Texas Water Development Board
Project Number 0704830670

Final Report: UTBEST3D hydrodynamic model

PI: Clint Dawson (The University of Texas at Austin)
E-mail: clint@ices.utexas.edu
Voice: (512) 475-8625

Team Members:

Clint Dawson, Professor

Jennifer Proft, Research Scientist

Vadym Aizinger, Visiting Research Scientist

Mauricio Santillana, Graduate Research Assistant

Christopher Mirabito, Graduate Research Assistant

Contents

List of Figures

List of Tables

1 Executive Summary

The University of Texas Bay and Estuary 3D (UTBEST3D) simulator solves the shallow water equations using a discontinuous Galerkin (DG) finite element approach on unstructured meshes. The method is based on the use of discontinuous, piecewise polynomial approximating functions for each primary variable, defined over each element. The potential advantages of the DG method over more standard approaches include the ability to model flows at multiple scales, including resolution of long wave and advection-dominated flows, local (elementwise) mass conservation, and the ability to easily adapt the mesh and polynomial order locally. The flexibility of the code allows for both lower order and/or higher order polynomials to be used to approximate the solutions, by simply setting a parameter in the input file. The method is also highly scalable on parallel machines.

The research undertaken during this project year was focused on two tasks.

The first task was to further develop the model, which included additional testing and debugging of turbulence models which were incorporated during the last contract year, and the incorporation and testing of wind stress terms into the code, towards the purpose of making the code fully operational for modeling Texas coastal waters. A wetting and drying algorithm has also been formulated for the model. The algorithm is currently being tested in a two-dimensional version of the code [6]. When this testing is completed, we will incorporate the algorithm into UTBEST3D.

The second task was to perform comparison studies for benchmark problems against the ELCIRC and SELFE simulators, which are currently in use by staff at the TWDB. For this comparison, we chose a well-known test case modeling tidal flow in a quarter annular harbor, where an M2 tide is imposed on an open ocean boundary. This problem is featured as a benchmark on the ELCIRC web site. The ELCIRC and SELFE models produce elevation and velocity output at each time step in the simulation; therefore for ease of comparisons we compared the time history of elevation between these two models with output from UTBEST3D at specific points throughout the domain. Grid convergence studies were also performed. Additional comparisons can and should be performed in future studies. This test case is barotropic, therefore we did not compare results for transported quantities such as salinity. Furthermore, ELCIRC and SELFE do not have options to perform harmonic analysis, so we have not performed analysis of the harmonic constituents of the solutions. In addition, ELCIRC and UTBEST3D are formulated to be mass conservative; however, possible mass balance errors in SELFE should be examined in future studies.

The findings of the project can be summarized as follows. The code allows for eight different options for specifying vertical viscosity, including five different turbulent models. All turbulent models have been tested and appear to be fully debugged. Wind stress has been incorporated into the code, and preliminary results on Galveston Bay indicate that wind direction is an important effect in modeling, e.g., salinity transport in the bay. Furthermore, comparisons between UTBEST3D, ELCIRC and SELFE on tidal flow in the quarter annular harbor indicate that the codes produce similar tidal responses. The grid convergence studies showed that UTBEST3D exhibited convergence with respect to the grid for higher-order (linears and quadratics) approximations. SELFE and UTBEST3D solutions showed good agreement as the grid is refined. However, the accuracy of ELCIRC appears to be comparable to that of the lowest order approximation (constants) in UTBEST3D. ELCIRC and SELFE are slightly more efficient with respect to CPU time, because a larger time step can be used. How these results extend to more realistic scenarios will be the subject of future studies.

2 Model Description

2.1 Introduction

In this report, we discuss recent improvements and extensions made to the UTBEST3D (University of Texas Bay and Estuary 3D) simulator, which has been developed at UT Austin by the investigators. These improvements include fully debugging and testing five different turbulence options in the code for improved modeling of vertical viscosity, particularly for turbulent baroclinic flows, and the incorporation of wind stress terms in the model. In addition, we report on recent comparisons between UTBEST3D two well-known simulators, ELCIRC and SELFE, on a benchmark problem.

The UTBEST3D model has been under development since 2003. The original motivation for this work was that, despite many recent advances in the development of large-scale simulators for modeling circulation in oceanic to continental shelf, coastal and estuarine environments, the search is still on for methods which are locally mass conservative, can handle very general types of elements, and are stable and higher-order accurate under highly varying flow regimes. Algorithms such as the DG method are of great interest within the ocean and coastal modeling communities. DG methods are promising because of their flexibility with regard to geometrically complex elements, use of shock-capturing numerical fluxes, adaptivity in polynomial order, ability to handle nonconforming grids, and local conservation properties; see [10] for a historical overview of DG methods. Recent studies in two dimensions [6] show that the DG method can easily handle wetting and drying. Since the method is local to an element, wetting and drying heuristics are easily incorporated at the element level.

In [2, 9], we investigated DG and related finite volume methods for the solution of the two-dimensional shallow water equations. Viscosity (second-order derivative) terms are handled in this method through the so-called local discontinuous Galerkin (LDG) framework [12], which employs a mixed formulation. Application of the methodology to three-dimensional shallow water models was described in [17, 3]. The 3D formulation is not a straightforward extension of the two-dimensional algorithm. In particular, it uses a special form of the continuity equation for the free surface elevation and requires postprocessing the elevation solution to smooth the computational domain.

Under funding from the TWDB from 2005-2007 and from other sources, we have taken UTBEST3D from a purely research code to a more fully developed simulator, with the goal of making the code operational with respect to all features important to modeling Texas coastal waters. These extensions include adding salinity and temperature transport in 2005, incorporating various one and two-equation turbulence models in 2006, and incorporating wind stress terms in 2007. Additional extensions, such as wetting and drying and parallel implementation, are underway.

The rest of this report is organized as follows. In Sections 2.2-2.4, we discuss the model equations, boundary conditions, and our assumptions about the time-varying computational domain. In Section 2.5, we outline the DG method for a simple advection equation, with the purpose of explaining the major ideas behind the approach. The full method applied to the 3D system is described in an appendix to this report. In Sections 2.6–2.8, we discuss the species and turbulent transport equations which are implemented in the model and the baroclinic option in the code. In Section 3, we describe some of the numerical experiments carried out under the project. Finally, we give conclusions and recommendations for future work.

2.2 Model and assumptions

For $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d, \mathbf{c} \in \mathbb{R}^e$, we denote by $\mathbf{ac}$ the tensor-product of $\mathbf{a}$ and $\mathbf{c}$ and by $\mathbf{a} \cdot \mathbf{b}$ the dot-product of $\mathbf{a}$ and $\mathbf{b}$.

Let $\Omega(t) \subset \mathbb{R}^3$ be the time-dependent domain. We assume the top boundary of the domain $\partial\Omega_{top}(t)$ is the only moving boundary. The bottom $\partial\Omega_{bot}$ and lateral $\partial\Omega_D(t)$ boundaries are assumed to be fixed (though the height of the lateral boundaries can vary with time according to the movements of the free surface). We also require the lateral boundaries to be strictly vertical (see Figure 1). The last requirement is only needed to assure that the horizontal cross-section of the domain $\Omega(t)$ (denoted by Ω_{xy}) doesn't change with time.

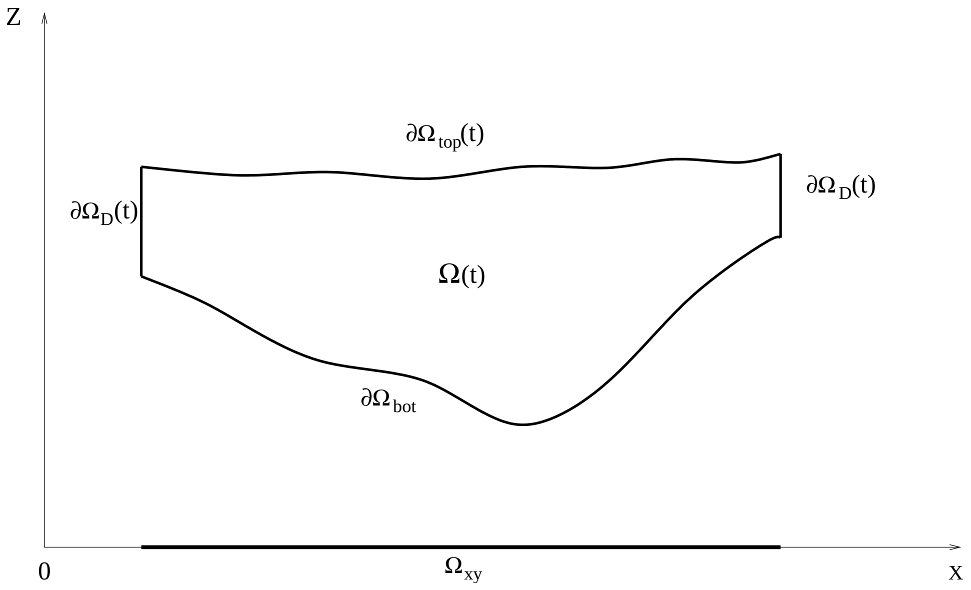
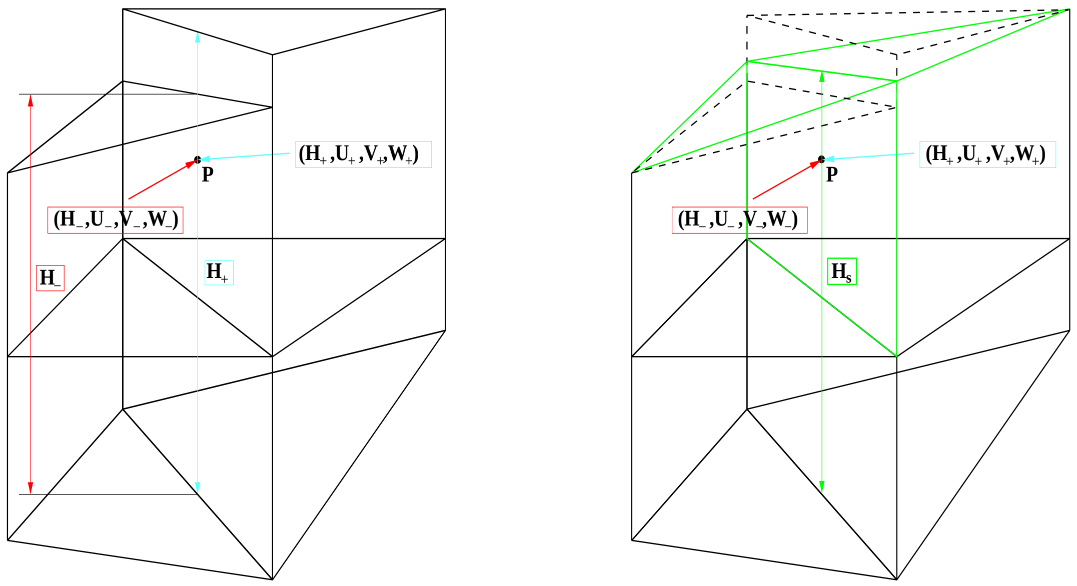


Figure 1: Vertical cross-section of the computational domain $\Omega(t)$.

Keeping in line with the specific anisotropy of $\Omega(t)$ we construct a 3D finite element mesh by extending a 2D triangular mesh of Ω_{xy} in the vertical direction, thus producing a 3D mesh of $\Omega(t)$ that consists of one or more layers of prismatic elements. In order to better reproduce the bathymetry and the free surface elevation of the computational domain we do not require top and bottom faces of prisms to be parallel to the xy-plane, although the lateral faces are required to be strictly vertical.

For a point $(x, y) \in \Omega_{xy}$ we denote by $z_b(x, y)$ the value of the z-coordinate at the bottom of the domain and by $\xi_s(t, x, y)$ at the top. A key feature of the 3D LDG model is the fact that all primary variables, including the free surface elevation, are discretized using discontinuous polynomial spaces. As a result, computed values of the free surface elevation may have jumps across inter-element boundaries. If the finite element grids were to follow exactly the computed free surface elevation field this would cause the elements in the surface layer to have mismatching lateral faces (staircase boundary). We avoid this difficulty by employing a globally continuous free surface approximation that is obtained from the computed values of the free surface elevation ξ with the help of a smoothing algorithm (see Figure 2). Thus H is the computed height of the water column, and H_s is the postprocessed height.

It must be noted here that solely the computational mesh is modified by the smoothing algorithm whereas the computed (discontinuous) approximations to all unknowns, including the free surface elevation, are left unchanged. This approach preserves the local conservation property of the LDG method and is essential for our algorithm's stability.



a) Before surface mesh smoothing b) After surface mesh smoothing

Figure 2: Illustration of mesh smoothing.

2.3 System of 3D shallow water equations

The momentum equations in conservative form (assuming constant density) are given by [27]

$$\partial_t \mathbf{u}_{xy} \ + \ \nabla \cdot (\mathbf{u}_{xy}\mathbf{u} \ - \ \mathcal{D}\nabla \mathbf{u}_{xy}) \ + \ g\nabla_{xy}\xi \ - \ f_c \mathbf{k} \times \mathbf{u}_{xy} \ = \ \mathbf{F}, \tag{1}$$

where the wind stress, the atmospheric pressure gradient, and the tidal potential are combined into a body force term $\mathbf{F}$, $\nabla_{xy} = (\partial_x, \partial_y)$, ξ is the value of the z coordinate at the free surface, $\mathbf{u} = (u, v, w)$ is the velocity vector, $\mathbf{u}_{xy} = (u, v)$ is the vector of horizontal velocity components, f_c is the Coriolis coefficient, $\mathbf{k} = (0, 0, 1)$ is a unit vertical vector, g is acceleration due to gravity, and $\mathcal{D}$ is the tensor of eddy viscosity coefficients defined as follows:

$$\mathcal{D} = \begin{pmatrix} D_u & 0 \\ 0 & D_v \end{pmatrix}, \tag{2}$$

with D_u, D_v 3×3 symmetric positive-definite matrices, and $\mathcal{D}\nabla\mathbf{u}_{xy} = \begin{pmatrix} D_u \nabla u \\ D_v \nabla v \end{pmatrix}$. In particular,

$$D_u \ = \ D_v \ = \ \begin{pmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & \nu_t \end{pmatrix},$$

where A_x, A_y are the horizontal and ν_t is the vertical eddy viscosity coefficient.

The continuity equation is

$$\nabla \cdot \mathbf{u} \ = \ 0. \tag{3}$$

2.4 Boundary conditions

The following boundary conditions are specified for the system:

- At the bottom boundary $\partial\Omega_{bot}$, we have no normal flow

$$\mathbf{u}(z_b) \cdot \mathbf{n} = 0 \tag{4}$$

and the quadratic slip condition for the horizontal velocity components

$$\nu_t \frac{\partial u}{\partial z}(z_b) = C_f \sqrt{u^2(z_b) + v^2(z_b)}\, u(z_b), \tag{5}$$
$$\nu_t \frac{\partial v}{\partial z}(z_b) = C_f \sqrt{u^2(z_b) + v^2(z_b)}\, v(z_b), \tag{6}$$

where $\mathbf{n} = (n_x, n_y, n_z)$ is an exterior unit normal to the boundary.

- The free surface boundary conditions have the form

$$\partial_t \xi \ + \ u(\xi)\ \partial_x \xi \ + \ v(\xi)\ \partial_y \xi \ - \ w(\xi) \ = \ 0, \tag{7}$$

and

$$\nabla u(\xi) \cdot \mathbf{n} \ = \ \nabla v(\xi) \cdot \mathbf{n} \ = \ 0 \tag{8}$$

in the case of no wind. In the presence of wind forcing however, the last equation is replaced by

$$\nu_t \frac{\partial \mathbf{u}}{\partial z}(\xi) = \tau_{\mathbf{s}}, \tag{9}$$

where $\tau_{\mathbf{s}}$ is the surface stress which can be specified directly or computed from the wind velocity at 10m above the water surface, $\mathbf{U}_{10}$, by

$$\tau_{\mathbf{s}} \ = \ \frac{\rho_a}{\rho_0} C_s |\mathbf{U}_{10}| \mathbf{U}_{10} \tag{10}$$

with $C_s = 10^{-3}(A_{W1} + A_{W2}|\mathbf{U}_{10}|)$ for $U_{low} \leq |\mathbf{U}_{10}| \leq U_{high}$ and C_s held constant at the extremal values outside of this interval. Similarly to [31] we set $A_{W1} = 0.1$, $A_{W2} = 0.063$, $U_{low} = 6\ m/s$, $U_{high} = 50\ m/s$.

On the lateral boundaries, we consider several common types of boundary conditions:

- Land boundary: No normal flow

$$u_{\mathbf{n}} = \mathbf{u} \cdot \mathbf{n} = 0, \tag{11}$$

and zero shear stress

$$\nabla u_{\boldsymbol{\tau}} \cdot \mathbf{n} \ = \ 0, \tag{12}$$

where $\boldsymbol{\tau}$ and $\mathbf{n}$ denote a unit tangential and a unit exterior normal vectors to the boundary, correspondingly.

- Open sea boundary: Zero normal derivative of the horizontal velocity components

$$\nabla u \cdot \mathbf{n} \ = \ \nabla v \cdot \mathbf{n} \ = \ 0, \tag{13}$$

and prescribed surface elevation $\xi_{os}(x, y, t)$

$$\xi = \xi_{os}(x, y, t). \tag{14}$$

- River boundary: Prescribed velocities

$$\mathbf{u} = \mathbf{u}_r, \tag{15}$$

  and prescribed surface elevation

$$\xi = \xi_r. \tag{16}$$

- Radiation boundary: Zero normal derivative of the horizontal velocity components

$$\nabla u \cdot \mathbf{n} \;=\; \nabla v \cdot \mathbf{n} \;=\; 0. \tag{17}$$

Analytically, the free surface elevation can be computed from (7). However, a computationally more robust method [27] is obtained by integrating continuity equation (3) over the total height of the water column. Taking into account boundary conditions (4) – (7) at the bottom and top boundaries we arrive at a 2D equation for the free surface elevation commonly called the primitive continuity equation (PCE),

$$\partial_t \xi \;+\; \partial_x \int_{z_b}^{\xi} u dz \;+\; \partial_y \int_{z_b}^{\xi} v dz \;=\; 0. \tag{18}$$

2.5 An overview of the DG method

In this section, we describe the DG method for a simple advection equation in one space dimension. A complete description of the particular DG method used to discretize the 3D model described above is given in the appendix to this document.

