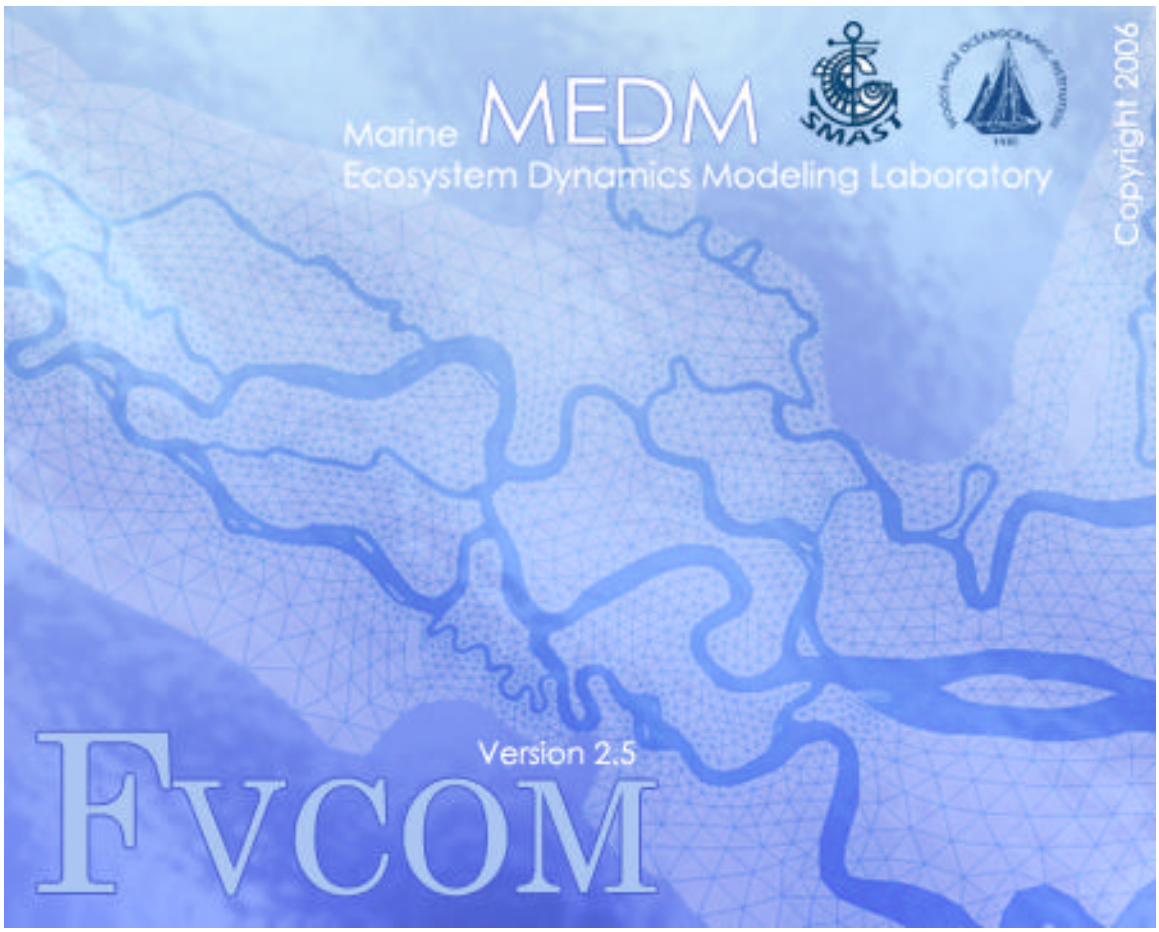
An Unstructured Grid, Finite-Volume Coastal Ocean Model

FVCOM User Manual

Changsheng Chen[1], Robert C. Beardsley[2] and Geoffrey Cowles[1]

[1]Department of Fisheries Oceanography, School for Marine Science and Technology
University of Massachusetts-Dartmouth, New Bedford, MA 02744

[2]Department of Physical Oceanography
Woods Hole Oceanographic Institution, Woods Hole MA 02543



Second Edition

DRAFT JULY 2006

Table of Content

FVCOM Software Users' License Agreement

All users should read this agreement carefully. A user, who receives any version of the source code of FVCOM, must accept all the terms and conditions of this agreement and also agree that this agreement is like any written negotiated agreement signed by you. You may be required to have another written agreement directly with Dr. Changsheng Chen at SMAST/UMASS-D and Dr. Robert C. Beardsley at WHOI

The Finite-Volume Coastal Ocean Model ("FVCOM") source code has been developed in the Marine Ecosystem Dynamics Modeling Laboratory led by Dr. C. Chen at the University of Massachusetts – Dartmouth (UMASS-D) in collaboration with Dr. R. Beardsley at the Woods Hole Oceanographic Institution.

A. Permitted Use and Restrictions on Redistribution

The user agrees that he/she will use the FVCOM source code, and any modifications to the FVCOM source code that the user may create, solely for internal, non-commercial purposes and shall not distribute or transfer the FVCOM source code or modifications to it to any person or third parties not participating in their primary research project without prior written permission from Dr. Chen. The term "non-commercial," as used in this End User License Agreement, means academic or other scholarly research which (a) is not undertaken for profit, or (b) is not intended to produce work, services, or data for commercial use.

B. Mandatory Participation in the FVCOM Community

The user agrees to openly participate in the FVCOM community through three primary mechanisms. These are (*a*) reporting code bugs and problems, (*b*) sharing major modifications made to the code, and (*c*) contributing to an open and ongoing discussion of model deficiencies, needed improvements and additions, and major successes. (Contact Drs. C. Chen, G. Cowles, or R. Beardsley). These mechanisms are intended to benefit the entire FVCOM user community through quick notification of code problems, possible solutions, major code improvements, and, in general, the further development of the FVCOM source code and the associated software tools needed to process, visualize and interpret FVCOM model output.

C. FVCOM Validation

The user agrees to inform Dr. Chen about any FVCOM model validation test case conducted by the user before formal publication of the test case results. This step is intended to minimize potential errors in gridding, model setup, boundary conditions and coding that could contribute to poor FVCOM performance in the validation test case. There is no intent here to exercise any prior restraint on publication.

D. Publication of FVCOM Results

The user agrees to acknowledge FVCOM in any publications resulting from the use of the FVCOM source code. The user agrees to use the name "FVCOM" to refer to the model.

Chapter 1: Introduction

Throughout much of the world oceans, the inner continental shelves and estuaries are characterized by barrier island complexes, inlets, and extensive intertidal salt marshes. Such an irregularly-shaped ocean-land margin system presents a serious challenge for oceanographers involved in model development even though the governing equations of ocean circulation are well defined and numerically solvable in terms of discrete mathematics. Two numerical methods have been widely used in ocean circulation models: (1) the finite-difference method (Blumberg and Mellor, 1987; Blumberg, 1994; Haidvogel et al., 2000) and (2) the finite-element method (Lynch and Naimie, 1993; Naimie, 1996). The finite-difference method is the most basic discrete scheme and has the advantage of computational and coding efficiency. Introducing an orthogonal or non-orthogonal curvilinear horizontal coordinate transformation into a finite-difference model can provide adequate boundary fitting in relatively simple coastal regions but these transformations are incapable of resolving the highly irregular inner shelf/estuarine geometries found in many coastal areas (Blumberg 1994; Chen et al. 2001; Chen et al. 2004a). The greatest advantage of the finite-element method is its geometric flexibility. Triangular grid meshes of arbitrary spatially-dependent size are commonly used in this method, and can provide an accurate fitting of the irregular coastal boundary. The P-type Finite-Element Method (Maday and Patera, 1988) or Discontinuous Galerkin Method (Reed and Hill, 1973; Cockburn *et al*., 1998) has recently been applied to ocean and have shown promise in improving both computational accuracy and efficiency.

We have developed a 3-D unstructured-grid, free-surface, primitive equation, Finite-Volume Coastal Ocean circulation Model (called FVCOM) (Chen et al. 2003a; Chen et al. 2004b). Unlike the differential form used in finite-difference and finite-element models, FVCOM discretizes the integral form of the governing equations. Since these integral equations can be solved numerically by flux calculation (like those used in the finite-difference method) over an arbitrarily-sized triangular mesh (like those used in the finite-element method), the finite-volume approach is better suited to guarantee mass conservation in both the individual control element and the entire computational domain. From a technical point of view, FVCOM combines the best attributes of finite-difference methods for simple discrete coding and computational efficiency and finite-element

methods for geometric flexibility. This model has been successfully applied to study several estuarine and shelf regions that feature complex irregular coastline and topographic geometry, including inter-tidal flooding and drying (see http://codfish.smast.umassd.edu or http://fvcom.smast.umassd.edu for descriptions of these initial applications).

This manual is provided to help users to 1) understand the basic discrete structure and numerical methods used in FVCOM and 2) learn how to use the model for their own applications. Detailed instructions are given for all steps (e.g., grid generation, model input and output, compilation, parallel computation, etc.). Several experiments are included to provide new users with simple examples of model setup and execution

The remaining chapters are organized as follows. Chapter 2: the model formulation; Chapter 3: the finite-volume discrete method; Chapter 4: the extermal forcings; Chapter 5: the open boundary treatments; Chapter 6: the 4-D data assimilation methods; Chapter 7: the sediment module; Chapter 8: the biological modules; Chapter 9: the tracer-tracking model; Chapter 10: the 3-D Lagrangian particle tracking; Chapter 11: the sea ice module, Chapter 12: the code parallelization; Chapter 13: the model coding description and general information; Chapter 14: the model installation; Chapter 15: the model setup; Chapter 16: examples of model applications, and Chapter 17: an example of the unstructured grid generation.

Users should be aware that this manual is only useful for the current version of FVCOM. FVCOM is in continually testing and improvement by a SMAST/UMASSD-WHOI effort led by Changsheng Chen and Robert C. Beardsley. Some very recent modifications may not have been included in this manual. If users find any inconsistency between this manual and the FVCOM code, it is likely to be due to a typo in the manual. Please report any problems with this manual as well as suggestions for improvement, so that future versions can be enhanced.

Chapter 2: The Model Formulation

2.1. The Primitive Equations in Cartesian Coordinates

The governing equations consist of the following momentum, continuity, temperature, salinity, and density equations:

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}-fv=-\frac{1}{\rho_o}\frac{\partial P}{\partial x}+\frac{\partial}{\partial z}(K_m\frac{\partial u}{\partial z})+F_u \tag{2.1}$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}+fu=-\frac{1}{\rho_o}\frac{\partial P}{\partial y}+\frac{\partial}{\partial z}(K_m\frac{\partial v}{\partial z})+F_v \tag{2.2}$$

$$\frac{\partial P}{\partial z}=-\rho g \tag{2.3}$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \tag{2.4}$$

$$\frac{\partial T}{\partial t}+u\frac{\partial T}{\partial x}+v\frac{\partial T}{\partial y}+w\frac{\partial T}{\partial z}=\frac{\partial}{\partial z}(K_h\frac{\partial T}{\partial z})+F_T \tag{2.5}$$

$$\frac{\partial S}{\partial t}+u\frac{\partial S}{\partial x}+v\frac{\partial S}{\partial y}+w\frac{\partial S}{\partial z}=\frac{\partial}{\partial z}(K_h\frac{\partial S}{\partial z})+F_S \tag{2.6}$$

$$\rho = \rho\,(T, S) \tag{2.7}$$

where x, y, and z are the east, north, and vertical axes in the Cartesian coordinate system; u, v, and w are the x, y, z velocity components; T is the temperature; S is the salinity; ρ is the density; P is the pressure; f is the Coriolis parameter; g is the gravitational acceleration; K_m is the vertical eddy viscosity coefficient; and K_h is the thermal vertical eddy

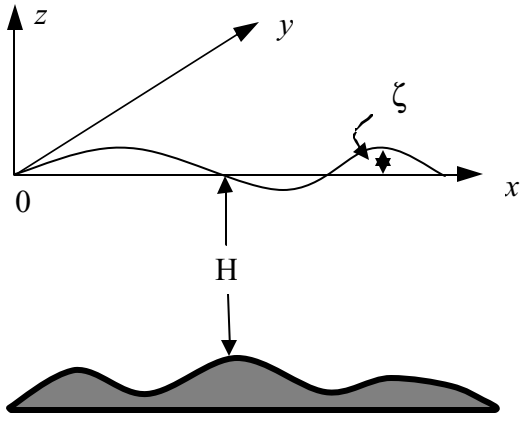


Fig. 2.1: Illustration of the orthogonal coordinate system: x: eastward; y: northward; z: upward.

diffusion coefficient. F_u, F_v, F_T, and F_S represent the horizontal momentum, thermal, and salt diffusion terms. The total water column depth is $D = H + \zeta$, where H is the bottom depth (relative to $z = 0$) and ζ is the height of the free surface (relative to z = 0).

The surface and bottom boundary conditions for temperature are:

$$\frac{\partial T}{\partial z} = \frac{1}{\rho c_p K_h}[Q_n(x,y,t) - SW(x,y,\zeta,t)], \quad \text{at } z = \zeta\,(x,\ y,\ t) \tag{2.8}$$

$$\frac{\partial T}{\partial z} = \frac{A_H \tan\alpha}{K_h}\frac{\partial T}{\partial n}, \quad \text{at } z = -H(x,\ y) \tag{2.9}$$

where $Q_n(x,y,t)$ is the surface net heat flux, which consists of four components: downward shortwave, longwave radiation, sensible, and latent fluxes, $SW(x,y,0,t)$ is the shortwave flux incident at the sea surface, and c_p is the specific heat of seawater. A_H is the horizontal thermal diffusion coefficient, α is the slope of the bottom bathymetry, and n is the horizontal coordinate shown in Figure 2.2 (Pedlosky, 1974; Chen *et al.*, 2004b).

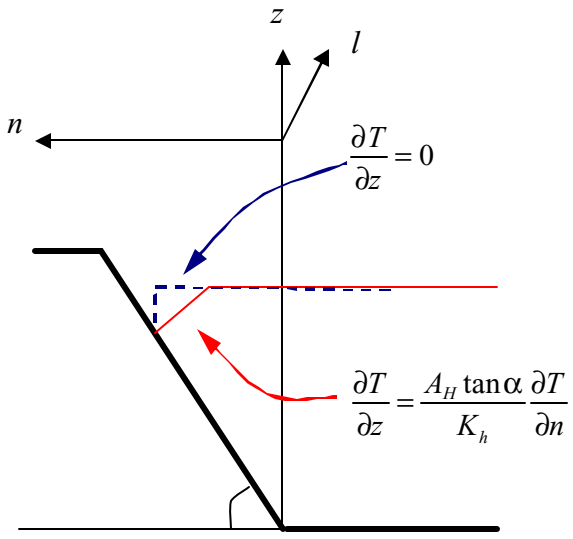


Fig. 2.2: Schematic of the no-flux boundary condition on the bottom slope.

The longwave, sensible and latent heat fluxes are assumed here to occur at the ocean surface, while the downward shortwave flux $SW(x,y,z,t)$ is approximated by:

$$SW(x,y,z,t) = SW(x,y,0,t)[\mathrm{Re}^{\frac{z}{a}} + (1-R)e^{\frac{z}{b}}] \tag{2.10}$$

where a and b are attenuation lengths for longer and shorter (blue-green) wavelength components of the shortwave irradiance, and R is the percent of the total flux associated with the longer wavelength irradiance. This absorption profile, first suggested by Kraus (1972), has been used in numerical studies of upper ocean diurnal heating by Simpson

and Dickey (1981a, b) and others. The absorption of downward irradiance is included in the temperature (heat) equation in the form of

$$\hat{H}(x,y,z,t)=\frac{\partial SW(x,y,z,t)}{\partial z}=\frac{SW(x,y,0,t)}{\rho c_p}[\frac{R}{a}e^{\frac{z}{a}}+\frac{1-R}{b}e^{\frac{z}{b}}] \tag{2.11}$$

This approach leads to a more accurate prediction of near-surface temperature than the flux formulation based on a single wavelength approximation (Chen *et al*., 2003b).

The surface and bottom boundary conditions for salinity are:

$$\frac{\partial S}{\partial z}=-\frac{S(\hat{P}-\hat{E})}{K_h\rho}\cos\gamma, \text{ at } z=\zeta\ (x,\ y,\ t) \tag{2.12}$$

$$\frac{\partial S}{\partial z}=\frac{A_H\tan\alpha}{K_h}\frac{\partial S}{\partial n}, \text{ at } z=-H(x,\ y) \tag{2.13}$$

where $\hat{P}$ and $\hat{E}$ are precipitation and evaporation rates, respectively. $\gamma=1/\sqrt{1+|\nabla\zeta|^2}$. Note that a groundwater flux can be easily added into the model by modifying the bottom boundary conditions for vertical velocity and salinity.

The surface and bottom boundary conditions for *u*, *v*, and *w* are:

$$K_m(\frac{\partial u}{\partial z},\frac{\partial v}{\partial z})=\frac{1}{\rho_o}(\tau_{sx},\tau_{sy}),\ w=\frac{\partial\zeta}{\partial t}+u\frac{\partial\zeta}{\partial x}+v\frac{\partial\zeta}{\partial y}+\frac{E-P}{\rho}, \text{ at } z=\zeta\ (x,\ y,\ t) \tag{2.14}$$

$$K_m(\frac{\partial u}{\partial z},\frac{\partial v}{\partial z})=\frac{1}{\rho_o}(\tau_{bx},\tau_{by}),\ w=-u\frac{\partial H}{\partial x}-v\frac{\partial H}{\partial y}+\frac{Q_b}{\Omega}, \text{ at } z=-H(x,\ y) \tag{2.15}$$

where (t_{sx},t_{sy}) and $(\tau_{bx},\tau_{by})=C_d\sqrt{u^2+v^2}\,(u,v)$ are the *x* and *y* components of surface wind and bottom stresses, Q_b is the groundwater volume flux at the bottom and Ω is the area of the groundwater source. The drag coefficient C_d is determined by matching a logarithmic bottom layer to the model at a height z_{ab} above the bottom, *i.e.*,

$$C_d=\max\left(k^2/\ln(\frac{z_{ab}}{z_o})^2,0.0025\right) \tag{2.16}$$

where k = 0.4 is the von Karman constant and z_o is the bottom roughness parameter.

The kinematic and heat and salt flux conditions on the solid boundary are specified as:

$$v_n=0;\frac{\partial T}{\partial n}=0;\frac{\partial S}{\partial n}=0, \tag{2.17}$$

where v_n is the velocity component normal to the boundary, and n is the coordinate normal to the boundary.

It should be pointed out here that in most popular finite-difference models, the bottom boundary conditions (2.9) and (2.13) for temperature and salinity are simplified as $\partial T / \partial z = \partial S / \partial z = 0$. One reason for this is the difficulty in the finite-difference method of calculating accurately α and $\partial T / \partial z$ or $\partial S / \partial z$ over an irregular bottom slope. The error caused by inaccurate calculation of these two terms in a finite-difference approach might be larger than their real values. This simplification is generally sound for much of the continental shelf in the coastal ocean where the bottom topography is smooth with small slope, but over the shelf break and continental slope where the bottom slope can be quite large, this simplification can destroy the nature of the dynamics of the bottom boundary layer and result in overestimation of vertical mixing and horizontal and vertical velocities. An example for the importance of the exact expression of the no normal flux condition at the bottom given in (2.9) and (2.13) can be seen in Chen *et al.* (2006a). In the finite-volume approach, the bottom slope and gradients of temperature and salinity for an irregular bottom shape can be directly calculated using a simple Green's theorem. Therefore, FVCOM can provide an accurate tracer flux at the bottom using (2.9) and (2.13). This is one of the advantages for using FVCOM in both coastal and deep ocean applications.

2.2. The Governing Equations in the σ-Coordinate

The σ-coordinate transformation is used in the vertical in order to obtain a smooth representation of irregular variable bottom topography. The σ-coordinate transformation is defined as:

$$\sigma = \frac{z - \zeta}{H + \zeta} = \frac{z - \zeta}{D} \tag{2.18}$$

where σ varies from -1 at the bottom to 0 at the surface. In this coordinate, equations (2.1)-(2.9) are given as

$$\frac{\partial \zeta}{\partial t} + \frac{\partial Du}{\partial x} + \frac{\partial Dv}{\partial y} + \frac{\partial \omega}{\partial \sigma} = 0 \tag{2.19}$$

$$\frac{\partial uD}{\partial t}+\frac{\partial u^2D}{\partial x}+\frac{\partial uvD}{\partial y}+\frac{\partial u\omega}{\partial \sigma}-fvD$$
$$=-gD\frac{\partial \zeta}{\partial x}-\frac{gD}{\rho_o}[\frac{\partial}{\partial x}(D\int_{\sigma}^{0}\rho\, d\sigma')+\sigma\rho\frac{\partial D}{\partial x}]+\frac{1}{D}\frac{\partial}{\partial \sigma}(K_m\frac{\partial u}{\partial \sigma})+DF_x \quad (2.20)$$

$$\frac{\partial vD}{\partial t}+\frac{\partial uvD}{\partial x}+\frac{\partial v^2D}{\partial y}+\frac{\partial v\omega}{\partial \sigma}+fuD$$
$$=-gD\frac{\partial \zeta}{\partial y}-\frac{gD}{\rho_o}[\frac{\partial}{\partial y}(D\int_{\sigma}^{0}\rho\, d\sigma')+\sigma\rho\frac{\partial D}{\partial y}]+\frac{1}{D}\frac{\partial}{\partial \sigma}(K_m\frac{\partial v}{\partial \sigma})+DF_y \quad (2.21)$$

$$\frac{\partial TD}{\partial t}+\frac{\partial TuD}{\partial x}+\frac{\partial TvD}{\partial y}+\frac{\partial T\omega}{\partial \sigma}=\frac{1}{D}\frac{\partial}{\partial \sigma}(K_h\frac{\partial T}{\partial \sigma})+D\hat{H}+DF_T \quad (2.22)$$

$$\frac{\partial SD}{\partial t}+\frac{\partial SuD}{\partial x}+\frac{\partial SvD}{\partial y}+\frac{\partial S\omega}{\partial \sigma}=\frac{1}{D}\frac{\partial}{\partial \sigma}(K_h\frac{\partial S}{\partial \sigma})+DF_S \quad (2.23)$$

$$\rho = \rho(T, S) \quad (2.24)$$

In the σ coordinate system, the horizontal diffusion terms are defined as:

$$DF_x \approx \frac{\partial}{\partial x}[2A_mH\frac{\partial u}{\partial x}]+\frac{\partial}{\partial y}[A_mH(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})] \quad (2.25)$$

$$DF_y \approx \frac{\partial}{\partial x}[A_mH(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})]+\frac{\partial}{\partial y}[2A_mH\frac{\partial v}{\partial y}] \quad (2.26)$$

$$D(F_T,F_S,F_{q^2},F_{q^2l})\approx[\frac{\partial}{\partial x}(A_hH\frac{\partial}{\partial x})+\frac{\partial}{\partial y}(A_hH\frac{\partial}{\partial y})](T,S,q^2,q^2l) \quad (2.27)$$

where A_m and A_h are the horizontal eddy and thermal diffusion coefficients, respectively. According to the argument made by Mellor and Blumberg (1985), the simplification made in (2.25)-(2-27) helps to ensure the validity of the locally 1-D bottom boundary layer simulation in the σ-coordinate transformation system. Physically speaking, these simplifications are equivalent to the assumption that horizontal diffusion occurs only parallel to the σ-layers. It is clear that this simplification can lead to additional vertical mixing in the slope region due to the σ transformation, thus making the model-predicted thermoclines too diffusive in the vertical. Questions related to the horizontal diffusion terms and the stability of FVCOM without these terms are being addressed in the FVCOM development and will be improved in a later version.

The boundary conditions are given as follows. At the surface where σ = 0,

$$(\frac{\partial u}{\partial \sigma},\frac{\partial v}{\partial \sigma}) = \frac{D}{\rho_o K_m}(\tau_{sx},\tau_{sy}),\ \omega = \frac{\hat{E}-\hat{P}}{\rho},\ \frac{\partial T}{\partial \sigma} = \frac{D}{\rho c_p K_h}[Q_n(x,y,t) - SW(x,y,0,t)]$$

$$\frac{\partial S}{\partial \sigma} = -\frac{S(\hat{P}-\hat{E})D}{K_h \rho}; \tag{2.28}$$

and at the bottom where σ = -1,

$$(\frac{\partial u}{\partial \sigma},\frac{\partial v}{\partial \sigma}) = \frac{D}{\rho_o K_m}(\tau_{bx},\tau_{by}),\ \omega = \frac{Q_b}{\Omega},\ \frac{\partial T}{\partial \sigma} = \frac{A_H D \tan\alpha}{K_h - A_H \tan^2\alpha}\frac{\partial T}{\partial n}$$

$$\frac{\partial S}{\partial \sigma} = \frac{A_H D \tan\alpha}{K_h - A_H \tan^2\alpha}\frac{\partial S}{\partial n}. \tag{2.29}$$

2.3. The 2-D (Vertically-Integrated) Equations

The sea-surface elevation included in the equations describes the fast moving ($\sqrt{gD}$) long surface gravity waves. In the explicit numerical approach, the criterion for the time step is inversely proportional to the phase speed of these waves. Since the sea-surface elevation is proportional to the gradient of water transport, it can be computed using vertically integrated equations. The 3-D equations then can be solved under conditions with a given sea-surface elevation. In this numerical method, called "mode splitting", the currents are divided into external and internal modes that can be computed using two distinct time steps. This approach has been successfully used in the Princeton Ocean Model (POM) and the Rutgers Ocean Model system (ROMs).

The 2-D (vertically-integrated) momentum and continuity equations are given as:

$$\frac{\partial \zeta}{\partial t} + \frac{\partial(\overline{u}D)}{\partial x} + \frac{\partial(\overline{v}D)}{\partial y} + \frac{\hat{E}-\hat{P}}{\rho} + \frac{Q_b}{\Omega} = 0 \tag{2.30}$$

$$\begin{aligned}&\frac{\partial \overline{u}D}{\partial t} + \frac{\partial \overline{u}^2 D}{\partial x} + \frac{\partial \overline{u}\overline{v}D}{\partial y} - f\overline{v}D \\ &= -gD\frac{\partial \zeta}{\partial x} - \frac{gD}{\rho_o}\{\int_{-1}^{0}\frac{\partial}{\partial x}(D\int_{\sigma}^{0}\rho\, d\sigma')d\sigma + \frac{\partial D}{\partial x}\int_{-1}^{0}\sigma\,\rho\, d\sigma\} + \frac{\tau_{sx}-\tau_{bx}}{\rho_o} + D\tilde{F}_x + G_x\end{aligned} \tag{2.31}$$

$$\begin{aligned}&\frac{\partial \overline{v}D}{\partial t} + \frac{\partial \overline{u}\overline{v}D}{\partial x} + \frac{\partial \overline{v}^2 D}{\partial y} + f\overline{u}D \\ &= -gD\frac{\partial \zeta}{\partial y} - \frac{gD}{\rho_o}\{\int_{-1}^{0}\frac{\partial}{\partial y}(D\int_{\sigma}^{0}\rho\, d\sigma')d\sigma + \frac{\partial D}{\partial y}\int_{-1}^{0}\sigma\,\rho\, d\sigma\} + \frac{\tau_{sy}-\tau_{by}}{\rho_o} + D\tilde{F}_y + G_y\end{aligned} \tag{2.32}$$

where G_x and G_y are defined as

$$G_x = \frac{\partial \overline{u}^2 D}{\partial x} + \frac{\partial \overline{u}\overline{v} D}{\partial y} - D\widetilde{F}_x - [\frac{\partial \overline{u^2} D}{\partial x} + \frac{\partial \overline{uv} D}{\partial y} - D\overline{F}_x] \tag{2.33}$$

$$G_y = \frac{\partial \overline{u}\overline{v} D}{\partial x} + \frac{\partial \overline{v}^2 D}{\partial y} - D\widetilde{F}_y - [\frac{\partial \overline{uv} D}{\partial x} + \frac{\partial \overline{v^2} D}{\partial y} - D\overline{F}_y] \tag{2.34}$$

and the horizontal diffusion terms are approximately given as

$$D\widetilde{F}_x \approx \frac{\partial}{\partial x}[2\overline{A}_m H \frac{\partial \overline{u}}{\partial x}] + \frac{\partial}{\partial y}[\overline{A}_m H(\frac{\partial \overline{u}}{\partial y} + \frac{\partial \overline{v}}{\partial x})] \tag{2.35}$$

$$D\widetilde{F}_y \approx \frac{\partial}{\partial x}[\overline{A}_m H(\frac{\partial \overline{u}}{\partial y} + \frac{\partial \overline{v}}{\partial x})] + \frac{\partial}{\partial y}[2\overline{A}_m H \frac{\partial \overline{v}}{\partial y}] \tag{2.36}$$

$$D\overline{F}_x \approx \frac{\partial}{\partial x}\overline{2A_m H \frac{\partial u}{\partial x}} + \frac{\partial}{\partial y}\overline{A_m H(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})} \tag{2.37}$$

$$D\overline{F}_y \approx \frac{\partial}{\partial x}\overline{A_m H(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})} + \frac{\partial}{\partial y}\overline{2A_m H \frac{\partial v}{\partial y}} \ . \tag{2.38}$$

The overbar "—" denotes the vertically integration. For example, for a given variable ψ ,

$$\overline{\psi} = \int_{-1}^{0} \psi \, d\sigma \ . \tag{2.39}$$

2.4. The Turbulent Closure Models

2.4.1. The Horizontal Diffusion Coefficients. The primitive equations (2.1)-(2.7) are not mathematically closed unless horizontal and vertical diffusion for momentum, temperature and salinity are determined. In FVCOM, the user may choose between using a constant value for the horizontal diffusion coefficient or the Smagorinsky eddy parameterization method (Smagorinsky, 1963). The Smagorinsky horizontal diffusion for the momentum is given as

$$A_m = 0.5C\Omega^u \sqrt{(\frac{\partial u}{\partial x})^2 + 0.5(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial y})^2} \tag{2.40}$$

where C is a constant parameter and Ω^u is the area of the individual momentum control element (see Chapter 3 for definition). It is clear that the value of A_m varies with the model resolution and the gradient of horizontal velocities: decreasing as the grid size or horizontal velocity gradients are reduced.

A similar formula is also used for scalars, which is proportional to the area of the individual tracer control element and the horizontal gradient of the tracer concentration. For water temperature (*T*), for example, it is given as

$$A_h = \frac{0.5C\Omega^{\zeta}}{P_r}\sqrt{(\frac{\partial u}{\partial x})^2 + 0.5(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial y})^2} \qquad (2.41)$$

where Ω^{ζ} is the area of the individual tracer control element (see Chapter 3 for definition) and P_r is the Prandtl number.

2.4.2. The Vertical Eddy Viscosity and Thermal Diffusion Coefficient. FVCOM features a wide choice of ocean turbulence closure models for the parameterization of vertical eddy viscosity (K_m) and vertical thermal diffusion coefficient (K_h). The Mellor and Yamada (1982) level 2.5 (MY-2.5) turbulent closure model is the most popular $q-ql$ type model (where q is the turbulent kinetic energy and l is the turbulent macroscale). FVCOM features an updated version of the MY-2.5 model, which includes a) the upper and lower bound limits of the stability function proposed by Galperin *et al.* (1988); b) the wind-driven surface wave breaking-induced turbulent energy input at the surface and internal wave parameterization by Mellor and Blumberg (2004); and c) the improved parameterization of pressure-strain covariance and shear instability-induced mixing in the strongly stratified region by Kantha and Clayson (1994).

Recently, the General Ocean Turbulent Model (GOTM) has become a very popular open-source community model (Burchard, 2002). The GOTM implements a number of turbulence modules which range from a simple Richardson parameterization to complex Reynolds-stress turbulence closure models. These modules include the MY-2.5 ($q-ql$) and ($k-\varepsilon$) turbulent closure models (where $k = q^2$[1] is the turbulent kinetic energy and ε is the turbulent dissipation). The $k-\varepsilon$ model is an alternative turbulent closure model that is very similar in dynamics to the $q-ql$ turbulent closure model. The most recent version of the $k-\varepsilon$ model also includes a more complete form of the pressure-strain covariance term with buoyancy, anisotropic production and vorticity contributions such that the cutoff of mixing is shifted from $R_i = 0.2$ (original MY-2.5 model) to $R_i = 1.0$

[1] k is in general use in the European ocean modeling community.

(Canuto *et al*., 2001). The GOTM library has been interfaced with FVCOM and the full functionality of GOTM is available to FVCOM users. Brief descriptions of the original MY-2.5 ($q-ql$) and the general form of the $k-\varepsilon$ model now featured in FVCOM are given below. Detailed descriptions of these models can be found in the GOTM manual and references listed in this paragraph.

2.4.2.1. The MY-2.5 Model. In the boundary layer approximation where the shear production of turbulent kinetic energy is produced by the vertical shear of the horizontal flow near the boundary, the equations for q^2 and $q^2 l$ can be simplified as

$$\frac{\partial q^2}{\partial t}+u\frac{\partial q^2}{\partial x}+v\frac{\partial q^2}{\partial y}+w\frac{\partial q^2}{\partial z}=2(P_s+P_b-\varepsilon)+\frac{\partial}{\partial z}(K_q\frac{\partial q^2}{\partial z})+F_q \qquad (2.42)$$

$$\frac{\partial q^2 l}{\partial t}+u\frac{\partial q^2 l}{\partial x}+v\frac{\partial q^2 l}{\partial y}+w\frac{\partial q^2 l}{\partial z}=lE_1(P_s+P_b-\frac{\tilde{W}}{E_1}\varepsilon)+\frac{\partial}{\partial z}(K_q\frac{\partial q^2 l}{\partial z})+F_l \qquad (2.43)$$

where $q^2=(u'^2+v'^2)/2$ is the turbulent kinetic energy; l the turbulent macroscale; K_q is the vertical eddy diffusion coefficient of the turbulent kinetic energy; F_q and F_l represent the horizontal diffusion of the turbulent kinetic energy and macroscale; $P_s=K_m(u_z^2+v_z^2)$ and $P_b=(gK_h\rho_z)/\rho_o$ are the shear and buoyancy production terms of turbulent kinetic energy; $\varepsilon = q^3/B_1 l$ is the turbulent kinetic energy dissipation rate; $W=1+E_2 l^2/(\kappa L)^2$ is a wall proximity function where $L^{-1}=(\eta-z)^{-1}+(H+z)^{-1}$; κ =0.4 is the von Karman constant; H is the mean water depth; and η is the free surface elevation. In general, F_q and F_l are kept as small as possible to reduce the effects of horizontal diffusion on the solutions. In FVCOM, F_q and F_l are parameterized using the Smagorinsky formulation shown above. However, the turbulent closure model can be run with both F_q and F_l set to zero in (2.42 and 2.43).

The turbulent kinetic energy and macroscale equations are closed by defining

$$K_m=lqS_m,\ K_h=lqS_h,\ K_q=0.2lq\ . \qquad (2.44)$$

S_m and S_h are defined as the stability functions

$$S_m=\frac{0.4275-3.354G_h}{(1-34.676G_h)(1-6.127G_h)} \text{ and } S_h=\frac{0.494}{1-34.676G_h} \qquad (2.45)$$

where $G_h = \frac{l^2 g}{q^2 \rho_o} \rho_z$. In the original MY level 2.5 turbulent closure model (Mellor and Yamada, 1974, 1982), S_m and S_h are functions of the gradient Richardson number. By removing a slight inconsistency in the scaling analysis, Galperin *et al*. (1988) simplified the MY turbulent closure model so that S_m and S_h depend only on G_h. G_h has an upper bound of 0.023 for the case of unstable ($\rho_z > 0$) stratification and a lower bound of -0.28 for the case of stable ($\rho_z < 0$) stratification. Parameters A_1, A_2, B_1, B_2, and C_1 are given as 0.92, 16.6, 0.74, 10.1, and 0.08, respectively.

In the original MY level 2.5 model, the surface and bottom boundary conditions for the turbulent kinetic energy and macroscale equations are given as

$$q^2 l = 0,\ q^2 = B_1^{\frac{2}{3}} u_{\tau s}^2 \text{ , at } z = \zeta\ (x, y, t), \tag{2.46}$$

$$q^2 l = 0,\ q^2 = B_1^{\frac{2}{3}} u_{\tau b}^2, \text{ at } z = -H(x, y), \tag{2.47}$$

where $u_{\tau s}$ and $u_{\tau b}$ are the water friction velocities associated with the surface and bottom. Since $q^2 \neq 0$ at the surface and bottom, l equals to zero at the boundaries. Thus K_m, K_h and K_q always remains zero at the surface and bottom. This simplification is reasonable for the bottom but ignores the turbulent energy flux due to surface waves during windy conditions.

Mellor and Blumberg (2004) introduced a new turbulent kinetic energy flux surface boundary condition into the MY level 2.5 model, in which

$$\frac{\partial q^2}{\partial z} = \frac{2\alpha_{CB} u_{\tau s}^3}{K_q};\ l = \max(kz_w, l_z) \text{ at } z = \zeta\ (x, y, t), \tag{2.48}$$

where α_{CB} is a parameter related to the wave age; l_z is the "conventional" empirical length scale; z_w is the wave-related roughness height. According to the "best" fit to field data (Terray *et al.*, 1996, 1997), α_{CB} can be approximated by

$$\alpha_{CB} = 15 \frac{c_p}{u_*} e^{-(0.04 c_p / u_*)^2} \tag{2.49}$$

where c_p is the phase speed of wave at the dominant frequency, u_* is the air friction velocity ($u_* = 30u_\tau$), and c_p / u_* is the "wave age". The value of α_{CB} changes significantly with the wave age: it is given as

$$\alpha_{CB} \cong \begin{cases} 0 & \text{for } c_p/u_* = 0 \quad \text{no waves : original MY 2.5 model} \\ 146 & \text{for } c_p/u_* = 10 \quad \text{younger wa ves} \\ 57 & \text{for } c_p/u_* = 30 \quad \text{mature waves} \end{cases} . \tag{2.50}$$

In general, l_z is proportional to *z*, which can be approximately estimated by

$$l_z = \kappa z \tag{2.51}$$

where κ = 0.4 is the von Karman constant. z_w is also an empirical parameter that is involved in complicated wave dynamics. According to an improved fit to the observational data (Terray *et al.*, 2000; Mellor and Blumberg, 2004), z_w can be determined by

$$z_w = 0.85 H_s \tag{2.52}$$

where H_s is the significant wave height defined as $4H_{rms}$ (H_{rms} is the rms wave height). As suggested by Donelan (1990) and Smith et al. (1992), H_s can be estimated based on the wave age and airside roughness parameter (z_o) in a form of

$$H_s = 2.0\left(\frac{c_p}{u_*}\right)^{2.5} z_o \tag{2.53}$$

Specifying $z_o = \alpha_{CH} u_* / g$ (Charnock's relation), $\alpha_{CH} = 0.45 u_* / c_p$ (Smith *et al.*, 1992; Janssen, 2001) and $u_* = (\rho_w / \rho_a) u_\tau^2$, (2.51) can be rewritten as

$$z_w = \beta \frac{u_\tau^2}{g}; \quad \beta = 665\left(\frac{c_p}{u_*}\right)^{1.5} . \tag{2.54}$$

According to field data, $\beta = 2.0 \times 10^5$ (Stacey, 1999).

2.4.2.2. **The $k-\varepsilon$ Turbulence Model.** In the boundary layer approximation (Rodi, 1980), the $k-\varepsilon$ model can be simplified as

$$\frac{\partial k}{\partial t} - \frac{\partial}{\partial z}\left(\frac{\nu_t}{\hat{\sigma}_k}\frac{\partial k}{\partial z}\right) = P + G - \varepsilon \tag{2.55}$$

$$\frac{\partial \varepsilon}{\partial t} - \frac{\partial}{\partial z}\left(\frac{\nu_t}{\hat{\sigma}_\varepsilon}\frac{\partial \varepsilon}{\partial z}\right) = c_1(P + c_3 G)\frac{\varepsilon}{k} - c_2\frac{\varepsilon^2}{k} \tag{2.56}$$

where ν_t is the eddy viscosity (which is the same as K_q in the MY level 2.5 model), $\hat{\sigma}_k$ is the turbulent Prandtl number that is defined as the ratio of turbulent eddy viscosity to conductivity, P is the turbulent shear production, and G is the turbulent buoyancy production. These two variables have the same definitions as P_s and P_b in the MY level 2.5 model. c_1, c_2, and c_3 are empirical constants. A detailed description of the standard and advanced $k-\varepsilon$ models was given by Burchard and Baumert (1995) and is briefly summarized next.

In the standard $k-\varepsilon$ model,

$$P = -\overline{u'w'}\frac{\partial \bar{u}}{\partial z} - \overline{v'w'}\frac{\partial \bar{v}}{\partial z} = \nu_t\left[\left(\frac{\partial \bar{u}}{\partial z}\right)^2 + \left(\frac{\partial \bar{v}}{\partial z}\right)^2\right] \tag{2.57}$$

$$G = -\frac{g}{\rho_o}\overline{w'\rho'} = -\frac{g}{\rho_o}\left(\frac{\nu_t}{\hat{\sigma}_k}\right)\frac{\partial \bar{\rho}}{\partial z} \tag{2.58}$$

where

$$\hat{\sigma}_k = \begin{cases} \dfrac{[1+(10/3)R_i]^{3/2}}{(1+10R_i)^{1/2}} & R_i \geq 0 \\ 1 & R_i < 0 \end{cases} \tag{2.59}$$

and R_i is the gradient Richardson number defined as

$$R_i = \frac{N_G^2}{N_P^2};\ N_G^2 = -\frac{g}{\rho_o}\frac{\partial \bar{\rho}}{\partial z};\ N_P^2 = \left(\frac{\partial \bar{u}}{\partial z}\right)^2 + \left(\frac{\partial \bar{v}}{\partial z}\right)^2 \tag{2.60}$$

The eddy viscosity ν_t can be estimated by

$$\nu_t = c_\mu\frac{k^2}{\varepsilon} \tag{2.61}$$

where c_μ is a constant. In this standard $k-\varepsilon$ model, the empirical constants are specified as

$$(c_\mu, c_1, c_2, \hat{\sigma}_k, \hat{\sigma}_\varepsilon) = (0.09, 1.44, 1.92, 1.00, 1.30) \tag{2.62}$$

In the advanced $k-\varepsilon$ model, the turbulence model consists of the k and ε equations plus 6 transport equations for the Reynolds stresses ($\overline{u'w'}$, $\overline{v'w'}$ and $\overline{w'^2}$) and the

turbulent heat fluxes ($\overline{u'T'}$, $\overline{v'T'}$ and $\overline{w'T'}$). In this model, the eddy viscosity (ν_t) is still given by (2.59), but c_μ is a function of the vertical shear of the horizontal velocity and vertical stratification. This function corresponds to the stability function S_m in the MY-2.5 model. ν_t and ν_T (thermal diffusion coefficient) are given as

$$\nu_t = c_\mu(\alpha_P, \alpha_{G,} F)\frac{k^2}{\varepsilon}, \quad \nu_t = c'_\mu(\alpha_P, \alpha_{G,} F)\frac{k^2}{\varepsilon} \tag{2.63}$$

where α_P and α_G are functions of dimensionless turbulent shear and turbulent buoyancy numbers in the forms of

$$\alpha_P = \frac{k^2}{\varepsilon^2}N_P^2; \quad \alpha_G = \frac{k^2}{\varepsilon^2}N_G^2 \ . \tag{2.64}$$

F is a near-wall correction factor.

The 8-component advanced turbulence model is mathematically closed with the specification of 11 empirical constants (Burchard and Baumert, 1995).

The surface boundary conditions for k and ε in the k-ε turbulent closure model described above are specified as

$$\begin{aligned} &\nu_t \frac{\partial k}{\partial z} = 0, \qquad \text{if } kc_\mu^{-1/2} > u_{\tau s}^2 \\ &k = u_{\tau s}^2 / c_\mu^{1/2}, \quad \text{otherwise} \\ &\varepsilon = \frac{k^{3/2}c_\mu^{3/4}}{\kappa\{H + z + 0.07H[1 - (u_{\tau s}^2 / kc_\mu^{1/2})]\}} \ . \end{aligned} \tag{2.65}$$

The bottom boundary conditions for k and ε are given as

$$\begin{aligned} &k = u_{\tau b}^2 / c_\mu^{1/2} \\ &\varepsilon = \frac{1}{\kappa(H + z)}u_{\tau b}^3 \end{aligned} \tag{2.66}$$

where κ is the von Karman constant.

The wave-induced turbulent kinetic energy flux at the surface was recently taken into account for the $k-\varepsilon$ model. A detailed description of the modified surface boundary conditions for k and ε is given in Burchard (2001).

2.5. The Primitive Equations in Spherical Coordinates

The FVCOM was originally coded for the local Cartesian coordinate system in which *f* may vary with latitude but the curvature terms due to the spherical shape of the earth were not included in the momentum equations. Therefore, it is suitable for regional applications but not for basin- or global-scale applications. To make FVCOM flexible for either regional or global application, we have built a spherical-coordinate version of FVCOM (Chen *et al.,* 2006b).

Consider a spherical coordinate system in which the x (eastward) and y (northward) axes are expressed as

$$x = r\cos\varphi(\lambda - \lambda_0)\,,\ y = r(\varphi - \varphi_0) \qquad (2.67)$$

where r is the earth's radius; λ is longitude; φ is latitude, and λ_0 and φ_0 are the reference longitude and latitude, respectively. The vertical coordinate z is normal to the earth's surface and positive in the upward direction. This coordinate system is shown in Fig. 2.3.

