MOD_FreeSurf2D Manual

Nick Martin

April 12, 2004

Contents

List of Figures

List of Tables

1 Introduction

MOD_FreeSurf2D is a computer model to solve the depth-averaged, shallow water equations. The model was designed to meet three criteria: modularity, computational efficiency, and minimum data requirements. This model of 2D surface water flow was constructed to be one of a series of modules to simulate hydraulic and sedimentary processes. It can be run on a stand-alone basis to simulate fluid flow in rivers, streams, or shallow estuaries or as one of a series of components to simulate sedimentary processes. Computational efficiency is provided by the semi-implicit, semi-Lagrangian, finite-volume numerical representation. Semi-implicit and semi-Lagrangian representations allow model time step sizes that exceed the Courant-Friedrichs-Lewy (CFL) criterion [11, 12, 6]. Finally, the model was designed to require a minimum of user input information. Topography, initial water depth, and a Manning's roughness coefficient value must be specified in combination with inflow and or outflow boundary conditions in order to simulate free-surface flow. The model has minimal data requirements because channel boundaries do not require specification.

2 Governing Equations

The governing equations for MOD_FreeSurf2D are the depth-averaged, shallow water equations. To understand the limitations of these equations, it is instructive to trace their development from first principles. The application of Newton's second law to an element of fluid produces the Navier-Stokes equations, which accurately represent Newtonian fluid flow physics in very general scenarios [13]. To describe fluid flow in specific cases, simplifying assumptions can reduce the complexity of the full Navier-Stokes equations. The primitive variable equations, equations (1), (2), and (3), describe three-dimensional water flow in rivers, lakes, estuaries, and oceans given the assumption of hydrostatic pressure distribution and the use of Reynolds averaging. In these equations, $u(x, y, z, t)$, $v(x, y, z, t)$, and $w(x, y, z, t)$ are the directional velocity components in the horizontal $x-$ and $y-$ directions and the vertical $z-$ direction. t represents time, and $\eta(x, y, t)$ is the water surface elevation measured from the undisturbed water surface. f is the Coriolis parameter, μ is the horizontal eddy viscosity coefficient, and ν is the vertical eddy viscosity coefficient.

$$\begin{aligned}\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = \\ -g\frac{\partial \eta}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu\frac{\partial u}{\partial z}\right) + \mathsf{f}v\end{aligned} \tag{1}$$

$$\begin{aligned}\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} = \\ -g\frac{\partial \eta}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu\frac{\partial v}{\partial z}\right) - \mathsf{f}u\end{aligned} \tag{2}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{3}$$

Applying the assumptions of a well mixed water column and of a small water depth to width ratio to equations (1), (2), and (3) permits the vertical integration of primitive variable equations. The vertically integrated forms of $x-$ direction velocity, $U(x, y, t)$, and $y-$ direction velocity, $V(x, y, t)$, are given by equations (4) and (5). The vertical integration occurs between the bed, given by $-h$ where h is the undisturbed water depth, and the water surface, η, which is measured positive upwards from the undisturbed water depth. For a graphical definition of total water depth, H, undisturbed water depth, h, and free surface elevation, η see Figure 1.

$$U = \left(\frac{1}{H}\right) \int_{-h}^{\eta} u \; dz \tag{4}$$

$$V = \left(\frac{1}{H}\right) \int_{-h}^{\eta} v \; dz \tag{5}$$

Vertical integration of the conservation of momentum equations in the primitive variable equations, (1) and (2), requires boundary conditions at the top and bottom of the water column. A prescribed wind stress coefficient, γ_T, and prescribed wind velocities, U_a and V_a, provide the boundary condition, equation (6), at the water surface. A Manning-Chezy formula, equation (7), that employs a Chezy friction coefficient, Cz, produces the bottom boundary [6].

$$\tau_x^w = \nu \frac{\partial u}{\partial z} = \gamma_T \left(U_a - U\right) \qquad \tau_y^w = \nu \frac{\partial v}{\partial z} = \gamma_T \left(V_a - V\right) \tag{6}$$

$$\nu \frac{\partial u}{\partial z} = \frac{g\sqrt{U^2 + V^2}}{Cz^2}\; U \qquad \nu \frac{\partial v}{\partial z} = \frac{g\sqrt{U^2 + V^2}}{Cz^2}\; V \tag{7}$$

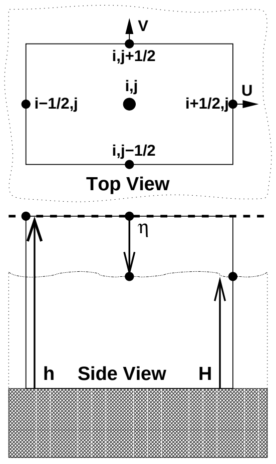

Figure 1: The finite volume layout for MOD_FreeSurf2D. Velocities, U and V are positioned at the center of volume faces. Total depth, H, and undisturbed depth, h, are located at volume edges. The free surface, η is measured from the volume center.

Vertical integration of the primitive variable equations with these water column boundary conditions yields the depth averaged, shallow water equations, equations (8), (9), and (10).

$$\begin{aligned}\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + V\frac{\partial U}{\partial y} = &- g\frac{\partial \eta}{\partial x} + \varepsilon\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right) + \frac{\gamma_T\big(U_a - U\big)}{H} \\ &- g\frac{\sqrt{U^2+V^2}}{Cz^2}U + \mathsf{f}V\end{aligned} \tag{8}$$

$$\begin{aligned}\frac{\partial V}{\partial t} + U\frac{\partial V}{\partial x} + V\frac{\partial V}{\partial y} = &- g\frac{\partial \eta}{\partial y} + \varepsilon\left(\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2}\right) + \frac{\gamma_T\big(V_a - V\big)}{H} \\ &- g\frac{\sqrt{U^2+V^2}}{Cz^2}V - \mathsf{f}U\end{aligned} \tag{9}$$

$$\frac{\partial \eta}{\partial t} + \frac{\partial (HU)}{\partial x} + \frac{\partial (HV)}{\partial y} = 0 \tag{10}$$

The depth averaged, shallow water equations are the governing equations for MOD_FreeSurf2D. Equations (8), (9), and (10) assume a hydrostatic pressure distribution, a well mixed water column, and a small depth to width ratio.

3 Numerical Approximations

A semi-implicit, semi-Lagrangian, finite volume numerical approximation represents the depth averaged, shallow water equations, equations (8), (9), and (10), in MOD_FreeSurf2D. The numerical representation is similar to the TRIM method [3, 6, 4, 15, 5]. The TRIM method provides a family of semi-implicit, semi-Lagrangian methods to represent free surface fluid flow. This family of methods includes a two- and three- dimensional finite volume method for hydrostatic flow [6], a three- dimensional finite element method for hydrostatic flow [15], and a three- dimensional non-hydrostatic method [5]. The numerical approximation in MOD_FreeSurf2D is similar to that of *Casulli and Cheng (1992)* for two-dimensional flow, except MOD_FreeSurf2D employs an improved semi-Lagrangian representation of the inertial and viscous terms.

The rectangular, finite volume grid layout for MOD_FreeSurf2D is the Arakawa C-grid, see Figure 1. Free surface elevation, η, is defined at the center of each computational volume. Total water depth, H, and directional velocity components, U and V, are defined at the midpoint of volume faces. Undisturbed water depth, h, is also defined at the midpoint of volume faces. The finite volume structure provides a control volume representation that is inherently mass conservative [7].

The combination of a semi-implicit free surface solution and a semi-Lagrangian representation of advection provides the advantages of a stable solution and of time steps that exceed the *CFL* criterion. In the semi-implicit process, the free surface elevation in the momentum equations, equations (8) and (9), and the velocity divergence in the continuity equation, equation (10), are treated implicitly, while the advective terms in the momentum equations, equations (8) and (9), are discretized explicitly. When the explicitly discretized advection terms are represented with an semi-Lagrangian approach, *Casulli (1990)* found that the *CFL* stability condition on time step duration could be relaxed. Additionally, *Robert (1981)* and *(1982* demonstrated that the association of a semi-Lagrangian treatment of advection and a semi-implicit representation of gravitational oscillations provides a sixfold increase in the maximum stable time step duration in atmospheric modeling.

3.1 Semi-Implicit Free Surface Approximation

Given the rectangular, finite volume grid layout, the two-dimensional continuity equation, equation (10), is approximated by equation (11) with the (i, j) subscript representing spatial location and the superscript, N or $N+1$, representing the temporal location. θ represents the degree of implicitness; $\theta = 1.0$ is fully implicit, and $1 > \theta \geq 0.50$ is semi-implicit. Δx and Δy represent the x- and y- direction volume lengths respectively. Δt is the computational time step duration. The θ value determines the degree of implicitness of the solution. An implicit, or semi-implicit, free surface elevation solution provides enhanced stability [3, 12].

$$\begin{aligned}
\eta_{i,j}^{N+1} =&\eta_{i,j}^{N} - \theta\frac{\Delta t}{\Delta x}\left(H_{i+1/2,j}^{N}U_{i+1/2,j}^{N+1} - H_{i-1/2,j}^{N}U_{i-1/2,j}^{N+1}\right)\\
&- \theta\frac{\Delta t}{\Delta y}\left(H_{i,j+1/2}^{N}V_{i,j+1/2}^{N+1} - H_{i,j_1/2}^{N}V_{i,j-1/2}^{N+1}\right)\\
&- (1-\theta)\frac{\Delta t}{\Delta x}\left(H_{i+1/2,j}^{N}U_{i+1/2,j}^{N} - H_{i-1/2,j}^{N}U_{i-1/2,j}^{N}\right)\\
&- (1-\theta)\frac{\Delta t}{\Delta y}\left(H_{i,j+1/2}^{N}V_{i,j+1/2}^{N} - H_{i,j_1/2}^{N}V_{i,j-1/2}^{N}\right)
\end{aligned} \tag{11}$$

3.2 Semi-Lagrangian Representation of Advection

With directional velocity terms defined at volume face midpoints, the numerical approximations to the conservation of momentum equations, equations (8) and (9) are given by equations (12) and (13). In these equations, the FU and FV terms are the semi-Lagrangian advection operators, and n is Manning's roughness coefficient.