Consider the one dimensional advection equation

$$u_t + cu_x = 0, \quad -\infty < x < \infty, \quad t > 0 \tag{19}$$

where c is a constant. We assume an initial condition $u(x,0) = u^0(x)$ is also specified. Partition the real line into intervals $I_j = [x_j, x_{j+1}]$ of length h_j. Multiply (19) by a test function w and integrate over I_j:

$$\int_{I_j} [u_t + cu_x] w dx = 0 \tag{20}$$

and integrate the second term by parts:

$$\int_{I_j} u_t w dx - \int_{I_j} cuw_x + cuw|_{x_j}^{x_{j+1}} = 0. \tag{21}$$

Next we choose test and trial spaces for u and w consisting of polynomials of degree $\leq k$ defined on each element I_j. Thus

$$u|_{I_j} \approx u_h|_{I_j} \equiv \sum_{j=0}^{k} u_j(t) P^j(x)$$

where $\{P^j(x), j = 0, \ldots, k\}$ represents some linearly independent set which spans all polynomials of degree zero up to k defined on I_j. These could be chosen to be, for example, the monomials $\{1, x, x^2, \ldots, x^k\}$, the usual Lagrange polynomials of degree k, or the set of Legendre polynomials of degree $\leq k$. Thus, u_h and w are both contained in the span of $\{P^j\}$, and we further note that no inter-element continuity is required on u_h or w. Therefore, there is an ambiguity in the boundary terms in (21). We resolve this ambiguity through the concept of *upwinding.* Let

$$w^+(x_j) = \lim_{x \to x_j^+} w(x) \tag{22}$$

$$w^-(x_j) = \lim_{x \to x_j^-} w(x) \tag{23}$$

denote the right and left limits of w at the point x_j. Define, the upwind value of u_h by

$$u_h^{\mathrm{up}}(x_j) = \begin{cases} u_h^-(x_j), & \text{if } c > 0 \\ u_h^+(x_j), & \text{if } c < 0 \end{cases} \tag{24}$$

Substituting into (21), we arrive at

$$\int_{I_j} \frac{\partial u_h}{\partial t} w dx - \int_{I_j} c u_h w_x + c[u_h^{\mathrm{up}}(x_{j+1}) w^-(x_{j+1}) - u_h^{\mathrm{up}}(x_j) w^+(x_j)] = 0. \tag{25}$$

Finally, we integrate (25) in time. This can be accomplished using standard, explicit Runge-Kutta methods. Normally, one chooses a method of equal order to the spatial order of the scheme. Therefore, if polynomials of degree k are used to approximate u, then a Runge-Kutta method of order k is used to integrate (25) in time. The particular Runge-Kutta methods used in UTBEST3D are discussed in the appendix.

The basic ideas of the DG method can now be extended to a system of *conservation laws*,

$$u_t + f(u)_x = 0. \tag{26}$$

We take the vector product with a test function w and integrate by parts,

$$\int_{I_j} u_t \cdot w dx - \int_{I_j} f(u) \cdot w_x + f(u) \cdot w|_{x_j}^{x_j+1} = 0. \tag{27}$$

Approximating u by u_h as above (each component is a discontinuous, piecewise polynomial), we arrive at

$$\int_{I_j} \frac{\partial u_h}{\partial t} \cdot w dx - \int_{I_j} f(u_h) \cdot w_x + \hat{f}_{j+1} \cdot w^-(x_{j+1}) - \hat{f}_j \cdot w^+(x_j) = 0. \tag{28}$$

The difference is in the evaluation of the flux function f at the interface x_j, we have represented this quantity as $\hat{f}_j$ above. Here one utilizes more sophisticated ideas from the theory of conservation laws which generalize the upwinding concept, namely local *Riemann solvers.* We simply note here that $\hat{f}_j$ is constructed from the left and right states at the interface,

$$\hat{f}_j = \hat{f}(u_h^-(x_j), u_h^+(x_j)).$$

An excellent discussion of Riemann solvers for the shallow water equations is given in the book by R. LeVeque on conservation laws [22].

2.6 Species transport

Species transport equations for salinity and temperature are included in the model. Transport is described by advection-diffusion equations of the form

$$r_t + \nabla \cdot (\mathbf{u}r) - \nabla \cdot (K_r \nabla r) = f, \quad \Omega(t) \times (0, T), \tag{29}$$

where $r = S$ for salinity or $r = T$ for temperature, and $K_r = \begin{pmatrix} \tilde{A}_x & 0 & 0 \\ 0 & \tilde{A}_y & 0 \\ 0 & 0 & \nu_r \end{pmatrix}$ is a specified diffusion tensor. These equations must be supplemented with initial and boundary conditions. The DG method is also applied to the solution of these equations. The details are given in the appendix.

2.7 Wetting and drying

A wetting and drying algorithm is currently being implemented and tested in a two-dimensional version of the DG code, in collaboration with Shintaro Bunya, Ethan Kubatko and Joannes Westerink [6]. While we had hoped to implement this algorithm in UTBEST3D during this contract year, we are still testing the methodology on field applications and comparing to measured data and analytical solutions. As this work is still ongoing and not finalized, we have not yet tested the wetting and drying algorithm in UTBEST3D.

2.8 Baroclinic model

Introduction of the transport equations for temperature and salinity allows us to expand our model to include baroclinic effects. In order to preserve stability of our scheme – which heavily depends on correct treatment of the coupling between pressure and momentum – and to provide the ability to carry out barotropic as well as baroclinic simulations with the minimum code modification we implemented the baroclinic forcing effects in the momentum equation as a correction to the standard pressure term used in the momentum equations in the barotropic case. This correction accounts for the difference between the actual and the reference densities.

In the baroclinic case the momentum equations are given by

$$\partial_t \mathbf{u}_{xy} + \nabla \cdot (\mathbf{u}_{xy}\mathbf{u} - \mathcal{D}\nabla \mathbf{u}_{xy}) + g\nabla_{xy}\xi + \frac{g}{\rho_0}\nabla_{xy}\int_z^{\xi} (\rho(T, S, \xi - \tilde{z}) - \rho_0) d\tilde{z} - f_c \mathbf{k} \times \mathbf{u}_{xy} = \mathbf{F}, \tag{30}$$

where ρ_0 is the reference density, and $\rho(T, S, p)$ is the density computed from the equation of state.

The equation of state used in UTBEST3D is due to Klinger [21] and is given by

$$\rho(T, S, p) = C(p)\ \beta(p)S - \alpha(T, p)T - \gamma(T, p)(35 - S)T, \tag{31}$$

where

$$C = 999.83 + 5.053p - 0.048p^2, \tag{32}$$
$$\beta = 0.808 - 0.0085p, \tag{33}$$
$$\alpha = 0.0708(1 + 0.351p + 0.068(1 - 0.0683p)T), \tag{34}$$
$$\gamma = 0.003(1 - 0.059p - 0.012(1 - 0.064p)T). \tag{35}$$

p is the height of the water column above the point expressed in kilometers, T is the temperature in degrees Celsius, and S is the salinity in psu.

The discretization of the baroclinic forcing term in the momentum equations is described in the appendix.

2.9 Turbulence

The issue of realistic modeling of the turbulent eddy viscosity and diffusivity terms is an active research topic. The vertical eddy viscosity coefficient is a particularly important parameter if one aims to achieve good vertical resolution of the computational domain.

UTBEST3D provides vertical eddy viscosity models of various levels of computational and conceptual complexity. In order of increasing complexity those include a constant eddy viscosity coefficient, an algebraic (zeroth order) model, as well as one and two equation models.

- The simplest model amounts to explicitly specifying diagonal entries to the tensors of eddy viscosity/diffusivity coefficients for all variables in (1) and (67).
- Two algebraic models implemented in UTBEST3D are due to Davies [16] and give good results at a reasonable computational cost in cases where accurate vertical resolution of flow is not important.

  In the first algebraic model the eddy viscosity and diffusivity coefficients are set equal to $C_t \frac{(\bar{u}^2+\bar{v}^2)}{\omega_a}$, where $\bar{u}$ and $\bar{v}$ are depth averaged horizontal velocity components, $C_t = 2 \times 10^{-5}$ is a dimensionless coefficient, and ω_a a typical long wave frequency set to $10^{-4}s^{-1}$.

  Model two is very similar to model one, except that the eddy viscosity is assumed to be proportional to $H\sqrt{\bar{u}^2+\bar{v}^2}$.
- The first order vertical eddy viscosity closure model solves a transport equation for the turbulent kinetic energy in addition to the mass, momentum, and species transport equations.

$$k_t + \nabla \cdot (\mathbf{u}k) - \frac{\partial}{\partial z}(\nu_k \frac{\partial}{\partial z} k) = \nu_t \left(\left(\frac{\partial u}{\partial z}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 \right) + \nu_r \frac{g}{\rho_0}\frac{\partial \rho}{\partial z} - \epsilon, \tag{36}$$

where ν_k is the vertical diffusivity coefficient for k and $\epsilon = (C_\mu^0)^3 k^{\frac{3}{2}} l^{-1}$ is the dissipation rate of the turbulent kinetic energy. The turbulent mixing length l is computed algebraically in this model and is set equal to $l(z) \ = \ \kappa\ (z - z_b)\ \sqrt{\xi - z}\ F_l(Ri)$ (see Delft3D-Flow manual [18]). $C_\mu^0 = \sqrt{0.3}$ is a calibration constant, $\kappa = 0.4$ is the von Karman constant, and $F_l(Ri)$ is the damping function accounting for stratification effects. F_l depends on the gradient Richardson number

$$Ri \ = \ \frac{-\frac{g}{\rho_0}\frac{\partial \rho}{\partial z}}{\left(\frac{\partial u}{\partial z}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2} \tag{37}$$

and is of the form:

$$F_l(Ri) \ = \ \begin{cases} e^{-2.3Ri}, & Ri \geq 0, \\ (1 - 14Ri)^{0.25}, & Ri < 0. \end{cases} \tag{38}$$

Once k is computed one can obtain the vertical eddy viscosity and diffusivity coefficients by taking $\nu_t = C_\mu^0 k^{\frac{1}{2}} l$ and $\nu_r = \nu_k = \frac{\nu_t}{0.7}$ correspondingly. Neumann type boundary conditions for k are used at the free surface and the sea bed $\nu_k \frac{\partial k}{\partial n} = 0$.

The discretization of (36) is very similar to those of species transport equations (29) and is not shown here. The only essential consideration when computing the transport equation for the turbulent kinetic energy is the velocity shear term on the right hand side of (36). Because we lack velocity information at the free surface and the sea bed the auxiliary flux variables computed as in (56) tend to underestimate $\frac{\partial \mathbf{u}}{\partial z}$, particularly for lower order DG spaces. This leads to the turbulent kinetic energy going negative in areas with slow flow. To address this issue we compute velocity shear terms in elements adjacent to the bottom boundary (as well as in those near the top boundary if the wind forcing is present) utilizing the law of the wall:

$$\frac{\partial \mathbf{u}}{\partial z} = \frac{\mathbf{u}}{(z-z_b)\ln\frac{z-z_b}{z_0}}, \tag{39}$$

where z_0 is the bottom roughness coefficient. In the presence of wind forcing we use in the top layer:

$$\frac{\partial \mathbf{u}}{\partial z} = \frac{\sqrt{\tau_\mathbf{s}}}{\kappa(\xi - z)}. \tag{40}$$

- The second order closure model implemented in UTBEST3D is based on the generic turbulence length scale model proposed by Warner et al [28]. The main advantage of this formulation is the ability to switch between several two equation models, including $k-\epsilon$ and Mellor-Yamada, by changing a few constant parameters. In addition to the transport equation for k, this model includes a second transport equation for derived quantity ψ

$$\psi_t + \nabla\cdot(\mathbf{u}\psi) - \frac{\partial}{\partial z}(\nu_\psi \frac{\partial}{\partial z}\psi) = \frac{\psi}{k}\left(C_1\nu_t\left(\left(\frac{\partial u}{\partial z}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2\right) + C_3\nu_r\frac{g}{\rho_0}\frac{\partial \rho}{\partial z} - C_2\epsilon F_{wall}\right), \tag{41}$$

where $\psi = (C_\mu^0)^p k^m l^n$ and C_3 is equal to C_3^- for stably stratified flow and C_3^+ otherwise. Depending on the choice of p, m, and n we obtain different closure schemes. The turbulent mixing length is computed using k and ψ. The eddy viscosity and diffusivity coefficients are obtained from $\nu_t = \sqrt{2}S_m k^{\frac{1}{2}} l$ and $\nu_r = \sqrt{2}S_h k^{\frac{1}{2}} l$, where S_m and S_h are stability functions given by:

$$S_h = \frac{0.4939}{1-30.19G_h}, \qquad S_m = \frac{0.392 + 17.07S_hG_h}{1-6.127G_h}, \tag{42}$$

where $G_h = G_{h_u} - \frac{(G_{h_u}-G_{h_c})^2}{G_{h_u}+G_{h0}-2G_{h_c}}$ and $G_{h_u} = \min(G_{h0}, \max(-0.28, \frac{g}{\rho_0}\frac{\partial\rho}{\partial z}\frac{l^2}{2k}))$ with $G_{h0} = 0.0233, G_{h_c} = 0.02$.

To improve stability properties of the two-equation model we employ Neumann boundary conditions for ψ at the free surface $\nu_\psi\frac{\partial\psi}{\partial z} = -n\nu_\psi(C_\mu^0)^p k^m \kappa l_s^{n-1}$ and at the sea bed $\nu_\psi\frac{\partial\psi}{\partial z} = n\nu_\psi(C_\mu^0)^p k^m \kappa l_b^{n-1}$, where the turbulent mixing length is derived from the law of the wall: $l_s = l_{|\xi_s} = z_1 e^{\frac{\kappa|u|}{\tau_s}}$ and $l_b = l_{|z_b} = z_0\frac{\kappa}{\sqrt{Cf}}$ with C_f being as in (6) and z_1 the surface roughness coefficient.

Values of the parameters for four popular two equation models are shown in Table 1. Discretization of (41) is also done similarly to (29).

3 Numerical results

3.1 Wind stress

Surface stresses due to wind, as given in (10) above, have been added to the model and tested. The test case below models a release of a high concentration of saline in the area of the West Bay off of Galveston Island. The domain and finite element mesh are shown in Figure 3. The mesh

	Mellor-Yamada [23]	$k-\epsilon$ [8]	$k-\omega$ [30]	generic [25]
p	0	3	-1	2
m	1	1.5	0.5	2
n	1	-1	-1	2/3
ν_k	$\frac{\nu_t}{2.44}$	ν_t	$\frac{\nu_t}{2}$	$\frac{\nu_t}{0.8}$
ν_ψ	$\frac{\nu_t}{2.44}$	$\frac{\nu_t}{1.3}$	$\frac{\nu_t}{2}$	$\frac{\nu_t}{1.07}$
C_1	0.9	1.44	0.555	1
C_2	0.5	1.92	0.833	1.22
C_3^+	1	1	1	1
C_3^-	2.53	-0.52	-0.58	0.1
k_{min}	7.6e-6	7.6e-6	7.6e-6	7.6e-6
ψ_{min}	1e-8	1e-8	1e-8	1e-8
F_{wall}	$1+1.33\left(\frac{l}{z-z_b}\right)^2+0.25\left(\frac{l}{\xi_s-z}\right)^2$	1	1	1

Table 1: Generic turbulence closure model parameters.

consists of 3397 surface elements with up to 4 vertical layers, as seen in Figure 3. The following tidal forcing with time(t) in hours was imposed at the open sea boundary:

$$\begin{aligned}\hat{\xi}(t) &= 0.075\cos(\tfrac{t}{25.82}+3.40)\\ &+ 0.095\cos(\tfrac{t}{23.94}+3.60)\\ &+ 0.100\cos(\tfrac{t}{12.66}+5.93)\\ &+ 0.395\cos(\tfrac{t}{12.42}+0.00)\\ &+ 0.060\cos(\tfrac{t}{12.00}+0.75)\quad (meters).\end{aligned} \tag{43}$$

The tide was ramped up over a 2 day period, and the time step used in the simulation was 2 seconds, although additional experiments determined that a larger time step of 4 seconds gave similar results.

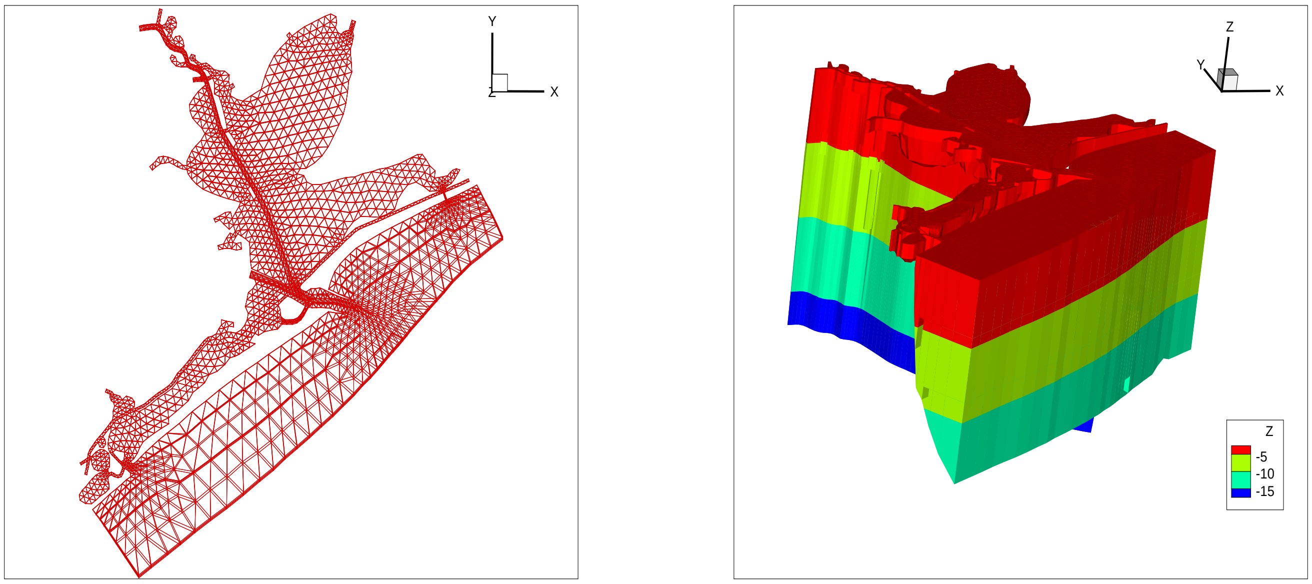
The background salinity in the model was assumed to be 35 ppt, with background temperature of 20 c. Additional salinity (for example, from a desalination plant) is introduced into the model by imposing a continuous inflow of salinity of $S_I = 70$ ppt near the location 29.2° N, 95.0° W. This is specified as an inflow boundary condition in the model; that is, in the saline transport equation, we impose an inflow flux condition

$$S\mathbf{u}\cdot\mathbf{n} = S_I\mathbf{u}\cdot\mathbf{n}, \tag{44}$$

when $\mathbf{u}\cdot\mathbf{n} < 0$, where $\mathbf{n}$ is the outward normal to the boundary at this location. The result is a plume of salinity which is transported through the domain. The simulations below examine the effect of wind on the extent and direction of this plume.