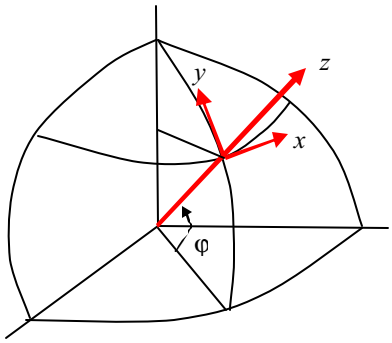


Fig. 2.3: Illustration of the spherical coordinate system.

The three-dimensional (3-D) internal mode flux forms of the governing equations of motion in the spherical and σ coordinates are given as

$$\frac{\partial u}{\partial t} + \frac{1}{r\cos\varphi}\left[\frac{\partial u^2 D}{\partial \lambda} + \frac{\partial uv\cos\varphi}{\partial \varphi}\right] + \frac{\partial u\varpi}{\partial \sigma} + \frac{uvD}{r}\tan\varphi - \frac{wuD}{r} - fvD$$
$$= -\frac{gD}{r\cos\varphi}\frac{\partial \zeta}{\partial \lambda} - -\frac{gD}{\rho_o r\cos\varphi}\left[\frac{\partial}{\partial \lambda}\left(D\int_{\sigma}^{0}\rho\, d\sigma'\right) + \sigma\rho\frac{\partial D}{\partial \lambda}\right] + \frac{1}{D}\frac{\partial}{\partial \sigma}\left(K_m\frac{\partial u}{\partial \sigma}\right) + DF_u \qquad (2.68)$$

$$\frac{\partial v}{\partial t} + \frac{1}{r\cos\varphi}\left[\frac{\partial uvD}{\partial \lambda} + \frac{\partial v^2\cos\varphi}{\partial \varphi}\right] + \frac{\partial v\varpi}{\partial \sigma} - \frac{u^2 D}{r}\tan\varphi - \frac{wvD}{r} + fuD$$
$$= -\frac{gD}{r}\frac{\partial \zeta}{\partial \varphi} - -\frac{gD}{\rho_o r}\left[\frac{\partial}{\partial \varphi}\left(D\int_{\sigma}^{0}\rho\, d\sigma'\right) + \sigma\rho\frac{\partial D}{\partial \varphi}\right] + \frac{1}{D}\frac{\partial}{\partial \sigma}\left(K_m\frac{\partial v}{\partial \sigma}\right) + DF_v \qquad (2.69)$$

$$\frac{\partial \zeta}{\partial t} + \frac{1}{r\cos\varphi}\left[\frac{\partial uD}{\partial \lambda} + \frac{\partial v\cos\varphi D}{\partial \varphi}\right] + \frac{\partial \omega}{\partial \sigma} = 0 \qquad (2.70)$$

$$\frac{\partial TD}{\partial t} + \frac{1}{r\cos\varphi}\left[\frac{\partial TuD}{\partial \lambda} + \frac{\partial Tv\cos\varphi D}{\partial} + \frac{\partial T\omega}{\partial \sigma} = \frac{1}{D}\frac{\partial}{\partial \sigma}\left(K_h\frac{\partial T}{\partial \sigma}\right) + D\hat{H} + DF_T\right. \qquad (2.71)$$

$$\frac{\partial SD}{\partial t}+\frac{1}{r\cos\varphi}[\frac{\partial S\,uD}{\partial\lambda}+\frac{\partial S\,v\cos\varphi D}{\partial}+\frac{\partial S\omega}{\partial\sigma}=\frac{1}{D}\frac{\partial}{\partial\sigma}(K_h\frac{\partial S}{\partial\sigma})+DF_S \tag{2.72}$$

$$\rho = \rho(T, S, P) \tag{2.73}$$

where u, v, and ω are zonal, meridional and σ-coordinate vertical components of the velocity, T is the temperature; S is the salinity; ρ is the total density that is equal to a sum of perturbation density ρ' and reference density ρ_o, P is the pressure; f is the Coriolis parameter; g is the gravitational acceleration; and K_m is the vertical eddy viscosity and K_h the thermal vertical eddy diffusion coefficients that are calculated using one of the above turbulence closure models (Chen *et al.*, 2004). $\hat{H}$ is the vertical gradient of the short-wave radiation. F_u, F_v, F_T, and F_S represent the horizontal momentum, thermal, and salt diffusion terms and the horizontal diffusion is calculated using the Smagorinsky eddy parameterization method (Smagorinsky, 1963). The relationship between ω and the true vertical velocity (w) is given as

$$\omega = w-\frac{u}{r\cos\varphi}(\sigma\frac{\partial D}{\partial\lambda}+\frac{\partial\zeta}{\partial\lambda})-\frac{v}{r}(\sigma\frac{\partial D}{\partial\varphi}+\frac{\partial\zeta}{\partial\varphi})-(\sigma\frac{\partial D}{\partial t}+\frac{\partial\zeta}{\partial t}) \tag{2.74}$$

The 2-D (vertically integrated) momentum and continuity equations are written as

$$\frac{\partial\zeta}{\partial t}+\frac{1}{r\,cos\,\varphi}[\frac{\partial\overline{u}D}{\partial\lambda}+\frac{\partial\overline{v}\,cos\,\varphi D}{\partial\varphi}]+\frac{\hat{E}-\hat{P}}{\rho}+\frac{Q_b}{\Omega}=0 \tag{2.75}$$

$$\begin{aligned}\frac{\partial\overline{u}}{\partial t}+\frac{1}{r\cos\varphi}[\frac{\partial\overline{u}^2D}{\partial\lambda}+\frac{\partial\overline{u}\,\overline{v}\cos\varphi}{\partial\varphi}]+\frac{\overline{u}\,\overline{v}D}{r}\tan\varphi-f\overline{v}D=-\frac{gD}{r\cos\varphi}\frac{\partial\zeta}{\partial\lambda}\\ -\frac{gD}{\rho_o r\cos\varphi}[\int_{-1}^{0}\frac{\partial}{\partial\lambda}(D\int_{\sigma}^{0}\rho\,d\sigma')d\sigma+\frac{\partial D}{\partial\lambda}\int_{-1}^{0}\sigma\rho d\sigma]+\frac{\tau_{s\lambda}-\tau_{b\lambda}}{\rho_o}+D\widetilde{F}_u+G_u\end{aligned} \tag{2.76}$$

$$\begin{aligned}\frac{\partial\overline{v}}{\partial t}+\frac{1}{r\cos\varphi}[\frac{\partial\overline{u}\overline{v}D}{\partial\lambda}+\frac{\partial\overline{v}^2\cos\varphi}{\partial\varphi}]+\frac{\partial\overline{v}\overline{\omega}}{\partial\sigma}-\frac{\overline{u}^2D}{r}\tan\varphi+f\overline{u}D\\ =-\frac{gD}{r}\frac{\partial\zeta}{\partial\varphi}--\frac{gD}{\rho_o r}[\int_{-1}^{0}\frac{\partial}{\partial\varphi}(D\int_{\sigma}^{0}\rho\,d\sigma')d\sigma+\frac{\partial D}{\partial\varphi}\int_{-1}^{0}\sigma\rho d\sigma]+\frac{\tau_{s\varphi}-\tau_{b\varphi}}{\rho_o}+D\widetilde{F}_v+G_v\end{aligned} \tag{2. 77}$$

where G_u and G_v are defined as

$$G_u = \frac{1}{r\cos\varphi}[\frac{\partial \overline{u}^2 D}{\partial\lambda} - \frac{\partial \overline{u^2} D}{\partial\lambda} + \frac{\partial \overline{uv}\cos\varphi D}{\partial\varphi} - \frac{\partial \overline{uv}\cos\varphi D}{\partial\varphi}] + D\overline{F}_u - D\widetilde{F}_u, \quad (2.78)$$

$$G_v = \frac{1}{r\cos\varphi}[\frac{\partial \overline{u}\overline{v} D}{\partial\lambda} - \frac{\partial \overline{uv} D}{\partial\lambda} + \frac{\partial \overline{v}^2 \cos\varphi D}{\partial\varphi} - \frac{\partial \overline{v^2}\cos\varphi D}{\partial\varphi}] + D\overline{F}_v - D\widetilde{F}_v \quad (2.79)$$

and

$$D\widetilde{F}_u \approx \frac{1}{r^2\cos^2\varphi}\frac{\partial}{\partial\lambda}[2\overline{A}_m H\frac{\partial \overline{u}}{\partial\lambda}] + \frac{1}{r}\frac{\partial}{\partial\varphi}[\overline{A}_m H(\frac{\partial \overline{u}}{r\partial\varphi} + \frac{\partial \overline{v}}{r\cos\varphi\partial\lambda})], \quad (2.80)$$

$$D\widetilde{F}_v \approx \frac{\partial}{r\cos\varphi\partial\lambda}[\overline{A}_m H(\frac{\partial \overline{u}}{r\partial\varphi} + \frac{\partial \overline{v}}{r\cos\varphi\partial}) + \frac{\partial}{r^2\partial\varphi}[2\overline{A}_m H\frac{\partial \overline{v}}{\partial\varphi}], \quad (2.81)$$

$$D\overline{F}_u \approx \frac{2}{r^2\cos^2\varphi}\overline{\frac{\partial}{\partial\lambda}A_m H\frac{\partial u}{\partial\lambda}} + \frac{\partial}{r^2\partial\varphi}\overline{A_m H(\frac{\partial u}{\partial\varphi} + \frac{\partial v}{\cos\varphi\partial\lambda})}, \quad (2.82)$$

$$D\overline{F}_v \approx \frac{1}{r^2\cos\varphi}\frac{\partial}{\partial\lambda}\overline{A_m H(\frac{\partial u}{\partial\varphi} + \frac{1}{\cos\varphi}\frac{\partial v}{\partial\lambda})} + \frac{2}{r^2}\frac{\partial}{\partial\varphi}\overline{A_m H\frac{\partial v}{\partial\varphi}}. \quad (2.83)$$

where the definitions of variables are the same as those described in the Cartesian coordinates. The spherical-coordinate version of FVCOM was developed based on the Cartesian coordinate version, in which all the boundary conditions and forcing used in the spherical-coordinate system are the same. The only difference is in the discrete approach, which is described later in chapter 3.

Chapter 3: The Finite-Volume Discrete Method

3.1. Design of the Unstructured Triangular Grids

Similar to a triangular finite element method, the horizontal numerical computational domain is subdivided into a set of non-overlapping unstructured triangular cells. An unstructured triangle is comprised of three nodes, a centroid, and three sides (Fig. 3.1). Let N and M be the total number of centroids and nodes in the computational domain, respectively, then the locations of centroids can be expressed as:

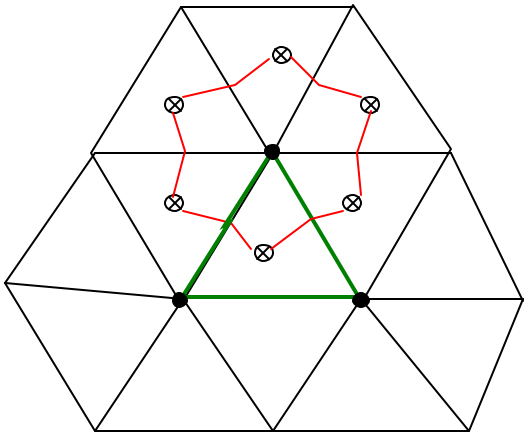


Fig. 3.1: Illustration of the FVCOM unstructured triangular grid. Variable locations: Node •: H, ζ, ω, D, s, θ, q^2, q^2l, A_m, K_h; Centroid ⊗: u, v.

$$[X(i), Y(i)], i = 1 : \mathrm{N}, \quad (3.1)$$

and the locations of nodes can be specified as:

$$[X_n(j), Y_n(j)], j = 1 : \mathrm{M}. \quad (3.2)$$

Since none of the triangles in the grid overlap, N should also be the total number of triangles. On each triangular cell, the three nodes are identified using integral numbers defined as $N_i(\hat{\mathrm{j}})$ where $\hat{\mathrm{j}}$ is counted clockwise from 1 to 3. The surrounding triangles that have a common side are counted using integral numbers defined as $NBE_i(\hat{\mathrm{j}})$ where $\hat{\mathrm{j}}$ is counted clockwise from 1 to 3. At open or coastal solid boundaries, $NBE_i(\hat{\mathrm{j}})$ is specified as zero. At each node, the total number of the surrounding triangles with a

connection to this node is expressed as $NT(\mathrm{j})$, and they are counted using integral numbers $NB_i(m)$ where m is counted clockwise from 1 to $NT(\mathrm{j})$.

To provide a more accurate estimation of the sea-surface elevation, currents and salt and temperature fluxes, u and v are placed at centroids and all scalar variables, such as ζ, H, D, ω, S, T, ρ, K_m, K_h, A_m and A_h are placed at nodes. Scalar variables at each node are determined by a net flux through the sections linked to centroids and the mid-point of the adjacent sides in the surrounding triangles (called the "tracer control element" or TCE), while u and v at the centroids are calculated based on a net flux through the three sides of that triangle (called the "momentum control element" or MCE).

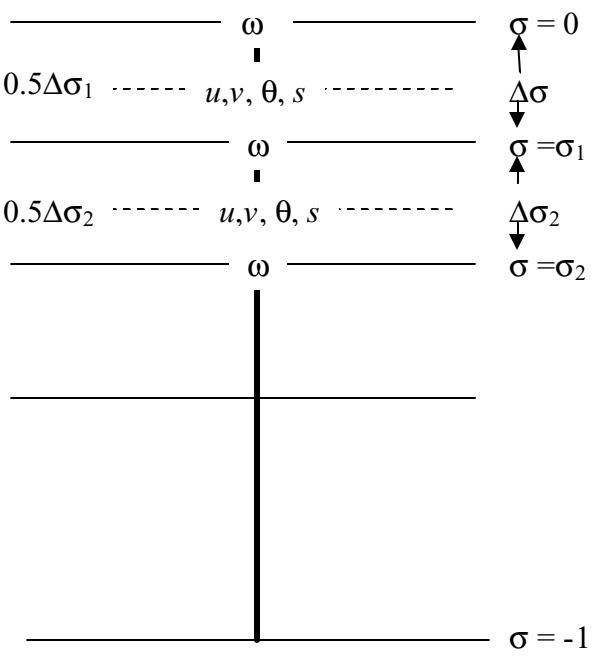


Fig. 3.2: The location of the model variables in the vertical sigma coordinate.

Similar to other finite-difference models such as POM and ROM, all the model variables except ω (vertical velocity on the sigma-layer surface) and turbulence variables (such as q^2 and $q^2 l$) are placed at the mid-level of each σ layer (Fig. 3.2). There are no restrictions on the thickness of the σ-layer, which allows users to use either uniform or non-uniform σ-layers.

3.2. The Discrete Procedure in the Cartesian Coordinates

3.2.1. The 2-D External Mode. Let us consider the continuity equation first. Integrating Eq. (2.30) over a given triangle area yields:

$$\iint \frac{\partial ?}{\partial \mathrm{t}} dxdy = -\iint [\frac{\partial(\overline{u}D)}{\partial x} + \frac{\partial(\overline{v}D)}{\partial x}] dxdy = -\oint_{s'} \overline{v}_n D ds', \tag{3.3}$$

where $\overline{v}_n$ is the velocity component normal to the sides of the triangle and s' is the closed trajectory comprised of the three sides. Eq. (3.3) is integrated numerically using the

modified fourth-order Runge-Kutta time-stepping scheme. This is a multi-stage time-stepping approach with second-order temporal accuracy. The detailed procedure for this method is described as follows:

$$?_j^0 = ?_j^n,\ R_\zeta^0 = R_\zeta^n = \sum_{m=1}^{NT(j)}\left[\left(\Delta x_{2m-1}\overline{v}_m^n - \Delta y_{2m-1}\overline{u}_m^n\right)D_{2m-1}^n + \left(\Delta x_{2m}\overline{v}_m^n - \Delta y_{2m}\overline{u}_m^n\right)D_{2m}^n\right], \tag{3.4}$$

$$?_j^k = ?_j^0 - \alpha^k \frac{\Delta t R_\zeta^{k-1}}{2\Omega_j^\zeta};\ \text{and}\ ?_j^{n+1} = ?_j^4, \tag{3.5}$$

where k =1,2,3,4 and $(\alpha^1,\alpha^2,\alpha^3,\alpha^4)$= (1/4, 1/3,1/2, 1). Superscript n represents the nth time step. Ω_j^ζ is the area enclosed by the lines through centroids and mid-points of the sides of surrounding triangles connected to the node where $?_j$ is located. $\overline{u}_m^n$ and $\overline{v}_m^n$ are defined as:

$$\overline{u}_m^n = \overline{u(NT(m))}^n, \overline{v}_m^n = \overline{v(NT(m))}^n. \tag{3.6}$$

Δt is the time step for the external mode, and

$$\Delta x_{2m-1} = x_{2m} - x_{2m-1}; \Delta x_{2m} = x_{2m+1} - x_{2m}, \tag{3.7}$$

$$\Delta y_{2m-1} = y_{2m} - y_{2m-1}; \Delta y_{2m} = y_{2m+1} - y_{2m}. \tag{3.8}$$

Similarly, integrating Eqs. (2.31) and (2.32) over a given triangular area gives:

$$\begin{aligned}\iint \frac{\partial \overline{u}D}{\partial t}dxdy = &-\oint_{s'}\overline{u}D\overline{v}_n ds' + \iint f\overline{v}Ddxdy - \iint gD\frac{\partial \zeta}{\partial x}dxdy \\ &- \iint\{\frac{gD^2}{\rho_o}\int_{-1}^{0}[\frac{\partial}{\partial x}\int_\sigma^0 \rho d\sigma - \int_\sigma^0 \frac{\partial \rho}{\partial \sigma}\frac{\sigma}{D}\frac{\partial D}{\partial x}d\sigma]d\sigma\}dxdy \\ &+ \iint\frac{\tau_{sx}-\tau_{bx}}{\rho_o}dxdy + \iint D\widetilde{F}_x dxdy + \iint G_x dxdy \end{aligned} \tag{3.9}$$

$$\begin{aligned}\iint \frac{\partial \overline{v}D}{\partial t}dxdy = &-\oint_{s'}\overline{v}D\overline{v}_n ds' - \iint f\overline{u}Ddxdy - \iint gD\frac{\partial \zeta}{\partial y}dxdy \\ &- \iint\{\frac{gD^2}{\rho_o}\int_{-1}^{0}[\frac{\partial}{\partial y}\int_\sigma^0 \rho d\sigma - \int_\sigma^0 \frac{\partial \rho}{\partial \sigma}\frac{\sigma}{D}\frac{\partial D}{\partial y}d\sigma]d\sigma\}dxdy \\ &+ \iint\frac{\tau_{sy}-\tau_{by}}{\rho_o}dxdy + \iint D\widetilde{F}_y dxdy + \iint G_y dxdy. \end{aligned} \tag{3.10}$$

Eqs. (3.9) and (3.10) are also integrated numerically using the modified fourth-order Runge-Kutta time-stepping scheme as follows:

$$\overline{u}_i^0 = \overline{u}_i^n,\ \overline{v}_i^0 = \overline{v}_i^n;\ \overline{R}_u^0 = \overline{R}_u^n,\ \overline{R}_v^0 = \overline{R}_v^n\ , \tag{3.11}$$

$$\overline{u}_i^k = \overline{u}_i^0 - \alpha^k \frac{\Delta t \overline{R}_u^0}{4\Omega_i^u \overline{D}_i},\ \overline{v}_i^k = \overline{v}_i^0 - \alpha^k \frac{\Delta t \overline{R}_v^0}{4\Omega_i^v \overline{D}_i}, \tag{3.12}$$

$$\overline{u}_i^{n+1} = \overline{u}_i^4,\ \overline{v}_i^{n+1} = \overline{v}_i^4 \tag{3.13}$$

where the definitions of k and α^k are the same as those shown in Eqs. (3.4)-(3.5). Ω_i^u and Ω_i^v are the triangular areas where $\overline{u}$ and $\overline{v}$ are located. In the grids used in this model, $\overline{u}$ and $\overline{v}$ are always located at the centroid, so that $\Omega_i^u = \Omega_i^v = \Omega_i$. $\overline{D}_i$ is the depth at the centroid, which is interpolated from depth values at the three surrounding nodes. $\overline{R}_u^n$ and $\overline{R}_v^n$ represent all the terms on the right of Eqs. (3.9) and (3.10), respectively. They are equal to

$$\overline{R}_u^n = ADVU + DPBPX + DPBCX + CORX + VISCX - G_x\ , \tag{3.14}$$

$$\overline{R}_v^n = ADVV + DPBPY + DPBCY + CORY + VISCY - G_y, \tag{3.15}$$

where *ADVU* and *ADVV*, *DPBPX* and *DPBPY*, *DPBCX* and *DPBCY*, *CORX* and *CORY*, *VISCX* and *VISCY* are the x and y components of the vertically integrated horizontal advection, barotropic pressure gradient force, Coriolis force, and horizontal diffusion terms, respectively. The definitions of G_x and G_y are the same as those shown in Eqs. (2.33) and (2.34) in the text.

The x and y components of the horizontal advection are calculated numerically by

$$ADVU = \sum_{m=1}^{3} \left(\overline{u}_{im} \overline{D}_m * \overline{v}_{nm} \hat{l}_m\right),\ ADVV = \sum_{m=1}^{3} \left(\overline{v}_{im} \overline{D}_m * \overline{v}_{nm} \hat{l}_m\right), \tag{3.16}$$

where $\overline{u}_{im}$, $\overline{v}_{im}$, and $\overline{v}_{nm}$ are the x, y and normal components of the velocity on the side m of a triangle cell, and $\overline{v}_{nm}$ is positive when its direction is outward. $\hat{l}_m$ and $\overline{D}_m$ are the length and mid-point water depth of the side m, respectively. They are equal to

$$\overline{D}_m = 0.5[D(N_i(\mathrm{j}_1)) + D(\mathrm{N_i}(\mathrm{j}_2))]\ , \tag{3.17}$$

$$\hat{l}_m = \sqrt{[X_n(N_i(\mathrm{j}_1)) - X_n(N_i(\mathrm{j}_2))]^2 + [Y_n(N_i(\mathrm{j}_1) - Y_n(N_i(\mathrm{j}_2))]^2}\ , \tag{3.18}$$

where

$$\mathrm{j}_2 = m + 1 - \mathrm{INT}(\frac{m+1}{4}) \times 3\ ;\ \mathrm{j}_1 = m + 2 - \mathrm{INT}(\frac{m+2}{4}) \times 3\ . \tag{3.19}$$

The velocity in the triangle cell i is assumed to satisfy the linear distribution given as

$$\overline{u}_i(x',y') = \phi_i^u(x',y') = \overline{u}_{i,0} + a_i^u x' + b_i^u y' \ , \tag{3.20}$$

$$\overline{v}_i(x',y') = \phi_i^v(x',y') = \overline{v}_{i,0} + a_i^v x' + b_i^v y' \ , \tag{3.21}$$

where the parameters a_i^u, b_i^u, a_i^v, and b_i^v are determined by a least-square method based on velocity values at the four cell centered points shown in Fig. 3.3 (one calculated cell

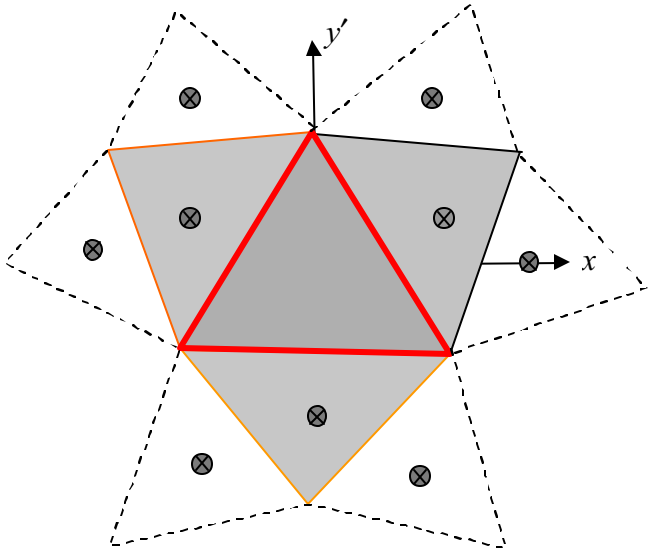


Fig. 3.3: Illustration of the local coordinate used to calculate the velocity and triangular cells used to determine the linear function of the horizontal velocity.

plus three surrounding cells). Then, the normal velocity component on the side m is given as

$$\overline{v}_{nm} = \hat{v}_m \cos\hat{\theta} - \hat{u}_m \sin\hat{\theta} \ , \tag{3.22}$$

where

$$\hat{\theta} = \arctan \frac{Y_n(N_i(j_2)) - Y_n(N_i(j_1))}{X_n(N_i(j_2)) - X_n(N_i(j_1))} \tag{3.33}$$

and

$$\hat{u}_{im} = 0.5[\phi_i^u(\overline{x}_m', \overline{y}_m') + \phi_{\mathrm{NBi}(m)}^u(\overline{x}_m', \overline{y}_m')] \ , \ \hat{v}_{im} = 0.5[\phi_i^v(\overline{x}_m', \overline{y}_m') + \phi_{\mathrm{NBi}(m)}^v(\overline{x}_m', \overline{y}_m')] , \tag{3.34}$$

where $\overline{x}_m'$ and $\overline{y}_m'$ are the mid-point of the side.

The momentum flux through the three sides of triangle cell i is calculated using a second-order accurate scheme (Kobayashi, 1999) as follows:

$$\overline{u}_{im} = \begin{cases} \phi_i^u(0,0), & \overline{v}_{nm} < 0 \\ \phi_{NB_i(m)}^u(x_{im}, y_{im}), & \overline{v}_{nm} \geq 0 \end{cases}, \quad \overline{v}_{im} = \begin{cases} \phi_i^v(0,0), & \overline{v}_{nm} < 0 \\ \phi_{NB_i(m)}^v(x_{im}, y_{im}), & \overline{v}_{nm} \geq 0 \end{cases} \tag{3.35}$$

where x_{im} and y_{im} are the cell-centered point of the surrounding triangle numbered $NB_i(m)$, and (0,0) indicates the location of the cell-centered point.

In the updated code, instead of calculating the normal velocity for each component on the side *m*, we directly calculate the flux through the side *m*, which is equal to

$$\overline{V}_{nm} = \hat{v}_n(X_n(N_i(j_2)) - X_m(N_i(j_1)) - \hat{u}_m(Y_n(N_i(\mathrm{j}_2)) - Y_n(N_i(\mathrm{j}_1)). \tag{3.36}$$

This method significantly improves the numerical accuracy by removing the calculation of the angle $\hat{\theta}$.

The area integration of the barotropic pressure gradient force terms can be converted to a trajectory integration using Stokes' theorem. They can then be calculated numerically by a simple discrete method as follows:

$$DPBPX = g\overline{D}_i \sum_{m=1}^{3} \overline{?}_m [Y_n(N_i(\mathrm{j}_1)) - Y_n(N_i(\mathrm{j}_2))], \tag{3.37}$$

$$DPBPY = g\overline{D}_i \sum_{m=1}^{3} \overline{?}_\mathrm{m} [X_n(N_i(\mathrm{j}_2)) - X_n(N_i(\mathrm{j}_1))], \tag{3.38}$$

where $\overline{?}_\mathrm{m} = 0.5[?(N_i(\mathrm{j}_1) + ?(N_i(\mathrm{j}_2))]$.

A similar approach is used to calculate the baroclinic pressure gradient force terms. These terms are rewritten into the form of the gradient to take the advantage of the flux calculation in the finite-volume method. For example, the x component of the baroclinic pressure gradient force can be rewritten as:

$$\begin{aligned} &-\frac{gD}{?_\mathrm{o}}[\frac{\partial}{\partial x}(D\int_\sigma^0 \rho\, ds') + s\rho\frac{\partial D}{\partial x}] = -\frac{gD}{?_\mathrm{o}}\{\frac{\partial}{\partial x}[D\int_\sigma^0 \rho\, d\mathrm{s} + \mathrm{s}?D] - D\frac{\partial ?\mathrm{s}}{\partial \mathrm{x}}\} \\ &= \frac{gD}{?_\mathrm{o}}\{\frac{\partial}{\partial x}[D\int_\sigma^0 \sigma\frac{\partial \rho}{\partial \sigma} d\mathrm{s}] + D\frac{\partial ?\mathrm{s}}{\partial \mathrm{x}}\} \end{aligned} \tag{3.39}$$

Integrating Eq. (3.39) from –1 to 0 and then integrating over a triangle cell area again, we get

$$DPBCX = \frac{g}{\rho_o}\{\iint [D\frac{\partial}{\partial x}\int_{-1}^{0}(D\int_{\sigma}^{0}\rho\frac{\partial\rho}{\partial\sigma'}d\sigma')d\sigma]dxdy + \iint D^2\frac{\partial}{\partial x}(\int_{-1}^{0}?sds\ )dxdy\}$$
$$= \frac{g}{\rho_o}\{\overline{D}\oint [D\int_{-1}^{0}(\int_{\sigma}^{0}\sigma\frac{\partial\rho}{\partial\sigma'}d\sigma')d\sigma]dy + \overline{D}^2\oint(\int_{-1}^{0}\rho\sigma\, d\sigma)dy\} \quad (3.40)$$

The discrete form of Eq. (3.40) is given as

$$DPBCX = \frac{0.5g}{\rho_0}\{\overline{D}_i\sum_{m=1}^{3}\overline{D}_m[PB_1(i)+PB_2(NB_i(m))][Y_n(N_i(j_1))-Y_n(N_i(j_2))]$$
$$+\overline{D}_i^2\sum_{m=`1}^{3}[PB_2(i)+PB_2(NB_i(m))][Y_n(N_i(j_1))-Y_n(N_i(j_2))]\} \quad (3.41)$$

where

$$PB_1(i) = \sum_{k'=1}^{KB-1}\{[\sigma(k')-\sigma(k'+1)]\sum_{k=1}^{k'}\sigma(k)[\rho(k)-\rho(k+1)]\}, \quad (3.42)$$

$$PB_2(i) = 0.5\sum_{k'=1}^{KB-1}[\rho(k)+\rho(k+1)]\sigma(k)[\sigma(k)-\sigma(k+1)]. \quad (3.43)$$

Similarly, we can derive the y component of the baroclinic pressure gradient force as

$$DPBCY = \frac{0.5g}{\rho_0}\{\overline{D}_i\sum_{m=1}^{3}\overline{D}_m[PB_1(i)+PB_2(NB_i(m))][X_n(N_i(j_2))-X_n(N_i(j_1))]$$
$$+\overline{D}_i^2\sum_{m=`1}^{3}[PB_2(i)+PB_2(NB_i(m))][X_n(N_i(j_2))-X_n(N_i(j_1))]\}. \quad (3.44)$$

The discrete forms of the Coriolis force terms are given as

$$CORX = -fv_iD_i\Omega_i^u;\ CORY = fu_iD_i\Omega_i^v \quad (3.45)$$

The x and y components of the horizontal diffusion can be rewritten as

$$\iint D\tilde{F}_x dxdy \approx \iint\{\frac{\partial}{\partial x}(2\overline{A}_mH\frac{\partial\overline{u}}{\partial x})+\frac{\partial}{\partial y}[\overline{A}_mH(\frac{\partial\overline{u}}{\partial y}+\frac{\partial\overline{v}}{\partial x})]\}dxdy$$
$$= 2\oint\overline{A}_mH\frac{\partial\overline{u}}{\partial x}dy-\oint\overline{A}_mH(\frac{\partial\overline{u}}{\partial y}+\frac{\partial\overline{v}}{\partial x})dx \quad (3.46)$$

and

$$\iint D\tilde{F}_y dxdy \approx \iint\{\frac{\partial}{\partial y}(2\overline{A}_mH\frac{\partial\overline{v}}{\partial y})+\frac{\partial}{\partial x}[\overline{A}_mH(\frac{\partial\overline{u}}{\partial y}+\frac{\partial\overline{v}}{\partial x})]\}dxdy$$
$$= -2\oint\overline{A}_mH\frac{\partial\overline{v}}{\partial y}dx+\oint\overline{A}_mH(\frac{\partial\overline{u}}{\partial y}+\frac{\partial\overline{v}}{\partial x})dy \quad (3.47)$$

The discrete forms of Eqs. (3.46) and (3.47) are given as

$$VISCX = \sum_{m=1}^{3}\{0.5\overline{H}_m[\overline{A}_m(i)+\overline{A}_m(NB(m))][a^u(i)+a^u(NB(m))] \\ [Y_n(N_i(\mathrm{j}_1))-Y_n(N_i(\mathrm{j}_2))]+0.25\overline{H}_m[\overline{A}_m(i)+\overline{A}_m(NB(m))] \\ [[b^u(i)+b^u(NB(m))+a^v(i)+a^v(NB(m))][X_n(N_i(\mathrm{j}_2))-X_n(N_i(\mathrm{j}_1))]\} \tag{3.48}$$

where $\overline{H}_m = 0.5[H(N_i(\mathrm{j}_1)+H(N_i(\mathrm{j}_2))]$, and

$$VISCY = \sum_{m=1}^{3}\{0.5\overline{H}_m[\overline{A}_m(i)+\overline{A}_m(NB(m))][b^v(i)+b^v(NB(m))] \\ [X_n(N_i(\mathrm{j}_2))-X_n(N_i(\mathrm{j}_1))]+0.25\overline{H}_m[\overline{A}_m(i)+\overline{A}_m(NB(m))] \\ [[b^u(i)+b^u(NB(m))+a^v(i)+a^v(NB(m))][Y_n(N_i(\mathrm{j}_1))-Y_n(N_i(\mathrm{j}_2))]\}. \tag{3.49}$$

The G_x and G_y terms are given as

$$G_x = ADVU + VICX - \overline{ADVU} - \overline{VISCX}\ , \tag{3.50}$$

$$G_y = ADVV + VICY - \overline{ADVV} - \overline{VISCY}\ , \tag{3.51}$$

where

$$\overline{ADVU} = \iint[\frac{\partial\overline{u^2}D}{\partial x}+\frac{\partial\overline{uv}D}{\partial y}]dxdy = \oint\overline{u^2}Ddy+\oint\overline{uv}Ddx \\ = \sum_{m=1}^{3}0.5\{[\overline{u^2(i)}+\overline{u^2(NB(m))}]\overline{D}_m[Y_n(N_i(\mathrm{j}_1))-Y_n(N_i(\mathrm{j}_2))] \\ +[\overline{u(i)v(i)}+\overline{u(NB(m))v(NB(m))}]\overline{D}_m[X_n(N_i(\mathrm{j}_2))-X_n(N_i(\mathrm{j}_1))]\}; \tag{3.52}$$

$$\overline{ADVV} = -\iint[\frac{\partial\overline{uv}D}{\partial x}+\frac{\partial\overline{v^2}D}{\partial y}]dxdy = -\oint\overline{uv}Ddy-\oint\overline{v^2}Ddx \\ = \sum_{m=1}^{3}0.5\{[\overline{u(i)v(i)}+\overline{u(NB(m))v(NB(m))}]\overline{D}_m[Y_n(N_i(\mathrm{j}_1))-Y_n(N_i(\mathrm{j}_2))] \\ +[\overline{v^2(i)}+\overline{v^2(NB(m))}]\overline{D}_m[X_n(N_i(\mathrm{j}_2))-X_n(N_i(\mathrm{j}_1))]\}; \tag{3.53}$$

$$\overline{VISCX} = \iint D\overline{F}_x dxdy \approx \iint[\frac{\partial}{\partial x}\overline{2A_mH\frac{\partial u}{\partial x}}+\frac{\partial}{\partial y}\overline{A_mH(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})}]dxdy \\ = \oint\overline{(2A_mH\frac{\partial u}{\partial x})}dy-\oint\overline{[A_mH(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})]}dx \\ = 2\oint\overline{H(A_m\frac{\partial u}{\partial x})}dy-\oint\overline{H[A_m(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})]}dx; \tag{3.54}$$

and

$$\overline{VISCY} = \iint D\overline{F}_y dxdy \approx \iint [\frac{\partial}{\partial y}\overline{2A_m H\frac{\partial v}{\partial y}} + \frac{\partial}{\partial x}\overline{A_m H(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})}]dxdy$$

$$= -\oint \overline{(2A_m H\frac{\partial v}{\partial y})}dx + \oint [\overline{A_m H(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})}]dy \qquad (3.55)$$

$$= -2\oint \overline{H(A_m\frac{\partial v}{\partial y})}dx + \oint \overline{H[A_m(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})]}dy.$$

Let us define

$$USH = \overline{A_m\frac{\partial u}{\partial x}},\ UVSH = \overline{A_m(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})},\ \text{and}\ VSH = \overline{A_m\frac{\partial v}{\partial y}}, \qquad (3.56)$$

where u and v are the x and y components of the velocity output from the 3-D model. At each σ level in a triangle cell, they can be expressed as a linear function as

$$u_{i,k}(x',y') = u_{i,k}(0,0) + a^u_{(i,k)}x' + b^u_{(i,k)}y',\quad v_{i,k}(x',y') = v_{i,k}(0,0) + a^v_{(i,k)}x' + b^v_{(i,k)}y'. \qquad (3.57)$$

Then at the triangle cell i, we have

$$USH(i) = \overline{A_m\frac{\partial u}{\partial x}} = \sum_{k=1}^{KB-1} A_m(k)a^u_{(i,k)}, \qquad (3.58)$$

$$VSH(i) = \overline{A_m\frac{\partial v}{\partial y}} = \sum_{k=1}^{KB-1} A_m(k)b^v_{(i,k)}. \qquad (3.59)$$

$$UVSH(i) = \overline{A_m(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})} = \sum_{k=1}^{KB-1} A_m(k)[a^u_{(i,k)} + b^v_{(i,k)}]\ . \qquad (3.60)$$

Therefore,

$$\overline{VISCX} = 2\oint \overline{H(A_m\frac{\partial u}{\partial x})}dy - \oint \overline{H[A_m(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})]}dx$$

$$= \sum_{m=1}^{3}\overline{H}_m[USH(i)+USH(NB(m))][Y_n(N_i(\mathrm{j}_1)) - Y_n(N_i(\mathrm{j}_2))] \qquad (3.61)$$

$$+ 0.5\sum_{m=1}^{3}\overline{H}_m[UVSH(i)+UVSH(NB(m))][X_n(N_i(\mathrm{j}_2)) - X_n(N_i(\mathrm{j}_1))];$$

$$\overline{VISCY} = -2\oint H\overline{(A_m \frac{\partial v}{\partial y})}\,dx + \oint H\overline{[A_m(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})]}\,dy$$
$$= \sum_{m=1}^{3} \overline{H}_m [VSH(i) + VSH(NB(m))][X_n(N_i(j_2)) - X_n(N_i(\mathrm{j}_1))] \quad (3.62)$$
$$+ 0.5\sum_{m=1}^{3} \overline{H}_m [UVSH(i) + UVSH(NB(m))][Y_n(N_i(\mathrm{j}_1)) - Y_n(N_i(\mathrm{j}_2))].$$

3.2.2. The 3-D Internal Mode. The momentum equations are solved numerically using a simple combined explicit and implicit scheme in which the local change of the currents is integrated using a first-order accurate forward Euler scheme. The advection terms are computed explicitly by a second-order accuracy upwind scheme, and the vertical diffusion is solved implicitly. A second-order accurate Runge-Kutta time stepping scheme is also incorporated in the updated version to increase the numerical integration to second-order accuracy (Kobayashi et al., 1999) (Note: The original FVCOM used the RK time stepping for the internal mode integration. When it was upgraded to the Fortran 95/99 version, this method was replaced by a simple Euler forward scheme for the advective terms. The RK method will be put back into FVCOM in the next release). The procedure for this method is very similar to that described above for the 2-D external mode. To provide a simple interpretation of the numerical approach for the 3-D internal mode, we focus our description here only on the first-order accuracy upwind scheme. It should be noted here that the second-order accuracy Runge-Kutta time stepping scheme is also incorporated in the model.

The 3-D momentum equations can be rewritten as:

$$\frac{\partial uD}{\partial t} + R_u = \frac{1}{D}\frac{\partial}{\partial \sigma}(K_m \frac{\partial u}{\partial \sigma}), \frac{\partial vD}{\partial t} + R_v = \frac{1}{D}\frac{\partial}{\partial \sigma}(K_m \frac{\partial v}{\partial \sigma}), \quad (3.63)$$

where

$$R_u = ADVU3 + CORX3 + DPBPX3 + BPBCX3 + HVISCX\ , \quad (3.64)$$

$$R_v = ADVV3 + CORY3 + DPBPY3 + BPBCY3 + HVISCY\ . \quad (3.65)$$

The numerical integration is conducted in two steps. In the first step, the "transition" velocity is calculated using all the terms except the vertical diffusion term in the

momentum equations. Then the true velocity is determined implicitly using a balance between the local change of the "transition" velocity and the vertical diffusion term.

