



NRL/FR/7322--03-10,050

Implementation of a Discontinuous Galerkin Discretization of the Conservation of Mass Equation in QUODDY

M.J. GUILLOT

*Department of Mechanical Engineering
University of New Orleans
New Orleans, LA*

C.A. BLAIN

*Ocean Dynamics and Prediction Branch
Oceanography Division*

C.N. DAWSON

*Department of Aerospace Engineering
University of Texas at Austin
Austin, TX*

March 19, 2003

REPORT DOCUMENTATION PAGE				<i>Form Approved</i> <i>OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 19-03-2003		2. REPORT TYPE Formal		3. DATES COVERED (From - To) October 29-31, 2002	
4. TITLE AND SUBTITLE Implementation of a Discontinuous Galerkin Discretization of the Conservation of Mass Equation in QUODDY				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER 602435N	
6. AUTHOR(S) M.J. Guillot,* C. A. Blain, and C.N. Dawson**				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory Oceanography Division Stennis Space Center, MS 39529-5004				8. PERFORMING ORGANIZATION REPORT NUMBER NRL/FR/7322--03-10,050	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research 800 North Quincy St. Arlington, Virginia 22217-5560				10. SPONSOR / MONITOR'S ACRONYM(S) ONR	
				11. SPONSOR / MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES * Department of Mechanical Engineering, University of New Orleans, New Orleans, LA ** Department of Aerospace Engineering, University of Texas at Austin, Austin, TX					
14. ABSTRACT Two variations of a discontinuous Galerkin method are investigated for the primitive form of the time-dependent, 2-D, depth-averaged conservation of mass equation for shallow water. Linear approximating functions are employed, which are defined locally on each element, providing three degrees of freedom per element. In the first method, the degrees of freedom are the nodal values for each element. Analytical integration rules are developed to evaluate the volume integrals, and Gaussian quadrature is used to evaluate the element edge integrals produced by the finite element approximation. Time integration is achieved using a second-order Runge-Kutta method. The method is implemented into the computer code QUODDY, replacing the generalized wave continuity equation (GWCE) formulation in the code. The implementation is validated on a test case involving tidal boundary conditions in the Gulf of Maine.					
15. SUBJECT TERMS QUODDY GWCE Galerkin					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UL	18. NUMBER OF PAGES 21	19a. NAME OF RESPONSIBLE PERSON Cheryl Ann Blain
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (include area code) 228-688-5450

CONTENTS

1. INTRODUCTION	1
2. ORIGINAL FINITE ELEMENT FORMULATION IN QUODDY	1
3. DISCONTINUOUS GALERKIN FORMULATION	2
3.1 Problem Statement	2
3.2 Basis and Test Functions	3
3.3 Linear Integration Rules for Triangular Elements	4
3.4 Edge Fluxes	4
4. SOLUTION ALGORITHM	6
4.1 Temporal Discretization	6
4.2 Volume Integrals	7
4.3 Interior and Boundary Edge Integrals	8
4.4 CFL Condition	9
4.5 Slope Limiting	9
5. IMPLEMENTATION INTO QUODDY5	11
5.1 Numerical Algorithm	12
5.2 Projection to and from Nodes	12
5.3 Input Modification	12
6. VALIDATION	13
6.1 Maximum Courant Number Verification	13
6.2 Gulf of Maine with Elevation Boundary Conditions	13
6.3 Discussion of Different Methods Investigated	14
7. CONCLUSIONS	17
8. RECOMMENDATIONS	17
REFERENCES	17

IMPLEMENTATION OF A DISCONTINUOUS GALERKIN DISCRETIZATION OF THE CONSERVATION OF MASS EQUATION IN QUODDY

1. INTRODUCTION

In this effort, a discontinuous Galerkin (DG) formulation is developed for the primitive 2-D unsteady depth-averaged conservation of mass equation. The formulation is implemented into QUODDY5, developed by Lynch and Werner [1] at Dartmouth University and capable of solving the 3-D shallow water hydrodynamics and temperature and salinity transport.

Traditional continuous Galerkin finite elements have historically not performed well when applied to the primitive form of the shallow water conservation of mass equation due to spurious oscillations introduced into the solution. In an effort to eliminate the spurious oscillations, Lynch and Gray [2] developed the generalized wave continuity equation (GWCE). The method eliminated the spurious oscillations and has been adopted in several shallow water hydrodynamic codes, for example QUODDY [1] and ADCIRC [3]. The term discontinuous Galerkin (DG) method refers to several finite element methods that use discontinuous discrete approximating spaces, such as the Bassi and Rebay method [4], the Local Discontinuous Galerkin (LDG) [5] method, the Oden, Babuska, and Baumann method [6], the interior penalty Galerkin methods of Wheeler [7], Douglas and Dupont [8], and the NIPG methods [9,10]. Cockburn et al. performed an extensive study of the DG method for hyperbolic conservation laws in a series of papers [11-15]. Arnold et al. [16] present a general framework of these methods. Application of these methods to a wide variety of problems can be found in Ref. 17.

A unique property of the DG method is that the DG solution satisfies the mass conservation equation locally element by element. In addition, the flexibility of DG methods facilitates dynamic h-p adaptivity. The basis functions are defined as piecewise discontinuous polynomials on each element, so the order of polynomials can be refined locally (p-adaptivity). Additionally, elements can be nonconforming, which facilitates local mesh refinement (h-adaptivity). These features can be exploited separately or in combination (h-p adaptivity) to locally refine the approximation in regions of high gradients. Moreover, because they are highly local, DG methods only require communication between elements that share faces; thus they are well-suited for parallel computation.

2. ORIGINAL FINITE ELEMENT FORMULATION IN QUODDY

QUODDY is capable of computing 3-D shallow water flow, as well as transport of temperature and salinity. The system of equations for computing the velocity field implemented in QUODDY are the GWCE, the 3-D primitive momentum equation, and 3-D incompressible conservation of mass equation. The GWCE is used to compute the elevation given the depth-averaged velocity components. The 3-D primitive momentum equations are used to compute the 3-D horizontal components of velocity, and the 3-D incompressible conservation of mass equation is used to compute the vertical velocity. Reference 1 provides a detailed discussion of the governing equations and finite element method implementation. QUODDY is capable of incorporating land, velocity, and elevation boundary conditions into the solution.

The numerical procedure currently in QUODDY is as follows:

1. Given an initial depth H and depth-averaged velocity field (U, V) , update the elevation by solving the GWCE using a three-level time differencing. The solution is semi-implicit in that all of the linear terms are solved implicitly, but all of the nonlinear terms involving velocity are lagged.
2. Using the new computed depth, compute the components of velocity for the entire 3-D domain using a two-level time differencing.
3. Compute the component of velocity such that the velocity field is divergence free, i.e., $\nabla \cdot \vec{V} = 0$.

