Comput. Methods Appl. Mech. Engrg. 198 (2009) 1766–1774

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

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

Dynamic *p*-adaptive Runge–Kutta discontinuous Galerkin methods for the shallow water equations

Ethan J. Kubatko [a], Shintaro Bunya [b], Clint Dawson [c,*], Joannes J. Westerink [d]

[a] Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, Columbus, OH 43210, United States
[b] Department of Quantum Engineering and Systems Science, The University of Tokyo, 7-3-1 Hongo, Bunkyo, Tokyo 113-8656, Japan
[c] Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712, United States
[d] Department of Civil Engineering and Geological Sciences, University of Notre Dame, Notre Dame, IN 46556, United States

ARTICLE INFO

Article history:
Received 1 March 2008
Received in revised form 11 October 2008
Accepted 2 January 2009
Available online 21 January 2009

Keywords:
Discontinuous Galerkin
Adaptivity
Shallow water equations

ABSTRACT

In this paper, dynamic *p*-adaptive Runge–Kutta discontinuous Galerkin (RKDG) methods for the two-dimensional shallow water equations (SWE) are investigated. The *p*-adaptive algorithm that is implemented dynamically adjusts the order of the elements of an unstructured triangular grid based on a simple measure of the local flow properties of the numerical solution. Time discretization is accomplished using optimal strong-stability-preserving (SSP) RK methods. The methods are tested on two idealized problems of coastal ocean modeling interest with complex bathymetry – namely, the idealization of a continental shelf break and a coastal inlet. Numerical results indicate the stability, robustness, and accuracy of the algorithm, and it is shown that the use of dynamic *p*-adaptive grids offers savings in CPU time relative to grids with elements of a fixed order *p* that use either local *h*-refinement or global *p*-refinement to adequately resolve the solution while offering comparable accuracy.

1. Introduction

In many coastal ocean modeling applications, a large-domain strategy is employed in which the open ocean boundary is placed in deep waters far from the area of detailed interest, such as a coastal inlet or estuarine system; see, for example, [2,16–18]. Such an approach facilitates simple open boundary condition specification and has been shown to minimize the influence of boundary conditions on the numerical solution in the coastal region [2]. Typically a large-domain, unstructured computational grid will consist of larger elements in the relatively quiescent waters of the deep ocean and then gradually transition to smaller elements in the region of the continental shelf and in the shallower coastal area; see, for example, Fig. 1. In the case of structured grids, a common approach is the use of nested grids (see, e.g., [13]), which, similarly, will transition from coarse structured grids in the deep ocean to finer structured grids in the coastal region.

The size of the elements of an unstructured computational grid in the coastal region will initially be dictated by the geometric complexity of the domain. The grid spacing must be made sufficiently small as to give adequate representation of important geometric features such as intricate coast lines, bathymetry and topography, or man made structures such as jetties or weirs. Typically after an initial "geometric design" of a computational grid, *h* (grid size) refinement, and/or *p* (polynomial order) refinement or enrichment, in the case of *hp* finite element methods, is added locally (by the user) to provide higher spatial resolution in areas that are known, assumed, or shown to require additional resolution due to local flow properties. The design of a large-domain computational grid that can adequately resolve the flow features of a given problem, without becoming excessively expensive in terms of computational resources, is often a time-consuming, iterative process. Additionally, given the temporal nature of the solutions, the use of a fixed, static-in-time computational grid is inefficient, in that any portion of the domain that develops a complex flow structure that requires additional resolution at *any* time must be adequately resolved with the computational grid for *all* time.

The use of dynamic *h*- and/or *p*-adaptive procedures offers a very natural and convenient solution to these problems. Starting with a computational grid that has been designed to adequately resolve the geometry of a given domain, adaptivity will automatically adjust the elements locally, based on either error estimates or some local characteristic of the solution such as the gradient, in order to provide the necessary spatial resolution to accurately capture the flow field as it evolves. For *hp*-adaptive algorithms, *p*-refinement is generally used in areas where the solution is smooth, while *h*-refinement is applied in areas where the solution develops discontinuities or in areas of singularities.

For many practical coastal ocean modeling applications, the solutions, while developing steep gradients, will remain smooth in large portions of the domain (the development of discontinuities

* Corresponding author.
E-mail address: clint@ices.utexas.edu (C. Dawson).

0045-7825/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2009.01.007

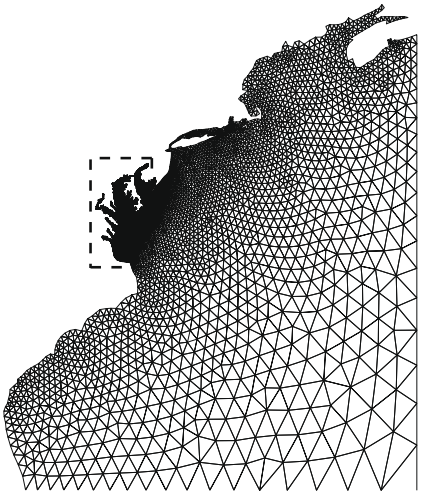

Fig. 1. An example of a large-domain unstructured grid used to model flow in the Chesapeake Bay area (boxed area). Most of the resolution is provided along the shallower coastal areas and, in particular, in the area of Chesapeake Bay.

in the form of either tidal bores or hydraulic jumps occur mostly in problems of special interest such as dam break scenarios or constricting channels). This situation then would favor the use of adaptive *p*-refinement over traditional *h*-refinement. Kubatko et al. [9] recently demonstrated the advantages of using *p*-refinement instead of *h*-refinement, for smooth advection dominated flows in coastal regions using DG methods. For a series of idealized harbor and inlet problems with smooth solutions, it was demonstrated that accurate numerical solutions could be obtained most efficiently by using relatively coarse computational grids consisting of higher-order elements. In that work, global *p*-refinement was applied, that is every element in the domain was refined to a fixed order *p*. In fact, all *hp* finite element methods for the two-dimensional shallow water equations (SWE) that have appeared in the literature have used this approach, i.e. they have utilized local *h*-refinement only, while using a globally defined *p* that is chosen prior to the computations (see, e.g., [6,7,9,11]). Such an approach using DG methods however, while shown to be more efficient than uniform *h*-refinement, does not take full advantage of the discontinuous nature of DG solutions, which allows for easy implementation of local *p*-refinement without the hindrance of maintaining C^0 continuity of the solution between elements as in continuous Galerkin (CG) methods. In the context of the large-domain strategy, local *p*-refinement is especially useful, and applying it adaptively, of course, is a natural choice given the time-varying nature of the solutions.

To this end, in this paper, we investigate a dynamic *p*-adaptive DG method for the two-dimensional SWE. While previous authors have investigated *h*-adaptive techniques using DG methods for the SWE [1,14], there has been no previous work in the area of *p*-adaptation for the SWE. The adaptive algorithm that is employed here makes use of a "sensor", which was proposed by Burbeau and Sagaut [3] in their work on *p*-adaptive DG methods for the Navier–Stokes equations, that can be used as a simple and computationally efficient means to indicate where to adjust the order of the elements of the computational grid locally. In regions where the solution is relatively uniform, a lower-order polynomial is used, otherwise a higher-order polynomial of a specified degree is used. The algorithm is easily implemented using the hierarchical basis proposed by Dubiner [5] at little additional computational cost. Although the employed adaptive algorithm is developed in a heuristic manner, it is shown to successfully reduce error levels and provide sharper resolution of key features of the flow field. Furthermore, use of the dynamic *p*-adaptive algorithm greatly simplifies the initial design of computational grids and is shown to provide savings in CPU time when compared to computational grids with elements of a fixed order *p* that use local *h*-refinement or global *p*-refinement to adequately resolve the solution.

