2$	$h/4$	$h/8$	h	$h/2$	$h/4$	$h/8$
1	11431	17611	28477	45667	13665	22708	36832	57985
2	17191	27601	42967	66769	25739	39614	60774	93475
3	22405	34927	54367	84859	36515	55881	85801	132615
4	26875	42043	65677	103247	47704	73195	112633	175368
5	31255	48997	76807	122707	59581	91519	141325	221995

When using the same interpolation order p and mesh resolution, we note the following on the relative computing time of various combinations of DG bases and meshes: (i) PNDG on rectangles and PNDG on skewed rectangles require the least amount of computing time compared to other combinations of DG bases and meshes tested; (ii) NDG on rectangles and NDG on skewed rectangles are faster than PNDG on triangles for $p \leq 3$ and become slower for $p > 4$; (iii) NDG on triangles is the slowest scheme among the combinations of DG bases and meshes. Note that, these costs are to be expected since, on quadrilaterals elements, the DOFs per element for PNDG are less than that of NDG for all interpolation orders p (see Table 4 for the numerical computing cost per element), and therefore the PNDG scheme requires less computing time, ranging from 1.4 to 2.4 times, to reach the final simulation times compared to NDG schemes on the same quadrilateral mesh. In addition, we note that, on the same quadrilateral mesh, the number of calls RKF45 made to a subroutine evaluating the RHS vector of the PNDG scheme are also fewer than that of NDG scheme; this results in an additional reduction in computing time for the PNDG scheme in comparison to the NDG scheme. By developing a crude estimate, as outlined in [13], the computational cost required to evaluate one RHS vector for PNDG on a quadrilateral mesh with N_{el} elements and for PNDG (and NDG) on a triangular mesh counterparts are

$$O\left(N_{el}(p+1)(p+2)M_p/2\right) + O\left(4N_{el}M_p(p+1)\right) \tag{41}$$

and

$$O\left(2N_{el}(p+1)^2(p+2)^2/4)\right) + O\left(3N_{el}(p+1)^2(p+2)/2\right), \tag{42}$$

respectively. Note that the first term in these equations corresponds to the cost of an area integration and the second term an edge integration. By comparing these crude estimates, it may be expected that PNDG schemes with quadrilaterals requires less computing time than PNDG or NDG scheme on triangles for all interpolation orders p, provided they roughly have the same $\mathcal{N}_{RHS}$. In brief, a larger cost per element for PNDG on quadrilaterals than that for PNDG or NDG on triangles is expected not to hinder the cost benefit achieved by reducing the number of elements. The computing times for numerical simulations (Table 3) shows this expected behavior (note that PNDG on quadrilaterals are approximately 2 time faster than PNDG on triangles and approximately 2.5 to 3 times faster than NDG on triangles). It is worth noting that $\mathcal{N}_{RHS}$ for PNDG on quadrilaterals and PNDG on triangles are roughly of the same order (see Table 5); therefore, in this case, the lower computing times of PNDG scheme with a quadrilateral mesh is attributed mainly to the reduction in the number of elements. As can be noticed from Table 5, $\mathcal{N}_{RHS}$ for PNDG on quadrilaterals is significantly lower than that of NDG on triangles. This implies that the lower computing times of the PNDG scheme with a quadrilateral mesh also stems from the larger time step sizes Δt employed in the PNDG scheme on quadrilaterals as compared to those in the NDG scheme on triangles. The NDG on quadrilaterals, due to the use of tensor product bases, have higher DOFs per element than that of PNDG and NDG on triangles, more precisely, $2/(1 + 1/(p + 1))$ times the number of DOFs per element. It could therefore be expected that the reduction of the number of elements from using the rectangular mesh might offset the higher cost of using tensor-product bases only up to a certain interpolation order p. A crude estimate made in [13] shows that the cost of evaluating one RHS vector of NDG with a quadrilateral is of

$$O(N_{el}(p + 1)^4) + O(4N_{el}(p + 1)^3) \tag{43}$$

By assuming that all constants in (42) and (43) are of the same order, it is found that the cost in evaluation of RHS vector of the NDG scheme on a quadrilateral mesh is greater than that of NDG or PNDG on a triangular-mesh counterpart for $p > 1$. The results in Table 4 show that the cost of evaluating one RHS vector for NDG on the rectangular mesh is indeed greater than that for NDG on the triangular-mesh counterpart for $p \geq 5$. Note that the point at which NDG on the rectangles becomes more expensive is significantly higher than the estimate. For the range of interpolation orders p used, the cost of calculating one RHS vector for NDG on the rectangular mesh is lower than that of PNDG on the triangular-mesh counterpart; their costs become closer as interpolation order increases. We speculate that the efficiency of memory traffic and cache management are partial reasons explaining why the NDG scheme with the quadrilaterals becomes less efficient (in terms of calculating one RHS vector) at an interpolation order p higher than the estimate. Note that although, for all p values considered, the numerical results in Table 4 show that the cost in evaluating one RHS vector for PNDG on triangles is higher than that for NDG on quadrilaterals, the point at which NDG on quadrilaterals demand more computing times occurs at $p = 4$. This stems from that $\mathcal{N}_{RHS}$ for NDG on quadrilaterals is in fact noticeably higher than $\mathcal{N}_{RHS}$ for PNDG on triangles (see Table 5) .

As can be observed from Table 5, on an identical triangular mesh, $\mathcal{N}_{RHS}$ for the PNDG scheme is significant lower than $\mathcal{N}_{RHS}$ for the NDG scheme. Likewise, the same can be said for $\mathcal{N}_{RHS}$ for the PNDG and NDG scheme on rectangles. This implies that, for the identical tolerance $(\varepsilon_r, \varepsilon_a)$ used, RKF45 automatically selects the larger values of Δt for PNDG and such the larger values have no effect on the spatial discretization errors. Note that we do not observe this behavior, namely Δt for PNDG being larger, in our calculations using the fourth order strong stability-preserving Runge-Kutta (SSPRK4) [29, 30] where the sizes of Δt are estimated based on a Courant-Friedrichs-Lewy (CFL) condition, *i.e.* Δt is guided by the stability criteria instead of an error estimator. In such calculations, the time step size Δt utilized in PNDG and NDG on the same computational mesh are, as expected, virtually identical. Although arguably somewhat of an unfair comparison, we note that, for most cases, RKF45/LxF-DG utilizes less computing times than the SSPRK4/LxF-DG scheme. This suggests a potential benefit for using a time integrator with variable time step based on an error estimate.

3.3.3. Computational cost and accuracy

The critical question when comparing numerical techniques is the computational cost for a specific level of accuracy, or conversely, an error level to be achieved for a given computational cost. Figures 10(a)-(d) show on a log-log scale the accuracy of H_h through L_2 errors versus the computing time required to integrate the problem through the final simulation time $t_f = 172800$. In this figure each curve represents the data computed on the four refined meshes with the interpolation order p being held constant. Figure legends indicate the combinations of DG basis, mesh configuration, and interpolation order p from which the data is obtained. In each figure, we plot the data from the PNDG scheme on triangles for inter-comparison purposes. It can be observed that all

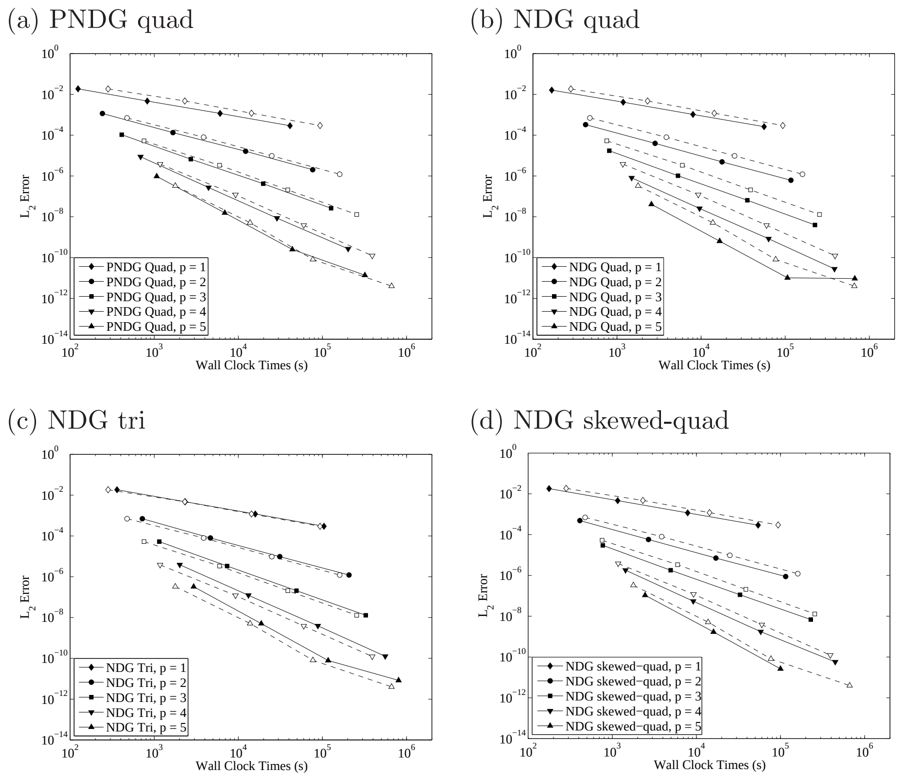


Figure 10: Normalized errors $|\Omega|^{-1/2}\|H - H_h\|_\Omega$ at $t_f = 172800$ *vs.* computing times in seconds of DG solutions on regular grids. Solid lines represent the data of (a) PNDG on rectangles, (b) NDG on rectangles, (c) NDG on triangles, and (d) NDG on skewed-rectangles. Dashed lines in (a)-(d) represent the data of PNDG on triangles: -◇- $p = 1$, -○- $p = 2$, -□- $p = 3$, -▽- $p = 4$, and -△- $p = 5$