3. DISCONTINUOUS GALERKIN FORMULATION

In this section, a DG formulation on triangular elements is developed and presented. The method begins by forming a weighted residual statement and integrating over a typical element. Summing over all elements produces a system of equations to advance the solution in time. Integrating the test and basis functions over each element produces volume and interface integrals that must be evaluated. In this formulation, the volume integrals are evaluated analytically and the element interface integrals are evaluated using Gaussian quadrature.

3.1 Problem Statement

The primitive form of the 2-D depth-averaged inviscid shallow water equations is written in conservative form as

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = S(x, t), \quad (1)$$

where

$$Q = \begin{bmatrix} H \\ UH \\ VH \end{bmatrix} \quad F = \begin{bmatrix} UH \\ UUH + g\frac{H^2}{2} \\ UVH \end{bmatrix} \quad G = \begin{bmatrix} VH \\ UVH \\ VH + g\frac{H^2}{2} \end{bmatrix} \quad S = \begin{bmatrix} 0 \\ gH \frac{\partial h}{\partial x} \\ gH \frac{\partial h}{\partial y} \end{bmatrix} \quad (2)$$

are the 3×1 vectors that represent conservation of mass, x -momentum, and y -momentum, respectively.

This effort focuses on incorporating the conservation of mass into QUODDY using the DG method. However, the continuous Galerkin method already implemented into QUODDY is retained for conservation of momentum.

Before developing the numerical formulation, element nomenclature used throughout the remainder of this work is defined. Each element contains three nodes and three edges as shown in Fig. 1(a). The edges between each element are denoted by Γ_k . Each interior edge is attached to two elements, defined as the left (L) and right (R) element. The unit normal to an edge \vec{n}_k is defined such that it always points outward from the left element. This is described in Fig. 1(b). Edges on the boundary are only attached to the left element. The boundary condition replaces the right element.

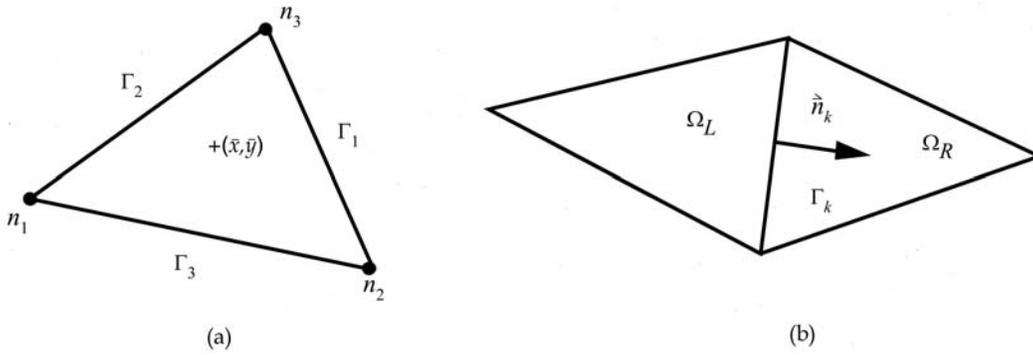


Fig. 1 — (a) Element and (b) edge nomenclature for typical interior elements

The DG method involves jumps and averages across edges. The jump and average of any quantity (a) across edge k are defined, respectively, as

$$\begin{aligned} [a]_k &= (a_R - a_L) \\ \{a\}_k &= \frac{1}{2}(a_L + a_R). \end{aligned} \quad (3)$$

Multiplying conservation of mass by a test function w and integrating over a typical element Ω_j produces the semidiscrete weak statement of conservation of mass and is written as

$$\frac{d}{dt} \int_{\Omega_j} w_j H_j d\Omega_j + \sum_{k=1}^3 \int_{\Gamma_k} w (\bar{F} \cdot \bar{n}_{k,o}) d\Gamma_k - \int_{\Omega_j} (\bar{F} \cdot \nabla w) d\Omega_j = 0, \quad (4)$$

where Γ_k are the edges of element j , and $\bar{F} = (UH)\hat{i} + (VH)\hat{j}$ is the flux vector. In Eq. (4), the normal $\bar{n}_{k,o} = n_x\hat{i} + n_y\hat{j}$ is the unit *outward* normal vector to element j . Therefore, $\bar{F} \cdot \bar{n}_{k,o}$ is the outward flux normal to the element edge k . As noted previously, normals in the implementation of the algorithm, defined as per Fig. 1(b), point from L to R . Therefore, for all right elements, the normal for an edge must be multiplied by minus one. This prevents the need to define two normals for each edge.

3.2 Basis and Test Functions

In the current work, linear basis and test functions are used. Linear basis and test functions are defined for the j^{th} element. Linear basis functions provide three degrees of freedom per element. Several formulations are possible depending on the degrees of freedom chosen. The general forms of the basis and test functions are

$$H_j(x) = \sum_{i=1}^3 a_i \phi_i \quad w_j(x) = \sum_{i=1}^3 b_i \phi_i. \quad (5)$$

The specific form of the basis functions ϕ_i depends on the degrees of freedom chosen. Commonly, the degrees of freedom are the element average and gradient, the three element vertices, or the three element

edge midpoints. In this work, two formulations are investigated: element average and gradient (denoted as Method 1) and element vertices (denoted as Method 2).

The functions ϕ_i and coefficients a_i are given in Table 1. The coefficients b_i (Eq. (5)) are arbitrary numbers that are eliminated from the final set of equations. The element centroid is written as (\bar{x}, \bar{y}) .

Table 1 — Test Functions and Coefficients

i	Element Average and Gradient		Element Vertices	
	a_i	ϕ_i	a_i	ϕ_i
1	H_{j_0}	1	H_1	$\frac{1}{3} + \frac{(y_2 - y_3)(x - \bar{x})(x_2 - x_3)(y - \bar{y})}{2A_e}$
2	$\frac{\partial H_j}{\partial x}$	$(x - \bar{x})$	H_2	$\frac{1}{3} + \frac{(y_3 - y_1)(x - \bar{x})(x_3 - x_1)(y - \bar{y})}{2A_e}$
3	$\frac{\partial H_j}{\partial y}$	$(y - \bar{y})$	H_3	$\frac{1}{3} + \frac{(y_1 - y_2)(x - \bar{x})(x_1 - x_2)(y - \bar{y})}{2A_e}$

3.3 Linear Integration Rules for Triangular Elements

Applying the linear basis and test functions of the DG method to triangular elements produces integrals of the form

$$I_x = \int_A (x - \bar{x})^2 dA \quad I_y = \int_A (y - \bar{y})^2 dA \quad I_{xy} = \int_A (x - \bar{x})(y - \bar{y}) dA. \quad (6)$$