This paper is organized as follows: In the next section, we present the system of governing equations – the two-dimensional SWE. In Section 3, we outline our approach for solving this system of equations, which uses a *p*-adaptive DG method in space and a class of optimal strong-stability-preserving (SSP) Runge–Kutta methods in time. After formulating the basic DG method, we describe the *p*-adaptive algorithm that is applied. This is followed by a presentation of the optimal SSP Runge–Kutta methods that are used to discretize the semi-discrete equations in time. Numerical results are then presented in Section 4 that demonstrate the computational advantages of using such an approach by applying the method to two idealized problems of coastal ocean modeling interest. Finally, in Section 5, we make some concluding remarks and discuss future work using adaptive DG methods for the SWE.

2. Governing equations

The SWE are applicable in describing free surface flow problems where the depth of a water body is sufficiently small compared with its horizontal flow scales. The two-dimensional SWE consist of the depth-integrated continuity equation and the equations for the balance of momentum in the horizontal, *x* and *y*, directions. The equations can be expressed in the form:

$$\partial_t \zeta + \partial_x(Hu) + \partial_y(Hv) = 0, \tag{1}$$

$$\partial_t(Hu) + \partial_x\left(Hu^2 + \frac{1}{2}g(H^2 - h^2)\right) + \partial_y(Huv) = g\zeta\partial_x h + F_x, \tag{2}$$

$$\partial_t(Hv) + \partial_x(Huv) + \partial_y\left(Hv^2 + \frac{1}{2}g(H^2 - h^2)\right) = g\zeta\partial_y h + F_y, \tag{3}$$

where ζ is the elevation of the free surface measured from the geoid (positive upward), h is the bathymetric depth measured from the geoid (positive downward), $H = \zeta + h$ is the total height of the water column (see Fig. 2), g is the gravitational constant, u and v are the depth-averaged velocities in the x and y directions, respectively, and F_x and F_y represent any additional terms that may be present due to Coriolis force, bottom friction, surface stresses such as wind or wave radiation stresses, etc. In particular, for the bottom friction we use:

$$F_x = -\tau H u + \cdots \tag{4}$$

$$F_y = -\tau H v + \cdots \quad \text{with } \tau = C_f \frac{\sqrt{u^2 + v^2}}{H}, \tag{5}$$

where C_f is the bottom drag coefficient (we take $C_f = 0.003$ here). Eqs. (1)–(3) are supplemented with suitable initial and boundary conditions.

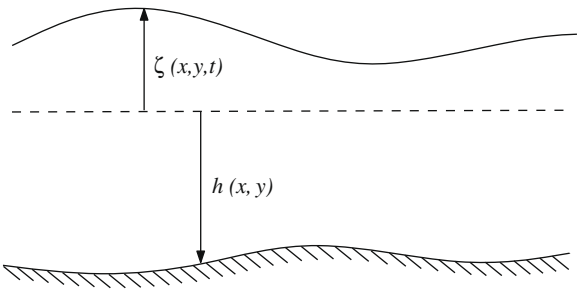

Fig. 2. Definition of the free surface elevation ζ and the bathymetric depth h relative to a geoidal surface.

Defining the following vectors:

$$\mathbf{w} = \begin{bmatrix} \zeta \\ Hu \\ Hv \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} 0 \\ g\zeta\partial_x h + F_x \\ g\zeta\partial_y h + F_y \end{bmatrix}, \tag{6}$$

and the flux function matrix:

$$\mathbf{F} = \begin{bmatrix} Hu & Hv \\ Hu^2 + \frac{1}{2}g(H^2 - h^2) & Huv \\ Huv & Hv^2 + \frac{1}{2}g(H^2 - h^2) \end{bmatrix}. \tag{7}$$

Eqs. (1)–(3) can be written in the compact form:

$$\partial_t w^{(i)} + \nabla \cdot \mathbf{F}^{(i)}(\mathbf{w}) = s^{(i)}(\mathbf{w}), \quad i = 1, 2, 3, \tag{8}$$

where $w^{(i)}$, $\mathbf{F}^{(i)}$, and $s^{(i)}$ are the i-th row entries of $\mathbf{w}$, $\mathbf{F}$, and $\mathbf{s}$, respectively.

3. Numerical method

3.1. Weak formulation and DG spatial discretization

Given a domain $\Omega \subset \mathbb{R}^2$ with a piecewise smooth boundary $\partial\Omega$, let T_h define a finite element partition of Ω into a set of non-overlapping subdomains Ω_e, which do not cross $\partial\Omega$. Let $\partial\Omega_e$ denote the boundary of Ω_e, and let $\mathbf{n}$ denote the unit outward normal of $\partial\Omega_e$.

The SWE can be put into a weak form by multiplying each equation by a sufficiently smooth test function v, integrating over Ω_e, and integrating the divergence term by parts:

$$(\partial_t w^{(i)}, v)_{\Omega_e} - (\nabla v, \mathbf{F}^{(i)})_{\Omega_e} + (\mathbf{F}^{(i)} \cdot \mathbf{n}, v)_{\partial\Omega_e} = (s^{(i)}, v)_{\Omega_e} \quad \forall \Omega_e, \tag{9}$$

where $i = 1$, 2, or 3 and $(\cdot,\cdot)_D$ denotes the $L^2(D)$ inner product.

A discrete weak formulation of the problem is obtained by approximating the solution components $w^{(i)}$ by $w_h^{(i)}$, which belong to the space of functions V_h defined by:

$$V_h = \left\{ v \in L^2(\Omega) : v|_{\Omega_e} \in P^k(\Omega_e) \ \forall \Omega_e \in T_h \right\}, \tag{10}$$

where P^k denotes the space of polynomials of degree k defined on Ω_e, and by choosing a test function $v_h \in V_h$. It is noted that there is no C^0 continuity requirement across element edges in the space V_h and, thus, k can vary from element to element. It is this fact that allows for easy implementation of local p-refinement within the framework of the DG method.

With this space of functions defined, the discrete weak formulation is:

$$(\partial_t w_h^{(i)}, v_h)_{\Omega_e} - (\nabla v_h, \mathbf{F}^{(i)})_{\Omega_e} + (\widehat{f}_n^{(i)}, v_h^{\text{int}})_{\partial\Omega_e} = (s^{(i)}, v_h)_{\Omega_e} \quad \forall v_h \in V_h, \tag{11}$$

where again $i = 1$, 2, or 3 and v_h^{int} denotes the trace of the test function from the interior of Ω_e. Note also that the quantity $\mathbf{F}^{(i)} \cdot \mathbf{n}$ in the boundary integral has been replaced by a so-called numerical flux $\widehat{f}_n^{(i)}$ (see, e.g., [4]).