the curves appear approximately as straight lines on a log-log scale. Therefore, the computational time as a function of accuracy in the total water column height H_h can be approximated by

$$T_c \sim c_2(\mathcal{E}_H)^{s_2} \tag{44}$$

where c_2 and s_2 are respectively the constant and the rate of the cost function. It can be easily deduced, by recalling that $\mathcal{E}_H \sim O(h^{p+1})$ (see section 3.3.1) and $T \sim O(h^{-2.7})$ (see section 3.3.2), that

$$s_2 \approx -\frac{2.7}{p+1}. \tag{45}$$

The constant pairs (c_2, s_2) for the cost functions associated with the tested DG schemes are tabulated in Table 6. Note that the combination of a low value for c_2

Table 6: Constant and rate (c_2, s_2) in the cost functions $T_c = c_2(\mathcal{E}_H)^{s_2}$, approximating the computing times T_c as a function of error in total water column height $\mathcal{E}_H$, of RKF45/LxF-DG schemes on regular grids.

DG bases & mesh	Cost coefficients (c_2, s_2)				
	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
NDG tri	(1.49,-1.37)	(1.07,-0.89)	(1.40,-0.68)	(2.27,-0.54)	(4.01,-0.44)
PNDG tri	(1.14,-1.40)	(0.65,-0.91)	(0.82,-0.70)	(1.15,-0.56)	(2.15,-0.45)
PNDG skewed-quad	(0.58,-1.39)	(0.69,-0.90)	(0.81,-0.70)	(1.08,-0.57)	(2.25,-0.47)
PNDG quad	(0.47,-1.40)	(0.51,-0.91)	(0.72,-0.69)	(1.16,-0.55)	(2.10,-0.45)
NDG skewed-quad	(0.68,-1.39)	(0.46,-0.89)	(0.63,-0.68)	(0.91,-0.55)	(1.93,-0.45)
NDG quad	(0.51,-1.41)	(0.33,-0.89)	(0.50,-0.67)	(0.76,-0.54)	(1.25,-0.45)

and high value for s_2 indicates high performance in terms of cost for a given level of accuracy. From Figure 10 and Table 6, it can be observed that in terms of cost per accuracy: (i) NDG on rectangles yield the highest performance; (ii) PNDG on rectangles are slightly more efficient than the schemes on triangles; (iii) NDG on triangles are the least efficient among the combinations of DG bases and mesh configurations. To gain more specific insight, we apply the derived cost functions and evaluate the computing time for a specific level of error ε, *i.e.* finding T_c by using (44) with $\mathcal{E}_H = \varepsilon$. Table 7 tabulates the resulting computing times for the various DG schemes with different combinations of DG bases and mesh configurations. Note that the values of ε are selected such that they are well within the range of data considered. In this table, the corresponding computing cost for the given levels of accuracy of PNDG on triangles are highlighted. The computing costs of other combinations of DG bases and mesh configurations

Table 7: Computing times $T_c^{p,\varepsilon}$ required to achieve a specified level of error ε in the total water column height H_h for the PNDG solution with an interpolation order p on regular triangular meshes and the time ratios of $T_c^{p,\varepsilon}$ for specific DG solutions on regular meshes relative to that of the PNDG scheme on triangles. A code [mn] denotes the DG scheme preceding it.

DG bases & mesh	$T_c^{p,\varepsilon}$ of [mn]/$T_c^{p,\varepsilon}$ of [m1]				
	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
	ε = 2.0e-03	ε = 3.0e-05	ε = 9.0e-07	ε = 2.0e-08	ε = 7.0e-09
PNDG tri [m1]	6841	8816	13731	23478	10762
	1.00	1.00	1.00	1.00	1.00
NDG tri [m2]	1.12	1.31	1.32	1.52	1.50
PNDG skewed-quad [m3]	0.48	0.94	1.03	1.18	1.38
PNDG quad [m4]	0.41	0.76	0.81	0.80	0.91
NDG skewed-quad [m5]	0.55	0.55	0.59	0.70	0.78
NDG quad [m6]	0.47	0.41	0.43	0.47	0.53

are reported as a ratio of the computing time for the specific scheme to the computing time for the PNDG on triangles for the same interpolation order p (the higher the ratio, the higher the computational cost required to achieve a specified level of accuracy in comparison to that of the PNDG scheme on triangles). From this table it can be seen that, for the specified levels of accuracy, NDG on rectangles are roughly 2 times more efficient than PNDG on triangles and approximately 3 times more efficients than NDG on triangles. PNDG on rectangles are approximately 1.1 to 1.3 times more efficient than PNDG on triangles (except for $p = 1$ where it is roughly 2.4 times more efficient). The use of skewed rectangle elements, as expected, degrades the performance of the schemes in comparison to use of rectangles elements. Nonetheless, in this numerical experiment, NDG on skewed rectangles still yields higher numerical performance than PNDG on triangles ($\approx$ 1.3 to 1.8 times higher).

The numerical results discussed above and in the previous sections clearly indicate the merit of using tensor product bases on quadrilaterals. For the mesh setting of interest, the NDG schemes based on the nodal tensor product bases are faster than the schemes using either the nodal bases or modal bases on triangles to reach the final time of simulation for low to moderate order p

(NDG on quadrilaterals eventually become slower as the interpolation p used becomes larger). The schemes using the nodal tensor product bases outperform those based on the nodal and modal bases on triangles in terms of accuracy and computational cost per accuracy. Note that, for a given interpolation order p, the nodal tensor-product basis can represent more cross polynomial terms than the bases on triangles; thus it can be expected in general that the approximate solution (for a problem with a smooth solution) from the nodal tensor-product elements would have higher or approximately the same level of accuracy as those from the bases on triangles. This expectation together with the presented numerical results demonstrates that the use of the methods with nodal tensor-product bases is particularly appealing for the low to moderate interpolation order p since a higher efficiency (in comparison to the methods with triangular elements) in terms of cost per accuracy would likely be the case. Note also that although they may not be particularly appealing in terms of cost per accuracy, the schemes based on the polymorphic bases on quadrilaterals show superiority in terms of the computing times required to reach the final times for the mesh setting of interest. This makes the PNDG scheme on quadrilaterals highly appealing in a scenario where the computational time available is limited. In addition, polymorphic bases furnish flexibility in adjusting the accuracy and cost of the discretization.

It is noticed from section 3.3.1 that, for a problem with a smooth solution, a significant gain in accuracy is obtained by raising the interpolation order p of the bases. Such a gain is, as seen in section 3.3.2, at the expense of additional computing time. To further gain insight into the effect of interpolation order p on the computational cost per accuracy, we tabulate in Table 8 the computing times required to achieve a specified level of error ε for each DG scheme using various interpolation orders p. Note that, in each DG scheme, the resulting cost associated with each interpolation order p is shown in italic type when their specified error levels ε fall within the range of raw data, *i.e.* when ε is smaller than $\mathcal{E}_H$ obtained on the coarsest mesh calculation and greater than that on the finest mesh calculation. Results corresponding to cases in which ε is out of the data range are printed in normal type. A numeric value inside parenthesis denotes the cost ratio of the resulting computational cost from using $p-1$ to that from using p. Results shown in this table clearly indicate the appeal of using higher order schemes from the perspective of cost per accuracy. As an example, suppose that an accuracy of 10^{-6} is required, the use of schemes with $p=1$ would require approximately on the order of 3 years of computing time, a prohibitively impractical cost (this corresponds to an expected cost on the serial machine; a dramatically lower computing time can be achieved by utilizing a parallel implementation). By using the schemes with $p = 2$, the computing times required are approximately on the order of 1 to 2 days. Note a reduction of computing time by a factor of roughly a thousand times. The schemes with $p = 3$ requires approximately on the order of 1 to 2 hours of computing time. Note a reduction of cost by a factor of roughly on the order of ten thousand times compared to the schemes with $p = 1$ and approximately on the order of ten times compared to the schemes with $p = 2$. The computing times required reduce further as the interpolation order p increases. Note that the reduction in computational cost stems from the fact that, for higher order schemes, much larger element sizes can be used in order to achieve a similar level of accuracy, in other words, the benefit from the reduction of the number of elements for higher order schemes far outweighs its higher cost per element. See Table 9 for the estimated number of elements required for the specified levels of accuracy (note that they are computed from the functions approximating errors as a function of N_{el}; see section 3.3.1). It is evident from Table 8 that the smaller the specified error level ε, the more pronounced the gain in computational cost per accuracy achieved by raising the interpolation order p of the scheme. Although the computational cost for a given level of accuracy reduces as the interpolation order p increases, the benefit diminishes as indicated by the reduction in the cost ratios inside parentheses shown in Table 8. Arguably, although the scheme with $p = 2$ shows the highest gain in terms of the cost reduction in comparison to the scheme with $p - 1$, using the schemes with $p = 3$ appears to be a more appealing choice due to an evident significant performance gain over using $p = 1$

Table 8: Computing time T_c^ε (in seconds) required to achieve a specified level of error ε in H_h for various DG solutions on regular grids. A numeric value in the parenthesis denotes the ratio between T_c^ε of a DG scheme order $p-1$ and that of p.