The integrals are evaluated as

$$\begin{aligned} I_x &= A \left(\frac{1}{12} (x_1^2 + x_2^2 + x_3^2) - \frac{1}{4} \bar{x}^2 \right) \\ I_y &= A \left(\frac{1}{12} (y_1^2 + y_2^2 + y_3^2) - \frac{1}{4} \bar{y}^2 \right) \\ I_{xy} &= A \left(\frac{1}{12} (x_1 y_1 + x_2 y_2 + x_3 y_3) - \frac{1}{4} \bar{x} \bar{y} \right). \end{aligned} \quad (7)$$

3.4 Edge Fluxes

The weak statement of conservation of mass includes integrals of the fluxes at the edges of each element. Upwinding is achieved by replacing the normal flux in the edge integral of Eq. (4) with an upwinded numerical flux function normal to the element edge, denoted by f_n . The numerical flux function is computed by solving a Riemann problem normal to each element interface. With this substitution, Eq. (4) becomes

$$\frac{d}{dt} \int_{\Omega_j} w_j H_j d\Omega_j = \int_{\Omega_j} (\bar{\mathbf{F}} \cdot \nabla w) d\Omega_j - \sum_{k=1}^3 \int_{\Gamma_k} w f_n d\Gamma_k. \quad (8)$$

This can be written in matrix form as

$$[\mathbf{M}] \frac{d[\mathbf{H}]}{dt} = [\mathbf{V}][\mathbf{H}] - [\mathbf{F}], \quad (9)$$

where $[\mathbf{M}]$ is the mass matrix, $[\mathbf{V}]$ is a 3×1 matrix involving the volumetric flux integral, and $[\mathbf{F}]$ is a 3×1 vector involving the edge flux integral. $[\mathbf{H}]$ is a 3×1 vector involving the degrees of freedom for each element. The specific form of the vectors and matrices depends on the method chosen.

3.4.1 Riemann Problem

The jumps in the dependent variables across element interfaces produce a Riemann problem normal to each element interface. The governing equations, as represented by Eqs. (1) and (2), are linearized by Roe-averaging, and the solution to the linearized Riemann problem computed. Once the solution to each local Riemann problem is obtained, the edge fluxes are computed. The solution to the linearized Riemann problem is well known (see, for example, Leveque [18]).

The solution to the normal Riemann problem is written in terms of the eigenvalues and eigenvectors of the normal Jacobian matrix of Eq. (1), which is written in quasilinear form as

$$\frac{\partial Q}{\partial t} + [A] \frac{\partial Q}{\partial x} + [B] \frac{\partial Q}{\partial y} = S(x, t), \quad (10)$$

where $[A]$ and $[B]$ are 3×3 matrices.

The Roe-averaged normal Jacobian, $[A]_n = [A]n_x + [B]n_y$, is written as

$$[A]_n = \begin{bmatrix} 0 & n_x & n_y \\ (-\bar{U}^2 n_x + g\bar{H}n_x - \bar{U}\bar{V}n_y) & (2\bar{U}n_x + \bar{V}n_y) & (\bar{U}n_y) \\ (-\bar{V}^2 n_y + g\bar{H}n_y - \bar{U}\bar{V}n_x) & (\bar{V}n_x) & (\bar{U}n_x + 2\bar{V}n_y) \end{bmatrix}. \quad (11)$$

The eigenvalues of $[A]_n$ are

$$\begin{aligned} \bar{\lambda}_1 &= \bar{u}_n - \sqrt{g\bar{H}} \\ \bar{\lambda}_2 &= \bar{u}_n \\ \bar{\lambda}_3 &= \bar{u}_n + \sqrt{g\bar{H}}, \end{aligned} \quad (12)$$

where $\bar{u}_n = \bar{U}n_x + \bar{V}n_y$ is the velocity normal to the edge and the overbars denote Roe-averaged values.

The corresponding right eigenvectors are

$$\bar{\mathbf{r}}_1 = \begin{bmatrix} 1 \\ \bar{U} - \sqrt{g\bar{H}n_x} \\ \bar{V} - \sqrt{g\bar{H}n_y} \end{bmatrix} \quad \bar{\mathbf{r}}_2 = \begin{bmatrix} 0 \\ -n_x \\ n_y \end{bmatrix} \quad \bar{\mathbf{r}}_3 = \begin{bmatrix} 1 \\ \bar{U} + \sqrt{g\bar{H}n_x} \\ \bar{V} + \sqrt{g\bar{H}n_y} \end{bmatrix}. \quad (13)$$

The Roe-averaged values are defined for an edge k as

$$\bar{H} = \frac{H_L + H_R}{2} \quad (14)$$

$$\bar{u}_n = \frac{u_{nL}\sqrt{H_L} + u_{nR}\sqrt{H_R}}{\sqrt{H_L} + \sqrt{H_R}}.$$

The subscript (L,R) denotes computing the given quantity within the left or right element at edge k . The value of the numerical flux function for conservation of mass on given edge depends on the signs of the eigenvalues at the edge. The numerical flux function for conservation of mass is written as

$$\left. \begin{array}{l} f_n = u_{nL}H_L \\ f_n = u_{nR}H_R \\ f_n = u_{nL}H_L + \frac{\bar{\lambda}_1}{\bar{\lambda}_3 - \bar{\lambda}_1} [\bar{\lambda}_3(H_R - H_L) - (u_{nR}H_R - u_{nL}H_L)] \end{array} \right\} \begin{array}{l} \bar{\lambda}_1 \geq 0 \text{ and } \bar{\lambda}_3 > 0 \\ \bar{\lambda}_1 < 0 \text{ and } \bar{\lambda}_3 \leq 0 \\ \bar{\lambda}_1 < 0 \text{ and } \bar{\lambda}_3 > 0 \end{array} \left. \vphantom{\begin{array}{l} f_n = u_{nL}H_L \\ f_n = u_{nR}H_R \\ f_n = u_{nL}H_L + \frac{\bar{\lambda}_1}{\bar{\lambda}_3 - \bar{\lambda}_1} [\bar{\lambda}_3(H_R - H_L) - (u_{nR}H_R - u_{nL}H_L)] \end{array}} \right\} \begin{array}{l} \text{Critical and Supercritical} \\ \text{Subcritical} \end{array} \quad (15)$$

4. SOLUTION ALGORITHM

The solution to Eq. (8) involves evaluating the integrals using the analytical integration rules presented in Section 3.3 for the volume integrals, and Gaussian quadrature for the edge integrals. The time derivative is approximated using a second-order Runge-Kutta method between time levels n and $n + 1$.