The specific details of the algorithm that we have implemented for the solution of Eq. (11) using the DG method can be in found in [9]. In particular, we note that we use an orthogonal, hierarchical basis for the space V_h, quadrature rules that are optimal (or in some cases near optimal) for the triangle to evaluate the area integrals, and the Roe or local Lax Friedrich method for the approximation of the boundary fluxes. In this paper, we limit ourselves to spatial approximations P^k with $k \leqslant 3$, although in previous work we have tested spatial approximations of $k \leqslant 7$ with good results [9].

Moving the second and third terms of Eq. (11) to the right-hand-side and inverting the (diagonal) mass matrix, the semi-discrete equations can be written in the compact form:

$$\frac{d}{dt}\mathbf{w}_h = \mathbf{L}_h(\mathbf{w}_h), \tag{12}$$

where $\mathbf{w}_h$ is a vector containing the degrees of freedom of the components, $w_h^{(i)}$ for each element and $\mathbf{L}_h$ is the DG spatial approximation.

3.2. A p-adaptive algorithm

The present section describes a simple and computationally efficient p-adaptive algorithm that can be easily implemented within the framework of the DG method. The algorithm makes use of a "sensor" originally proposed in [3] where a dynamic p-adaptive method was developed and applied to the solution of the Navier–Stokes equations. Our proposed adaptive algorithm makes use of this sensor as an indicator of where we apply p-refinement, but includes some important modifications of the algorithm proposed in [3], which we have found, through numerical experiments, improve the results. In what follows, we assume that the solutions are smooth. In the case of discontinuous solutions, the scheme could be extended along the lines of what is described in [3].

The algorithm can be described as follows: First, upper and lower limits of p approximation, P^L and P^H are defined, where the superscripts L and H denote low and high (relatively speaking) p approximations, respectively (i.e. $H > L$). Initially we set $P^k(\Omega_e) = P^L$ for all Ω_e (we note that for all the problems considered here we use zero initial conditions, i.e. the water starts from rest, so there is no error in the interpolation of the initial condition regardless of the value of P^L). At the end of each time step, a sensor is computed as an indicator of where to apply p-refinement. The sensor is defined by:

$$\Theta_j^{(i)} = \left| \frac{w_h^{(i)}(m_j) - w_h^{(i)}(c)}{d_j} \right|, \tag{13}$$

where $w_h^{(i)}(m_j)$ and $w_h^{(i)}(c)$ are the values of the i-th solution component evaluated at the midpoint m_j of edge j, and the barycenter c of the given element, respectively, and d_j is the distance between m_j and c (see Fig. 3). The sensor is computed for the edges of each element for each solution component. If at some time t one or more of the $\Theta_j^{(i)}$ for a given element exceed a tolerance ϵ (specified by the user), then for the next time step the p approximation on that element for all the solution components $w_h^{(i)}$ is increased by one; otherwise the P^L approximation is still used.

Once an element is refined, by which we mean p-refined, it is either further refined or unrefined at subsequent time steps according to the following rules: (1) It is unrefined if all of the $\Theta_j^{(i)} \leqslant \epsilon$. (2) It is further refined if any of the $\Theta_j^{(i)} > \epsilon$ provided, of course, that the resulting $P^k(\Omega_e) \leqslant P^H$. However, the following

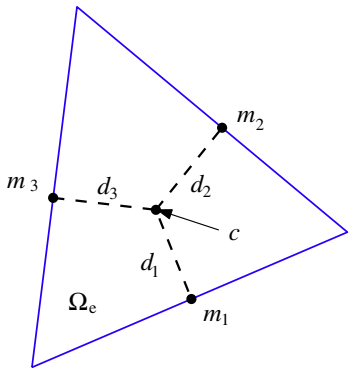

Fig. 3. Distance d_j used in the computation of $\Theta_j^{(i)}$.

additional constraint is added for the case of unrefining elements: once an element is refined it cannot be unrefined until it has been held at the current level of p approximation for a fixed number of time steps n_{lock} (specified by the user). This was implemented because it was observed from numerical experiments that in many instances elements were refined but then unrefined before the higher-order solution components became non-zero and the use of the p-adaptive algorithm did not result in solutions of increased accuracy. This also helps control the "flickering" of the order of the elements. We comment on the selection of a value for the n_{lock} parameter in the numerical results section. Additionally, we only allow the polynomial degree of an element to be increased by one during a given time step.

In summary, the algorithm is succinctly given as follows:

(1) Define P^L and P^H
(2) Initially at time $t = 0$ set $P^k(\Omega_e) = P^L \ \forall \Omega_e$
(3) At each time t^n compute the $\Theta_j^{(i)} \ \forall \Omega_e$
  (i) If any $\Theta_j^{(i)} > \epsilon^{(i)}$ then $P^k(\Omega_e)|_{t^{n+1}} = \begin{cases} P^{k+1} & \text{if } k+1 \leqslant H \\ P^k & \text{if } k+1 > H \end{cases}$
  (ii) If all $\Theta_j^{(i)} \leqslant \epsilon^{(i)}$ then
  $P^k(\Omega_e)|_{t^{n+1}} = \begin{cases} P^{k-1} & \text{if } k-1 \geqslant L \quad \text{and } n_{\text{count}} \geqslant n_{\text{lock}} \\ P^k & \text{if } k-1 < L \end{cases}$

where n_{count} is the number of time steps the element has been at the current P^k approximation.

We note that we have implemented the algorithm using the orthogonal, hierarchical basis of Dubiner [5]. Thus, refining p on a given element simply amounts to adding additional basis functions or higher-order "modes" to the lower-order space. Likewise, unrefining p on a given element simply amounts to taking away basis functions or higher-order modes. We note that in the case of unrefinement, given our choice of basis, simply taking away the higher-order basis functions is equivalent to taking an L^2 projection of $w_h^{(i)}$ onto the lower-order space. Finally, we also remark that along edges where the neighboring elements are of different order it is necessary to use the edge quadrature rule dictated by the higher-order element in order to maintain accuracy and stability.

3.3. The time discretization

The systems of ODE's given by Eq. (12) are discretized in time using explicit s-stage SSP Runge–Kutta methods, which can be written in the general form:

$$\begin{aligned} \mathbf{w}^{(0)} &= \mathbf{w}^n, \\ \mathbf{w}^{(i)} &= \sum_{l=0}^{i-1} \left\{ \alpha_{il}\mathbf{w}^{(l)} + \Delta t \beta_{il} \mathbf{L}(\mathbf{w}^{(l)}) \right\}, \quad i = 1, 2, \ldots, s, \\ \mathbf{w}^{n+1} &= \mathbf{w}^{(s)}, \end{aligned} \tag{14}$$

where α_{il} and β_{il}, which are subject to certain constraints (see, e.g., [15]), are the set of coefficients that define the Runge–Kutta method and the vectors $\mathbf{w}_h^{(n)}$ and $\mathbf{w}_h^{(n+1)}$ are the numerical solutions at times t and $t + \Delta t$, respectively, where Δt is the time step. Following [15], we denote an s-stage, mth order SSP Runge–Kutta by SSP(s,m).