DG bases	p	Estimated computing time T_c^ε			
& mesh		ε = 5.0e-03	ε = 1.0e-04	ε = 1.0e-06	ε = 1.0e-08
	1	*1898*	452923	2.85e+08	1.80e+11
	2	82(23.1)	*2934*(154.4)	197228(1446.1)	1.33e+07(13546.9)
PNDG tri	3	33(2.5)	512(5.7)	*12757*(15.5)	317901(41.7)
	4	22(1.5)	200(2.6)	*2629*(4.9)	*34605*(9.2)
	5	24(0.9)	140(1.4)	1133(2.3)	*9154*(3.8)
	1	*2173*	470487	2.64e+08	1.48e+11
	2	121(18.0)	*3950*(119.1)	239892(1101.1)	1.46e+07(10177.3)
NDG tri	3	51(2.3)	736(5.4)	*16901*(14.2)	387849(37.6)
	4	41(1.3)	344(2.1)	*4230*(4.0)	*52042*(7.5)
	5	42(1.0)	235(1.5)	1799(2.4)	*13768*(3.8)
	1	*924*	213139	1.29e+08	7.80e+10
	2	82(11.3)	*2796*(76.2)	178196(723.4)	1.14e+07(6863.8)
PNDG skewed-quad	3	33(2.4)	*521*(5.4)	*13178*(13.5)	333343(34.1)
	4	22(1.5)	211(2.5)	*2953*(4.5)	*41296*(8.1)
	5	27(0.8)	168(1.3)	*1455*(2.0)	*12580*(3.3)
	1	*777*	185355	1.17e+08	7.35e+10
	2	64(12.2)	*2241*(82.7)	148117(787.7)	9789892(7502.1)
PNDG quad	3	28(2.3)	*426*(5.3)	*10367*(14.3)	252525(38.8)
	4	21(1.3)	178(2.4)	*2207*(4.7)	*27334*(9.2)
	5	23(0.9)	133(1.3)	1053(2.1)	*8361*(3.3)
	1	*1048*	236797	1.39e+08	8.25e+10
	2	51(20.5)	*1667* (142.1)	*100492*(1390.9)	6059758(13615.6)
NDG skewed-quad	3	23(2.2)	330(5.0)	*7576*(13.3)	*173840*(34.9)
	4	17(1.4)	149(2.2)	*1895*(4.0)	*24177*(7.2)
	5	21(0.8)	117(1.3)	915(2.1)	*7129*(3.4)
	1	*886*	217372	1.41e+08	9.20e+10
	2	37(23.8)	*1227*(177.1)	*75121*(1882.6)	4598333(20009.8)
NDG quad	3	18(2.1)	247(5.0)	*5487*(13.7)	*121693*(37.8)
	4	13(1.3)	111(2.2)	1336(4.1)	*16131*(7.5)
	5	13(1.0)	78(1.4)	612(2.2)	*4830*(3.3)

while showing moderate gains when compared to the schemes with $p = 2$.

We note that realistic scenarios of coastal flow problems usually involve a number of factors, *e.g.* spatially varying bathymetry, curved boundaries, forms of frictions. The various aspects that need to be taken into account in order to retain high order convergence of high order DG schemes for these flow problems are addressed in [31].

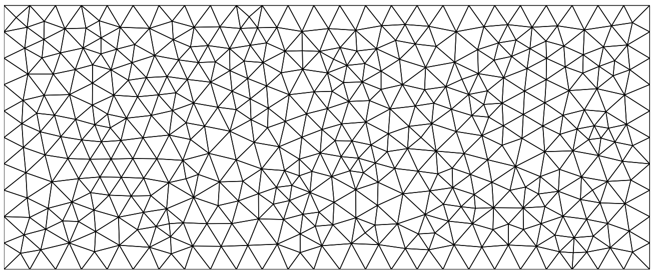
3.4. Solution computed on unstructured meshes

Next we consider DG solutions on unstructured meshes of different configurations, *i.e.* an unstructured triangular mesh, a mixed triangular-quadrilateral mesh, and a polygonal mesh. In all configurations, the approximate solution is computed on a base coarse mesh and three refinements, namely from the coarsest to finest, h, $h/2$, $h/4$, and $h/8$. Figure 11 shows the coarsest mesh of each configuration. The coarsest triangular mesh consists of 802 triangular elements and the lengths of all element edges are at most equal to 4500. The finer triangular meshes are obtained by applying successive regular refinements (see Table 10(a) for the number of triangles in each triangular mesh). With the triangular mesh at hand, we obtain the mixed triangular-quadrilateral mesh and polygonal mesh of the corresponding resolution as follows. The mixed triangular-quadrilateral mesh (referred to the mixed mesh for briefness) is built by simply merging pairs of two adjacent triangles in the original triangular mesh into quadrilaterals. The merging process is conducted in such a way that every resulting quadrilateral element has a determinate Jacobian. In other words, we do not merge two triangles forming a quadrilateral with reflex interior angles (greater or equal to 180°). As can be noticed in Figure 11, the resulting mixed meshes contain triangular elements scattered over the computational domain with moderately skewed-rectangular elements as their neighbors. Table 10(b) lists the number of triangular and quadrilateral elements in each mixed mesh. For the polygonal mesh, a polygonal element is formed by first collecting a set of all triangles sharing a vertex and subsequently connecting a line between the centroids of any two elements in such a set having a common edge. In this way,

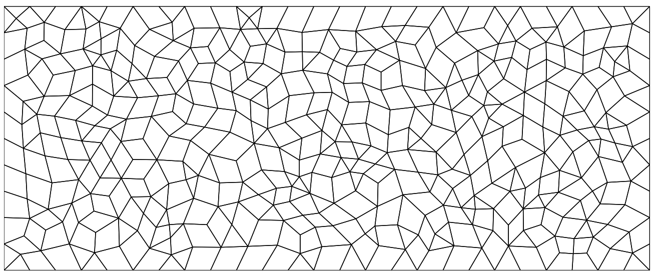
Table 9: Estimated number of elements N_{el}^{ε} required to achieve a specified level of error ε in H_h for various DG solutions on regular grids. A numeric value in the parenthesis denotes the ratio between N_{el}^{ε} of a DG scheme order $p-1$ and that of p.

DG bases	p	Estimated number of elements N_c^{ε}			
& mesh		ε = 5.0e-03	ε = 1.0e-04	ε = 1.0e-06	ε = 1.0e-08
	1	*1798*	92646	9598376	994416075
	2	130(13.9)	*1684*(55.0)	34370(279.27)	701620(1417.3)
PNDG tri	3	49(2.6)	349(4.8)	*3495*(9.8)	35000(20.1)
	4	27(1.8)	129(2.7)	*822*(4.3)	*5233*(6.7)
	5	19(1.4)	72(1.8)	332(2.5)	*1539*(3.4)
	1	*1798*	92646	9598376	994416075
	2	130(13.9)	*1684* (55.0)	34370(279.3)	701620(1417.3)
NDG tri	3	49(2.6)	349(4.8)	*3495* (9.8)	35000(20.1)
	4	27(1.8)	130(2.7)	*827* (4.2)	*5278* (6.6)
	5	19(1.4)	71(1.8)	330(2.5)	*1529* (3.5)
	1	*1000*	50045	5011018	501757514
	2	108(9.22)	*1388*(36.06)	27888(179.68)	560455(895.27)
PNDG skewed-quad	3	39(2.75)	*285*(4.87)	*2918*(9.56)	29904(18.74)
	4	21(1.88)	104(2.75)	*678*(4.30)	*4438*(6.74)
	5	16(1.30)	61(1.69)	*293*(2.31)	*1406*(3.16)
	1	*898*	44881	4487603	448714139
	2	91(9.9)	*1179*(38.1)	24086(186.3)	491961(912.1)
PNDG quad	3	35(2.6)	*247*(4.8)	*2478*(9.7)	24903(19.8)
	4	19(1.8)	90(2.7)	*572*(4.3)	*3611*(6.9)
	5	14(1.4)	50(1.8)	237(2.4)	*1111*(3.3)
	1	*881*	45237	4666633	481410000
	2	52(17.10)	*676*(66.90)	*14007*(333.16)	290151(1659.17)
NDG skewed-quad	3	19(2.75)	131(5.16)	*1292*(10.84)	*12737*(22.78)
	4	10(1.90)	47(2.76)	*302*(4.28)	*1917*(6.64)
	5	7(1.48)	25(1.93)	114(2.64)	*530* (3.62)
	1	*787*	40553	4200530	435094788
	2	39(20.0)	*524*(77.3)	*11076*(379.3)	233911(1860.1)
NDG quad	3	14(2.8)	100(5.3)	*981*(11.29)	*9655*(24.2)
	4	7(2.0)	34(2.9)	220(4.5)	*1412*(6.8)
	5	5(1.5)	18(1.9)	82(2.7)	*383*(3.7)

(a) Triangular mesh, $N_{el} = 802$.