4.1 Temporal Discretization

Two variations of a second-order Runge-Kutta method were investigated. Each is a predictor-corrector type discretization. The first method advances the solution from time level to time level as follows:

$$\mathbf{H}^{n+\frac{1}{2}} = \mathbf{H}^n + \frac{\Delta t}{2} L(\mathbf{H}^n, \mathbf{V}^n, \mathbf{F}^n) \quad (16)$$

$$\mathbf{H}^{n+1} = \mathbf{H}^n + \Delta t L\left(\mathbf{H}^{n+\frac{1}{2}}, \mathbf{V}^{n+\frac{1}{2}}, \mathbf{F}^{n+\frac{1}{2}}\right).$$

The second method, known as Henn's method, advances the solution through the following predictor-corrector sequence:

$$\begin{aligned}\mathbf{H}^* &= \mathbf{H}^n + \Delta t L(\mathbf{H}^n, \mathbf{V}^n, \mathbf{F}^n) \\ \mathbf{H}^{n+1} &= \mathbf{H}^n + \frac{\Delta t}{2} (L(\mathbf{H}^n, \mathbf{V}^n, \mathbf{F}^n) + L(\mathbf{H}^*, \mathbf{V}^*, \mathbf{F}^*)),\end{aligned}\quad (17)$$

where $L(\mathbf{H}, \mathbf{V}, \mathbf{F})$ are written as

$$L(\mathbf{H}, \mathbf{V}, \mathbf{F}) = \mathbf{M}^{-1} \mathbf{V} \mathbf{H} - \frac{1}{A} \mathbf{M}^{-1} \mathbf{F}. \quad (18)$$

As discussed below, the second method is somewhat more stable than the first.

A 3×3 system of algebraic equations is produced for each element to update the value of H_j . Note that each element is coupled only to its neighboring elements via the edge flux integrals.

4.2 Volume Integrals

The volume integral involving the time derivative at time produces, for methods 1 and 2, respectively,

$$\left[\int_{A_j} w H_j d\Omega_j \right] = A_j \mathbf{M} \mathbf{H} = A_j \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{I_x}{A} & \frac{I_{xy}}{A} \\ 0 & \frac{I_{xy}}{A} & \frac{I_y}{A} \end{bmatrix} \begin{bmatrix} H_{j0} \\ \frac{\partial H_j}{\partial x} \\ \frac{\partial H_j}{\partial y} \end{bmatrix} = A_j \begin{bmatrix} \frac{1}{6} & \frac{1}{12} & \frac{1}{6} \\ \frac{1}{12} & \frac{1}{6} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{12} & \frac{1}{6} \end{bmatrix} \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix}. \quad (19)$$

Similarly, the volume integral of the flux produces,

$$\begin{aligned} \left[\int_{\Omega_j} (\bar{\mathbf{F}}^n \cdot \nabla w) d\Omega_j \right] &= A_j \mathbf{V}^n \mathbf{H} = A_j \begin{bmatrix} 0 & 0 & 0 \\ U_{j0}^n & \left(\frac{I_x}{A} \frac{\partial U_j^n}{\partial x} + \frac{I_{xy}}{A} \frac{\partial U_j^n}{\partial y} \right) & \left(\frac{I_{xy}}{A} \frac{\partial U_j^n}{\partial x} + \frac{I_y}{A} \frac{\partial U_j^n}{\partial y} \right) \\ V_{j0}^n & \left(\frac{I_x}{A} \frac{\partial V_j^n}{\partial x} + \frac{I_{xy}}{A} \frac{\partial V_j^n}{\partial y} \right) & \left(\frac{I_{xy}}{A} \frac{\partial V_j^n}{\partial x} + \frac{I_y}{A} \frac{\partial V_j^n}{\partial y} \right) \end{bmatrix} \begin{bmatrix} H_{j0} \\ \frac{\partial H_j}{\partial x} \\ \frac{\partial H_j}{\partial y} \end{bmatrix} \\ &= \frac{1}{24} \begin{bmatrix} (y_2 - y_3)(2U_1^n + U_2^n + U_3^n) & (y_2 - y_3)(U_1^n + 2U_2^n + U_3^n) & (y_2 - y_3)(U_1^n + 2U_2^n + U_3^n) \\ (y_3 - y_1)(2U_1^n + U_2^n + U_3^n) & (y_3 - y_1)(U_1^n + 2U_2^n + U_3^n) & (y_3 - y_1)(U_1^n + 2U_2^n + U_3^n) \\ (y_1 - y_2)(2U_1^n + U_2^n + U_3^n) & (y_1 - y_2)(U_1^n + 2U_2^n + U_3^n) & (y_1 - y_2)(U_1^n + 2U_2^n + U_3^n) \end{bmatrix} \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix}. \end{aligned} \quad (20)$$

In Eq. (20), the velocities have been projected from nodal degrees of freedom to DG degrees of freedom. The projection algorithm is described in Section 5.2.

4.3 Interior and Boundary Edge Integrals

For a given edge k attached to element j , the edge flux integral is written as

$$\mathbf{F} = \sum_{k=1}^3 \int_{\Gamma_k} f_n^n \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} d\Gamma_k. \quad (21)$$

The edge flux integrals are evaluated using Gaussian quadrature. Both methods produce integrals of the following form:

$$\int_{\Gamma_k} f_n^n \begin{bmatrix} 1 \\ (x - \bar{x}_j) \\ (y - \bar{y}_j) \end{bmatrix} d\Gamma_k = \int_{\Gamma_k} f_n^n \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} d\Gamma_k - \begin{bmatrix} 1 \\ \bar{x}_j \\ \bar{y}_j \end{bmatrix} \int_{\Gamma_k} f_n^n d\Gamma_k. \quad (22)$$

Note that, in general, the integrand depends not only on the value of the edge flux, but also on the values $(x - \bar{x}_j)$ and $(y - \bar{y}_j)$ for the given element j . Thus, the same edge integral will have two different values for the two elements sharing the edge. For computational efficiency, the integrals given by the first expression on the RHS of Eq. (22) are evaluated using Gaussian quadrature for each edge, and the total integral for an edge connected to element j is found by summing the two terms on the RHS of Eq. (22). Using Gaussian quadrature, this is evaluated as

$$\int_{\Gamma_k} f_n^n \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} d\Gamma_k = \frac{L_k}{2} \sum_{m=1}^{N_p} w_m \begin{bmatrix} f_{n,m}^n \\ x_m f_{n,m}^n \\ y_m f_{n,m}^n \end{bmatrix}, \quad (23)$$

where N_p is the total number of integration points, (x_m, y_m) are the Gaussian integration points, w_m represents the Gaussian weighting factor, and L_k is the length of the edge. This implementation uses a two-point Gaussian quadrature.

For the interior edges, the flux is evaluated using the Roe-averaged variables based on the left and right states adjacent to the edge and substituting these into Eq. (15). The boundary edges are evaluated similarly, but the right state is replaced by an appropriate boundary condition.

There are five possible boundary conditions implemented into QUODDY implementation of the boundary condition for each case described below.

1. **Land:** Zero normal flow boundary condition is applied to external boundaries.
2. **Island:** Zero normal flow boundary condition is applied to internal boundaries.

Land and island boundaries are zero normal flow boundaries. At these boundaries, the Roe-averaged normal velocity is set to zero. This is accomplished by setting the right value of the normal velocity equal to the negative of the left value and setting the right value of the elevation equal to the left value, i.e.,

$$\begin{aligned} u_{nR} &= -u_{nL} \\ H_R &= H_L \end{aligned} \quad (24)$$

at land and island boundaries.