Let $u^*_{i,k}$ and $v^*_{i,k}$ be the x and y components of the "transition" velocity at the mid-point between the k and k+1 σ -levels in triangular cell i. They can be determined numerically as follows:

$$u^*_{i,k} = u^n_{i,k} - \frac{\Delta t_I}{\Omega_i \Delta\sigma \overline{D}_i} R^n_{u,(i,k)}, \quad v^*_{i,k} = v^n_{i,k} - \frac{\Delta t_I}{\Omega_i \Delta\sigma \overline{D}_i} R^n_{v,(i,k)}, \tag{3.66}$$

where $\Delta\sigma = \sigma_k - \sigma_{k+1}$, and Δt_I is the time step for the internal mode. Each term in $R^n_{u,(i,k)}$ and $R^n_{v,(i,k)}$ is computed as follows:

$$\begin{aligned} ADVU3^n_{(i,k)} &= \iint [\int_{\sigma_{k+1}}^{\sigma_k} (\frac{\partial u^2 D}{\partial x} + \frac{\partial uvD}{\partial y} + \frac{\partial u\omega}{\partial \sigma}) d\sigma] dxdy \\ &= (\sigma_k - \sigma_{k+1}) \sum_{m=1}^{3} u^n_{i,k}(m) \overline{D}_m v^n_{n,k}(m) \hat{l}_m + \Omega_i [(u^n_{i,k-1} + u^n_{i,k}) \omega^n_{i,k} - (u^n_{i,k} + u^n_{i,k+1}) \omega^n_{i,k+1}]; \end{aligned} \tag{3.67}$$

$$\begin{aligned} ADVV3^n_{(i,k)} &= \iint [\int_{\sigma_{k+1}}^{\sigma_k} (\frac{\partial uvD}{\partial x} + \frac{\partial v^2 D}{\partial y} + \frac{\partial v\omega}{\partial \sigma}) d\sigma] dxdy \\ &= (\sigma_k - \sigma_{k+1}) \sum_{m=1}^{3} v^n_{i,k}(m) \overline{D}_m v^n_{n,k}(m) \hat{l}_m + \Omega_i [(v^n_{i,k-1} + v^n_{i,k}) \omega^n_{i,k} - (v^n_{i,k} + v^n_{i,k+1}) \omega^n_{i,k+1}]; \end{aligned} \tag{3.68}$$

$$CORX3 = -fv_i \overline{D}_i (\sigma_k - \sigma_{k+1}) \Omega_i, \; CORY3 = fu_i \overline{D}_i (\sigma_k - \sigma_{k+1}) \Omega_i; \tag{3.69}$$

$$\begin{aligned} HVISCX^n_{(i,k)} &= \iint (\int_{\sigma}^{0} DF_x d\sigma) dxdy \approx \iint \{\int_{\sigma}^{0} \frac{\partial}{\partial x} [2A_m H \frac{\partial u}{\partial x}] + \frac{\partial}{\partial y} [A_m H (\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})] d\sigma\} dxdy \\ &= [2 \oint A_m H \frac{\partial u}{\partial x} dy - \oint \overline{A}_m H (\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) dx] (\sigma_k - \sigma_{k+1}) \\ &= \{\sum_{m=1}^{3} 0.5 \overline{H}_m [A_m(i) + A_m(NB(m))] (a^u_{(i,k)} + a^u_{(NB(m),k)}) [Y_n(N_i(j_1)) - Y_n(N_i(j_2))] \\ &+ \sum_{m=1}^{3} 0.25 \overline{H}_m (b^u_{(i,k)} + b^u_{(NB(m),k)} + a^v_{(i,k)} + a^v_{(NB(m),k)}] \\ &[X_n(N_i(j_2)) - X_n(N_i(j_1))] [A_m(i) + A_m(NB(m))]\} (\sigma_k - \sigma_{k+1}); \end{aligned} \tag{3.70}$$

$$HVISCY^{n}_{(i,k)} = \iint (\int_{\sigma}^{0} DF_y d\sigma) dxdy \approx \iint \{\int_{\sigma}^{0} \frac{\partial}{\partial y}[2A_m H \frac{\partial v}{\partial y}] + \frac{\partial}{\partial x}[A_m H(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})] d\sigma\} dxdy$$

$$= [-2\oint A_m H \frac{\partial v}{\partial y} dx + \oint \overline{A}_m H(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) dy](\sigma_k - \sigma_{k+1})$$

$$= \{\sum_{m=1}^{3} 0.5\overline{H}_m [A_m(i) + A_m(NB(m))](b^{v}_{(i,k)} + b^{v}_{(NB(m),k)}][X_n(N_i(\mathrm{j}_2) - X_n(N_i(\mathrm{j}_1))] \quad (3.71)$$

$$+ \sum_{m=1}^{3} 0.25\overline{H}_m (b^{u}_{(i,k)} + b^{u}_{(NB(m),k)} + a^{v}_{(i,k)} + a^{v}_{(NB(m),k)}]$$

$$[Y_n(N_i(\mathrm{j}_1)) - Y_n(N_i(\mathrm{j}_2))][A_m(i) + A_m(NB(m))]\}(\sigma_k - \sigma_{k+1});$$

$$DPBPX^{n}_{(i,k)} = g\overline{D}_i(\sigma_k - \sigma_{k+1}) \sum_{m=1}^{3} \overline{?}^{n}_{m}[Y_n(N_{\mathrm{i}}(\mathrm{j}_1)) - Y_n(N_i(\mathrm{j}_2))]; \quad (3.72)$$

$$DPBPY^{n}_{(i,k)} = g\overline{D}_i(\sigma_k - \sigma_{k+1}) \sum_{m=1}^{3} \overline{?}^{\mathrm{n}}_{\mathrm{m}}[X_n(N_i(\mathrm{j}_2)) - X_n(N_i(\mathrm{j}_1))]; \quad (3.73)$$

$$PBCX3 = -\frac{g}{\rho_o}\{\iint_{\sigma_{k+1}}^{\sigma_k} D\frac{\partial}{\partial x}[D(\int_{\sigma}^{0} \sigma \frac{\partial \rho}{\partial \sigma} d\sigma)] d\sigma dxdy + \iint_{\sigma_{k+1}}^{\sigma_k} D^2 \frac{\partial \rho \sigma}{\partial x} d\sigma\, dxdy\}$$

$$= -\frac{g}{\rho_o}\{\overline{D}_i \oint (D\int_{\sigma}^{0} \sigma \frac{\partial \rho}{\partial \sigma} d\sigma) dy + \overline{D_i^2} \oint \rho\sigma\, dy\}[\sigma_k - \sigma_{k+1}] \quad (3.74)$$

Let

$$PBC(i) = \int_{\sigma}^{0} \sigma \frac{\partial \rho}{\partial \sigma} d\sigma = \sum_{k'=1}^{k} \sigma(k')[\rho(k') - \rho(k'+1)], \quad (3.75)$$

then

$$DPBCX = -\frac{0.5g}{\rho_o}(\sigma_k - \sigma_{k+1})$$

$$\{\overline{D}_i \sum_{m=1}^{3} \overline{D}_m [PBC(i) + PBC(NB_i(m))][Y_n(N_i(\mathrm{j}_1)) - Y_n(N_i(\mathrm{j}_2))] \quad (3.76)$$

$$+ \overline{D_i^2} \sum_{m=1}^{3} [\rho(i) + \rho(NB_i(m))]\sigma(k)[Y_n(N_i(\mathrm{j}_1)) - Y_n(N_i(\mathrm{j}_2))]\};$$

Similarly, we can derive the y-component of the baroclinic pressure gradient as

$$DPBCY = -\frac{0.5g}{\rho_o}(\sigma_k - \sigma_{k+1})$$

$$\{\overline{D}_i \sum_{m=1}^{3} \overline{D}_m [PBC(i) + PBC(NB_i(m))][X_n(N_i(j_2)) - X_n(N_i(j_1))] \quad (3.77)$$

$$+ \overline{D_i^2} \sum_{m=1}^{3} [\rho(i) + \rho(NB_i(m))]\sigma(k)[X_n(N_i(j_2)) - X_n(N_i(j_1))]\}.$$

The mathematic forms of the two equations in (3.63) are the same, so that they can be solved numerically using the same approach. The method used to numerically solve these equations was adopted directly from ECOM-si and POM (Blumberg 1994). For example, a detailed description of this method is given below for the *u*-component of the momentum equation. The implicit discrete form of the first equation in (3.63) is given as

$$A_{i,k}u_{k+1}^{n+1} + B_{i,k}u_k^{n+1} + C_{i,k}u_{k-1}^{n+1} = u^* \quad (3.78)$$

where

$$A_{i,k} = -\frac{2K_m(k+1)\Delta t}{[D^{n+1}]^2(\sigma_k - \sigma_{k+1})(\sigma_k - \sigma_{k+2})};$$

$$C_{i,k} = -\frac{2K_m(k)\Delta t}{[D^{n+1}]^2(\sigma_k - \sigma_{k+1})(\sigma_{k-1} - \sigma_{k+1})}; \quad (3.79)$$

$$B_{i,k} = 1 - A_{i,k} - C_{i,k}.$$

This is a tri-diagonal equation and it ranges from $k = 2$ to KB-2, where KB is the number of total σ-levels in the vertical. The solution for $u(k)$ is calculated by

$$u(k) = -\frac{A_{i,k}}{B_{i,k} + C_{i,k}VH(k-1)}u(k+1) + \frac{u^* - C_{i,k}VHP(k-1)}{B_{i,k} + C_{i,k}VH(k-1)} \quad (3.80)$$

where

$$VH(k) = -\frac{A_{i,k}}{B_{i,k} + C_{i,k}VH(k-1)}; \; VHP(k) = \frac{u_{i,k}^* - C_{i,k}VHP(k-1)}{B_{i,k} + C_{i,k}VH(k-1)}. \quad (3.81)$$

The equation for temperature or salinity as well as other passive tracers also can be rewritten as the form shown in (3.78), so they can be solved numerically using the exact same approach discussed above. The only difference is that T is calculated at nodes and has the same control volume as that used for ζ. To apply a second-order upwind scheme for the temperature advection term, we used Green's theorem to calculate the temperature

gradient at nodes (Barth, 1993; Wu and Bogy, 2000). Computing T at nodes has shown a significant improvement in the advective temperature flux over steep bottom topography.

In the original version of FVCOM, the horizontal advection terms in a tracer equation are calculated using the second-order accurate upwind scheme shown in Fig. 3.4. Let ψ be an arbitrary tracer variable (which can be T, S or q or others), the value of ψ at the centroid of a triangle can be determined by

$$\psi = \psi_i + \frac{\partial \psi}{\partial \lambda}\Delta\lambda + \frac{\partial \psi}{\partial \varphi}\Delta\psi$$
$$\frac{\partial \psi}{\partial \lambda} = \frac{r}{\Omega_{\psi_i}}\oint \psi d\varphi; \frac{\partial \psi}{\partial \varphi} = \frac{r}{\Omega_{\psi_i}}\oint [\psi(\cos\varphi)_{\psi_i}]d\lambda \qquad (3.82)$$

where subscript "i" represents the index of the node over TCE and Ω_{ψ_i} is the area of the

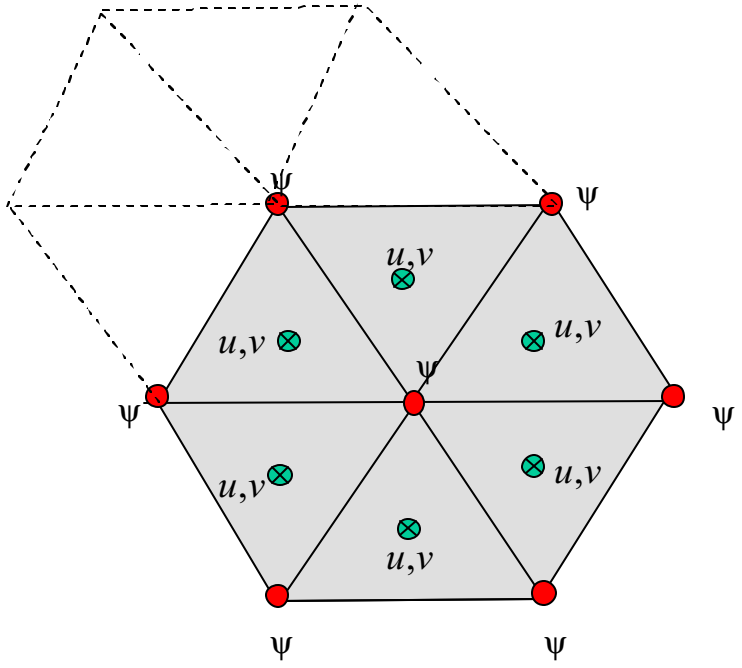


Fig. 3.4: Illustration of the second-order upwind scheme used to calculate the horizontal advection terms in a tracer equation.

larger TCE with a center node at ψ_i. In the upwind scheme, ψ at the centroid is always calculated using the flux determined with velocity and distribution of ψ in the upwind side of the control volume.

In the recent applications to submarine banks and islands where the bottom topography is steep, we found that the central difference scheme used in the vertical advection term can cause odd-even decoupling due to the unstable nature of the scheme. To retaining the second-order accuracy upwind scheme, we introduced a second-order accurate Multidimensional Positive Definite Advection Transport Algorithm (MPDATA) with Flux Corrected Transport (FCT) as an option for the calculation of the vertical advection term (Smolarkiewicz 1984; Smolarkiewicz and Grabowski 1990). This work was done by Hu Song, a graduate student at MEDM, who applied FVCOM to study the tidal pumping process on the steep flanks of Georges Bank This approach provides an optimal control of an artificial numerical error due to over-or-under-shooting of the tracer value over steep bottom topography. FCT ensures a second-order accurate positive definite tracer calculation in a 3-D volume away from discontinuties. A brief description of the algorithm is given here.

The 3-D advection equations can be written as

$$\frac{\partial \psi}{\partial t} + HADV + VADV = 0 \tag{3.83}$$

where *HADV* and *VADV* are the horizontal and vertical advection terms in the tracer equation, respectively. Define $\psi_{i,k}^{n+1}$ as the value of ψ at the *i*th node and *k*th vertical level at the time step *n*+1. Then $\psi_{i,k}^{n+1}$ can be calculated following two steps.

Step 1:

$$\psi_{i,k}^{(*)^0} = \psi_{i,k}^{n} - \min(1, \beta_{i,k})\Psi_{i,k}^{n}, \tag{3.84}$$

where

$$\Psi_{i,k}^{n} = (1-\alpha_i)\psi_{i,k}^{n} + \alpha_i \Delta t(HADV + VDAV) \ , \tag{3.85}$$

$$\beta_{i,k} = \begin{cases} [\dfrac{\psi_{i,k}^{n} - \psi_{i,k}^{\min}}{\Psi_{i,k}^{n}}]^{+} - [\dfrac{\psi_{i,k}^{\max} - \psi_{i,k}^{n}}{\Psi_{i,k}^{n}}]^{-} & \text{as } \Psi_{i,k}^{n} \neq 0 \\ 0 & \text{as } \Psi_{i,k}^{n} = 0 \end{cases}, \tag{3.86}$$

$$F_v = -\frac{1}{2\Delta\sigma}\{[\omega_{i,k}^n]^-\psi_{i,k-1}^n + [\omega_{i,k}^n]^+\psi_{i,k}^n - [\omega_{i,k+1}^n]^-\psi_{i,k}^n - [\omega_{i,k+1}^n]^+\psi_{i,k+1}^n\}. \quad (3.87)$$

Step 2:

$$\psi_{i,k}^{(*)m} = \psi_{i,k}^{(*)m-1} - \frac{\Delta t}{2\Delta\sigma}\left\{[\omega_{i,k}^{(*)m}]^{mon^-}\psi_{i,k-1}^{(*)m-1} + [\omega_{i,k}^{(*)m}]^{mon^+}\psi_{i,k}^{(*)m-1} - [\omega_{i,k+1}^{(*)m}]^{mon^-}\psi_{i,k}^{(*)m-1} - [\omega_{i,k+1}^{(*)m}]^{mon^+}\psi_{i,k+1}^{(*)m-1}\right\} \quad (3.88)$$

$$\psi_{i,k}^{n+1} = \psi_{i,k}^{(*)IORD}, \quad (3.89)$$

where m is the iteration number from 1 to IORD. The default value of IORD is 4. For any arbitrary function Φ, $[\Phi]^- = \min(0, \Phi)$ and $[\Phi]^+ = \max(0, \Phi)$. The terms related to ω in (3.88) are defined as

$$[\omega_{i,k}^{(*)m}]^{mon} = \min(1, \beta_{i,k-1}^{\uparrow}, \beta_{i,k}^{\downarrow})[\omega_i^{(*)m}]^+ + \min(1, \beta_{i-1}^{\downarrow}, \beta_i^{\uparrow})[\omega_i^{(*)m}]^-, \quad (3.90)$$

$$\beta_{i,k}^{\downarrow} = \frac{\psi_{i,k}^{(*)m-1} - \psi^{\min}}{\frac{\Delta t}{\Delta\sigma}\{[\omega_{i,k}^{(*)m}]^+\psi_{i,k}^{(*)m-1} - [\omega_{i,k+1}^{(*)m}]^-\psi_{i,k}^{(*)m-1}\} + \varepsilon}, \quad (3.91)$$

$$\beta_{i,k}^{\uparrow} = \frac{\psi^{\max} - \psi_i^{(*)k-1}}{\frac{\Delta t}{\Delta\sigma}\{[\omega_{i+1}^{(*)m}]^+\psi_{i,k+1}^{(*)m-1} - [\omega_{i,k}^{(*)m}]^-\psi_{i,k-1}^{(*)m-1}\} + \varepsilon}, \quad (3.92)$$

$$w_i^{(*)m} = \frac{\{\Delta\sigma \mid w_{i,k}^{(*)m-1} \mid -\Delta t \mid \omega_{i,k}^{(*)m-1} \mid^2\}(\psi_{i,k-1}^{(*)m-1} - \psi_{i,k}^{(*)m-1})}{(\psi_{i,k-1}^{(*)m-1} - \psi_{i,k}^{(*)m-1} + \varepsilon)\Delta\sigma} \quad (3.93)$$

where ε is a small value set up as 10^{-10}, $\omega_{i,k}^{(*)0} = \omega_{i,k}^n$.

3.3. Transport Consistency of External and Internal Modes

In a mode-split model, an adjustment must be made in every internal time step to ensure the consistency in the vertically integrated water transport produced by external and internal modes (Chen et al., 2004b). In FVCOM, the vertical velocity at the σ-surface (ω) is calculated by

$$\omega_{i,k+1} = \omega_{i,k} + \frac{\Delta\sigma_k}{\Delta t_I}(\zeta_i^{n+1} - \zeta_i^n) + \frac{\Delta\sigma_k}{\Omega_i}\oint_l v_{N,k}^n Ddl \,, \tag{3.94}$$

where *i, k*, and *n* is indicators of the *i*th node point, *k*th σ level and *n*th time step, respectively; Ω_i^n is an area of the *i*th TCE at the *n*th time step. N is an indicator of the velocity component normal to the boundary of a TCE with a length of l. To conserve the volume on the *i*th TCE, the vertically-integrated form of eq. (3.94) must satisfy

$$\frac{\zeta_i^{n+1} - \zeta_i^n}{\Delta t_I} + \sum_{k=1}^{kb-1}\frac{\Delta\sigma_k}{\Omega_i^n}\oint_l v_{N,k}^n Ddl = 0 \,, \tag{3.95}$$

where kb is the total number of the σ levels. Since ζ_n^{n+1} is calculated through the vertically integrated continuity equation over I_{split} external mode time steps (where $I_{split} = \frac{\Delta t_I}{\Delta t_E}$), Eq. (3.95) is valid only if

$$\Delta t_I \sum_{k=1}^{kb-1}\frac{\Delta\sigma_k}{\Omega_i^n}\oint_l v_{N,k}^n Ddl = \Delta t_E \sum_{\ddot{n}=1}^{I_{split}}\frac{1}{\Omega_i^{\hat{n}}}\oint_l \bar{v}_N^{\hat{n}} Ddl \,. \tag{3.96}$$

Because the numerical accuracy depends on the time step, the left and right sides of Eq.(3.96) is not exactly equal to each other unless I_{split} = 1. Therefore, to ensure the conservation of the volume transport throughout the water column of the *i*th TCE, the internal velocity in each σ-layer must be calibrated using the difference of an inequality of vertically integrated external and internal water transport before ω is calculated. Since the vertically integrated transport features the barotropic motion, the easiest calibration way is to adjust the internal velocity by distributing the "error" uniformly throughout the water column. ω, which is calculated with adjusted internal velocities through (3.94), not only guarantees that the volume transport is conservative in the whole water column but also in an individual TCE volume in each σ layer. Because the water temperature and salinity or other tracers are calculated in the same TCE volume as that used for ω, this transport adjustment also conserves the mass in an individual TEC volume in each σ layer.

3.4. The Wet/Dry Treatment Technique

One of the most difficult problems in estuarine modeling is to provide an accurate simulation of the water transport flooding onto and draining out of the inter-tidal zone, which can be several times as large as the transport in the main channel of a river. In the last three decades, two types of approaches have been developed to solve this issue: one is the moving-boundary method and the other is the wet/dry point treatment method. In the first method, the computational domain is bounded by an interface line between land and water where total water depth and normal transport are equal to zero (Lynch and Gray 1980). Because this boundary moves over flooding and draining cycles, the model grid must be re-generated at every time step. This method seems to work for idealized estuaries with simple geometries (Sidén and Lynch 1988; Austria and Aldama 1990), but it is not practical for application to realistic estuaries with complex geometry including tidal creeks, islands, barriers, and inlets. In the second method, the computational domain covers the maximum flooding area. Numerical grids consist of wet and dry points with a boundary defined as an interface line between land and water. The wet and dry points are distinguished by the local total water depth of $D = H(x, y) + \zeta(x, y, t)$ (where H is the reference water depth and ζ is the surface elevation). The wet point is a grid point with $D > 0$, otherwise, $D = 0$ at dry points. At dry points, velocities are automatically specified as zero, but salinity retains the same value from the previous time step. Since this method is relatively simple, it has been widely used to simulate the water transport over inter-tidal areas in estuarine models (Leendertse 1970 & 1987; Flather and Heaps 1975; Ip et al. 1998, Zheng et al. 2003a).

The wet/dry point treatment technique works only for the case in which a finite-value solution of the governing equations exists as D approaches zero. In a z-coordinate system, to ensure numerical stability of a 3-D model, the thickness of the layer closest to the surface must be greater than the amplitude of the tidal elevation (Davis et al. 1997). This makes it difficult to apply this method for a shallow estuary in which the amplitude of the tidal elevation is the same order of magnitude as the local water depth. Because irregular bathymetry is poorly resolved in a z-coordinate model, it is not likely that the water transport can be accurately simulated over realistic bathymetric inter-tidal zones during

both flooding and draining periods. Examples of the z-coordinate wet/dry methods can be seen in Casulli and Cheng (1991 & 1992); Cheng et al. (1993); Casulli and Cattani (1994); and Hervouet and Janin (1994). In a σ-coordinate system, the wet/dry point treatment is no longer valid as D becomes zero. One simple way to avoid numerical singularity is to specify zero velocities at all dry points. This approach, however, can not ensure volume conservation in the numerical computation due to discontinuous water removal from elements that turn to dry over one time step. An alternative way is to add a viscous boundary layer (h_c) at the bottom and redefine wet/dry points using a sum of D and h_c. The grid is treated as a wet point for $D > h_c$, otherwise it is a dry point. In terms of the nature of the vertical structure of turbulent mixing, a viscous layer always exists below the log boundary layer near a solid wall (Wilcox, 2000). However, to avoid adding additional water transport into a dynamic system, the viscous layer should be sufficiently small to satisfy a motionless condition. Good examples of the application of this method can be found in Ip et al. (1998) and Zheng et al. (2003b).

No matter which methods are used to simulate the flooding/draining process over the intertidal zone in an estuary, they must be validated with respect to mass conservation. Because in all of these methods the dry and wet points are determined using some empirical criteria, the estimation of the water transport in the dry-wet transition zone depends on 1) the criterion used to define the wet/dry points; 2) the time step used for numerical integration, 3) the horizontal and vertical resolutions of model grids, 4) amplitudes of surface elevation, and 5) bathymetry. In a σ-coordinate transformation model, it might be also related to the thickness of the bottom viscous layer ($D_{\min}$).

A new wet/dry point treatment method has been developed for use with FVCOM (see Chen et al. 2006c). This method has been validated in a series of tidal simulations using an idealized semi-enclosed estuary with an inter-tidal zone. Relationships of the time step with discrete grid resolution, amplitude of external forcing, the slope of the inter-tidal zone and thickness of the bottom viscous layer are discussed and the criterion for the selection of the time step is derived. The rule used in validation is mass conservation, which, we believe, is a prerequisite condition for an objective evaluation of the wet/dry point treatment technique in estuaries.

3.4.1 Criteria. The wet or dry criterion for node points is given as

$$\begin{cases} \text{wet}, & \text{if } D = H + \zeta + h_B > D_{\min} \\ \text{dry} & \text{if } D = H + \zeta + h_B \le D_{\min} \end{cases} \tag{3.97}$$

and for triangular cells is given as

$$\begin{cases} \text{wet}, & \text{if } D = \min(h_{B,\hat{i}}, h_{B,\hat{j}}, h_{B,\hat{k}}) + \max(\zeta_{\hat{i}}, \zeta_{\hat{j}}, \zeta_{\hat{k}}) > D_{\min} \\ \text{dry}, & \text{if } D = \min(h_{B,\hat{i}}, h_{B,\hat{j}}, h_{B,\hat{k}}) + \max(\zeta_{\hat{i}}, \zeta_{\hat{j}}, \zeta_{\hat{k}}) \le D_{\min} \end{cases} \tag{3.98}$$

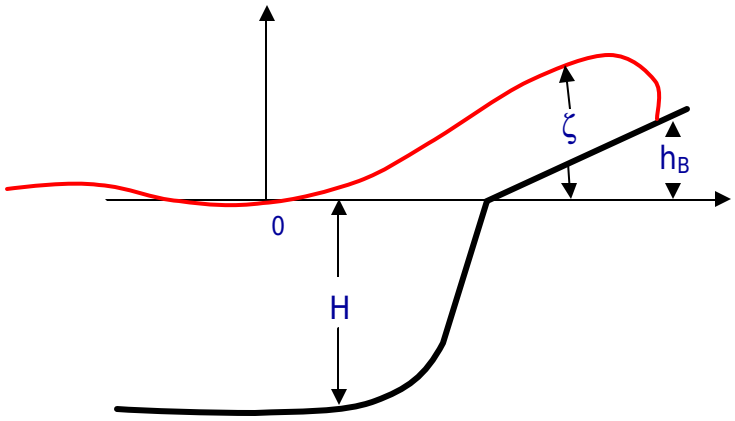


Fig. 3.5: Definition of reference depth (H), surface level (ζ) and bathymetric height (h_B).

where $D_{\min}$ is the thickness of a viscous layer specified at the bottom, h_B is the bathymetric height related to the edge of the main channel of a river (Fig.3.5) and $\hat{i}$, $\hat{j}$ and $\hat{k}$ are the integer numbers to identify the three node points of a triangular cell.

When a triangular cell is treated as dry, the velocity at the centroid of this triangle is specified to be zero and no flux is allowed on the three boundaries of this triangle. This triangular cell is removed from the flux calculation in the TCE. For example, the integral form of the continuity equation in FVCOM is written as

$$\iint_{TCE} \frac{\partial \zeta}{\partial t} dxdy = -\iint_{TCE} [\frac{\partial (\overline{u} D)}{\partial x} + \frac{\partial \overline{v} D}{\partial y}] dxdy = -\oint_{l} \overline{v}_N D dl \tag{3.99}$$

where $\overline{u}$ and $\overline{v}$ are the x and y components of the vertically-averaged velocity. In a dry/wet point system, only wet triangles are taken into account in the flux calculation in a TCE since the flux on boundaries of the dry triangle is zero (see Fig. 3.6). This approach always ensures the volume conservation in a TCE that contains the moving boundary

between the dry and wet triangles over an integration interval. The same approach is used to calculate the tracer flux (temperature, salary and other scalar tracers) and momentum flux in a MCE.

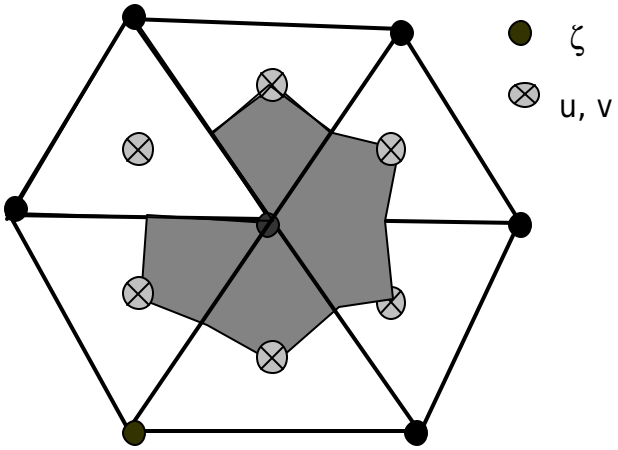


Fig. 3.6: Illustration of dye/wet triangles within a TCE.

One of the critical issues in applying the wet/dry point treatment technique into a split mode model is to ensure mass conservation in the individual TCE which is crossed by the moving boundary. Because $\Omega_i^{\hat{n}}$ may change within I_{split} external time integrations due to the occurrence of dry triangles and ζ is treated as zero when D is less than $D_{\min}$, we have:

$$\zeta_i^{n+1} - \zeta_i^{n} \neq \Delta t_E \sum_{\hat{n}=1}^{I_{split}} \frac{1}{\Omega_i^{\hat{n}}} \oint_l \overline{v}_N^{\hat{n}} D dl \,. \tag{3.100}$$

In this case, the external and internal mode adjustment through Eq.(3.96) can not guarantee that ω reaches zero at $\sigma = -1$ for the internal mode. To ensure the volume conservation, an additional adjustment for ζ_i^{n+1} must be made in the TCE when ω is calculated by Eq.(3.95).

The additional sea level adjustment works in general, but fails in the case where ζ is very close to $D_{\min}$ (for example, $\Delta\zeta = \zeta - D_{\min} < 10^{-1}$). When this happens, small errors in the calculation of volume flux can rapidly accumulate through nonlinear feedbacks of tracer advection and eventually destroy the nature of the mass conservation.

FVCOM always ensures the mass conservation in the wet-dry transition zone if I_{split} =1. For computational efficiency, however, We want to find an approach so that the mass conservation is still guaranteed for the case in which I_{split} >1. In additional to the criterion of general numerical instability, in a case with inclusion of the flooding/drying process, the choice of I_{split} is restricted by many other factors including the surface elevation, bathymetry, thickness of the bottom viscous layer and horizontal/vertical resolutions. A discussion on the relationship of I_{split} to these factors is given next through numerical experiments for idealized cases.

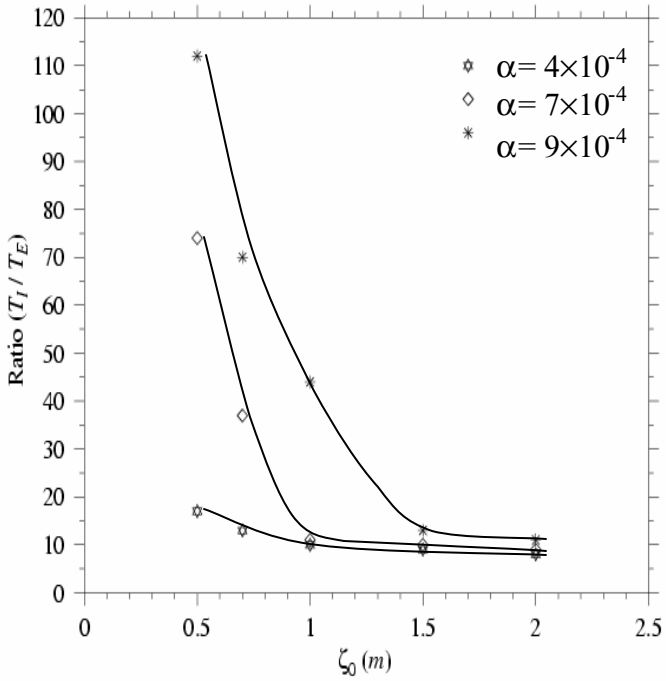


Fig. 3.7: The model-predicted relationship of the ratio of the internal to external mode time steps ($\Delta t_I / \Delta t_E$) with the tidal forcing amplitude (ζ_o) and the bottom slope of the inter-tidal zone (α). In these experiments, Δt_E = 4.14 sec, kb = 6, $D_{\min}$ = 5 cm.

3.4.2. The upper-bound limit of I_{split}. By simulating the flooding/drying process in an idealized semi-enclosed channel with an inter-tidal zone, we examined the relationship of the model forcing and geometric parameters on the upper-bound limit of I_{split} (Chen et al., 2004c). A brief summary of the model results is given below.

The relationship with α and ζ_o. The model results show that the upper-bound value of I_{split} varies with the bottom slope of the inter-tidal zone (α) and amplitude of tidal forcing (ζ_o) (Fig. 3.7). Considering a standard case with a constant slope of $\alpha = 4.0\times10^{-4}$, the upper-bound value of I_{split} gradually becomes smaller as ζ_o becomes larger. It is below 10 at ζ_o=2.0 m and up to 15 at ζ_o=0.5 m. When the slope

is up to 7.0×10^{-4} (a change in the height of the inter-tidal zone up to 1.4 m over a distance of 2 km), however, the upper-bound value of I_{split} dramatically increases in a tidal forcing range of ζ_o < 1.0 m, even though it remains only slightly higher than the standard case with larger tidal forcing. The model still conserves mass in the wet-dry transition zone at I_{split} = 70 at ζ_o = 0.5 m. I_{split} becomes even more flexible in the case with steeper

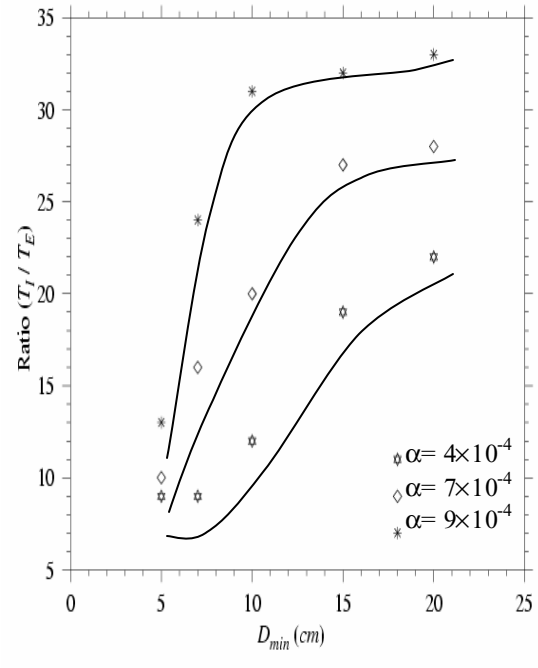


Figure 3.8: The model-derived relationship of $\Delta t_I/\Delta t_E$ with ζ_o for the three cases with α = 4.0×10^{-4}, 7.0×10^{-4} and 9.0×10^{-4}. In the three cases, Δt_E = 4.14 sec, kb = 6, and D_{mi} = 1.5 m.

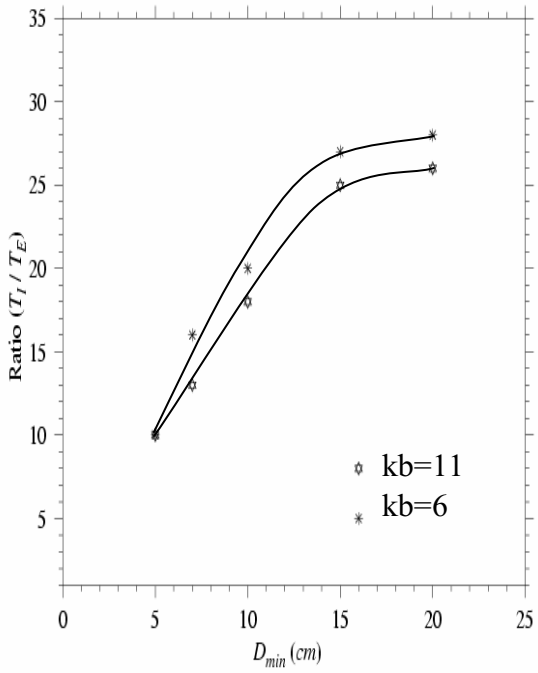


Figure 3.9: The model-derived relationship of I_{split} with $D_{\min}$ for the two cases with kb = 6 and 11, respectively. In these two cases, α = 4.14 sec, Δt_E = 4.0×10^{-4}, and ζ_o = 1.5 m.

slope of $\alpha = 9.0\times10^{-4}$ (a change in the height of the inter-tidal zone up to 1.8 m over a distance of 2 km). In the tidal forcing range of ζ_o < 1.5 m, the upper bound value of I_{split} increases almost exponentially with the decrease of ζ_o. Even in the larger tidal forcing range of ζ_o > 1.5 m, the upper-bound value of I_{split} exceeds 10.

The relationship with $D_{\min}$. In general, under given tidal forcing, vertical/horizontal resolutions, and external mode time step, the upper bound value of I_{split} increases as $D_{\min}$ becomes larger (Fig. 3.8). In the standard case with ζ_o = 1.5 m and $\alpha = 4.0\times10^{-4}$, for example, I_{split} must be smaller or equal to 9 for the case with $D_{\min}$ =5 cm, but it could be 22 for the case with $D_{\min}$ = 20 cm. I_{split} could be much larger in the case with a steeper slope of the inter-tidal zone. In the cases with $\alpha = 7.0\times10^{-4}$ and 9.0×10^{-4}, the upper-bound value of I_{split} could be up to 10 and 13, respectively for $D_{\min}$ =5 cm and up to 28 and 33, respectively for $D_{\min}$ = 20 cm.

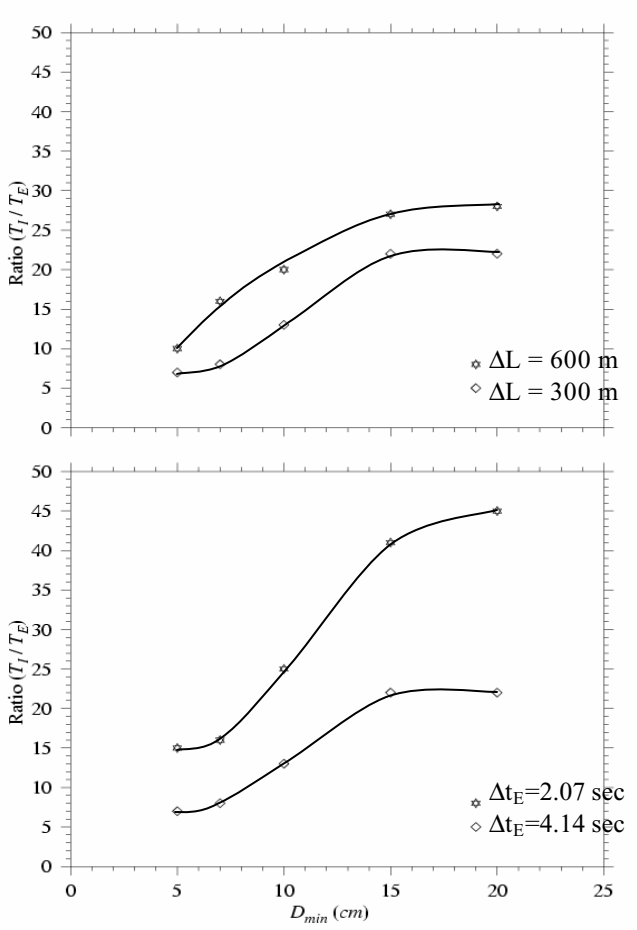


Fig. 3.10: The model-derived relationship of I_{split} with ΔL (upper panel) and Δt_E (lower panel). In the upper panel case, Δt_E = 4.14 sec, $\alpha = 7.0\times10^{-4}$, kb =6, and ζ_o = 1.5 m. In the lower panel case, ΔL = 300 m, $\alpha = 7.0\times10^{-4}$, kb =6, and ζ_o = 1.5 m.

The relationship with kb (# of sigma levels). The up-bound limit of I_{split} with respect to vertical resolution is sensitive to the thickness ($D_{\min}$) of a viscous layer specified in the model (Fig. 3.9). For a standard case with kb = 6 and $\alpha = 7.0\times10^{-4}$, for example, with a tidal forcing of ζ_o = 1.5, the upper-bound value of I_{split} is 10 at $D_{\min}$ = 5 cm and up to 28 at $D_{\min}$ = 20 cm. Keeping the same forcing condition but increasing kb to 11, we found that the up-bound value of I_{split} remains almost the same at $D_{\min}$ =5 cm but drops significantly as $D_{\min}$ increases.

The relationship with ΔL and Δt_E. For a given Δt_E, the upper-bound value of I_{split} decreases as horizontal resolution increases (Fig. 3.10). For a standard case, for example, with a tidal forcing of ζ_o = 1.5 m, the upper-bound limit of I_{split} is cutoff at 10 for $D_{\min}$ = 5 cm and at 28 for $D_{\min}$ = 20 cm. These values, however, drop to 7 for $D_{\min}$ = 5 and to 21 for $D_{\min}$ = 20 cm when ΔL decreases to 300 m (Fig. 3.10: upper panel). Similarly, for a given ΔL = 300 m, the upper-bound value of I_{split} increases significantly as Δt_E decreases, which jumps from 7 to 15 at $D_{\min}$ = 5 cm and from 21 to 45 at $D_{\min}$ = 20 as Δt_E decreases from 4.14 sec to 2.07 sec (Fig.3.10: lower panel). In a range of $D_{\min}$ shown in Fig. 3.10, for the two cases with $(\Delta t_E)_1$ and $(\Delta t_E)_2$, $(I_{split})_2$ is approximately estimated by

$$(I_{split})_2 \sim (I_{split})_1 \frac{(\Delta t_E)_1}{(\Delta t_E)_2} .$$

It should be noted here that the actually upper-bound limit of I_{split} in application to realistic estuaries and coastal flooding areas might be different from the results presented here for our idealized test case. The key point of presenting these idealized model results is to inform users about the dependence of the upper-bound of I_{split} on tidal ranges, bottom slope in the inter-tidal zone, thickness of the specified viscous layer, external time step, and vertical/horizontal resolution, and provide a guide for choosing I_{split} in realistic applications.

3.5. Finite-Volume Discrete Methods in Spherical Coordinate System

The numerical methods used to solve the spherical coordinate version of FVCOM are the same as those used in the Cartesian coordinate version of FVCOM with two exceptions, the redefinition of the meridian flux and North Pole treatment. In both Cartesian and spherical coordinates, we have introduced a new flux corrected second-order scheme to calculate the tracer advection. The discrete procedure of FVCOM was given in detail in Chen et al. (2003) and Chen et al. (2004), and brief descriptions of the re-definition of meridian flux, the discrete scheme for the tracer advection, and North Pole treatment are given below. The text is directly adopted from Chen et al. (2006b).

Following the same approach used in the Cartesian coordinate version of FVCOM, the horizontal numerical computational domain is subdivided into a set of non-overlapping unstructured triangular cells. An unstructured triangle is comprised of three nodes, a centroid, and three sides (Fig. 3.11), on which *u* and *v* are placed at centroids and

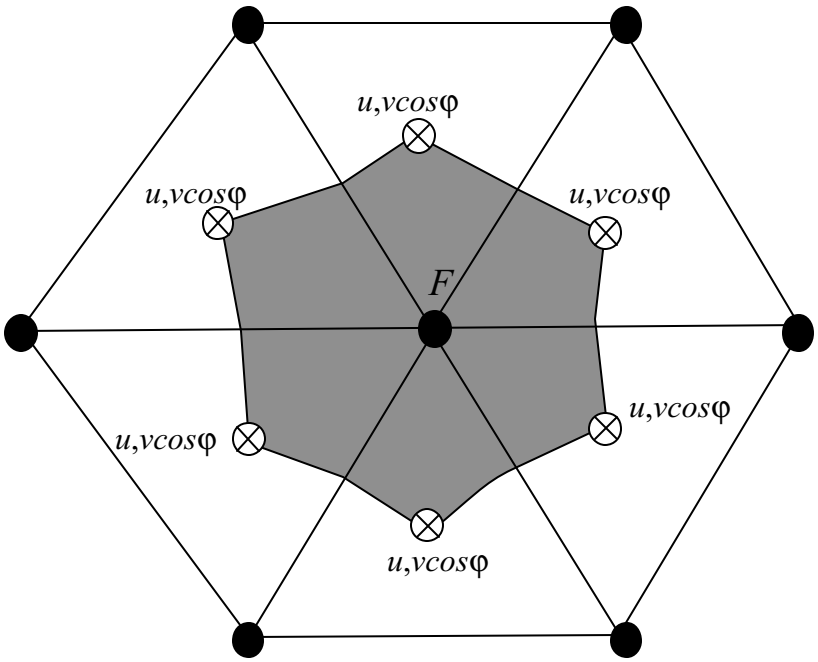


Fig. 3.11: Schematic of the control volume used to calculate scalar variables and vertical velocity used in FVCOM. *F* is a general symbol representing scalar variables such as "ζ, *T*, *S*, K_m, K_h, and vertical velocity ω. • is the node of triangles where scalar variable or vertical velocity is calculated and ⊗ is the centroid of a triangle where the horizontal velocity is calculated.

all scalar variables, such as ζ, *H*, D, ω, *S*, *T*, ρ, K_m, K_h, A_m, and A_h are placed at nodes.). *u* and *v* at centroids are calculated based on the net flux through three sides of that triangle (called the momentum control element: MCE), while scalar variables at each node are determined by the net flux through the sections linked to centroids and the middle point of the sideline in the surrounding triangles (called the tracer control element: TCE).

In both 2-D (external mode) and 3-D (internal mode) momentum equations, the advection term is calculated in the flux form using a second-order accurate upwind finite-difference scheme (Kobayashi et al., 1999; Hubbard, 1999, Chen et al. 2003a), which is the same as that used in the Cartesian FVCOM. In this scheme, the velocity in the triangle cell *i* is assumed to satisfy the linear distribution given as

$$u_i(x',y') = \phi_i^u(x',y') = u_i(x_c,y_c) + a_i^u(x'-x_c) + b_i^u(y'-y_c) \tag{3.101}$$

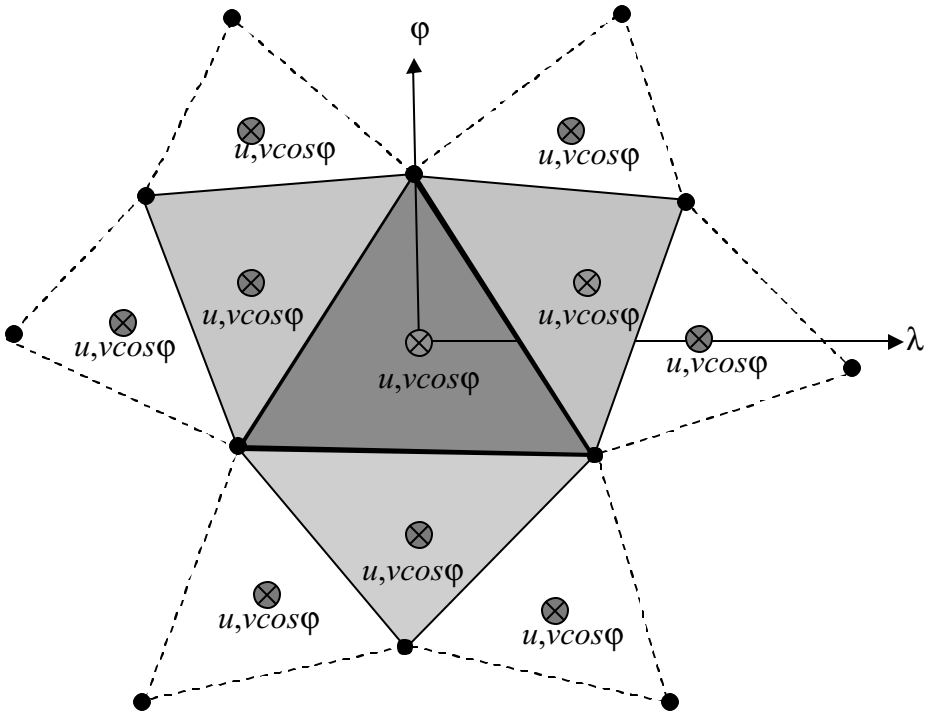


Fig. 3.12: Schematic of the momentum control volume (bounded by heavy solid lines) used to calculate the horizontal velocity. Light gray filled triangles are surrounding meshes required to solve the linear equation to determine the velocity distribution.

$$v_i(x',y') = \phi_i^v(x',y') = v_i(x_c,y_c) + a_i^v(x'-x_c) + b_i^v(y'-y_c) \tag{3.102}$$

where (x_c, y_c) is the location of the center of the triangular cell *i* and (x', y') is the location of any point in three adjacent triangular cells. The parameters a_i^u, b_i^u, a_i^v, and b_i^v are determined by a least-square method based on velocity values at the four cell-centered points shown in Fig. 3.12. On the sphere, the curved nature of the surface needs

to be taken into account when the area, lengths, and center of a triangle are calculated. The area of a triangle on a sphere equals

$$ART = r^2\delta \tag{3.103}$$

and

$$\delta = 2\arcsin\left[\frac{\sqrt{\sin\ p\sin(\ p-a)\sin(\ p-b)\sin(\ p-c)}}{2\cos\frac{a}{2}\cos\frac{b}{2}\cos\frac{c}{2}}\right] \tag{3.104}$$

where $p = \frac{1}{2}(a+b+c)$, and a, b, and c are the arc length of three side boundaries of a triangle. The arc length (defined as $\overline{AB}$) between two points (λ_a, φ_a) and (λ_b, φ_b) is calculated as follows. The x, y and z at these two points can be given as

$$x_a = r\cos\varphi_a\cos\lambda_a,\ y_a = r\cos\varphi_a\sin\lambda_a,\ z_a = r\sin\varphi_a\ , \tag{3.105}$$

$$x_b = r\cos\varphi_b\cos\lambda_b,\ y_b = r\cos\varphi_b\sin\lambda_b,\ z_b = r\sin\varphi_b\ . \tag{3.106}$$

The "string" distance (defined as $\overline{ab}$) between these two points is equal to

$$\overline{ab} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2}\ . \tag{3.107}$$

According to the *cosine* theorem, the central angle α of the arc can be expressed as

$$\alpha = \arccos\frac{2r^2 - ab^2}{2r^2}. \tag{3.108}$$

Therefore, the arc length $\overline{AB}$ between point A and point B is equal to

$$\overline{AB} = r\alpha\ . \tag{3.109}$$

The x_c, $y_{c,}$ and z_c at the string center are equal to

$$x_c = r\frac{\cos\varphi_a\cos\lambda_a + \cos\varphi_b\cos\lambda_b}{2}, \tag{3.110}$$

$$y_c = r\frac{\cos\varphi_a\sin\lambda_a + \cos\varphi_b\sin\lambda_b}{2}\ , \tag{3.111}$$

$$z_c = r\frac{\sin\varphi_a + \sin\varphi_b}{2}. \tag{3.112}$$

The arc lengths in the x and y directions between two points (λ_a, φ_a) and (λ_b, φ_b) are given as

$$dx = r\cos\varphi d\lambda \text{ and } dy = rd\varphi \ . \tag{3.113}$$

After the area of the MCE and arc lengths relative to the center of the MCE are determined, the distribution of u and v in four triangles shown in Fig. 3.12 can be determined, and the second-order upwind scheme used by the Cartesian FVCOM code can be directly used to calculate the advection terms in the momentum equations (see Chen et al. 2003a for details). Unlike the Cartesian version of FVCOM, the line integral for the flux calculation is done with respect to λ and φ rather than the arc length of the boundary line of the TCE or MCE. Thus, the meridian flux determined by v is calculated using $v\cos\varphi$, although the momentum equations are still solved for u and v. This method guarantees that the line integral flux calculation method is valid in the spherical coordinate system because the integral around a closed boundary path constructed by λ and φ equals zero.