We use an $m = (H+1)$th order accurate SSP Runge–Kutta method in conjunction with a P^H or hybrid P^L/P^H DG spatial discretization. The SSP schemes that are used are those identified in [10] that produce optimal, in terms of computational efficiency, $(H+1)$th order accurate so-called RKDG methods, i.e. methods that use a P^H DG spatial discretization and $(H+1)$th order accurate Runge–Kutta time discretization. For a P^1 DG spatial discretization, we use the optimal three-stage, second-order SSP Runge–Kutta scheme (SSP(3,2)). In conjunction with a P^2 or hybrid P^1/P^2 DG spatial discretization, we use the optimal SSP(5,3) scheme, and with a P^3 or hybrid P^L/P^3 DG spatial discretization ($L = 1$ or 2) we use the optimal SSP(6,4) scheme. As an example of these schemes, the optimal SSP(3,2) Runge–Kutta method is given by:

Table 1
CFL conditions for the SSP Runge–Kutta methods used in this paper.

s, m	CFL_{L^2}
3, 2	0.5882
5, 3	0.4060
6, 4	0.2747

$$\begin{aligned} \mathbf{w}_h^{(1)} &= \mathbf{w}_h^{(n)} + \frac{1}{2}\Delta t \mathbf{L}_h(\mathbf{w}_h^{(n)}), \\ \mathbf{w}_h^{(2)} &= \mathbf{w}_h^{(1)} + \frac{1}{2}\Delta t \mathbf{L}_h(\mathbf{w}_h^{(1)}), \\ \mathbf{w}_h^{(n+1)} &= \frac{1}{3}\mathbf{w}_h^{(1)} + \frac{2}{3}\mathbf{w}_h^{(2)} + \frac{1}{3}\Delta t \mathbf{L}_h(\mathbf{w}_h^{(2)}). \end{aligned} \tag{15}$$

See [15] and the references therein for the α_{il} and β_{il} values for the other schemes.

Due to the fact that these schemes are explicit, the size of the time step is restricted by a CFL condition. We note that the use of the stage-exceeding-order ($s > m$) SSP Runge–Kutta schemes in conjunction with DG spatial discretizations results in RKDG methods with improved CFL conditions for linear stability over the "standard" RKDG methods, i.e. ones that use $s = m$ Runge–Kutta SSP schemes, as demonstrated in [10]. The CFL conditions obtained from a one-dimensional, linear stability analysis in that work for the schemes mentioned above are summarized in Table 1. Using these results an estimate for the time step restriction for the two-dimensional SWE is given by:

$$\Delta t \leqslant \min_{\Omega_e} \left(\frac{h_e \times \text{CFL}_{L^2}}{|\lambda_{\max}|} \right), \tag{16}$$

where h_e is a suitably defined element diameter of Ω_e and $\lambda_{\max}$ is an estimate of the maximum (in absolute value) of the eigenvalues of the Jacobian of the flux function matrix $\mathbf{F}$ with respect to $\mathbf{w}$ over Ω_e.

4. Numerical results

In this section, we apply the p-adaptive algorithm described in the previous section to two test cases of coastal modeling interest. The test cases are problems examining flow in domains representing idealizations of a continental shelf break and a coastal inlet.

The test cases that are examined are problems that exhibit a wide range of scales and require very localized refinement in specific regions of the domain in order to obtain accurate solutions. With h-type finite element SWE models, this is achieved by providing additional spatial resolution in the form of h-refinement in key areas of the domain. However, determining the location and extent of the h-refinement that is needed for a particular problem can be a time consuming process. Furthermore, the level of h-refinement required can also lead to a significant increase in computational cost.

With our numerical tests, we demonstrate the following main points: (i) The stability of the p-adaptive algorithm; (ii) the efficiency advantages that can be obtained using grids with dynamic, local p-refinement using the adaptive procedure versus grids with elements of a fixed order p that use either local h-refinement or global p-refinement to adequately resolve the solution, as is typically done for refinement with hp methods for the SWE; (iii) the ability of the dynamic p-adaptive methods to detect, and subsequently resolve, important flow features that evolve during the course of a simulation.

4.1. Relative costs of h- and p-refinement

Before presenting the numerical results we remark on the relative costs of the P^1, P^2, and P^3 RKDG methods. Recall that in conjunction with a P^k DG spatial approximation we use an optimal $(k+1)$-order RK time discretization. The relative costs of the methods, on the same computational grid, are mainly a function of three factors: (i) The ratio of the degrees of freedom, (ii) the ratio of the CFL conditions given in Table 1, and (iii) the ratio of the number of stages of the RK method. Thus, the cost of the P^2 method relative to the P^1 method is approximately $(6/3)(0.5882/0.4060)(5/3) \approx 4.83$, i.e. the P^2 method is 4.83 times more expensive than the P^1 method, and similarly the relative cost of the P^3 method to the P^1 method is approximately $(10/3)(0.5882/0.2747)(6/3) \approx 14.28$. We note that the true measures of the relative costs of the methods are, of course, not this simple and involve additional factors such as the number of edge and area quadrature points, code structure, etc., however, we have found that the consideration of these three factors alone gives a fairly reasonable estimate that is consistent with what is observed computationally.

We can compare these measures with using h-refinement. For example, refining the elements uniformly as shown in Fig. 4 gives a four fold increase in the number of degrees of freedom and requires a halving of the time step. Thus, the computational cost is increased by a factor of 8 compared to the above mentioned factor of approximately 5 using p-refinement from $p = 1$ to $p = 2$. In the former case, the error (optimally) decreases by a factor of 4, while in the latter case it decreases (again optimally) by at least two orders of magnitude. Indeed, in our previous work involving DG methods for the SWE [9], it was established that for problems with smooth solutions global p-refinement is significantly more efficient in obtaining a specified error level than using global h-refinement.

With respect to the efficiency of the p-adaptive methods, we note the following. As mentioned previously, in the case when we use a P^L/P^H p-adaptive method an $(H+1)$-order RK method is used in order to maintain high-order accuracy. Thus, the P^1/P^2 method will be between 50% and 100% of the cost of the P^2 method, depending on the number of elements that are refined, the P^2/P^3 method will be somewhere between 60% and 100% of the cost of the P^2 method, and the P^1/P^3 will be somewhere between 30% and 100% of the cost of the P^3 method. To a small extent, the value used for the n_{lock} parameter in Section 3.2 also effects the efficiency of the method. Again, this parameter was included in the p-adaptive algorithm to help control the "flickering"of the element order. For the problems considered here, the value of n_{lock} was chosen based on numerical experiment. Specifically, n_{lock} was increased from 0 until the presence of "flickering" elements was nearly eliminated (this was determined by a visual inspection of the data). Values of n_{lock} in the range of 10–20 proved to eliminate this "flickering" and improve error levels without significantly affecting the efficiency of the method. Development of a more rigorous procedure for the selection of the n_{lock} parameter will be considered in future work. Finally, we remark that the efficiency of the dynamic p-adaptive methods could also be improved by implementing adaptive time stepping. This will also be considered in future work.

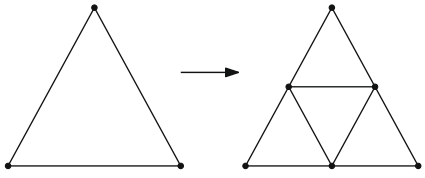

Fig. 4. A standard 1:4 grid refinement.