3. **Non-zero normal velocity:** Value of velocity is specified at given boundary.

At non-zero normal velocities, the boundary specified normal velocity replaces the right value as

$$u_{nR} = u_{n,BC}. \quad (25)$$

4. **Geostrophic outflow:** Elevation is specified at the given boundary.

5. **Surface elevation:** Elevation is specified at the given boundary.

Since geostrophic outflow and surface elevation boundary conditions both specify elevation, they are handled the same way in the DG formulation. The specified surface elevation at the boundary replaces the right value on elevation prescribed edges as

$$H_R = H_{BC}. \quad (26)$$

The velocity on the boundary is found by a conservation of mass constraint, such that

$$u_{nR} = \left(\frac{H_L}{H_{BC}} \right) u_{n,BC}. \quad (27)$$

4.4 CFL Condition

The explicit formulation of the DG method restricts the time step via a Courant-Friedrichs-Levy (CFL) condition. The courant number is defined for each edge k and the time step is limited by the maximum courant number in the mesh. The courant number is defined as

$$C_k = \frac{\Delta t |\bar{\lambda}|_{k,\max} L_k}{\min(A_L, A_R)} \leq 1 \quad \forall \Gamma_k \in \Omega, \quad (28)$$

where $|\bar{\lambda}|_{k,\max}$ is the maximum absolute value of $\bar{\lambda}_i$, $i = 1, 2, 3$ on each edge.

4.5 Slope Limiting

The stability of the DG method cannot be ensured without incorporating some type of slope limiting into the algorithm. Two-dimensional slope limiters are a subject of current research. Cockburn and Shu [14] develop a total variation-bounded (TVB) slope limiter valid for linear approximations on rectangles and triangles with certain restrictions. Hoteit et al. [19] propose a new slope limiter they claim achieves limiting without introducing excessive numerical viscosity. In that work, they also review the slope limiter of Cockburn and Shu. In this effort, the slope limiter of Cockburn and Shu is currently being

implemented into the algorithm and is briefly described below. The discussion follows closely the discussion presented in Hoteit et al. [19] and is presented here for completeness. For further details, the reader should consult the references.

Let K_o denote an arbitrary element and K_i , $i=1,2,3$ the neighbors of the element. Further, let b_i , $i = 0, 1, 2, 3$ denote position vectors of the centroids of the elements, and m_i denote the position vectors of the midpoints of the edges of element K_o . This is illustrated in Fig. 2.

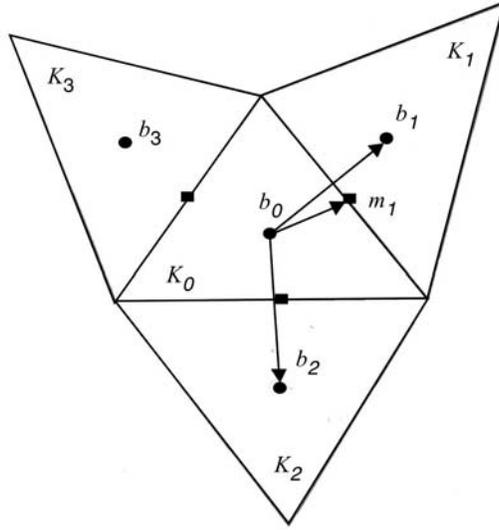


Fig. 2 — Triangle nomenclature for slope limiting procedure of Cockburn and Shu

For any edge m_1 , we can write

$$m_1 - b_0 = \alpha_1(b_1 - b_0) + \alpha_2(b_2 - b_0), \quad (29)$$

where (α_1, α_2) are constants that depend only on the geometry. Additionally, for any *linear* function u_h , we can write

$$u_h(m_1) - u_h(b_0) = \alpha_1(u_h(b_1) - u_h(b_0)) + \alpha_2(u_h(b_2) - u_h(b_0)). \quad (30)$$

The element average \bar{u}_h is the value of the function u_h evaluated at the element centroid. Additionally, we can define

$$\tilde{u}_h(m_1, K_0) = u_h(m_1) - \bar{u}(K_0), \quad (31)$$

which, by Eq. (30), can be written as

$$\tilde{u}_h(m_1, K_0) = \alpha_1(\bar{u}_{K_1} - \bar{u}_{K_0}) + \alpha_2(\bar{u}_{K_2} - \bar{u}_{K_0}) \equiv \Delta \bar{u}(m_1, K_0). \quad (32)$$

With the above definitions, the slope limiting proceeds as described below. First compute the quantity Δ_i for each edge of the element K_o as

$$\Delta_i = M(\tilde{u}_h(m_1, K_0), \nu(\Delta\bar{u}(m_1, K_0))), \quad (33)$$

where M is a *minmod* function defined by

$$M(a_1, a_2) = \begin{cases} s \min |a_i| & \text{if } s = \text{sign}(a_1) = \text{sign}(a_2) \\ 0 & \text{otherwise} \end{cases} \quad (34)$$

and ν is a constant greater than one. Cockburn and Shu use $\nu = 1.5$. With Δ_i computed by Eq. (33), there are two possibilities:

1. If $\sum_{i=1}^3 \Delta_i = 0$, then

$$\Delta\Pi u_h = \bar{u}_{K_0} + \sum_{i=1}^3 \Delta_i \phi_i(x, y), \quad (35)$$

where $\Delta\Pi u_h$ is the slope-limited value of u_h and $\phi(x, y)$ are the basis functions.

2. If $\sum_{i=1}^3 \Delta_i \neq 0$, compute the following:

$$\begin{aligned} \text{pos} &= \sum_{i=1}^3 \max(0, \Delta_i) & \text{neg} &= \sum_{i=1}^3 \max(0, -\Delta_i) \\ \theta^+ &= \min\left(1, \frac{\text{neg}}{\text{pos}}\right) & \theta^- &= \min\left(1, \frac{\text{pos}}{\text{neg}}\right) \\ \hat{\Delta}_i &= \theta^+ \max(0, \Delta_i) - \theta^- \max(0, -\Delta_i). \end{aligned} \quad (36)$$

Then set

$$\Delta\Pi u_h = \bar{u}_{K_0} + \sum_{i=1}^3 \hat{\Delta}_i \phi_i(x, y). \quad (37)$$

The limiting operator locally conserves mass and guarantees that the reconstructed gradient of $\Delta\Pi u_h$ is not greater than that of u_h .

5. IMPLEMENTATION INTO QUODDY5

The details of the numerical implementation of the DG algorithm are described below.

5.1 Numerical Algorithm

QUODDY contains a subroutine called ELEVATIONQ5 that computes the depth H using the GWCE and a continuous Galerkin finite-element approximation. The numerical implementation of the DG method replaces ELEVATIONQ5 with a subroutine ELEVATIONQ5DG, which serves as the driver program to compute the depth using the DG method. The subroutine calls additional subroutines to complete the computations described in Section 4.