The flux through the TCE is calculated using the line integral around the closed path defined by λ and φ. In the 2-D continuity equation, for example,

$$\iint_{\Omega} \frac{\partial ?}{\partial t} r^2 \cos\varphi d\lambda d\varphi = \iint_{\Omega} \frac{1}{r\cos\varphi}\{\frac{\partial(\overline{u}D)}{\partial\lambda} + \frac{\partial[(\overline{v}\cos\varphi)D]}{\partial\varphi}\} r^2 \cos\varphi d\lambda d\varphi \ , \tag{3.114}$$

Therefore, we have

$$\frac{\partial ?}{\partial t} = -\frac{r}{\Omega}[\oint_{\varphi}(\overline{u}D)d\varphi' - \oint_{\lambda}(\overline{v}\cos\varphi D)d\lambda'] \ , \tag{3.115}$$

where Ω is the area of the TCE. The line integral is done around the closed path of λ or φ, which guarantees volume conservation. The momentum and continuity equations for the 2-D mode are integrated numerically using the modified fourth-order Runge-Kutta time-stepping scheme. A detailed description of this method was given in Chen et al. (2003a).

The unstructured triangular grid used in FVCOM allows us to nest the spherical and polar stereographic projection coordinates to avoid the singularity problem at the North Pole. Eight equilateral triangles are set up around the pole (Fig. 3.13), at which all variables at nodes and centroids of these triangles are calculated in the polar stereographic projection coordinates. Variables in MCEs and TCEs with no direct connection to the North Pole are calculated in spherical coordinates. The velocity in

these two coordinates are converted to spherical coordinates by the relationship defined as

$$\begin{pmatrix} u_p \\ v_p \end{pmatrix} = \begin{pmatrix} -\sin\lambda & -\cos\lambda \\ \cos\lambda & -\sin\lambda \end{pmatrix} \begin{pmatrix} u_s \\ v_s \end{pmatrix}, \tag{3.116}$$

$$u_p = h_x \frac{dx}{dt}, v_p = h_y \frac{dy}{dt}, h_x = h_y = \frac{1+\sin\varphi}{2}. \tag{3.117}$$

A so-called "North Pole Nesting Module" is used in the spherical-coordinate version

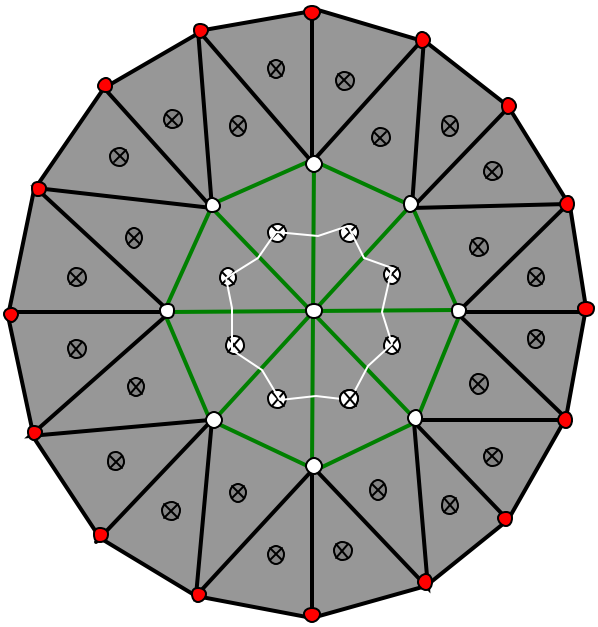
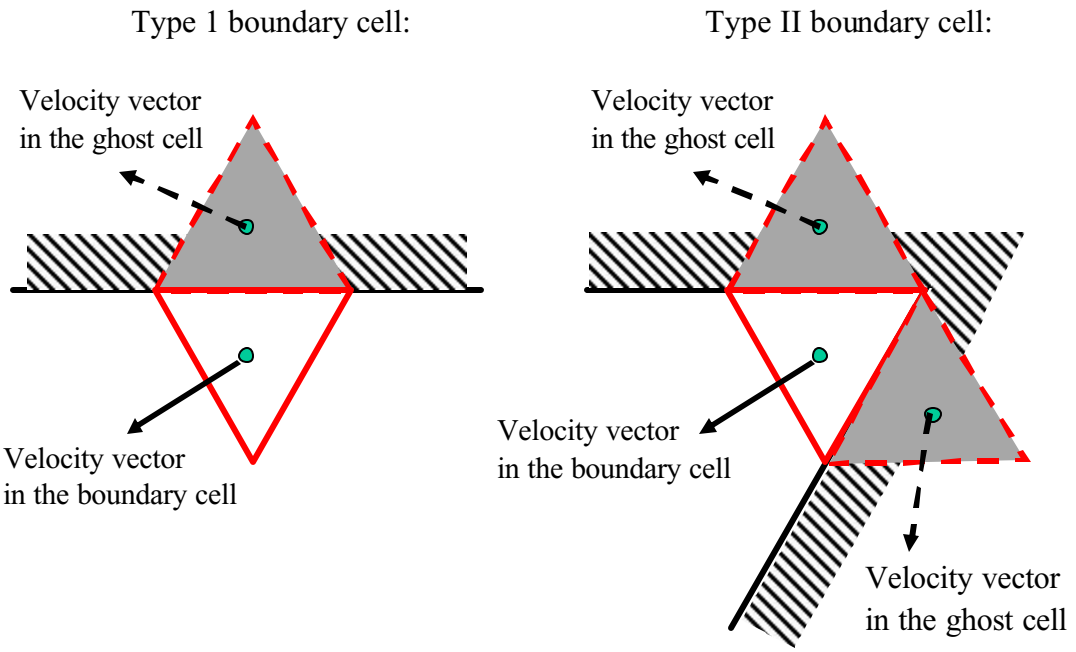


Fig. 3.13: Illustration of the nested spherical-polar stereographic projection grid at the North Pole.

of FVCOM, in which all cells connected to the North Pole are defined as the polar cells. This module allows an automatic selection of two systems based on the definition of a cell for the parallelized computation. This approach efficiently uses the flexibility of the unstructured grid to make FVCOM run directly in spherical coordinates for basin or larger (even global) scale ocean application without the need for "grid rotation" or "multi-projection" methods.

3.6 Ghost-Cell Treatment for the Coastal Boundary Condition

In the original version of FVCOM, the velocity in a cell adjacent to a solid boundary is updated using the same method as interior cells and is then adjusted so that the component normal to the wall is zero. This treatment works well for applications where the velocity field must only make small adjustments to the coastline, such as where the flow is predominantly tangential or the coast is smoothly varying. For cases where the coastal angle is rapidly changing or is normal to the locally forced flow, as in the case of an island, the original condition is unrealistic. By removing the normal component, an incoming wave is simply damped and energy reflection is reduced. We have thus implemented an improved treatment in FVCOM using the traditional ghost cell approach in the current version. Brief descriptions of the methodology used in this treatment are given below.



For the type I solid boundary, where a triangle adjacent to a boundary has a single solid and two interior edges, a ghost cell is automatically created in the model in which the velocity has the same tangential component but opposite normal component to the adjacent boundary cell. The resulting velocity at the edge satisfies the proper solid boundary condition for inviscid flow. The adjacent boundary cell can thus be updated

using the same techniques used in interior cells. For type II solid boundaries, where the boundary triangle has two solid edges, two ghost cells are automatically created in the model with the same treatment as the type I sold boundary cell. Although it does not affect the stability of the model run, users should try to avoid the inclusion of type II cells when generating the mesh since they are not strongly linked to the interior flow..

Fig. 3.14 shows the model-computed distribution of the near-surface current vector at the end of the 10th model day for the case of a river discharge plume over an idealized continental shelf. It can be clearly see that the original treatment of the solid boundary can affect the near-coastal distribution of the current and also can overestimate the along-shelf water transport.

The ghost treatment is particularly important for the wave problem around an island. When the water is stratified, selection of the ghost cell boundary treatment might require a smaller time step when compared to the original setup. In the case where the boundary edge is

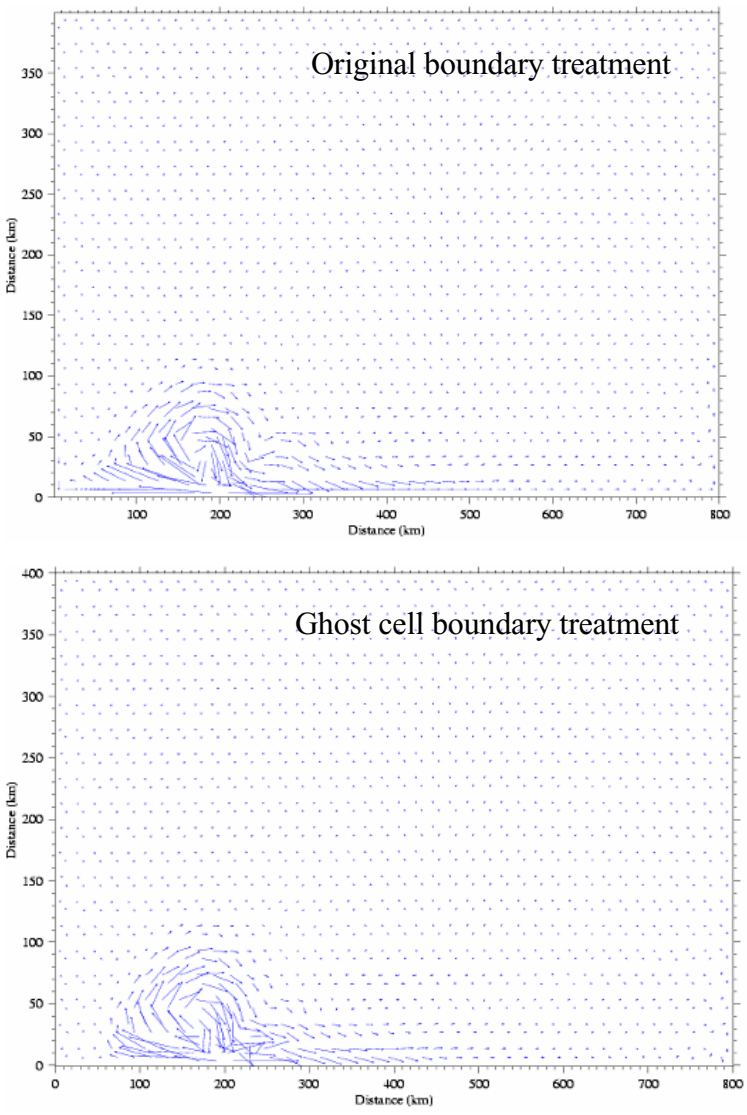


Fig. 3.14: The near-surface distribution of the current vector at the end of the 10th model day for a river discharge plume experiment.

normal to the dominant flow direction, the ghost cell boundary condition leads to a reflection energy which can generate high frequency internal waves that require a reduction in time step

Chapter 4: External Forcing

External physical forcings driving FVCOM include the surface wind stress, heat flux, precipitation/evaporation, tides, river discharges and groundwater flux. Brief descriptions of these forcing processes and how they are applied in FVCOM are given below.

4.1. Wind Stress, Heat Flux and Precipitation/Evaporation

The surface wind stress, heat flux and precipitation/evaporation are added to FVCOM using state-of-the-art methods that are widely used in ocean circulation modeling. Users can either run FVCOM with a constant uniform forcing or with time-dependent, spatially non-uniform forcing fields specified from observational data or from the output of a meteorological model or a combination of both. FVCOM has been loosely coupled with the fifth-generation NCAR/Penn State mesoscale meteorological model (called MM5), so that the output of the MM5 wind stress, heat flux and precipitation/evaporation fields can be used to drive FVCOM, and the FVCOM assimilated surface water temperature can be feedback to the MM5 model for use in the sensible and latent heat flux estimation (Chen et al. 2005).

It should be noted here that the original version of FVCOM included all subroutines used to input the precipitation minus evaporation field. Since this part of the model code has not been tested or used in any of our current applications, it is not released in the present version 2.5. For users who wish to test this code and/or use the precipitation minus evaporation field in their applications, please contact directly Dr. Chen. We will add the precipitation/evaporation module into the release version of FVCOM after we complete an initial validation.

4.2. Tidal Forcing

FVCOM has incorporated the Foreman (1978) tidal forecasting program to compute realistic tidal elevation data for the initial conditions and on the open boundary. Up to six tidal harmonic constituents can be included in the model. They are given as

$$\zeta_o = \bar{\zeta}_o + \sum_{i=1}^{N_o} \hat{\zeta}_i \cos(\omega_i t - \theta_i) \tag{4.1}$$

where $\overline{\zeta}_o$ is the mean elevation relative to a water level at rest; $\hat{\zeta}_i$, ω_i and θ_i are the amplitude, frequency and phase of the *i*th tidal constituent. N_o is the total number of tidal constituents, which is set up to be 6 in the current version of FVCOM. This number, however, can be increased according to users' needs. The six included tidal constituents are: 1) S_2 tide (period = 12 hours); 2) M_2 tide (period = 12.42 hours); 3) N_2 tide (period = 12.66 hours); 4) K_1 tide (period = 23.94 hours); 5) P_1 tide (period = 24.06 hours), and O_1 tide (period = 25.82 hours).

In larger scale regional applications, the equilibrium tide generated by the gravitational-centrifugal potential can not be ignored and the potential gradient force must be included in the tidal simulation. The equilibrium tidal gradient force is derived from the elevations given as

$$\overline{\zeta}_{e(semi)} = \sum_{i=1}^{N_{semi}} \beta_i A_{e_i} \cos^2 \varphi \cos(\omega_{e_i} t + 2\lambda) \tag{4.2}$$

for semidiurnal tides and

$$\overline{\zeta}_{e(diurnal)} = \sum_{i=1}^{N_{diurnal}} \beta_i \hat{A}_{e_i} \cos 2\varphi \cos(\hat{\omega}_{e_i} t + \lambda) \tag{4.3}$$

for diurnal tides. Here A_{ei} and ω_{e_i} are the amplitude and frequency of the *i*th semidiurnal equilibrium tidal elevation, respectively; φ is latitude; λ is longitude; N_{semi} and $N_{diurnal}$ are the total number of the semidiurnal and diurnal equilibrium tidal constituents included in the model, respectively, and β_i is the parameter specified as

$$\beta_i = 1 + K_{Love} - H_{Love} . \tag{4.4}$$

K_{Love} and H_{Love} are different for different tidal constituents. In the current version of FVCOM, 5 equilibrium tidal constituents are included and the values of K_{Love} and H_{Love} for these constituents are given in Table 4.1.

Table 4.1: Parameters of K_{Love} and H_{Love}

	S_2	M_2	N_2	K_1	O_1
K_{Love}	0.302	0.302	0.302	0.256	0.298
H_{Love}	0.602	0.602	0.602	0.52	0.603

Similarly, FVCOM also includes the atmospheric tidal potential forcing. For example, the atmospheric S_2 tidal forcing is derived from the elevation given as

$$\zeta_{a(S_2)} = \overline{A}_{a(S_2)} \cos^2 \varphi \cos(\omega_{a(S_2)} t + 2\lambda - \alpha_a) \tag{4.5}$$

where $\overline{A}_{a(S_2)}$ and $\omega_{a(S_2)}$ are the amplitude and frequency of the atmospheric S_2 tidal elevation, and α_a is a parameter given as 112.0. $\overline{A}_{a(S_2)}$ and $\omega_{a(S_2)}$ are specified according to Haurwitz (1956) (see Chapman and Lindzen, 1970).

The tidal forcing at the open boundary can be specified by either amplitude/phase or by time series calculated using Foreman's tidal forecast model. In the former case, the model time base can be arbitrary, i.e., not tied to a specific Gregorian time. The harmonic analysis is carried out by using a least square fitting. In the second case, true clock time must be specified and the results can be analyzed directly using Foreman's harmonic analysis program. Normally amplitude/phase specification is used to test the model, and prescribed elevation is used in simulations for a specific Gregorian time period.

4.3. Methods to Add the Discharge from the Coast or a River

FVCOM incorporates two methods for including the discharge of fresh water or tracer transport from the coastal solid boundary. The first is to inject the water into the tracer control element (TCE) (Fig. 3.6) and the second is to input the water into the momentum control element (MCE) (Fig. 3.7). In each of these methods, the tracer concentration (such as salinity, temperature or others) can be either specified or calculated through the tracer equation. The discrete expressions for these two approaches are described in detail below.

4.3.1. The TCE Method

Let Q be the water volume transport into a TCE with an area of Ω^{ζ} and a depth of D (shown in Fig. 4.1). The surface elevation at the coastal node in this TCE can be calculated by

$$\frac{\partial \zeta}{\partial t} = [-\oint_s v_n D ds + Q] / \Omega^{\zeta}, \tag{4.6}$$

where v_n is the velocity component normal to the boundary line of the TCE and s is the closed trajectory of the boundary of the TCE. The way to include Q in the continuity equation is equivalent to adding the flux into a TCE from its coastal boundary lines (see the heavy line shown in Fig. 4.1). Since this boundary line links to the two momentum

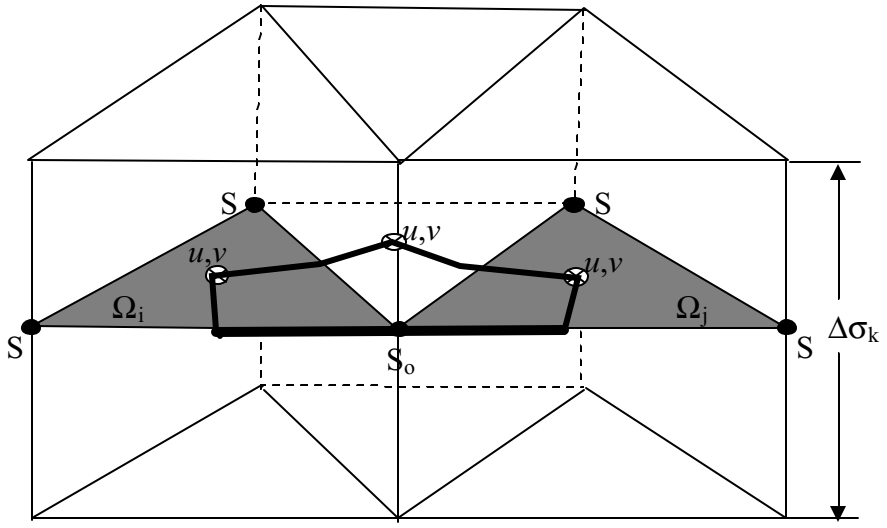


Fig. 4.1: Illustration of the river discharge as a point.

control elements (MCE) (shaded gray in Fig. 4.1), the contribution of Q to the momentum in these two elements needs to be taken into account.

For the external mode, defining that l_i and l_j are half-lengths of the coastal sides of the two MCE triangles with areas of Ω_i and Ω_j, respectively, the vertically-averaged x and y components of the velocity resulting from Q equal to

$$U_o = \frac{Q\cos\hat{\theta}}{D(l_i + l_j)},\ V_o = \frac{Q\sin\hat{\theta}}{D(l_i + l_j)} \tag{4.7}$$

where $\hat{\theta}$ is the angle of the coastline relative to the x direction. The contributions of Q to the x and y vertically-integrated momentum equations in the MCE with an area of Ω_i or Ω_j are given as $0.5QU_o$ and $0.5QV_o$, respectively.

For the internal mode, define that R_{Qk} is the percentage of Q in the k*th* sigma layer which satisfies the condition of

$$\sum_{k=1}^{KM-1} R_{Qk} = 1, \tag{4.8}$$

where KM is the number of sigma levels in the vertical. The transport entering the kth sigma layer in the TCE is equal to QR_{Qk} and the x and y components of the velocity resulted from this amount of the water transport are given as

$$U_{ok} = \frac{QR_{Qk}\cos\hat{\theta}}{D(l_i + l_j)\Delta\sigma_k}; V_{ok} = \frac{QR_{Qk}\sin\hat{\theta}}{D(l_i + l_j)\Delta\sigma_k} \tag{4.9}$$

where $\Delta\sigma_k$ is the thickness of the kth sigma layer. Therefore, the contributions of QR_{Qk} to the x and y momentum equation in the kth sigma layer of the MCE with an area of Ω_i or Ω_j are given as $0.5QR_{Qk}U_{ok}$ and $0.5QR_{Qk}V_{ok}$, respectively.

The tracer concentration (such as salinity, temperature, or others) at the coastal node of the TCE can be either specified or calculated. For the first case, the tracer concentration at the coastal node is specified by users at each time step, so that no calculation is needed to solve the tracer equation for the TCE where Q is added. This method is built on an assumption that no mixing occurs in the TCE where the water is injected from the coast or rivers. It is also the method that is usually used in finite-difference models for point sources. The advantage of this method is that it is conceptually simple, but it may cause unrealistic buoyancy gradients near the discharge source, especially for the case with coarse horizontal resolution.

For the second case, the tracer concentration at the coastal node where Q is added is calculated directly from the tracer equation, with an assumption that the water injected into the system directly contributes to the tracer transport in the TCE (where the discharge source is located) and the tracer concentration at the coastal node of this TCE is determined by the adjusted net tracer flux and mixing. For example, defining that S_{ok} is the salinity in the kth sigma layer at the coastal node of the TCE where Q is added, it can be determined by the salinity equation in an integral form as

$$\frac{\partial S_{ok} D}{\partial t} = [-\oint_{s-(l_i+l_j)} v_{nk} S_k D ds + \iint_{\Omega^\zeta} F_s dx dy + QR_{Qk}\hat{S}_{ok}]/\Omega^\zeta \tag{4.10}$$

where v_{nk} is the velocity component normal to the boundary of the TCE in the kth sigma layer, S_k is the salinity at nodes of triangles connecting to the coastal node of the TCE, F_s is the horizontal and vertical diffusion terms in the salinity equation, and $\hat{S}_{ok}$ is the salinity contained in the water volume of Q.

4.3.2. The MCE Method

Let Q be the water volume transport into a MCE, $\hat{\theta}$ is the angle of the coastline relative to the x direction, and l is the length of the coastal boundary of the MCE (Fig. 4.2). The vertically-averaged x and y components of the velocity driven by Q can be estimated as

$$U_o = \frac{Q}{Dl}\cos\hat{\theta}\ ;\ V_o = \frac{Q}{Dl}\sin\hat{\theta}\ . \tag{4.11}$$

Using the same definition of R_{Qk} described in (4.8), the x and y components of the velocity in the kth sigma layer in the MCE are given as

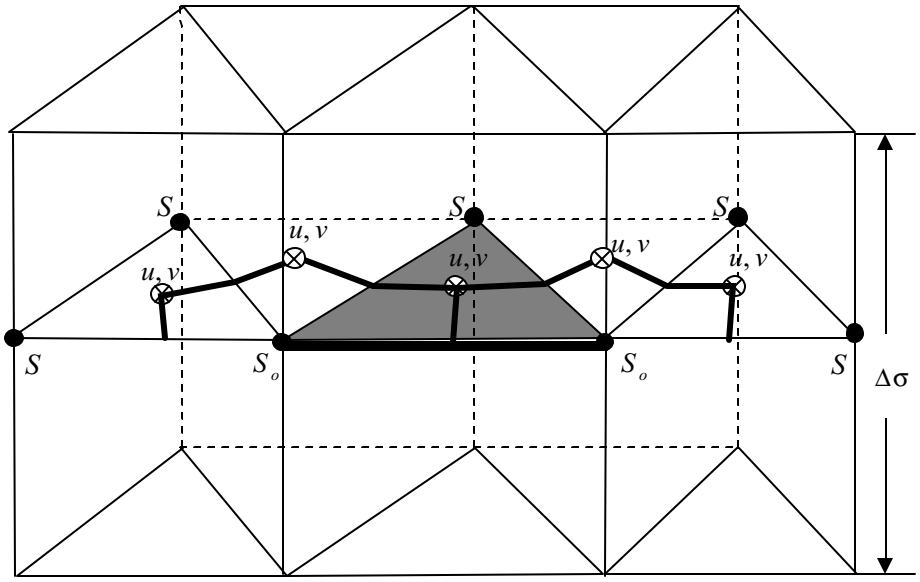


Fig. 4.2: Illustration of the freshwater discharge entering a MCE from the coastal boundary.

$$U_{ok} = \frac{QR_{Qk}\cos\hat{\theta}}{Dl\Delta\sigma_k};\ V_{ok} = \frac{QR_{Qk}\sin\hat{\theta}}{Dl\Delta\sigma_k}\quad . \tag{4.12}$$

Therefore, the contributions of the freshwater discharge to the external x and y momentum equations of the MCE are QU_o and QV_o, respectively, while the contributions to the internal x and y momentum equations of the MCE in the kth sigma layer are equal to $QR_{Qk}U_{ok}$ and $QR_{Qk}V_{ok}$, respectively.

Because the freshwater discharge is injected in the computational domain from a single MCE, we assume that there is an along-coast gradient of sea level due to this discharge. A simple way to satisfy this condition is to choose the same geometric shape for the two surrounding TCEs that connect to the MCE where the freshwater is input and assume that the freshwater equally flow into these two TCEs. Let i and j represent the two TCEs connected to the freshwater source, then

$$Q_i = Q_j = Q/2 . \tag{4.13}$$

Note that this assumption is only valid when the two surrounding TCEs have the same geometric shapes. Therefore, the surface elevation at the coastal node in the ith and jth TCEs can be calculated by

$$\frac{\partial \zeta_I}{\partial t} = [-\oint_s v_{nI} D_I ds + Q_I]/\Omega_I^{\zeta} , \tag{4.14}$$

where I are either i or j, and v_{nI} is the velocity component normal to the boundary of the Ith TCE and s is the closed trajectory of the boundary of this TCE.

The tracer concentration (such as salinity, temperature, or others) at the coastal node of the ith or jth TCE can also be either specified or calculated. The method used to include the freshwater discharge into the tracer equation is the same as those described in the TCE method.

4.4. Criterions for Horizontal Resolution and Time Step

In most coastal ocean models, the tracer concentration at grid points representing freshwater runoff is specified. This approach seems very straightforward because the tracer concentration of the runoff is easily determined by direct measurements taken at either the mouth or upstream of a river. In view of numerical computation, however, this

method is built on an assumption that no mixing occurs in the individual computational volume connected to the freshwater source. It should be generally sound in the case with sufficient horizontal resolution, but it may cause unrealistic buoyancy gradients near the source of the runoff and thus exaggerate the spatial scale and propagation speed of the low-salinity plume in the case with coarse horizontal resolution. A criterion is derived here to find a minimum horizontal resolution or time step to ensure the mass conservation in the calculation of a tracer concentration that is injected from a point source.

Considering a case with no vertical and horizontal diffusion, the vertical integral form of the salinity equation in FVCOM with river runoff is given as

$$\frac{\partial SD}{\partial t} = [-\oint v_n SDds + Q\hat{S}_o]/\Omega^{\zeta} \tag{4.15}$$

where S is the salinity at nodes of triangles connecting to the coastal node of the TCE, v_n is the velocity component normal to the boundary of the TCE, Q is the volume transport of the runoff, $\hat{S}_o$ is the salinity of the runoff water, and Ω^{ζ} is the area of the TCE containing the runoff. For simplification, let us consider a first-order forward time integration scheme. Assuming that at the n time step, the computational domain is filled with water with salinity S^n, then at the $n+1$ time step, the salinity at the node point connected to the runoff can be estimated by

$$S^{n+1} = S^n(\frac{D^n}{D^{n+1}} - \frac{\Delta t\oint v_n D^n ds}{D^{n+1}\Omega^{\zeta}}) + \frac{\Delta tQ\hat{S}_o}{D^{n+1}\Omega^{\zeta}} \quad , \tag{4.16}$$

where Δt is the internal mode time step used for numerical computation. For simplification, a forward numerical scheme is used here for time integration. Replacing the transform term in (4.16) using the continuity equation yields

$$S^{n+1} = S^n + \frac{\Delta tQ}{D^{n+1}\Omega^{\zeta}}(\hat{S}_o - S^n) \ . \tag{4.17}$$

Specifying that $\hat{S}_o = 0$ for a freshwater discharge case, (4.17) can be simplified as

$$S^{n+1} = (1 - \frac{\Delta tQ}{D^{n+1}\Omega^{\zeta}})S^n \ . \tag{4.18}$$

To ensure positivity of the tracer, (4.18) must satisfy

$$1-\frac{\Delta t Q}{D^{n+1}\Omega^{\zeta}}\geq 0 \quad \text{or} \quad D^{n+1}\geq\frac{\Delta t Q}{\Omega^{\zeta}} \ . \tag{4.19}$$

Assuming that $\Omega^{\zeta} \sim L^2$, $\Delta t \sim I_{\text{split}} L/\sqrt{gD}$, $D^{n+1} \sim D$, where L and D are the typical scales of the horizontal resolution of the TCE and water depth, then in order to ensure that the salinity at the node point connected to the runoff will remain zero, L must scale as

$$L \sim \frac{I_{\text{split}} Q}{D\sqrt{gD}} \ . \tag{4.20}$$

For given values of $Q \sim 10^3$ m^3/s, D ~ 10 m, and $I_{split} = 10$, for example, L should be of order 100 m.

In many coastal ocean numerical experiments, computer limitations frequently force L to be set at values significantly larger than that required in the criterion (4.20). This suggests that if we consider the mass conservation of the TCE connected to the runoff, the salinity calculated by (4.17) would be smaller than the salinity specified for the runoff water. On the other hand, if the horizontal resolution specified in the model is coarser than that required in (4.20), then, the model would not ensure mass conservation in the TCE that is connected to the runoff and would also exaggerate the salinity gradient near the runoff source. Subsequently, it would overestimate the intensity of the low-salinity plume and cause an unrealistically faster propagation of the plume in the downstream direction.

In addition, for given local water depth, discharge rate, and horizontal resolution, the internal time step must satisfy the following criterion in order to ensure mass conservation:

$$\Delta t \leq \frac{D^{n+1}\Omega^{\zeta}}{Q} \tag{4.21}$$

The time step must be reduced as the freshwater discharge rate increases. This criterion must be satisfied at the shallow river mouth with a large discharge rate. Reducing the

horizontal resolution is also helpful, but in the case of a very large river discharge rate, (4.21) should be checked when an internal time step is specified.

This diagnostic analysis described above is made for a simple first-order discrete scheme, which differs from the discrete scheme used in FVCOM. The required condition shown in the criterion (4.20) might be less restricted when the higher order discrete scheme is used. Recently we added the FTC method into the second-order upwind scheme, which is helpful to avoid the occurrence of the negative salinity value at the mouth of the river when the horizontal resolution is coarse.

4.5. Groundwater Input through the Bottom

Two methods for the groundwater input have been implemented in FVCOM. The first only considers the change of the salinity due to the groundwater input and no volume flux of the groundwater is added. The second includes both salt and volume transports at the bottom and also the change of the volume in the continuity equation. Brief descriptions for these two methods are given below.

4.5.1. A Simple Salt Balance Groundwater Flux Form

Assuming that the volume flux of the groundwater is an order of magnitude smaller than the total volume of the water body, we can ignore the change of the volume due to the groundwater input and only treat the groundwater input through a salt balance model. Let Q_b (units: m^3/s) be the freshwater discharge rate at the bottom; V_b be the volume of the bottom TCE with the freshwater discharge from the bottom; and Ω is the bottom area of that TCE. Our assumption is that the change of the salinity in this TCE due to freshwater input is equal to the salinity loss due to the bottom diffusion. That is

$$\frac{V_b \times S}{V_b + Q_b \times \Delta t} = \frac{V_b \times S - (K_h \frac{\partial S}{\partial z})\Big|_{z=-H} \times \Omega \times \Delta t}{V_b} \qquad (4.22)$$

where K_h is the vertical salinity diffusion coefficient (m^2/s). Considering a salinity loss over a unit time interval ($\Delta t = 1$), (4.22) can be re-written as

$$K_h \left.\frac{\partial S}{\partial z}\right|_{z=-H} = \frac{V_b \times Q_b \times S}{(V_b + Q_b) \times \Omega} \tag{4.23}$$

In the sigma coordinate, (4.23) is rewritten as

$$\left.\frac{\partial S}{\partial \sigma}\right|_{\sigma=-1} = \frac{D \times V_b \times Q_b \times S}{K_h (V_b + Q_b) \times \Omega} \ . \tag{4.24}$$

It should be pointed out here that this condition works only for the groundwater case where the discharge water is fresh, i.e., with zero salinity. In this simplification, since no volume flux is taken into account in the flux calculation, (4.24) can not be applied to simulate the water transport due to groundwater at the bottom. A volume flux at the bottom must be included for a realistic application.

4.5.2. A Complete Form of the Groundwater Input

The complete groundwater input module is built for allowing users to add both volume and salt fluxes at the bottom. This module includes the volume change in the continuity equation due to the groundwater input, and also the salt change in the salinity equation. Groundwater flux (units: m^3/s) and salinity are required to be specified at nodes where the groundwater sources are.

The groundwater module is still under the validation phase. The code has passed benchmark tests for idealized cases. This module will be released in the next version of FVCOM after it passes internal testing.

Chapter 5: Open Boundary Treatments

One of the difficulties in applying an ocean model to a coastal region is how to specify a proper open boundary condition that allows the momentum or mass to be radiated out of or flow in the computational domain. Based on a criterion of a minimum reflection rate at open boundaries, Chapman (1985) made a comprehensive evaluation of numerical schemes for open boundarys used in finite-difference models. Since the last FVCOM workshop held in June 2005, two major modifications were made to improve the open boundary treatment in FVCOM. First, we have built an open boundary radiation module that includes popular open boundary condition treatment methods. Second, we have extended our finite-volume open boundary treatment method to include subtidal forcing and flux on the open boundary. A brief description of these open boundary treatment methods is given next.

5.1. Original Setup of the Open Boundary Treatment

In the original version of FVCOM, only two types of open boundaries were considered: one is for the case with specified tidal elevation at the open boundary and the other is for the case in which the free surface elevation is unknown at the open boundary.

In the first case, the velocity at the centroid of the boundary momentum control volume (MCE) is calculated through the linear momentum equations without inclusion of vertical and horizontal diffusion terms. The tracer values (e.g., water temperature and salinity) at the nodes of individual boundary tracer control elements (TCE) are calculated through three steps (Fig. 5.1). First, we calculate the flux out of or into individual MCE's at the boundary through the continuity equation given as

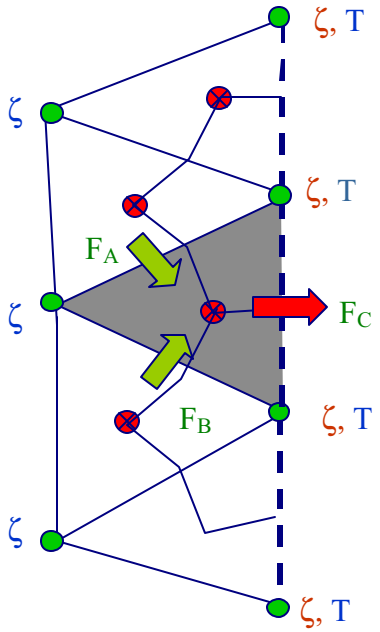


Fig.5.1: Illustration of the flux calculation at the open boundary for given tidal elevation at node points of boundary triangles.

$$F_C = \frac{\partial \zeta}{\partial t}\Omega^u - (F_A + F_B). \qquad (5.1)$$

Second, after the flux is determined, we calculate the vertically-averaged water temperature ($\overline{T}$) based on a net vertically-averaged flux through the boundary of a tracer control element (TCE).

Third, we use a gravity-wave radiation boundary condition to calculate $T' = T - \overline{T}$ at the node of individual TCE's in each layer in the vertical. The same approach can be used to calculate the boundary value for salinity (S) and other tracer values.

This method can guarantee to radiate the fastest surface gravity wave energy out of the computational domain and to ensure mass conservation throughout the water column. In a case with complex bathymetry at the boundary, a sponge layer is usually specified around this area with a damping zone weighted from the open boundary to the interior with a specified influence radius to ensure that the radiation condition will also suppress the perturbation wave energy reflected back into the computational domain from the boundary. This method works for the application of FVCOM to the Gulf of Maine/Georges Bank region and has shown to be stable for long integrations (seasonal to years).

It should be noted here that in order to reduce the error due to interpolation, we recommend that the triangles at the open boundary are constructed in the shape shown in Fig. 5.2. In this way, no interpolation from surrounding points is required when the gravity radiation boundary condition is applied. Also, in the FVCOM code, the perturbation values of T and S at the open boundary are calculated using the implicit gravity wave radiation condition described in Chen (1992). However, FVCOM is written such that users can apply other methods as desired.

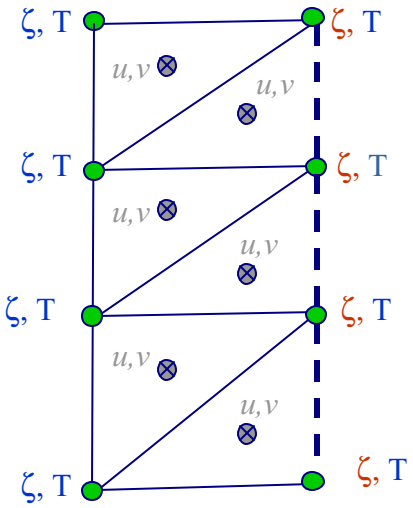


Fig. 5.2: Illustration of the recommended shapes of triangular grids at the open boundary.

In the second case, the open boundary condition treatment method is very similar to those used in the first case, except adding a gravity wave radiation condition for the free surface before the rest of the calculations are carried out. In the FVCOM code, the implicit gravity radiation condition described in Chen (1992) is used. This is consistent with the mass flux calculation used to determine the total flux through individual boundary MCE's. Also, any other radiation conditions that work for finite-difference models can be applied to FVCOM to replace the current gravity wave radiation condition.

5.2 Popular Radiation Open Boundary Conditions

Some model validation experiments have shown that the gravity wave radiation condition coded in the original version of FVCOM could cause a decrease in sea level in the entire computational domain due to wave reflection. This evidence is consistent with the analysis made by Chapman (1985), because the gravity wave radiation condition is not guaranteed to be non-reflecting in shallow water, particularly in cases with buoyancy-driven flow through the open boundary. At the present time, there is no unique radiation condition that works for all cases. To provide users more choices, we have implemented five popular open boundary conditions (OBCs) into FVCOM. In each OBC, the surface elevation at the open boundary is calculated by a radiation condition and the velocity in the open cell is computed using the linear or nonlinear momentum equations. The first method is the same as that used in the original OBC setup of FVCOM. In the second method, the vertically integrated flux at the open boundary is first computed according to the mass conservation law in the continuity equation and the vertically averaged velocity in the external mode is then calculated using the fully nonlinear momentum equations. The perturbation velocity, which is defined as the difference between the 3-D and 2-D currents, is determined using the linear momentum equations. In order to calculate the flux at the open boundary, ghost cells are added at the open boundary in which the velocity is specified as the same value and direction in the open boundary cell. In fact, the perturbation velocity is solved in the no-gradient condition. The list of these five radiation conditions are given in Table 5.1.

Table 5.1: FVCOM OBCs for Surface Elevation

Type 1	**Active (ASL)** The sea level is specified at the OB. For example, tidal amplitude and phases (original FVCOM setup)
Type 2	**Clamped (ASL-CLP) (Beardsley and Haidvogel (1981)** ζ=0 at OBC
Type 3	**Implicit Gravity Wave Radiation (GWI) (Chapman, 1985)** $\zeta_t + C_o\zeta_n = 0\,; C_o = \sqrt{gH}$, n is the normal direction to the OB.
Type 4	**Partial Clamped Gravity Wave Radiation (BKI) (Blumberg and Kantha, 1985)** $\zeta_t + C_o\zeta_n = -\zeta / T_f\,; C_o = \sqrt{gH}$ T_f : user specified frictional timescale.
	Explicit Orlanski Radiation (ORE) (Orlanski, 1976; Chapman, 1985)

Type 5	$\zeta_t + C_o\zeta_n = 0;\ C_o = \begin{cases} \dfrac{\Delta n}{\Delta t} & if \quad -\dfrac{\zeta_t}{\zeta_n} \geq \dfrac{\Delta n}{\Delta t} \\ -\dfrac{\zeta_t}{\zeta_n} & if \quad -\dfrac{\zeta_t}{\zeta_n} < \dfrac{\Delta n}{\Delta t} \\ 0 & if \quad -\dfrac{\zeta_t}{\zeta_n} \leq 0 \end{cases}$

The code was checked by running a validation experiment for a freshwater discharge case over an idealized shelf (Fig. 5.3). The computational domain is constructed with a semi-enclosed rectangular basin with a length of 800 km, a width of 400 km. The water depth is 10 m at the coast and increases to 100 m over a cross-shelf distance of

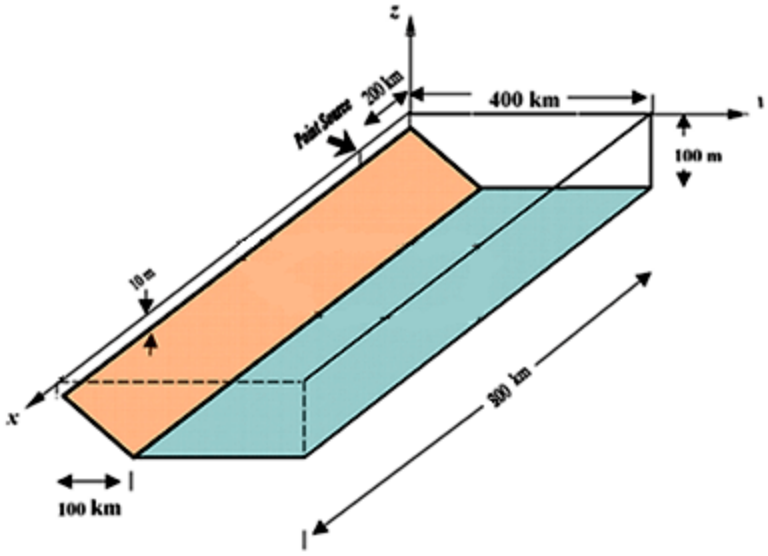


Fig. 5.3: The illustration of an idealized, straight-coastline continental shelf used for OBC case tests.

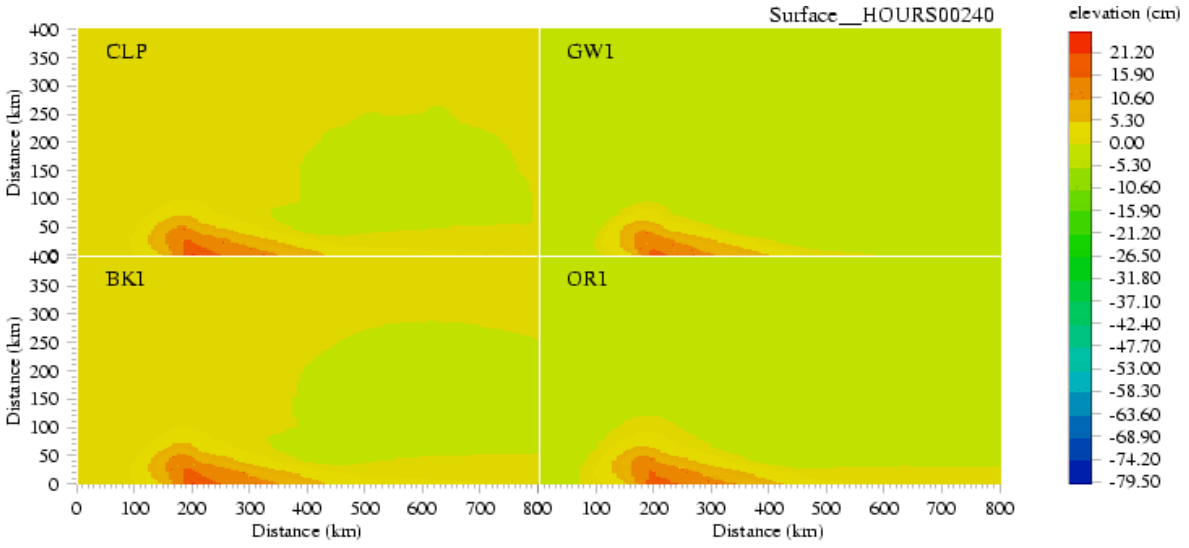


Fig. 5.4: Comparison of spatial distributions of the water elevation at the end of the 10th model day. (a) CLP; (b) GWI; (c) BKI; (d) ORI.