$$\begin{aligned} U_{i+1/2,j}^{N+1} = \; & FU_{i+1/2,j}^{N} - (1-\theta)\frac{g\Delta t}{\Delta x}\left(\eta_{i+1,j}^{N} - \eta_{i,j}^{N}\right) - \theta\frac{g\Delta t}{\Delta x}\left(\eta_{i+1,j}^{N+1} - \eta_{i,j}^{N+1}\right) \\ & - g\Delta t\frac{\left[\left(U_{i+1/2,j}^{N}\right)^2 + \left(V_{i+1/2,j}^{N}\right)^2\right]^{0.5}}{Cz_{i,j}^2 H_{i+1/2,j}^{N}} U_{i+1/2,j}^{N+1} + \Delta t\frac{\gamma_T\left(U_a - U_{i+1/2,j}^{N+1}\right)}{H_{i+1/2,j}^{N}} \end{aligned} \tag{12}$$

$$\begin{aligned} V_{i,j+1/2}^{N+1} = \; & FV_{i,j+1/2}^{N} - (1-\theta)\frac{g\Delta t}{\Delta y}\left(\eta_{i,j+1}^{N} - \eta_{i,j}^{N}\right) - \theta\frac{g\Delta t}{\Delta y}\left(\eta_{i,j+1}^{N+1} - \eta_{i,j}^{N+1}\right) \\ & - g\Delta t\frac{\left[\left(U_{i,j+1/2}^{N}\right)^2 + \left(V_{i,j+1/2}^{N}\right)^2\right]^{0.5}}{Cz_{i,j}^2 H_{i,j+1/2}^{N}} V_{i,j+1/2}^{N+1} + \Delta t\frac{\gamma_T\left(V_a - V_{i,j+1/2}^{N+1}\right)}{H_{i,j+1/2}^{N}} \end{aligned} \tag{13}$$

$$Cz_{i,j} = \frac{H_{i,j}^{N^{1/6}}}{n_{i,j}} \tag{14}$$

The semi-Lagrangian advection operators, equations (15) and (16), represent the contribution of the advective, viscous, and Coriolis terms from the governing equations, (8) and (9). A semi-Lagrangian advection method employs a Lagrangian algorithm across the underlying Eulerian model grid. The Lagrangian component of the scheme traces the path line of particle initially located at a volume face, which is the velocity definition location from Figure 1, backwards along the particle path line a distance corresponding to the simulation time step duration, Δt. The particle departure point is the location of the particle at the beginning of the current time step. Again, this location is obtained by tracing the particle backwards along the path line [2]. The method is only semi-Lagrangian, and partially Eulerian, because the velocity value at the departure point is obtained by interpolation from the surrounding, known velocity values defined on the Eulerian grid.

$$\begin{aligned} FU^N_{i+1/2,j} =& U^N_{sL_{bicubic}} + \\ & \varepsilon\Delta t\left(\frac{U^N_{i+1/2-a+1,j-b} - 2U^N_{i+1/2-a,j-b} + U^N_{i+1/2-a-1,j-b}}{\Delta x^2}\right) \\ & + \varepsilon\Delta t\left(\frac{U^N_{i+1/2-a,j-b+1} - 2U^N_{i+1/2-a,j-b} + U^N_{i+1/2-a,j-b-1}}{\Delta y^2}\right) \\ & \mathsf{f}\Delta t V^N_{sL_{bilinear}} \end{aligned} \tag{15}$$

$$\begin{aligned} FV^N_{i,j+1/2} =& V^N_{sL_{bicubic}} + \\ & \varepsilon\Delta t\left(\frac{V^N_{i-a+1,j+1/2-b} - 2V^N_{i-a,j+1/2-b} + V^N_{i-a-1,j+1/2-b}}{\Delta x^2}\right) \\ & + \varepsilon\Delta t\left(\frac{V^N_{i-a,j+1/2-b+1} - 2V^N_{i-a,j+1/2-b} + V^N_{i-a,j+1/2-b-1}}{\Delta y^2}\right) \\ & \mathsf{f}\Delta t U^N_{sL_{bilinear}} \end{aligned} \tag{16}$$

In equations (15) and (16), the subscripts $sL_{bicubic}$ and $sL_{bilinear}$ denote bicubic and bilinear interpolation from the underlying Eulerian grid at the departure point. In the viscous terms, ε is the horizontal eddy viscosity which is set to a fixed value. The subscripts on the velocity terms in the viscous terms denote the location on the Eulerian grid relative to the departure point, $(i + 1/2 - a, j - b)$ in equation (15) where a is the x- direction Courant number rounded down, $a = U\frac{\Delta t}{\Delta x}$ and b is the y- direction Courant number rounded down, $b = V\frac{\Delta t}{\Delta y}$. The Coriolis f-parameter is calculated with a Coriolis f-plane model shown in equation (17) [8]. Equations (15) and (16) provide an explicit, semi-Lagrangian representation of advection.

$$\mathsf{f} = 2\Omega \sin \theta_0 \tag{17}$$

3.2.1 Path Line Tracing Methods

In MOD_FreeSurf2D, three different particle path line tracing methods are available. One method is the semi-analytical path line tracing method of *Pollock (1988).* The other two methods are the Runge-Kutta method and the Euler method which are explicit, linear, multi-step algorithms. All three methods employ a number of partial time steps to trace a particle path line to the departure point from the initial particle locations at the center of volume faces.

The semi-analytical path line tracing method was developed for particle tracking in ground water flow models [10]. The assumption that each directional velocity component varies linearly in its coordinate directions within each computational volume or cell underlies the method. Linear variation allows the derivation of an analytical expression for the path line of a particle across a volume. Although the semi-analytic method was developed for forward particle tracking, the method can easily be adapted to trace the particle path line back through time to find the particle location at the beginning of the time step.

To trace backwards, equations (18) and (19) allow the computation of the particle location along any point of the particle path line within a particular volume, see Figure 2. The exit point, (x_e, y_e), of a particle traversing a volume is calculated by finding the volume traversal time, τ_e in equation (24), given: the particle entry point to the computational volume, (x_p, y_p); the velocity component in the x-direction, equation (20), and in the y-direction, equation (21), at the entry point; and the velocity gradients across the volume in the x-direction, equation (22), and y-direction, equation (23). Once τ_e is determined, equations (18) and (19) generate the volume exit point. The partial time step, τ_e, is generated for each particle,p, and for each volume, k, that the particle traverses. A particle is initially placed at the center of each volume face so that $p = 1, \ldots, P$ for P total particles. The number of volumes traversed, K, is determined by the velocity field and by the model time step, Δt, and is given by equation (25). The semi-analytic method permits the tracing of a particle across an entire computational volume in one partial time step. As a result, the SUT method provides both accuracy and computational efficiency.

$$x_e = x_2 - \frac{1}{A_x}\left[U_{x2}^N - \frac{U_{xp}^N}{\exp(A_x\tau_e)}\right] \tag{18}$$

$$y_e = y_2 - \frac{1}{A_y}\left[V_{y2}^N - \frac{V_{yp}^N}{\exp(A_y\tau_e)}\right] \tag{19}$$

$$U_{xp}^N = U_{x2}^N - A_x\,(x_2 - x_p) \tag{20}$$

$$V_{yp}^N = V_{y2}^N - A_y\,(y_2 - y_p) \tag{21}$$

$$A_x = \frac{U_{x2}^N - U_{x1}^N}{\Delta x} \tag{22}$$

$$A_y = \frac{V_{y2}^N - V_{y1}^N}{\Delta y} \tag{23}$$

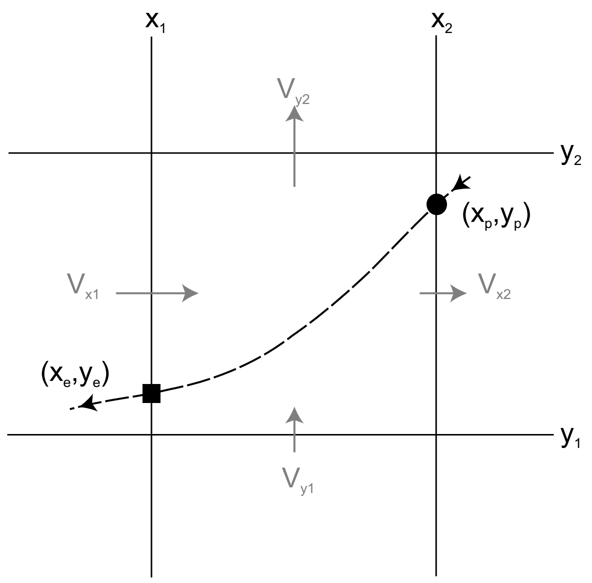

Figure 2: Schematic for the layout employed in the semi-analytical path line tracing method. x_e is the x-coordinate of the particle volume exit location. y_e is the y-coordinate of the particle volume exit location. x_p is the x-coordinate of the particle volume entry location. y_p is the y-coordinate of the particle volume entry location. U_{x2} is the defined velocity location on the downwind side of the volume in the x-direction. U_{x1} is the defined velocity location on the upwind side of the volume in x-direction. V_{y2} is the defined velocity location on the downwind side of the volume in the y-direction. V_{y1} is the defined velocity location on the upwind side of the volume in y-direction. x_2 is the x-coordinate for U_{x2}. x_1 is the x-coordinate for U_{x1}. y_2 is the x-coordinate for V_{y2}. y_1 is the x-coordinate for V_{y1}.