5.2 Projection to and from Nodes

The input for QUODDY5 provides values of the dependent variables at the nodes. However, the DG method 1 requires the degrees of freedom to be defined in terms of the element average and the derivative with respect to x and y . Additionally, once the depth is computed in terms of the DG degrees of freedom, they must be projected back to the nodes so that they can be used to compute the velocities using the continuous Galerkin method.

The projection from the nodes is found by solving the 3×3 matrix,

$$\begin{bmatrix} 1 & (x_1 - \bar{x}_j) & (y_1 - \bar{y}_j) \\ 1 & (x_2 - \bar{x}_j) & (y_2 - \bar{y}_j) \\ 1 & (x_3 - \bar{x}_j) & (y_3 - \bar{y}_j) \end{bmatrix} \begin{bmatrix} H_{j0} \\ \frac{\partial H_j}{\partial x} \\ \frac{\partial H_j}{\partial y} \end{bmatrix} = \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix}, \quad (38)$$

for each element. Similar projections are used to compute the DG degrees of freedom for U and V .

After computing the depth using the DG method, it is necessary to project the DG degrees of freedom back onto the nodes. This is not as straightforward as projecting from the nodes, since each node can be connected to any number of elements. The projection back onto the nodes is found by computing the value of the depth at the node using the DG degrees of freedom for each element attached to the node, and then dividing by the total number of elements attached to the node. The algorithm proceeds as follows. Let e_k denote an element attached to node n , and $H_n(e_k)$ the elevation computed at node n using the basis functions on e_k . If a total of K elements are attached to node n , then the average value H_n at the node is given by

$$H_n = \frac{1}{K} \sum_{k=1}^K H_n(e_k). \quad (39)$$

5.3 Input Modification

The DG implementation into QUODDY changes the input minimally. The idea was to make the implementation as transparent to the user as possible. Only one file, the *.inq* file, has been changed. A typical *.inq* is shown in Fig. 3. Only one line has been added to the file. The last line is a flag that allows the user to choose “GWCE” to use the original formulation or “DGSL” to choose the DG method. Within the code itself, anyplace where modification to existing code has been made, the lines *mjg_begin* and *mjg_end* have been placed between the modified code. Finally, a file called **quoddy_5.1.1_DG_SUBS.f** that contains the DG subroutines has been added to the existing suite of subroutines. This file must be added to the makefile and compiled with the existing code.

```

{Comment:}
bank150 barotropic tidal solution
{Case name:}
../bank150/bank150
{Coordinates: CARTESIAN/SPHERICAL}
CARTESIAN
{Boundary element incidence list:}
../bank150/bank150_elev.bel
{Initial condition file:}
COLD-START 01 01 1996 0.0
{Echo file:}
bank150_elev.CT.echo
{Simulation parameters:}
SI UNITS          [units]
1.00 1.00 1.00  [x, y, and z scaling factor]
43.500           [degree latitude]
10.0             [minimum depth]
01 01 1996 536544.0 [end date (d m y) and time (sec) of simulation]
15.65625         [time step (seconds)]
21               [number of vertical nodes]
GWCE             [GWCE or DGSL]

```

Fig. 3 — Typical *.inq* file

6. VALIDATION

The implementation is validated in the test cases described below.

6.1 Maximum Courant Number Verification

Before choosing a test case, the courant number restriction was verified for a trivial case in which the surface elevation is initially zero and zero elevation boundary conditions are imposed. The elevation throughout the domain should remain zero for all $t > 0$. The Gulf of Maine mesh is used in this case. The maximum courant number for each element is shown in Fig. 4 along with the instability in the solution after 40 time steps, using a time step $\Delta t = 25$ s. If the time step is reduced so that the maximum courant number in the mesh is below unity, the solution remains stable.

The implementation was tested on a case provided by the developers of QUODDY. The test case is the Gulf of Maine with periodic tidal (elevation) boundary conditions applied on all boundaries. The mesh and bathymetry are shown in Fig. 5. The depth is in meters.

6.2 Gulf of Maine with Elevation Boundary Conditions

The elevation was computed using both the DG and GWCE methods. The tidal boundary conditions were ramped up over three tidal periods and a total of 10 tidal periods were computed. The slopes of the total depth are held constant at their initial values in the computations. A less restrictive slope limiter developed by Cockburn and Shu is currently being implemented into the algorithm.

The elevations computed using the DG and GWCE methods are shown at 3 days along with the differences in elevation in Fig. 6. The computed elevation solutions are very similar and generally within 1% to 2%. The corresponding velocity vectors are shown in Fig. 7. The velocity vectors are also quantitatively very similar.

6.3 Discussion of Different Methods Investigated

Two methods of computing the degrees of freedom per element were investigated. In the first, the element degrees of freedom were the element average and gradient and in the second, they were the values at the element vertices. It was found that each method produced similar results, although, in the second method, computing elevation boundary conditions was facilitated by simply setting the value at the given boundary node to the value given by the prescribed boundary condition. Additionally, the element vertices method produced a simpler mass matrix that reduced the computational effort required. In addition to the methods investigated, Cockburn and Shu [15] used the element edge midpoints as the degrees of freedom. This has the advantage of producing a diagonal mass matrix for each element. However, in any case, the mass matrix is a 3×3 matrix that is easily inverted by hand, so this is not a great advantage, and Hoteit et al. [19] noted that using the midpoints of the edges produces “excessive smearing” of the solution.

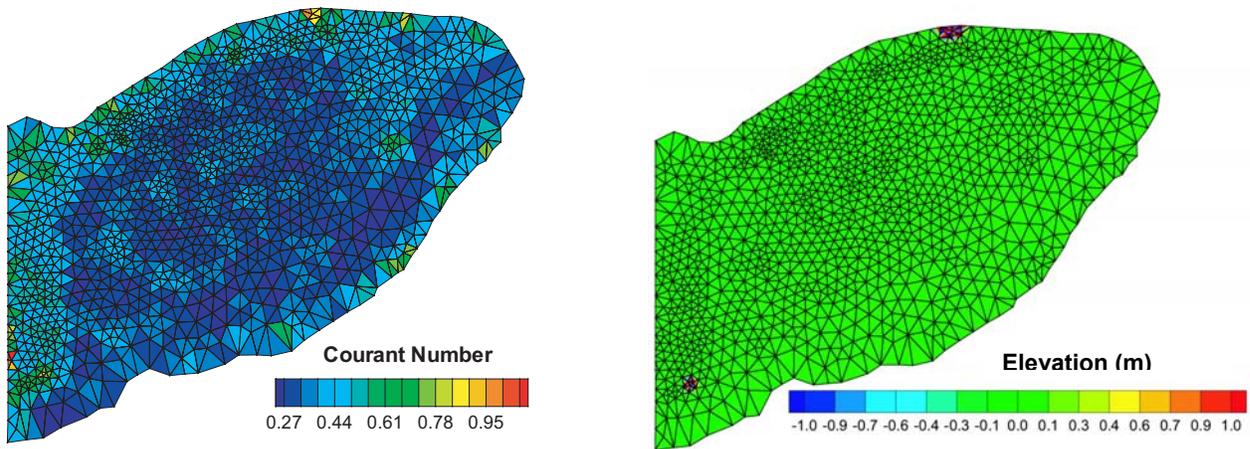


Fig. 4 — Courant number (left) and instability in elevation solution (right) after 40 time steps, $\Delta t = 25$