(b) Mixed triangular-quadrilateral mesh, $N_{el} = 479$.



(c) Polygonal mesh, $N_{el} = 437$.

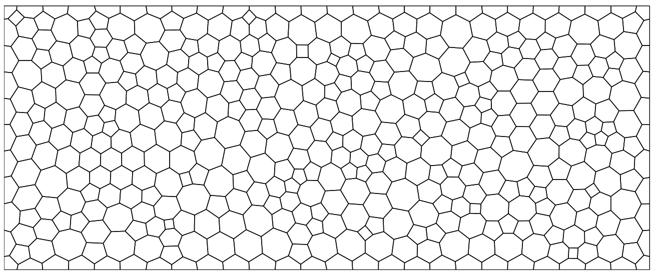


Figure 11: Coarsest unstructured mesh used in the SWE with a manufactured solution; (a) triangular mesh, (b) mixed triangular-quadrilateral mesh, and (c) polygonal mesh.

Table 10: Number of elements categorized by shapes in computational meshes used in DG solution of SWE; (a) triangular meshes, (b) mixed triangular-quadrilateral meshes, and (c) polygonal meshes.

(a) triangular meshes

Mesh res.	Tri
h	802
$h/2$	3208
$h/4$	12812
$h/8$	51328

(b) mixed tri/quad meshes

Mesh res.	tri	quad	Total
h	156	323	479
$h/2$	432	1388	1820
$h/4$	1272	5780	7052
$h/8$	3224	24052	27276

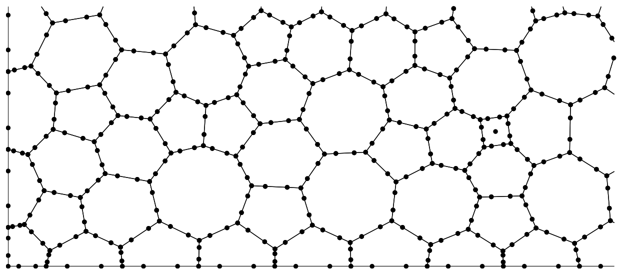
(c) polygonal meshes

Mesh res.	quad	pentagon	hexagon	heptagon	octagon	Total
h	12	151	178	89	7	437
$h/2$	12	221	1346	89	7	1675
$h/4$	12	361	6088	89	7	6557
$h/8$	12	641	25196	89	7	25945

the number of sides of the resulting polygon corresponds to the number of triangles sharing the vertex. The total number of polygons in the resulting mesh thus equals the total number of vertices in the given triangular mesh. Note that any triangulation of a given set of n points yields $2n - 2 - k$ triangles [32] where k is the number of points lying on the boundary of the convex hull of the considered set. Therefore, for a triangular mesh with the number of vertices in the interior far greater than the number of vertices lying on the boundary, the number of elements in the resulting polygonal mesh would be fewer than that in the considered triangular mesh. Table 10(c) tabulates the number of elements classified by shapes in each resulting polygonal mesh. For the same so-called resolution, the total number of elements in the polygonal mesh is less than that of its associated triangular mesh.

For these numerical simulations, the parameters in the RKF45 time integrator are set to $\varepsilon_r = 5 \times 10^{-6}$ and $\varepsilon_a = 5 \times 10^{-9}$. For the PNDG scheme on a mixed mesh, the polymorphic bases with $p_d = 1$ are utilized for quadrilateral elements. For calculations on the polygonal meshes, we consider two strategies in defining a set of nodal points for the polymorphic bases. The first strategy defines the nodal sets by setting $p_d = 1$ in (18). Figure 12(a) shows an enlarged view of the nodal distribution with $p = 3$ on the polygonal h-mesh. Notice that this strategy lead to a scenario in which most nodal points lie on element boundaries. The second strategy, employed by [14] to solve the compressible Navier-Stokes equations, uses a variant approach for the elements whose nodal set constructed by setting $p_d = 0$ contain less than or equal to a single nodal point in the interior. For such elements, we define the set of nodes by adjusting the parameter p_d in (19) so that $r_{\max} = 1$ and in addition $p - (N_{\text{gon}} - p_d) > 0$ for $p \geq 3$. This strategy ensures the existence of interior nodes for all elements (see Figure 12(b)). We find that the second strategy yields noticeably more

(a) $p_d = 1$.



(b) $r_{\max} = 2$

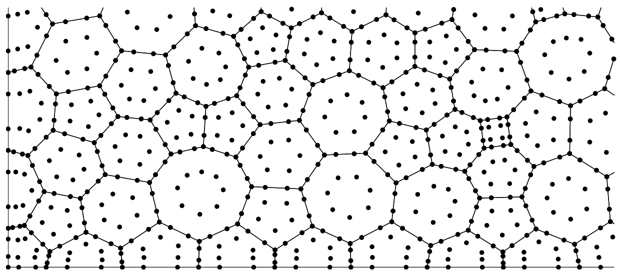


Figure 12: Enlarged view of nodal distribution on the coarsest polygonal mesh with $p = 3$ elements: (a) $p_d = 1$, and (b) $r_{\max} = 1$.

accurate approximate solutions than the first strategy (at least two times more accurate in the L_2-norm for $p \geq 3$). Note that the results associated with the PNDG scheme on polygonal elements that we report below, unless indicated, correspond to those obtained by using the second strategy.

Table 11 tabulates the normalized L_2 errors in the total water column height H_h at the final time of simulation $t_f = 172800$. Within each interpolation order p, we highlight the combination of DG basis and the mesh configuration that overall yields the most accurate solutions. The results for other combinations are tabulated as the ratio of the error from the specific scheme to the error associated with the most accurate scheme. The last column in this table reports the numerical order of convergence of each DG scheme. All DG schemes, regardless of bases or mesh configurations, converge approximately at the rate $O(h^{p+1})$ for the total water column height H_h. The combination of bases and mesh configurations ordered from most to least accurate is as follows: (i) PNDG and NDG on triangles, (ii) NDG on mixed triangles-quadrilaterals, (iii) PNDG on polygons (note that, for $p = 1$, this is the most accurate scheme), and (iv) PNDG on mixed triangles-quadrilaterals. The data from the calculation on the mixed meshes indicates, as expected, the least accurate element type (observed in the regular-grid calculations in section 3.3.1) dictates the error levels of the schemes on mixed meshes. More precisely, it can be observed that the PNDG solution on mixed meshes is less accurate in comparison to the PNDG solution on triangular meshes and the error ratio increases as interpolation order p employed increases; this clearly reflects the effect of using the less accurate quadrilateral polymorphic elements. At similar resolution, the NDG scheme on a mixed mesh, within the range of p tested, yields a less accurate solution than the schemes on triangular meshes; however, their accuracies become closer as the interpolation order p increases (error levels in the NDG solution on a mixed mesh in comparison to the scheme on a triangular mesh reduce from approximately 1.6 times higher at $p = 1$ to approximately 1.1 times higher at $p = 5$). This indicates that the error levels in the NDG solutions on mixed meshes is strongly dictated by the presence of triangular nodal elements.

Table 12 reports computing times in seconds used by the DG schemes on the implemented-mesh configurations. In Table 12, we highlight, within every interpolation order p, the values of computing times associated with the fastest scheme. The computing times from the other schemes are tabulated as the ratios

Table 11: Normalized L_2 errors in the approximate total water column height $\mathcal{E}_H \equiv |\Omega|^{-1/2}\|H - H_h\|_{\Omega_h}$ of the overall most accurate scheme for a given order p and error ratios (specific scheme relative to the most accurate for that p), as computed on unstructured meshes, and rate of h-convergence. A code [ppmn] denotes the DG method preceding it.