$$\tau_{e_{k,p}} = \min\left(\frac{1}{A_x}\ln\left[\frac{U^N_{xp}}{U^N_{x1}}\right], \frac{1}{A_y}\ln\left[\frac{V^N_{yp}}{V^N_{y1}}\right], \Delta t - \sum \tau_{e_{k,p}}\right) \tag{24}$$

$$\Delta t = \sum_{k=1}^{K} \tau_{e_{k,p}} \tag{25}$$

Fourth-order, four-step, explicit Runge-Kutta schemes, see equations (26) and (27) adapted from [16], have been widely employed for path line tracing. In this method, the partial time step, τ, is calculated with equation (29) which enforces the *CFL* criterion. The four step method in equations (26) and (27) provides the particle location at the end of each partial time step. Each of the velocity values in these equations is obtained with bilinear interpolation from the underlying Eulerian grid, and the subscript b on the time level (i.e. N) denotes bilinear interpolation. After M partial time steps, the particle has been traced backwards along the path line to location the departure point. Each partial time step will move the particle no farther than one computational volume backwards because of the *CFL* restriction contained in equation (29). The Runge-Kutta method is the most accurate of the three methods and is the most computationally expensive.

$$x_{s-1} = x_s - \frac{\tau}{6}\left(U^{N_b}_{x_s,y_s} + 2U^{N_b}_{x_{sp1},y_{sp1}} + 2U^{N_b}_{x_{sp2},y_{sp2}} + U^{N_b}_{x_{sp3},y_{sp3}}\right) \tag{26}$$

$$x_{sp1} = x_s + U^{N_b}_{x_s,y_s}\frac{\tau}{2}$$

$$x_{sp2} = x^s + U^{N_b}_{x_{sp1},y_{sp1}}\frac{\tau}{2}$$

$$x_{sp3} = x^s + U^{N_b}_{x_{sp2},y_{sp2}}\tau$$

$$y_{s-1} = y_s - \frac{\tau}{6}\left(V^{N_b}_{x_s,y_s} + 2V^{N_b}_{x_{sp1},y_{sp1}} + 2V^{N_b}_{x_{sp2},y_{sp2}} + V^{N_b}_{x_{sp3},y_{sp3}}\right) \tag{27}$$

$$y_{sp1} = y_s + V^{N_b}_{x_s,y_s}\frac{\tau}{2}$$

$$y_{sp2} = y_s + V^{N_b}_{x_{sp1},y_{sp1}}\frac{\tau}{2}$$

$$y_{sp3} = \; y_s + V^{N_b}_{x_{sp2},y_{sp2}}\tau$$

$$s = \; M, M-1, M-2, \ldots, 2, 1 \tag{28}$$

$$\tau \leq \min\left[\frac{\Delta x}{\max_{i,j}|U|}, \frac{\Delta y}{\max_{i,j}|V|}\right] \tag{29}$$

One-step, first order, explicit Euler schemes are also commonly used for path line tracing. Equations (30) and (31) provide the particle location at the end of each partial time step using the Euler method. Again, the subscript b on the time level denotes bilinear interpolation. Just as with the Runge-Kutta method, equation (29) generates the partial time step duration. The Euler scheme is the least accurate of the three methods but is the most computationally efficient.

$$x_{s-1} = x_s - U^{N_b}_{x_s,y_s}\tau \tag{30}$$

$$y_{s-1} = x_s - V^{N_b}_{x_s,y_s}\tau \tag{31}$$

All three particle tracking methods will generate relatively accurate simulations for model simulation time step sizes that exceed the *CFL* criterion. The Runge-Kutta method will always provide the best departure point location. The SUT method often provides equivalent accuracy to the Runge-Kutta method and give improved accuracy relative to the Euler method. Although the partial time steps in the semi- Lagrangian algorithm are limited by the *CFL* restriction, the entire model time step is limited by the semi-implicit solution to the free surface equation. Because the Lagrangian algorithm follows the motion of the fluid, the representation provides improved accuracy in the representation of advection.

3.3 Implementation of Numerical Representation

To provide a compact notational form, the A and G variables in equations (32) and (33) are defined [6]. When substituted into the numerical velocity representation, equations (12) and (13), a notationally compact velocity representation is obtained, equations (34) and (35). The A and G variables are also employed in MOD_FreeSurf2D as part of the calculation algorithm.

$$A^N_{i+1/2,j} = H^N_{i+1/2,j} + \Delta t \gamma_T + g\Delta t \frac{\left[\left(U^N_{i+1/2,j}\right)^2 + \left(V^N_{i+1/2,j}\right)^2\right]^{0.5}}{Cz^2} \tag{32}$$

$$\begin{aligned} G^N_{i+1/2,j} =& H^N_{i+1/2,j} F U^N_{i+1/2,j} - H^N_{i+1/2,j}\left(1-\theta\right)\frac{g\Delta t}{\Delta x}\left(\eta^N_{i+1,j} - \eta^N_{i,j}\right) \\ &+ \Delta t \gamma_T U_a \end{aligned} \tag{33}$$

$$U^{N+1}_{i+1/2,j} = \frac{G^N_{i+1/2,j}}{A^N_{i+1/2,j}} - \theta \frac{g\Delta t}{\Delta x}\left(\eta^{N+1}_{i+1,j} - \eta^{N+1}_{i,j}\right)\frac{H^N_{i+1/2,j}}{A^N_{i+1/2,j}} \tag{34}$$

$$V^{N+1}_{i,j+1/2} = \frac{G^N_{i,j+1/2}}{A^N_{i,j+1/2}} - \theta \frac{g\Delta t}{\Delta y}\left(\eta^{N+1}_{i,j+1} - \eta^{N+1}_{i,j}\right)\frac{H^N_{i,j+1/2}}{A^N_{i,j+1/2}} \tag{35}$$

Equation (11) has multiple unknowns, η^{N+1}, U^{N+1}, and V^{N+1}. Before a semi-implicit solution can be obtained for the system of equations for free surface elevation for each volume in the simulation domain, equation (11) must be transformed into an equation for one unknown variable, η^{N+1}. Substituting equations (34) and (35) yields equation (36) which has only free surface elevations as unknowns.

Arranging the unknowns, $N+1$ terms, on the left side and the knowns, N terms, on the right side provides the system of equations for free surface elevation represented by equation (37). This system is penta-diagonal, positive definite and is solved with the preconditioned conjugate gradient method. This system is easily normalized to yield the identity matrix as the main diagonal [6]. Equations (38) and (39) provide a notationally compact form of the right-hand side of the free surface system of equations. equation (40) is the compact form of the left-hand side of the system.

$$\begin{aligned}
\eta_{i,j}^{N+1} =&\eta_{i,j}^{N} - \theta\frac{\Delta t}{\Delta x}\{H_{i+1/2,j}^{N}\left[\frac{G_{i+1/2,j}^{N}}{A_{i+1/2,j}^{N}} - \theta\frac{g\Delta t}{\Delta x}\left(\eta_{i+1,j}^{N+1} - \eta_{i,j}^{N+1}\right)\frac{H_{i+1/2,j}^{N}}{A_{i+1/2,j}^{N}}\right] \\
&- H_{i-1/2,j}^{N}\left[\frac{G_{i-1/2,j}^{N}}{A_{i-1/2,j}^{N}} - \theta\frac{g\Delta t}{\Delta x}\left(\eta_{i,j}^{N+1} - \eta_{i-1,j}^{N+1}\right)\frac{H_{i-1/2,j}^{N}}{A_{i-1/2,j}^{N}}\right]\} \\
&- \theta\frac{\Delta t}{\Delta y}\{H_{i,j+1/2}^{N}\left[\frac{G_{i,j+1/2}^{N}}{A_{i,j+1/2}^{N}} - \theta\frac{g\Delta t}{\Delta y}\left(\eta_{i,j+1}^{N+1} - \eta_{i,j}^{N+1}\right)\frac{H_{i,j+1/2}^{N}}{A_{i,j+1/2}^{N}}\right] \\
&- H_{i,j-1/2}^{N}\left[\frac{G_{i,j-1/2}^{N}}{A_{i,j-1/2}^{N}} - \theta\frac{g\Delta t}{\Delta y}\left(\eta_{i,j}^{N+1} - \eta_{i,j-1}^{N+1}\right)\frac{H_{i,j-1/2}^{N}}{A_{i,j-1/2}^{N}}\right]\} \\
&- (1-\theta)\frac{\Delta t}{\Delta x}\left(H_{i+1/2,j}^{N}U_{i+1/2,j}^{N} - H_{i-1/2,j}^{N}U_{i-1/2,j}^{N}\right) \\
&- (1-\theta)\frac{\Delta t}{\Delta y}\left(H_{i,j+1/2}^{N}V_{i,j+1/2}^{N} - H_{i,j_1/2}^{N}V_{i,j-1/2}^{N}\right)
\end{aligned} \tag{36}$$

$$
\begin{aligned}
\Bigg[1 &+ \frac{g\theta^2\Delta t^2}{\Delta x^2}\frac{H^N_{i+1/2,j}}{A^N_{i+1/2,j}} + \frac{g\theta^2\Delta t^2}{\Delta x^2}\frac{H^N_{i-1/2,j}}{A^N_{i-1/2,j}} \\
&\quad + \frac{g\theta^2\Delta t^2}{\Delta x^2}\frac{H^N_{i,j+1/2}}{A^N_{i,j+1/2}} + \frac{g\theta^2\Delta t^2}{\Delta y^2}\frac{H^N_{i,j-1/2}}{A^N_{i,j-1/2}}\Bigg]\eta^{N+1}_{i,j} \\
&- \left[\frac{g\theta^2\Delta t^2}{\Delta x^2}\frac{H^N_{i+1/2,j}}{A^N_{i+1/2,j}}\right]\eta^{N+1}_{i+1,j} - \left[\frac{g\theta^2\Delta t^2}{\Delta x^2}\frac{H^N_{i-1/2,j}}{A^N_{i-1/2,j}}\right]\eta^{N+1}_{i-1,j} \\
&- \left[\frac{g\theta^2\Delta t^2}{\Delta y^2}\frac{H^N_{i,j+1/2}}{A^N_{i,j+1/2}}\right]\eta^{N+1}_{i,j+1} - \left[\frac{g\theta^2\Delta t^2}{\Delta y^2}\frac{H^N_{i,j-1/2}}{A^N_{i,j-1/2}}\right]\eta^{N+1}_{i,j-1} \\
&= \eta^N_{i,j} - \frac{\theta\Delta t}{\Delta x}\left[\frac{H^N_{i+1/2,j}G^N_{i+1/2,j}}{A^N_{i+1/2,j}} - \frac{H^N_{i-1/2,j}G^N_{i-1/2,j}}{A^N_{i-1/2,j}}\right] \\
&\quad - \frac{\theta\Delta t}{\Delta y}\left[\frac{H^N_{i,j+1/2}G^N_{i,j+1/2}}{A^N_{i,j+1/2}} - \frac{H^N_{i,j-1/2}G^N_{i,j-1/2}}{A^N_{i,j-1/2}}\right] \\
&\quad - \frac{(1-\theta)\Delta t}{\Delta x}\left(H^N_{i+1/2,j}U^N_{i+1/2,j} - H^N_{i-1/2,j}U^N_{i-1/2,j}\right) \\
&\quad - \frac{(1-\theta)\Delta t}{\Delta x}\left(H^N_{i+1/2,j}U^N_{i+1/2,j} - H^N_{i-1/2,j}U^N_{i-1/2,j}\right)
\end{aligned}
\tag{37}
$$