100 km. The model is configured with the unstructured triangular grid with an open boundary located in the downstream region at 600 km from the freshwater source. The horizontal

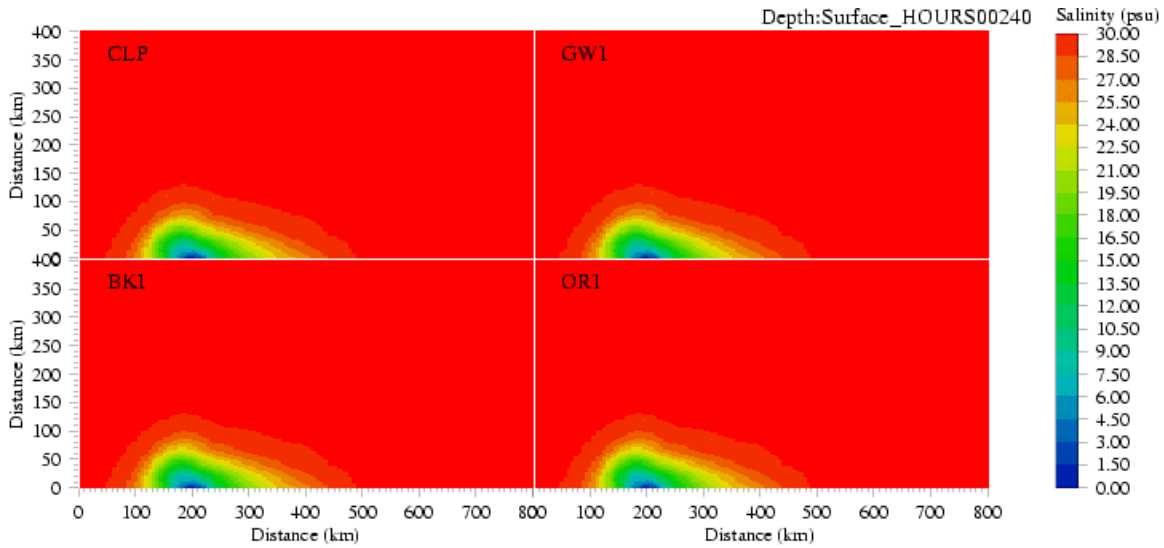


Fig. 5.5: Comparison of spatial distributions of the surface salinity at the end of the 10th model day. (a) CLP; (b) GWI; (c) BKI; (d) ORI.

resolution is ~20 km and 10 sigma levels are used in the vertical with a vertical resolution ranging from 1 m at the coast to 30 m off the shelf. Freshwater discharge rate is 1000 m^3/s. For initial conditions, a constant uniform value of salinity (30 PSU) was specified.

The results are summarized as follows:

CLP: No significant reflection was found in the first 10 days. When the plume reaches the OB, the reflection generates a flow along the OB. This artificial flow seems to have minor influence on the sea level and salinity in the interior in a time scale of 50 days. This is consistent with Chapman's (1985) finding. For a short-time model run, this condition is ok.

GWI: The distributions of velocity and salinity look very similar to those predicted by the CLP condition. However, the sea level in the entire domain drops with time. On the 50th day, the drop in sea level exceeds 0.5 m. GWI is not recommended for use in river discharge cases.

BKI: The sea level variation is the same as that predicted by CLP, but the velocity field is better. The distribution of salinity is very similar to CLP and GWI.

ORE: The distributions of velocity and salinity are very similar to those predicted by BKI. However, as with GWI, the sea level drops with time. The drop rate of the sea level is smaller than that found in GWI.

In this simple case, no significant differences were found for the choice of linear and nonlinear methods to calculate the velocity in open boundary cells. The nonlinear effects become critically important in estuarine applications when an open boundary is specified in the shallow areas that include the intertidal zone.

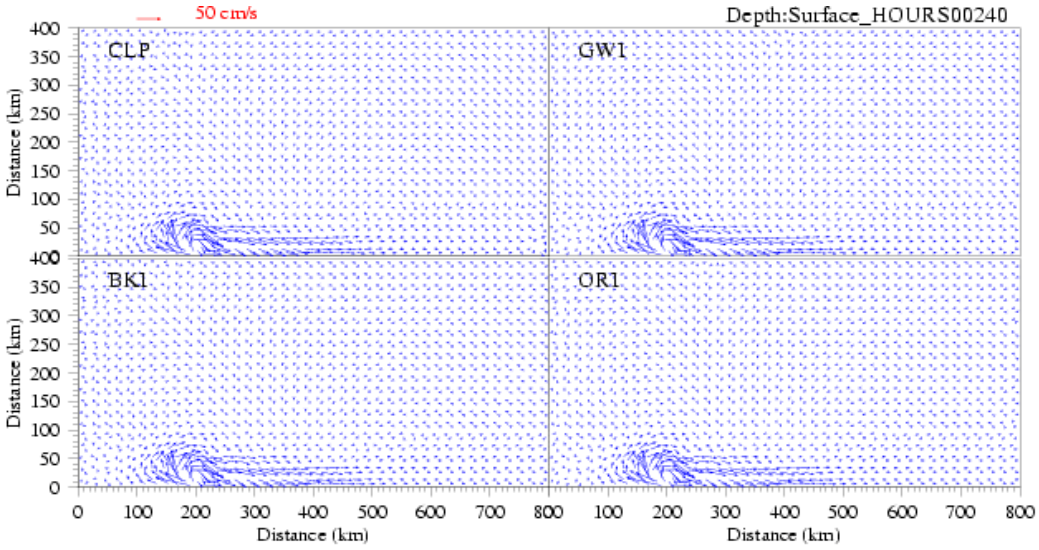


Fig. 5.6: Comparison of spatial distributions of the surface velocity at the end of the 10th day. (a) CLP; (b) GWI; (c) BKI; (d) ORI.

5.3 A New Finite-Volume Open Boundary Condition Module

By including various types of the open boundary radiation condition, FVCOM can be run for two types of numerical experiments: one is for the case with specified tidal harmonic elevations (amplitudes and phases of tidal constituents) at open boundary nodes, and the other is for the case with the free surface elevation calculated using various radiation condition at the open boundary nodes. The former is an active open boundary condition which is commonly used in coastal simulation experiments in regions with significant tidal forcing. Because the surface elevation is specified at the open boundary node, however, this method does not allow any implementation of the time-dependent volume flux and the setup of the subtidal elevation variation generated by other physical processes such as winds, precipitation minus evaporation,

and density-adjusted geostrophic flow at the open boundary. This limitation may be ok if one is interested only in the circulation in the interior, but it definitely restricts the model application to the regional ocean, where along-shelf and/or along-shelfbreak flows and fronts and western boundary currents are frequently found. To remove this limitation, a new finite-volume open boundary condition for FVCOM has been developed that allows us to include both tidal and subtidal forcing at the open boundary. A brief description of the algorithm used to derive this new open boundary condition is given next.

Assume all variables in the 2-D external mode equations can be decomposed into tidal and subtidal components as follows:

$$\begin{aligned} &\zeta = \zeta_T + \zeta', U = U_T + U', V = V_T + V' \\ &D = H + \varsigma_T + \varsigma' = D_T + \varsigma' \end{aligned} \tag{5.2}$$

where $U = \int_{-1}^{0} u d\sigma$, $V = \int_{-1}^{0} v d\sigma$, and the subscript T denotes the tidal component and apostrophe ' denotes the subtidal component. Substituting (5.2) into the continuity equation yields

$$\frac{\partial \zeta_T}{\partial t} + \frac{\partial \zeta'}{\partial t} + \frac{\partial U_T D_T}{\partial x} + \frac{\partial U_T \zeta'}{\partial x} + \frac{\partial U' D}{\partial x} + \frac{\partial V_T D_T}{\partial y} + \frac{\partial V_T \zeta'}{\partial y} + \frac{\partial V' D}{\partial y} = 0 \tag{5.3}$$

Because the tidal motion satisfies the continuity equation itself, Eq. (5.3) can be simplified to

$$\frac{\partial \varsigma'}{\partial t} + \frac{\partial U_T \varsigma'}{\partial x} + \frac{\partial U' D}{\partial x} + \frac{\partial V_T \varsigma'}{\partial y} + \frac{\partial V' D}{\partial y} = 0 \tag{5.4}$$

Integrating Eq. (5.4) over a TCE at the open boundary (see Fig. 5.7) produces

$$\iint_{\Omega_c} \frac{\partial \zeta'}{\partial t} dxdy = -[\oint_s (\vec{U}_T)_n \zeta' ds + \oint_s (\vec{U}')_n D ds] \tag{5.5}$$

where $(\vec{U}_T)_n$ and $(\vec{U}')_n$ are the tidal and subtidal vertically-integrated velocity components normal to the boundary of the triangle, respectively; Ω_c is the area of a TCE; and s is the closed trajectory comprising the boundary of a TCE.

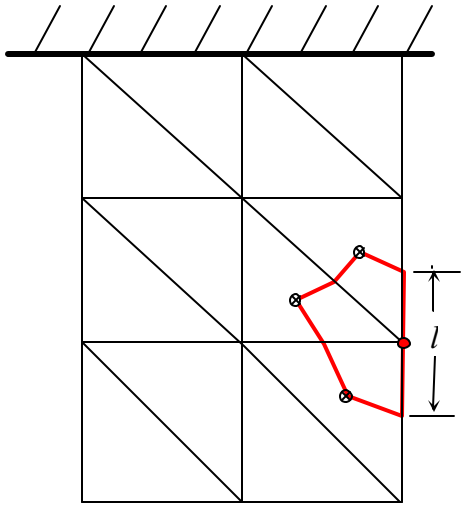


Fig. 5.7: Illustration of a TCE on the open boundary

Eq. (5.5) indicates that the subtidal surface elevation at the open boundary node is determined by the interaction between the tidal current and subtidal elevation and the net flux driven by the

vertically-integrated subtidal current. The first term can be easily determined based on the known tidal currents, while the second term is only determinable when the subtidal current at the open boundary edge is known. Once the two terms on the right hand side of Eq. 5.5 are computed, $(?')^{N+1}$ can be easily derived using the same second-order four-stage Runge-Kutta time-stepping scheme as that configured in the external mode. Total surface elevation at N+1 time step at a boundary node is then equal to

$$\zeta^{N+1} = \zeta_T^{N+1} + \zeta'^{N+1} \tag{5.6}$$

where ζ_T^{N+1} is the tidal elevation that is usually specified as the open boundary forcing and ζ'^{N+1} is the subtidal surface elevation that is determined by the mass conservative procedure.

The discrete approach to calculate the two terms on the right hand side of Eq. (5.5) is described here. The first term at the time step N can be rewritten as

$$\oint_s (\vec{U}_T^{\,N})_n \zeta' ds = \int_{s'} (\vec{U}_T^{\,N})_n \zeta' ds + \int_l (\vec{U}_T^{\,N})_n \zeta' ds \tag{5.7}$$

where s' is the trajectory of the TCE with a direct connection to the centroid of triangles of the TCE and l is the sideline length of the TCE on the open boundary. The first term on the right hand side of Eq. (5.7) can be determined numerically by

$$\begin{aligned}\int_{s'} (\vec{U}_T^{\,N})_n \zeta' ds = \sum_{m=1}^{NT} [&(\Delta X_{2m-1} (V_T^{\,N})_m - \Delta Y_{2m-1} (U_T^{\,N})_m) \cdot (\zeta'^{N})_m \\ &+ (\Delta X_{2m} (V_T^{\,N})_m - \Delta Y_{2m} (U_T^{\,N})_m) \cdot (\zeta'^{N})_m]\end{aligned} \tag{5.8}$$

where NT is the total number of the sidelines connected to centroids of triangles in the open boundary TCE. The second term on the right hand side of Eq. (5.7) can be determined numerically by

$$\oint_l (\vec{U}_T^{\,N})_n \zeta'^{N} dl = (\vec{U}_T^{\,N})_n \cdot \zeta'^{N} \cdot |\vec{l}\,| \tag{5.9}$$

where $|\vec{l}\,|$ is the length of l. $(\vec{U}_T^{\,N})_n$ on l can be calculated inversely from the continuity equation of the tidal motion as

$$(\vec{U}_T^{\,N})_n \cdot |\vec{l}\,| = -(\iint \frac{\partial \zeta_T}{\partial t} dxdy + TF) / D_T^{\,N} \tag{5.10}$$

and

$$TF = \sum_{m=1}^{NT}[(\Delta X_{2m-1}(V_T{}^N)_m - \Delta Y_{2m-1}(U_T{}^N)_m)\cdot(D_T{}^N)_m + (\Delta X_{2m}(V_T{}^N)_m - \Delta Y_{2m}(U_T{}^N)_m)\cdot(D_T{}^N)_m] \quad (5.11)$$

The second term on the right hand side of Eq. (5.5) can be rewritten in discrete form as

$$\oint_s (\vec{U}'^N)_n D ds = \oint_{s'} (\vec{U}'^N)_n D ds + \oint_l (\vec{U}'^N)_n D^N dl \quad (5.12)$$

where

$$\oint_{s'} (\vec{U}'^N)_n D ds = \sum_{m=1}^{NT}[(\Delta X_{2m-1}(V'^N)_m - \Delta Y_{2m-1}(U'^N)_m)\cdot(D^N)_m + (\Delta X_{2m}(V'^N)_m - \Delta Y_{2m}(U'^N)_m)\cdot(D^N)_m] \quad (5.13)$$

and

$$\oint_l (\vec{U}'^N)_n D^N dl = (\vec{U}'^N)_n D^N \cdot |\vec{l}\,|. \quad (5.14)$$

$(\vec{U}'^N)_n$ on l can be determined by two methods: one is to use a radiation boundary condition to calculate $(\vec{U}'^N)_n$ on the boundary and the other is to specify the subtidal flux on the boundary. A default setup for these two methods is described below.

I) BKI radiation condition

The Blumberg and Kantha implicit gravity wave radiation condition is chosen as the default setup for the first method. Let $\hat{U}_B$ be the value $(\vec{U}'^N)_n$ on the boundary, then

$$\hat{U}_B{}^{N+1} = [(1-\frac{\Delta t}{T_f})\hat{U}_B{}^N + c\frac{\Delta t}{\Delta n}\hat{U}_{B-1}{}^{N+1}]/(1+c\frac{\Delta t}{\Delta n}) \quad (5.15)$$

where B is a boundary point and B-1 is the interior point next to the boundary point; ?n is the distance between the interior and boundary point. This is a passive open boundary condition.

II) Flux condition

In many coastal applications, the subtidal flow field at the open boundary can be determined using either observational data or outputs from regional or basin or global ocean models. For example, to run an East China Sea regional model, one must specify the volume flux of the Kuroshio where it enters the model domain through the open boundary. In this case, $(\vec{U}'^N)_n$ on l can be determined using the value specified by the user. The volume flux produced by the

subtidal current on the boundary can be specified by the same procedure used to add the river discharge on the boundary edge. For details, please refer to Chapter 4. We call this method an active open boundary condition, because it is controlled by the user.

In the case without tides, this boundary treatment procedure is like a traditional radiation condition and the model can be run directly with this condition. In the case with both tidal and subtidal forcings, the model needs to run with only tidal forcing first, so that the tidal currents at the centroid and at nodes of open boundary cells can be determined. These data are automatically output into a file as the input file for the model run with combined tidal and other forcing.

When a radiation boundary condition is specified for the subtidal current at the open boundary, a sponge layer might be required to filter the high frequency numerical noise due to wave reflection at the open boundary.

This method has been tested by running FVCOM with this condition for idealized cases and applications to the real ocean. The first experiment was to re-run the river discharge plume case shown in Fig. 5.3. The results are identical to that found in the original BKI radiation condition simulation. The results of three other idealized test cases are presented next.

a) Channel Flow

This test case was used by the USGS Woods Hole Field Center to test their sediment module "SED_TEST1" in ROMS. Detailed description of the configuration can be found at http:// woodshole.er.usgs.gov/project-pages/sediment-transport/Test_Case_1.htm. A brief description is

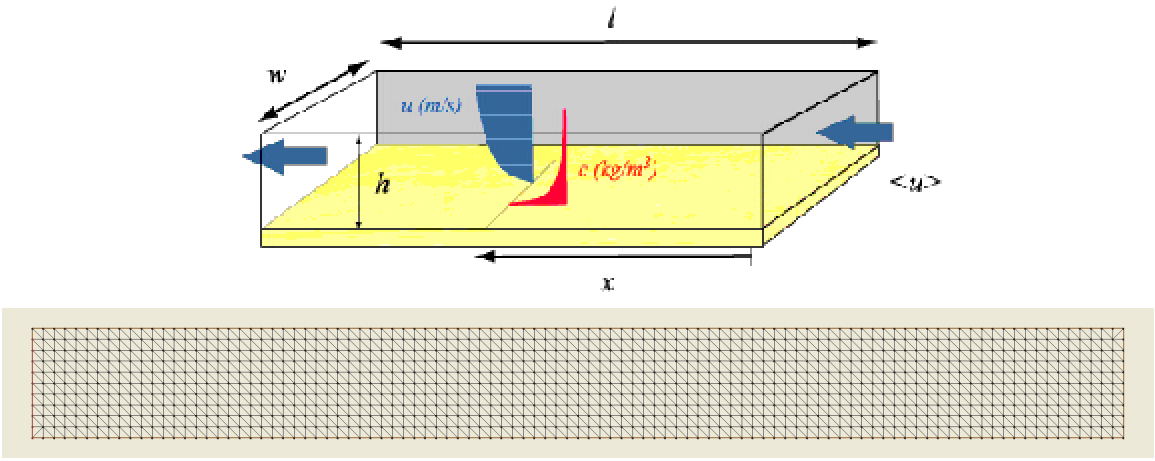


Fig. 5.8: (a) Schematic of channel flow test case. (b) Model grid used

repeated here.

The model domain is a long, narrow rectangular box with length l = 10,000 m, width w = 1000 m, and uniform depth h = 10 m (Fig. 5.8a. This figure is taken directly from the USGS website). Fresh water with T = 20°C, S = 0 ppt flows in from the eastern open boundary and flows out from the western open boundary. There is no rotation and no surface wind stress and heat flux forcing. Inflow is maintained as a steady flow with volume transport of 10 m^3/s/m of width and uniformly distributed in the vertical. Northern and southern sides of the box are solid boundaries. Bottom roughness z_0 = 0.005.

The computational domain is configured with an uniform triangular grid in the horizontal and sigma layers in the vertical. Horizontal resolution is 100 m and vertical resolution is 1 m (with 10 sigma layers). The fluid is initially at rest, and the model spins up with a ramping period of 4 hrs.

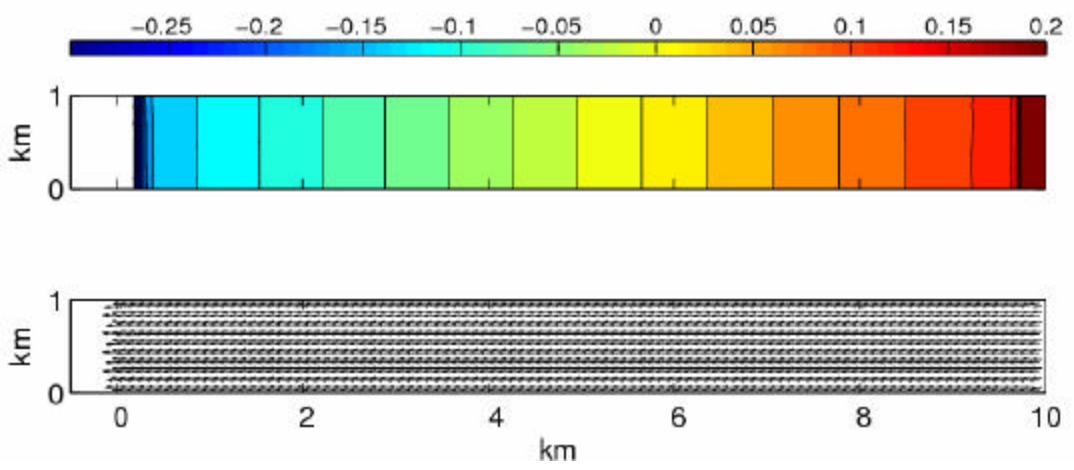


Fig. 5.9: (a) Sea surface elevation (m) and (b) surface layer velocity vectors at t = 200 hrs. A sponge layer is added at the outflow open boundary.

The model runs stably with inflow and outflow conditions. Fig. 5.9 shows the sea surface elevation and velocity vectors in the surface layer at the 200th hour. A sponge layer, with a friction coefficient increased from 0 to 0.05 at the open boundary over a distance of r = 400 m, is used to absorb disturbances and suppress computational noise at the outflow side. The sponge layer only affects the numerical solution near the boundary with no influence on the interior solution.

b) Flow in A Sloping-Bottom Channel.

This is a case used by investigators at the USGS Woods Hole Field Center to test their estuarine module "ESTUARY_TEST" in ROMS. Detailed information about the experimental

design can be viewed at http://woodshole.er.usgs.gov/project-pages/sediment-transport/Test_Case_2.htm. A brief description is given below.

The model domain is a long, narrow rectangular channel with length (east-west) l = 100,000 m, width (north-south) w = 10,000 m, and depth changing linearly from h = 10 m at the western end to 5 m at the eastern end. A constant inflow transport of 5 $m^3/s/m$ of width is specified at the eastern end and the same amount of outflow transport is specified at the western end. The M_2 tidal forcing with an amplitude of 10 cm is specified at both open boundaries. The experiment was made for a barotropic case in which salinity and temperature are constant. No rotation, no wind and no heating/cooling are included. Both lateral sides are treated as solid walls. The model was run for the fully nonlinear case with bottom roughness z_0 = 0.005.

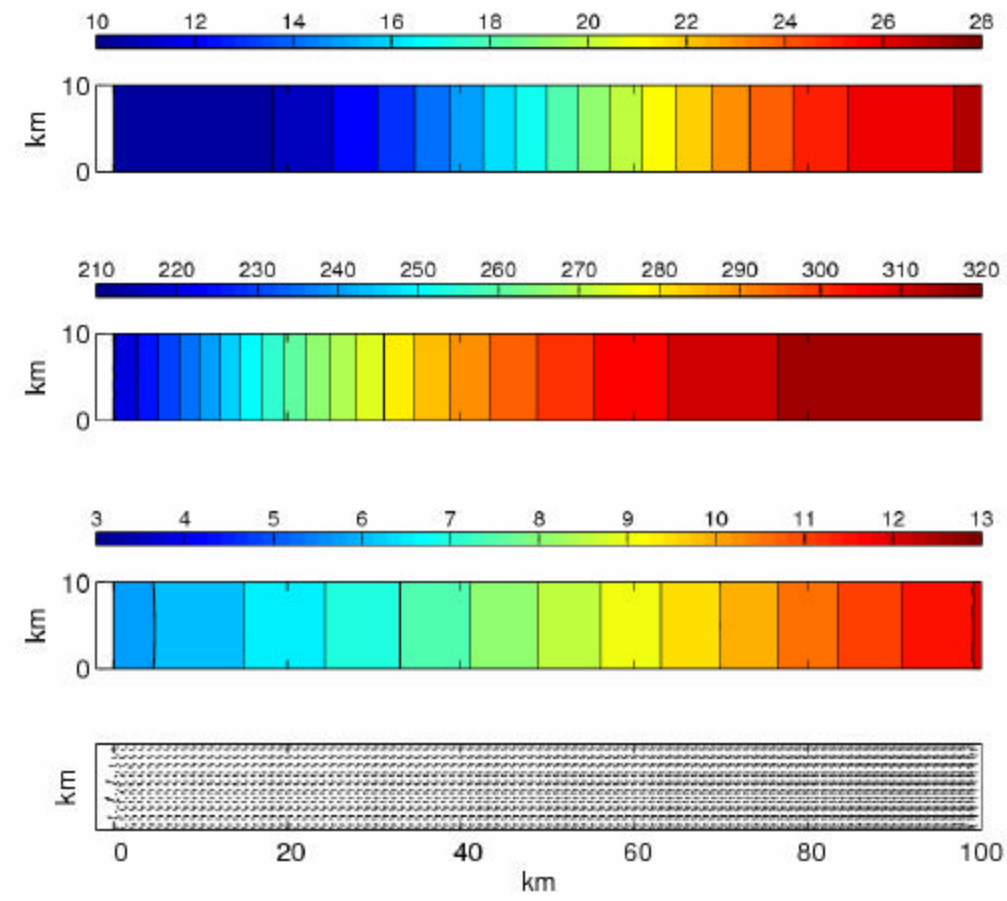


Fig. 5.10: (a) M2 amplitude (in cm), (b) M2 phase (°), (c) Z_0 (in cm), and (d) surface layer residual velocity vector for the estuary test case at 30 days.

Simulation results are shown in Fig. 5.10. The predicted amplitude and phase of M_2 elevation (Fig. 5.10a-b) are physically realistic and closely resemble those found in the case with only tidal

forcing (figures are not shown here). The residual water elevation Z_o (Fig. 5.10c) and residual surface velocity vector (Fig. 5.10d) are also quite similar to those found in the case with only inflow and outflow transport specified (figures are not shown here).

c) Flow over An Idealized Continental Shelf

Consider an idealized continental shelf constructed by a zonal channel that is 600 km long and 200 km wide (Fig. 5.11). The bathymetry is constant in the along-shelf direction and has a hyperbolic tangent profile in the cross-shelf direction. The coastline is located on the northern side which is treated as a solid boundary and all other sides are open boundaries. Assume that the tide propagates shoreward from the open ocean, so that tidal forcing is only specified at the offshore open boundary. An along-shelf

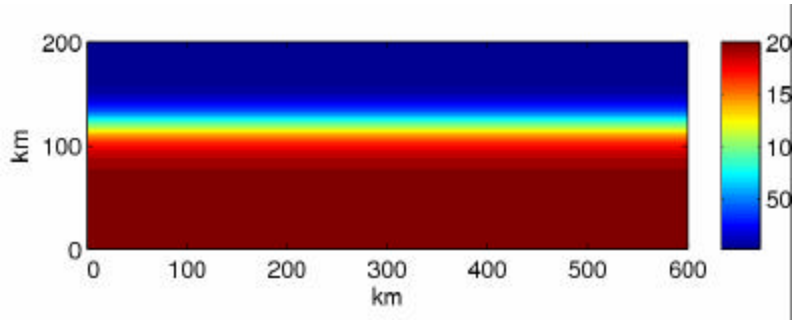


Fig. 5.11: Distribution of bottom depth (in m) in the continental shelf case.

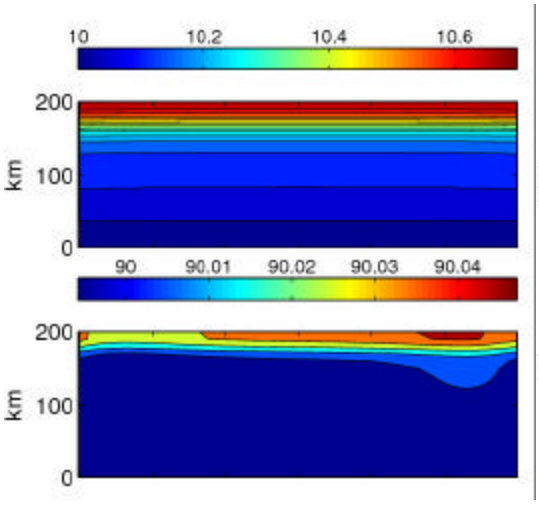


Fig. 5.12: Distributions of the M_2 amplitude (in cm) (upper) and phase (°) for the case with only tide forcing.

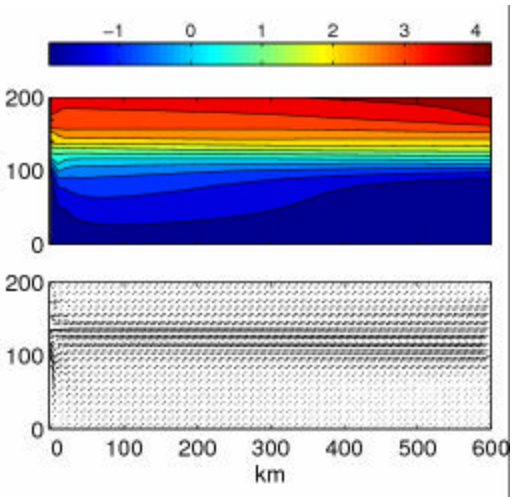


Fig. 5.13: Distribution of the mean surface elevation (in cm) (upper), and surface layer residual velocity vector for the continental shelf case with only mean flow forcing.

current with a volume transport of 7.35 Sv is prescribed at the shelf break on the upstream open boundary and the same amount of the transport is also specified at the downstream open

boundary. The model experiment was carried out for a barotropic case in which the water temperature and salinity remain constant in time.

The computational domain is configured with a uniform triangular grid with horizontal resolution of 10 km. A total of 10 sigma layers are used in the vertical. The model was spin up over a ramping period of 24 hrs for the case with only tidal forcing and 48 hrs for the case with both tidal and subtidal forcing.

Figs. 5.12 and 5.13 show the model-predicted amplitude and phase of the M_2 elevation and the sea surface elevation and current for the case with only tidal forcing and for the case with only inflow and outflow transport, respectively. Fig. 5.14 shows the model results for these variables for the case with both tidal forcing and inflow/outflow transports. The basic patterns

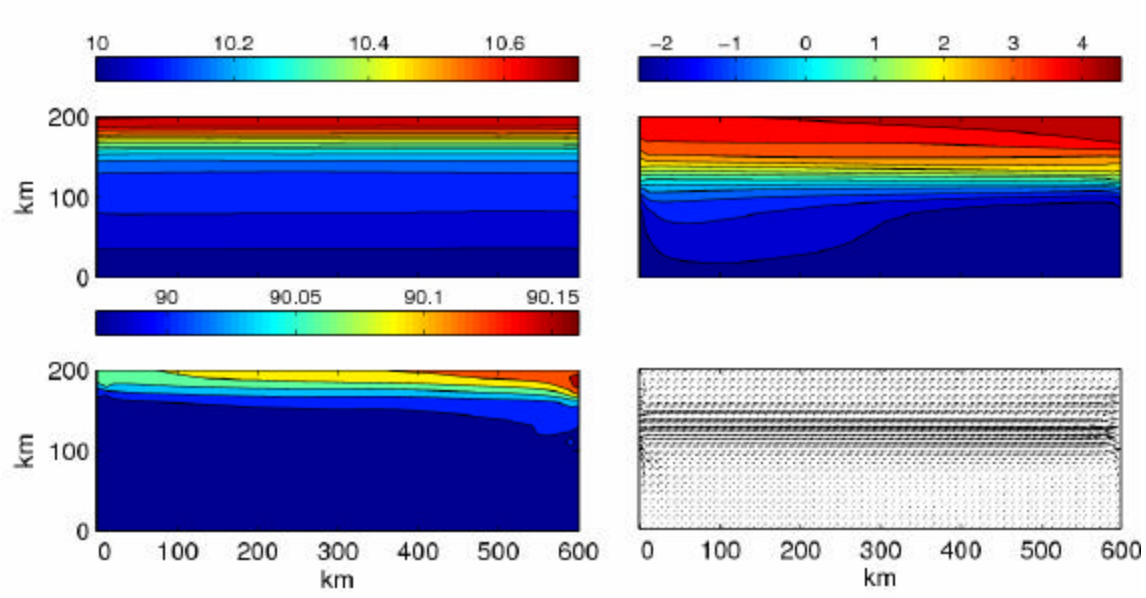


Fig. 5.14: Left panel: distributions of the M_2 amplitude (in cm) (upper) and phase (°) (lower). Right panel: distributions of (in cm) (upper) and surface layer residual velocity vector (cm/s) (lower) for the continental shelf test case forced by tidal and subtidal flow flux at the open boundary.

are very similar to the model run with separate boundary forcing. The difference of M_2 elevation phase near the eastern boundary between Fig. 5.14 and Fig.5.12 is probably due to the time series interpolation (used to force the model) or the nonlinear interaction between tidal waves and mean flow.

Chapter 6: Data Assimilation Methods

The most widely used techniques for data assimilation are (a) nudging, (b) optimal interpolation (OI), (c) variational methods (most notably adjoint methods) and (d) Kalman filtering (KF). Nudging, the most basic method, is used in MM5 to merge model-predicted values of physical variables directly to observations given *a priori* statistical assumptions about the model noise and errors in the observational data. The OI method uses the error covariance of the observations and model predictions to find their most likely linear combination (Lorenc, 1981). Similar to nudging, OI requires *a priori* statistical assumptions about the model noise and observational errors. Variational methods are based on control theory, in which a cost function, defined by the difference between model-derived and measured quantities, is minimized in a least-square sense under the constraint that the governing equations of the model remain satisfied (Le Dimet and Talagrand, 1986; Thacker and Long, 1988; Tziperman and Thacker, 1989; Bergamasco et al., 1993; Morrow and De Mey, 1995). In this framework, an adjoint system to the model can be constructed which can directly yield the sensitivity of this cost function to the specified control variables which may be the model initial or boundary conditions. This adjoint can be derived directly from the model's governing equations, however, in practice it is more consistent to use an adjoint derived from the discrete representation of the governing equations in the model. This discrete system adjoint can be constructed directly from the model source code using automatic differentiation (AD) methods that are being developed by C. Wunsch's group at MIT.

Kalman Filters are the most sophisticated statistical approaches, and are commonly used for nowcasting/forecasting of ocean and atmospheric models (Evensen, 1992, 1993; Blanchet, 1997; Ghil and Malanotte-Rizzoli, 1991). Dr. Malanotte-Rizzoli and her collaborators have constructed a series of assimilation packages first applied to idealized models as proof-of-concept tests and successively applied to fully realistic, primitive equations models. They include a Reduced Rank Kalman filter (RRKF) (Buehner and Malanotte-Rizzoli, 2003; Buehner et al., 2003), an Ensemble Kalman Filter (EnKF) (Zang and Malanotte-Rizzoli, 2003), deterministic and stochastic ensemble Kalman filters (Lyu et al, 2005), and an Ensemble Transform Kalman Filter (ETKF) (Lyu and Malanotte-Rizzoli, 2005). The EnKF has shown advantages in dealing with strongly

nonlinear systems which, we believe, is most relevant to the coastal ocean. The ETKF was introduced by Bishop et al. (2001) for optimally deploying adaptive observations and is presently the most used approach in meteorology. Lyu and Malanotte-Rizzoli have applied both RRKF and ETKF to design optimal fixed and adaptive observational arrays in an idealized model of the wind-driven circulation in a double gyre ocean, with the final objective the extension to a fully realistic Ocean General Circulation Model. Kalman Filters could be used to design optimal observational arrays which give the minimum trace of the forecast error covariance over the region of interest. In particular, the filters could be used to conduct sensitivity studies of the effectiveness of the optimal fixed/adaptive network to improve model forecasts with respect to a) number and type of observations; b) targeting different regions characterized by different dynamics and energetics; c) duration of the targeted (adaptive) lead time corresponding to successive assimilations; and d) ensemble size in a filter among other factors.

Several data assimilation modules are being developed for FVCOM. They include 3-D nudging, OI and a suite of Kalman Filters. FVCOM currently has data-assimilation capabilities for hindcast skill improvement which utilize the 3-D nudging and Kalman Filters. The nudging method is fast and practical for forecast applications, but it is prone to causing unphysical behavior if the data coverage is too coarse or the associated parameters are not properly set. OI requires the covariance of variables, which are usually hard to estimate due to the scarcity of required data. Recently we have built a covariance map of SST for the Gulf of Maine, which can be used to apply OI for SST assimilation. A group of investigators led by C. Wunsch at MIT are developing a new FORTRAN 95 automatic differentiation (AD) tool for community use. We plan to work with them to use their new AD tool to generate an adjoint model of FVCOM for use in both data assimilation and sensitivity analysis. By collaborating with P. Malanotte-Rizzoli at MIT, we have implemented a Kalman Filter module into FVCOM. This set includes 1) Reduced Rank Kalman Filter (RRKF), 2) Ensemble Kalman Filter (EnKF) and 3) Ensemble Transform Kalman Filter for hind- and now-casting applications. This module has been tested for idealized river discharge, tidal wave, and estuarine flooding/drying process cases. It is being tested presently for realistic data assimilation in

the Gulf of Maine. Users who are interested in using this filter module will have access once it is fully tested.

A brief description of Nudging, OI and Kalman Filters is given here. The Kalman Filters involve complex mathematic equations and the details of these equations were given in a set of published papers by Rizzoli and her collaborators. Flow charts of the filters are presented here to tell users how they are implemented and work in FVCOM.

6.1. The Nudging Method

Let $\alpha(x, y, z, t)$ be a variable selected to be assimilated and $F(\alpha, x, y, z, t)$ represents the sum of all the terms in the governing equation of $\alpha(x, y, z, t)$ except the local temporal change term, then the governing equation of $\alpha(x, y, z, t)$ with inclusion of nudging assimilation is given as

$$\frac{\partial \alpha(x, y, z, t)}{\partial t} = F(\alpha, x, y, z, t) + G_\alpha \frac{\sum_{i=1}^{N} W_i^2(x, y, z, t)\gamma_i(\alpha_o - \hat{\alpha})_i}{\sum_{i=1}^{N} W_i(x, y, z, t)} \tag{6.1}$$

where α_o is the observed value; $\hat{\alpha}$ is the model-predicted value; N is the number of observational points within the search area; γ_i is the data quality factor at the ith observational point with a range from 0 to 1; and G_α is a nudging factor that keeps the nudging term to be scaled by the slowest physical adjustment process. The selection of G_α must satisfy the numerical stability criterion given by

$$G_\alpha < \frac{1}{\Delta t} \tag{6.2}$$

Normally, G_α is set to approximately the magnitude of the Coriolis parameter. $W_i(x, y, z, t)$ is a product of weight functions given as

$$W_i(x, y, z, t) = w_{xy} \cdot w_\sigma \cdot w_t \cdot w_\theta \tag{6.3}$$

where w_{xy}, w_σ, w_t, and w_θ are horizontal, vertical, temporal and directional weighting functions, respectively. The mathematical expressions of these functions are given as

$$w_{xy} = \begin{cases} \dfrac{R^2 - \hat{r}^2}{R^2 + \hat{r}^2}, & 0 \le \hat{r} \le R \\ 0, & \hat{r} > R \end{cases} \tag{6.4}$$

$$w_{\sigma} = \begin{cases} 1 - \dfrac{\left|\sigma_{obs} - \sigma\right|}{R_{\sigma}}, & \left|\sigma_{obs} - \sigma\right| \le R_{\sigma} \\ 0, & \left|\sigma_{obs} - \sigma\right| > R_{\sigma} \end{cases} \tag{6.5}$$

$$w_t = \begin{cases} 1, & \left|t - t_o\right| < T_w / 2 \\ \dfrac{T_w - \left|t - t_o\right|}{T_w / 2}, & T_w / 2 \le \left|t - t_o\right| \le T_w \\ 0, & \left|t - t_o\right| > T_w \end{cases} \tag{6.6}$$

$$w_{\theta} = \frac{\left|\left|\Delta\theta\right| - 0.5\pi\right| + c_1\pi}{(0.5 + c_1)\pi} \tag{6.7}$$

where R is the search radius; $\hat{r}$ is the distance from the location where the data exists; R_{σ} is the vertical search range; T_w is one-half the assimilation time window; and $\Delta\theta$ is the directional difference between the local isobath and the computational point with c_1 a constant ranging from 0.05 to 0.5.

6.2 The OI Method

Optimal interpolation (OI) is an alternative simple data assimilation method similar to the nudging method. A detailed comparison of OI with other data assimilation methods was given in detail in Kantha and Clayson (2000) and a brief description of this scheme is repeated here for users who are not familiar with this scheme.

Let X_f, X_a and X_o be the model forecast, assimilated (analysis) and observed values of a model variable X, respectively, and assume that they satisfy a linear relationship given as

$$X_a = X_f + \sum_{k=1}^{M} a_k (X_{o,k} - X_{f,k}) \tag{6.8}$$

where M is the total data points involved in the optimal interpolation for X at an node point. Defining that the true value of X is X_T at the assimilated node and $X_{T,k}$ at the

kth observed point, and $e_a = X_a - X_T$; $e_f = X_f - X_T$; $e_{o,k} = X_{o,k} - X_{T,k}$; and $e_{f,k} = X_{f,k} - X_{T,k}$, the analysis error e_a is equal to

$$e_a = e_f + \sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k}) \tag{6.9}$$

The analysis error covariance $P_a = e_a^2$, which is given as

$$\begin{aligned} P_a &= [e_f + \sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k})][e_f + \sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k}) \\ &= e_f^2 + 2\sum_{k=1}^{M} a_k (e_f e_{o,k} - e_f e_{f,k}) + [\sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k})]^2 \end{aligned} . \tag{6.10}$$

In the least square fitting method, the error in e_a must be a minimum when the first differentiation condition of $\partial P_a / \partial a_k = 0$ is satisfied, i.e.,

$$\begin{cases} (e_{o,1} - e_{f,1}) \sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k}) = (e_f e_{f,1} - e_f e_{o,1}) \\ (e_{o,1} - e_{f,2}) \sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k}) = (e_f e_{f,2} - e_f e_{o,2}) \\ \vdots \\ \vdots \\ (e_{o,M} - e_{f,M}) \sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k}) = (e_f e_{f,M} - e_f e_{o,M}) \end{cases} \tag{6.11}$$

Assuming that $e_{f,k}$ is not correlated with $e_{o,k}$, then (6.11) can be simplified to

$$\begin{cases} \sum_{k=1}^{M} a_k (e_{o,k} e_{o,1} + e_{f,k} e_{f,1}) = e_f e_{f,1} \\ \sum_{k=1}^{M} a_k (e_{o,k} e_{o,2} + e_{f,k} e_{f,2}) = e_f e_{f,2} \\ \vdots \\ \vdots \\ \sum_{k=1}^{M} a_k (e_{o,k} e_{o,M} + e_{f,k} e_{f,M}) = e_f e_{f,M} \end{cases} \tag{6.12}$$

or

$$\begin{cases} (e_{o,1}^2 + e_{f,1}^2)a_1 + (e_{o,2}e_{0,1} + e_{f,2}e_{f,1})a_2 \cdots + (e_{o,M}e_{o,1} + e_{f,M}e_{f,1})a_M = e_f e_{f,1} \\ (e_{o,1}e_{o,2} + e_{f,1}e_{f,2})a_1 + (e_{o,2}^2 + e_{f,2}^2)a_2 \cdots + (e_{o,M}e_{o,2} + e_{f,M}e_{f,2})a_M = e_f e_{f,2} \\ \vdots \\ \vdots \\ (e_{o,1}e_{o,2} + e_{f,1}e_{f,2})a_1 + (e_{o,2}e_{o,M} + e_{f,2}e_{f,M})a_2 \cdots + (e_{o,M}^2 + e_{f,M}^2)a_M = e_f e_{f,M} \end{cases} \quad (6.13)$$

This can be written in matrix form as

$$\hat{P} \cdot \hat{a} = \hat{f} \quad (6.14)$$

where

$$\hat{P} = \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1M} \\ P_{21} & P_{22} & \cdots & P_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ P_{M1} & P_{M2} & \cdots & P_{MM} \end{pmatrix}; \ \hat{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{pmatrix}; \ \hat{f} = \begin{pmatrix} e_f e_{f,1} \\ e_f e_{f,2} \\ \vdots \\ e_f e_{f,M} \end{pmatrix} \quad (6.15)$$

and

$$P_{i,k} = e_{o,i}e_{o,k} + e_{f,i}e_{f,k}, \ i = 1, 2, \ldots.\text{M}; \ k = 1,2,\ldots..\text{M}. \quad (6.16)$$

When the observational and forecast error covariance values are known or specified, parameter a_k can be determined by using a state-of-the-art linear algebraic equation solver to solve (6.14).

In real applications, for simplification, we can assume that the observational errors are zero and the forecast error covariance satisfies a normal distribution given by

$$P_{ik} = e^{-(\frac{r_{ik}}{d})^2} \quad (6.17)$$

where r_{ik} is the horizontal distance between i and k points and d is the correlation radius. With this approach, the OI scheme should be very similar to the nudging data assimilation scheme.

The nudging and OI data assimilation methods are practical approaches for the purpose of model application to the real-time simulation and assimilation. However, they lack rigorous scientific support and are not generally useful for sensitivity studies of model parameters.