DG Bases & Mesh	p	$\frac{\mathcal{E}_H \text{ in [p}p\text{m}n]}{\mathcal{E}_H \text{ in [p}p\text{m1]}}$ on $\frac{h}{2^q}$-mesh				h-conv. rate
		h	$h/2$	$h/4$	$h/8$	
PNDG ngon [p1m1]	1	8.02e-03	1.96e-03	4.90e-04	1.23e-04	2.04
		1.00	1.00	1.00	1.00	2.04
PNDG tri [p1m2]	1	1.23	1.28	1.28	1.28	1.99
NDG tri [p1m3]	1	1.23	1.28	1.28	1.28	1.99
NDG mixed [p1m4]	1	1.51	1.58	1.57	1.56	2.05
PNDG mixed [p1m5]	1	1.64	1.70	1.68	1.68	2.06
PNDG tri [p2m1]	2	2.52e-04	3.01e-05	3.72e-06	4.64e-07	3.03
		1.00	1.00	1.00	1.00	3.03
NDG tri [p2m2]	2	1.00	1.00	1.00	1.00	3.03
NDG mixed [p2m3]	2	1.26	1.31	1.31	1.30	3.10
PNDG ngon [p2m4]	2	1.73	1.71	1.76	1.79	3.06
PNDG mixed [p2m5]	2	2.90	2.90	2.84	2.84	3.13
NDG tri [p3m1]	3	1.42e-05	8.82e-07	5.51e-08	3.44e-09	4.00
		1.00	1.00	1.00	1.00	4.00
PNDG tri [p3m2]	3	1.00	1.00	1.00	1.00	4.00
NDG mixed [p3m3]	3	1.17	1.23	1.21	1.20	4.11
PNDG ngon [p3m4]	3	1.90	1.82	1.89	1.95	4.06
PNDG mixed [p3m5]	3	4.41	4.67	4.59	4.61	4.10
NDG tri [p4m1]	4	7.77e-07	2.43e-08	7.63e-10	2.92e-11	4.91
		1.00	1.00	1.00	1.00	4.91
PNDG tri [p4m2]	4	1.10	1.10	1.09	0.90	5.00
NDG mixed [p4m3]	4	1.12	1.21	1.21	1.00	5.10
PNDG ngon [p4m4]	4	2.47	2.17	2.19	1.89	5.11
PNDG mixed [p4m5]	4	7.01	7.95	7.99	6.56	5.08
NDG tri [p5m1]	5	4.69e-08	7.34e-10	1.21e-11	1.47e-11*	5.96
		1.00	1.00	1.00	1.00*	5.96
PNDG tri [p5m2]	5	1.01	1.00	0.98	0.22*	5.98
NDG mixed [p5m3]	5	1.04	1.13	1.09	0.58*	6.11
PNDG ngon [p5m4]	5	3.64	3.34	3.52	0.46*	6.13
PNDG mixed [p5m5]	5	15.67	17.82	16.44	0.34	6.11

* a case in which the temporal errors are not negligible in comparison to the spatial errors.

Table 12: Computing times T_c (in seconds) of the overall fastest scheme for a given interpolation order p and time ratios (specific scheme relative to the fastest scheme for that p), as computed on unstructured meshes through $t_f = 172800$. A code [ppmn] denotes the DG method preceding it. s denotes the rate of computing times as function of h, *i.e.* $T_c \sim O(h^s)$

DG Bases& Mesh	p	$\frac{T_c \text{ of } [\text{p}p\text{m}n]}{T_c \text{ of } [\text{p}p\text{m}1]}$ on $\frac{h}{2^q}$-mesh				s-rate
		h	$h/2$	$h/4$	$h/8$	
PNDG mixed [p1m1]	1	424	2859	19790	135336	-2.85
		1.00	1.00	1.00	1.00	-2.85
NDG mixed [p1m2]	1	1.38	1.26	1.29	1.35	-2.85
PNDG ngon [p1m3]	1	1.41	1.64	1.38	1.35	-2.78
PNDG tri [p1m4]	1	1.48	1.55	1.48	1.50	-2.77
NDG tri [p1m5]	1	2.10	1.84	2.07	1.97	-2.76
PNDG mixed [p2m1]	2	771	5007	34629	236580	-2.84
		1.00	1.00	1.00	1.00	-2.84
NDG mixed [p2m2]	2	1.47	1.43	1.59	1.55	-2.88
PNDG ngon [p2m3]	2	1.55	1.70	1.49	1.33	-2.72
PNDG tri [p2m4]	2	1.47	1.63	1.66	1.63	-2.81
NDG tri [p2m5]	2	2.10	1.95	2.33	2.17	-2.80
PNDG mixed [p3m1]	3	1275	8351	55112	362117	-2.79
		1.00	1.00	1.00	1.00	-2.79
PNDG tri [p3m2]	3	1.43	1.45	1.60	1.76	-2.82
PNDG ngon [p3m3]	3	1.72	1.84	1.63	1.59	-2.71
NDG mixed [p3m4]	3	1.55	1.69	1.82	1.82	-2.88
NDG tri [p3m5]	3	2.06	1.93	2.20	2.30	-2.79
PNDG mixed [p4m1]	4	2057	13197	90361	667173	-2.86
		1.00	1.00	1.00	1.00	-2.86
PNDG tri [p4m2]	4	1.39	1.38	1.69	1.68	-2.89
PNDG ngon [p4m3]	4	1.69	1.81	1.52	1.40	-2.72
NDG mixed [p4m4]	4	1.73	1.98	1.94	1.75	-2.86
NDG tri [p4m5]	4	2.12	2.10	2.23	2.11	-2.79
PNDG mixed [p5m1]	5	3195	19918	147844	1126169	-2.85
		1.00	1.00	1.00	1.00	-2.85
PNDG tri [p5m2]	5	1.40	1.54	1.72	1.72	-2.91
PNDG ngon [p5m3]	5	1.84	1.83	1.57	0.63	-2.71
NDG mixed [p5m4]	5	1.83	2.09	1.88	1.81	-2.87
NDG tri [p5m5]	5	2.04	2.09	2.04	1.82	-2.76

of the computing time of the specific scheme to that of the fastest scheme. The information in this table suggests that when using the same interpolation order p and similar mesh resolution: (i) PNDG on mixed meshes uses the least amount of computing time compared to other combinations; (ii) NDG on mixed meshes is faster than PNDG on triangular meshes for $p < 3$ and becomes slower for $p \geq 3$; (iii) Computing times required in PNDG on polygonal meshes is in between PNDG on triangular meshes and NDG on mixed meshes; (iv) NDG on triangular meshes is the slowest among the combinations of DG bases and mesh configurations. The reason why the PNDG scheme on triangles runs faster than the NDG scheme on triangles stems mainly from the former scheme requiring noticeably fewer (ranging from 30% to 60%) total number of RHS evaluations in the time integration $\mathcal{N}_{RHS}$ than the latter scheme (see Table 13 for $\mathcal{N}_{RHS}$ of these DG schemes). In other words, for a given tolerance parameter, the RKF45 integrator selects the larger values of Δt for integrating the ODEs arising from the DG spatial discretization with polymorphic triangular elements. We do not observe this behavior when employing the SSPRK4 time integrator with Δt being chosen based on the CFL-type condition. Table 13 indicates that, for similar mesh resolution, $\mathcal{N}_{RHS}$ in the PNDG scheme on a mixed mesh is roughly the same as that in the PNDG scheme on a triangular mesh solution. The same can be said for $\mathcal{N}_{RHS}$ in the NDG scheme for a mixed triangular-quadrilateral element solution and for the NDG scheme for a triangular element solution. This reflects that the time step size used in the mixed mesh calculations is strongly dictated by the triangular elements, whose size are typically smaller than their quadrilateral neighbors.

Figures 13(a)-(d) show, on a log-log scale, the accuracy of H_h through the L_2 error versus the computing times required for the simulation. Each curve in these figures represents data for solution on four meshes with the interpolation order p being fixed. See the legends accompanying the plot for the combinations of DG basis, mesh configuration, and interpolation order with which the curves are associated. Additionally, in each figure, the data from the PNDG scheme on triangles is plotted for comparison purposes. As the curves appear as straight

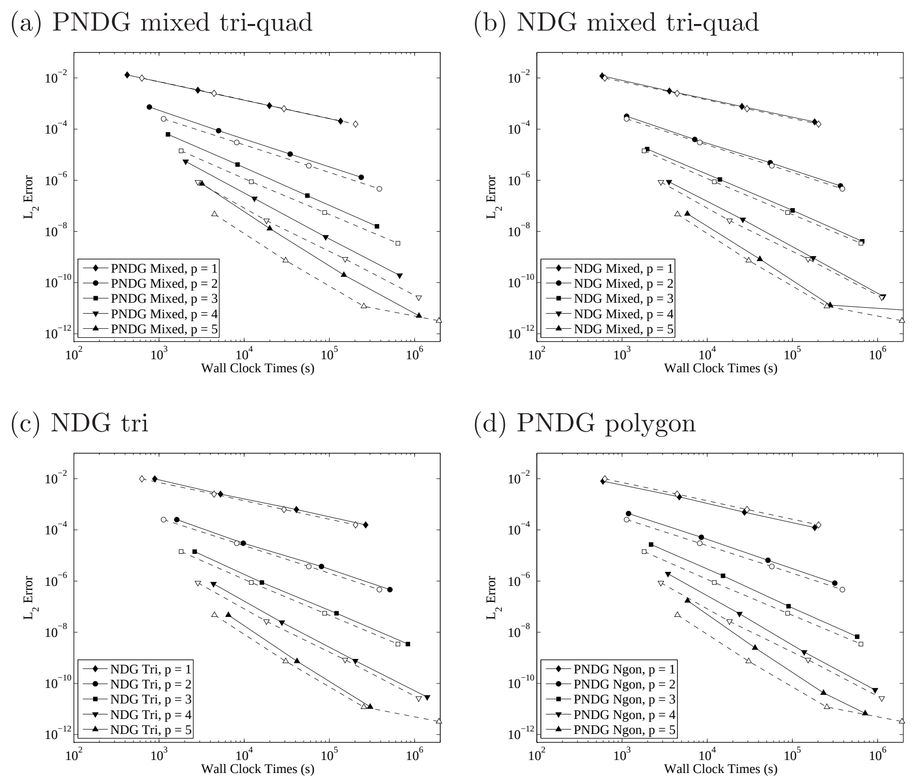


Figure 13: Normalized errors $|\Omega|^{-1/2}\|H - H_h\|_\Omega$ at $t_f = 172800$ *vs.* computing times (in seconds) in DG solutions on unstructured meshes. Solid lines represent the data of (a) PNDG on mixed tri-quad meshes, (b) NDG on mixed tri-quad meshes, (c) NDG on tri meshes, and (d) PNDG on polygon meshes. Dash lines in (a)-(d) represent the data of PNDG on triangles: —◇— $p = 1$, —○— $p = 2$, —□— $p = 3$, —▽— $p = 4$, and —△— $p = 5$