$$
\begin{aligned}
\delta_{i,j} =&\eta^N_{i,j} - \frac{(1-\theta)\,\Delta t}{\Delta x}\left(H^N_{i+1/2,j}U^N_{i+1/2,j} - H^N_{i-1/2,j}U^N_{i-1/2,j}\right) - \\
&\frac{(1-\theta)\,\Delta t}{\Delta y}\left(H^N_{i,j+1/2}V^N_{i,j+1/2} - H^N_{i,j-1/2}V^N_{i,j-1/2}\right)
\end{aligned}
\tag{38}
$$

$$
\begin{aligned}
RHS =&\delta_{i,j} - \frac{\theta\Delta t}{\Delta x}\frac{H^N_{i+1/2,j}G^N_{i+1/2,j}}{A^N_{i+1/2,j}} + \frac{\theta\Delta t}{\Delta x}\frac{H^N_{i-1/2,j}G^N_{i-1/2,j}}{A^N_{i-1/2,j}} - \\
&\frac{\theta\Delta t}{\Delta y}\frac{H^N_{i,j+1/2}G^N_{i,j+1/2}}{A^N_{i,j+1/2}} + \frac{\theta\Delta t}{\Delta y}\frac{H^N_{i,j-1/2}G^N_{i,j-1/2}}{A^N_{i,j-1/2}}
\end{aligned}
\tag{39}
$$

$$
\begin{aligned}
\Big[1 + \frac{g\theta^2\Delta t^2}{\Delta x^2}\frac{H^{N^2}_{i+1/2,j}}{A^N_{i+1/2,j}} + \frac{g\theta^2\Delta t^2}{\Delta x^2}\frac{H^{N^2}_{i-1/2,j}}{A^N_{i-1/2,j}} + \\
\frac{g\theta^2\Delta t^2}{\Delta y^2}\frac{H^{N^2}_{i,j+1/2}}{A^N_{i,j+1/2}} + \frac{g\theta^2\Delta t^2}{\Delta y^2}\frac{H^{N^2}_{i,j-1/2}}{A^N_{i,j-1/2}}\Big]\eta^{N+1}_{i,j} \\
-\Big[\frac{g\theta^2\Delta t^2}{\Delta x^2}\frac{H^{N^2}_{i+1/2,j}}{A^N_{i+1/2,j}}\Big]\eta^{N+1}_{i+1,j} - \Big[\frac{g\theta^2\Delta t^2}{\Delta x^2}\frac{H^{N^2}_{i-1/2,j}}{A^N_{i-1/2,j}}\Big]\eta^{N+1}_{i-1,j} \\
-\Big[\frac{g\theta^2\Delta t^2}{\Delta y^2}\frac{H^{N^2}_{i,j+1/2}}{A^N_{i,j+1/2}}\Big]\eta^{N+1}_{i,j+1} - \Big[\frac{g\theta^2\Delta t^2}{\Delta y^2}\frac{H^{N^2}_{i,j-1/2}}{A^N_{i,j-1/2}}\Big]\eta^{N+1}_{i,j-1} \\
= LHS
\end{aligned}
\tag{40}
$$

4 Wetting and Drying

One advantage of the MOD_FreeSurf2D, and of the TRIM method and of most methods that employ the Arakawa C-grid, is a natural and elegant treatment of wetting and drying boundaries on the underlying Eulerian grid. The finite volume layout in Figure 1 portrays an individual volume with total water depth, H, and undisturbed water depth, h, located at volume faces. The free surface elevation for each volume, η, is defined at the volume center. Employing this layout, the total water depth across the simulation domain at each time step is updated with equations (41) and (42). The total water depth is the sum of the free surface elevation, $\eta_{i,j}^{N+1}$ or $\eta_{i+1,j}^{N+1}$, and the undisturbed water depth, $h_{i+1/2,j}$. Consequently, volume faces wet or dry in a conservative manner as flow conditions warrant. The model automatically calculates the water/land boundary without requiring additional user input.

$$H_{i+1/2,j}^{N+1} = max\left(0, h_{i+1/2,j} + \eta_{i,j}^{N+1}, h_{i+1/2,j} + \eta_{i+1,j}^{N+1}\right) \tag{41}$$

$$H_{i,j+1/2}^{N+1} = max\left(0, h_{i,j+1/2} + \eta_{i,j}^{N+1}, h_{i,j+1/2} + \eta_{i,j+1}^{N+1}\right) \tag{42}$$

5 Boundary Conditions

In a general categorization, domain boundaries can be either open or closed [1]. Closed boundaries are boundaries that do not allow water flow while open boundaries are simulation domain boundaries across which water may flow. Closed boundaries do not need specification in MOD_FreeSurf2D because equations (41) and (42) will determine the closed boundary locations in the simulation domain. However, open boundaries must be specified in the program.

Both Dirichlet and radiation boundary conditions can be applied to open boundaries in MOD_FreeSurf2D. Dirichlet conditions are available for velocity, total water depth, and for water flux see equations (43), (44), and (45) respectively. The subscript Bnd denotes a location on the boundary. Dirichlet total water depth and water flux boundaries should only be applied to inflow boundaries. Dirichlet velocity conditions may be applied to both inflow and outflow boundaries [1].

$$V_{Bnd} = Constant \tag{43}$$

$$H_{Bnd} = Constant \tag{44}$$

$$H_{Bnd} U_{Bnd} = Constant \tag{45}$$

Two types of radiation boundary conditions are available. Both radiation conditions should only be applied to outflow open boundaries. The first type of radiation boundary is a radiation condition, equation (46), for normal velocity, subscript n, at the domain boundary. The drift velocity term, U_{upw}, in this condition is simply upwinded, normal directional velocity component [14]. This simple condition does an adequate job of allowing information to propagate out of the simulation domain.

$$\frac{\partial U}{\partial t} + U_{upw}\frac{\partial U}{\partial n} = 0 \tag{46}$$

Another radiation boundary condition available, equation (47) [9], is applied to the free surface elevation, η. *Orlanski (1976)* developed this boundary condition to limit, or absorb, wave reflections at open boundaries. The core of the method is the calculation of a propagation velocity, C, from grid points surrounding the boundary with the leap-frog finite difference representation in equation (48). The subscript Bnd represents the boundary location. The propagation velocity, C, is then employed with a leap-frog finite difference stencil for free surface elevation to calculate the free surface elevation at the boundary, see equation (49) for the new time step, $N+1$.

$$\frac{\partial \eta}{\partial t} + C\frac{\partial \eta}{\partial n} = 0 \tag{47}$$

$$C = \; -\frac{\left[\eta^{N}_{Bnd-1/2,j} - \eta^{N-2}_{Bnd-1/2,j}\right]}{\left[\eta^{N}_{Bnd-1/2,j} + \eta^{N-2}_{Bnd-1/2,j} - \eta^{N-1}_{Bnd-3/2,j}\right]}\,\frac{\Delta x}{2\Delta t} \tag{48}$$

$$\eta^{N+1}_{Bnd,j} = \frac{\left[1-(\Delta t/\Delta x)\,C\right]}{\left[1+(\Delta t/\Delta x)\,C\right]}\eta^{N-1}_{Bnd,j} + \frac{2\,(\Delta t/\Delta x)\,C}{\left[1+(\Delta t/\Delta x)\,C\right]}\eta^{N}_{Bnd-1/2,j}$$

for $0 \;\le C \;<\; \Delta x/\Delta t$ and

$$\eta^{N+1}_{Bnd,j} = \eta^{N}_{Bnd-1/2,j} \tag{49}$$

for $C \;\ge\; \Delta x/\Delta t$

6 Using MOD_FreeSurf2D

MOD_FreeSurf2D is a collection of **Matlab** scripts (**.m* files). The advantages of having the program within **Matlab** are that the program will work on any operating system that **Matlab** works on without modification and that the output, visual and otherwise, from the program is easily customizable to the user's needs. Additionally, **Matlab** provides the ability to compile the program as **C** code using the **Matlab Compiler**. The primary disadvantage is that the program requires that user have **Matlab**.

MOD_FreeSurf2D requires a minimum of four input files and can employ a maximum of eight input files. The input files provide the topography, initial water depth, Manning's roughness coefficient, initial velocities, and source and parameter specification for a simulation. Both the input files and the **.m* scripts that compose the program must be located within the **Matlab** path for the program to run correctly. The suggested protocol is to add the directory containing the **.m* scripts to the **Matlab** path with the *addpath* command and to run the program from the directory location of the input files. So, the current directory should contain the input files. The actual program files should be located in another directory that is part of the **Matlab** path. When the program files are within the **Matlab** path and the input files are located in the current directory, the program is run by typing *MOD_FreeSurf2D* at the **Matlab** command prompt. Any output files will be written to the current directory.