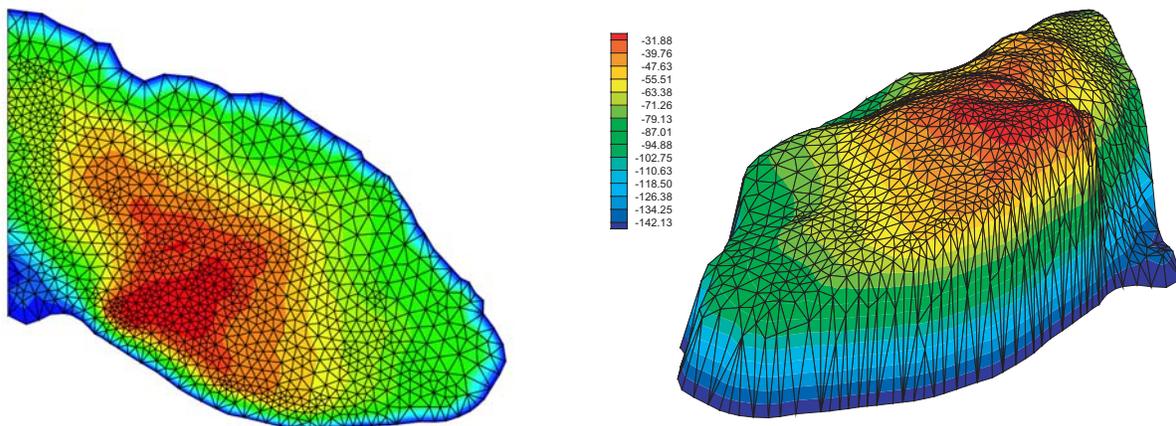


Fig. 5 — Gulf of Maine mesh and bathymetry (left) plan view and (right) 3-D view

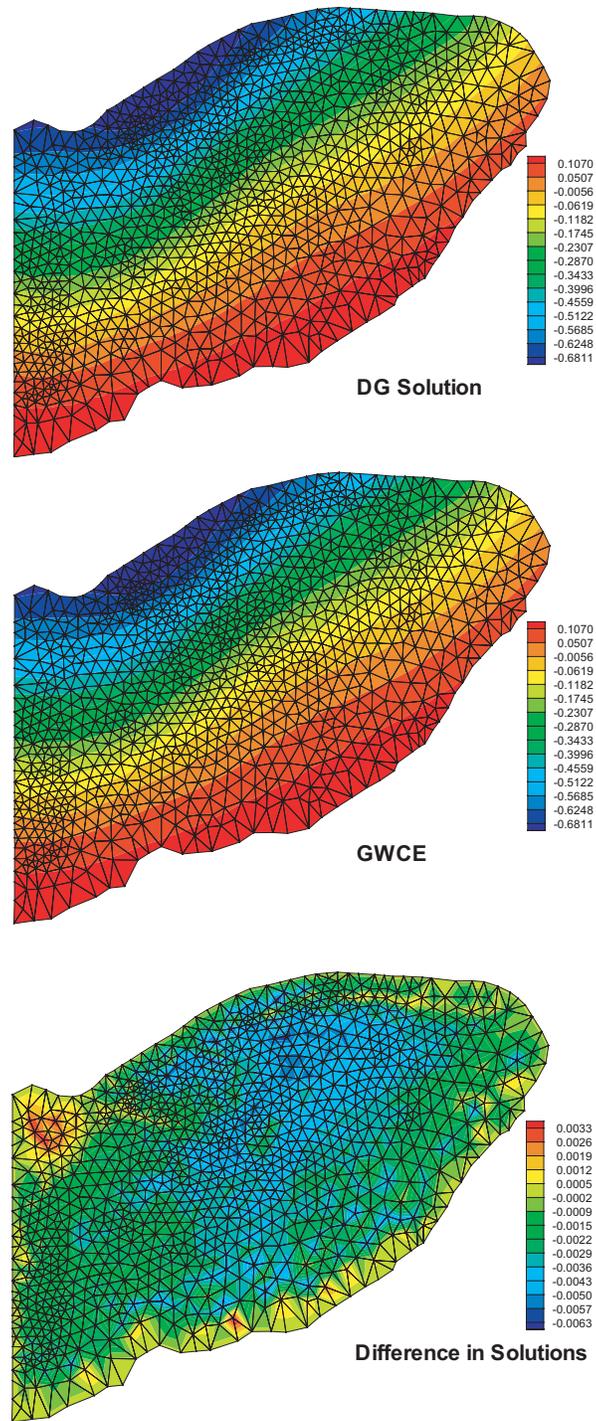


Fig. 6 — Elevation solution in meters at $t = 3$ days using DG and GWCE methods, and difference in solutions