6.3. The Kalman Filters

Let $\boldsymbol{x}^t$ be an array of the true values, $\boldsymbol{x}^f$ be an array of the forecast values, $\boldsymbol{x}^a$ an array of analysis values, and $\boldsymbol{y}$ an array of observational values. We can define that

Analysis error: $$\boldsymbol{e}_a = \boldsymbol{x}^a - \boldsymbol{x}^f \tag{6.18}$$

Forecast error: $$\boldsymbol{e}_f = \boldsymbol{x}^t - \boldsymbol{x}^f \tag{6.19}$$

Observational error: $$\boldsymbol{e}_o = \boldsymbol{y} - \boldsymbol{x}^f \tag{6.20}$$

The forecast error covariance $\boldsymbol{P}$ and the observational error covariance $\boldsymbol{R}$ thereafter can be defined as

$$\boldsymbol{P}^f = \overline{\boldsymbol{e}_f \boldsymbol{e}_f^T}, \quad \boldsymbol{R} = \overline{\boldsymbol{e}_o \boldsymbol{e}_o^T} \tag{6.21}$$

In a forecast model, the value of $\boldsymbol{x}^f$ at time step i can be predicted by

$$\boldsymbol{x}^f(i) = \boldsymbol{M}_{i-1 \to i}[\boldsymbol{x}^a(i-1)] \tag{6.22}$$

where $\boldsymbol{M}$ presents the nonlinear model operator. The forecast error covariance is equal to

$$\boldsymbol{P}^f = \boldsymbol{M}_{i-1 \to i} \boldsymbol{P}^a (i-1) \boldsymbol{M}_{i-1 \to i}^T + \boldsymbol{Q}(i-1) \tag{6.23}$$

where $\boldsymbol{Q}(i-1)$ is the system error covariance matrix. In the Kalman filter forecast model system, the analysis values $\boldsymbol{x}^a$ is calculated by

$$\boldsymbol{x}^a(i) = \boldsymbol{x}^f(i) + \boldsymbol{K}(i)[\boldsymbol{y}(i) - \boldsymbol{H}\,\boldsymbol{x}^f(i)] \tag{6.24}$$

where $\boldsymbol{K}$ is the Kalman gain matrix, which is equal to

$$\boldsymbol{K}(i) = \boldsymbol{P}^f(i)\boldsymbol{H}^T[\boldsymbol{H}\boldsymbol{P}^f(i)\boldsymbol{H}^T + \boldsymbol{R}(i)]^{-1} \tag{6.25}$$

and $\boldsymbol{H}$ is an observation operator that functions as an objective map to interpolate the model data onto the observational points. The analysis error covariance is given as

$$\boldsymbol{P}^a(i) = [\boldsymbol{I} - \boldsymbol{K}(\mathrm{i})]\,\boldsymbol{P}^f(i). \tag{6.26}$$

In general, the size of the covariance matrix is huge. For example, $P^f : [\boldsymbol{N} \times \boldsymbol{N}]$ and $M : [\boldsymbol{N} \times \boldsymbol{N}]$, and N can be of $O(10^6 - 10^7)$, which makes it impractical to use this method on most computers. For this reason, a family of Kalman Filters (Reduced Rank Kalman Filter, Ensemble Kalman Filter, Ensemble Square Root Kalman Filter and Ensemble Transform Kalman Filter) have been developed that require less computational

power than the original filter. A brief description of these other Kalman Filters and how they are implemented in FVCOM is given below.

6.3.1 Reduced Rank Kalman Filter (RRKF)

Let $\boldsymbol{x}^f$ be an array with a dimension of N (FVCOM output and RRKF input); $\boldsymbol{x}^a$ an array with a dimension of N (that are output from RRKF to use to refresh the initial conditions for FVCOM input); $\boldsymbol{y}$ an array with a dimension of $\boldsymbol{N}_o$ (that is an input for

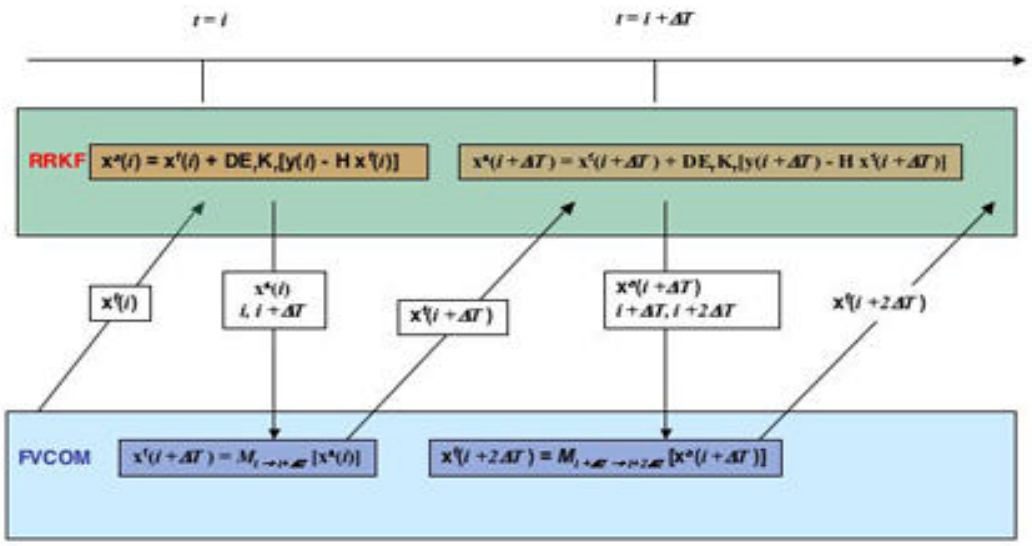


Fig. 6.1: Flow chart of RRKF in FVCOM.

RRKF); $\boldsymbol{E}_r$ the resolved empirical orthogonal functions (EOFs) with dimensions of $N \times \boldsymbol{N}_e$ (the stationary input of RRKF); $\boldsymbol{K}_r$ the stationary Kalman gain in the reduced space of $\boldsymbol{N}_e \times \boldsymbol{N}_o$] (the stationary input of RRKF (stationary input; $\boldsymbol{N}_e$ the dimension of the EOFs subspace; $\boldsymbol{i}$ and $\boldsymbol{i}+\Delta\boldsymbol{T}$ two subsequent assimilation time steps of FVCOM; $\Delta\boldsymbol{T}$ the assimilation interval; $\boldsymbol{M_i}$ the forward FVCOM model with initial conditions specified by the analysis solution; $\boldsymbol{H}$ an observation operator that functions as an objective map to interpolate the model data onto the observational points; and $\boldsymbol{D}$ is the spatially averaged

standard deviation of each variable. The flow chart of RRKF in FVCOM is illustrated in Fig. 6.1.

A detailed description of RRKF was given in Buehner and Malanotte-Rizzoli (2003). S. Lyn, who worked with P. Rizzoli as a postdoctoral investigator at MIT, helped us implement RRKF into FVCOM. He worked together with P. Xue, Q. Xu and Z. Lai in testing the RRKF code in idealized cases.

The RRKF is developed based on linear theory. In a linear system, if the observational network, observational and model error covariances are stationary and all neutrally stable and unstable modes are measurable, the forecast error covariance reaches an asymptotically stationary result and then a stationary Kalman gain ($\boldsymbol{K_r}$) can be derived efficiently by the doubling algorithm (Anderson and Moore 1979). This stationary $\boldsymbol{K_r}$ is calculated from the control model run and it is applied in the data assimilation process of the RRKF.

The RRKF in FVCOM is operated following the procedure given below:

Step 1: Determination the number of resolved EOFs (E_r) from the control model run:

$$\boldsymbol{D}^{-1}\hat{\boldsymbol{X}}_f\hat{\boldsymbol{X}}_f^{T}\boldsymbol{D}^{-1}=\boldsymbol{E}\Lambda\boldsymbol{E}^{T} \tag{6.18}$$

where $\hat{\boldsymbol{X}}_f=\boldsymbol{x}^f-\bar{\boldsymbol{x}}^f$, $\bar{\boldsymbol{x}}^f$ the mean of $\boldsymbol{x}^f$, E the EOF matrix; and Λ is the diagonal covariance eigenvalue matrix.

Step 2: Linearization of the model in the resolved EOF subspace (build $\boldsymbol{M}_r$ from $\boldsymbol{E}_r$):

$$\boldsymbol{M}_{r\,i}=\frac{1}{\alpha}\boldsymbol{E}_r^{T}\boldsymbol{D}^{-1}[\boldsymbol{M}(\boldsymbol{x}_0+\alpha\boldsymbol{D}\boldsymbol{e}_i)-\boldsymbol{M}(\boldsymbol{x}_0)] \tag{6.19}$$

where $\boldsymbol{M}$ presents the nonlinear model; subscripts "*r*" and "*i*" of $\boldsymbol{M}$ denote the linearized model in the resolved subspace and the *i*th column of $\boldsymbol{M}_r$; α is the perturbation size; $\boldsymbol{e}_i$ the *i*th retained EOF; and $\boldsymbol{x}_0$ is the specified time mean of a long model run without assimilation.

Step 3: Projection of the error covariance into the resolved EOFs subspace and estimation of the model and observation errors:

$$\boldsymbol{P}_r^f=\boldsymbol{E}_r^T\boldsymbol{P}^f\boldsymbol{E}_r;\ \boldsymbol{P}_r^a=\boldsymbol{E}_r^T\boldsymbol{P}^{\alpha}\boldsymbol{E}_r;\ \boldsymbol{M}_r=\boldsymbol{E}_r^T\boldsymbol{M}\boldsymbol{E}_r;\boldsymbol{Q}_r=\gamma\Lambda \tag{6.20}$$

$$\boldsymbol{R}=\boldsymbol{R}_m+\boldsymbol{H}_u\boldsymbol{P}_u^f\boldsymbol{H}_u^T \tag{6.21}$$

where $\boldsymbol{P}_r^a$ is the analysis error covaricance matrix in the resolved subspace; $\boldsymbol{Q}_r$ the 'pesudo' model error covariances; $\boldsymbol{R}$ the observational error covariance; $\boldsymbol{R}_m$ the actual measurement error; and $\boldsymbol{P}_r^f$ the forecast error covaricance matrix in the resolved subspace.

Step 4: Calculation of the stationary Kalman gain K_r in the reduced subspace by the doubling algorithm, estimation of the difference between the observations and forecast and projection to the full space by multiplying E_r

$$\boldsymbol{K}_r(t) = \boldsymbol{P}_r^f(t)(\boldsymbol{H}_r\boldsymbol{P}_r^f(t)\boldsymbol{H}_r^t + \boldsymbol{R})^{-1} \tag{6.22}$$

$$\boldsymbol{P}_r^a(t) = (\boldsymbol{I} - \boldsymbol{K}_r(t)\boldsymbol{H}_r)\boldsymbol{P}_r^f(t) \tag{6.23}$$

$$\boldsymbol{P}_r^f(t+1) = \boldsymbol{M}_r\boldsymbol{P}_r^a(t)\boldsymbol{M}_r^T + \boldsymbol{Q}_r \tag{6.24}$$

where $\boldsymbol{H}_r = \boldsymbol{H}\boldsymbol{D}\boldsymbol{E}_r$ and $\boldsymbol{P}_r^f$ is asymptotically stationary as $t\to\infty$ if $\boldsymbol{H}$, $\boldsymbol{R}$ and $\boldsymbol{Q}$ is stationary with linear dynamics.

Step 5: Data assimilation by using a stationary Kalman gain K_r

$$\boldsymbol{x}^a(t) = \boldsymbol{x}^f(t) + \boldsymbol{D}\boldsymbol{E}_r\boldsymbol{K}_r[\boldsymbol{y}(t) - \boldsymbol{H}\boldsymbol{x}^f(t)] \tag{6.25}$$

RRKF works efficiently in a linear system but not for a nonlinear system. We tested it for various idealized cases such as tidal waves in the circular lakes, the flooding/drying process in the rectangular shape estuary. It produces a fast convergence solution for the linear tidal wave case, but never converges in the estuarine case characterized with nonlinear dynamics.

6.3.2. Ensemble Kalman Filter (EnKF)

Evensen (1994) suggested that the error covariance relative to the mean of ensemble model results could provide a better estimation of the error covariance defined in the classical Kalman Filter. The EnKF is constructed by running a forecast model driven by a set of initial conditions and then estimate the error covariance relative to the ensemble mean to determine the ensemble analysis values for the next time step forecast. This approach is illustrated in Fig. 6.

Let k denote the kth ensemble and N_E the total number of the ensemble members selected in the forecast model run. The forecast value at time step i for the kth ensemble model run can be estimated by

$$\boldsymbol{x}_k^f(i) = \boldsymbol{M}_{i-\Delta t \to i}[\boldsymbol{x}_k^a(i-1)], \quad k = 1,2,...,N_e \tag{6.26}$$

and the analysis values at time step i for the kth ensemble model run are calculated by

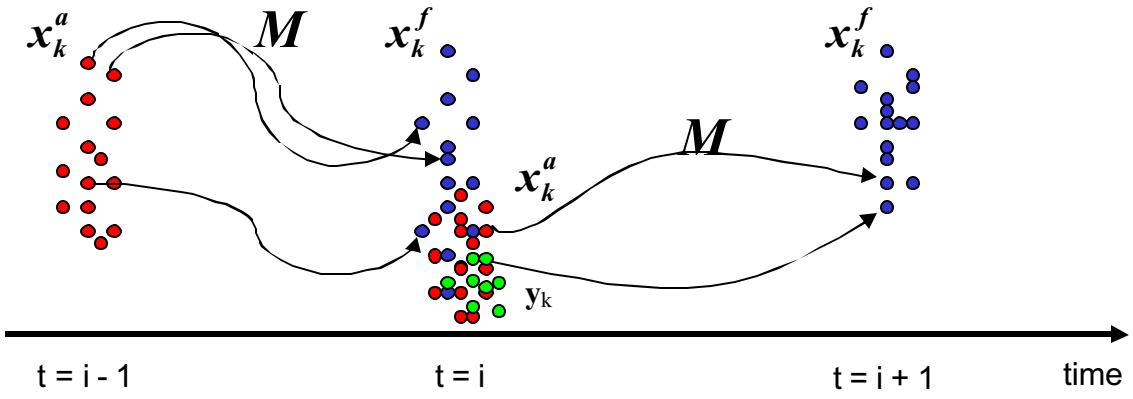


Fig. 6.2: The illustration of the forecast model run with EnKF. This figure was drawn by S. Lyn.

$$\boldsymbol{x}_k^a(i) = \boldsymbol{x}_k^f(i) + \boldsymbol{K}(i)[\boldsymbol{y}_k(i) - \boldsymbol{H}\boldsymbol{x}_k^f(i)], \quad k = 1,\ 2, ...N_e \tag{6.27}$$

Define that

$$\boldsymbol{X}_f = \left\{[\boldsymbol{x}_k^f - \overline{\boldsymbol{x}}^f]/\sqrt{\boldsymbol{N}_e - \boldsymbol{1}}\right\}, \quad k = 1, 2, ... N_e \tag{6.28}$$

then the forecast error covariance can be estimated by

$$\boldsymbol{P}^f(i) \approx \boldsymbol{X}_f(i)\boldsymbol{X}_f^T(i): \quad [N \times N_e][N_e \times N] \tag{6.29}$$

The Kalman gain is equal to

$$\boldsymbol{K}(i) = \boldsymbol{X}_f \boldsymbol{X}_f^T \boldsymbol{H}^T (H \boldsymbol{X}_f \boldsymbol{X}_f^T \boldsymbol{H}^T + \boldsymbol{R})^{-1} \tag{6.30}$$

To conduct the EnKF, we need to create an ensemble of the observational data constructed with the perturbation relative to the real value, *i. e.*,

$$\boldsymbol{y}_k = \boldsymbol{y} + \delta_k,\ \delta_k = N(0, \sqrt{R}) \tag{6.31}$$

where

$$\boldsymbol{R} = \overline{\delta\delta^T} \tag{6.32}$$

In the situation with a sufficiently large number of ensembles, the ensemble analysis error covariance matrix can be updated with a relationship as

$$\boldsymbol{P}_e^a(t) = (\boldsymbol{I} - \boldsymbol{K}_r(t)\boldsymbol{H})\boldsymbol{P}_e^f(t) \tag{6.33}$$

This will give us optimal analysis values in the maximum likelihood sense and in the

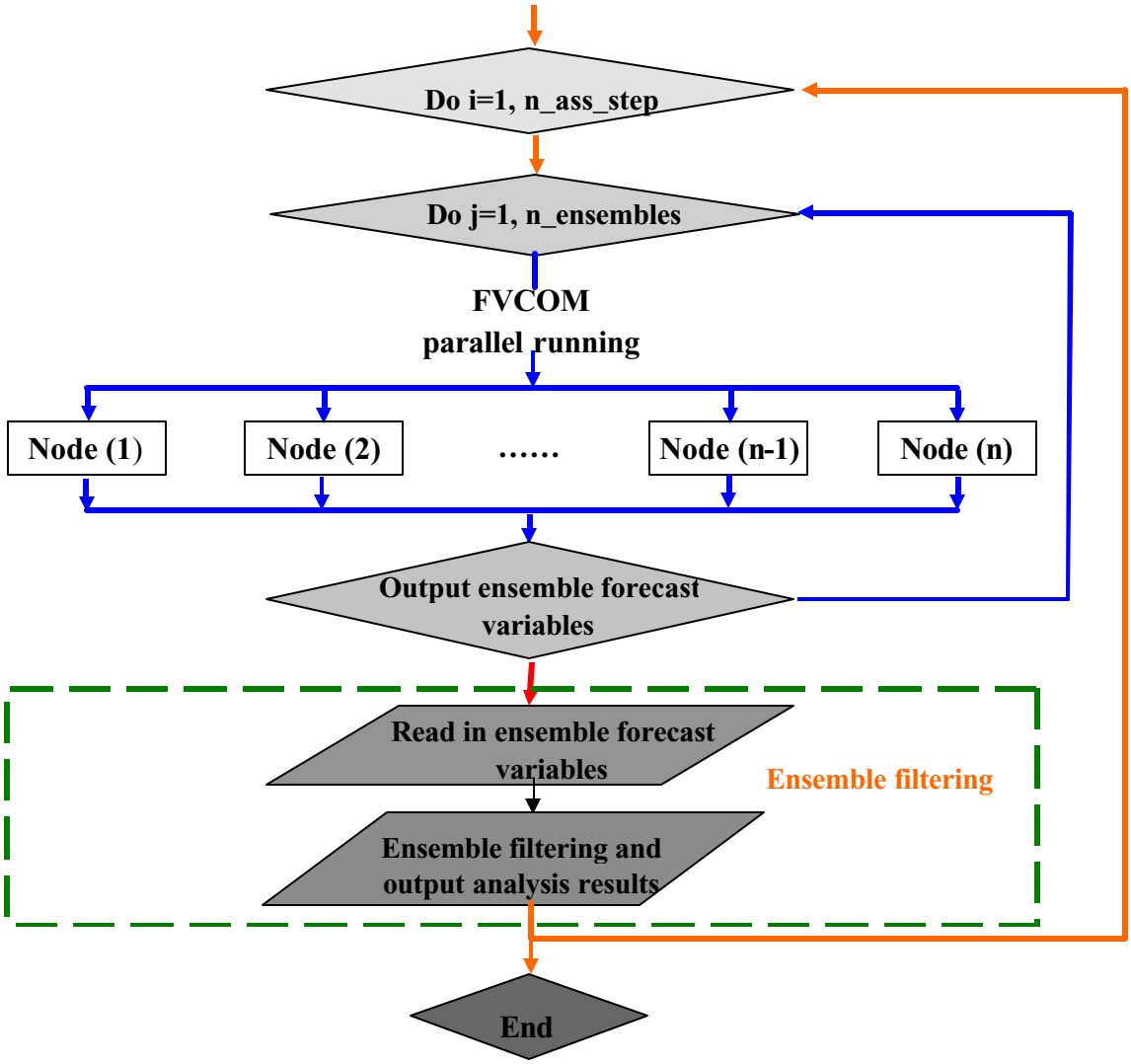


Fig. 6.3: Flow chart schematic of EnKF implemented in FVCOM.

minimum variance sense. In the situation with a small number of ensembles, the perturbed observations required by EnKF may cause a rank deficiency problem for the estimation of $\boldsymbol{P}^f$ and an underestimate of $\boldsymbol{P}^a$, which leads to filter divergence (Whitaker and Hamill, 2002). The solution is to conduct EnKF with covariance localization (Houterkamer and Mitchell, 2001) and covariance inflation (Wang and Bishop, 2003).

The flow chart schematic of EnKF implemented in FVCOM is shown in Fig. 6.3. EnKF is developed for the full nonlinear system. For a linear system, RRKF works well and also fast, but it sometimes fails to resolve linear waves in the idealized, linear coastal ocean system. In such a case, EnKF works well.

6.3.3. Ensemble Square-Root Kalman Filter (EnSRF)

EnSKF is a stochastic filter that requires perturbed sets of observational values. In this system, the perturbed observational values are usually constructed by a control observation in addition to random noise sampled from the assumed observational error distribution. There are derivatives of EnKF that are conducted with deterministic observational ensembles. These filters are known as the EnSRF (Whitaker and Hamill, 2002), Ensemble Transform Kalman Filter (ETKF) (Bishop et al., 2001) and the Ensemble Adjustment Kalman Filter (EAKF) (Anderson, 2001). Actually, these filters are a family of the deterministic square root filter. A brief description of one type of EnSRF is given below.

Assuming that the forecast and observational error covariance is Gaussian distributed, the ensemble Kalman Filter provides optimal analysis values which satisfy the classical Kalman Filter covariance form as

$$\boldsymbol{P}_e^a = (\boldsymbol{I} - \boldsymbol{K}_r \boldsymbol{H})\,\boldsymbol{P}_e^f \tag{6.34}$$

and the Kalman gain is

$$\boldsymbol{K}_e = \boldsymbol{P}_e^f \boldsymbol{H}^T [\boldsymbol{H}\boldsymbol{P}_e^f \boldsymbol{H}^T + \boldsymbol{R}_e]^{-1} \quad . \tag{6.35}$$

Eq. (6.34) can be rewritten into

$$\boldsymbol{P}_e^a = \boldsymbol{X}_a \boldsymbol{X}_a^T \tag{6.36}$$

where

$$\boldsymbol{X}_a = \boldsymbol{X}_f \hat{\boldsymbol{T}} \tag{6.37}$$

and $\hat{\boldsymbol{T}}$ is the square root matrix defined as

$$\hat{\boldsymbol{T}} = \{\boldsymbol{I} - \boldsymbol{X}_f \boldsymbol{H}^T [\boldsymbol{H}\boldsymbol{X}_f \boldsymbol{X}_f^T \boldsymbol{H}^T + \boldsymbol{R}]^{-1} \boldsymbol{H}\boldsymbol{X}_f\} \tag{6.47}$$

where $\boldsymbol{I}$ is $N_e \times N_e$ unit matrix. In this case, an ensemble of the analysis deviation $\boldsymbol{X}_a$ can be estimated deterministically from an ensemble of the forecast deviation, and also the analysis ensemble has a desired error covariance.

Assuming that the observation errors are uncorrelated, the observations can be assimilated serially. In this case, $\boldsymbol{HX}_f \boldsymbol{X}_f^T \boldsymbol{H}^T + \boldsymbol{R}$ in (6.47) is simplified to be a scalar in the case of a single observation. Then the square root matrix can be easily calculated as

$$\hat{\boldsymbol{T}} = [1 - \beta\, \boldsymbol{X}_f^T \boldsymbol{H}^T \boldsymbol{H}\boldsymbol{X}_f][1 - \beta\, \boldsymbol{X}_f^T \boldsymbol{H}^T \boldsymbol{H}\boldsymbol{X}_f]^T \tag{6.48}$$

where $\beta = [a + \sqrt{Ra}\,]^{-1}$.

EnSRF does not require a perturbed observation set, so that the ensemble mean of analysis values $\overline{\boldsymbol{x}}^a$ can be updated directly by the mean of forecast ensemble produced from a traditional Kalman Filter equation as

$$\overline{\boldsymbol{x}}^a(i) = \overline{\boldsymbol{x}}^f(i) + \boldsymbol{X}_f \boldsymbol{X}_f^T \boldsymbol{H}^T [\boldsymbol{H}\boldsymbol{X}_f \boldsymbol{X}_f^T \boldsymbol{H}^T + \boldsymbol{R}]^{-1} (\boldsymbol{y} - \boldsymbol{H}\overline{\boldsymbol{x}}^f) \tag{6.49}$$

and each analysis member $\boldsymbol{x}_k^a$ can be calculated by

$$\overline{\boldsymbol{x}}_{\boldsymbol{k}}^{\boldsymbol{a}}(i) = \overline{\boldsymbol{x}}^a(i) + \boldsymbol{X}_{\mathrm{k}}^{\mathrm{a}} \sqrt{N_e - 1} \quad k = 1,2,\ldots\ldots N_e \tag{6.50}$$

The flow-chart schematic for EnSRF in FVCOM is the same as Fig. 6.3.

6.3.4. Ensemble Transform Kalman Filter (ETKF)

The ETKF is very similar to EnSRF with a linear transformation of the forecast perturbation into the analysis perturbation by $\hat{\boldsymbol{T}}$

$$\boldsymbol{X}_{\boldsymbol{a}} = \boldsymbol{X}_f \hat{\boldsymbol{T}} \tag{6.51}$$

Bishop et al. (2001) proposed the form of the transformation matrix T as:

$$\hat{\boldsymbol{T}} = C(\boldsymbol{G} + \boldsymbol{I})^{-1/2} \tag{6.52}$$

where columns of the matrix C contain the eigenvectors of $\boldsymbol{X}_f^T \boldsymbol{H}^T \boldsymbol{R}^{-1} \boldsymbol{H}\boldsymbol{X}_f$ and $\boldsymbol{G}$ is the nonzero diagonal matrix that satisfies a relationship with C as

$$\boldsymbol{X}_f^T \boldsymbol{H}^T \boldsymbol{R}^{-1} \boldsymbol{H}\boldsymbol{X}_f = \boldsymbol{C}\boldsymbol{G}\boldsymbol{C}^T \tag{6.53}$$

In this case,

$$\boldsymbol{x}^a(i) = \overline{\boldsymbol{x}}^f(i) + \boldsymbol{X}_f \boldsymbol{C}\boldsymbol{G}^{1/2}(\boldsymbol{G}+\boldsymbol{I})^{-1}\boldsymbol{E}^T\{\boldsymbol{R}^{-1/2}\boldsymbol{y} - \boldsymbol{H}\overline{\boldsymbol{x}}^f) \quad (6.54)$$

where

$$\boldsymbol{E} = \boldsymbol{H}\boldsymbol{X}_f\boldsymbol{C}\,\boldsymbol{G}^{-1/2} \quad (6.55)$$

In FVCOM, ETKF is used for adaptive observation optimization in which $\boldsymbol{X}_a$ is used to evaluate the analysis ensemble error covariance under different observational strategies. This approach allows us to use the model to determine the optimal observational network for a selected region.

6.3.5. The Validation Experiments

The FVCOM development team has worked with P. Rizzoli at MIT to validate a package of Kalman Filters implemented into FVCOM. The test problems include tidal oscillations in a flat-bottom circular basin, tidal flooding/draining process in an idealized estuary with an intertidal zone; river discharge plume over an idealized continental shelf, and also adaptive field measurement designs using the ETKF for the plume case. Since we have not yet published any results, we include only a brief summary here.

Test problem: Tidal Oscillations in a Flat-Bottom Circular Basin

Conditions: linear, 2-D, no friction, shallow water, the radius of the basin R = 50 km,

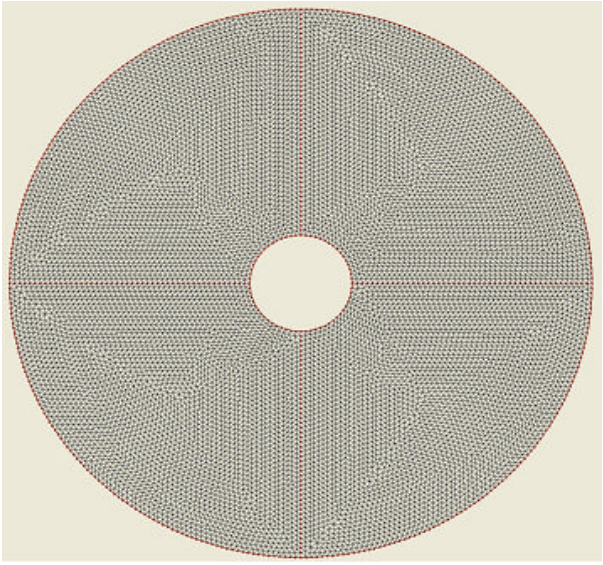


Fig. 6.4: Illustration of the circular basin and unstructured grid.

M_2 tidal forcing at the open boundary. Case I: normal oscillation in which water depth H = 10 m, M_2 tidal elevation at the OB is given as 1.0 m. Case II: near-resonance in which H = 1 m, M_2 tidal elevation at the OB is given as 1 mm. See Fig. 6.4 for the computational domain and triangular grid used to configure FVCOM. The results for the near-resonance case are shown here.

In the near-resonance case, both RRKF and EnKF show a fast convergence to the true solution after a perturbation. Fig. 6.5 shows the RMS analysis results of RRKF. Only one current measurement is made and used in the filter. At 0 hour, the perturbation is

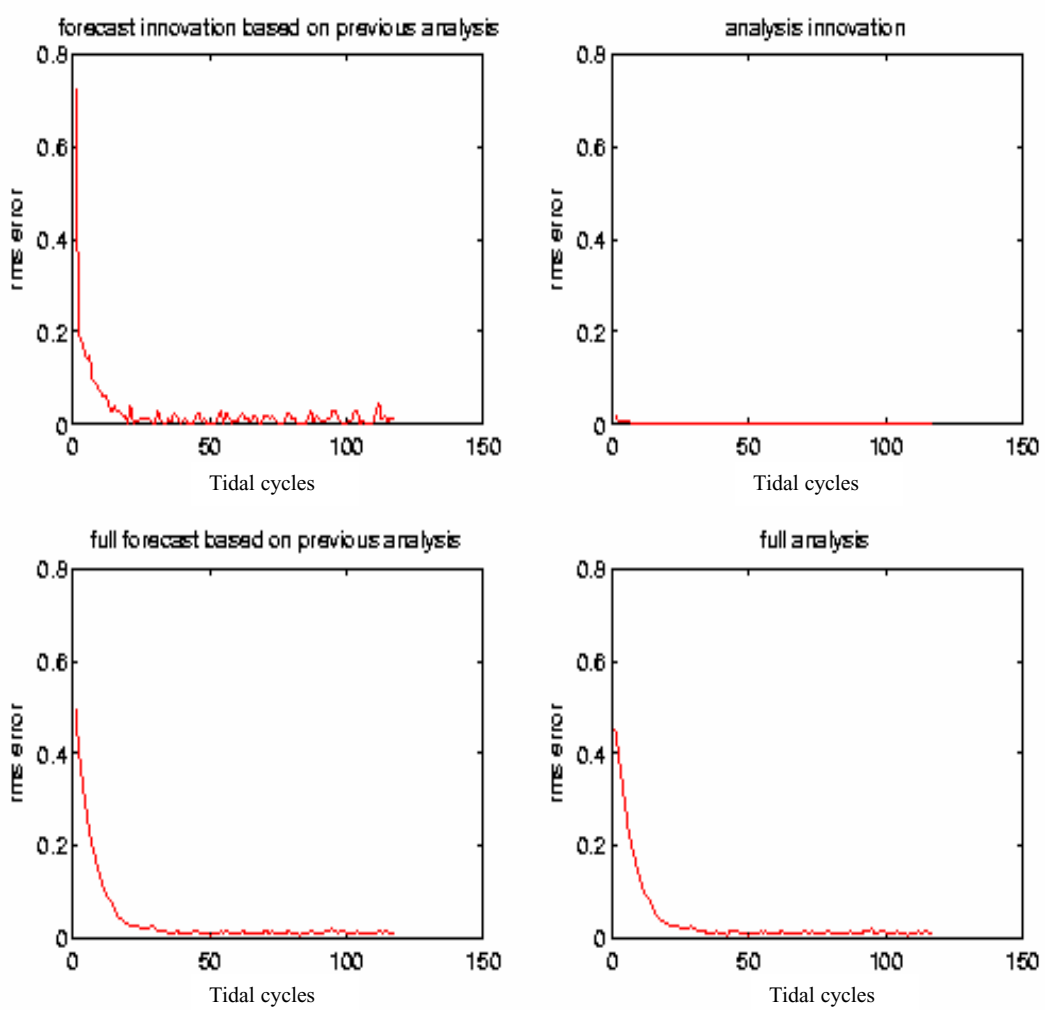


Fig. 6.5: RMS error analysis of RRKF for the near-resonance case.

generated by replacing the model-computed surface elevation and currents with zeros. RRKF quickly induces the solution to converge towards the true state in just one tidal cycle.

Fig. 6.6 shows the surface elevation patterns for the true state, analysis state and error at 0 hour (when the perturbation is generated), 1 hour later, and then 12 hours after the perturbation was generated. Even with this extreme initial perturbation, RRKF works quickly in this case to return the solutions back to the true state.

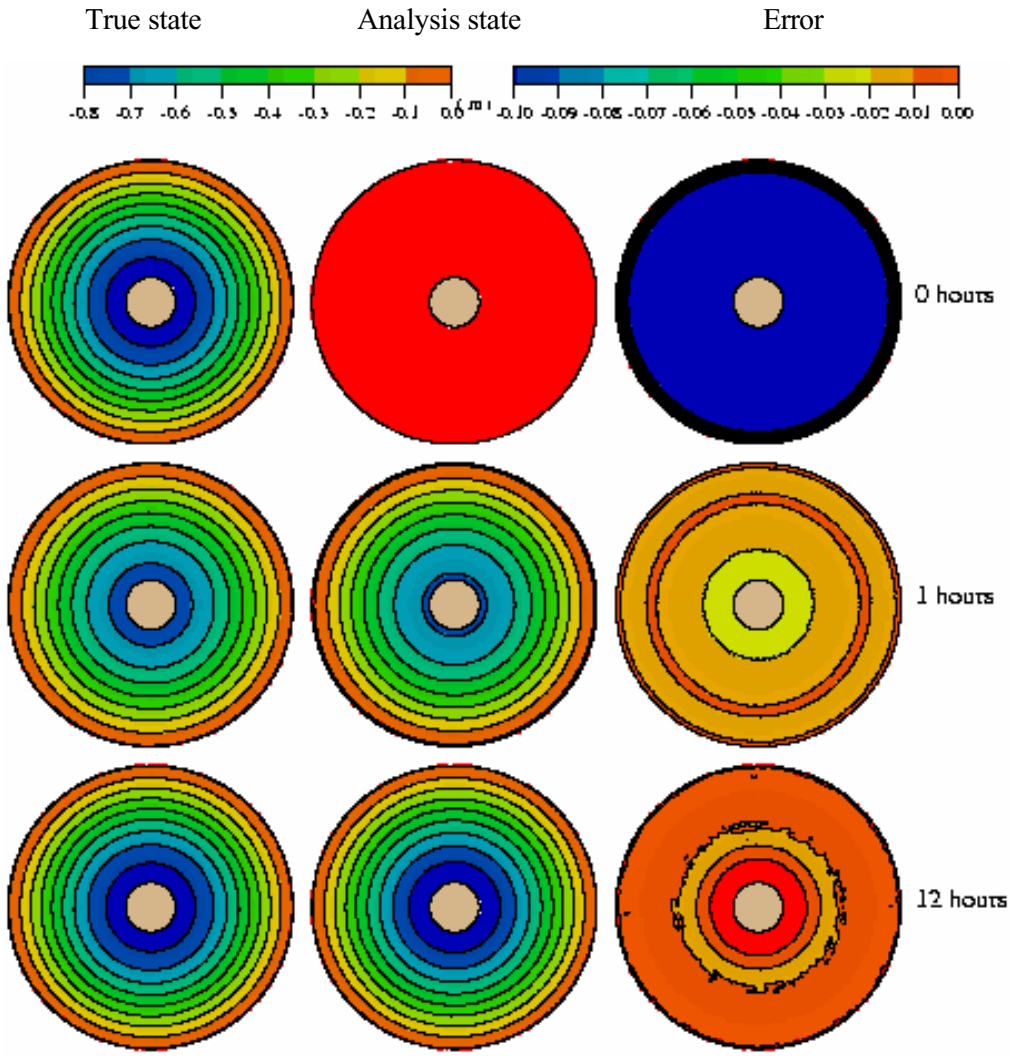


Fig. 6.5: The distributions of the surface elevation of true state (left), analysis state (middle) and error (right) at 0 hour (perturbation is created), 1 hour and 12 hours.

In the EnKF experiment, 20 ensembles were chosen and one surface elevation measurement was taken. Fig. 6.6 shows the RMS analysis results of EnKF for the near-resonance case. Even though we only included one measurement site, the model solution

quickly converged towards the true solution in less than one tidal cycle. 20 ensembles were constructed from previous time frames initialized with random perturbations.

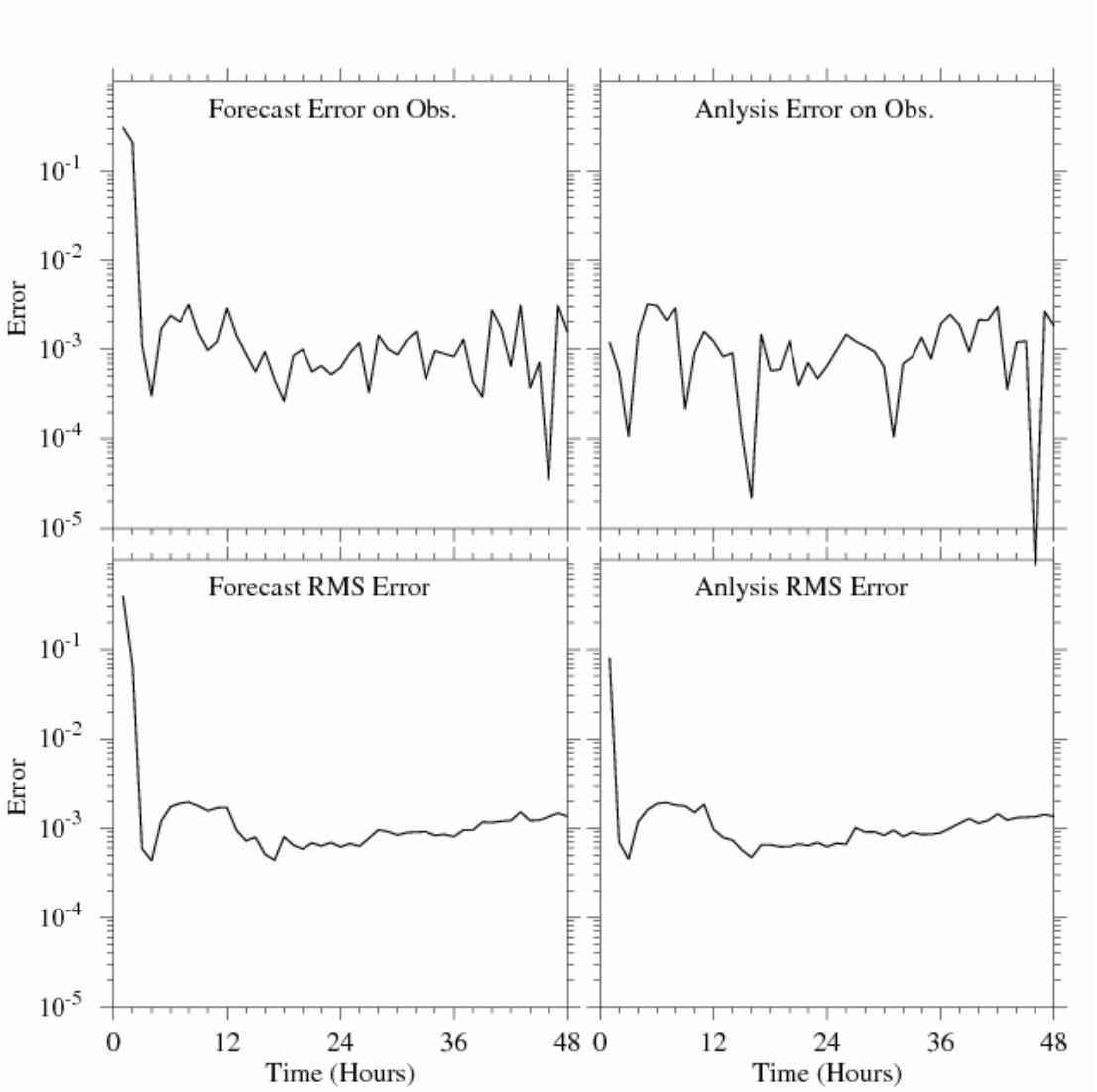


Fig. 6.6: The RMS analysis of EnKF for the near-resonance case.

Fig. 6.7 shows the distributions of the surface elevation for the true state, analysis state and rms errors predicted by EnKF. It can be seen that EnKF works efficiently to suppress the perturbation and direct the numerical solution back to the true state.

We also did experiments to examine the locations of the best measurement sites to achieve optimal model convergence rate, as well as examine the necessary duration of a ship-tracking measurement required to achieve sufficient model convergence.

The Kalman Filters implemented in FVCOM render the model applicable for forecast/hindcast applications for the real coastal ocean.

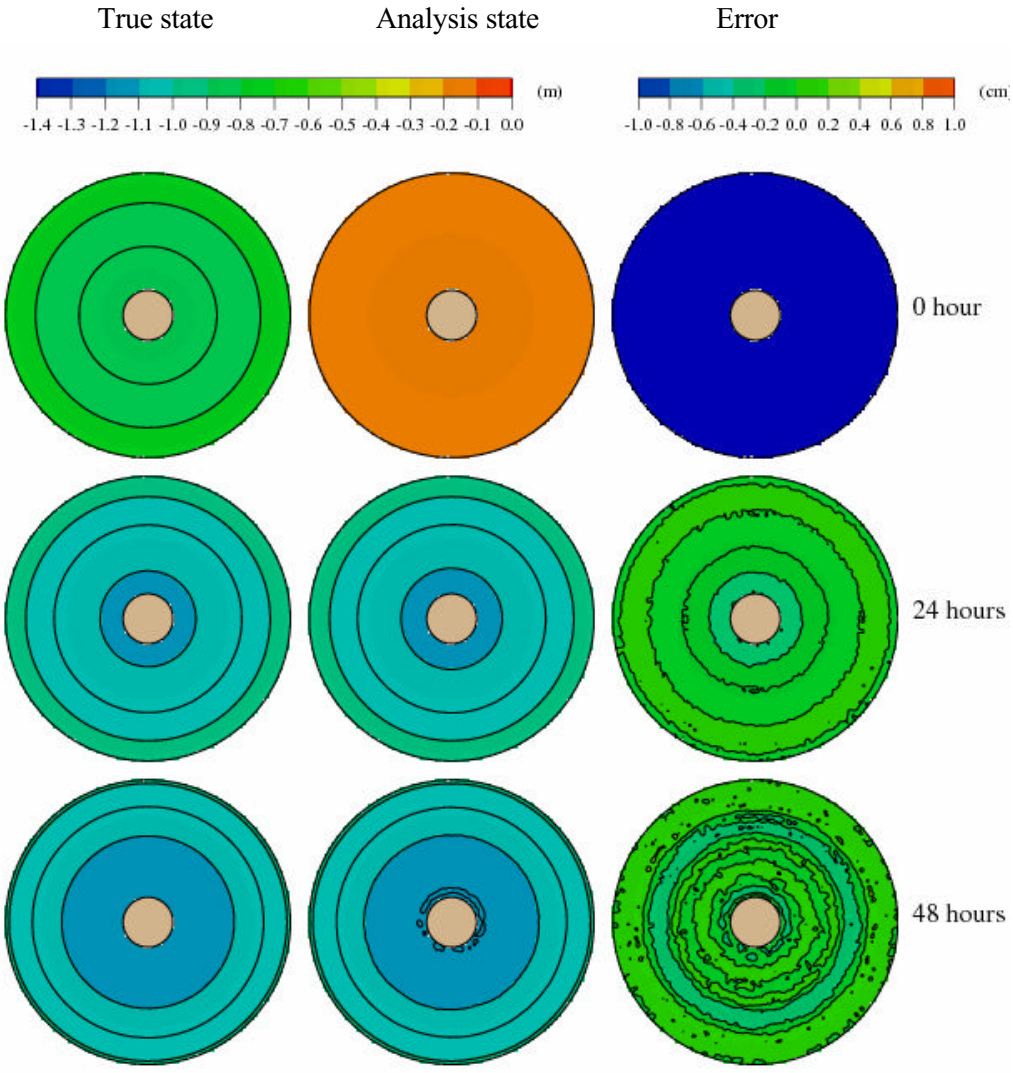


Fig. 6.7: The distributions of surface elevation of the true state (left), analysis state (middle) and error (right) at 0 hour (time of perturbation creation), 1 hour and 12 hours later. In this case, the EnKF was used with 20 ensembles.

Chapter 7: FVCOM Sediment Module

In the early development of FVCOM, we implemented a simple sediment model to the Fortran 77 version when FVCOM was applied to study the Satilla River Estuary. That simple sediment model consisted of a 3-D passive tracer equation with the inclusion of sinking sedimentation via resuspension processes. This model was not included in the parallel version of FVCOM because it was too simple to be a practical tool for the study of sediment transport processes in the coastal ocean.

G. Cowles has implemented a more complete 3-D sediment model module for FVCOM. This sediment model is based on the Community Model for Coastal Sediment Transport developed by the USGS and other researchers (see http://woodshole.er.usgs.gov/project-pages/sediment-transport/), which includes suspended sediment and bedload transport, layered bed dynamics based on active layer concept, flux-limited solution of sediment setting, unlimited number of sediment classes and bed layers and cohesive sediment erosion/deposition algorithms. The implementation described herein is based on J. Warner's (USGS) coding for the Community Model in the Regional Ocean Modeling System (ROMS). A major effort was made to convert the structured-grid ROMS code to the unstructured-grid FVCOM code and to use the mass conservative finite-volume approach to calculate the sediment advection. By incorporating the same sediment dynamics and utilizing similar parameterizations and empirical constants, direct comparisons between FVCOM and ROMS results can be made. In addition, further development to the model can easily by implemented into FVCOM. This manual includes a brief description of the model dynamics, instructions on using the sediment module, some details on the coding implementation, and results from a test case with suspended load only.