6.1 Domain Layout

In plan view layout, the simulation domain is divided into rectangular computational volumes. Figure 3 displays the numbering of volumes in the model according to the layout of rows, labeled i, and columns, labeled j. A computational volume's number corresponds to the volume's index in computational vectors and matrices. Equation (50) provides the volume index that uniquely identifies each volume in the domain. This index provides the location of a value in calculation vectors for variables that are defined at volume centers. Examples of volume center variables are topographic elevation, undisturbed water depth, and Manning's n values. Volume face indexes are employed to reference quantities defined at the center of volume faces, like directional velocity components and total water depth. Equation (51) gives the calculation for x-face volume indexes, and equation (52) provides the y-face indexes. In Figure 3 x-face are represented by x's. An additional column, relative to volume center variables, exists for all x-face variables. The y-face variable locations are denoted with squares in Figure 3 and have an additional row relative to volume-center variables.

$$Volume\ Index = \Big(\big(i-1\big) \times \big(\#\ of\ columns\big)\Big) + j \tag{50}$$

$$X - face\ Index = \Big(\big(i-1\big) \times \big(\#\ of\ columns + 1\big)\Big) + l \tag{51}$$

$$Y - face\ Index = \Big(\big(k-1\big) \times \big(\#\ of\ columns\big)\Big) + j \tag{52}$$

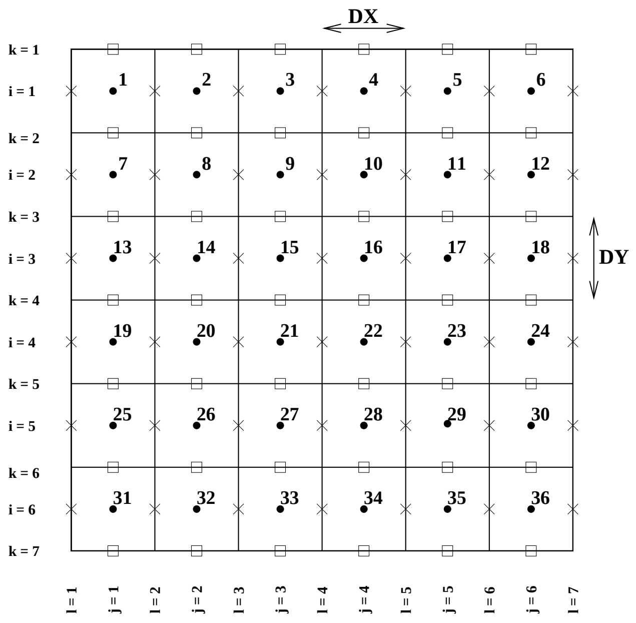

Figure 3: Schematic diagram of the plan view computation mesh employed in the program MOD_FreeSurf2D. The filled circles are the volume center locations for variables like η. The numbers provides the volume index for the center location for each volume. The squares are y- face locations for variables like V and H. The x's mark the x- face locations for variables like U and H.

6.2 Input Files

MOD_FreeSurf2D requires at least four input files and may use up to eight input files. The four required input files are **input.txt**, **Topo.txt**, **Depth.txt**, and **Mann.txt**. Optional input files are **TopoX.txt**, **TopoY.txt**, **BU.txt**, and **BV.txt**.

6.2.1 input.txt

The file **input.txt** provides parameter specification and source specification to MOD_FreeSurf2D. Comment lines begin with #. The file is read using *fscanf* and all values must be separated by a space. Additionally, each line should have one space or whitespace character before the newline. Sources are specified by volume center index, see Figure 3 for the source location.

The source must be specified as either and x-face or a y-face location and the specified source location should be for a volume adjacent to a simulation boundary. When multiple source volumes are specified, the listing of source volumes must be on one line. For example, if a Dirichlet total depth, inflow open boundary source is located at the left side of Figure 3 at volumes 13 and 19 with depth values of 1.0 and 1.5 meters respectively; this source would be entering the domain across an x-face (the domain boundary face for volumes 13 and 19). To specify this source, $TDEPXVOL$ = [13 19], and $TDEPXDEP$ = [1.0 1.5] in the **input.txt** file. The order of depth values in $TDEPXDEP$ corresponds to the order of the source locations in $TDEPXVOL$. All boundary conditions are stipulated in a similar fashion.

An example **input.txt** file is given below and displays the layout and syntax of source/parameter file that the program is expecting. This example input file applies to a different simulation than the example discussed above. A description of each parameter and source variable can be found in the Global Parameters and Variables Section, Section 8.

```
# input.txt
# This file provides input PARAMETERS and sources for
# MOD_FreeSurf2D. This file should have '#' at the start of
# every comment line otherwise all values should have whitespace
# before the variable value and whitespace after the variable value.
# This means that every variable should be followed by a
# space before the end line character.
#
#
# ##################################
# Simulation layout parameters.
#
# Start time for simulation in hours.
STARTTIME = 0.0
# # End time in simulation in hours.
ENDTIME = 0.01944
#
# Simulation name.
SIMNAME = db04x2_mn0075
#
# If DATUM equals -9999 then the program will calculate the DATUM.
# Otherwise the DATUM value will be set to the user's specification.
DATUM = -9999
#
# DX is the x-direction volume dimension in meters.
DX = 0.1000
#
# DY is the y-direction volume dimension in meters.
DY = 0.2500
#
# NUMROWS is the number of rows in the domain layout.
NUMROWS = 86
# NUMCOLS is the number of columns in the domain layout.
NUMCOLS = 16
# OUTINT is the output interval in number of time steps.
OUTINT = 1
# #############################
# Time stepping and computational parameters.
#
# Time interval in seconds for fluid flow calculations.
FLUID_DT = 0.069
#
# Degree of implicitness parameter. THETA of 1.0 is fully implicit.
# THETA of 0.5 is semi-implicit.
THETA = 1.0
#
# Minimum allowable depth.
HCUTOFF = 0.01
#
# Convergence criterion for pre-conditioned conjugate gradient solver.
EPSILON = 1E-12
#
```

```
# Maximum number of iterations for pcg solver.
MAXITER = 100
#
# Pre-conditioner to employ with solver.
# 0 is no preconditioner.
# 1 is Jacobi preconditioner.
# 2 is incomplete Cholesky factorization.
PRECOND = 2
#
# ##############################
# Path line tracing parameters.
#
# Tracing method. 1 use SUT Method. 2 use Runge-Kutta 3 use the
# Euler method.
PATHTRAC = 1
#
# Maximum number of partial steps for Runge-Kutta and Euler.
MAXSTEPS = 1000
#
# Minimum number of partial steps for Runge-Kutta and Euler.
MINSTEPS = 3
#
# Maximum Courant number to limit path line tracing across this
# number of volumes.
MAXCR = 50
#
##############################
#
# Assorted computational constants.
#
# Gravitational constant in [m/s^2]
G = 9.8
#
# Rouse Number.
KAPPA = 0.40
#
# Manning's n for wall roughness — not tested yet.
MNWAL = 0.0
#
# Kinematic viscosity of water in [m^2/s]
NUK = 0.000001
#
# Density of water [kg/m^3]
RHOW = 1000
#
# Eddy viscosity coefficient.
```

```
EVIS = 0.000001
#
############################
# Coriolis Parameters - this model employs an f-plane Coriolis
# model.
# Central latitude for the simulation domain.
CENTRALLATITUDE = 38
#
# Rotation rate of the earth [1/s]
COROMEGA = 0.73E-4
#
############################
# Wind surface stress parameter set-up. This needs to be tested.
# Have never actually employed values for these.
#
# Wind stress coefficient in the x-direction.
GAMMATX = 0.0
#
# Wind speed in x-direction.
UA = 0
#
# Wind stress coefficient in the y-direction.
GAMMATY = 0.0
#
# Wind speed in the y-direction.
VA = 0
#
############################
# Source/Boundary Condition information.
# The variables that give the volume locations (e.g. TDEPDXVOL)
# need to have the volumes listed in the following format —
# [ Vol1 Vol2 Vol3 ... ]
#
# Inflow boundary conditions/source conditions.
#
# Dirichlet Total Water Depth
# 1 turns on Dirichlet total water depth sources, 0 turns off.
TDEPDIRCBC = 0
#
# X-face Dirichlet total depth boundaries.
TDEPDXVOL = [ 0 ]
# Corresponding depth in meters.
TDEPDXDEP = 0
#
# Y-face Dirichlet total depth boundaries.
TDEPDYVOL = [ 0 ]
```

```
# Corresponding depth in meters.
TDEPDYDEP = 0.0
#
#
# Dirichlet Velocity boundary. Can be either inflow or outflow.
# Velocity is a vector quantity for this BC.
#
# 1 turns on Dirichlet velocity sources, 0 turns off.
VELDIRCBC = 0
#
# X-face boundaries.
VELDXVOL = [ 0 ]
# corresponding velocities in [m/s].
VELDXVEL = 0.0
#
# Y-face boundaries.
VELDYVOL = [0];
# Corresponding velocities in [m/s].
VELDYVEL = 0.0
#
#
# Dirichlet Flux inflow boundary. This boundary allows the
# specification of the total flux entering the domain for each
# specified volume.
# Only specify magnitude, not direction.
#
# 1 turns on Dirichlet flux sources, 0 turns off.
QINBC = 0
#
# X-face sources
QINXVOL = [ 0 ]
# Corresponding flux in [m^2/s]
QINXFLUX = 0.0
#
# Y-face sources
QINYVOL = [ 0 ]
# Corresponding flux in [m^2/s]
QINYFLUX = 0.0
#
#
# Radiation boundary conditions. Only use for outflow.
#
#
# 1. Velocity radiation. Sets the velocity with a radiation
# condition.
#
```

```
# 1 turns on radiation velocity outflow, 0 turns off.
RADVELBC = 1
#
# X-Face boundaries.
RVELXVOL = [ 0 ]
# Y-Face boundaries.
RVELYVOL = [ 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372
1373 1374 1375 1376 ]
#RVELYVOL = [ 0 ]
#
#
# 2. Free surface absorbing radiation. Sets the free surface with
# a radiation condition. This BC comes from Orlanski (1976).
#
# 1 turns on radiation free surface sources, 0 turns off.
RADORLFSBC = 0
#
# X-Face sources
RORLFSXVOL = [ 0 ]
# Y-Face sources
RORLFSYVOL = [ 0 ]
#
#
# 3. Flux radiation. Sets the velocity by employing a specified
# flux value. For this boundary the free surface absorbing
# radiation boundaries must be set. This condition, then, simply
# enforces a specified total flux across all of the radiation
# absorbing x-face or y-face boundary volumes. This boundary
# condition is over specified.
#
# 1 turns on radiation flux sources, 0 turns off.
RADFLUXBC = 0
#
# X-Face flux in [m^2/s].
RFLUXXFLUX = 0.0
# Y-Face flux in [m^2/s].
RFLUXYFLUX = 0.0
#
#EOF
```

6.2.2 Topo.txt, Depth.txt, and Mann.txt

The three other required input files, **Topo.txt**, **Depth.txt**, and **Mann.txt**, provide variables defined at volume centers. **Topo.txt** gives the topographic elevations at each volume center location. **Depth.txt** provides undisturbed water depth at each volume center, and **Mann.txt** gives the values for Manning's roughness coefficient which is also defined at the volume center location. The format for each of these files is a row, column text file. The location of each value in Table 1 should correspond to the volume index location from Figure 3. Each value should be whitespace separated and each row should end with a newline. The value of topographic elevation, undisturbed water depth, and Manning's n provided by each of these files represents this value at the center of the volume.