Table 13: The number of calls to a subroutine computing RHS vector, $\mathcal{N}_{RHS}$, required in RKF45/PNDG and RKF45/NDG solutions on unstructured meshes.

	PNDG tri				NDG tri			
p	h	$h/2$	$h/4$	$h/8$	h	$h/2$	$h/4$	$h/8$
1	19885	31591	49615	78086	25333	40345	65725	107825
2	27739	43315	68463	114481	37255	60037	100039	165263
3	34291	53383	85873	162888	49555	81217	136848	227104
4	40825	64569	116478	238871	63259	104149	176867	295447
5	47411	77641	154236	316054	77155	128361	218873	368107

	PNDG mixed tri-quad				NDG mixed tri-quad			
p	h	$h/2$	$h/4$	$h/8$	h	$h/2$	$h/4$	$h/8$
1	19945	30793	47581	73909	25391	39616	63083	102645
2	27589	41965	65233	103365	37975	58744	97429	159448
3	34183	52891	83044	135073	51745	81187	134995	223861
4	40747	63616	102480	185952	66997	104309	175191	298435
5	47257	74840	130566	255856	83179	131042	219151	394405

lines, the cost functions are readily approximated by $T_c = c_2(\mathcal{E}_H)^{s_2}$. Table 14 tabulates the constant pairs (c_2, s_2) for the DG schemes tested. Note that

Table 14: Constant and rate (c_2, s_2) in the cost functions $T_c = c_2(\mathcal{E}_H)^{s_2}$ of various DG solutions on unstructured meshes.

DG bases & mesh	Cost coefficients (a, s)				
	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
PNDG mixed	(1.04,-1.39)	(1.07,-0.91)	(1.75,-0.68)	(2.20,-0.56)	(4.29,-0.47)
PNDG polygon	(0.86,-1.36)	(1.24,-0.89)	(1.94,-0.67)	(3.03,-0.53)	(5.81,-0.44)
NDG mixed	(1.40,-1.37)	(0.73,-0.92)	(1.11,-0.69)	(1.84,-0.55)	(3.30,-0.45)
NDG tri	(1.40,-1.39)	(0.72,-0.93)	(1.06,-0.70)	(1.40,-0.57)	(2.53,-0.46)
PNDG tri	(0.91,-1.41)	(0.43,-0.95)	(0.59,-0.71)	(0.74,-0.58)	(0.95,-0.50)

the computational costs of the DG schemes tested behaves approximately like $O(\mathcal{E}_H^{-2.7/(p+1)})$. To gain more quantitative insight, we evaluate from the cost functions a computing time corresponding to a given level of error ε. The results are tabulated in Table 15. In this table, the results of PNDG on triangular meshes are highlighted in the gray box. The results of the other schemes are

Table 15: Computing times $T_c^{p,\varepsilon}$ required to achieve a specified level of error ε in the total water column height H_h for the PNDG solution with an interpolation order p on unstructured triangular meshes and the time ratios of $T_c^{p,\varepsilon}$ for specific DG solutions on unstructured meshes relative to that of the PNDG scheme on triangles. A code [mn] denotes the DG scheme preceding it.

DG bases & mesh	$T_c^{p,\varepsilon}$ of [mn] / $T_c^{p,\varepsilon}$ of [m1]				
	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
	$\varepsilon = 2.0\text{e-}03$	$\varepsilon = 3.0\text{e-}05$	$\varepsilon = 9.0\text{e-}07$	$\varepsilon = 2.0\text{e-}08$	$\varepsilon = 7.0\text{e-}09$
PNDG tri [m1]	5928	8215	12297	23590	10951
	1.00	1.00	1.00	1.00	1.00
PNDG mixed [m2]	0.98	1.66	1.89	2.02	2.52
PNDG polygon[a] [m3]	0.70	1.61	1.76	1.66	2.17
NDG tri [m4]	1.33	1.33	1.39	1.40	1.40
NDG mixed [m5]	1.16	1.19	1.27	1.30	1.35

[a] Polygon elements with $r_{\max} = 1$.

reported as a ratio of the estimated time of the specific scheme to that of the PNDG scheme on triangles for the same interpolation order p (the higher the ratios, the higher the computational cost required to achieve a specified level of accuracy in comparison to the PNDG on triangles). It is worthwhile to point out that, when compared to DG solutions on regular meshes, the data in Table 7 and 15 clearly indicate that PNDG solutions on unstructured triangular meshes and PNDG solutions on regular triangular meshes exhibit roughly similar levels of performance in terms of cost to achieve a specified level of accuracy (the same can be said in this regard for NDG solutions on unstructured triangular meshes and NDG solutions on regular triangular meshes).

It can be seen from Table 15 that, in terms of cost required to achieve a specified level of accuracy for DG solutions on unstructured meshes, almost all schemes are less efficient than PNDG on triangles. The PNDG scheme on mixed meshes is the least efficient in the cost per accuracy aspect. In comparison to the PNDG scheme on triangles, except for $p = 1$, the PNDG scheme on mixed meshes requires higher computational cost (ranging approximately from 1.7 to 2.5 times higher) to achieve the specified level of accuracy. This implies that the gain in computing time from introducing quadrilateral elements in this PNDG scheme is not enough to offset the loss of accuracy, thus resulting in lower efficiency. Note that for $p = 1$, the PNDG scheme on polygons is approximately 30% more efficients than the PNDG scheme on triangles. The PNDG scheme on polygons becomes more expensive (approximately 1.6 to 2.2 times more expensive) than the PNDG scheme on triangles as the interpolation order p increases. The NDG scheme on mixed elements yields slightly better performance than the NDG scheme on triangular elements for the ranges of interpolation order p tested (approximately 10% higher for $p = 1$ and 2 and 5% higher for $p = 3$ and 4). This implies that the gain in computing times using the NDG scheme on mixed elements in comparison to the NDG scheme on triangles slightly outweighs the small loss in accuracy from using the NDG scheme on mixed elements on a similar mesh resolution. This demonstrates the appeal of using tensor product bases even in this case where mixed meshes are built in a naïve way. Despite yielding almost identical error levels for a given triangular mesh and interpolation order p, the PNDG scheme on triangles performs better than the NDG scheme on triangles in terms of cost per accuracy. This stems from the fact that the former scheme has faster runtimes due to the RKF45 time integrator (automatically) employs larger values of Δt for the ODEs arising from the spatial DG discretization using the polymorphic triangular elements than using the nodal triangular elements. Although not shown here, the PNDG scheme and NDG scheme on triangles show similar performance when the SSPRK4 with the time step sizes Δt based on the CFL-type condition is utilized instead as a time integrator.

To examine the effect of interpolation order p on the cost per accuracy aspect, we tabulate in Table 16 the computing times required to achieve the specified error levels ε for each DG solution on unstructured meshes. Note that, as in Table 8, in each scheme, the estimated runtime associated with each interpolation order p is displayed in italic type for the case where ε falls within the range of the raw data and normal type when extrapolated. Moreover, the numerical value inside parenthesis corresponds to the computational cost ratio of the estimated runtime of the $(p - 1)$ scheme to that of the p scheme. Thus

Table 16: Computing time T_c^ε (in seconds) required to achieve a given level of error ε in H_h of various DG solutions on unstructured grids. A numeric value in the parenthesis denotes the ratio between T^ε of a DG scheme with interpolation order $p-1$ and that of p.