4.2. Test case 1: idealized continental shelf break

The domain of the first test case, including its areal extent and its bathymetry, is shown in Fig. 5. The rectangular domain is 1500 km in the x (east–west) direction and 1000 km in the y (north–south) direction (see Fig. 5a). The eastern edge of the domain ($x = 1500$ km) is an open ocean boundary along which a tidal forcing is specified. The three remaining boundaries are specified as no-normal flow boundaries. The bathymetry of the domain is an idealization of the bathymetry of the Atlantic Ocean along a cross section extending perpendicular from the east coast of the US ($x = 0$) out into the deep ocean (see Fig. 5b) and is given by:

$$h(x,y) = 2500 + \frac{2480}{\tanh(3)} \tanh(0.010(x-300)). \tag{17}$$

This equation defines a depth along the coast ($x = 0$) equal to 20 m. At approximately $x = 200$ km the depth begins to increase more rapidly at the continental shelf break. Then at approximately $x = 350$ km a more gradual slope is resumed until a depth of approximately 5000 m is reached at $x = 1500$ km. This bathymetric profile is uniform in the y direction. A gravitational constant of $g = 9.81$ m/s^2 was used.

The domain is discretized using two different computational grids. The first, which we will refer to as the base grid and denote by h_0, consists of 1200 uniform triangular elements as shown in Fig. 6a. The node-to-node spacing of the base grid is 50 km throughout the domain. The second (refined) grid, h_r (Fig. 6b) provides additional spatial resolution in the vicinity of the continental slope and on the continental shelf by halving the node-to-node spacing in those regions. Previous studies that have solved similar problems [2,8] using the computational model ADCIRC [12], which uses a CG finite element method with P^1 elements for the spatial

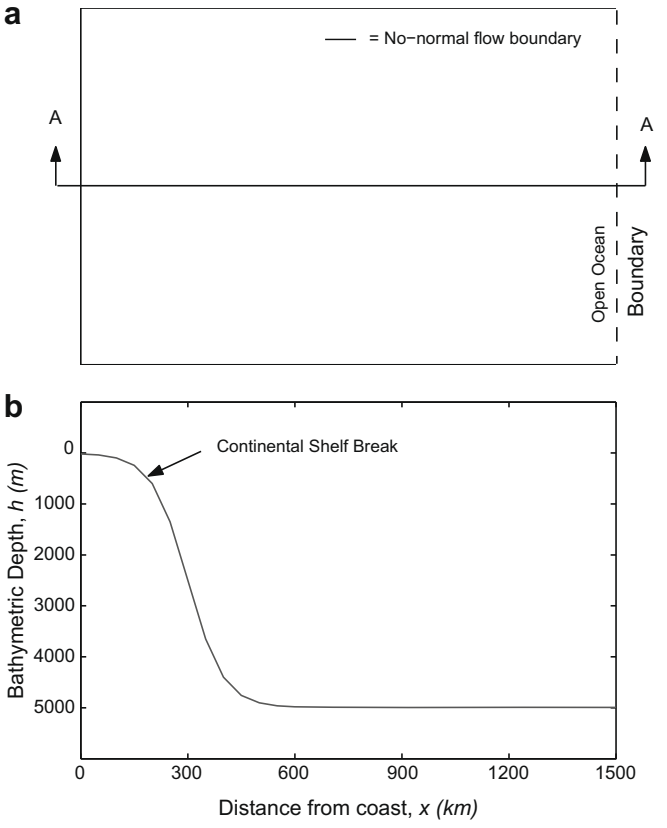

Fig. 5. (a) Plan view of the domain for test case 1. (b) Bathymetric profile along cross section A-A.

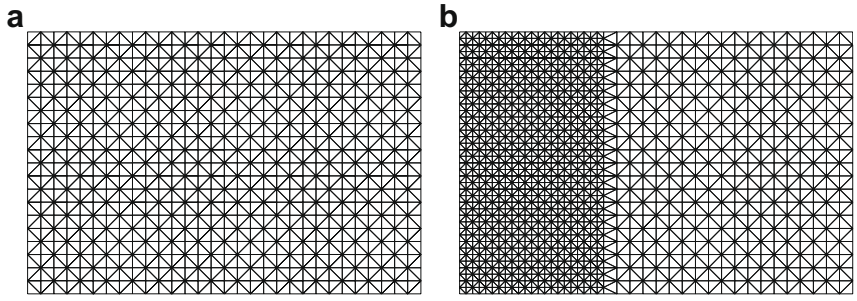

Fig. 6. The computational grids of test case 1: (a) the base grid consisting of 1200 elements. (b) Local refinement of the base grid in the region of the continental shelf and break.

discretization, have shown that providing additional spatial resolution in the form of *h*-refinement in these areas, especially in the shallower coastal region and in areas where the slope of the bathymetry changes greatly, is crucial in obtaining accurate solutions.

Using the base grid, we perform simulations (described below) with P^1, P^2, and P^3 DG spatial discretizations globally, i.e. the same P^k approximation throughout the entire domain, and with adaptive P^1/P^2, P^1/P^3, and P^2/P^3 DG spatial discretizations. Using the second grid, which provides local *h*-refinement in the vicinity of the continental slope and on the continental shelf, we perform a simulation using global P^1 elements. A highly resolved simulation is used to compute an "exact" solution from which errors are computed.

The simulations begin from rest and motion is induced by specifying an M_2 tidal signal (a period of approximately 12.42 h) along the open ocean boundary. The forcing is applied gradually, building up to a 1 m amplitude over a period of 2 days. The simulations are run for a period of 5 days, and solutions are recorded every 15 min from $T = 4$ days to $T = 5$ days. The time steps that are used in the simulations are approximately 50% of the time step that is computed from the estimate given by Eq. (17). This is done to insure that the spatial errors are the dominant portion of the error, while also taking into account the fact that higher-order, and also smaller, elements impose stricter time step restrictions, as already noted, and thus affect the overall efficiency of the method. A gravitational constant of $g = 9.81\ \mathrm{m/s^2}$ was used.

A comparison of the errors and run times of the different simulations are shown in Table 2. The L^1 errors reported are obtained by multiplying the error at the barycenter of each element by the element area, summing theses errors over the whole domain, and then dividing by the total area of the domain.

We note the following regarding these results. (i) The CPU times of the P^2 and P^3 methods relative to the P^1 method are roughly in line with what was to be expected based on the estimates discussed in the previous section. (ii) The errors for the P^2 and P^1/P^2 methods are very comparable (on the same order of magnitude) with the P^2 method giving only slightly better results. Likewise, the P^3 and P^2/P^3 errors are very comparable, again, with the P^3 method giving only slightly lower error levels. In both cases, the CPU times of the dynamic *p*-adaptive methods are approximately 70% of the CPU time of their global P^k counterparts (recall that given the additional constraints imposed by the higher-order time stepping these results can be no better than 50%). Thus, the P^1/P^2 and P^2/P^3 methods are more efficient than using global *p*-refinement while offering comparable accuracy. (iii) Unfortunately, the use of the P^1/P^3 method, which stood to provide the greatest gains in efficiency, did not pay off for this problem. While the P^1/P^3 simulation ran in only 55% of the CPU time of the P^3 method it only offered levels of accuracy comparable to the P^2 method but took roughly twice as long to run. (iv) The use of the local *p*-refinement is clearly more efficient than using local *h*-refinement for this problem. For example, the P^1/P^2 simulation ran in less time (78%) than the *h*-refined h_r, P^1 solution but gave errors an order of magnitude lower. (v) As shown in Fig. 7, which shows the simulation during a period of peak incoming tide, elements were only adapted in the coastal region and in the vicinity of the continental shelf as indicated by the shaded elements. Finally, we note that in the case that the "deep ocean" portion of a grid takes up a larger percentage of the total area of the domain slightly better efficiency results would most likely be observed for the *p*-adaptive methods.