Volume 1	Volume 2	Volume 3	Volume 4	Volume 5	Volume 6
Volume 7	Volume 8	Volume 9	Volume 10	Volume 11	Volume 12
Volume 13	Volume 14	Volume 15	Volume 16	Volume 17	Volume 18
Volume 19	Volume 20	Volume 21	Volume 22	Volume 23	Volume 24
Volume 25	Volume 26	Volume 27	Volume 28	Volume 29	Volume 30
Volume 31	Volume 32	Volume 33	Volume 34	Volume 35	Volume 36

Table 1: Format for the files **Topo.txt**, **Depth.txt**, and **Mann.txt**. Volume elevation or bathymetry values should be in meters. The delimiters between values can be either spaces or tabs.

6.2.3 TopoX.txt, TopoY.txt, BU.txt, and BV.txt

Four optional input files may be provided by the user. **TopoX.txt**, **TopoY.txt**, **BU.txt**, and **BV.txt** provide MOD_FreeSurf2D with additional information. **TopoX.txt** and **TopoY.txt** give topographic elevations for x-face and y-face locations. These two files provide the topography at the definition location for total water depth and for directional velocity components. The format for these two files is a column vector format. Each line number represents the index location given by either equation (51) or (52). Table 2 displays an example of this format. If **TopoX.txt** and **TopoY.txt** are not provided, the program will calculate topographic elevation at these locations using linear interpolation between the values provided by **Topo.txt**.

BU.txt and **BV.txt** give MOD_FreeSurf2D initial velocity values. **BU.txt** provides the beginning x-direction velocity values, U, and **BV.txt** gives the initial V values. If initial velocity values are not provided, MOD_FreeSurf2D initializes all velocity values to zero. The format for these two files is also column vector format. Each line number represents the index location given by either equation (51) or (52). Table 2 displays an example of this format.

Face Index 1
Face Index 2
Face Index 3
Face Index 4
Face Index 5
Face Index 6
$\vdots \quad \vdots \quad \vdots$

Table 2: Format for the files **TopoX.txt**, **TopoY.txt**, **BU.txt**, and **BV.txt**. Volume face elevations or should be in meters. Velocity values should be in meters per second. Each line should end with a newline.

7 Model Outline

I. Global Initialization and Setup

   A. **Program Wrapper**
      `MOD_FreeSurf2D.m` is the file that will call the initialization and setup scripts and that will call the fluid flow program wrapper, `fluid.m`.

   B. **Read Parameter Values and Source Information**
      `rEADiNPUT.m` obtains parameter values, source information, and boundary information from the file **input.txt**. See Section 6.2.1.

   C. **Determine Numeric Precision Values**
      `sETpRECISION.m` sets the smallest real number with the **Matlab** *eps* function. Other important numeric precision values are also set including the square root of the smallest real number and the minimum water depth.

   D. **Set-up Simulation Domain**
      `sETdOMAIN.m` reads the input files **Topo.txt**, **Depth.txt**, and **Mann.txt** and sets domain layout parameters and index vectors. `gENfACEiNDEX.m` creates most of the index vectors. If **TopoX.txt** and **TopoY.txt** exist, `sETdOMAIN.m` will read in these files also. Otherwise, `sETtOPO.m` is called to interpolate volume face topographic elevation values.

   E. **Initialize Remaining Global Variables**
      `fLUIDiNIT.m` initializes the global variables that have not been initialized by `rEADiNPUT.m` and `sETdOMAIN.m`.

   F. **Start Fluid Flow Calculations**
      `fluid.m` is fluid flow program main wrapper. The remainder of the program will execute through `fluid.m`. The program will only return to `MOD_FreeSurf2D` for final wrap-up and exit.

II. Fluid Flow Calculation Set-up
   The script `fluid.m` handles the set-up for fluid flow calculations. This script will also call the `mainfluid.m` program where the the core of the fluid flow calculations are completed.

   A. **Assign Initial Values**
      Given the information obtained from the input files, assign initial values for all calculation vectors. `sETiNITIALvECTORS.m` sets values for h and for η. `sETtOTALdEPTH` sets values for H, and

`vELsET.m` obtains U and V from **BU.txt** and **BV.txt** if they exist. Output files are also opened and quantities for mass balance calculations are prepared by `fLUIDmASSsETUP.m`

B. **Initial Source and Boundary Condition Set-up**
Source declarations and boundary conditions that need to set initial velocity or water depth values are handled.

C. **Determine Number of Time Steps**
Determine the number of time steps so that can implement an integer **for** loop for efficiency. Also, set counter variables before main loop.

III. Main Loop — March Through Time

The main loop is the time loop that will handle fluid flow progression from the initial time to the end time. `mainfluid.m` contains the functions calls and calculations for the main part of the program.

A. **Calculate Chezy Coefficient**
`cHEZYcALC.m` calculates the Chezy coefficient, Cz, with equation (14).

B. **Get Values for A Vectors**
`sETafACES.m` generates the A calculation vector with equation (32). Values for A are returned in denominator form ($1/A$) because A values will be employed in the denominator for the remaining calculations.

C. **Calculate Semi-Lagrangian Advection Operators**
`fUXcALC.m` and `fVYcALC.m` produce the FU and FV terms respectively. Both of these functions employ either the semianalytical path line tracing method in `sEMIapATH.m` or the classical Runge-Kutta path line tracing method in `rkiNT.m`.

D. **Get Values for G Vectors**
`sETgfACES.m` calculates the values for the G calculation vectors, see equation (33).

E. **Free Surface Calculation**
`fREEsURF.m` provides the interface to the main fluid program for the functions that generate a solution to the free surface system of equations.

1. Generate $\delta_{i,j}$
   `dELTAcALC.m` calculates the value for δ using equation (38).
2. Calculate RHS of System
   `rhscALC.m` produces the right-hand side of the system of equations. The right-hand side is displayed in equation (39).

3. Produce the LHS of System
   `lhscALC.m` provides the left-hand side of the system and normalizes both the right- and left- hand sides. The left-hand side is represented by equation (40).
4. Pre-conditioned Conjugate Gradient Solution
   The free surface system of equations is solved with pre-conditioned conjugate gradients (pcg). The solver employed is the *pcg* solver in **Matlab**.
5. Return η^{N+1}
   Un-normalize the newly obtained η value and return to `mainfluid.m`.

F. **Explicit Velocity Update**
Now that have values for η^{N+1}, calculate values for x- and y- face velocities with equations (34) and (35). `vELcALC.m` is the function for the velocity update.

G. **Calculate New Total Water Depth**
`sETtOTALdEPTH.m` calculates new values of total water depth, H, using equations (41) and (42).

H. **Update Calculation Variables**
`aVEdEPTHcALC.m` sets new face depths that are indexed by volume and sets the new average depth, defined at the volume center, for each volume. `vELpARAMsET.m` sets volume averaged velocity values and velocity direction vectors.

IV. Wrap Up

Return to `fluid.m` to close the remaining open files. Also, write U, V, and H to text files with `rIToUTPUT.m`. These output files are written to the current directory. Return to `MOD_FreeSurf2D.m` to write out stopwatch calculation time and exit from program.

8 Global Parameters and Variables

This section provides a list of the parameters and variables that are employed globally in the MOD_FreeSurf2D. The parameter values are passed to the program from the file **input.txt**, see Section 6.2.1. The variables, used globally, are initialized in either `sETdOMAIN.m` or in `fLUIDiNIT.m`. In terms of naming conventions, parameters are listed in all capitals. Variables have a mixture of lowercase and uppercase characters. Generally, the first letter of each word that is joined/modified to create the variable name will be capitalized and the remainder of the name will be lower case. Usually, function names will have the first letter of each word that is joined/modified to create the function name in lowercase with the remainder of the name capitalized.

aXDen$(NUMINCX)$ is the denominator form of the **A** calculation vector, see equation (32), for x-face locations. The denominator form is simply $1/A$.

aYDen$(NUMINCY)$ is the denominator form of the **A** calculation vector, see equation (32), for y-face locations. The denominator form is simply $1/A$.

BH$(NUMNODES)$ A boolean variable that tells if each volume center is wet. $1 =$ wet and $0 =$ dry.

BHux$(NUMINCX)$ A boolean variable that tells if each x-face is wet or not. $1 =$ wet and $0 =$ dry.

BHvy$(NUMINCY)$ A boolean variable that tells if each y-face is wet or not. $1 =$ wet and $0 =$ dry.

CENTRALLATITUDE The central latitude, θ_0 of the simulation domain for use with the *f-plane* Coriolis model, see equation (17).

COROMEGA The earth's rotation rate, Ω to be used in Coriolis f-plane model from equation (17). 0.73E-4 $1/seconds$.

Cz($NUMNODES$) The Chezy coefficient, from equation (14), for each computational volume. Defined at the volume center.

CzX($NUMINCX$) The Chezy coefficient, from equation (14), for each volume x-face. Obtained by upwinding.

CzY($NUMINCY$) The Chezy coefficient, from equation (14), for each volume y-face. Obtained by upwinding.

DATUM The reference level employed to calculate the free surface height and the undisturbed water depth. **DATUM** is the elevation of the reference level given in *meters*.