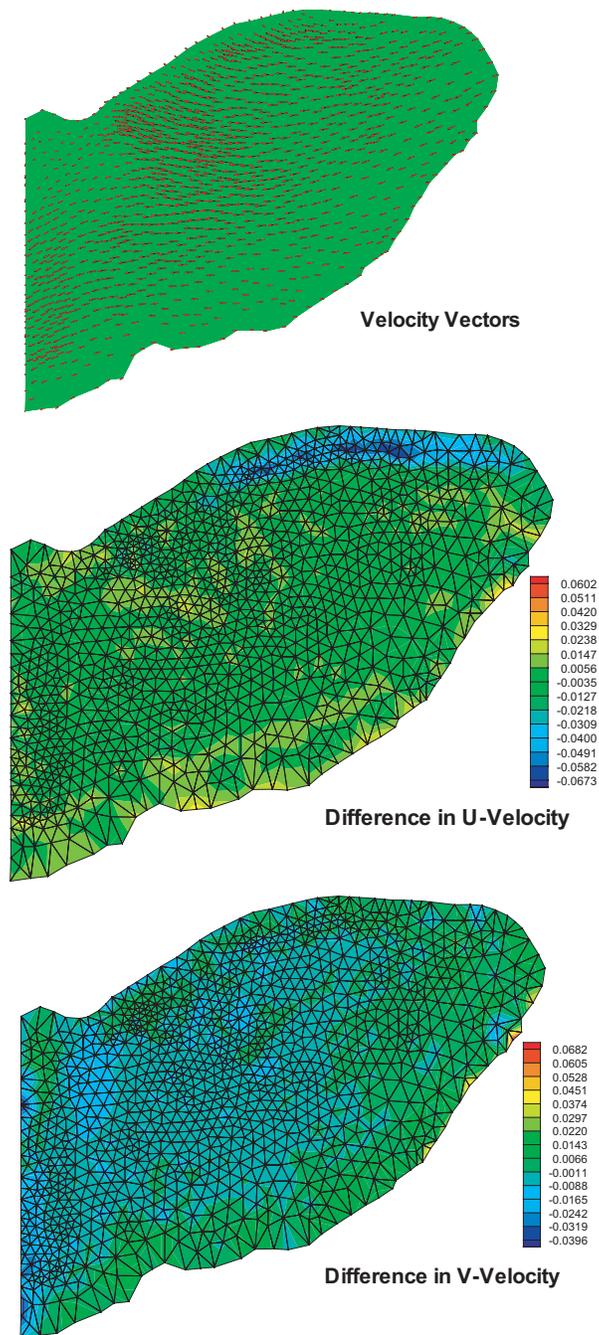


Fig. 7 — Velocity solution in m/s at $t = 3.0$ days using DG and GWCE methods, and difference in solutions

Two different second-order Runge-Kutta methods were also investigated. It was found that Henn's method, described by Eq. (17), produced somewhat more stable results than the method described by Eq. (16).

While both methods produced unstable results if the slopes were not restricted, the instability using Henn's method grew at a slower rate than it did using the other method.

7. CONCLUSIONS

A discontinuous Galerkin method was implemented into QUODDY5 to compute the depth using the primitive conservation of mass equation, replacing the GWCE formulation in the code. Several variations were investigated and the advantages and disadvantages of each were discussed.

The explicit formulation of the DG method limits the time step because of a CFL condition as compared to the semi-implicit formulation of the GWCE method. This time-step restriction is problem and mesh dependent, but for the Gulf of Maine case, the maximum allowable time step of the DG method was approximately 1/10 of the time step using the GWCE method, resulting in longer run times for the same problem.

The DG method requires a slope limiter to ensure stability of the solution. The TVB slope limiter of Cockburn and Shu is currently being implemented into the code.

8. RECOMMENDATIONS

Future work in this area should include developing a fully DG version of QUODDY. This will facilitate a comparison of the DG method using the primitive conservation of mass equation with the continuous Galerkin method using the GWCE method. Since strengths of the DG method include ease of parallelization and h-p adaptivity, longer term goals should include developing a parallel version of the method including dynamic mesh adaptivity.

REFERENCES

1. D. Lynch and F. Werner, "Three Dimensional Hydrodynamics on Finite Elements. Part II: Non-Linear Time-Stepping Model," *Internat. J. Numer. Methods Fluids* **12**, 507-533, 1991.
2. Lynch, D.R. and Gray, W.G., "A Wave Equation Model for Finite Element Tidal Computations," *Comp. Fluids* **7**(3), 207-228, 1979.
3. R.A. Luetlich, J.J. Westerink, and N.W. Scheffner, "ADCIRC, An Advanced Three-dimensional Circulation Model for Shelves, Coasts and Estuaries, Report 1: Theory and Methodology of ADCIRC-2DDI and ADCIRC-3DL," USACE Waterways Experiment Station Technical Report DRP-92-6, Vicksburg, MS, 1992.
4. F. Bassi and S. Rebay, "A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier-Stokes Equations," *J. Comput. Phys.* **131**, 267-279, 1997.
5. B. Cockburn and C.W. Shu, "The Local Discontinuous Galerkin Method for Convection-diffusion Systems," *SIAM J. Numer. Anal.* **35**, 2440-2463, 1998.
6. J.T. Oden, I. Babuska, and C.E. Baumann, "A Discontinuous HP Finite Element Method for Diffusion Problems," *J. Comput. Phys.* **146**, 491-519, 1998.

7. M.F. Wheeler, "An Elliptic Collocation-Finite Element Method with Interior Penalties," *SIAM J. Numer. Anal.* **15**(1), 152-161, 1978.
8. J. Douglas, Jr. and T. Dupont, "Interior Penalty Procedures for Elliptic and Parabolic Galerkin Methods," *Lecture Notes in Physics, Vol. 58* (Springer-Verlag, Berlin, 1976).
9. B. Riviere, M.F. Wheeler, and V. Girault, "Improved Energy Estimates for Interior Penalty, Constrained and Discontinuous Galerkin Methods for Elliptic Problems. Part I," *Comp. Geosciences* **3**, 337-360, 1999.
10. B. Riviere, M.F. Wheeler, and V. Girault, "A Priori Error Estimates for Finite Element Methods Based on Discontinuous Approximation Spaces for Elliptic Problems," *SIAM J. Numer. Anal.* **39**(3) 902-931, 2001.
11. B. Cockburn and P.A. Gremaud, "A Priori Error Estimates for Numerical Methods for Scalar Conservation Laws: Part I: The General Approach," *Math. Comp.* **65**(214), 533-573, 1996.
12. B. Cockburn and C.W. Shu, "TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Scalar Conservation Laws. Part II: General Framework," *Math. Comp.* **52**(186), 411-435, 1989.
13. B. Cockburn, S.Y. Lin, and C.W. Shu, "TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws, Part III: One-dimensional Systems," *J. Comput. Phys.* **84**(1), 90-113, 1989.
14. S. Hou, B. Cockburn, and C.W. Shu, "The Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws, Part IV: The Multidimensional Case," *Math. Comp.* **54**(190), 545-581, 1990.
15. B. Cockburn and C.W. Shu, "The Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws, Part V: Multi-dimensional Systems," *J. Comput. Phys.* **141**, 199-224, 1998.
16. D. Arnold, F. Brezzi, C. Cockburn, and D. Marini, "Discontinuous Galerkin Methods for Elliptic Problems, First International Symposium on Discontinuous Galerkin Methods," B. Cockburn, G.E. Karniadakis, and C.W. Shu, eds., *Lecture Notes in Computational Science and Engineering, Vol. 11* (Springer-Verlag, Berlin, 2000), pp. 89-101.
17. Proceedings, First International Symposium on Discontinuous Galerkin Methods (B. Cockburn, G.E. Karniadakis, and C.W. Shu, eds.), *Lecture Notes in Computational Science and Engineering, Vol. 11* (Springer-Verlag, Berlin, 2000).
18. R.J. Leveque, "Numerical Methods for Conservation Laws," *Lectures in Mathematics*, ETH, Zurich, 1990.
19. H. Hoteit, P. Ackerer, R. Mosé, J. Erhel, and B. Philippe, "New Two-Dimensional Slope Limiters for Discontinuous Galerkin Methods on Arbitrary Meshes," <http://www.inria.fr/rrrt/rr-4491.html>.