The first test case examines the movement of the salinity plume in the event of no or neglible wind stress. In Figure 4, we plot the salinity concentration at 6, 8, 9, 10, 11 and 12 days. In these figures we are zooming in on the region of the plume, which is in the lower West Bay, and we are plotting an areal view of the solution. Under these conditions, the plume is observed to migrate to the north-northeast over the twelve day period. In this case, the flow is driven by tides.

The second test case examines the movement of the salinity plume in the event of a constant wind vector of (0,-10) m/s; that is, wind blowing out of the north. In Figure 5, we plot the salinity concentration at 6, 8, 9, 10, 11 and 12 days. In this case, we obtain a very different salinity profile. In fact, the saline plume has been highly dispersed, so that concentrations are near background concentrations over most of the region, except for a small area near the source and along the



areal view 3D view

Figure 3: Wind stress test case: Galveston Bay Mesh.

coastline south of the source region. In this case, the wind stress forcing apparently seems to counteract the predominant direction of flow due to tidal forcing, causing significant variation in the velocity field and dispersion of the plume.

The third test case examines the movement of the salinity plume in the event of a constant wind vector of (0,10) m/s; that is, wind blowing out of the south. In Figure 6, we plot the salinity concentration at 6, 8, 9, 10, 11 and 12 days. In this case, the wind stress seems to act in concert with the flow due to tides, resulting in a well-defined plume of salinity which extends from the West Bay to the entrance of Galveston Bay at the end of 12 days.

3.2 The quarter annular harbor problem and comparisons with ELCIRC and SELFE

The ELCIRC and SELFE models [31, 5], developed at the Oregon Health & Science University by Baptista *et al*, are unstructured grid models for 3D baroclinic circulation. ELCIRC uses an extension of well-known staggered-grid finite difference methods to unstructured grids. The method is low-order, but volume conserving. It has been widely used for modeling the Columbia River, and has been applied to other coastal environments.

One of the benchmark cases available on the ELCIRC web site, which has also been used in testing UTBEST3D, is tidal flow in a quarter annular harbor. An areal plot of the domain, discretized with a coarse triangular mesh consisting of 96 elements and 63 nodes, with one layer in the vertical direction, is shown in Figure 7. The boundary of the domain consists of three land boundares along the inner radius, horizontal ($y = 0$) and vertical ($x = 0$) boundaries, and an open ocean boundary along the outer radius. The units here are in meters. Along the outer boundary, elevation is specified using a standard M2 tide with magnitude .3048 m. The runs are cold started, and the tide is ramped up over a 2 day time period. The total simulation time is 5 days. The bathymetry is assumed to be a constant 10 m. A no-slip boundary condition is assumed at the bottom of the domain and no horizontal or vertical eddy viscosity is specified.

Six locations within the domain were chosen where elevations and velocities were interpolated from model output at specific times during each run. These locations are labeled 1-6 in Figure 7. For this test case, we run a full nonlinear model (no nonlinear terms are “turned-off” in the simulation). Therefore, in order to study the behavior of the models, we performed grid refinement studies and examined the convergence of the solutions under grid refinement. Thus, in addition to the grid shown in Figure 7, simulations were also performed on two refined grids, obtained by refining the coarse grid using edge bisection (vertical edges only–there is no refinement in z). These grids are shown in Figure 8 and are labeled “medium” and “fine” grids in the discussion below.

First, we plot elevation solutions between 2 and 5 days obtained by UTBEST3D for piecewise constant approximations on the coarse grid, and the medium and fine grids, at locations 1, 2, 3 and 5, in Figure 9. The solutions obtained at locations 4 and 6 were virtually identical to those at locations 1 and 2, which they should be by symmetry, therefore we don’t show these solutions in the plots below. Here we used a small time step of 2 seconds to fully resolve the solution in time. The maximum (in time) relative difference between the *coarse* and *medium* grid solutions and the *medium* and *fine* grid solutions was measured at each location. Define

$$\begin{aligned} E_{cm} &= \frac{\max((\text{medium solution}) - (\text{coarse solution}))}{\max(\text{medium solution})} \\ E_{mf} &= \frac{\max((\text{fine solution}) - (\text{medium solution}))}{\max(\text{fine solution})}. \end{aligned}$$

The errors obtained at each location for this case as are follows:

$$\begin{aligned} \text{Location 1}: &\quad E_{cm} = 12.87\%, \quad E_{mf} = 8.72\% \\ \text{Location 2}: &\quad E_{cm} = 11.17\%, \quad E_{mf} = 6.74\% \\ \text{Location 3}: &\quad E_{cm} = 16.04\%, \quad E_{mf} = 8.64\% \\ \text{Location 5}: &\quad E_{cm} = 19.68\%, \quad E_{mf} = 6.74\% \end{aligned}$$

We note that the error at each location is reduced by nearly a factor of two as the grid is refined by a factor of two, which is consistent with first order approximations.

Next, we show similar results for piecewise linear approximations in Figure 10. Here there are very small differences between solutions on the different grids, indicating that the solution is essentially resolved even on the coarsest grid. The relative errors at each location are given as follows:

$$\begin{aligned} \text{Location 1}: &\quad E_{cm} = 2.61\%, \quad E_{mf} = .64\% \\ \text{Location 2}: &\quad E_{cm} = 1.22\%, \quad E_{mf} = .39\% \\ \text{Location 3}: &\quad E_{cm} = 2.12\%, \quad E_{mf} = .65\% \\ \text{Location 5}: &\quad E_{cm} = 1.52\%, \quad E_{mf} = .39\% \end{aligned}$$

Note that the errors are reduced by close to a factor of four as the grid is refined by a factor of two, indicating second order convergence. We also tested piecewise quadratic solutions. In Figure 11, we compare constant, linear and quadratic solutions obtained on the coarse grid for each location. As seen in the figure, there is substantial difference between the constant and linear solutions. At location 1, this difference is about 32%. However, there is less than 1% difference between the linear and quadratic solutions, which provides further indication that these solutions are converged.

While the size of the time step was very small in these runs, we found that increasing the time step did not substantially degrade the quality of the solutions. In Figure 12, we compare piecewise constant solutions on the coarse grid at location 1 for $\Delta t = 2$ and $\Delta t = 450$ seconds, and piecewise linear solutions for $\Delta t = 2$ and $\Delta t = 225$ seconds. Several numerical tests were performed to find the maximum allowable time step in each case. It is expected that the CFL (Courant-Friedrichs-Levy) time-step constraint is more severe with increasing polynomial order. As observed in the figure, there are some differences in the solutions for the piecewise constant case, around 12% relative difference. For the piecewise linear case, the difference is less than 1%, again indicating that even with a much larger time step the linear solution is essentially resolved.

In the next sequence of figures, we present comparisons between solutions generated using UTBEST3D, ELCIRC and SELFE. In Figure 13, we compare solutions generated by the two codes at locations 1, 2, 3 and 5 on the coarse grid. Here we are using piecewise linear approximations in UTBEST3D. The time step for ELCIRC and SELFE was large, 1047 seconds, this was preset in the input file obtained from the ELCIRC web site. The UTBEST3D solutions are plotted between days 2 and 5, while the ELCIRC and SELFE solutions start at time zero. Qualitatively, the solutions are very similar in phase, but exhibit differences in amplitudes. At location 1 for example, the maximum error between UTBEST and SELFE is about .1 meters or 12.5% relative difference, and the maximum error between UTBEST and ELCIRC is about .13 meters or 16% relative difference. SELFE and ELCIRC exhibit a maximum difference of about .1 meters in the ramp-up period, but around .05 meters afterward.

We also performed comparisons on the medium and fine grids. These results are given in Figures 14 and 15. For these two grids, we see much better agreement between UTBEST3D and SELFE than either code with ELCIRC. In fact, on the fine grid, at location 1 the maximum difference between UTBEST3D and SELFE is about .03 meters, or about 4% relative difference, whereas the differences between both UTBEST3D and SELFE with ELCIRC are near .2 meters, or almost 25% relative difference.

Recall from Figure 10 that the UTBEST3D linear solution shows relatively small differences between coarse, medium and fine grids at all locations. Therefore, the differences between the coarse and fine grid solutions must lie in the ELCIRC solution. To examine this further, we show in Figure 16 the ELCIRC solutions generated on the coarse and fine grids at location 1. As seen in the figure, there is a significant difference between these two solutions, especially during the ramp-up phase. Furthermore, in Figure 17, we compare the ELCIRC solution at location 1 to the piecewise constant solution in UTBEST3D, both generated on the coarse grid. As seen in the figure, these solutions compare quite well, with a maximum relative difference of less than 10%, which is further confirmation that ELCIRC uses a low-order method.

We conclude this section with a comparison of CPU times for various runs. These runs were performed on a Linux workstation, with a dual-core AMD 1 GHz Opteron processor, and cache size of 1024 KB. The results are presented in Table 2. The ELCIRC run on the coarse grid took around 2 seconds, comparable to the UTBEST3D run for piecewise constants with the maximum allowable stable time step. As seen in Figure 17, the solutions were similar for these cases. The higher-order solutions in UTBEST3D, using linear and quadratic solutions, required 8 seconds and 118 seconds to run on the coarse grid, using the largest stable time step. The higher-order solutions have more degrees of freedom per element, and also require more accurate numerical integrations and more accurate time-stepping procedures. Refining the grid in UTBEST3D requires refining the CFL time step. As observed for the piecewise constant case, the medium grid run with the largest allowable time step took 5 seconds. For the fine grid, the maximum time step was reduced by a factor of 4 over the medium grid, and the run took 13 times longer, or roughly 65 seconds. We might have expected closer to a factor of 16, since the fine grid has 4 times the number of elements as the medium grid, and the time step was also cut by a factor of 4.

Code	Polynomial degree	Grid	Time step (sec)	CPU (sec)
ELCIRC	–	Coarse	1047	2
ELCIRC	–	Medium	1047	7
ELCIRC	–	Fine	1047	28
SELFE	–	Coarse	1047	4
SELFE	–	Medium	1047	8
SELFE	–	Fine	1047	26
UTBEST3D	0	Coarse	450	1
UTBEST3D	0	Medium	225	5
UTBEST3D	0	Fine	66.25	64
UTBEST3D	1	Coarse	225	8
UTBEST3D	1	Medium	112.5	79
UTBEST3D	1	Fine	33.125	1100
UTBEST3D	2	Coarse	112.5	118

Table 2: CPU times for various runs.

4 Conclusions and recommendations

The UTBEST3D model is close to being an operational model for the Texas coast. Over the past year, we have hardened the turbulence models in the code and implemented and tested the addition of wind stress. Over the next contract year, we will test the code with input provided by the TWDB for Corpus Christi Bay, which should provide a rigorous application for the model where we can also compare with data. Additional model comparisons between UTBEST3D and SELFE will also be investigated.

5 Bibliography

References

[1] V. Aizinger, *A Discontinuous Galerkin Method for Two- and Three-Dimensional Shallow-Water Equations*, PhD Thesis, University of Texas at Austin, 2004.

[2] V. Aizinger, C. Dawson, *A discontinuous Galerkin method for two-dimensional flow and transport in shallow water*, Advances in Water Resources, 25, pp. 67-84, 2002.

[3] V. Aizinger, C. Dawson, *The local discontinuous Galerkin method for three-dimensional shallow water flow*, Comp. Meth. Appl. Mech. Eng., 196, pp. 734–746, 2007.

[4] D.N. Arnold, F. Brezzi, B. Cockburn, L.D. Marini, *Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems*, SIAM J. Num. Anal., 39, pp. 1749-1779, 2002.

[5] A.M. Baptista, Y.L. Zhang, A. Chawla, M.A. Zulauf, C. Seaton, E.P. Myers, J. Kindle, M. Wilkin, M. Burla and P.J. Turner, *A cross-scale model for 3D baroclinic circulation in estuary-plume-shelf systems: II. Application to the Columbia River*, Continental Shelf Research, 25, pp. 935-972, 2005.

[6] S. Bunya, E.J. Kubatko, J.J. Westerink, C. Dawson and S. Yoshimura, *A mass conserving moving boundary method for discontinuous Galerkin solutions to the shallow water equations*, in preparation.

[7] J. C. Butcher, *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*, John Wiley and Sons, New York, 1987.

[8] Burchard, H., Bolding, K., *Comparative analysis of four second-moment turbulence closure models*, Ocean Model., 3, pp 33-50, 2001.

[9] S. Chippada, C. N. Dawson, M. Martinez, M. F. Wheeler, *A Godunov-type finite volume method for the system of shallow water equations*, Comput. Meth. Appl. Mech. Engrg., 151, pp. 105-129, 1998.

[10] B. Cockburn, G. Karniadakis and C.-W. Shu, *The development of discontinuous Galerkin methods*, in Discontinuous Galerkin Methods: Theory, Computation and Applications, B. Cockburn, G. Karniadakis and C.-W. Shu, editors, Lecture Notes in Computational Science and Engineering, volume 11, Part I: Overview, pp.3-50, Springer, 2000.

[11] B. Cockburn, C.-W. Shu, *The Runge-Kutta local projection P^1-discontinuous Galerkin method for scalar conservation laws*, RAIRO Modél. Math. Anal. Numér., 25, pp. 337-361, 1991.

[12] B. Cockburn, C.-W. Shu, *The local discontinuous Galerkin finite element method for convection-diffusion systems*, SIAM J. Numer. Anal., 35, pp. 2440-2463, 1998.

[13] B. Cockburn, C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws II: General framework*, Math. Comp., 52, pp. 411-435, 1989.

[14] B.Cockburn, S.Hou, C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case*, Math. Comp., 54, pp. 545-581, 1990.

[15] B. Cockburn, C.-W. Shu, *The Runge-Kutta discontinuous Galerkin finite element method for conservation laws V: Multidimensional systems*, J. Comput. Phys., 141, pp. 199-224, 1998.

[16] A. M. Davies, *A three-dimensional model of the Northwest European continental shelf, with application to the M4 tide*, J. Phys. Oceanogr., 16(5), pp. 797-813, 1986.

[17] C.Dawson, V.Aizinger, *A Discontinuous Galerkin Method for Three-Dimensional Shallow Water Equations*, Journal of Scientific Computing, to appear.

[18] *Delft3D-FLOW User Manual*, WL—Delft Hydraulics, 2005.

[19] A. Harten, P. D. Lax, B. van Leer, *On upstream differencing and Godunov-type schemes for hyperbolic conservation laws*, SIAM Rev., 25, pp. 35-61, 1983.

[20] A. Harten, J. M. Hyman, *Self-Adjusting Grid for One-Dimensional Hyperbolic Conservation Laws*, J. Comp. Phys., 50, pp. 235-269, 1983.

[21] http://mason.gmu.edu/ bklinger/seawater.pdf.

[22] R.J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser, Basel, 1992.

[23] Mellor, G.L., Yamada, T., *Development of a turbulence closure model for geophysical fluid problems*, Rev. Geophys. Space Phys., 20, pp. 851-875, 1982.

[24] P.L. Roe, *Approximate Riemann solvers, parameter vectors, and difference schemes,* J. Comput. Phys., 43, pp. 357-372, 1981.

[25] Umlauf, L., Burchard, H., *A generic length-scale equation for geophysical turbulence models*, J. Marine Res., 61, pp 235-265, 2003.

[26] A. Valiani, V. Caleffi, A. Zanni, *Finite Volume scheme for the 2D Shallow Water equations: Application to a flood event in the Toce river*, Universit degli Studi di Ferrara, preprint.

[27] C. B. Vreugdenhil, *Numerical Methods for Shallow-Water Flow*, Kluwer, 1994.

[28] Warner, J.C., Sherwood, C.R. Arango, H.G., Signell, R.P., *performance of four turbulenc closure models implemented using a generic length scale method*, Ocean Modeling, 8, pp. 81-113, 2005.

[29] T. Weiyan, *Shallow Water Hydrodynamics*, Elsevier Oceanography Series, 55, Elsevier, 1992.

[30] Saffman, P.G., Wilcox, D.C., *Turbulence-model predictions for turbulent boundary layers*, AIAA J. 12(4), pp. 541-546, 1974.

[31] Zhang, Y.-L., Baptista, A.M. and Myers, E.P. *A cross-scale model for 3D baroclinic circulation in estuary-plume-shelf systems: I. Formulation and skill assessment*, Cont. Shelf Res. 24, pp. 2187-2214, 2004.

6 Appendix: The LDG discretization, approximating spaces and time-stepping

We introduce the following sets of element and face indices:

- I_e - set of element indices for prismatic elements in $\Omega(t)$;
- $I_{e,xy}$ - set of element indices for triangular elements in Ω_{xy};
- I_f - set of face indices for prism faces;
- $I_{int} \subset I_f$ - set of interior face indices;
- $I_{ext} \subset I_f$ - set of exterior face indices;
- $I_{lat} \subset I_{int}$ - set of interior lateral face indices;
- $I_{horiz} \subset I_{int}$ - set of interior horizontal face indices;
- $I_D \subset I_{ext}$ - set of exterior lateral face indices;
- $I_{top} \subset I_{ext}$ - set of indices for exterior faces on the top boundary,
- $I_{bot} \subset I_{ext}$ - set of indices for exterior faces on the bottom boundary.