DX The length of the rectangular cells in the x-direction in *meters*.

DY The length of the rectangular cells in the y-direction in *meters*.

EVIS Specified value for eddy viscosity, ε.

ENDTIME The time in hours at which the simulation ends.

EPSILON Convergence criteria for the conjugate gradient solver.

Eta($NUMNODES$) Free surface elevation in *meters* for each node at time step $= N$. **Eta** corresponds to η^N.

EtaNew($NUMNODES$) Free surface elevation in *meters* for each node at time step $= N + 1$. **EtaNew** corresponds to η^{N+1}.

EtaXP1($NUMINCX$) Current value of η arranged so that each value represents $(i + 1, j)$ for each x-face index $(i + 1/2, j)$.

EtaXM1($NUMINCX$) Current value of η arranged so that each value represents (i, j) for each x-face index $(i + 1/2, j)$.

EtaYP1($NUMINCY$) Current value of η arranged so that each value represents $(i, j + 1)$ for each y-face index $(i, j + 1/2)$.

EtaYM1($NUMINCY$) Current value of η arranged so that each value represents (i, j) for each y-face index $(i, j + 1/2)$.

FCOR Coriolis f-term (f) in the Navier-Stokes equations. This variable is calculated with equation (17) in the function `fcORcALC.m`.

FLUID_DT The time step in seconds.

Fux($NUMINCX$) X-face advection operator. See $FU^N_{i+1/2,j}$ in equation (15).

Fvy($NUMINCY$) Y-face advection operator. See $FV^N_{i,j+1/2}$ in equation (16).

G Gravitational constant $= 9.8\ meters/second$.

GAMMATX Wind stress coefficient in the x-direction (γ_{tx}).

GAMMATY Wind stress coefficient in the y-direction (γ_{ty}).

gX($NUMINCX$) The **G** calculation vector from equation (33) for x-face locations.

gY($NUMINCY$) The **G** calculation vector from equation (33) for y-face locations.

H($NUMNODES$) Average total water depth for each volume. Or total water depth defined at volume center.

h1($NUMNODES$) Total water depth for the east face, $(i+1/2, j)$, of each volume (i, j).

h2($NUMNODES$) Total water depth for the south face, $(i, j-1/2)$, of each volume (i, j).

h3($NUMNODES$) Total water depth for the west face, $(i-1/2, j)$, of each volume (i, j).

h4($NUMNODES$) Total water depth for the north face, $(i, j+1/2)$, of each volume (i, j).

HCUTOFF The minimum allowable water depth for calculations in meters. This value will be made greater or equal to the precision value given by **PRECH**.

HDEPTH($NUMNODES$) The undisturbed water depth from the file **Depth.txt** at volume center locations.

HNODE($NUMNODES$) The undisturbed water depth from **DATUM** at volume center locations. This value is obtained from **HDEPTH**.

HUX($NUMNODES$) The undisturbed water depth, $h_{i+1/2,j}$ at x-face locations.

Hux($NUMINCX$) The total water height at each x-face location. **Hux** is equivalent to $H_{i+1/2,j}$. See equation (41).

HVY($NUMNODES$) The undisturbed water depth, $h_{i,j+1/2}$, at y-face locations.

Hvy($NUMINCY$) The total water height at each y-face location. **Hvy** is equivalent to $H_{i,j+1/2}$. See equation (42).

KAPPA Von Karmann's number ≈ 0.40.

LASTROW The index for the last volume in the second to last row. $ULASTROW = (NUMROWS - 1) * NUMCOLS$

MAXITER The maximum number of iterations for the conjugate gradient solver.

MN($NUMNODES$) Manning's n value for each volume. Obtained from the file **Mann.txt**.

MnX($NUMINCX$) Manning's n upwinded to x-faces.

MnY($NUMINCY$) Manning's n upwinded to y-faces.

MAXCR Maximum number of volumes to traverse as part of the path line tracing algorithm, Section 3.2.1.

MAXSTEPS The maximum number of partial time steps, equation (29), to employ for the Runge-Kutta method of path line tracing.

MINSTEPS The minimum number of partial time steps, equation (29), to employ for the Runge-Kutta method of path line tracing.

NUMCOLS The number of columns in the domain. This value is calculated from the input topography file.

NUMINCX The number of x-face locations in the simulation domain. See equation (51).

NUMINCY The number of y-face locations in the simulation domain. See equation (52).

NUK The kinematic viscosity of water = 1E-6 $meters^2/s$.

NUMNODES The total number of nodes in the simulation domain.

NUMROWS The number of rows in the simulation domain. This value is calculated from the input topography file.

OUTINT The output interval or the number of time steps between writing output information to the output file **Output.txt**.

PATHTRAC Select the path line tracing method, see Section 3.2.1. 1 = the semi-analytic method, or 2 = the Runge-Kutta method.

PREC Precision of the computer. For *Matlab*, the precision is equivalent *eps*.

PRECH Minimum allowable depth.

PRECOND The preconditioner to employ for the conjugate gradient solver. 0 is no preconditioner. 1 is a Jacobi preconditioner. 2 is incomplete Cholesky factorization.

QINBC Turn on Dirichlet boundary conditions for total flux flowing into the domain, equation (45). This boundary condition allows the specification of a fixed water flux into the domain. 0 = turn off. 1 = turn on.

QINXFLUX($\# \ of \ specified \ x - face \ volumes$) The water flux coming into the domain across the face, that corresponds to the volume index provided in **QINXVOL**. Flux should be in $meters^2/second$.

QINYFLUX($\# \ of \ specified \ y - face \ volumes$) The water flux coming into the domain across the face, that corresponds to the volume index provided in **QINYVOL**. Flux should be in $meters^2/second$.

QINXVOL($\# \ of \ specified \ x - face \ volumes$) Volume indexes corresponding to volumes that have a x-face that is also a domain boundary that is chosen as a specified inflow boundary. Volume indexes can be calculated with equation (50).

QINYVOL($\# \ of \ specified \ y - face \ volumes$) Volumes that have a y-face that is also a domain boundary that is chosen as a specified inflow boundary. Volume indexes can be calculated with equation (50).

qRHS($NUMNODES$) The right-hand side of the free surface system of equations from equation (39).

RADFLUXBC Turn on radiation flux boundary condition. 0 = off and 1 = on. This boundary condition is over specified and will set the outflow velocity for each specified volume in order to provide the specified flux value. The free surface elevation that is calculated to provide the outflow depth is calculated with **RADORLFSBC**.

RADORLFSBC Turn on radiation free surface boundary condition, equations (47), (48), and (49). 0 = off. 1 = on.

RFLUXXFLUX Total flux in cubic meters per second across all of the x-faces specified by **RORLFSXVOL**.

RFLUXYFLUX Total flux in cubic meters per second across all of the y-faces specified by **RORLFSYVOL**.

RINC Increment for obtaining columns of values from a vector format variable for y-faces. Equal to **NUMCOLS**.

RADVELBC Turn on radiation velocity outflow boundary condition, equation (46). 0 = off. 1 = on.

RORLFSXVOL($\#\ of\ specified\ x-face\ volumes$) Volume indexes corresponding to volumes that have a x-face that is also a domain boundary that is chosen as a radiation free surface boundary. Volume indexes can be calculated with equation (50).

RORLFSYVOL($\#\ of\ specified\ y-face\ volumes$) Volume indexes corresponding to volumes that have a y-face that is also a domain boundary that is chosen as a radiation free surface boundary. Volume indexes can be calculated with equation (50).

ROWBEGIN($NUMROWS$) Index corresponding to the first volume in every row.

ROWEND($NUMROWS$) Index corresponding to the last volume in every row.

RVELXVOL($\#\ of\ specified\ x-face\ volumes$) Volume indexes corresponding to volumes that have a x-face that is also a domain boundary that is chosen as a radiation velocity boundary. Volume indexes can be calculated with equation (50).

RVELYVOL($\#\ of\ specified\ y-face\ volumes$) Volume indexes corresponding to volumes that have a y-face that is also a domain boundary that is chosen as a radiation velocity boundary. Volume indexes can be calculated with equation (50).

RHOW Density of water. Density is considered constant in this program and is 1000 $kilograms/meters^3$.

Sides($NUMNODES$) The number of wet volume faces for each volume.

sLHS($NUMNODES, NUMNODES$) The left-hand side for the free surface system of equations. See equation (40).

SIMNAME The simulation name that is output to the file **Mass.txt**.

STARTTIME The time in hours at which the simulation starts.

TDEPDIRCBC Turn on Dirichlet boundary conditions for total water depth, equation (44). This boundary condition allows the specification of a fixed water depth inflow boundary. 0 = turn off. 1 = turn on.

TDEPDXDEP($\# \ of \ x - face \ specified \ volumes$) The depth for each x-boundary face that is specified with **TDEPDXVOL**. Each index in **TDEPDXDEP** corresponds to the same index in **TDEPDXVOL**. Depth should be given in $meters$.

TDEPDYDEP($\# \ of \ y - face \ specified \ volumes$) The depth for each y-boundary face that is specified with **TDEPDYVOL**. Each index in **TDEPDYDEP** corresponds to the same index in **TDEPDYVOL**. Depth should be given in $meters$.

TDEPDXVOL($\# \ of \ x - face \ specified \ volumes$) The volume indexes for domain boundaries, which are x-faces, to apply Dirichlet total water depth boundary conditions. Volume indexes can be calculated with equation (50). Specified volumes must be situated on the simulation domain edge/boundary.

TDEPDYVOL($\# \ of \ y - face \ specified \ volumes$) The volume indexes for domain boundaries, which are y-faces, to apply Dirichlet total water depth boundary conditions. Volume indexes can be calculated with equation (50). Specified volumes must be situated on the simulation domain edge/boundary.

THETA Time stepping criterion.

$\theta = 1$ is implicit.
$\theta = 0.50$ is centered in time.

u($NUMINCX$) X-direction component of velocity. Defined at x-faces. **u** is equivalent to $U_{i+1/2,j}$ in equation (34).

UA Wind speed in the x-direction in *(meters/second)*.

UAve($NUMNODES$) The average x-direction velocity for each computational volume.