Table 2
Errors and relative run-times for the seven different simulations for test case 1.

Simulation	L^1 error Surf. Elev.	L^1 error Velocity	Relative CPU times
h_0, P^1	5.1040e–02	7.6230e–03	1.00
h_0, P^2	1.0430e–03	9.4058e–05	4.93
h_0, P^3	1.1610e–04	2.1276e–05	16.28
h_0, P^1/P^2	2.2208e–03	1.2832e–04	3.48
h_0, P^1/P^3	1.0577e–03	9.4513e–05	9.00
h_0, P^2/P^3	1.5948e–04	2.4664e–05	11.40
h_r, P^1	1.1492e–02	7.3230e–03	4.49

4.3. Test case 2: idealized inlet

The second test case is an idealization of a coastal inlet. The domain, shown in Fig. 8, measures 4.5 km in the *x* (east–west) direction and 3 km in the *y* (north–south) direction, and consists of two distinct regions – a back-bay region to the east and an open ocean region to west. These regions are connected by a 0.75 km long channel that extends into the open ocean region via a pair of jetties. The bathymetry in the back-bay region and through the chan-

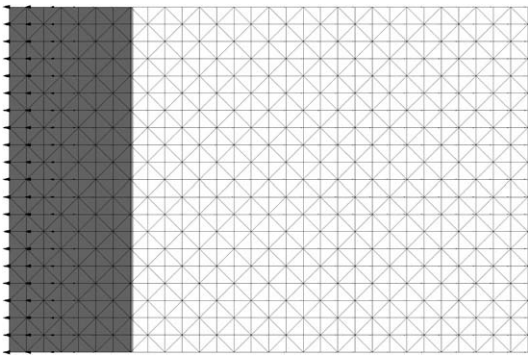

Fig. 7. Degree of the elements for the adaptive P^1/P^2 simulations (shaded elements=P^2) during a time of peak incoming tide with velocity vectors superimposed.

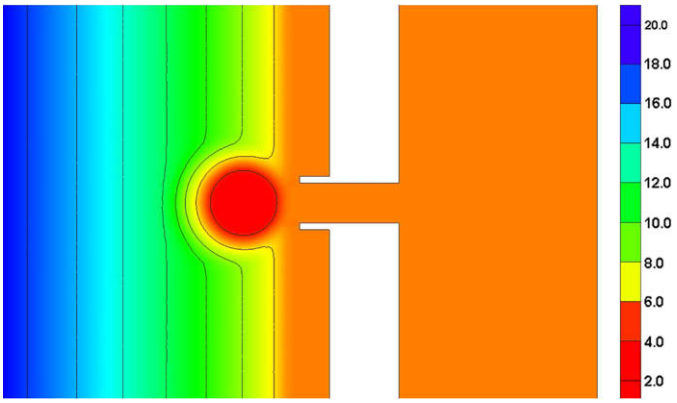

Fig. 8. The domain of test case 2 with bathymetric contours.

nel is flat at a depth of 5 m below resting water level. The bathymetry in the open ocean region varies linearly from a depth of 5 m at the entrance of the channel to a depth of 19 m at the far western boundary with the exception of a large ebb shoal, which measures approximately 750 m in diameter with a peak height 2 m below the geoid, located at the western entrance of the channel. The far western boundary is an open ocean boundary at which a tidal forcing will be specified while the remaining boundaries are all specified as no-normal flow. Again, a gravitational constant of $g = 9.81\ \mathrm{m/s^2}$ was used.

The computational grid used for the simulations is shown in Fig. 9. As was investigated for the previous problem, we examine the additional efficiency advantages that can be realized using *p*-adaptive grids and make a qualitative comparison of the results obtained using these grids compared to the grids using fixed, global *p*-refinement. In particular, we examine the effectiveness of the *p*-adaptive grids to adequately resolve the fine-scale flow features of the problem such as the structure of the jet issuing from the inlet and the interaction of the jet with the ebb shoal – namely, the formation and propagation of eddies around the ebb shoal and in the channel. In contrast to the previous problem, we do not examine a locally *h*-refined grid as the extent of *h*-refinement that is needed for this problem in order to adequately resolve the solution results in a grid with a number of elements that is nearly equal to the number of elements in a globally *h*-refined grid. It has already been established that using global *h*-refinement is less efficient for this problem than using global *p*-refinement [9].

We ran five different simulations using the computational grid as shown in Fig. 9 with P^1, P^2, and P^3 elements globally and with adaptive P^1/P^2 and P^2/P^3 grids. Based on the results of the previous test case we do not test the P^1/P^3 method. Again, the simulations start from rest, and an M_2 tidal forcing is applied gradually building up to the full amplitude over a period of one day. The forcing is

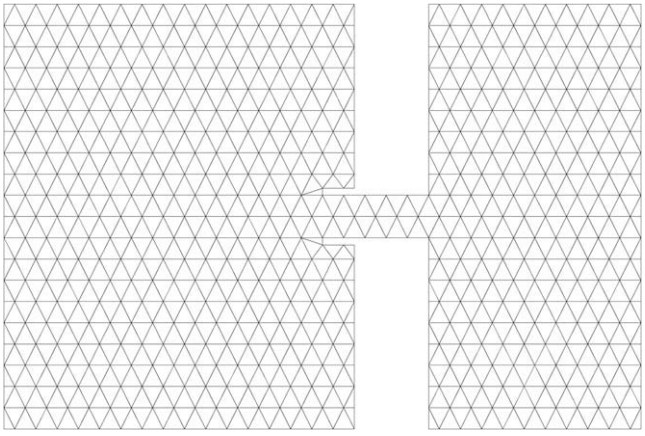

Fig. 9. The computational grid for test case 2.

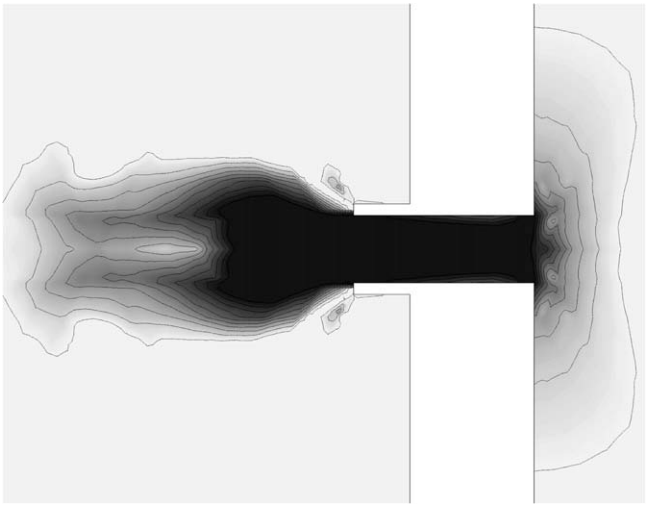

Fig. 10. Velocity contours of the P^1 solution.