DG bases	p	Computing time T_c^ε			
& mesh		ε = 5.0e-03	ε = 1.0e-04	ε = 1.0e-06	ε = 1.0e-08
	1	*1651*	383839	2.343e+08	1.431e+11
	2	72(23.1)	*2684*(143.0)	*191317*(1224.8)	1.364e+07(10490.6)
PNDG tri	3	28(2.5)	446(6.0)	*11479* (16.7)	*295298*(46.2)
	4	18(1.6)	171(2.6)	2465(4.7)	*35430*(8.3)
	5	15(1.2)	103(1.7)	973(2.5)	*9202*(3.9)
	1	*1625*(0.0)	370764	2.214e+08	1.323e+11
	2	131(12.4)	*4569*(81.1)	298055(743.0)	1.944e+07(6802.9)
PNDG mixed	3	65(2.0)	935(4.9)	*21579*(13.8)	498174(39.0)
	4	43(1.5)	394(2.4)	*5266*(4.1)	*70426*(7.1)
	5	51(0.9)	317(1.2)	2720(1.9)	*23356*(3.0)
	1	*1193*	246429	1.310e+08	6.959e+10
	2	140(8.5)	*4534*(54.3)	*271615*(482.1)	1.627e+07(4276.8)
PNDG polygon	3	67(2.1)	926(4.9)	*20221*(13.4)	*441395*(36.9)
	4	52(1.3)	420(2.2)	*4878*(4.1)	*56685*(7.8)
	5	61(0.9)	343(1.2)	2640(1.8)	*20294*(2.8)
	1	*2202*	503685	3.017e+08	1.807e+11
	2	96(22.8)	*3597*(140.0)	*254810*(1183.8)	1.805e+07(10009.8)
NDG tri	3	42(2.3)	644(5.6)	*15882* (16.0)	*391924*(46.1)
	4	28(1.5)	262(2.5)	3575(4.4)	*48803* (8.0)
	5	30(1.0)	181(1.4)	1537(2.3)	*13027* (3.7)
	1	*1923*	441173	2.651e+08	1.593e+11
	2	85(22.6)	*3208*(137.5)	*230425*(1150.5)	1.655e+07(9626.1)
NDG mixed	3	37(2.3)	577(5.6)	*14538*(15.9)	*366022*(45.2)
	4	29(1.3)	258(2.2)	3423(4.2)	*45326* (8.1)
	5	26(1.1)	166(1.6)	1440(2.4)	*12521* (3.6)

each ratio in parenthesis reflects in the gain in cost efficiency achieved by the increasing the interpolation order by one. The data, which exhibits a similar trend to the DG solutions on regular meshes (see Table 8 for data of the regular-mesh solutions), clearly show the benefit of using the higher order schemes. More precisely, to achieve a specified level of accuracy, the computational cost required for the higher order scheme is considerably less than that required for the scheme with $p = 1$. As an example, for a moderate level of specified accuracy ($\varepsilon = 1.0 \times 10^{-4}$ or 1.0×10^{-6}), the computational costs in the schemes with $p = 3$ are typically three to four orders of magnitude lower than the schemes with $p = 1$. The computational cost for a specified level of error decreases as the interpolation order p used in the scheme increases; however, the benefit gain from raising the interpolation order p diminishes as is clearly indicated by the reduction in the cost ratios (the numeric value inside the parentheses shown in Table 16). Although the scheme with $p = 2$ exhibits the highest cost reduction from the perspective of comparing the cost required in the scheme with p to that required in the scheme with $p - 1$, the use of schemes with $p = 3$ appears, to some extent, to be more appealing in the sense that the scheme yields significant gains in performance over the scheme with $p = 1$ while still showing relatively large gains when compared to the scheme with $p = 2$.

4. Conclusions

In this work, we present a comprehensive assessment of polymorphic nodal discontinuous Galerkin (PNDG) and nodal discontinuous Galerkin (NDG) solutions of the time-dependent nonlinear SWE. The PNDG scheme utilizes two bases for $P^p(K)$ on polygons, namely, the Gram-Schmidt basis and its associated nodal basis, in realizing an efficient spatial DG discretization. The NDG scheme employs the Lagrange nodal basis for $P^p(K)$ on triangles and the tensor-product nodal basis $Q^p(K)$ on quadrilaterals. A problem with a manufactured solution that is intrinsically two-dimensional is used as a test problem. Here, the local Lax-Friedrichs flux is used as the numerical flux in the calculations. The integration in time is carried out using the RKF45 time integrator, which has a mechanism to adjust Δt to control temporal errors. The numerical solutions show that all the methods considered exhibit spectral convergence (in the broken L_2 norm) with respect to the interpolation order p. All the schemes tested exhibit an $O(h^{p+1})$ convergence rate for the water column height. Note that this rate is higher than the theoretical estimated rate $O(h^{p+1/2})$ of a Lax-Friedrichs DG method for nonlinear conservation laws.

In terms of computational cost required to achieve a specified level of accuracy, the numerical results demonstrate clearly the superiority of the higher order schemes. For a specified level of error, the computational cost required decreases as the interpolation order p used in the scheme increases. The benefit gained from employing a one-higher order interpolant however diminishes as the interpolation order p increases. The use of cubic or bi-cubic interpolants ($p = 3$), is particularly appealing due to dramatic improvement in cost as compared to (bi-)linear interpolants and moderate gain over (bi-)quadratic interpolants. Note that, in the test problem employed here, for a moderate specified level of error, the computational cost for the schemes with $p = 3$ are typically about three to four orders of magnitude, in other words a thousand to ten thousand times, lower than the scheme with $p = 1$.

One of the main objectives of this work is to gain more insight on whether elements of shapes other than triangles, in particular quadrilaterals, which reduce the number of elements in the computational mesh would improve the efficiency of DG solutions. We thus consider a mesh setting in which computational meshes of various element shapes are derived from a given triangular mesh. The numerical results, as computed on the regular meshes in which all elements are either quadrilaterals or triangles, demonstrates clearly the merit of using quadrilateral elements, especially those using a nodal tensor-product basis. For a given interpolation order p and a similar mesh resolution, the NDG scheme on quadrilaterals yields a more accurate solution than other schemes (note that the number of elements in the quadrilateral mesh is half that of its triangular mesh counterpart). The higher accuracy of this scheme is attributed mainly to the fact that, for a given interpolation order p, the tensor-product basis spans additional cross polynomial terms not belonging to the polynomial bases used in the other schemes. Moreover, the schemes with nodal tensor-product elements yield the highest performance in terms of cost to achieve a specified level of accuracy among all schemes tested for the range of interpolation orders p considered. For a given level of error, the computational cost required for the NDG scheme on rectangles is approximately 2 times less than that required for the PNDG scheme on triangles; the gain in computational efficiency of the NDG scheme on rectangles over the NDG scheme on triangles is approximately 2.4 to 3 times. The use of skewed-rectangular elements, as expected, degrades the accuracy in solution in comparison to the use of rectangular elements. Nonetheless, even with moderately-skewed rectangular elements, the scheme based on tensor-product bases still yields higher performance in terms of cost per accuracy than the scheme on triangles (approximately 1.3 to 1.8 times higher than the PNDG scheme on regular a triangular mesh).

The numerical experiments on the (conformal) mixed triangular-quadrilateral meshes constructed by naïvely merging two triangles into one quadrilateral shows that the presence of triangular elements (which are scattered all over the computational domain) in the NDG scheme dictates the accuracy of the solution (and analogously the presence of quadrilateral elements dictates the accuracy of the solution for the PNDG scheme). In this case, the NDG scheme exhibits only slightly better performance (only 4% to 10% better) than the NDG scheme on triangles in terms of the computational cost required to achieve a specified level of accuracy. This scheme is approximately 1.2 to 1.3 times less efficient (in terms of cost per accuracy) than the PNDG scheme on triangles. On a triangular mesh, the PNDG scheme has better efficiency than the NDG scheme because of the gain associated with the RKF45 time integration scheme, which selects larger values of Δt for the ODEs arising from the polymorphic triangular elements. Calculations on all quadrilateral and all triangular meshes can be readily used to explain the observed error levels in the NDG scheme on mixed meshes. Results from these calculations, which reveal that the nodal tensor-product elements yield significant higher accuracy than the triangular elements, evidently imply that the smaller triangular elements must be considered in order to have a similar level of accuracy.

The numerical results provide evidence that there may be a substantial benefit of using quadrilateral elements, in particular, those with nodal tensor-product bases (we also believe that the modal tensor-product bases would yield a similar conclusion). To fully gain benefit of the tensor-product elements, we recommend using such elements in areas where solutions varies smoothly and shapes of elements should be kept mildly skewed; in a mixed mesh, a nonconformal mesh with triangular elements having shorter edge lengths than the adjacent quadrilaterals may be considered in order for error to stay uniform in transition regions (alternatively, one may consider using higher order in these areas).

Finally, although they may not be particularly appealing in terms of cost per accuracy, the PNDG scheme on quadrilaterals, for the mesh settings of interest, shows superiority in terms of the computing times (due to its cost in evaluating the RHS vector being intrinsically low) through the final simulation time. This makes the scheme highly appealing in a scenario where the computational time available is limited and the use of a specific mesh is dictated.

5. Acknowledgements

This work was supported by National Science Foundation grants OCI-0749015, OCI-0746232, and DMS-0915118.