Let us denote $h = \xi - z_b$. Then we can rewrite the mass and momentum conservation equations (18), (1) in the following compact form:

$$\partial_t h \;+\; \nabla_{xy} \cdot \mathbf{C}_h(\mathbf{c}) \;=\; 0, \tag{45}$$

$$\partial_t \mathbf{u}_{xy} \;+\; \nabla \cdot (\mathcal{C}_{\mathbf{u}}(\mathbf{c}) \;-\; \mathcal{D}\nabla \mathbf{u}_{xy}) \;=\; \mathbf{M}(\mathbf{c}), \tag{46}$$

where $\mathbf{c} = (h, u, v, w)$ is the vector of state variables,

$$\mathbf{C}_h(\mathbf{c}) = \begin{pmatrix} \int_{z_b}^{\xi} u \; dz \\ \int_{z_b}^{\xi} v \; dz \end{pmatrix}, \quad \mathcal{C}_{\mathbf{u}}(\mathbf{c}) = \begin{pmatrix} \mathbf{C}_u(\mathbf{c}) \\ \mathbf{C}_v(\mathbf{c}) \end{pmatrix} \; = \; \begin{pmatrix} u^2 + gh & uv & uw \\ uv & v^2 + gh & vw \end{pmatrix},$$

$$\mathbf{M}(\mathbf{c}) \; = \; \begin{pmatrix} F_x - g\partial_x z_b + f_c v \\ F_y - g\partial_y z_b - f_c u \end{pmatrix}.$$

6.1 Weak formulation

First, let us introduce an auxiliary variable $\mathcal{Q}$ and rewrite the second-order momentum equations (46) in mixed form

$$\partial_t \mathbf{u}_{xy} + \nabla \cdot (\mathcal{C}_{\mathbf{u}}(\mathbf{c}) + \sqrt{\mathcal{D}}\mathcal{Q}) = \mathbf{M}(\mathbf{c}), \tag{47}$$

$$\mathcal{Q} = -\sqrt{\mathcal{D}} \nabla \mathbf{u}_{xy}. \tag{48}$$

Let $\mathcal{T}_{\Delta x}$ be a partition of the domain $\Omega(t) \subset \mathbf{R}^d, \; d = 3$ into prisms with strictly vertical lateral (side) faces, and let $\Omega_e(t) \in \mathcal{T}_{\Delta x}$. To obtain a weak form of the momentum equations we multiply (47), (48) by arbitrary smooth test functions $\boldsymbol{\phi}$ and $\mathit{\Psi}$, integrate them on each element $\Omega_e(t) \in \mathcal{T}_{\Delta x}$, and integrate by parts obtaining

$$\begin{aligned} (\partial_t \mathbf{u}_{xy}, \boldsymbol{\phi})_{\Omega_e(t)} \;&+\; \Big\langle (\mathcal{C}_{\mathbf{u}}(\mathbf{c}) + \sqrt{\mathcal{D}}\mathcal{Q}) \cdot \mathbf{n}_e, \boldsymbol{\phi} \Big\rangle_{\partial\Omega_e(t)} \\ &-\; \Big((\mathcal{C}_{\mathbf{u}}(\mathbf{c}) + \sqrt{\mathcal{D}}\mathcal{Q}) \cdot \nabla, \boldsymbol{\phi} \Big)_{\Omega_e(t)} \;=\; (\mathbf{M}(\mathbf{c}), \boldsymbol{\phi})_{\Omega_e(t)}\,, \\ (\mathcal{Q}, \mathit{\Psi})_{\Omega_e(t)} \;&=\; -\Big\langle \mathbf{u}_{xy} \; (\sqrt{\mathcal{D}} \; \mathbf{n}_e), \mathit{\Psi} \Big\rangle_{\partial\Omega_e(t)} \;+\; \Big(\mathbf{u}_{xy} \; (\sqrt{\mathcal{D}} \; \nabla), \mathit{\Psi} \Big)_{\Omega_e(t)}\,, \end{aligned}$$

where $\mathbf{n}_e$ is a unit exterior normal to $\partial\Omega_e(t)$. This weak formulation is well defined for $\mathbf{u}_{xy}(t, x, y, z) \in H^1(0, T; \; V^{d-1})$; $\boldsymbol{\phi}(x, y, z) \in V^{d-1}$; $\mathcal{Q}(t, x, y, z) \in V^{d-1 \times d}$, $\forall t \in [0, T]$; and $\mathit{\Psi}(x, y, z) \in V^{d-1 \times d}$, where

$$V \;=\; L^2(\Omega(t)) \;\cap\; \{u : u_{|\Omega_e(t)} \in H^1(\Omega_e(t)), \; \forall \Omega_e(t) \in \mathcal{T}_{\Delta x}\}. \tag{49}$$

Fixing the direction of the unit normal $\mathbf{n}$ on the interior faces we can sum over all elements $\Omega_e(t) \in \mathcal{T}_{\Delta x}$ and obtain a weak form of the momentum equations

$$\begin{aligned} &\sum_{e \in I_e} (\partial_t \mathbf{u}_{xy}, \boldsymbol{\phi})_{\Omega_e(t)} \;+\; \sum_{i \in I_{int}} \Big\langle (\mathcal{C}_{\mathbf{u}}(\mathbf{c}) + \sqrt{\mathcal{D}}\mathcal{Q}) \cdot \mathbf{n}, [\boldsymbol{\phi}] \Big\rangle_{\gamma_i(t)} \\ &\quad+\; \sum_{i \in I_{ext}} \Big\langle (\mathcal{C}_{\mathbf{u}}(\mathbf{c}) + \sqrt{\mathcal{D}}\mathcal{Q}) \cdot \mathbf{n}, \boldsymbol{\phi} \Big\rangle_{\gamma_i(t)} - \sum_{e \in I_e} \Big((\mathcal{C}_{\mathbf{u}}(\mathbf{c}) + \sqrt{\mathcal{D}}\mathcal{Q}) \cdot \nabla, \boldsymbol{\phi} \Big)_{\Omega_e(t)} \\ &\quad=\; \sum_{e \in I_e} (\mathbf{M}(\mathbf{c}), \boldsymbol{\phi})_{\Omega_e(t)}\,, \end{aligned} \tag{50}$$

$$\begin{aligned} &\sum_{e \in I_e} (\mathcal{Q}, \mathit{\Psi})_{\Omega_e(t)} \;=\; -\sum_{i \in I_{int}} \Big\langle \mathbf{u}_{xy} \; (\sqrt{\mathcal{D}} \; \mathbf{n}), [\mathit{\Psi}] \Big\rangle_{\gamma_i(t)} \\ &\quad-\; \sum_{i \in I_{ext}} \Big\langle \mathbf{u}_{xy} \; (\sqrt{\mathcal{D}} \; \mathbf{n}), \mathit{\Psi} \Big\rangle_{\gamma_i(t)} \;+\; \sum_{e \in I_e} \Big(\mathbf{u}_{xy} \; (\sqrt{\mathcal{D}} \; \nabla), \mathit{\Psi} \Big)_{\Omega_e(t)}\,. \end{aligned} \tag{51}$$

Discretization of the primitive continuity equation is done in a similar way. Let us denote by Π the orthogonal projection operator from the xyz-space onto the xy-plane ($\Pi(x, y, z) = (x, y)$), and let $\Omega_{e,xy} = \Pi\Omega_e(t)$. Since the free surface is the only moving boundary of $\Omega(t)$, $\Omega_{e,xy}$ are not time-dependent. We multiply (45) by an arbitrary smooth test function $\delta = \delta(x, y)$, integrate it over $\Omega_{e,xy}$, and integrate by parts. Then the mass balance in the water column corresponding to $\Omega_{e,xy}$ can be expressed as

$$(\partial_t h, \delta)_{\Omega_{e,xy}} \; + \; \langle \mathbf{C}_h(\mathbf{c}) \cdot \mathbf{n}, \delta \rangle_{\partial\Omega_{e,xy}} \; - \; (\mathbf{C}_h(\mathbf{c}) \cdot \nabla_{xy}, \delta)_{\Omega_{e,xy}} \; = \; 0.$$

Recalling that $\mathbf{C}_h = \left(\int_{z_b}^{\xi} u dz, \int_{z_b}^{\xi} v dz \right)$ we can rewrite the equation above in a special 2D/3D form

$$(\partial_t h, \delta)_{\Omega_{e,xy}} + \sum_{\Pi\Omega_e(t) = \Omega_{e,xy}} \langle \mathbf{u}_{xy} \cdot \mathbf{n}_{xy}, \delta \rangle_{\partial\Omega_{e,lat}(t)} - \sum_{\Pi\Omega_e(t) = \Omega_{e,xy}} (\mathbf{u}_{xy} \cdot \nabla_{xy}, \delta)_{\Omega_e(t)} = 0,$$

where $\mathbf{n}_{xy} = (n_x, n_y)$, $\partial\Omega_{e,lat}(t)$ denotes the lateral boundary faces of prism $\Omega_e(t)$, and the summation is over the set of 3D elements in the water column corresponding to $\Omega_{e,xy}$. Note that the expression above is well defined for any $\delta(x, y) \in \mathcal{H} \overset{\text{def}}{=} L^2(\Omega_{xy}) \cap \{h : h_{|\Pi\Omega_e(t)} \in H^1(\Pi\Omega_e(t)), \forall \Omega_e(t) \in \mathcal{T}_{\Delta x}\}$ and $h(t, x, y) \in H^1(0, T; \mathcal{H})$. Summing over all elements $\Omega_e(t) \in \mathcal{T}_{\Delta x}$ we obtain a weak form of the PCE

$$\begin{aligned} &\sum_{e \in I_{e,xy}} (\partial_t h, \delta)_{\Omega_{e,xy}} \; + \; \sum_{i \in I_{lat}} \langle \mathbf{u}_{xy} \cdot \mathbf{n}_{xy}, [\delta] \rangle_{\gamma_i(t)} \\ &\quad + \; \sum_{i \in I_D} \langle \mathbf{u}_{xy} \cdot \mathbf{n}_{xy}, \delta \rangle_{\gamma_i(t)} \; - \; \sum_{e \in I_e} (\mathbf{u}_{xy} \cdot \nabla_{xy}, \delta)_{\Omega_e(t)} \; = \; 0. \end{aligned} \tag{52}$$

To discretize the continuity equation we multiply (3) by an arbitrary smooth test function σ, integrate it over $\Omega_e(t)$, and integrate by parts obtaining

$$\langle \mathbf{u} \cdot \mathbf{n}, \sigma \rangle_{\partial\Omega_e(t)} \; - \; (\mathbf{u} \cdot \nabla, \sigma)_{\Omega_e(t)} \; = \; 0.$$

Summing over all elements $\Omega_e(t) \in \mathcal{T}_{\Delta x}$ we get a weak form of the continuity equation

$$\sum_{i \in I_{int}} \langle \mathbf{u} \cdot \mathbf{n}, [\sigma] \rangle_{\gamma_i(t)} \; + \; \sum_{i \in I_{ext}} \langle \mathbf{u} \cdot \mathbf{n}, \sigma \rangle_{\gamma_i(t)} \; - \; \sum_{e \in I_e} (\mathbf{u} \cdot \nabla, \sigma)_{\Omega_e(t)} \; = \; 0. \tag{53}$$

6.2 Semi-discrete formulation

Next, we seek to approximate

$$(h(t, \cdot), \mathbf{u}_{xy}(t, \cdot), w(t, \cdot), \mathcal{Q}(t, \cdot)),$$

a solution to problem (50) – (53), with a function

$$(H(t, \cdot), \mathbf{U}_{xy}(t, \cdot), W(t, \cdot), \mathcal{Q}(t, \cdot)) \in \mathcal{H}_\Delta \times U_\Delta \times W_\Delta \times Z_\Delta,$$

where $\mathcal{H}_\Delta \subset \mathcal{H}$, $U_\Delta \subset V^{d-1}$, $W_\Delta \subset V$, and $Z_\Delta \subset V^{d-1 \times d}$ are some finite-dimensional subspaces. For this purpose, we can use the weak formulation with one important modification. Since the approximation spaces utilized in the DG methods do not guarantee continuity across the inter-element boundaries, all integrands in the integrals over interior faces have to be replaced by suitably chosen numerical fluxes that preserve consistency and stability of the method. A semi-discrete finite element solution $(H(t,\cdot), \mathbf{U}_{xy}(t,\cdot), W(t,\cdot), \mathcal{Q}(t,\cdot))$ is obtained by requiring that for any $t \in [0,T]$, for all $\Omega_e(t) \in \mathcal{T}_{\Delta x}$, and for all $(\delta, \boldsymbol{\phi}, \varPsi, \omega) \in \mathcal{H}_\Delta \times U_\Delta \times W_\Delta \times Z_\Delta$ the following holds:

$$\begin{aligned}
&\sum_{e\in I_{e,xy}} (\partial_t H, \delta)_{\Omega_{e,xy}} + \sum_{i\in I_{lat}} \left\langle \frac{\hat{C}_{h,\mathbf{n}}(\mathbf{C}_-, \mathbf{C}_+)}{H_s}, [\delta] \right\rangle_{\gamma_i(t)} \\
&\quad + \sum_{i\in I_D} \left\langle \frac{\hat{C}_{h,\mathbf{n}}(\mathbf{C}_-, \mathbf{C}_+)}{H_s}, \delta \right\rangle_{\gamma_i(t)} - \sum_{e\in I_e} \left(\frac{\mathbf{U}_{xy} H}{H_s} \cdot \nabla_{xy}, \delta \right)_{\Omega_e(t)} = 0, \qquad (54)
\end{aligned}$$

$$\begin{aligned}
&\sum_{e\in I_e} (\partial_t \mathbf{U}_{xy}, \boldsymbol{\phi})_{\Omega_e(t)} + \sum_{i\in I_{int}} \left\langle \hat{\mathbf{C}}_{\mathbf{u},\mathbf{n}}(\mathbf{C}_-, \mathbf{C}_+) + \sqrt{\mathcal{D}}\hat{\mathcal{Q}} \cdot \mathbf{n}, [\boldsymbol{\phi}] \right\rangle_{\gamma_i(t)} \\
&\quad + \sum_{i\in I_{ext}} \left\langle \mathbf{C}_{\mathbf{u},\mathbf{n}}(\mathbf{C}_-, \mathbf{C}_+) + \sqrt{\mathcal{D}}\mathcal{Q} \cdot \mathbf{n}, \boldsymbol{\phi} \right\rangle_{\gamma_i(t)} - \sum_{e\in I_e} \left((\mathcal{C}_{\mathbf{u}}(\mathbf{C}) + \sqrt{\mathcal{D}}\mathcal{Q}) \cdot \nabla, \boldsymbol{\phi} \right)_{\Omega_e(t)} \\
&\quad = \sum_{e\in I_e} (\mathbf{M}(\mathbf{C}), \boldsymbol{\phi})_{\Omega_e(t)}, \qquad (55)
\end{aligned}$$

$$\begin{aligned}
&\sum_{e\in I_e} (\mathcal{Q}, \varPsi)_{\Omega_e(t)} = -\sum_{i\in I_{int}} \left\langle \hat{\mathbf{U}}_{xy}\ (\sqrt{\mathcal{D}}\ \mathbf{n}), [\varPsi] \right\rangle_{\gamma_i(t)} \\
&\quad - \sum_{i\in I_{ext}} \left\langle \mathbf{U}_{xy}\ (\sqrt{\mathcal{D}}\ \mathbf{n}), \varPsi \right\rangle_{\gamma_i(t)} + \sum_{e\in I_e} \left(\mathbf{U}_{xy}\ (\sqrt{\mathcal{D}}\ \nabla), \varPsi \right)_{\Omega_e(t)}, \qquad (56)
\end{aligned}$$

$$\sum_{i\in I_{int}} \left\langle \hat{\mathbf{U}} \cdot \mathbf{n}, [\sigma] \right\rangle_{\gamma_i(t)} + \sum_{i\in I_{ext}} \left\langle \hat{\mathbf{U}} \cdot \mathbf{n}, \sigma \right\rangle_{\gamma_i(t)} - \sum_{e\in I_e} (\mathbf{U} \cdot \nabla, \sigma)_{\Omega_e(t)} = 0, \qquad (57)$$

where $(\hat{C}_{h,\mathbf{n}}(\mathbf{C}_-, \mathbf{C}_+), \hat{\mathbf{C}}_{\mathbf{u},\mathbf{n}}(\mathbf{C}_-, \mathbf{C}_+))$ is an approximation to the nonlinear boundary flux $(\mathbf{U}_{xy} H \cdot \mathbf{n}_{xy}, \mathcal{C}_{\mathbf{u}} \cdot \mathbf{n})$ that depends on the values of the state variables $\mathbf{C}_-, \mathbf{C}_+$ on both sides of the discontinuity. The stability of the method depends to a large degree on this approximation satisfying certain entropy conditions. We will address this issue in some detail in Section 6.3. The restrictions on the linear boundary fluxes $\hat{\mathbf{U}}_{xy}, \hat{\mathcal{Q}}$ are much less severe. They can be set equal to arithmetic averages of the values of the corresponding variables on both sides of the discontinuity or some other consistent numerical flux.

Remark 1: The continuity equation is, unlike the mass and momentum conservation equations, not time-dependent, its main role being computation of the vertical velocity component W to maintain a divergence-free velocity field. Regarding (57) and the kinematic boundary condition at the bottom (4) as an initial value problem for W, we can compute W element-by-element in each water column starting at the bottom and using the solution from the element below as an initial condition.

The choice of the boundary flux $\hat{\mathbf{U}}$ in (57) also merits a special mention. On the interior lateral faces it should be set equal to $\frac{\hat{C}_{h,\mathbf{n}}(\mathbf{C}_-,\mathbf{C}_+)}{H_s}$, exactly as it is in the discrete form of the primitive continuity equation (54), in order to preserve the local mass conservation properties of our numerical scheme. On the interior horizontal faces $\hat{\mathbf{U}}$ can be approximated by the average or upwind values of the corresponding variable.

6.3 Riemann solvers for the 3D problem

In this section, we will show how to utilize the boundary flux formulation shown in (54) in a Riemann solver. Since we managed to transform all boundary integrals into a 3D form, we end up with a well-posed Riemann problem that can be solved to produce a numerical boundary flux on the lateral faces satisfying the entropy condition (for a discussion of different entropy conditions see, e.g., [22]).