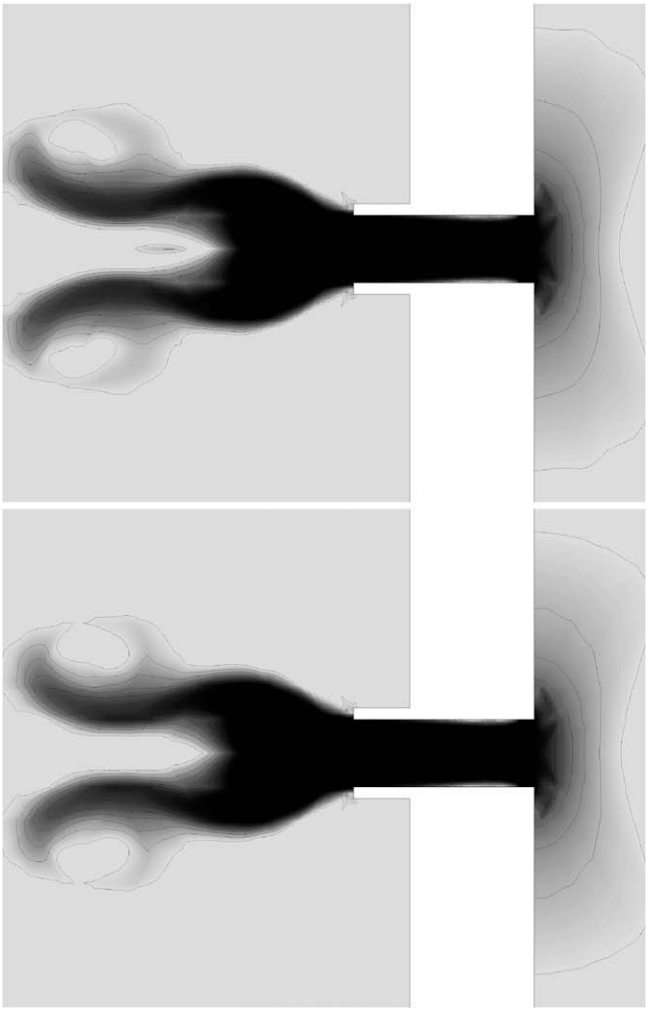

Fig. 11. Velocity contours of the P^2 (top) and P^1/P^2 (bottom) solutions.

specified such that the maximum velocity in the center of the channel during maximum flood tide is approximately 1 m/s. The simulations are run for a period of 2 days. The time steps are chosen as described for the previous problem.

We examine the solution during an approximate time of maximum ebb tide. The P^1 solution velocity contours are shown in Fig. 10, the P^2 and P^1/P^2 solutions are shown in Fig. 11, and finally the P^3 and P^2/P^3 solutions are shown in Fig. 12. In both of the p-adaptive cases, it can be observed that the results of the P^L/P^H methods are qualitatively very similar to the results of the fixed, globally p-refined grids that use P^H elements. More specifically, first it is noted that at this level of coarse h resolution the P^1 method fails to adequately resolve the eddies that form during the times of maximum ebb tide around the ebb shoal. However, the P^2 solution provides substantially better results than the P^1 solution with the eddies that are shed off of the ebb shoal now clearly being visible. As can be noted from Fig. 11, the P^2 and P^1/P^2 solutions are almost indistinguishable with the P^2 solution providing only slightly better resolution of the eddies. Going from P^2 to P^3 the improvement in the resolution of the eddies is, perhaps, less dramatic than going from P^1 to P^2, however, there is still a visible improvement in the solution. The use of the P^2/P^3 method offers slightly "sharper" resolution of the eddies than the P^2 method, but this time the global p and p-adaptive methods are not as similar to each other as the P^2 and P^1/P^2 solutions. However, a close visual inspection of the Figs. 10–12 reveals that both the P^3 and P^2/P^3 methods are resolving smaller scale eddies in the eastern part of the channel that were not resolved with the other simulations. With respect to efficiency, the adaptive P^L/P^H methods ran, again, in approximately 70% of the time as their P^H counterparts while offering significantly better resolution of the eddies than the P^1 method. Finally, Figs. 13 and 14 show where the elements have been refined during ebb and flood tides, respectively. It can be observed that the elements are refined by the dynamic p-adaptive algorithm in the area of strong currents and track the movement of the eddies through the inlet, into the back-bay, and as they shed off of the ebb shoal.

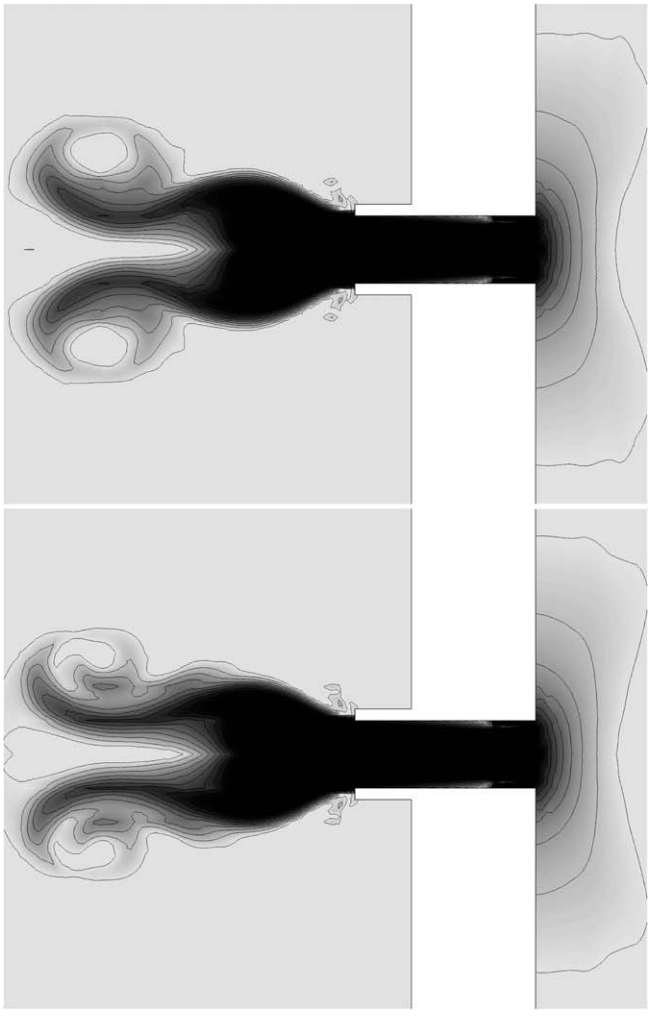

Fig. 12. Velocity contours of the P^3 (top) and P^2/P^3 (bottom) solutions.