Appendix

A. Evaluation of the stiffness matrix

In this section, we describe an approach used in evaluating the stiffness matrices defined in section 2.2. Consider the stiffness matrix associated with the x-directed flux, *i.e.*

$$\boldsymbol{S}_x = (S_{x,(i,j)}),\ S_{x,(i,j)} = \int\limits_K \frac{\partial \phi_i}{\partial x} \phi_j d\boldsymbol{x},\ i = 1, \ldots, M_p,\ j = 1, \ldots, M_p. \tag{46}$$

To compute such a matrix, the partial derivative of ϕ_i with respect to x is written in the nodal representation, more precisely

$$\frac{\partial \phi_i}{\partial x} = \sum_{n=1}^{M_p} D_{x,(n,i)} \phi_n(\boldsymbol{x})$$

where

$$D_{x,(i,j)} \equiv \left. \frac{\partial \phi_j}{\partial x} \right|_{\boldsymbol{x}_i}$$

denotes an entry of the so-called derivative matrix $\boldsymbol{D}_x$. Substituting and manipulating yield the following result

$$\boldsymbol{S}_x = \boldsymbol{D}_x^T \boldsymbol{\mathcal{M}} \tag{47}$$

where $\boldsymbol{\mathcal{M}}$ is a mass matrix (with respect to the nodal basis functions) defined and evaluated as follows

$$\boldsymbol{\mathcal{M}} \equiv \int\limits_K \boldsymbol{\phi}\boldsymbol{\phi}^T d\boldsymbol{x} = J \int\limits_{\mathcal{K}} (\boldsymbol{V}^{-1})^T \widetilde{\boldsymbol{\phi}} \widetilde{\boldsymbol{\phi}}^T \boldsymbol{V}^{-1} d\boldsymbol{\xi} = (\boldsymbol{V}^{-1})^T \boldsymbol{V}^{-1} \tag{48}$$

where $J = (\Delta X)^2$ is the Jacobian of the geometric transformation (10). The remaining task for determining the stiffness matrix is to find the derivative matrix $\boldsymbol{D}_x$. Since $\boldsymbol{V}^T \boldsymbol{\phi} = \widetilde{\boldsymbol{\phi}}$, it follows that $\boldsymbol{V}^T \partial \boldsymbol{\phi}/\partial x = \partial \widetilde{\boldsymbol{\phi}}/\partial x$. The derivative matrix can thus be determined from

$$\boldsymbol{D}_x \boldsymbol{V} = \boldsymbol{\mathcal{D}}_x \tag{49}$$

where $\boldsymbol{\mathcal{D}}_x$ is a matrix with the entries

$$\mathcal{D}_{x,(i,j)} \equiv \left.\frac{\partial \widetilde{\phi}_j}{\partial x}\right|_{\boldsymbol{x}_i}, \quad i = 1, \ldots, M_p,\ j = 1, \ldots, N_p$$

which can be computed easily in practice. It is noted that the stiffness matrix associated with y-directed flux, $\boldsymbol{S}_y = \int_K \partial\boldsymbol{\phi}/\partial y \boldsymbol{\phi}^T \, d\boldsymbol{x}$, can be computed in an analogous way.

References

[1] B. Cockburn, C.-W. Shu, Runge-Kutta discontinuous Galerkin methods for convection-dominated problems, Journal of Scientific Computing 16 (2001) 173–261.

[2] B. Cockburn, Discontinuous Galerkin methods, Zeitschrift fur Angewandte Mathematik und Mechanik 83 (2003) 731–754.

[3] V. Aizinger, C. Dawson, A discontinuous Galerkin method for two-dimensional flow and transport in shallow water, Advances in Water Resoures 25 (2002) 67–84.

[4] C. Eskilsson, S. J. Sherwin, A triangular spectral/hp discontinuous Galerkin method for modelling 2D shallow water equations, International Journal for Numerical Methods in Fluids 45 (2004) 605–623.

[5] E. J. Kubatko, J. J. Westerink, C. Dawson, hp Discontinuous Galerkin methods for advection dominated problems in shallow water flow, Computer Methods in Applied Mechanics and Engineering 196 (2006) 437–451.

[6] E. J. Kubatko, S. Bunya, C. Dawson, J. J. Westerink, C. Mirabito, A performance comparison of continuous and discontinuous finite element shallow water models, Journal of Scientific Computing 40 (2009) 315–339.

[7] S. Bunya, E. J. Kubatko, J. J. Westerink, C. Dawson, A wetting and drying treatment for the Runge-Kutta discontinuous Galerkin solution to the

shallow water equations, Computer Methods in Applied Mechanics and Engineering 198 (2009) 1548–1562.

[8] C. Dawson, E. J. Kubatko, J. J. Westerink, C. Trahana, C. Mirabito, C. Michoskia, N. Pandaa, Discontinuous Galerkin methods for modeling hurricane storm surge, Advances in Water ResourcesDOI 10.1016/j.advwatres.2010.11.004, In press.

[9] T. Kärnä, B. de Brye, O. Gourgue, J. Lambrechts, R. Comblen, V. Legat, E. Deleersnijder, A fully implicit wetting-drying method for DG-FEM shallow water models, with an application to the scheldt estuary, Computer Methods in Applied Mechanics and Engineering 200 (2011) 509–524.

[10] Y. Xing, X. Zhang, C.-W. Shu, Positivity-preserving high order well-balanced discontinuous Galerkin methods for the shallow water equations, Advances in Water Resources 33 (2010) 1476–1493.

[11] F. X. Giraldo, T. Warburton, A high-order triangular discontinuous Galerkin oceanic shallow water model, International Journal for Numerical Methods in Fluids 56 (7) (2008) 899–925.

[12] C. Dawson, J. J. Westerink, J. C. Feyen, D. Pothina, Continuous, discontinuous, and coupled discontinuous-continuous Galerkin finite elements methods for the shallow water equations, International Journal for Numerical Methods in Fluids 52 (2006) 63–88.

[13] D. Wirasaet, S. Tanaka, E. J. Kubatko, J. J. Westerink, C. Dawson, A performance comparison of nodal discontinuous Galerkin methods on triangles and quadrilaterals, International Journal for Numerical Methods in Fluids 64 (2010) 1326–1362.

[14] G. J. Gassner, F. Lörcher, C.-D. Munz, J. S. Hesthaven, Polymorphic nodal elements and their application in discontinuous Galerkin methods, Journal of Computational Physics 228 (2009) 1573–1590.

[15] D. R. Lynch, W. G. Gray, Anaytical solutions for computer flow model testings, Journal of the Hydraulics Division-ASCE 104 (1978) 1409–1428.

[16] P. J. Roache, Verification and Validation in Computational Science and Engineering, Hermosa Publishers, 1998.

[17] R. J. LeVeque, Finite Volume Methods for Hyperbolic Problems, Cambridge University Press, Cambridge, 2002.

[18] J. Qiu, B. C. Khoo, C.-W. Shu, A numerical study of the performance of the Runge-Kutta discontinuous Galerkin method based on different numerical fluxes, Journal of Computational Physics 212 (2006) 540–565.

[19] D. Gottlieb, S. A. Orszag, Numerical Analysis of Spectral Methods: Theory and Applications, SIAM, 1987.

[20] 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.

[21] J. S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Application, Springer Science+Business Media, LLC, 2008.

[22] G. H. Golub, C. F. van Loan, Matrix Computations, 3rd Edition, The Johns Hopkins University Press, 1996.

[23] A. Quarteroni, R. Sacco, F. Saleri, Numerical Mathematics, Springer, 2000.

[24] J. S. Hesthaven, From electrostatics to almost optimal nodal sets for polynomial in simplex, SIAM Journal on Numerical Analysis 35 (1998) 655–676.

[25] J. C. Butcher, Numerical Methods for Ordinary Differential Equations, John Wiley & Sons, 2003.

[26] L. F. Shampine, H. A. Watts, S. M. Davenport, Solving nonstiff ordinary differential equations–state of art, SIAM Review 18 (1976) 376–411, see also URL `http://www.netlib.org/fmm/rkf45.f`.

[27] J. J. Dongarra, J. D. Croz, S. H. Hammarling, R. J. Hanson, An extended set of Fortran basic linear algebra subprograms, Technical Memorandum 41, Argonne National Laboratory, Argonne, IL, see also URL `http://www.netlib.org/blas` (September 1998).

[28] Q. Zhang, C. Shu, Error estimates to smooth solutions of Runge-Kutta discontinuous Galerkin methods for scalar conservative laws, SIAM Journal on Numerical Analysis 42 (2004) 641–666.

[29] S. Gottlieb, C. Shu, E. Tadmor, Strong stability-preserving high-order time discretization methods, SIAM Review 43 (2001) 89–112.

[30] E. J. Kubatko, J. J. Westerink, C. Dawson, Semi discrete discontinuous Galerkin methods and stage-exceeding-order, strong-stability-preserving Runge-Kutta time discretizations, Journal Computational Physics 228 (2007) 832–848.

[31] E. J. Kubatko, C. Dawson, D. Wirasaet, Requirements for achieving high-order accuracy for discontinuous Galerkin solutions of the shallow water equations, *In preparation* (2011).

[32] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, Computational Geometry: Algorithms and Applications, 3rd Edition, 2010.