Let $\mathbf{P}$ be a point on Γ, where Γ is an interior lateral boundary face in the 3D mesh (see Figure 2). Let $\mathbf{n} = (n_x, n_y, n_z)$ be a unit normal to Γ at $\mathbf{P}$. Note that n_z is equal to 0 since all lateral faces are strictly vertical. We denote by $\mathbf{c} = (h, u, v)$ the vector of state variables. Note that we don't include the vertical velocity component in $\mathbf{c}$ because w only enters the normal boundary flux when multiplied by n_z. Then we define the left and right states $\mathbf{c}_L, \mathbf{c}_R$ at $\mathbf{P}$ as $\mathbf{c}_L = \lim_{\varepsilon\to 0-} \mathbf{c}(\mathbf{P} + \varepsilon\mathbf{n}), \mathbf{c}_R = \lim_{\varepsilon\to 0+} \mathbf{c}(\mathbf{P} + \varepsilon\mathbf{n})$. Our task is to compute an entropy solution $\hat{\mathbf{C}}_{\mathbf{n}}(\mathbf{c}_L, \mathbf{c}_R)$ to the Riemann problem for the nonlinear boundary flux $\mathbf{C}_{\mathbf{n}} = (\mathbf{C}_h \cdot \mathbf{n}, \mathbf{C}_u \cdot \mathbf{n}, \mathbf{C}_v \cdot \mathbf{n})$ at $\mathbf{P}$ (see (54), (55)).

6.4 Riemann solver of Roe

In this solver, an approximation to the normal boundary flux is given by

$$\hat{\mathbf{C}}_{\mathbf{n}}(\mathbf{c}_L, \mathbf{c}_R) = \mathbf{C}_{\mathbf{n}}(\mathbf{c}_L) + \sum_{i=1}^{3} \alpha_i \hat{\lambda}_i^- \hat{\mathbf{r}}_i, \tag{58}$$

where $\hat{\lambda}_i$ are the eigenvalues and $\hat{\mathbf{r}}_i$ the corresponding eigenvectors of matrix $\hat{R}_{\mathbf{n}}(\mathbf{c}_L, \mathbf{c}_R)$ defined below, $x^- = min\{0, x\}$, and α_i are calculated from

$$\sum_{i=1}^{3} \alpha_i \hat{\mathbf{r}}_i = \mathbf{c}_R - \mathbf{c}_L. \tag{59}$$

The matrix $\hat{R}_{\mathbf{n}}(\mathbf{c}_L, \mathbf{c}_R)$ has to satisfy the following three conditions [24]:

(i) $\hat{R}_{\mathbf{n}}(\mathbf{c}_L, \mathbf{c}_R)(\mathbf{c}_R - \mathbf{c}_L) = \mathbf{C}_{\mathbf{n}}(\mathbf{c}_R) - \mathbf{C}_{\mathbf{n}}(\mathbf{c}_L)$;

(ii) $\hat{R}_{\mathbf{n}}(\mathbf{c}_L, \mathbf{c}_R)$ is diagonalizable with real eigenvalues;

(iii) $\hat{R}_{\mathbf{n}}(\mathbf{c}_L, \mathbf{c}_R) \to C'_{\mathbf{n}}(\mathbf{c})$ smoothly as $\mathbf{c}_L, \mathbf{c}_R \to \mathbf{c}$, where

$$C'_{\mathbf{n}}(\mathbf{c}) = \begin{pmatrix} un_x + vn_y & hn_x & hn_y \\ gn_x & 2un_x + vn_y & un_y \\ gn_y & vn_x & un_x + 2vn_y \end{pmatrix}. \tag{60}$$

We claim that setting $\hat{R}_{\mathbf{n}}(\mathbf{c}_L, \mathbf{c}_R)$ equal to $C'_{\mathbf{n}}(\overline{\mathbf{c}})$, where $\overline{\mathbf{c}} = \frac{1}{2}(\mathbf{c}_L + \mathbf{c}_R)$, satisfies conditions on $\hat{R}_{\mathbf{n}}$. Indeed, we obtain the following eigenvalues and eigenvectors for $C'_{\mathbf{n}}(\mathbf{c})$:

$$\begin{aligned} \lambda_1(\mathbf{c}) &= \tfrac{3}{2}u_{\mathbf{n}} - \tfrac{1}{2}a, \quad & \mathbf{r}_1(\mathbf{c}) &= \begin{pmatrix} h \\ u - \frac{n_x}{2}(u_{\mathbf{n}} + a) \\ v - \frac{n_y}{2}(u_{\mathbf{n}} + a) \end{pmatrix}; \\ \lambda_2(\mathbf{c}) &= \tfrac{3}{2}u_{\mathbf{n}}, & \mathbf{r}_2(\mathbf{c}) &= \begin{pmatrix} 0 \\ -n_y \\ n_x \end{pmatrix}; \\ \lambda_3(\mathbf{c}) &= \tfrac{3}{2}u_{\mathbf{n}} + \tfrac{1}{2}a, & \mathbf{r}_3(\mathbf{c}) &= \begin{pmatrix} h \\ u - \frac{n_x}{2}(u_{\mathbf{n}} - a) \\ v - \frac{n_y}{2}(u_{\mathbf{n}} - a) \end{pmatrix}; \end{aligned} \tag{61}$$

where $u_{\mathbf{n}} = un_x + vn_y$ and $a = \sqrt{u_{\mathbf{n}}^2 + 4gh}$. Therefore, condition (ii) is satisfied. Clearly, $C'_{\mathbf{n}}(\overline{\mathbf{c}}) \to C'_{\mathbf{n}}(\mathbf{c})$ smoothly as $\mathbf{c}_L, \mathbf{c}_R \to \mathbf{c}$. Finally, the first condition can be verified by simply substituting the appropriate values in (i).

6.5 Entropy fix for Roe's solver

The Roe's solver described in the previous section is quite adequate for most problems involving shocks, but it might experience difficulties with certain types of rarefaction waves (sonic rarefactions). There are several ways to modify Roe's algorithm that can fix this problem. Here, we give an approach presented in [20].

Let λ_p be the pth eigenvalue in (61). Then we define

$$\mathbf{c}_{p,L} \;=\; \mathbf{c}_L \;+\; \sum_{i=1}^{p-1} \alpha_i \hat{\mathbf{r}}_i, \qquad \mathbf{c}_{p,R} \;=\; \mathbf{c}_L \;+\; \sum_{i=1}^{p} \alpha_i \hat{\mathbf{r}}_i, \tag{62}$$

and, similarly to (58), the normal boundary flux is computed as

$$\hat{\mathbf{C}}_{\mathbf{n}}(\mathbf{c}_L, \mathbf{c}_R) = \mathbf{C}_{\mathbf{n}}(\mathbf{c}_L) + \sum_{i=1}^{3} \alpha_i \tilde{\lambda}_i \hat{\mathbf{r}}_i, \tag{63}$$

where

$$\tilde{\lambda}_i \;=\; \lambda_i^-(\mathbf{c}_{p,L})\, \frac{\lambda_i^+(\mathbf{c}_{p,R}) - \lambda_i(\overline{\mathbf{c}})}{\lambda_i^+(\mathbf{c}_{p,R}) - \lambda_i^-(\mathbf{c}_{p,L})}, \tag{64}$$

with $x^- = min\{0, x\}$ and $x^+ = max\{0, x\}$.

6.6 Lax-Friedrichs solver

The simplest Riemann solver supported in our 3D simulator is the Lax-Friedrichs scheme. In this method, the normal boundary flux is approximated by

$$\hat{\mathbf{C}}_{\mathbf{n}}(\mathbf{c}_L, \mathbf{c}_R) \;=\; \frac{1}{2}(\mathbf{C}_{\mathbf{n}}(\mathbf{c}_L) + \mathbf{C}_{\mathbf{n}}(\mathbf{c}_R)) \;+\; \frac{1}{2}|\hat{\lambda}_{max}|(\mathbf{c}_L - \mathbf{c}_R), \tag{65}$$

where $\hat{\lambda}_{max}$ is the largest (in absolute value) eigenvalue of $C'_{\mathbf{n}}(\overline{\mathbf{c}})$ (see (61)).

6.7 HLL solver

In the HLL Riemann solver, proposed by Harten, Lax, and van Leer [19], an approximation to the nonlinear boundary flux is computed as

$$\hat{\mathbf{C}}_{\mathbf{n}}(\mathbf{c}_L, \mathbf{c}_R) \;=\; \begin{cases} \mathbf{C}_{\mathbf{n}}(\mathbf{c}_L) & \text{if } s_L \geq 0, \\ \frac{s_R \mathbf{C}_{\mathbf{n}}(\mathbf{c}_L) \;-\; s_L \mathbf{C}_{\mathbf{n}}(\mathbf{c}_R) \;+\; s_L s_R (\mathbf{c}_R - \mathbf{c}_L)}{s_R - s_L} & \text{if } s_L \leq 0 \leq s_R, \\ \mathbf{C}_{\mathbf{n}}(\mathbf{c}_R) & \text{if } s_R \leq 0, \end{cases} \tag{66}$$

where we choose (see [26]) $s_L \;=\; min\{u_{\mathbf{n},L} - a_L, u^*_{\mathbf{n}} - a^*\}$, $s_R \;=\; max\{u_{\mathbf{n},R} + a_R, u^*_{\mathbf{n}} + a^*\}$ with $u_{\mathbf{n}} = un_x + vn_y$, $a = \sqrt{gh}$, and

$$u^* = \frac{1}{2}(u_{\mathbf{n},L} + u_{\mathbf{n},R}) + a_L - a_R, \quad a^* = \frac{1}{2}(a_L + a_R) + \frac{1}{4}(u_{\mathbf{n},L} - u_{\mathbf{n},R}).$$

6.8 Treatment of boundary conditions

In this section, we will discuss specific implementation issues concerning the boundary conditions at the top and bottom domain boundaries (4), (6), (8) and the lateral boundary.

- Bottom: On the bottom boundary, we compute the value of $\mathcal{Q}$ from (6) and the value of W from (4), the latter using the computed value of $\mathbf{U}_{xy}$ at the sea bed.
- Free surface: At the free surface boundary, we set $\mathcal{Q}$ equal to 0 and take the velocity values from the interior.

The boundary conditions on the lateral boundaries are handled as follows.

- Land boundary: On the land boundary, we use zero boundary conditions for $\mathcal{Q}$ and set $\hat{\mathbf{U}}_{xy} = \mathbf{U}_{xy} - (\mathbf{U}_{xy} \cdot \mathbf{n})\mathbf{U}_{xy}$, where $\mathbf{U}_{xy}$ is the value of the LDG solution from the interior of the domain. In the Riemann solver, we define the reflected velocity vector $\mathbf{U}^R_{xy} = \mathbf{U}_{xy} - 2(\mathbf{U}_{xy} \cdot \mathbf{n})\mathbf{U}_{xy}$ and solve a Riemann problem for the boundary flux $\hat{\mathbf{C}}_{\mathbf{n}}((H, \mathbf{U}_{xy}), (H, \mathbf{U}^R_{xy}))$.
- Open sea boundary: On the open sea boundary, we take zero boundary conditions for $\mathcal{Q}$ and set $\hat{\mathbf{U}}_{xy} = \mathbf{U}_{xy}$. In the Riemann solver, we compute the boundary flux $\hat{\mathbf{C}}_{\mathbf{n}}((H, \mathbf{U}_{xy}), (\xi_{os} - z_b, \mathbf{U}_{xy}))$
- River boundary: Here, we set $\hat{\mathbf{u}} = \mathbf{u}_r$, and in the Riemann solver compute the boundary flux $\hat{\mathbf{C}}_{\mathbf{n}}((H, \mathbf{U}_{xy}), (\xi_r - z_b, \mathbf{u}_r))$ The values of $\mathcal{Q}$ are taken from the interior.
- Radiation boundary: We set $\mathcal{Q}$ equal to 0 and take values from the interior for all other variables.

6.9 Species transport

The DG method is also applied to the solution of the species transport equations (29). The handling of the diffusion terms is similar to what is described above for $\mathbf{u}_{xy}$, therefore to simplify the discussion we assume $K_r = 0$.

Denote by R a discontinuous approximation to the species concentration r. Multiplying (29) by a discontinuous test function κ, integrating by parts, and approximating $\mathbf{u}$ by $\mathbf{U}$, we obtain the semi-discrete method

$$\begin{aligned}\sum_{e\in I_e} (\partial_t R, \kappa)_{\Omega_e(t)} \;+\; \sum_{i\in I_{int}} \left\langle R^{\mathrm{up}}\ \hat{\mathbf{U}} \cdot \mathbf{n}, [\kappa] \right\rangle_{\gamma_i(t)} \\ + \sum_{i\in I_{ext}} \left\langle R^{\mathrm{up}}\ \hat{\mathbf{U}} \cdot \mathbf{n}, \kappa \right\rangle_{\gamma_i(t)} - \sum_{e\in I_e} (R\ \mathbf{U} \cdot \nabla, \kappa)_{\Omega_e(t)} \;=\; 0. \end{aligned} \tag{67}$$

Here $\hat{\mathbf{U}}$ is computed using the same method as in (57), and R^{up} is the upwind value of R, determined by the sign of $\hat{\mathbf{U}} \cdot \mathbf{n}$.

6.10 Baroclinic model

The discretization of the baroclinic forcing term in the momentum equations (30) is carried out in two steps. First, we compute an L^2-projection of the density field obtained with the help of the equation of state into the discontinuous approximation space chosen for the density variable. In the second step, this density field is integrated exactly in the z-direction starting at the surface and going to the bottom providing thus the density forcing term in momentum equations (30). In the baroclinic case, the discretized momentum equations are given by

$$\begin{aligned}\sum_{e\in I_e}(\partial_t \mathbf{U}_{xy},\boldsymbol{\phi})_{\Omega_e(t)} + \sum_{i\in I_{int}}\left\langle \hat{\mathbf{C}}_{\mathbf{u},\mathbf{n}}(\mathbf{C}_-,\mathbf{C}_+) + \left(\frac{g}{\rho_0}\int_z^{\xi_s}(\bar{\rho}(T,S,\xi-\tilde{z})-\rho_0)d\tilde{z}\right)\mathbf{n} + \sqrt{\mathcal{D}}\hat{\mathcal{Q}}\cdot\mathbf{n},[\boldsymbol{\phi}]\right\rangle_{\gamma_i(t)} \\ + \sum_{i\in I_{ext}}\left\langle \mathbf{C}_{\mathbf{u},\mathbf{n}}(\mathbf{C}_-,\mathbf{C}_+) + \left(\frac{g}{\rho_0}\int_z^{\xi_s}(\rho(T,S,\xi-\tilde{z})-\rho_0)d\tilde{z}\right)\mathbf{n} + \sqrt{\mathcal{D}}\mathcal{Q}\cdot\mathbf{n},\boldsymbol{\phi}\right\rangle_{\gamma_i(t)} \\ - \sum_{e\in I_e}\left((\mathcal{C}_{\mathbf{u}}(\mathbf{C}) + \frac{g}{\rho_0}\int_z^{\xi_s}(\rho(T,S,\xi-\tilde{z})-\rho_0)d\tilde{z} + \sqrt{\mathcal{D}}\mathcal{Q})\cdot\nabla,\boldsymbol{\phi}\right)_{\Omega_e(t)} = \sum_{e\in I_e}(\mathbf{M}(\mathbf{C}),\boldsymbol{\phi})_{\Omega_e(t)}\,,\end{aligned} \tag{68}$$

where ξ_s is the value of the z-coordinate at the free surface mesh (comp. to Figure 2).

6.11 Approximating spaces and time-stepping

The approximations $H \in \mathcal{H}_\Delta$, $\mathbf{U}_{xy} \in U_\Delta$, $W \in W_\Delta$ and $\mathcal{Q} \in Z_\Delta$ are constructed from basis functions consisting of complete polynomials in (x, y, z) defined on each element. Currently implemented are piecewise constant, linear and quadratic polynomial spaces. Each of those spaces can be used to approximate any of the primary unknowns in UTBEST3D independently of approximation spaces utilized by other unknowns. Thus, one may approximate H by piecewise constants, U by linears, and T by piecewise quadratics etc.

After spatial discretization, one arrives at a system of ODEs

$$\mathbf{y}'(t) \;=\; \mathbf{L}_h(\mathbf{y}(t),\, t), \tag{69}$$

where $\mathbf{y}$ represents all degrees of freedom associated with the time-dependent state variables, and $\mathbf{L}_h$ stands for the LDG space discretization operator. The time-stepping method we use is based on an explicit TVD Runge-Kutta method given below, with the order of the method matching the highest order of the spatial discretization. For example, if the highest order is linear, then we integrate in time using a second-order Runge-Kutta method. At each stage of the Runge-Kutta method, the solution process consists of four steps:

1. Solve (57) for the vertical velocity component from the previous time stage. In order to preserve the local mass conservation properties of our LDG scheme we must compute W with the same boundary flux as in the discrete version of the primitive continuity equation (54). Thus, this step requires solution of the Riemann problem on interior lateral faces, the results of which can be then stored for use in 3.

2. Compute the values of auxiliary variable $\mathcal{Q}$ using (56). Since $\mathcal{Q}$ is discontinuous, the computation of $\mathcal{Q}$ is completely local to an element, and only involves the solution of element-wise systems of equations.

3. Compute species (and turbulent closure scheme unknowns) transport.

4. Compute $\mathbf{C}$ from (54), (55) using solutions to the Riemann problem on interior lateral faces obtained in step 1. The linear term $\hat{\mathcal{Q}}$ can be taken equal to the average of values of $\mathcal{Q}$ on both sides of the discontinuity.

5. Update (when desired) the position of the free surface and perform surface mesh smoothing. Update the geometry of prisms and faces in the surface layer.

6.11.1 The TVD Runge-Kutta time-stepping method

To solve the system of ODEs (69), the traditional Runge-Kutta methods offer a wide variety of explicit and implicit schemes of various order (see, e.g., [7]). For most problems with smooth solutions, these methods can be utilized in the time-stepping routine without any reservations. However, for problems with discontinuities or very steep gradients, numerical solutions obtained using traditional Runge-Kutta schemes may suffer from spurious oscillations. This was the principal motivation behind the total variation diminishing (TVD) Runge-Kutta methods introduced by Cockburn and Shu in [11] – [15]. These schemes – denoted $RK\Lambda\Pi P$ – can capture discontinuities without or with essentially dampened oscillations.