7.1. Governing Equations

The model includes transport of both suspended load and bedload. There exists a region near the bottom where there is no clear distinction between these two sources. In this implementation, however, the loads are computed separately and added together to

produce the total load. The suspended load model uses a concentration-based approach subject to the following evolution equation:

$$\frac{\partial C_i}{\partial t}+\frac{\partial uC_i}{\partial x}+\frac{\partial vC_i}{\partial y}+\frac{\partial (w-w_i)C_i}{\partial z}=\frac{\partial}{\partial x}(A_H\frac{\partial C_i}{\partial x})+\frac{\partial}{\partial y}(A_H\frac{\partial C_i}{\partial y})+\frac{\partial}{\partial z}(K_h\frac{\partial C_i}{\partial z}) \quad (7.1)$$

where C_i is the concentration of sediment I, A_h is the horizontal eddy viscosity and K_h is the vertical eddy viscosity. The settling velocity, w_i, is prescribed by the user for each sediment type in the input parameter file. At the surface, a no-flux boundary condition is used for the sediment concentration:

$$K_h\frac{\partial C_i}{\partial z}=0, \quad z=\zeta \quad (7.2)$$

At the bottom, the sediment flux is the difference between deposition and erosion:

$$K_h\frac{\partial C_i}{\partial z}=E_i-D_i, \quad z=\zeta \quad (7.3)$$

The erosion rate is calculated as:

$$E_i=\Delta t Q_i(1-P_b)F_{bi}\left(\frac{\tau_b}{\tau_{ci}}-1\right) \quad (7.4)$$

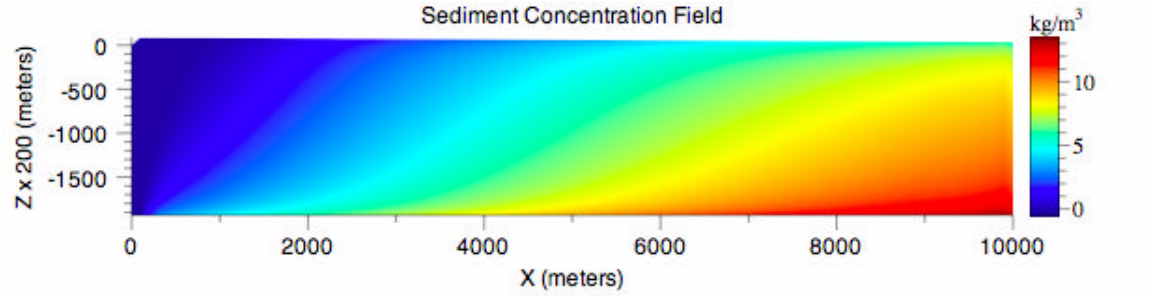
where Q_i is the erosive flux, P_b is the bottom porosity, F_{bi} is the fraction of sediment i in the bottom, τ_b is the bottom shear stress, and τ_{ci} is the critical shear stress of sediment i.

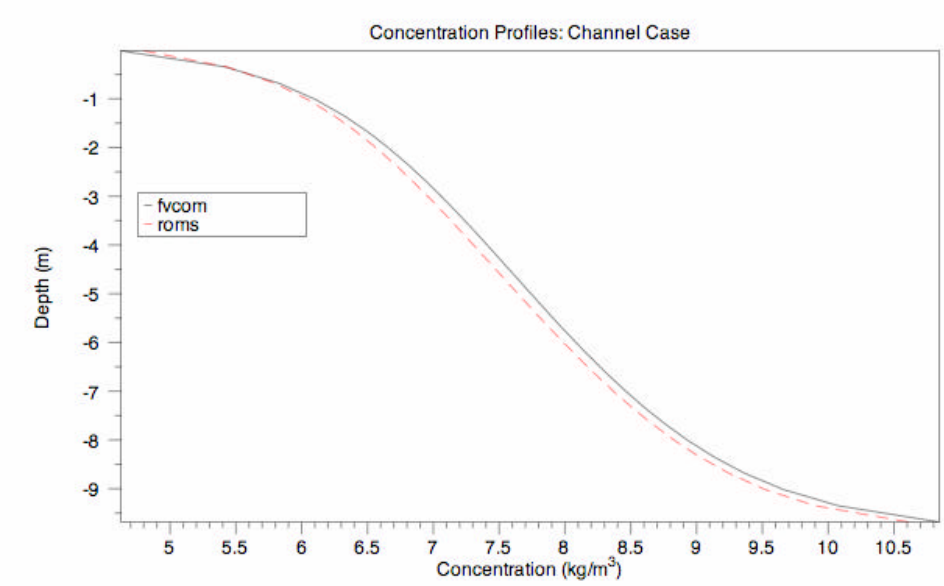
Sediment is scoured when the local bottom shear stress reaches a critical user-defined value and is removed at a rate defined by the user. The resulting concentration profile is dependent on a balance between advection, vertical diffusion, introduction of new material through erosion, and loss of material from the water column through settling. A constant, user-defined sink rate w_i is used to model settling for each sediment type. The settling term (term 4 in the evolution equation) must be carefully calculated due to the sharp gradients in the concentration profile that can occur near the bottom. In this implementation, a flux-limited scheme is used for the settling equation. This scheme introduces antidiffusion by means of a min-mod limiter and maintains second-order spatial accuracy away from extrema. The settling flux through the bottommost control volume is saved and recycled as the depositional flux D_i in the evolution equation bottom boundary condition. The bedload is treated using the Meyer-Peter Muller scheme to calculate the local load. The transport is determined by calculating the divergence of the

local load. Users can select the empirical parameters of the Meyer-Peter Muller scheme through the sediment input file.

7.2. A Simple Test Case

The test case described here corresponds to Case 1: Channel Flow of the USGS Community Model for Coastal Sediment Transport. This case tests the primary dynamics of suspended load in the sediment model including treatment of settling, erosion, vertical diffusion, and advection. Neither the bedload nor any specific dynamics of the bed strata are considered in this case. See the USGS web page provided above for specifics regarding the case setup. Fig. 7.1 shows the resulting sediment concentration field and Fig.7.2 displays the steady state sediment concentration profile at x = 8000 m for both ROMS and FVCOM. The difference between the two profiles is quite small.





Chapter 8: FVCOM Biological Modules

In 2005, we started implementing the Generalized Biological Module (GBM) into FVCOM. The original plan was to convert the Harvard University's structured-grid adjustable biological module code to an unstructured-grid version and add it into FVCOM as a biological module. At that time, FVCOM already had several biological modules, including: 1) a water quality model (coded for parallel execution), 2) a simple NPZ model, and 3) a 9 component NPZD model (coded for serial execution only). The idea was to build a GBM which would allow users to select either a pre-built biological model (such as NPZ, NPZD, etc) or construct their own biological model using the pre-defined pool of biological variables and parameterization functions. This module could be run simultaneously together with FVCOM with parallel execution (so-called "online" mode) or driven separately by FVCOM output ("offline" mode). This module would act as a platform that would allow users to examine the relative importance of different physical and biological processes under well-calibrated physical fields. Following the code structure of the Harvard GBM, R. Tian formulated and implemented a GBM for FVCOM. Early usage of this module indicated that the generality made the coding difficult to follow and the GBM difficult to setup, run and debug. After meeting with R. Tian and R. Ji (WHOI), we decided that the strategy of making the module so general was introducing unnecessary complications and that a different route would be sought towards building a biological model with similar capabilities and a far simpler coding structure. Named the Flexible Biological Module (FBM), this new module could be driven by the physical model in FVCOM or by other popular ocean models. The implementation described next is based on C. Chen's notes that were used to construct the FBM for FVCOM.

8.1 Flexible Biological Module (FBM)

8.1.1. Flow Chart of FBM

The structure of the Flexible Biological Module was developed by dividing lower trophic food web processes into 7 state variable groups: 1) nutrients [$N(i)$, i=1, nn], 2) phytoplankton [$P(i)$, i=1,np], 3) zooplankton [$Z(i)$, i=1, nz], 4) detritus [$D(i)$, i=1,nd], 5)

dissolved organic matter [*DOM*(*i*), *i*=1,nm], 6) bacteria [*B*(*i*), *i*=1, nb], and 7) auxiliary state variables [*Y*(*i*), *i*=1, *ny*]. The flow chart of the transformation among these variables is shown in Fig. 8.1. We named this system "Flexible Biological Module (FBM)" to emphasize two points. First, this module provides a platform that allows users to build their own parallelized biological model from a discrete set of functions that is independent of the physical model. This module can be run simultaneously with linkage to unstructured-grid (e.g.: FVCOM) and structured-grid ocean models through the connection to the physical model dependent 3-D advection and diffusion variables or it can be run separately by itself in 1-D applications. This structure is adapted from the popular General Ocean Turbulence Model (GOTM) system which uses the same approach. The second reason we chose the descriptor "Flexible" is that we realized that the range of existing biological models is too vast and complex to try to encompass in a generalized way. In the FBM code, the biological module is an independent 1-D system that is self-maintained and upgraded without linking to a physical model. It is easy to extend the FBM to a 3-D case by linking to the advection and diffusion modules of any physical model. It can be also converted to a Lagrangrian-based biological model by linking it with the 3-D Lagrangian particle-tracking module.

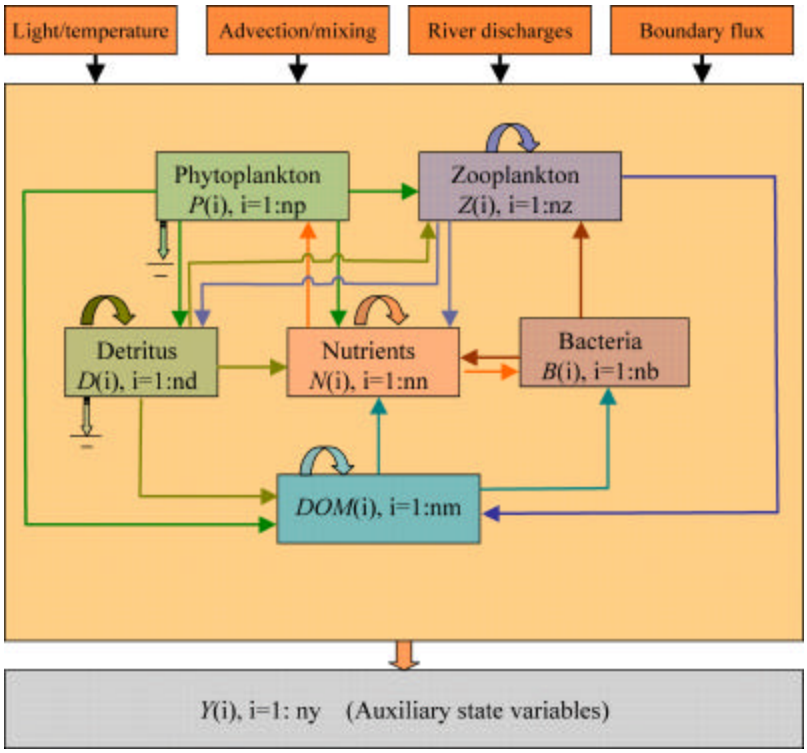


Fig. 8.1: Flow chart of the flexible biological module.

The biological module in FVCOM includes point source input from rivers, nudging at lateral boundaries, air-sea interaction at the surface and benthic flux at the bottom.

Because these physical processes are already documented in the FVCOM manual and code, we will focus our description of the FBM here on the internal biological processes.

8.1.2. Equations and Functions in the FBM

8. 1.2.1. Nutrients

Nutrients in the biological module include 1) ammonium (NH_4^+), nitrate (NO_3^-), nitrite (NO_2^-), phosphate (PO_4^-), and silicate [$Si(OH)_4$]. We also include iron (Fe) in the nutrient equation because in many coastal and open ocean environments it acts as a limiter in autotrophic processes. Two processes are included for nutrient field modification: 1) external loading and advection (physical processes) and 2) internal regeneration resulting from respiration by phytoplankton, zooplankton and bacteria and remineralization from detritus. The loss of nutrients is accounted for in uptake by phytoplankton and bacteria. A brief description of biological processes is shown in Fig. 8.2. To facilitate model use, carbon is used as the standard unit for all autotrophic and heterotrophic variables. For

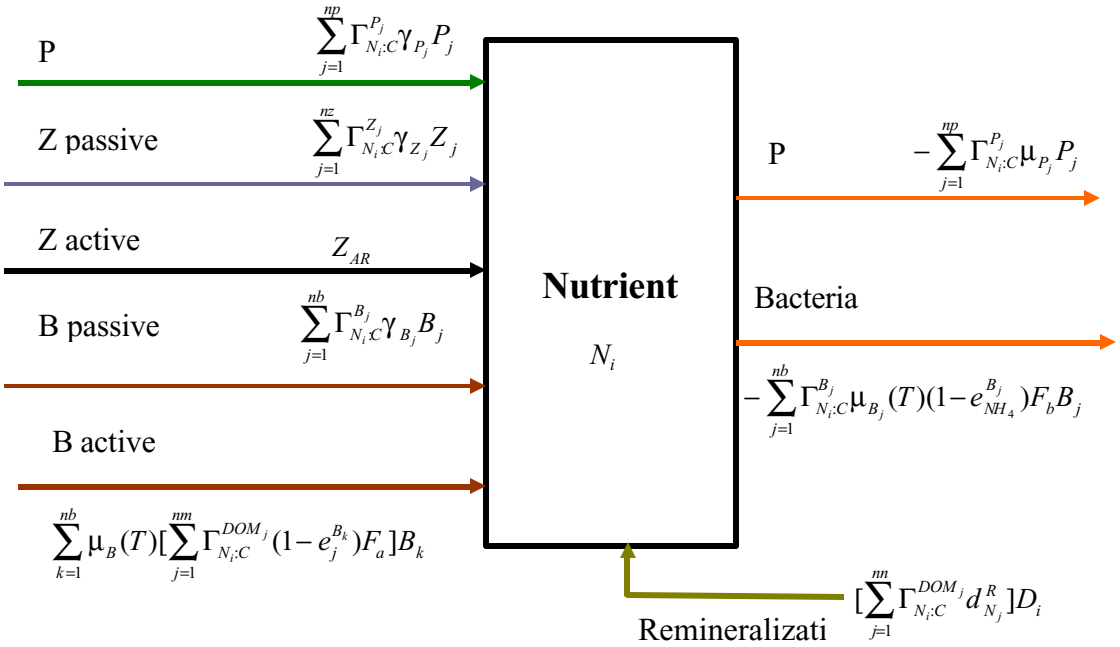


Fig. 8.2: Inflow and outflow chart for the equation for nutrient N_i

nutrients, the standard is the nutrient itself. Therefore, the ratio of a carbon to nutrient unit is defined to support conversion from one base to another. Define [N_i, i=1, nn] as

the general form of the nutrient variables where *i* represents the *i*th nutrient variable and *nn* is the total number of nutrients included in the module. We can write the general form of the nutrient equation as

$$\frac{\partial N_i}{\partial t} = Loss\_N_i + Gain\_N_i + ADV\_N_i + HDIFF\_N_i + VDIFF\_N_i \tag{8.1}$$

where "Loss_N_i" represents the *i*th nutrient uptake by phytoplankton and bacteria; "Gain_N_i" is the *i*th nutrient regeneration through the respiration of phytoplankton, zooplankton and bacteria, decomposition of detritus, and remineralization of DOM; ADV_N_i is the change of the *i*th nutrient due to horizontal and vertical advection; and HDIFF_N_i and VDIFF_N_i are the change of the *i*th nutrient due to horizontal and vertical diffusion, respectively. ADV_N_i and HDIFF_N_i are the two terms that link Eq. (8.1) to a 3-D physical model. We also take nitrification and denitrification into account in the nutrient equations for NO_3^-, NO_2^- and NH_4^+. The mathematic expressions for "Loss_N_i" and "Gain_ N_i" are

$$\begin{aligned} Loss\_N_i &= P\_take + B\_uptake \\ &= -\sum_{j=1}^{np} \Gamma_{N_i:C}^{P_j} \mu_j P_j - \sum_{j=1}^{nb} \Gamma_{N_i:C}^{B_j} \mu_{B_j}(T)[1 - e_{NH_4}^{B_j}] F_b B_j \end{aligned} \tag{8.2}$$

and

$$\begin{aligned} Gain\_N_i = {} & P\_respiration + Z\_passive\ \ respiration + Z\_active\ \ respiration \\ & + B\_passive\ \ respiration + B\_active\ \ respiration \end{aligned} \tag{8.3}$$

where

$$P\_respiration = \sum_{j=1}^{np} \Gamma_{N_i:C}^{P_j} \gamma_{P_j} P_j \tag{8.4}$$

$$Z\_passive\ respiration = \sum_{j=1}^{nz} \Gamma_{N_i:C}^{Z_i} \gamma_{Z_j} Z_j \tag{8.5}$$

$$B\_passive\_respiration = \sum_{j=1}^{nb} \Gamma_{N_i:C}^{B_j} \gamma_{B_j} B_j \tag{8.6}$$

$$Z\_active\,respiration = \sum_{k=1}^{nz}\{\sum_{j=1}^{np}\Gamma_{N_i:C}^{P_j}[(1-e_{P_j}(k)-\alpha_{P_j}^{D}(k)-\alpha_{P_j}(k)]G_{P_j}$$

$$+\sum_{\substack{j=1\\ j\neq i}}^{nz}[1-e_{Z_j}(k)-\alpha_{Z_j}^{D}(k)-\alpha_{Z_j}(k)]+G_{Z_j}\sum_{j=1}^{nb}\Gamma_{N_i:C}^{B_j}[1-e_{B_j}(k)-\alpha_{B_j}(k)]B_j \qquad (8.7)$$

$$+\sum_{j=1}^{nd}\Gamma_{N_i:C}^{D_j}[1-e_{D_j}-\alpha_{D_j}(k)-\alpha_{B_j}^{D}(k)]D_j\}$$

$$B\_active\_respiration = \sum_{k=1}^{nb}\mu_B(T)[\sum_{j=1}^{nm}\Gamma_{N_i:C}^{DOM_j}(1-e_j^{B_k})F_a]B_k \qquad (8.8)$$

$$D\_re\min eralization = [\sum_{j=1}^{nn}\Gamma_{N_i:C}^{DOM_j}d_{N_j}^{R}]D_j \qquad (8.9)$$

$$P\_uptake = -\sum_{j=1}^{np}\Gamma_{N_i:C}^{P_j}\mu_{P_j}P_j \qquad (8.10)$$

$$B\_uptake = -\sum_{j=1}^{nb}\Gamma_{N_i:C}^{B_j}\mu_{B_j}(T)(1-e_{NH_4}^{B_j})F_b \qquad (8.11)$$

where $\Gamma_{N_i:C}^{P_j}$, $\Gamma_{N_i:C}^{Z_j}$, $\Gamma_{N_i:C}^{B_j}$, $\Gamma_{N_i:C}^{D_j}$ and $\Gamma_{N_i:C}^{DOM_j}$ are the ratio of the *i*th nutrient N_i unit to carbon for the *j*th phytoplankton P_j, zooplankton Z_j, bacteria B_j, detritus D_j, and DOM_j. Parameters used for the nutrient equation is listed on Table 8.1.

We also consider the nitrification from ammonium to nitrate through nitrifying bacteria. It has been recognized that nitrification is a process related to light. A simple empirical formula for the nitrification rate (Olson, 1981) is given here as an option:

$$Q_{NH_4\rightarrow NO_3} = \begin{cases} 0 & I(t,z) \geq 0.1I_{\max} \\ \dfrac{0.1I_{\max} - I(t,z)}{0.1I_{\max}} & I(t,z) < 0.1I_{\max} \end{cases} \qquad (8.12)$$

where $I_{\max}$ is the maximum light intensity at the surface, and I is the light intensity in the water column.

Table 8.1: Parameters used in the nutrient equations

Name	Definition	Unit
$\Gamma_{N_i:C}^{P_j}$,	ratio of the *i*th nutrient N_i unit to carbon for the *j*th	mmol N_i:mmol C

	phytoplankton P_j	
$\Gamma_{N_i:C}^{Z_j}$	ratio of the ith nutrient N_i unit to carbon for the jth zooplankton Z_j	mmol N_i:mmol C
$\Gamma_{N_i:C}^{B_j}$	ratio of the ith nutrient N_i unit to carbon for the jth bacteria B_j	mmol N_i:mmol C
$\Gamma_{N_i:C}^{D_j}$	ratio of the ith nutrient N_i unit to carbon for the jth detritus D_j.	mmol N_i:mmol C
$\Gamma_{N_i:C}^{DOM_j}$	ratio of the ith nutrient N_i unit to carbon for the jth DOM_j	mmol N_i:mmol C

8.1.2.2. Phytoplankton

Phytoplankton growth is an autotrophic process that is dependent on light photosynthesis and nutrient limitation. The local change of the phytoplankton is controlled by the uptake of nutrients via respiration, mortality, DOM exudation, sinking

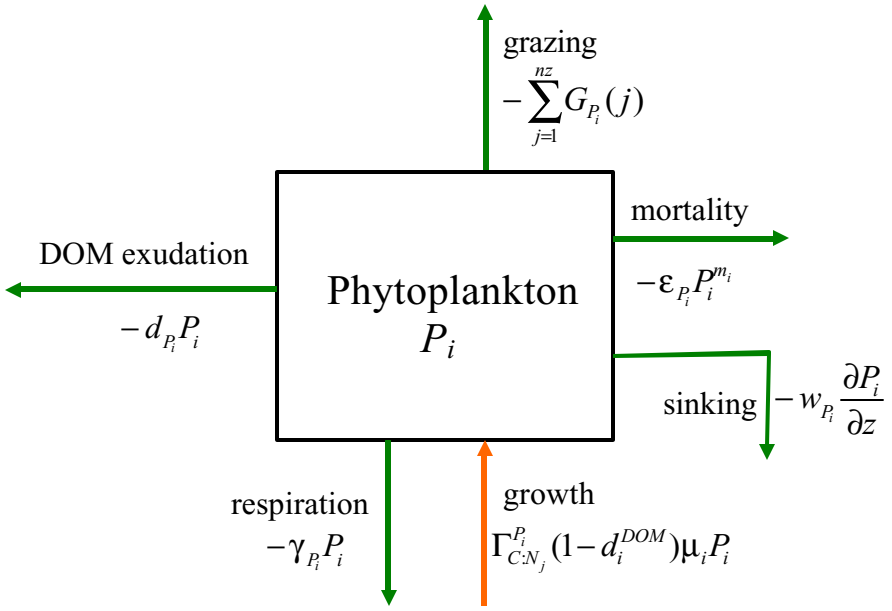


Fig. 8.3: Flow chart of the input and output in the phytoplankton species P_i equation.

and grazing physical advection and diffusion processes. The inflow and outflow chart of the ith phytoplankton species is shown in Fig. 8.3. The equation for the ith phytoplankton species can be written as

$$\frac{\partial P_i}{\partial t} = \text{Growth of } P_i + \text{Respiratio n of } P_i + \text{Mortality of } P_i + \text{DOM exudation of } P_i + \text{Sinking of } P_i + \text{Grazing of } P_i + \text{ADV\_}P_i + \text{HDIFF\_}P_i + \text{VDIFF\_}P_i \quad (8.13)$$

where

$$\text{Growth of } P_i = (1 - d_i^{DOM})\mu_{Pi}\theta_{i(chl:N_i)}P_i \quad (8.14)$$

$$\text{Respiratio n of } P_i = -\gamma_{P_i}P_i \quad (8.15)$$

$$\text{Mortality of } P_i = -\varepsilon_{P_i}P_i^{m_i} \quad (8.16)$$

$$\text{DOM exudation of } P_i = -d_{P_i}P_i \quad (8.17)$$

$$\text{Sinking of } P_i = -w_{P_i}\frac{\partial P_i}{\partial z} \quad (8.18)$$

$$\text{Grazing of } P_i = -\sum_{j=1}^{nz} G_{P_i}(j) \quad (8.19)$$

and ADV_P_i, HDIFF_P_i and VDIFF_P_i are the change of P_i due to advection, horizontal and vertical diffusion, respectively. $G_{P_i}(j)$ is the grazing loss by the jth zooplankton species. Definitions of parameters used in (8.13)-(8.19) are given in Table 8.2.

Table 8.2: Parameters used in (8.13)-(8.19)

$\mu_{P\iota}$	gross growth rate of P_i	mmol C (mg Chl)$^{-1}$ s^{-1}
$\Gamma_{C:N_j}^{P_j}$	ratio of C to nitrogen	mmol C:mmol N
d_i^{DOM}	active DOM exudation	fraction
γ_{P_i}	passive DOM exudation	s^{-1}
ε_{P_i}	mortality rate of P_i	s^{-1}
m_i	power of P_i in the mortality term	dimensionless
γ_{P_i}	respiration rate of P_i	s^{-1}

w_{P_i}	sinking velocity of P_i	m s^{-1}

There are many empirical methods to determine μ_{Pi}. Three factors are commonly considered in the estimation of μ_{Pi}: light intensity, water temperature and nutrient limitation. In a multiple nutrient case, two popular empirical formulas for μ_i are given as

$$\mu_{iA} = \mu_i(T)\min[\mu_i(I), \mu_i(N_1), \mu_i(N_2)\cdots\mu_i(N_{nn})] \tag{8.20}$$

$$\mu_{iB} = \mu_i(T)\mu_i(I)\min[\mu_i(N_1), \mu_i(N_2),,,\mu_i(N_{nn})] \tag{8.21}$$

where $\mu_i(I)$ is the normalized light limitation function, $\mu_i(T)$ is the growth limitation due to the water temperature, and $\mu_i(N_i)$ is the nutrient limitation function due to the ith nutrient. To make the selection more flexible, we combine (8.20) and (8.21) together as follows:

$$\mu_i = \mu_{\max}^{Pi}(\alpha\mu_{iA} + (1-\alpha)\mu_{iB}) \tag{8.22}$$

where α is a weighting coefficient ranging from 0 to 1 and μ^{Pi}_{max} is the maximum growth rate of phytoplankton P_i

$\mu_i(T)$ is usually expressed as an exponential function as

$$\mu(T) = e^{-\alpha_T|T-T_{opt}|} \tag{8.23}$$

where T_{opt} is the optimal water temperature at which the maximum growth rate is measured and α_T is the exponential decay rate of $\mu_i(T)$ relative to the water temperature difference.

The selection of $\mu_i(I)$ is empirical, which relies on the best fit of the relationship between the growth rate and light intensity. To make the code more general, we include 13 forms for $\mu_i(I)$ here (see Table 8.3). These empirical formulas are added into the code in the form of a Fortran function to make it easy for users to select one of the existing formulas or add their own one.

Table 8.3: **Functions of $\mu_i(I)$ included in the FVCOM Biological Module**

Function Name	Equation	References

EXP_LIGHT	$e^{-k\|z\|}$	Franks et al. (1986)
SL62_LIGHT	$\frac{I}{I_{opt}} e^{\left(1-\frac{I}{I_{opt}}\right)}, I = I_o e^{-k\|z\|}$	Steele (1962)
MM_LIGHT	$\frac{\alpha_I I}{K_I + \alpha_I I}$	Baly (1935); Tamiya et al. (1953); Caperon (1967); Kiefer and Mitchell (1983)
LB_LIGHT	$\frac{\alpha_I I}{\sqrt[n]{K_I^n + (\alpha_I I)^n}}$	Bannister (1979); Laws and Bannister (1980)
V65_LIGHT	$\frac{\alpha_I I}{\sqrt{I_{opt}^2 + \alpha_I^2 I^2}} \frac{1}{[1+(\beta \frac{I}{I_{opt}})^2]^{n/2}}$	Vollenweider (1965)
PE78_LIGHT	$\frac{I}{I_{opt}} \frac{2+\alpha_I}{1+\alpha_I \frac{I}{I_{opt}} + \left(\frac{I}{I_{opt}}\right)^2}$	Peeters and Eilers (1978)
WNS74_LIGHT	$1 - e^{-\frac{\alpha_I I}{\mu_{max}}}$	Webb et al. (1974)
PGH80_LIGHT	$(1 - e^{-\frac{\alpha_I I}{\mu_{max}}}) e^{-\frac{\beta I}{\mu_{max}}}$	Platt et al. (1980)
JP76_LIGHT	$\tanh(\frac{\alpha_I I}{\mu_{max}})$	Jassby and Platt (1976)
BWDC99_LIGHT	$\tanh[\frac{\alpha_I (I - I_o)}{\mu_{max}}] e^{\beta (I_{opt} - I)}$	Bissett et al. (1999)

Note: I_{opt} is the optimal light intensity at which the phytoplankton growth rate reaches its maximum value; I is the light intensity; I_o is the compensation light intensity; α_I is a light parameter related to the slope of the light function; μ_{max} is the maximum growth rate; and β is the parameter determining the photo inhibition. All of these equations are derived or proposed based on fits to observations, and the choice of function to use should be made with caution, particularly in a situation without any observational data for guidance.

In our biological module, the nutrient limiting function $\mu_i(N_i)$ is specified using the Michaelis-Menten form given as

$$\mu_i(N_j)=\begin{cases}\dfrac{N_j-N_{j\min}}{K_{js}+N_j-N_{j\min}} & N_j>N_{j\min}\\ 0 & N_j\le N_{j\min}\end{cases},\quad j=1,2,3\cdots nn \tag{8.24}$$

where N_j is the *j*th nutrient concentration, $N_{j\min}$ is the threshold of the *j*th nutrient concentration and K_{js} is the half-saturation constant of the *j*th nutrient concentration. Note that the subscript "*i*" represents the *i*th phytoplankton species.

In the case when both ammonium and nitrate are considered, the phytoplankton usually prefer ammonium relative to nitrate. The limiting function of nitrate is usually inhibited by ammonium. A widely used expression of nitrate with consideration of the inhibition factor due to ammonium is given here as

$$\mu_i(NO_3)=\begin{cases}\dfrac{NO_3-(NO_3)_{\min}}{K_{NO_3}+NO_3-(NO_3)_{\min}}\cdot\dfrac{K_{NH_4}}{K_{NH_4}+NH_4-(NH_4)_{\min}} & \begin{cases}NO_3>(NO_3)_{\min}\\ NH_4>(NH_4)_{\min}\end{cases}\\ \dfrac{NO_3-(NO_3)_{\min}}{K_{NO_3}+NO_3-(NO_3)_{\min}} & \begin{cases}NO_3>(NO_3)_{\min}\\ NH_4\le(NH_4)_{\min}\end{cases}\\ 0 & NO_3\le(NO_3)_{\min}\end{cases} \tag{8.25}$$

and the ammonium concentration is given as

$$\mu_i(NH_4)=\frac{NH_4-(NH_4)_{\min}}{K_{NH_4}+NH_4-(NH_4)_{\min}} \tag{8.26}$$

If only nitrate and ammonium concentrations are considered as nutrients, then the total nitrogen is the sum of (8.25) [for $\mu_i(NO_3)$] and (8.26) [for $\mu_i(NH_4)$].

8.1.2.3. Zooplankton

Ocean and estuarine ecosystems generally feature multiple zooplankton species that are competitive in the lower trophic food web dynamics. Since it is very difficult to create a simple platform to include all heterotrophic processes involving zooplankton, particularly when various life stages of the different dominant zooplankton species are

considered, we have chosen to include only the basic transformations involving zooplankton in the lower trophic food web, which are described next.

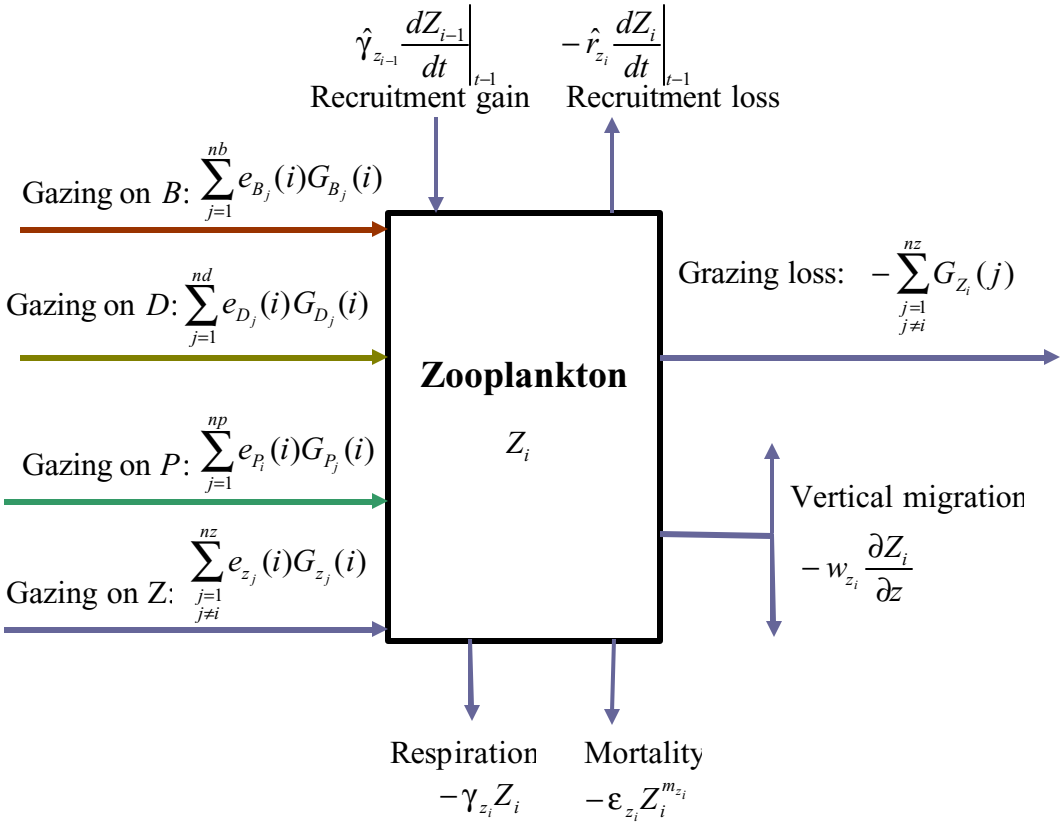


Fig. 8.4: Inflow and outflow chart of the *i*th zooplankton species Z_i.

The inflow and outflow chart for an individual zooplankton species and its interaction with other zooplankton is shown in Fig. 8.4. The governing equation for the *i*th zooplankton species Z_i is given as

$$\begin{aligned}\frac{\partial Z_i}{\partial t} &= Grazing\_P + Grazing\_D + Grazing\_B + Grazing\_Z \\ &\quad - Respiration\_Z_i - Mortality\_Z_i - Grazing\_Z_i + Migration\_Z_i \\ &\quad + Recruitment\ gain\_Z_i - Recruitment\ loss\_Z_i\end{aligned} \tag{8.27}$$

and

$$Grazing\_P = \sum_{j=1}^{np} e_{P_j}(i) G_{P_j}(i) \tag{8.28}$$

$$Grazing\_D = \sum_{j=1}^{nd} e_{D_j}(i) G_{D_j}(i) \tag{8.29}$$

$$Grazing\_B = \sum_{j=1}^{nb} e_{B_j}(i) G_{B_j}(i) \tag{8.30}$$

$$Grazing\_Z = \sum_{\substack{j=1 \\ j \neq i}}^{nz} e_{z_j}(i) G_{z_j}(i) \tag{8.31}$$

$$Respiration\_Z_i = -\gamma_{Z_i} Z_i \tag{8.32}$$

$$Mortality\_Z_i = -\varepsilon_{Z_i} Z_i^{m_{z_i}} \tag{8.33}$$

$$Grazing\_Z_i = -\sum_{\substack{j=1 \\ j \neq i}}^{nz} e_{z_i}(j) G_{z_i}(j) \tag{8.34}$$

$$Migration\_Z_i = -w_{z_i} \frac{\partial Z_i}{\partial z} \tag{8.35}$$

$$Recruitment\ gain\_Z_i = \hat{\gamma}_{Z_{i-1}} \left. \frac{dZ_{i-1}}{dt} \right|_{t-1} \tag{8.36}$$

$$Recruitment\ loss\_Z_i = \hat{\gamma}_{Z_i} \left. \frac{dZ_i}{dt} \right|_{t-1} \tag{8.37}$$

where $e_{P_j}(i)$, $e_{D_j}(i)$, $e_{B_j}(i)$, and $e_{z_j}(i)$ are the grazing efficiencies of Z_i on P_j, D_j, B_j, and other Z_j; γ_{Z_i} is the respiration rate of Z_i; ε_{Z_i} is the mortality rate of Z_i; w_{Z_i} is the vertical migration velocity (which can be specified to be a constant or vertical profile or time-dependent function related to light and food limitation); $\hat{\gamma}_{Z_i}$ is the recruitment success rate of Z_i; and m_{Z_i} is the power of Z_i for mortality. Detailed description of the units used for these parameters is given in Table 8.4

Table 8.4: Parameters used in the zooplankton equation

Symbol	Definition	Unit
$e_{P_j}(i)$	grazing efficiencies of Z_i on P_j	fraction
$e_{D_j}(i)$	grazing efficiencies of Z_i on D_j	fraction
$e_{B_j}(i)$	grazing efficiencies of Z_i on B_j	fraction
$e_{z_j}(i)$	grazing efficiencies of Z_i on Z_j	fraction

γ_{Z_i}	respiration rate of Z_i	s^{-1}
ε_{Z_i}	mortality rate of Z_i	s^{-1}
$\hat{\gamma}_{Z_i}$	recruitment success rate of Z_i	fraction
m_{Z_i}	power of Z_i for the mortality	dimensionless
$g_{\max}^{P_j}(i)$	maximum grazing rate of Z_i on P_j	s^{-1}
K_P	half-saturation constant	mmol C m^{-3}
$\sigma_{P_j}(i)$	preference coefficient of Z_i on P_j	(mmol C m^{-3})$^{-1}$
$\lambda_{P_j}(i)$	Ivelv constant for P_j, grazed by Z_i	(mmol C m^{-3})$^{-1}$
$\alpha_T^{Z_i}$	water temperature influence constant for Z_i grazing	(C°)$^{-1}$
$d_{\max}(i)$	maximum grazing rate of Z_i on D	s^{-1}
$\sigma_{D_j}(i)$	preference coefficient Z_i on D_j	(mmol C m^{-3})$^{-1}$
K_{D_j}	half saturation constant of D_j	mmol C m^{-3}
m_{D_j}	power of D_j used in the grazing function	dimensionless
w_R	random generation within a range from -1 to 1 with 50% probability at each migration time step	dimensionless
$w_{\max}$	maximum vertical migration speed	m s^{-1}
κ_{Z_i}	vertical migration constant	(mmol C m^{-3})$^{-1}$

$G_{P_j}(i)$, $G_{D_j}(i)$, $G_{B_j}(i)$ and $G_{Z_j}(i)$ are the grazing functions of Z_i on P_j, D_j, B_j, and Z_j ($j \neq i$). There are various types of functions used for grazing processes. For example, we can easily find multiple functions used for $G_{P_j}(i)$. The simplest one is the Lotka-Volterra function given as

$$G_{P_j}(i) = g_{P_j} P_j Z_i \tag{8.38}$$

where g_{P_j} is the grazing rate (Chen, 2003). This function represents an unlimited grazing process without a saturation stage. It is used in some theoretical studies of the phytoplankton-zooplankton model, but not in real ocean applications. Attention should be paid when selecting the proper function in the case with multiple types of prey. For these reason, we include various choices of the grazing functions. These functions can be used for the single prey case by setting the prey number to 1. Detailed descriptions of the functions included in the FBM for *P*, *D*, *B*, and *Z* are given next.

a) $G_{P_j}(i)$

A review of the available literature shows that the existing grazing functions can be divided into 3 groups: 1) grazing on a single prey; 2) grazing on multiple prey, and 3) grazing on multiple prey with active switching. In most cases, functions used in group 1 are a special case of the functions in groups 2 and 3. Because many of the functions have similar behavior, we incorporate into FBM some of the functions from groups 2 and 3 to provide users with a suitable range of choices.

In general,

$$G_{P_j}(i) = g_{P_j}(i) Z_i \tag{8.39}$$

where $g_{P_j}(i)$ is the grazing function of Z_i for P_j. The various formulas of $g_{P_j}(i)$ are given in Table 8.5.

Table 8.5: Grazing functions of zooplankton

Name	Function	References
IVLE1_G	$g_{P_j}(i) = g_{\max}^{P_j}(i)(1 - e^{-\lambda_{P_i}(i)P_j})$	Ivlev (1955)
RECTI_G	$g_{P_j}(i) = \left\{ \begin{array}{l} g_{\max}(i)\dfrac{\sigma_{P_j}(i)P_j}{K}, \text{ for } R \le K \\ g_{\max}(i)\dfrac{\sigma_{P_j}(i)P_j}{R}, \text{ for } R > K \end{array} \right\}$, $R = \sum_{j=1}^{np} \sigma_{P_j}(i)P_j$	Armstrong (1994)
CLI_G	$g_{P_j}(i) = g_{\max}(i)\sigma_{P_j}(i)P_j(1 - e^{-\sigma_{P_j}(i)P_j})$	Leonard et al. (1999)

IVLE2_G	$g_{P_J}(i) = g_{\max}(i)\left(1 - e^{-\lambda_P(i)R}\right)\frac{\sigma_{P_j}(i)P_j}{R}$, $R = \sum_{j=1}^{np}\sigma_{P_j}(i)P_j$	Hofmann and Ambler (1988)
MM1_G	$g_{P_j}(i) = g_{\max}(i)\frac{\sigma_{P_j}(i)P_j}{K_P + \sum_{j=1}^{np}\sigma_{P_j}(i)P_j}$	Moloney and Field (1991)
MM2_G	$g_{P_j}(i) = g_{\max}(i)\left(\frac{R - P_c}{K_P + R - P_c}\right)\frac{\sigma_{P_j}(i)P_j}{R}$, $R = \sum_{j=1}^{np}\sigma_{P_j}(i)P_j$	Evans (1988); Lancelot et al. (2000); Leising et al. (2003)
MM3_G	$g_{P_j}(i) = g_{\max}(i)\frac{\sigma_{P_j}(i)P_j}{1 + \sum_{j=1}^{np}\sigma_{P_j}(i)P_j}$	Verity (1991); Fasham et al. (1999); Strom and Loukos (1998); Tian et al. (2001)
SMM_G	$g_{P_j}(i) = g_{\max}(i)\frac{\sigma_{P_i}(i)P_i^2}{K_P\sum_{j=1}^{np}\sigma_{P_j}(i)P_j + \sum_{j=1}^{np}\sigma_{P_j}(i)P_j^2}$	Fasham et al. (1990); Strom and Loukos (1998); Pitchford and Brindley (1999); Spitz et al. (2001)
GSF1_G	$g_{P_j}(i) = g_{\max}(i)\frac{[\sigma_{P_j}(i)P_j]^{m_g}}{\sum_{j=1}^{np}[\sigma_{P_J}(i)P_j]^{m_g}}$	Tansky(1978); Matsuda et al. (1986)
GSF2_G	$g_{P_j}(i) = g_{\max}(i)\frac{(\sigma_{P_j}(i)P_j)^{m_g}}{\left(\sum_{j=1}^{np}\left(\sigma_{P_j}(i)P_j\right)\right)^{m_g}}$	Vance (1978)
GSMM_G	$g_{P_j}(i) = g_{\max}(i)\frac{\{\sigma_{P_j}(i)[P_j - (P_j)_c\}^{m_g}}{1 + \sum_{j=1}^{np}\{\sigma_{P_j}(i)[P_j - (P_j)_c]^{m_g}}$	Gismervik and Andersen (1997)

Note: $g_{\max}(i)$ is the maximum grazing rate of Z_i on P; K_P is half-saturation constant; P_c is the threshold value below which grazing is zero; $\sigma_{P_j}(i)$ is the preference coefficient of Z_i on P_j, $\lambda_{P_j}(i)$ is the Ivlev constant for P_j, grazed by Z_i.