UDirect($NUMINCX$) The direction for u at each x-face location.

UOld($NUMINCX$) Values for u at the previous time step, N.

ULASTROW The index for the last x-face in the second to last row. $ULASTROW = (NUMROWS - 1) * XINC$

UVM1$(NUMINCX)$ Indexes corresponding to $(i-1/2,j)$ x-face for x-face $(i+1/2,j)$.

UVP1$(NUMINCX)$ Indexes corresponding to $(i+3/2,j)$ x-face for x-face $(i+1/2,j)$.

UXM1$(NUMINCX)$ Indexes corresponding to (i,j) volume for x-face $(i+1/2,j)$.

UXP1$(NUMINCX)$ Indexes corresponding to $(i+1,j)$ volume for x-face $(i+1/2,j)$.

v$(NUMINCX)$ Y-direction component of velocity. Defined at y-faces. **v** is equivalent to $V_{i,j+1/2}$ in equation (35).

VA Wind speed in the y-direction in *(meters/second).*

VAve$(NUMNODES)$ The average y-direction velocity for each computational volume.

VDirect$(NUMINCY)$ The direction for v at each y-face location.

VELDIRCBC Turn on Dirichlet boundary conditions for water velocity, equation (43). This boundary condition allows the specification of a fixed inflow or outflow water velocity. 0 = turn off. 1 = turn on.

VELDXVEL$(\#\ of\ x-face\ specified\ volumes)$ The velocity for each x-boundary face that is specified with **VELDXVOL**. Each index in **VELDXVEL** corresponds to the same index in **VELDXVOL**. Velocity should be given in *meters/second*.

VELDYVEL$(\#\ of\ y-face\ specified\ volumes)$ The velocity for each y-boundary face that is specified with **VELDYVOL**. Each index in **VELDYVEL** corresponds to the same index in **VELDYVOL**. Velocity should be given in *meters/second*.

VELDXVOL$(\#\ of\ x-face\ specified\ volumes)$ The volume indexes for domain boundaries, which are x-faces, to apply Dirichlet velocity boundary conditions. Volume indexes can be calculated with equation (50). Specified volumes must be situated on the simulation domain edge/boundary.

VELDYVOL$(\#\ of\ y-face\ specified\ volumes)$ The volume indexes for domain boundaries, which are y-faces, to apply Dirichlet velocity boundary conditions. Volume indexes can be calculated with equation (50). Specified volumes must be situated on the simulation domain edge/boundary.

VOld$(NUMINCY)$ The value for v at the previous time step, N.

VVM1($NUMINCY$) Indexes corresponding to $(i, j-1/2)$ y-face for y-face $(i, j+1/2)$.

VVP1($NUMINCY$) Indexes corresponding to $(i, j+3/2)$ y-face for y-face $(i, j+1/2)$.

VYM1($NUMINCY$) Indexes corresponding to (i, j) volume for y-face $(i, j+1/2)$.

VYP1($NUMINCY$) Indexes corresponding to $(i+1, j)$ volume for y-face $(i, j+1/2)$.

X1($NUMNODES$) Indexes corresponding to $(i+1/2, j)$ face for volume (i, j).

X3($NUMNODES$) Indexes corresponding to $(i-1/2, j)$ face for volume (i, j).

XM1($NUMNODES$) Indexes corresponding to $(i-1, j)$ volume for volume (i, j).

XP1($NUMNODES$) Indexes corresponding to $(i+1, j)$ volume for volume (i, j).

XINC Increment for obtaining columns of values from a vector format variable, x-face variable. Equal to **NUMCOLS+1**.

XXeI($NUMINCX$) X-coordinates of x-faces.

XYeI($NUMINCX$) Y-coordinates of x-faces.

XINDEX($NUMCOLS+1$) The location of all the volume spacings in the x-direction.

$$XINDEX = 0 : DX : NUMCOLS \times DX$$

YINDEX($NUMROWS+1$) The location of all the volume spacings in the y-direction.

$$YINDEX = 0 : DY : NUMROWS \times DY$$

Y2($NUMNODES$) Indexes corresponding to $(i, j-1/2)$ face for volume (i, j).

Y4($NUMNODES$) Indexes corresponding to $(i, j+1/2)$ face for volume (i, j).

YM1($NUMNODES$) Indexes corresponding to $(i, j-1)$ volume for volume (i, j).

YP1($NUMNODES$) Indexes corresponding to $(i, j+1)$ volume for volume (i, j).

YXeI($NUMINCY$) X-coordinates of y-faces.

YYeI($NUMINCY$) Y-coordinates of y-faces.

ZTopo($NUMNODES$) Topographic elevation in *meters*. **ZTopo** is obtained by reading in the file **Topo.txt**. The elevation is defined at the center of each volume.

ZtX($NUMINCX$) Topographic elevation in *meters* at x-face locations. **ZtX** is obtained either by reading in the file **TopoX.txt** or by linear interpolation from the **ZTOPO** values.

ZtY($NUMINCY$) Topographic elevation in *meters* at y-face locations. **ZtY** is obtained either by reading in the file **TopoY.txt** or by linear interpolation from the **ZTOPO** values.

9 Symbol List

Table 3: Symbol List

Symbol List	
Symbol	Description
A	Calculation variable see equation (32)
a	x-direction Courant Number
b	y-direction Courant Number
C	Calculation variable see equation (48)
Cz	Chezy coefficient
FU	X-direction advective operator, equation (15)
FV	Y-direction advective operator, equation (16)
f	Coriolis f-parameter see equation (17)
G	Calculation variable see equation (33)
g	Gravitational constant
H	Total water depth
h	Undisturbed water depth
M	Number of partial time steps
N	Time index with known values
n	Manning's roughness coefficient
t	Time
Δt	Time step length
u	Velocity in x
$u^k(x,y)$	Bilinearly interpolated U value at (x,y)
U	Depth averaged velocity in x
U_a	Wind velocity in x-direction
v	Velocity in y
$v^{N_k}(x,y)$	Bilinearly interpolated V value at (x,y)
V	Depth averaged velocity in y
V_a	Wind velocity in y-direction
U_{x1}	Velocity at x_1
U_{x2}	Velocity at x_2
V_{y1}	Velocity at y_1
V_{y2}	Velocity at y_2
w	Velocity in z

continued on next page

continued from previous page	
Symbol	Description
x	Direction easting
x_1	Upstream volume boundary in x
x_2	Downstream volume boundary in x
x_e	Particle volume exit coordinate in x
x_p	Particle volume entrance coordinate in x
Δx	Volume length in x
y	Direction northing
y_1	Upstream volume boundary in y
y_2	Downstream volume boundary in y
y_e	Particle volume exit coordinate in y
y_p	Particle volume entrance coordinate in y
Δy	Volume length in y
z	Vertical direction
ε	Horizontal eddy viscosity coefficient
η	Free surface elevation
γ_T	Wind stress coefficient
μ	Horizontal eddy viscosity
ν	Vertical eddy viscosity
Ω	Earth's rotation rate
θ	Time stepping criterion
θ_0	Central latitude
τ	Partial time step
τ^W	Wind shear stress

10 Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. EAR-0207177. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation. This material is also based upon work supported under a Stanford Graduate Fellowship. Additionally, I would like to thank Y.H. Tseng for his boundary condition contributions and R.L. Street for his insights and suggestions on free surface flow modeling.

References

[1] V. I. Agoshkov, A. Quarteroni, and F. Saleri. Recent developments in the numerical simulation of shallow water equations i: Boundary conditions. *Applied Numerical Mathematics*, 15:175–200, 1994.

[2] Rodolfo Bermejo. On the equivalence of semi-lagrangian schemes and particle-in-cell finite element methods. *Monthly Weather Review*, 118:979–987, 1990.

[3] Vicenzo Casulli. Semi-implicit finite difference methods for the two-dimensional shallow water equations. *Journal of Computational Physics*, 86:56–74, 1990.

[4] Vicenzo Casulli. Numerical simulation of three-dimensional free surface flow in isopycnal co-ordinates. *International Journal for Numerical Methods in Fluids*, 25:645–658, 1997.

[5] Vicenzo Casulli. A semi-implicit finite difference method for non-hydrostatic, free-surface flows. *International Journal for Numerical Methods in Fluids*, 30:425–440, 1999.

[6] Vicenzo Casulli and Ralph T. Cheng. Semi-implicit finite difference methods for three-dimensional shallow water flow. *International Journal for Numerical Methods in Fluids*, 15:629–648, 1992.

[7] Clive A. J. Fletcher. *Computational Techniques for Fluid Dynamics: Volume I*, volume I of *Springer Series in Computational Physics*. Springer-Verlag, Berlin, second edition, 1991.

[8] Pijush K. Kundu. *Fluid Mechanics*. Academic Press, San Diego, CA, 1990.

[9] I. Orlanski. A simple boundary condition for unbounded hyperbolic flows. *Journal of Computational Physics*, 21:251–269, 1976.

[10] David W. Pollock. Semianalytical computation of path lines for finite-difference models. *Ground Water*, 26(6):743–750, 1988.

[11] A. Robert. A stable numerical integration scheme for the primitive meteorological equations. *Atmosphere-Ocean*, 19:35–46, 1981.

[12] A. Robert. A semi-lagrangian and semi-implicit numerical integration scheme for the primitive meteorological equations. *Journal of the Meteorological Society of Japan*, 60:319–325, 1982.

[13] Robert L. Street, Gary Z. Watters, and John K. Vennard. *Elementary Fluid Mechanics*. John Wiley & Sons, New York, 7th edition, 1996.

[14] Yu Heng Tseng. *On the Development of a Ghost-cell Immersed Boundary Method and its Application to Large Eddy Simulation and Geophysical Fluid Dynamics*. PhD thesis, Stanford Universitty, 2003.

[15] R. A. Walters and Vincenzo Casulli. A robust, finite element model for hydrostatic surface water flows. *Communications in Numerical Methods in Engineering*, 14:931–940, 1998.

[16] Chunmiao Zheng and Gordon D. Bennett. *Applied contaminant transport modelling: theory and practice*. Van Nostrand Reinhold, New York, 1995.