The main idea of a $RK\Lambda\Pi P$ method is, first, to reformulate the explicit Runge-Kutta scheme in some suitable form and then to perform, where needed, a limiting procedure on the degrees of freedom corresponding to the higher-order (linear, quadratic, etc) basis functions.

The explicit Runge-Kutta scheme used in $RK\Lambda\Pi P$ methods can be written as follows:

$$\begin{aligned}
\mathbf{y}^{(0)} &= \mathbf{y}^{n-1}, \\
\mathbf{y}^{(i)} &= \sum_{l=0}^{i-1} \Big[\alpha_{il}\,\mathbf{y}^{(l)} \;+\; \beta_{il}\,\Delta t\,\mathbf{L}_h(\mathbf{y}^{(l)}, t^{n-1} + \delta_l \Delta t)\Big],\ i = 1, \dots, s, \\
\mathbf{y}^{n} &= \mathbf{y}^{(s)},
\end{aligned} \tag{70}$$

where s is the number of stages.

- The first order scheme is simply the forward Euler method.
- In the second order scheme (s=2) the coefficients are:

$$\begin{aligned}
&\alpha_{10} = \beta_{10} = 1,\ \alpha_{20} = \alpha_{21} = \beta_{21} = \frac{1}{2},\ \beta_{20} = 0, \\
&\delta_0 = 0,\ \delta_1 = 1.
\end{aligned} \tag{71}$$

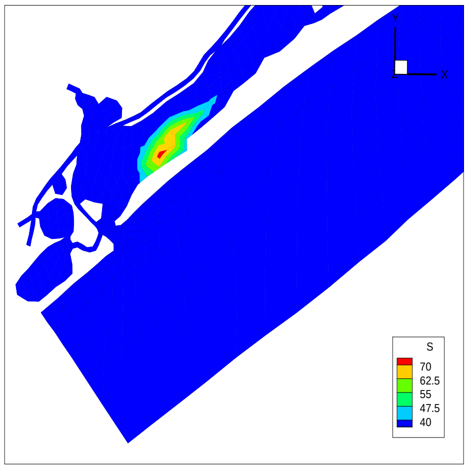
- The coefficients for the third order scheme (s=3) are as follows:

$$\begin{aligned}
&\alpha_{10} = \beta_{10} = 1,\ \alpha_{20} = \frac{3}{4},\ \alpha_{21} = \beta_{21} = \frac{1}{4},\ \beta_{20} = 0, \\
&\alpha_{30} = \frac{1}{3},\ \beta_{30} = \alpha_{31} = \beta_{31} = 0,\ \alpha_{32} = \beta_{32} = \frac{2}{3}, \\
&\delta_0 = 0,\ \delta_1 = 1,\ \delta_2 = \frac{1}{2}.
\end{aligned} \tag{72}$$

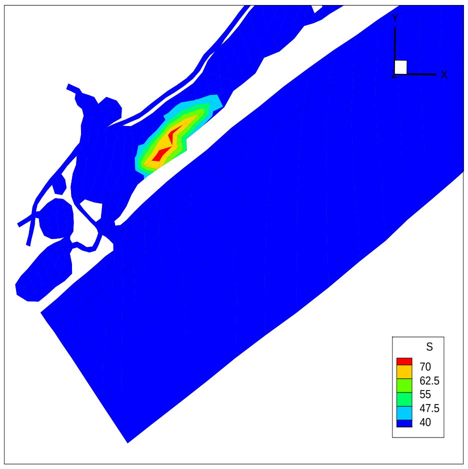
The second component of the $RK\Lambda\Pi P$ methods is the local projection operator $\Lambda\Pi$ whose purpose is to control the magnitude of the higher-order degrees of freedom. Examples of $\Lambda\Pi$ operators for 1D are given in [13] and for standard 2D element shapes in [14]. With the local projection operator, the $RK\Lambda\Pi P$ method is defined as

$$\begin{aligned}
\mathbf{y}^{(0)} &= \mathbf{y}^{n-1}, \\
\mathbf{y}^{(i)} &= \Lambda\Pi\left(\sum_{l=0}^{i-1} \Big[\alpha_{il}\,\mathbf{y}^{(l)} \;+\; \beta_{il}\,\Delta t\,\mathbf{L}_h(\mathbf{y}^{(l)}, t^{n-1} + \delta_l \Delta t)\Big]\right),\ i = 1, \dots, s, \\
\mathbf{y}^{n} &= \mathbf{y}^{(s)}.
\end{aligned} \tag{73}$$

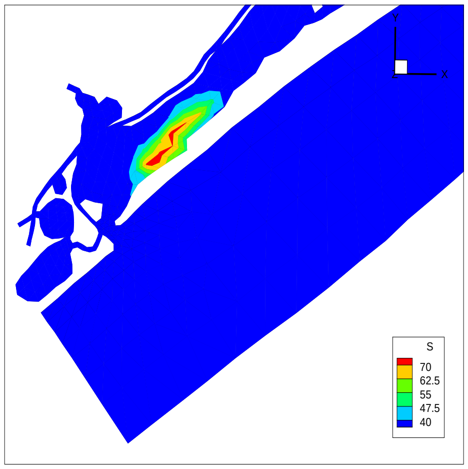
For a suitable choice of the $\Lambda\Pi$ operator and for Δt satisfying the CFL condition, the scheme above can be shown to be TVD in the 1D [13] and 2D [14] cases. We note that in our implementation, we avoid the use of the local projection operator where possible, since our experience with various limiters shows that they generally interfere with the wave structure, thus we take $\Lambda\Pi = I$.



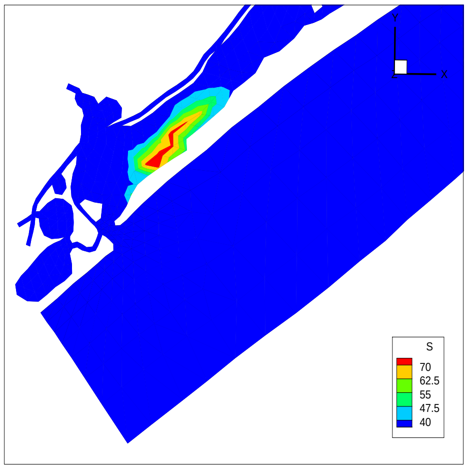
Salinity (6 days)



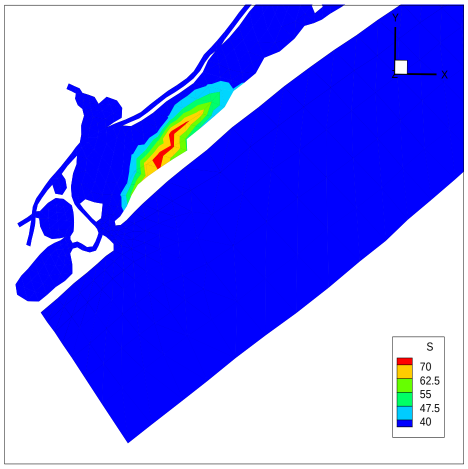
Salinity (8 days)



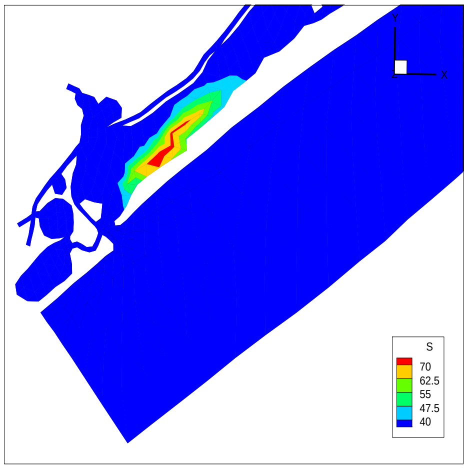
Salinity (9 days)



Salinity (10 days)

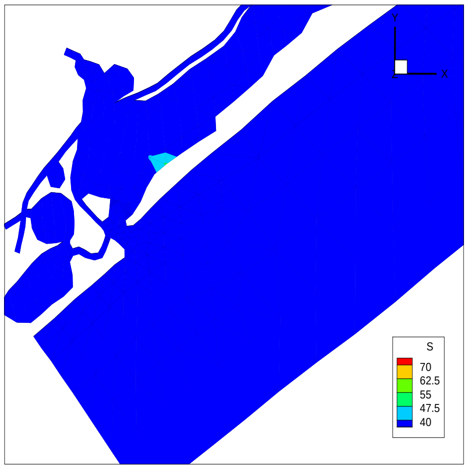


Salinity (11 days)

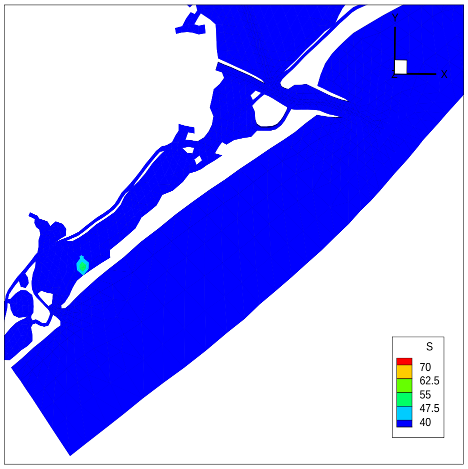


Salinity (12 days)

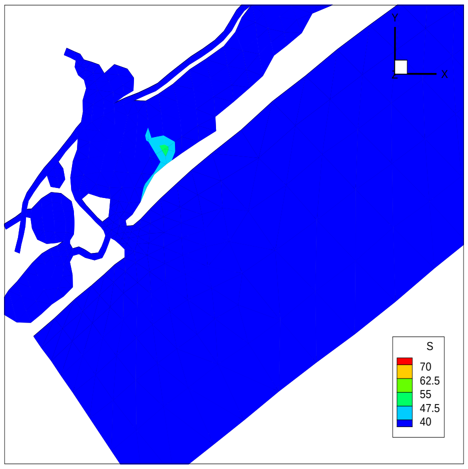
Figure 4: Areal view of salinity concentration for the case with no wind stress. Units are ppt.



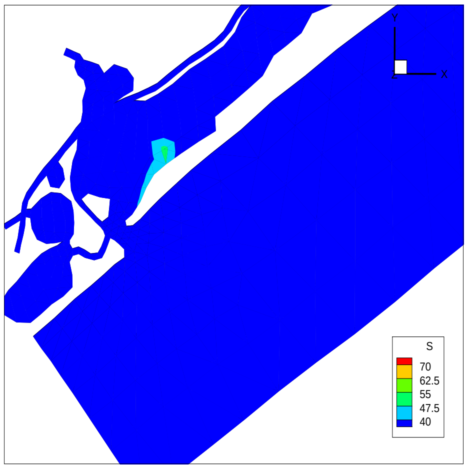
Salinity (6 days)



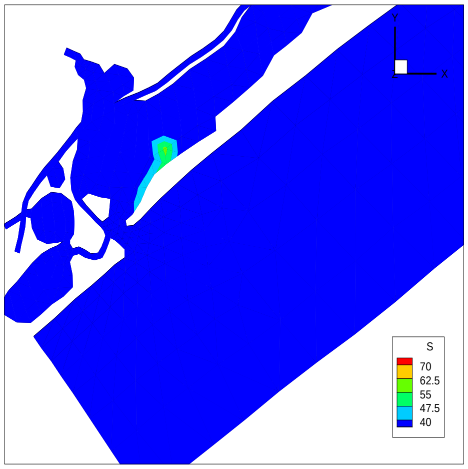
Salinity (8 days)



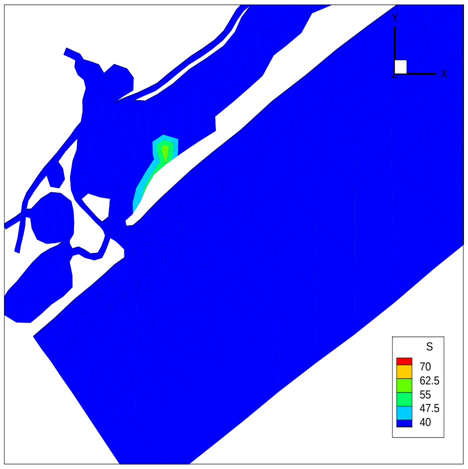
Salinity (9 days)



Salinity (10 days)

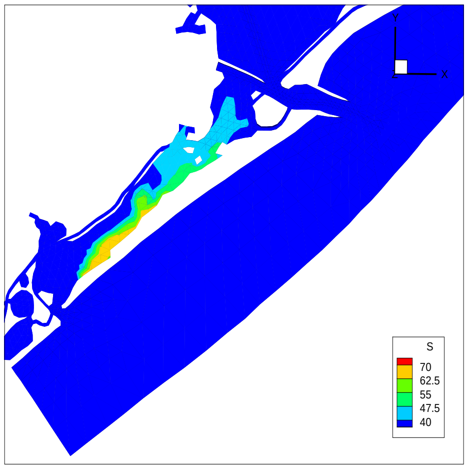


Salinity (11 days)

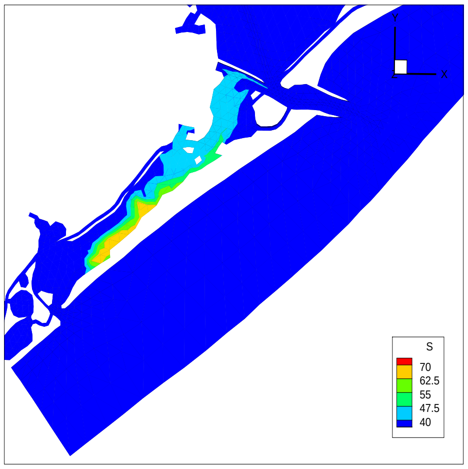


Salinity (12 days)

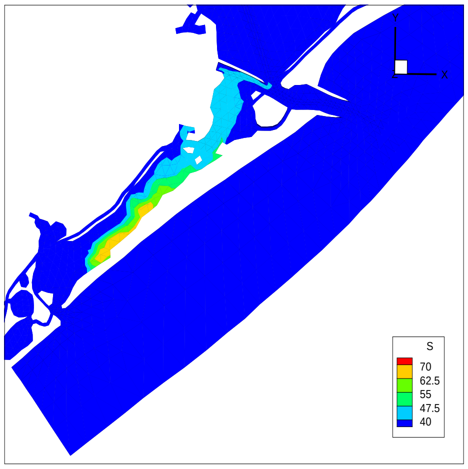
Figure 5: Areal view of salinity concentration for the case with constant wind blowing from the north at 10 m/s. Units are ppt.



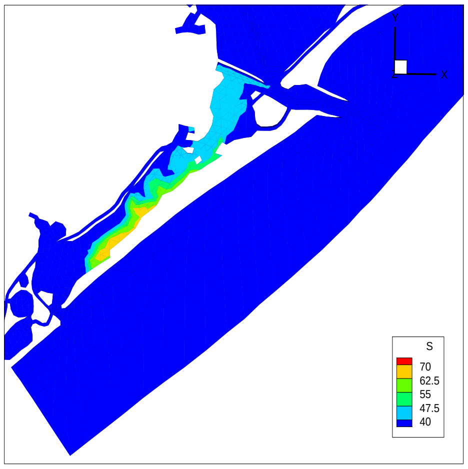
Salinity (6 days)



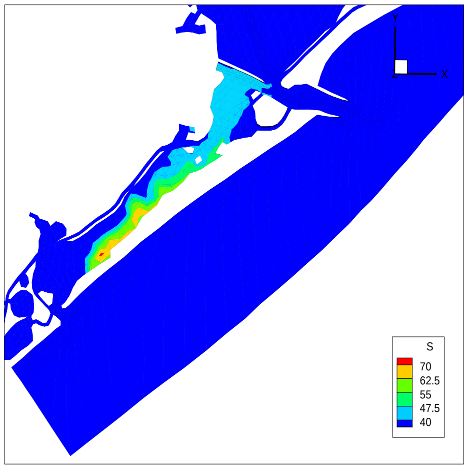
Salinity (8 days)



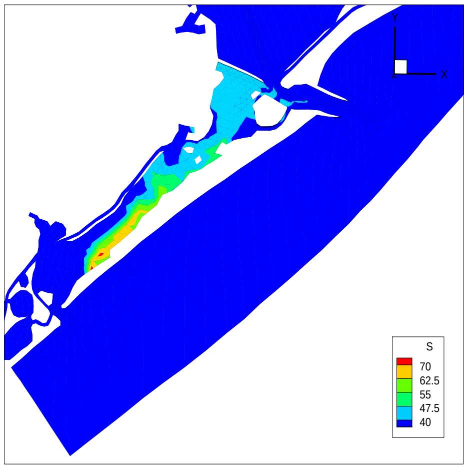
Salinity (9 days)



Salinity (10 days)



Salinity (11 days)



Salinity (12 days)

Figure 6: Areal view of salinity concentration for the case with constant wind blowing from the south at 10 m/s. Units are ppt.