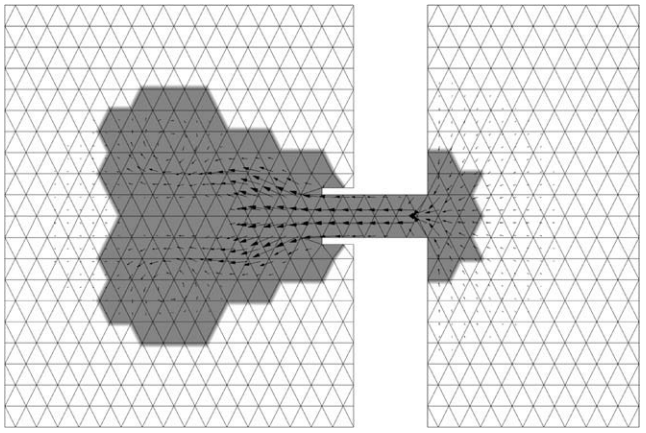

Fig. 13. Degree of the elements during ebb tide with velocity vectors superimposed (shaded elements=P^2).

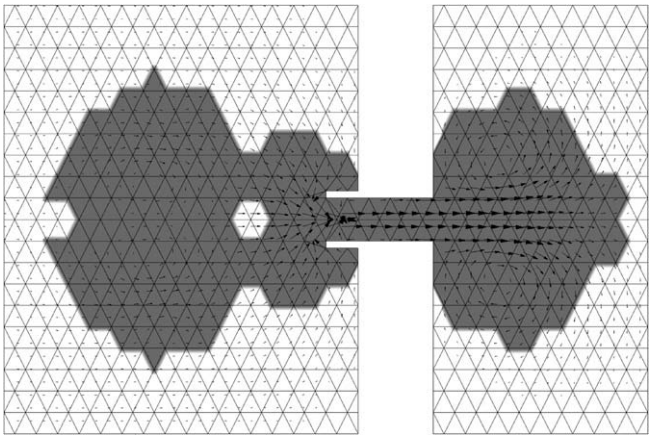

Fig. 14. Degree of the elements during flood tide with velocity vectors superimposed (shaded elements=P^2).

5. Concluding remarks

In this paper, we have developed and implemented a dynamic p-adaptive RKDG method for the two-dimensional SWE on unstructured triangular grids. The algorithm was described in detail, and the method was shown to perform well in two test cases. Specifically, a quantitative analysis comparing error levels and run times for the case of an idealized shelf break problem showed that the use of dynamic p-adaptive methods was more computationally efficient than the use of fixed grids that use either local h-refinement or global p-refinement to resolve the solution in the shallow coastal area and along the shelf break while offering comparable accuracy. Likewise, for the problem of the idealized coastal inlet, the p-adaptive algorithm was shown to provide qualitatively sim-

ilar results in the resolution of the fine scale flow features that developed around the ebb shoal as those in the case of using the higher-order approximation globally, but ran in 70% of the CPU time of the globally refined p solutions.

Future work in the area of adaptivity using DG methods will involve the implementation of h-adaptation, which allows the use of hanging nodes or non-conforming grids, the development and implementation of a fully hp-adaptive model, and the application of these models to field studies.

Acknowledgements

This work was supported by National Science Foundation Grants DMS 0620697 and 0620696 and by the Office of Naval Research, Award Number: N00014-06-1-0285. The authors would also like to thank Chris Mirabito for performing a number of simulations.

References

[1] P.E. Bernard, N. Chevaugeon, V. Legat, E. Deleersnijder, J.F. Remacle, High-order h-adaptive discontinuous Galerkin methods for ocean modelling, Ocean Dyn. 57 (2007) 109–121.
[2] C.A. Blain, J.J. Westerink, R.A. Luettich, The influence of domain size on the response characteristics of a hurricane storm-surge model, J. Geophys. Res. 99 (1994) 18467–18479.
[3] A. Burbeau, P. Sagaut, A dynamic p-adaptive discontinuous Galerkin method for viscous flow with shocks, Comput. Fluids 34 (2005) 401–417.
[4] B. Cockburn, C.W. Shu, Runge–Kutta discontinuous Galerkin methods for convection–dominated problems, J. Sci. Comput. 16 (2001) 173–261.
[5] M. Dubiner, Spectral methods on triangles and other domains, J. Sci. Comput. 6 (1998) 345–390.
[6] C. Eskilsson, S.J. Sherwin, A triangular spectral/hp discontinuous Galerkin for modelling 2D shallow water, Int. J. Numer. Methods Fluids 45 (2004) 605–623.
[7] F.X. Giraldo, J.S. Hesthaven, T. Warburton, Nodal high-order discontinuous Galerkin methods for the spherical shallow water equations, J. Comput. Phys. 181 (2002) 499–525.
[8] S.C. Hagen, J.J. Westerink, R.L. Kolar, One-dimensional finite element grids based on a localized truncation error analysis, Int. J. Numer. Methods Fluids 32 (2000) 241–261.
[9] E.J. Kubatko, J.J. Westerink, C. Dawson, hp discontinuous Galerkin methods for advection dominated problems in shallow water flow, Comput. Methods Appl. Mech. Engrg. 196 (2006) 437–451.
[10] E.J. Kubatko, J.J. Westerink, C. Dawson, Semidiscrete discontinuous Galerkin methods and stage-exceeding-order, strong-stability-preserving Runge–Kutta time discretizations, J. Comput. Phys. 222 (2007) 832–848.
[11] J.C. Levin, D.B. Haidvogel, B. Chua, A.F. Bennet, M. Iskandarani, Euler–Lagrange equations for the spectral element shallow water system, Ocean Model. 12 (2006) 348–377.
[12] R.A. Luettich, J.J. Westerink, N.W. Scheffner, ADCIRC: an advanced three-dimensional circulation model for shelves, coasts and estuaries, in: Report 1: theory and methodology of ADCIRC-2DDI and ADCIRC-3DL, Dredging Research Program Technical Report DRP-92-6, US Army Engineers Waterways Experiment Station, Vicksburg, MS, 1992.
[13] L.Y. Oey, P. Chen, A nested-grid ocean model: with application to the simulation of meanders and eddies in the Norwegian coastal current, J. Geophys. Res. 97 (1992) 20063–20086.
[14] J.F. Remacle, S.S. Frazao, X. Li, M.S. Shephard, An adaptive discretization of shallow-water equations based on discontinuous Galerkin methods, Int. J. Numer. Methods Fluids 52 (2006) 903–923.
[15] S.J. Ruuth, Global optimization of explicit strong-stability-preserving Runge–Kutta methods, Math. Comput. 75 (2005) 183–207.
[16] J. Shen, H. Wang, M. Sisson, W. Gong, Storm tide simulation in the Chesapeake Bay using an unstructured grid model, Estuar. Coast. Shelf Sci. 68 (2006) 1–16.
[17] J.J. Westerink, R.A. Luettich, A.M. Baptista, N.W. Scheffner, P. Farrar, Tide and storm-surge predictions using a finite element model, J. Hydraul. Engrg. 118 (1992) 1373–1390.
[18] J.J. Westerink, R.A. Luettich, J.C. Muccino, Modeling tides in the western North-Atlantic using unstructured graded grids, Tellus 46A (1994) 178–199.