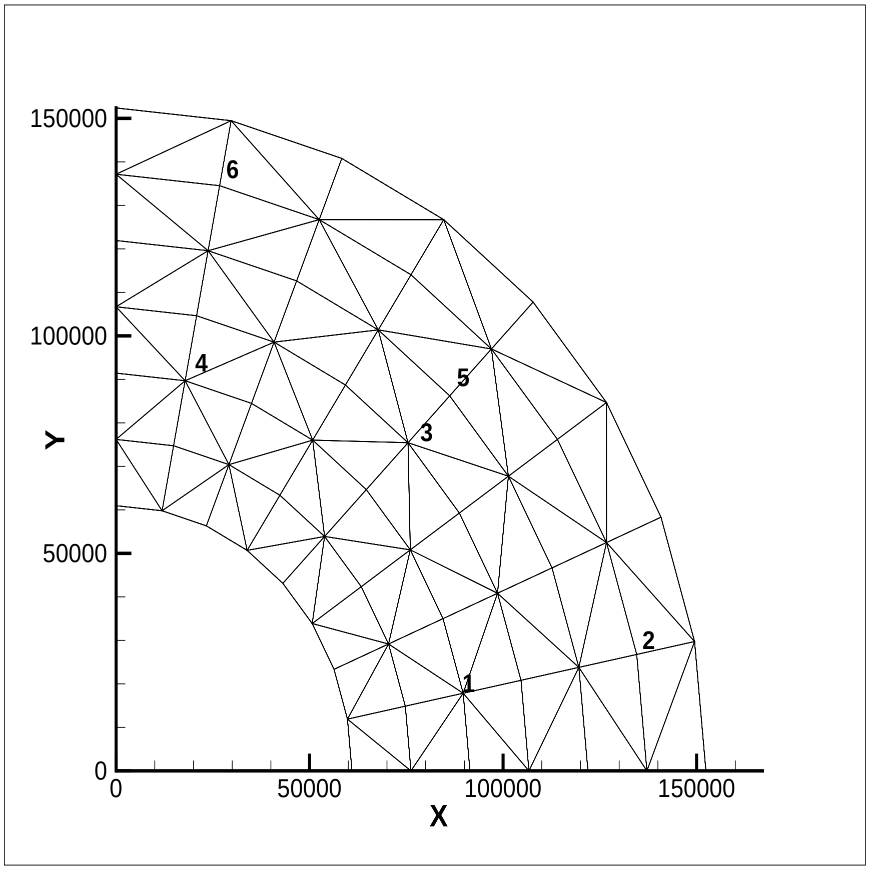
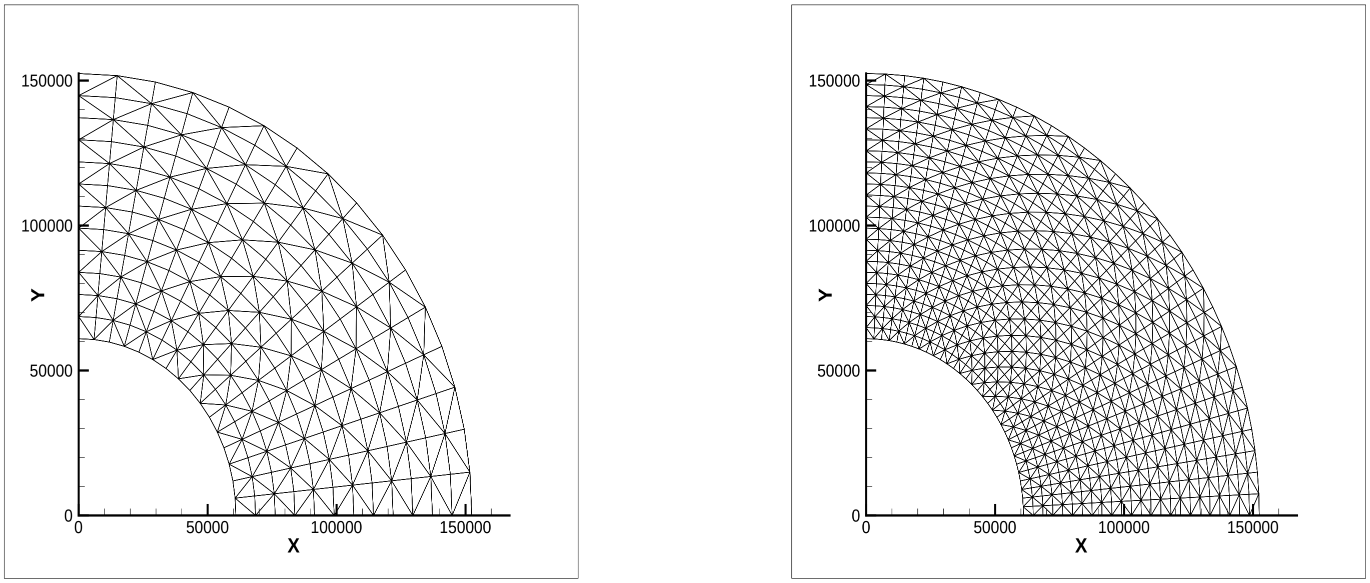


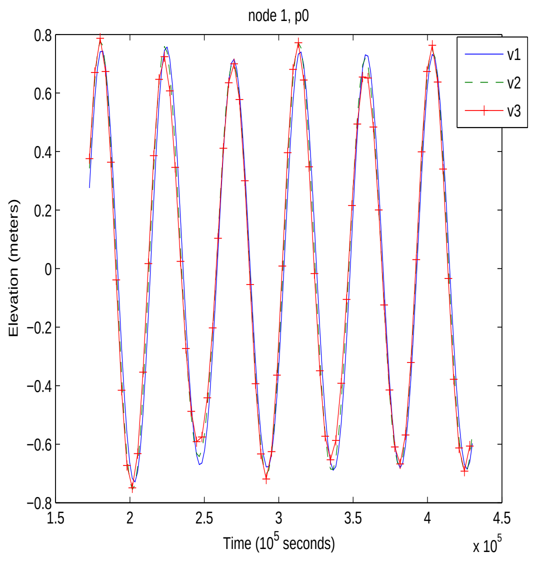
Figure 7: Areal view of quarter annular domain with coarse triangulation. Units are in meters.



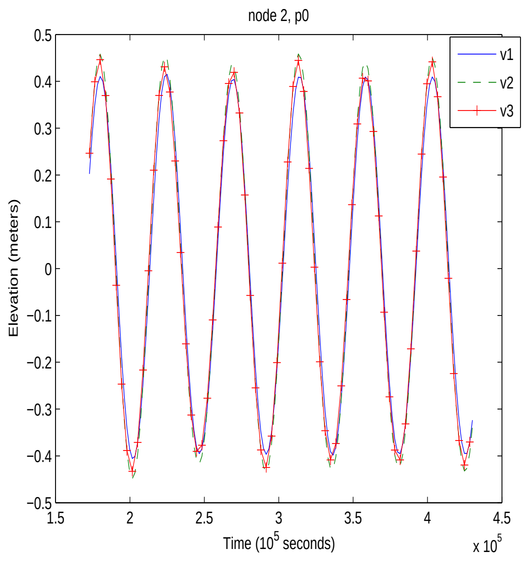
Mesh once refined

Mesh twice refined

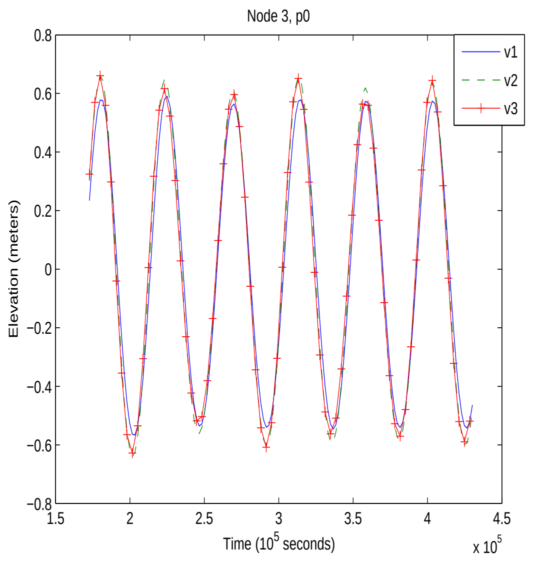
Figure 8: Once and twice refined meshes for the quarter annular harbor



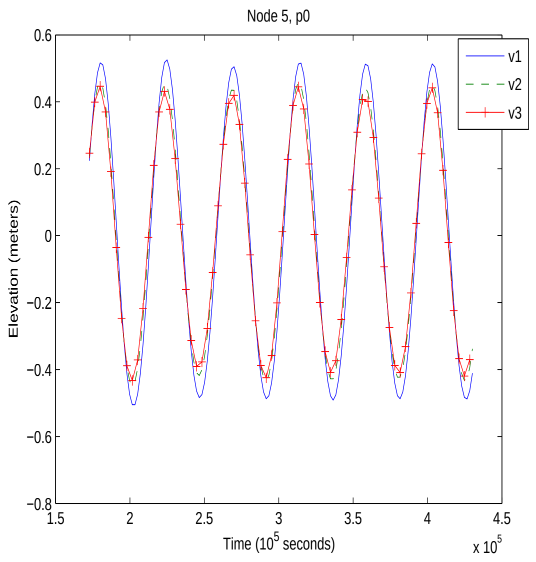
Location 1



Location 2

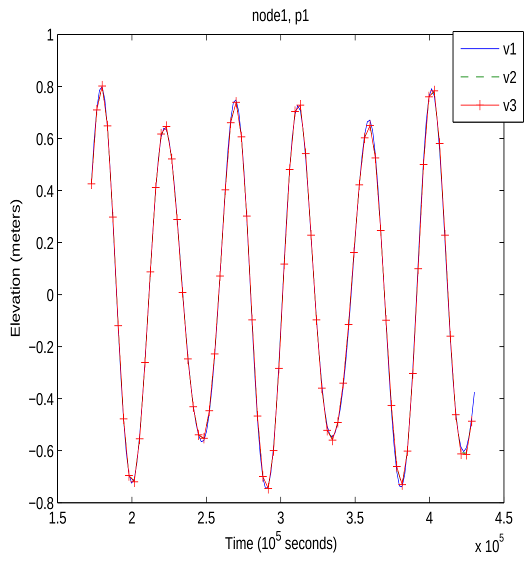


Location 3

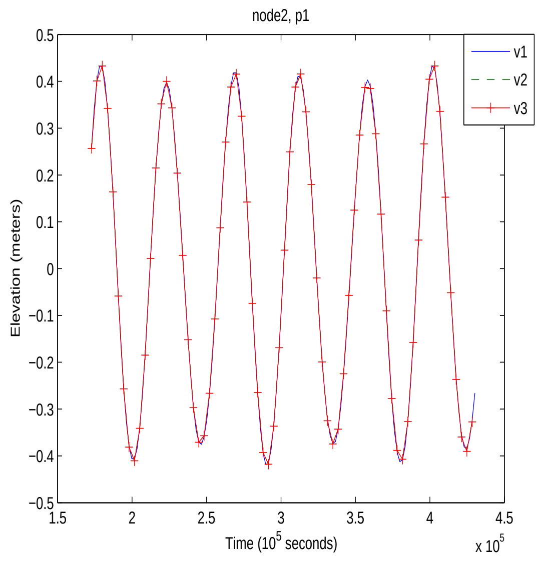


Location 5

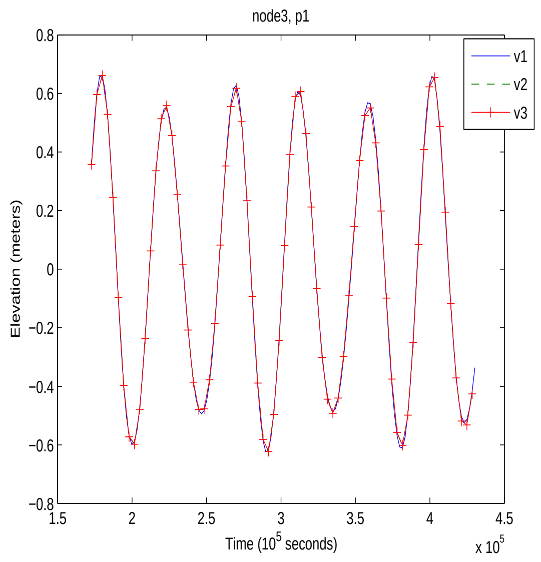
Figure 9: Comparison of UTBEST3D solutions for piecewise constants on coarse grid, medium grid and fine grid.



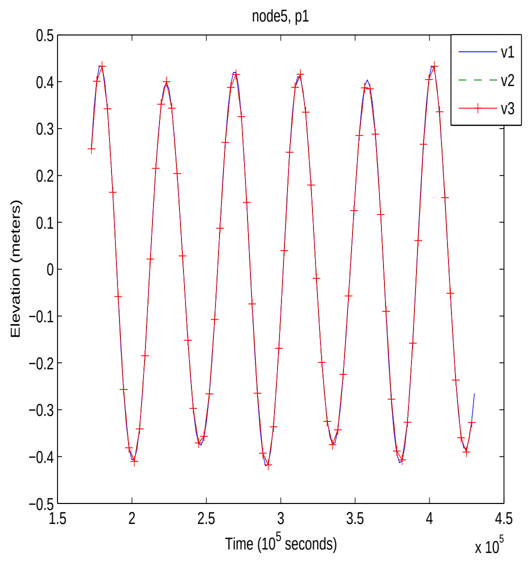
Location 1



Location 2

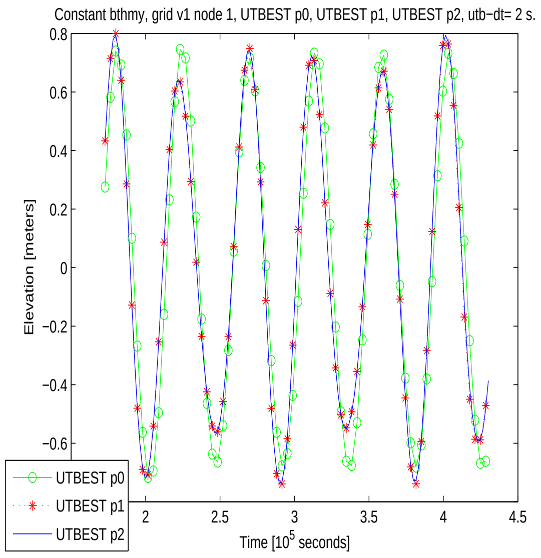


Location 3

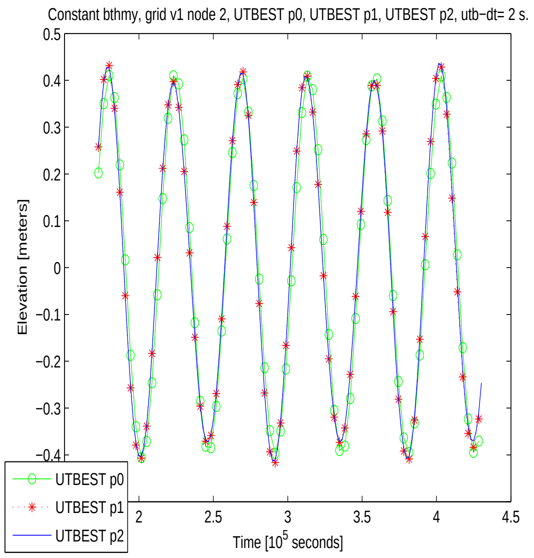


Location 5

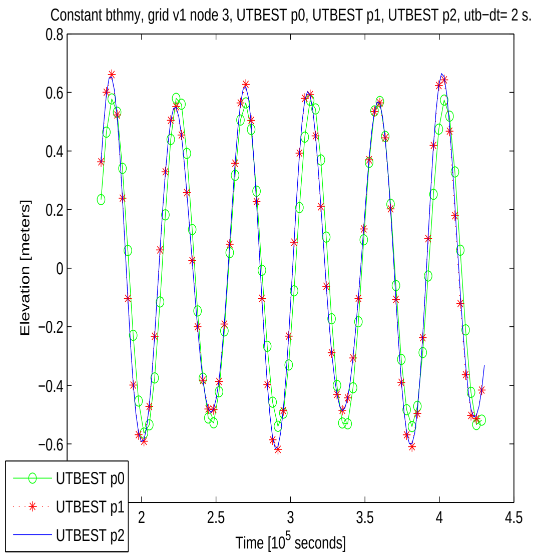
Figure 10: Comparison of UTBEST3D solutions for piecewise linears on coarse grid, medium grid and fine grid.



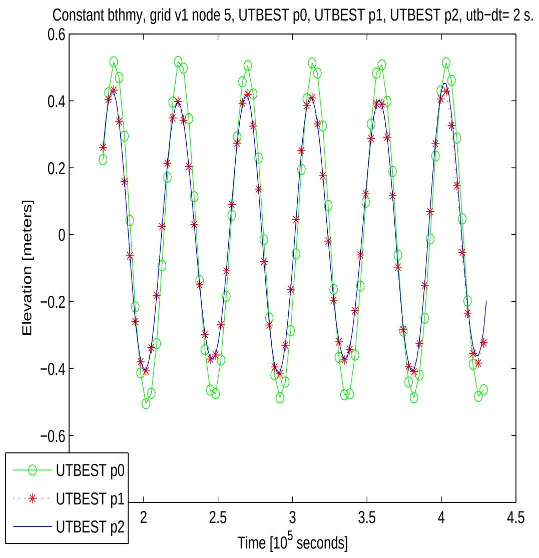
Location 1



Location 2

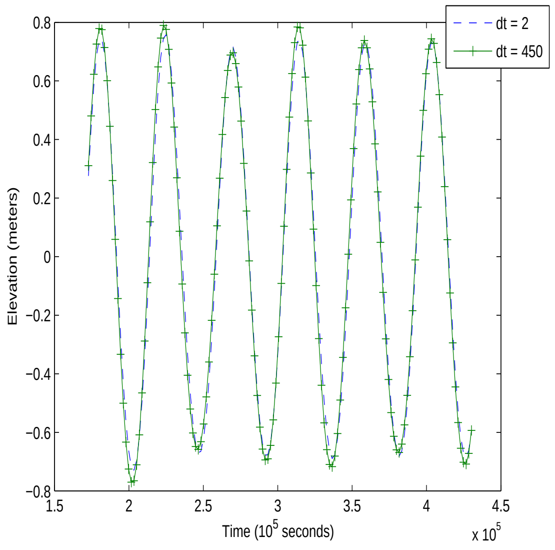


Location 3

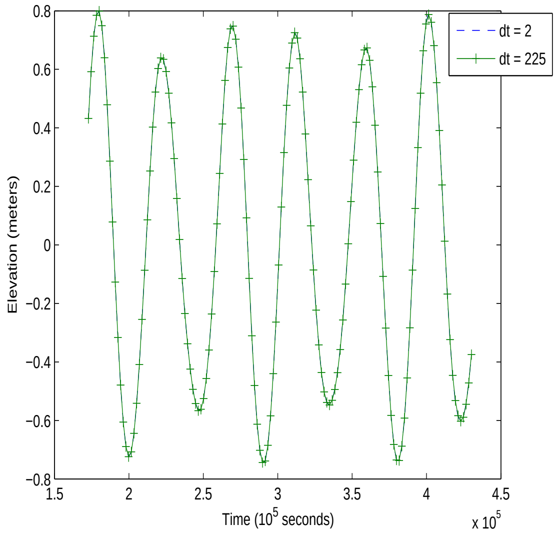


Location 5

Figure 11: Comparison of UTBEST3D solutions for piecewise constant, linear and quadratic solutions on the coarse grid.

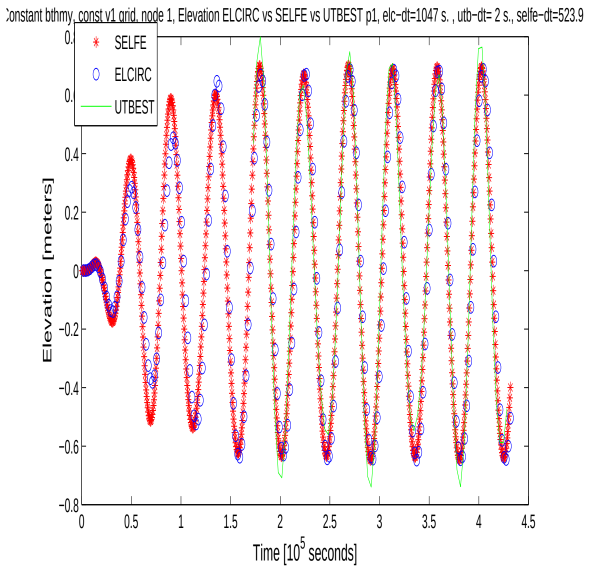


Constants: $\Delta t = 2$ vs. 450 seconds.

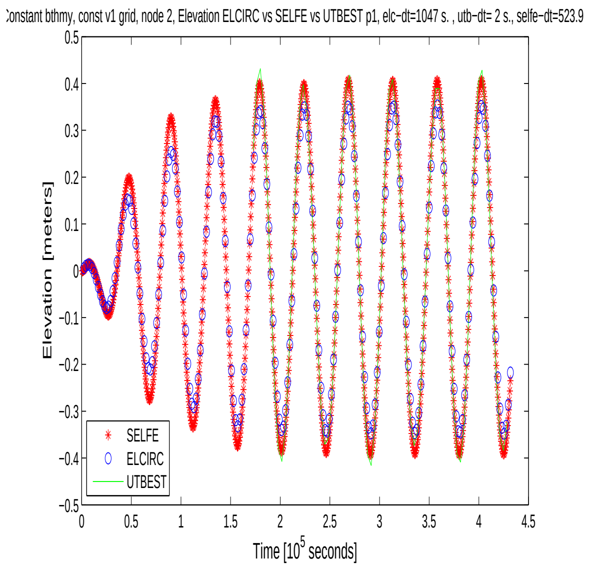


Linears: $\Delta t = 2$ vs. 225 seconds.

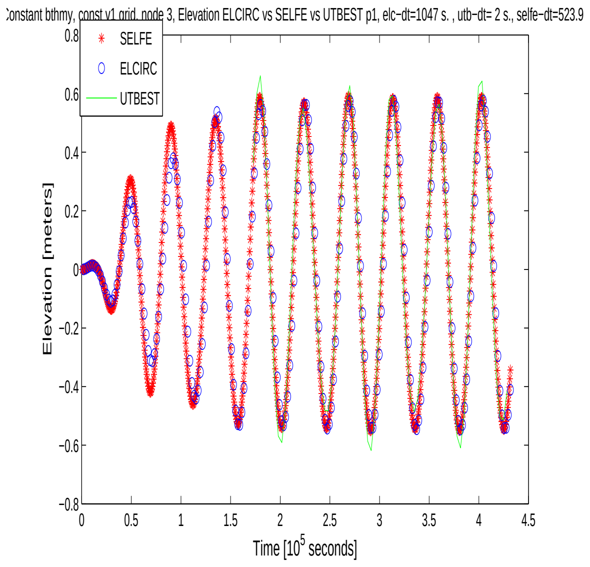
Figure 12: Comparison of small vs. large time steps for piecewise constants and piecewise linear solutions at location 1.



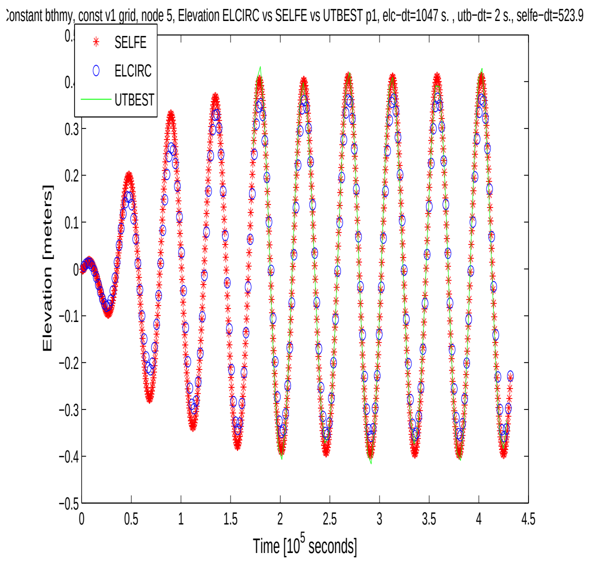
Location 1



Location 2

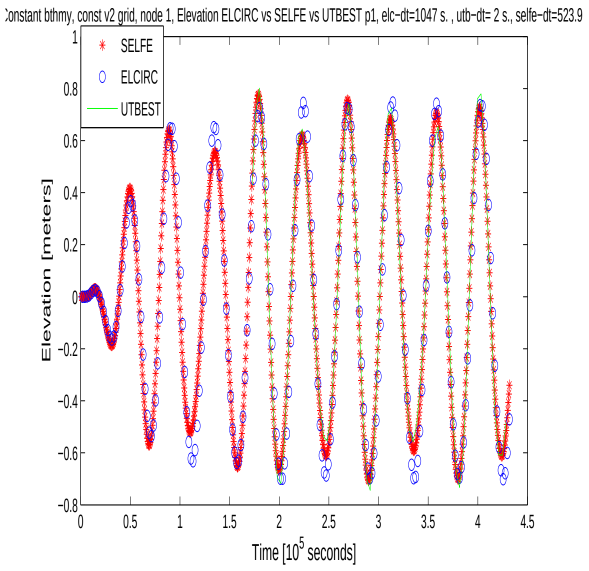


Location 3

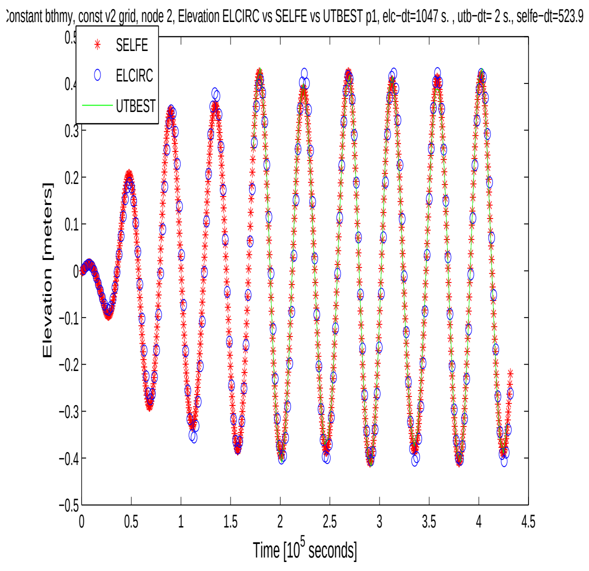


Location 5

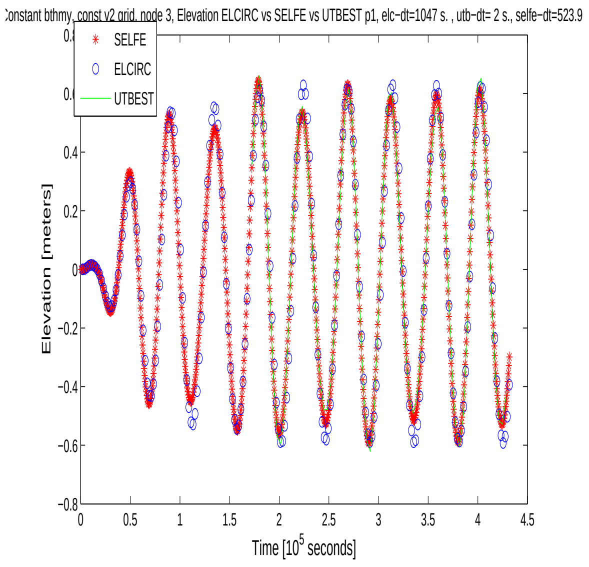
Figure 13: Comparison of UTBEST3D solutions for piecewise linears with ELCIRC and SELFE on coarse grid



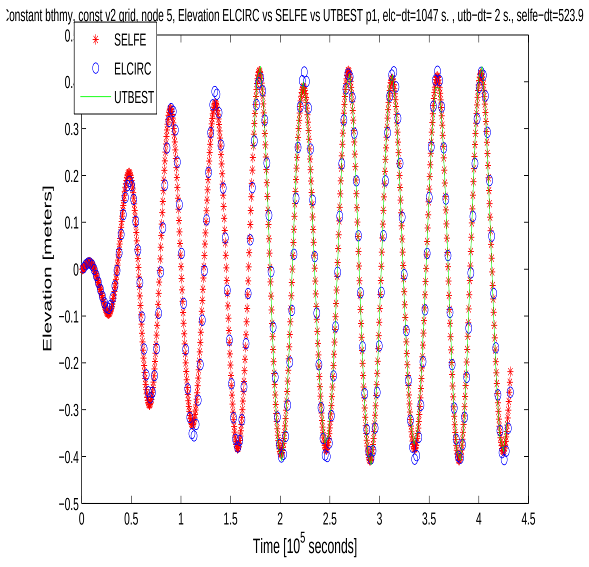
Location 1



Location 2

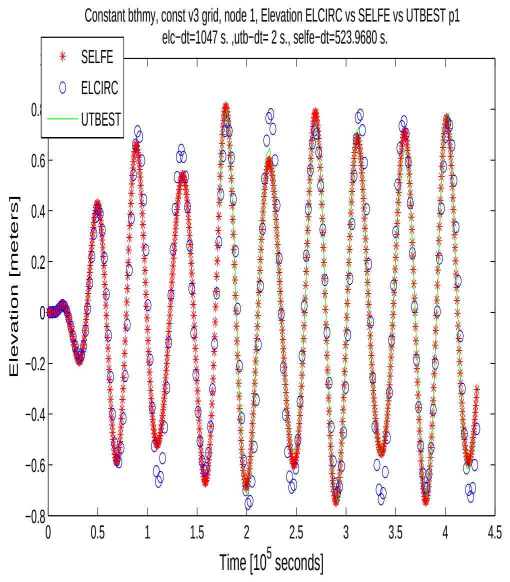


Location 3

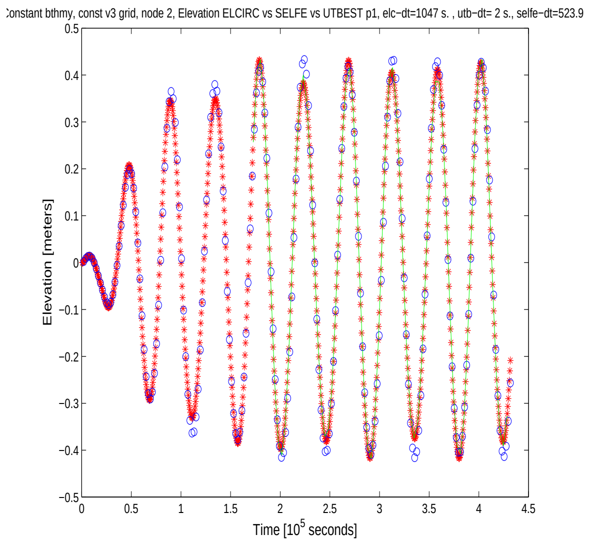


Location 5

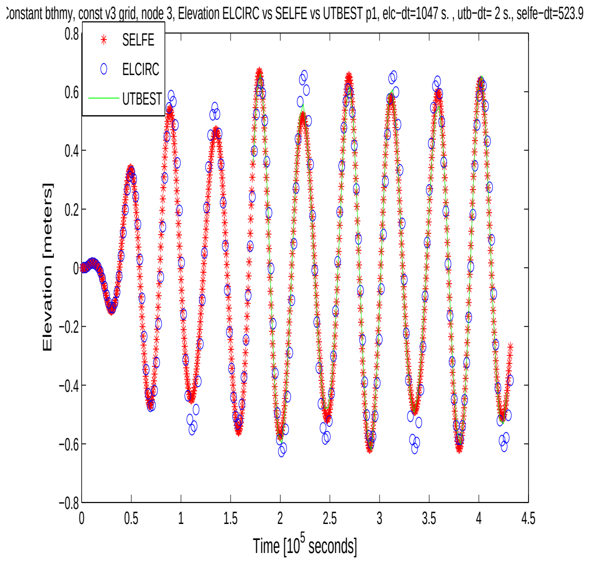
Figure 14: Comparison of UTBEST3D solution for piecewise linears with ELCIRC and SELFE on medium grid.



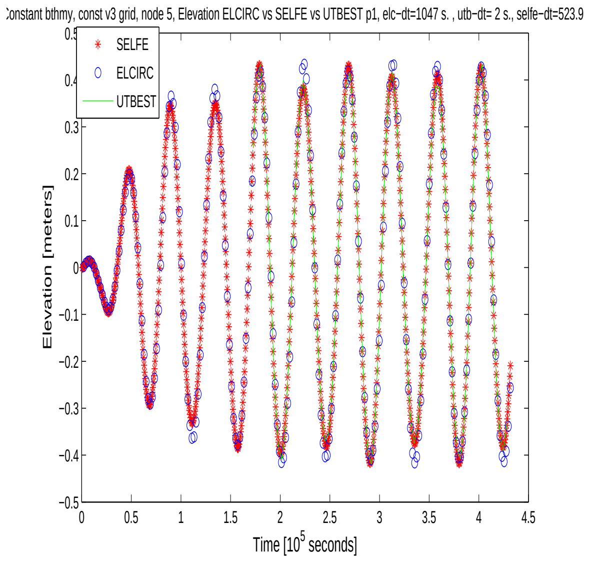
Location 1



Location 2



Location 3



Location 5

Figure 15: Comparison of UTBEST3D solution for piecewise linears with ELCIRC and SELFE on fine grid.

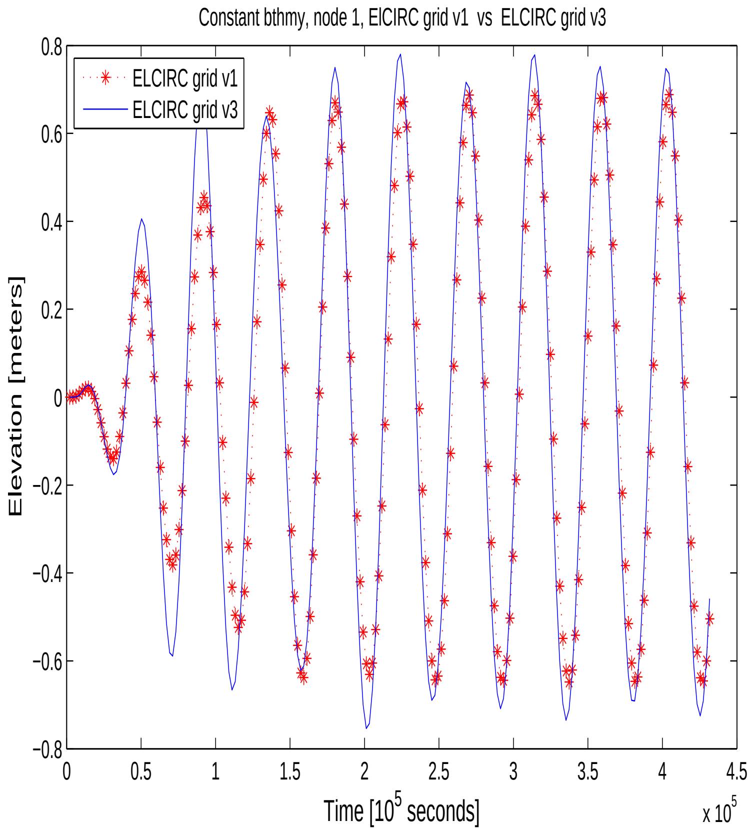


Figure 16: Comparison of ELCIRC solutions at location 1 on coarse and fine grids.

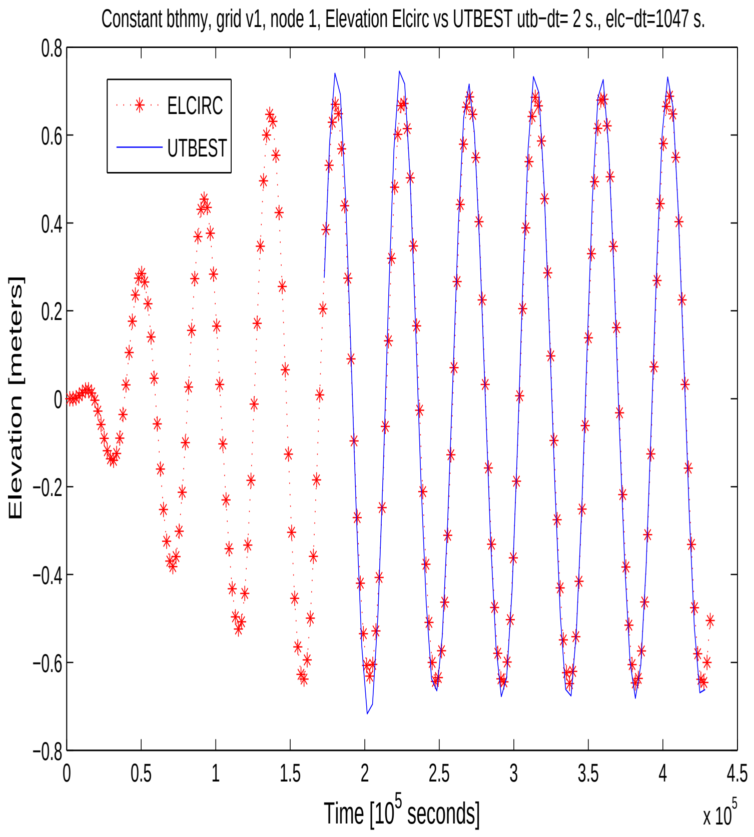


Figure 17: Comparison of ELCIRC and UTBEST3D piecewise constant solutions at location 1 on the coarse grid.