In the code, we also consider the influence of water temperature on grazing. In this case,

$$g_{\max}(i) = \hat{g}_{\max}(i) e^{-\alpha_T^{Z_i}\left|T - T_{opt}(Z_i)\right|} \tag{8.40}$$

where $\alpha_T^{Z_i}$ is the water temperature influence constant for Z_i grazing.

b) $G_{D_j}(i)$

We include two methods to calculate $G_{D_j}(i)$ here. First, it is represented using the switching Michaelis Menten function given as

$$G_{D_j}(i) = d_{\max}(i) \frac{\{\sigma_{D_j}(i)[D_j - (D_j)_c\}^{m_{D_j}}}{K_{D_j} + \sum_{j=1}^{nd} \{\sigma_{D_j}(i)[D_j - (D_j)_c\}^{m_{D_j}}} D_j \tag{8.41}$$

where $d_{\max}(i)$ is the maximum grazing rate of Z_i on D_j, $\sigma_{D_j}(i)$ is the preference coefficient of Z_i on D_j; K_{D_j} is the half saturation constant of D_j and m_{D_j} is the power of D_j. Secondly, we can add $G_{D_j}(i)$ into one component for the total grazing of Z_i [see the description given below in ***e***].

c) $G_{B_j}(i)$

By replacing D with B, (8.41) can be directly used to calculate $G_{B_j}(i)$. The grazing function of Z_i on B_j can be expressed as

$$\boldsymbol{G_{B_j}(i) = b_{max}(i) \frac{\{\sigma_{B_j}(i)[B_j - (B_j)_c\}^{m_{B_j}}}{K_{B_j} + \sum_{j=1}^{nb} \{\sigma_{B_j}(i)[B_j - (B_j)_c\}^{m_{B_j}}} B_j} \tag{8.42}$$

where $b_{\max}(i)$ is the maximum grazing rate of Z_i on B_j, $\sigma_{B_j}(i)$ is the preference coefficient of Z_i on B_j; K_{B_j} is the half saturation constant of B_j and m_{B_j} is the power of

B_j. Alternatively, we can also add $G_{B_j}(i)$ into one component for the total grazing of Z_i [see the description given below in ***e***].

d) $G_{Z_j}(i)$

The grazing of other zooplankton species Z_i on Z_j ($j \neq i$) can be estimated by replacing P_j by Z_j in the grazing functions listed in Table 8.5 or by replacing D_j or B_j in (8.41) or (8.42) by Z_j. In general, to make the code clearer, we prefer to select the same form as those we choose for $G_{P_j}(i)$.

e) Combination of $G_{P_j}(i), G_{D_j}(i), G_{B_j}(i)$ and $G_{Z_j}(i)$

We have listed many choices for calculating $G_{P_j}(i), G_{D_j}(i), G_{B_j}(i)$ and $G_{Z_j}(i)$. If these grazing functions are computed using the different distinct formulas, it would involve a determination of many parameters. To make the code simpler, we also add an expression for the grazing of Z_i on phytoplankton, detritus, bacteria and other zooplankton species together using a generalized switching Michaelis-Menten function given as

$$R_j = \sum_{i=1}^{np} \{\sigma_{P_j}(i)[P_i - (P_j)_c]\}^{m_{P_j}} + \sum_{i=1}^{nd} \{\sigma_{D_j}(i)[D_i - (D_j)_c]\}^{m_{D_j}} + \sum_{i=1}^{np} \{\sigma_{B_j}(i)[B_i - (B_j)_c]\}^{m_{B_j}} + \sum_{\substack{i=1 \\ j \neq i}}^{nz} \{\sigma_{Z_j}(i)[Z_i - (Z_j)_c]\}^{m_{Z_j}} \tag{8.43}$$

and

$$G_{P_j}(i) = \begin{cases} g_i(T) \dfrac{\{\sigma_{P_j}[P_j - (P_j)_c]\}^{m_{P_j}}}{1 + R_j} Z_i & \text{if } P_j > (P_j)_c \\ 0 & \text{if } P_j \leq (P_j)_c \end{cases} \tag{8.44}$$

$$G_{D_j}(i) = \begin{cases} g_i(T) \dfrac{\{\sigma_{D_j}[D_j - (D_j)_c]\}^{m_{D_j}}}{1 + R_j} Z_i & \text{if } D_j > (D_j)_c \\ 0 & \text{if } D_j \leq (D_j)_c \end{cases} \tag{8.45}$$

$$G_{B_j}(i) = \begin{cases} g_i(T) \dfrac{\{\sigma_{B_j}[B_j - (B_j)_c]\}^{m_{B_j}}}{1 + R_j} Z_i & \text{if } B_j > (B_j)_c \\ 0 & \text{if } B_j \leq (B_j)_c \end{cases} \tag{8.46}$$

$$G_{Z_j}(i) = \begin{cases} g_i(T)\dfrac{\{\sigma_{D_j}[Z_j - (Z_j)_c]\}^{m_{Z_j}}}{1+R_j} Z_i & \text{if } Z_j > (Z_j)_c \\ 0 & \text{if } Z_j \le (Z_j)_c \end{cases} \quad (8.47)$$

where

$$g_i(T) = \hat{g}_{\max}(i) e^{-\alpha_T^{Zi}|T - T_{opt}(Z_i)|} \quad (8.48)$$

It should be noted that the half saturation constant is scaled by the preference coefficient in (8.44)-(8.48). Therefore, the preference coefficient described here should be scaled by the half saturation constant.

f) *Grazing* $\_ Z_i$

The loss of Z_i grazed by other zooplankton species should be parameterized using the same form shown above in ***d***.

g) Vertical migration velocity (w_{Z_i})

Many observations reveal diurnal vertical migration in many zooplankton species. The influence of vertical migration on the spatial distribution of zooplankton species has been widely studied in the past twenty years (Anderson and Nival, 1991). The driving mechanism is complicated, which is related to both environmental and biological factors such as light, food abundance, water temperature, salinity, gravity, available dissolved oxygen, turbulence, predation, species, age, sex, state of feeding, reproduction, energy conservation, and biological cycles, etc. (Forward, 1988; Ohman, 1990; Anderson and Nival, 1991). For example, some zooplankton species migrate beneath the euphotic layer to escape predation in daylight and then move back near the surface for feeding during the night (Franks and Widder, 1997). This diurnal migration process can be parameterized by specifying w_{Z_i} using a diurnal period function related to the daily light variation. Since this is case dependent, we leave it as a "user specified" value in the code.

The vertical migration of zooplankton is also related to food abundance. For example, when zooplankton migrate upward to reach the maximum chlorophyll layer, they may not continue towards the surface layer where food is scare. If only food abundance is considered, the vertical migration can be estimated by

$$w_{Z_i} = w_R \cdot w_{\max}(1 - e^{-\kappa_{Z_i} TF}) \tag{8.49}$$

where w_R is a random number in the range of -1 to 1 for 50% probability at each migration time step, $w_{\max}$ is the maximum migration velocity, and κ_{Z_i} is the constant value for the ith zooplankton Z_i and TF represents the total food abundance.

8.1.2.4. Detritus

Detritus refers to fecal pellets, dead phytoplankton and zooplankton, unassimilated phytoplankton, other species of zooplankton and detritus due to zooplankton grazing as

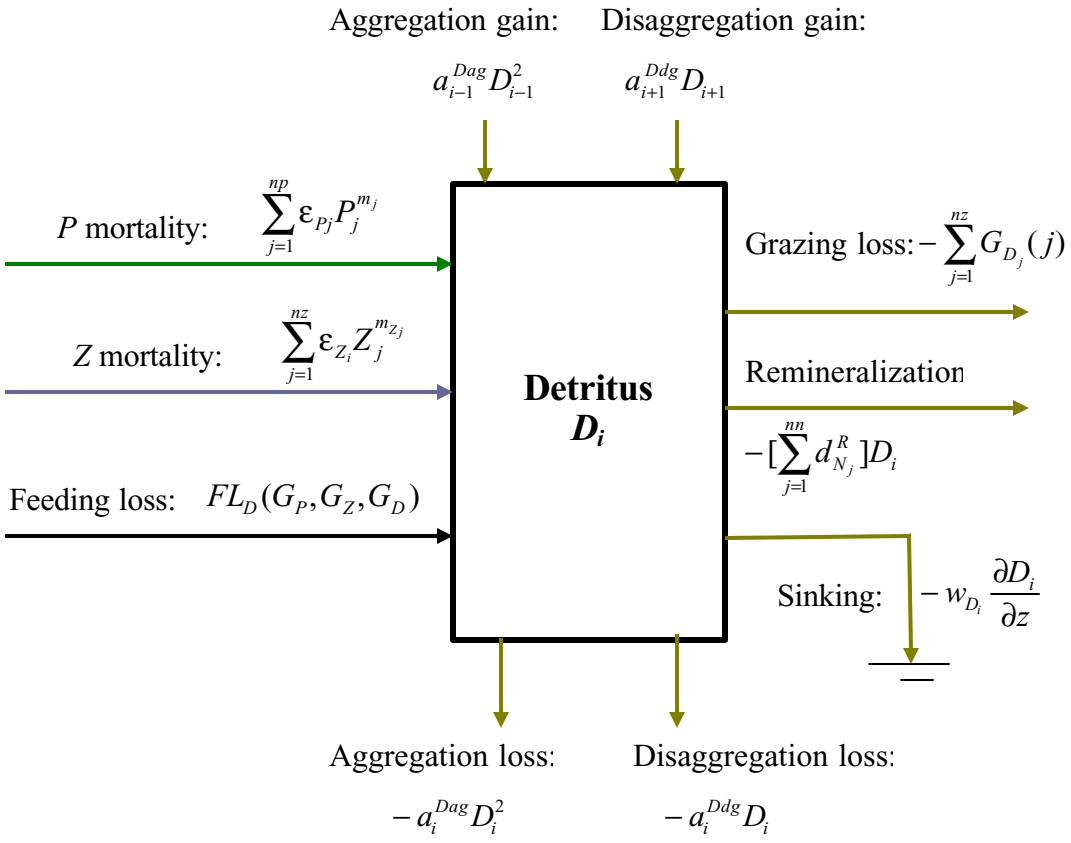


Fig. 8.5: Inflow and outflow chart of detritus D_i

well as aggregation and disaggregration gains of the internal detritus cycling. The loss of detritus is caused by grazing loss by zooplankton on phytoplankton, other zooplankton, and bacteria, remineralization to nutrients, aggregation and disaggregation losses of the

internal detritus cycling and vertical sinking. The inflow and outflow chart describing these processes is shown in Fig. 8.5.

The equation for the balance of these terms are written as

$$\frac{\partial D_i}{\partial t} = P\_mortality + Z\_mortality + Feeding\_loss + Aggregation\_gain + Disaggregation\_gain - Grazing\_loss - Remineralization \quad (8.50)$$

$$- Aggregation\_loss + Disaggregation\_loss + Vertical\_sinking - ADV\_D_i - HDIFF\_D_i - VDIFF\_D_i$$

where

$$P\_mortality = \sum_{j=1}^{np} \varepsilon_{P_j} P_j \quad (8.51)$$

$$Z\_mortality = \sum_{j=1}^{nZ} \varepsilon_{z_j} Z_j \quad (8.52)$$

$$Feeding\_loss = FL_D(G_P, G_z, G_D) = \sum_{k=1}^{nz}[\sum_{j=1}^{np} \alpha_{P_j}^{D}(k) P_j + \sum_{\substack{j=1 \\ j\neq i}}^{nz} \alpha_{Z_j}^{D}(k) Z_j + \sum_{j=1}^{nd} \alpha_{B_j}^{D}(k) D_j] \quad (8.53)$$

$$Aggregation\_gain = a_{i-1}^{Dag} D_{i-1}^2 \quad (8.54)$$

$$Disaggregation\_gain = a_{i+1}^{Ddg} D_{i+1} \quad (8.55)$$

$$Grazing\_loss = -\sum_{j=1}^{nz} G_{D_i}(j) \quad (8.56)$$

$$Remineralization = -[\sum_{j=1}^{nn} d_{N_j}^{R}] D_i \quad (8.57)$$

$$Aggregation\_loss = a_i^{Dag} D_i^2 \quad (8.58)$$

$$Disaggregation\_loss = a_i^{Ddg} D_i \quad (8.59)$$

$$Vertical\_sinking = -w_{D_i} \frac{\partial D_i}{\partial z} \quad (8.60)$$

where $\alpha_{P_j}^{D}(i)$, $\alpha_{Z_j}^{D}(i)$ and $\alpha_{D_j}^{D}(i)$ are coefficients of feeding loss to detritus by zooplankton grazing on P_j, Z_j, and D_j. w_{D_i} is the vertical sinking velocity of D_i. $d_{N_j}^{R}$ is the remineralization rate of D_i to nutrient N_j. The descriptions of parameters used in the detritus equations and their units are given in Table 8.6.

Table 8.6: Parameters used in the detritus equation

Name	Difinition	Unit
$\alpha_{P_j}^{D}(i)$	coefficient of feeding loss to detritus by zooplankton grazing on phytoplankton P_j	fraction
$\alpha_{Z_j}^{D}(i)$	coefficient of feeding loss to detritus by zooplankton grazing on other zooplankton Z_j	fraction
$\alpha_{D_j}^{D}(i)$	coefficient of feeding loss to detritus by zooplankton grazing on detritus D_j.	fraction
w_{D_i}	vertical sinking velocity of D_i.	m s^{-1}
$d_{N_j}^{R}$	remineralization rate of D_i to nutrient N_j.	s^{-1}
$a_{D_i}^{Dag}$	aggregation coefficient of detritus D_i	s^{-1}
$a_{D_i}^{Ddg}$	disaggregation coefficient of detritus D_i	s^{-1}

8.1.2.5 Bacteria

Bacteria contribute to the lower trophic level food web mainly through uptake of nutrients and decomposition of DOM, grazing by microzooplankton and bacteria respiration. In our biological module, no explicit form for the mortality of bacteria is

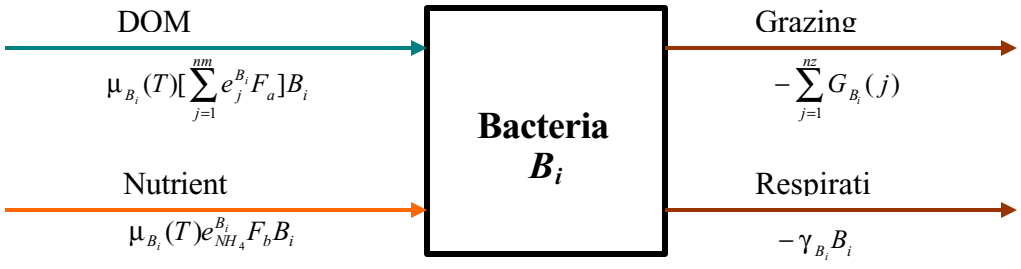


Fig. 8.6: The inflow and outflow chart of the bacteria equation in the FBM.

included. Because the size of individual bacteria is very small, the usual loop in which dead bacteria deposit into the detritus pool and are subsequently remineralized to dissolved nutrients can be simplified by simply including it into the bacteria respiration. For a nitrogen-limited system, the growth of bacteria described here is mainly controlled by DOM and NH_4 uptake. For a phosphate-limited system, the bacteria can also uptake phosphorus. To make the equations more general, we include these two cases in the code.

The bacteria equation can be expressed as

$$\frac{\partial B_i}{\partial t} = Uptake\_DOM + Uptake\_N - Grazing\ loss\_B_i - Respiration\_B_i + ADV\_B_i + HDIFF\_B_i + VDIFF\_B_i \tag{8.61}$$

a) Bacteria uptake

In a nitrogen-limited system,

$$Uptake\_DOM = \mu_{B_i}(T)[\sum_{j=1}^{nm} e_j^{B_i} F_a(DOM_j, NH_4)]B_i \tag{8.62}$$

where $e_j^{B_i}$ is the gross growth efficiency rate of the ith bacteria B_i, $\mu_{B_i}(T)$ is the growth rate of the ith bacteria B_i and T is the water temperature. In the code, $\mu_{B_i}(T)$ is given as

$$\mu_{B_i}(T) = \mu_{\max}^{B_i} e^{-\alpha_T^{B_i}|T-T_{opt}|} \tag{8.63}$$

where $\mu_{\max}^{B_i}$ is the maximum growth rate of the ith bacteria B_i, T_{opt} is the optimal water temperature at which the maximum growth rate occurs and $\alpha_T^{B_i}$ is the exponential decay rate of the temperature limiting factor.

$$F_a(DOM_j, NH_4) = \begin{cases} \dfrac{\sigma_j^{DOM}[DOM_j - (DOM_j)_c]}{1 + \sum_{j=1}^{nm} \sigma_j^{DOM}[DOM_j - (DOM_j)_c] + \Psi(DON, NH_4)} & \text{if } DOM_j > (DOM_j)_c \\ 0 & \text{if } DOM_j \le (DOM_j)_c \end{cases} \tag{8.64}$$

and

$$\Psi(DON, NH_4) =$$

$$\begin{cases} Min\begin{Bmatrix} \sigma_{B_i}^{NH_4}[NH_4 - (NH_4)_c], \\ \delta_{B_i}\sum_{j=1}^{nm}\sigma_j^{B_i}[DON_j - (DON_j)_c] \end{Bmatrix} & \text{if} \quad \begin{matrix} NH_4 > (NH_4)_c \\ DON_j > (DON_j)_c \end{matrix} \\ 0 & \text{if} \quad \begin{matrix} NH_4 \le (NH_4)_c \\ DON_j \le (DON_j)_c \end{matrix} \end{cases} \tag{8.65}$$

where σ_j^{DOM} is the preference coefficient for DOM_j, the subscript "c" means the critical value for the minimum uptake, $\sigma_{B_i}^{NH_4}$ is the preference coefficient for the ith B_i, $\sigma_j^{B_i}$ is the preference coefficient for the jth DON_j, and δ_{B_i} is the uptake ratio of NH_4^+ to DON.

δ_{B_i} can be given as either a constant value or calculated by

$$\delta_{B_i} = \begin{cases} \dfrac{\overline{e}^{B_i}(N:C)_{B_i}}{e_{B_i}^{NH_4}(N:C)_{DOM}} - 1 & \dfrac{\overline{e}^{B_i}(N:C)_{B_i}}{e_{B_i}^{NH_4}(N:C)_{DOM}} > 1 \\ 0 & \dfrac{\overline{e}^{B_i}(N:C)_{B_i}}{e_{B_i}^{NH_4}(N:C)_{DOM}} \le 1 \end{cases} \tag{8.66}$$

where

$$\overline{e}^{B_i} = \frac{1}{nm}\sum_{j=1}^{nm} e_j^{B_i} \tag{8.67}$$

In a nitrogen-limiting system, $Uptake\_N$ is expressed by $Uptake\_NH_4$. The nutrient uptake is considered only in the situation when $\sum_{j=1}^{nm}\sigma_j^{DOM}[DOM_j - (DOM_j)_c] > 0$. In this case,

$$Uptake\_NH_4 = \mu_{B_i}(T)e_{B_i}^{NH_4}F_b(DOM, NH_4)B_i \tag{8.68}$$

where

$$F_b(DOM, NH_4) = \frac{\Psi(DON, NH_4)}{1 + \sum_{j=1}^{nm}\sigma_j^{DOM}[DOM_j - (DOM_j)_c] + \Psi(DON, NH_4)} \tag{8.69}$$

In the case when $Uptake\_DOM$ is equal to zero,

$$Uptake\_NH_4 \equiv 0 \tag{8.70}$$

In a phosphorus-limiting system (like rivers and lakes), Eq. (8.69) remains unchanged except that $F_a(DOM_j, NH_4)$ is replaced by $F_a(DOM_j, PO_4)$. $F_a(DOM_j, PO_4)$ is given as

$$F_a(DOM_j, PO_4) = \begin{cases} \dfrac{\sigma_j^{DOM}[DOM_j - (DOM_j)_c]}{1 + \sum_{j=1}^{nm} \sigma_j^{DOM}[DOM_j - (DOM_j)_c] + \Psi(PO_4)} & \text{if } DOM_j > (DOM_j)_c \\ 0 & \text{if } DOM_j \le (DOM_j)_c \end{cases} \quad (8.71)$$

and

$$\Psi(PO_4) = \begin{cases} \sigma_{B_i}^{PO_4}[PO_4 - (PO_4)_c] & \text{if } PO_4 > (PO_4)_c \\ 0 & \text{if } PO_4 \le (PO_4)_c \end{cases} \quad (8.72)$$

where $\sigma_{B_i}^{PO_4}$ is the preference coefficient for PO_4.

$Uptake\_N$ is expressed by $Uptake\_PO_4$. The nutrient uptake is considered only in the situation when $\sum_{j=1}^{nm} \sigma_j^{DOM}[DOM_j - (DOM_j)_c] > 0$. In this case,

$$Uptake\_PO_4 = \mu_{B_i}(T) e_{B_i}^{PO_4} F_b(DOM, PO_4) B_i \quad (8.73)$$

where

$$F_b(DOM, PO_4) = \frac{\Psi(PO_4)}{1 + \sum_{j=1}^{nm} \sigma_j^{DOM}[DOM_j - (DOM_j)_c] + \Psi(PO_4)} \quad (8.74)$$

b) Grazing loss

The grazing loss is accounted for by the microzooplankton grazing in a form of

$$Grazing\ loss\_B_i = -\sum_{j=1}^{nz} G_{B_i}(j) \quad (8.75)$$

where $G_{B_i}(j)$ is the grazing loss of B_i by the *j*th zooplankton species**.**

c) Respiration loss

The respiration loss of the *i*th bacteria is given as

$$Respiration\ loss\_B_i = -\gamma_{B_i} B_i \quad (8.76)$$

where γ_{B_i} is *i*th bacteria B_i's respiration rate.

Table 8.7: Parameters used in (8.61)-(8.76)

Parameter	Definition	Unit
$e_j^{B_i}$	gross growth efficiency rate of the *i*th bacteria B_i for the *j*th DOM_j	fraction
$\mu_{\max}^{B_i}$	maximum growth rate of the *i*th bacteria B_i	s^{-1}
T_{opt}	optimal water temperature at which the maximum growth rate occurs	°C
$\alpha_T^{B_i}$	exponential decay rate of the temperature limiting factor	$(°C)^{-1}$
σ_j^{DOM}	preference coefficient for DOM_j	$(\text{mmol C m}^{-3})^{-1}$
$\sigma_{B_i}^{NH_4}$	preference coefficient for the *i*th B_i	$(\text{mmol N m}^{-3})^{-1}$
$\sigma_j^{B_i}$	preference coefficient for the *j*th DON_j	$(\text{mmol N m}^{-3})^{-1}$
γ_{B_i}	*i*th bacteria B_i's respiration rate	s^{-1}

8.1.2.6. DOM

The variation of the DOM concentration in the ocean is a complex process. In our

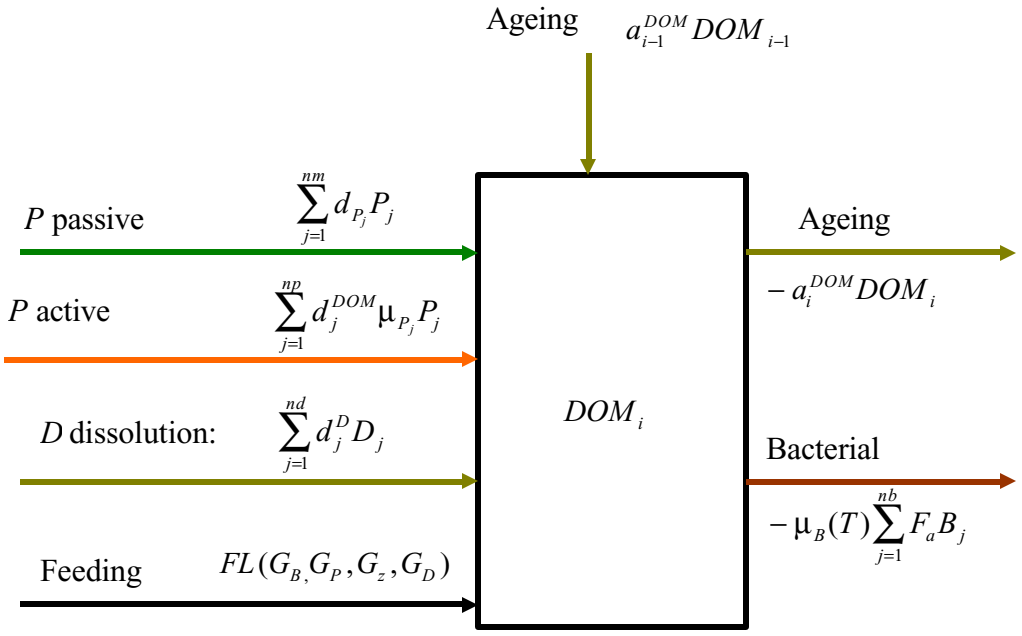


Fig. 8.7: Inflow and outflow chart of DOM_i for the DOM equation.

FBM, the local change of the DOM concentration is controlled by phytoplankton passive and active exudations, detritus dissolution, feeding loss from bacteria, phytoplankton, zooplankton, and detritus, and ageing gain via bacteria uptake and ageing loss. An inflow and outflow chart for DOM_i is shown in Fig. 5 and the equation is given as

$$\frac{\partial DOM_i}{\partial t} = P\_passive\_exudation + P\_active\_exudation + D\_dissolution + Feeding\_loss + Ageing\_gain - Ageing\_loss - B\_uptake + ADV\_DOM_i + HDIFF\_DOM_i + VDIFF\_DOM_i \quad (8.77)$$

where ageing gain and loss are internal processes between multiple DOM_i. In the majority of realistic ocean ecosystem studies, DOM is normally treated as a single variable in which the internal ageing process is not considered. Each term in (8.77) is described below:

$$P\_passive\_exudation = \sum_{j=1}^{np} d_{P_j} P_j \quad (8.78)$$

$$P\_passive\_exudation = \sum_{j=1}^{np} d_j^{DOM_i} \mu_{P_j} P_j \quad (8.79)$$

$$D\_dissolution = \sum_{j=1}^{nd} d_j^D D_j \quad (8.80)$$

$$Ageing\_gain = a_{i-1}^{DOM_i} DOM_{i-1} \quad (8.81)$$

$$Ageing\_loss = -a_i^{DOM_i} DOM_i \quad (8.82)$$

$$B\_uptake = -\mu_B(T) \sum_{j=1}^{nb} F_a B_j \quad (8.83)$$

where $a_i^{DOM_i}$ is the ageing coefficient of DOM_i for the case when DOM is defined based on its bioavailability. The definitions of other parameters are given in the phytoplankton and detritus equations. The feeding loss is the sum of the unassimilated portions of phytoplankton, zooplankton, detritus and bacteria defined as

$$FL(G_P, G_z, G_B, G_D) = \sum_{k=1}^{nz}[\sum_{j=1}^{np}\alpha_{P_j}(k)P_j + \sum_{\substack{j=1 \\ j\neq i}}^{nz}\alpha_{Z_j}(k)Z_j + \sum_{j=1}^{nb}\alpha_{B_j}(k)B_j + \sum_{j=1}^{nd}\alpha_{D_j}(k)D_j] \quad (8.84)$$

where $\alpha_{P_j}(i)$, $\alpha_{Z_j}(i)$, $\alpha_{B_j}(i)$ and $\alpha_{D_j}(i)$ are coefficients of grazing loss by Z_i on phytoplankton, other zooplankton, bacteria and detritus.

The parameters used in this equation are listed in Table. 8.8.

Table 8.8: Parameters used in the DOM equation

Name	Definition	Unit
$a_i^{DOM_i}$	ageing coefficient of DOM_i	s^{-1}
$\alpha_{P_j}(i)$	coefficient of grazing loss by Z_i on phytoplankton	fraction
$\alpha_{Z_j}(i)$	coefficient of grazing loss by Z_i on other zooplankton	fraction
$\alpha_{B_j}(i)$	coefficient of grazing loss by Z_i on bacteria	fraction
$\alpha_{D_j}(i)$	coefficient of grazing loss by Z_i on detritus	fraction

To help users understand the FBM code, we have summarized all variables and parameters into Table. 8.9. Their implementation in the module is the same as that shown in this table.

Table 8.9: Definitions and units of variables and parameters used in the FBM

Name	Definition	Unit
B_i	concentration of the *i*th bacteria	mmol C m^{-3}
D_i	concentration of the *i*th detritus	mmol C m^{-3}
DOM_i	concentration of the *i*th *DOM*	mmol C m^{-3}
N_i	concentration of the *i*th nutrient	mmol C m^{-3}

P_i	concentration of the *i*th phytoplankton	mmol C m^{-3}
Z_i	concentration of the *i*th zooplankton	mmol C m^{-3}
$a_i^{DOM_i}$	ageing coefficient of the *i*th ***DOM***	s^{-1}
α	weight coefficient (0,1) for phytoplankton growth	dimensionless
α_I	light parameter related to the slope of the light function (see in Table 8.3)	dimensionless in MM-, LB-, V65-, PE78-LIGHT; mmol C(mg Chla)$^{-1}$(Wm^{-2})$^{-1}s^{-1}$, in WNS74-, PGH80-, JP76-, BWDC99-LIGHT
a_i^{Dag}	aggregation coefficient of detritus D_i	s^{-1}
a_i^{Ddg}	disaggregation coefficient of detritus D_i	s^{-1}
$\alpha_{B_j}(k)$	coefficient of grazing loss of grazing loss by Z_k on bacteria B_j	fraction
$\alpha_{D_j}(k)$	coefficient of grazing loss of grazing loss by Z_k on detritus D_j	fraction
$\alpha_{P_j}(k)$	coefficient of grazing loss of grazing loss by Z_k on P_j	fraction
$\alpha_{Z_j}(k)$	coefficient of grazing loss of grazing loss by Z_k on Z_j	fraction
$\alpha_{B_j}^D(k)$	unassimilation coefficient for detritus by zooplankton grazing on detritus B_j.	fraction
$\alpha_{D_j}^D(i)$	unassimilation coefficient for detritus by zooplankton grazing on detritus D_j.	fraction
$\alpha_{P_j}^D(i)$	unassimilation coefficient for detritus by zooplankton grazing on phytoplankton P_j	fraction
$\alpha_{Z_j}^D(i)$	unassimilation coefficient for detritus by zooplankton grazing on other zooplankton Z_j	fraction
α_T	exponential decay rate of $\mu_i(T)$ for P	(°C)$^{-1}$
$\alpha_T^{B_i}$	exponential decay rate of the temperature limiting factor for B_i	(°C)$^{-1}$
$\alpha_T^{Z_i}$	exponential decay rate of the temperature limiting factor for Z_i	(°C)$^{-1}$
β	photoinhibition coefficient in PGH80 and BWDC99_LIGHT functions	mmol C(mg Chla)$^{-1}$(Wm^{-2})$^{-1}s^{-1}$
d_j^D	dissolution rate of detritus D_j	s^{-1}
d_i^{DOM}	DOM active exudation of P_i	fraction

$d_{\max}(i)$	maximum grazing rate of Z_i on D	s^{-1}
$d_{N_j}^{R}$	remineralization rate of D_i by nutrient N_j	s^{-1}
d_{P_i}	DOM passive exudation rate of P_i	s^{-1}
$e_j^{B_i}$	gross growth efficiency rate of the ith bacteria B_i for the jth DOM_j	fraction
$e_{B_i}^{NH_4}$	growth efficiency of B_i on NH_4^+	fraction
$e_{B_j}^{PO_4}$	growth efficiency of B_i on PO_4^{3-}	fraction
$e_{B_j}(i)$	grazing efficiencies of Z_i on B_j	fraction
$e_{D_j}(i)$	grazing efficiencies of Z_i on D_j	fraction
$e_{P_j}(i)$	grazing efficiencies of Z_i on P_j	fraction
$e_{z_j}(i)$	grazing efficiencies of Z_i on Z_j	fraction
ε_{P_i}	mortality rate of P_i	s^{-1}
ε_{Z_i}	mortality rate of Z_i	s^{-1}
$g_{\max}^{P_j}(i)$	maximum grazing rate of Z_i on P_j	s^{-1}
$\Gamma_{N_i:C}^{B_j}$	ratio of the ith nutrient N_i unit to carbon for the jth bacteria B_j.	mmol N_i:mmol C
$\Gamma_{N_i:C}^{D_j}$	ratio of the ith nutrient N_i unit to carbon for the jth detritus D_j.	mmol N_i:mmol C
$\Gamma_{N_i:C}^{DOM_j}$	ratio of the ith nutrient N_i unit to carbon for the jth DOM_j.	mmol N_i:mmol C
$\Gamma_{N_i:C}^{P_j}$,	ratio of the ith nutrient N_i unit to carbon for the jth phytoplankton P_j.	mmol N_i:mmol C
$\Gamma_{N_i:C}^{Z_j}$	ratio of the ith nutrient N_i unit to carbon for the jth zooplankton Z_j.	mmol N_i:mmol C
$\Gamma_{N_i:C}^{Z_k B_j}$	ratio of the ith nutrient N_i unit to carbon for active respiration and unassimilated material from Z_k grazing on B_j	mmol N_i:mmol C
$\Gamma_{N_i:C}^{Z_k D_j}$	ratio of the ith nutrient N_i unit to carbon for active respiration and unassimilated material from Z_k grazing on D_j	mmol N_i:mmol C
$\Gamma_{N_i:C}^{Z_k P_j}$,	ratio of the ith nutrient N_i unit to carbon for active respiration and unassimilated material from Z_k grazing on P_j	mmol N_i:mmol C
$\Gamma_{N_i:C}^{Z_k Z_j}$	ratio of the ith nutrient N_i unit to carbon for active respiration and unassimilated material from Z_k grazing on Z_j	mmol N_i:mmol C
γ_{B_i}	respiration rate of the ith bacteria B_i	s^{-1}

γ_{P_i}	respiration rate of the *i*th phytoplankton P_i;	s^{-1}
γ_{Z_j}	passive respiration rate of the *i*th zooplankton Z_i;	s^{-1}
$\hat{\gamma}_{Z_i}$	recruitment success rate of Z_i	Fraction
$I_{\max}$	maximum light intensity at the surface	Wm^{-2}
I_o	light intensity at the sea surface	Wm^{-2}
I_{opt}	optimal light for phytoplankton	Wm^{-2}
k	light attenuation coefficient	m^{-1}
K	saturation for Z grazing in RECTI_G	mmol C m^{-3}
K_{D_j}	half saturation constant of D_j	mmol C m^{-3}
K_{B_j}	half saturation constant of B_j	mmol C m^{-3}
K_I	half-saturation of the light-*P* growth function	Wm^{-2}
K_P	half-saturation for grazing in MM1_G, MM2_G, SMM_G,	mmol C m^{-3}
K_{js}	half-saturation of nutrient N_j	mmol N_j m^{-3}
κ_{Z_i}	vertical migration constant	(mmol C m^{-3})$^{-1}$
$\lambda_{P_j}(i)$	Ivlev constant for P_j grazed by Z_i	(mmol C m^{-3})$^{-1}$
m_I	power of P_i in the mortality term	dimensionless
m_{D_j}	power of D_j used in the grazing function	dimensionless
m_{Z_i}	Power of Z_i for the mortality.	dimensionless
$\mu_{\max}^{B_i}$	maximum growth rate of the *i*th bacteria B_i	s^{-1}
$\mu_{\mathbf{max}}^{P_i}$	maximum growth rate of the *i*th bacteria P_i	mmol C (mg Chla)$^{-1}$ s^{-1}
$\mu_{\max}$	parameter in WNS74_LIGHT, PGH80_LIGHT, BWDC90_LIGHT	mmol C (mg Chla)$^{-1}$ s^{-1}
μ_{P_j}	gross growth rate of phytoplankton P_i	mmol C (mg Chla)$^{-1}$ s^{-1}
μ_I	gross growth rate of phytoplankton P_i	mmol C(mg Chl$^{)-1}s^{-1}$
N	power of light in LB_LIGHT, V65_LIGHT functions	dimensionless
$N_{j\min}$	threshold of nutrient *Nj*	mmol N_j m^{-3}
$(N:C)_{B_i}$	N:C ratio in bacteria B_i	mmol N/mmol C
$(N:C)_{DOM}$	N:C ratio in bacteria DOM_i	mmol N/mmol C
σ_j^{DOM}	preference coefficient for DOM_j	(mmol C m^{-3})$^{-1}$

$\sigma_j^{B_i}$	preference coefficient for the jth DON_j	(mmol N m^{-3})$^{-1}$
$\sigma_{B_i}^{NH_4}$	preference coefficient for the ith B_i	(mmol N m^{-3})$^{-1}$
$\sigma_{B_i}^{PO_4}$	preference coefficient of B_i for PO_4	(mmol P m^{-3})$^{-1}$
$\sigma_{B_j}(i)$	preference coefficient Z_i on B_j	dimensionless
$\sigma_{D_j}(i)$	preference coefficient Z_i on D_j	dimensionless
$\sigma_{P_j}(i)$	preference coefficient of Z_i on P_j	dimensionless
$\theta_{i(chl:N_i)}$	ratio of chlorophyll a to carbon in P_i	mg Chl / mmol C
$T_{opt}^{B_i}$	optimal water temperature at which the maximum growth rate occurs for bacteria B_i	°C
T_{opt}	optimal temperature for phytoplankton	°C
$T_{opt}(Z_i)$	optimal temperature for zooplankton Z_i	°C
TF	total food abundance for vertical migration	(mmol C m^{-3})$^{-1}$
w_{D_i}	vertical sinking velocity of D_i.	m s^{-1}
$w_{\max}$	maximum vertical migration speed	m s^{-1}
w_{P_i}	Sinking velocity of P_I	m s^{-1}
w_R	random generation within a range from -1 to 1 with 50% probability at each migration time step	dimensionless
w_{z_i}	vertical migration velocity of Z_i	m s^{-1}

8.2. Pre-selected Biological Models

FVCOM includes several types of companion finite-volume biological and water quality model modules within FVCOM. These models can be run either online (together with the FVCOM physical model) or offline (driven by FVCOM output). To provide users with information about these biological and water quality models, brief descriptions of their formulations are given in this chapter.

8.2.1. The Nutrient-Phytoplankton-Zooplankton (NPZ) Model

The NPZ model is a simple model developed by Franks et al. (1986). This model was originally coupled into ECOM-si for the study of phytoplankton dynamics in the tidal mixing front on Georges Bank (Franks and Chen, 1996; 2001), and coupled into FVCOM as an option for a simple biological module in 2002. The schematic of the food web loop

of the NPZ model is shown in Fig. 8.8 and the governing equations of this model are given in the form of

$$\frac{dP}{dt}-\frac{\partial}{\partial z}(K_h\frac{\partial P}{\partial z})=\frac{V_m N}{k_s+N}f(I_o)P-ZR_m(1-e^{-\lambda P})-\varepsilon P+F_P \tag{8.85}$$

$$\frac{dZ}{dt}-\frac{\partial}{\partial z}(K_h\frac{\partial Z}{\partial z})=\gamma ZR_m(1-e^{-\lambda P})-gZ++F_z \tag{8.86}$$

$$\frac{dN}{dt}-\frac{\partial}{\partial z}(K_h\frac{\partial N}{\partial z})=-\frac{V_m N}{k_s+N}f(I_o)P+(1-\gamma)ZR_m(1-e^{-\lambda P})+\varepsilon P+gZ+F_N \tag{8.87}$$

where V_{m} the maximum phytoplankton growth rate; k_{s} the half-saturation constant for phytoplankton growth; R_{m} the maximum grazing rate of phytoplankton by zooplankton; λ the grazing efficiency of phytoplankton by zooplankton; γ the fraction of ingested

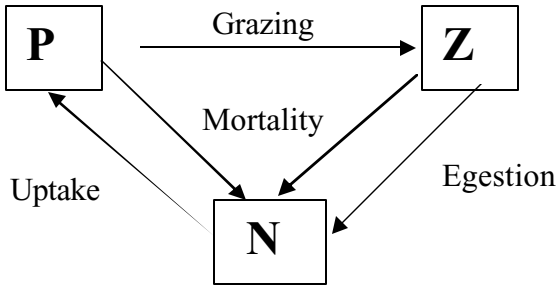


Fig. 8.8: Schematic of the food web loop of the NPZ model

phytoplankton unassimilated by zooplankton; g the zooplankton death rate; ε the phytoplankton death rate; K_h the vertical diffusion coefficient for N, P, and Z; and F_P, F_Z and F_N are the horizontal diffusion terms for P, Z, and N.

This NPZ model presents a simple food web in which dissolved nutrients are taken up by phytoplankton following Michaelis-Menten kinetics, and phytoplankton are grazed by zooplankton through an Ivlev functional response.

Some modifications have been made on the original formulation of the NPZ model used by Franks et al. (1986). First, instead of using a constant mortality rate, we include the Steele and Henderson (1992) and Kawamiya *et al*. (1995) mortality formulas in which ε is proportional to the biomass of P in the form of

$$\varepsilon=\delta P \tag{8.88}$$

where δ is the new (non-constant) mortality rate that was assumed to be constant before. Numerical experiments have revealed that this mortality rate formula is robust enough to model a conservative biological system in conditions without extra sources and sinks. Second, a spatially-dependent diffuse attenuation coefficient of irradiance is used, which allows the model to capture the spatial variation of the photosynthesis process due to the spatial difference in light penetration.

The boundary conditions of N, P, and Z are specified by the users based on the scientific problems that are addressed in the model. For a condition with no flux at the surface and bottom, the boundary condition of N, P, and Z are given as

$$\frac{\partial P}{\partial z} = \frac{\partial Z}{\partial z} = \frac{\partial N}{\partial z} = 0, \quad \text{at } z = \zeta(t,x,y) \text{ and } -H(x,y) \tag{8.89}$$

The initial distributions of biological state variables N, P, and Z must be specified according to either field data or some simple theory. In many cases, they are specified by using the steady state solution of the NPZ model.

The biological parameters used to drive the NPZ model must also be specified. Since these parameters vary over a wide range with time and space, these parameters must be selected from observational data or theoretical consideration. Some examples can be seen in our previous modeling efforts made on Georges Bank (Franks and Chen, 1996 and 2001), Jiaozhou Bay (Chen et al., 1999), and the Louisiana and Texas shelf (Chen et al., 1997), etc.

Equations (8.85)-(8.89) have been converted to the σ-coordinate system and are solved by the finite-volume method using the same basic numerical method used to solve the FVCOM temperature equation.

8.2.2. The Phosphorus-Controlled Lower Trophic Level Food Web Model

A phosphorus-controlled ecosystem model was developed by Chen et al. (2002). The schematic of this model is shown in Fig. 8.9, which was built based on the observed features of the lower trophic level food web in Lake Michigan (Scavia and Fahnenstiel, 1987). The governing equations are given as:

$$\frac{dP_L}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial P_L}{\partial z}) = LP(\text{uptake}) - LP(\text{mortality}) - LZLP(\text{grazing}) - LP(\text{sinking}) \tag{8.90}$$

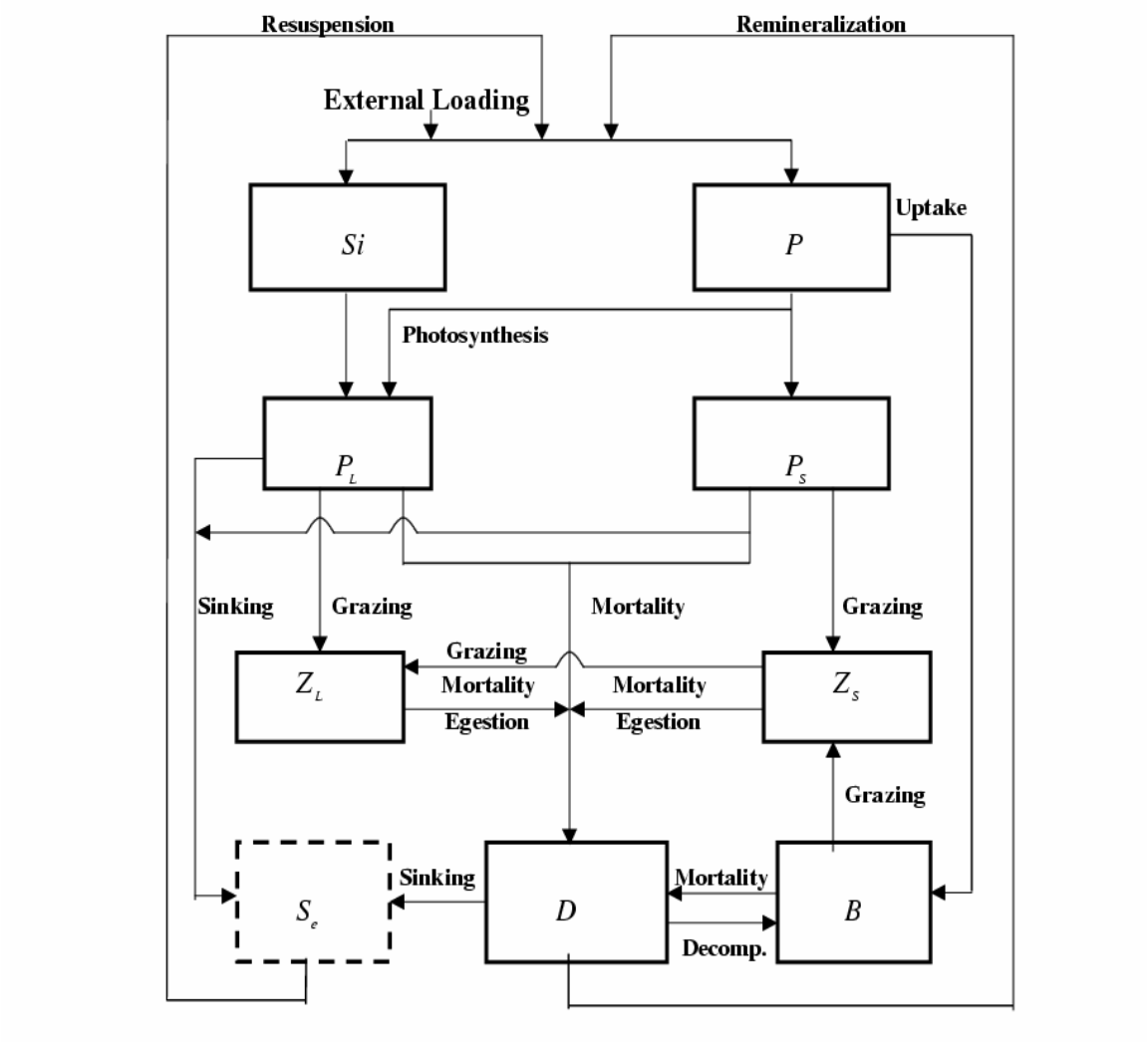


Fig. 8.9: Schematic of the NPZDB model.

$$\frac{dP_S}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial P_S}{\partial z}) = SP(\text{uptake}) - SP(\text{mortality}) - SZSP(\text{grazing}) \tag{8.91}$$

$$\frac{dZ_L}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial Z_L}{\partial z}) = \varepsilon^{Z_L} \sigma_P LZLP(\text{grazing}) + \varepsilon^{Z_{LS}} LZSZ(\text{grazing}) - LZ(\text{mortaity}) \tag{8.92}$$

$$\frac{dZ_S}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial Z_S}{\partial z}) = \varepsilon^{Z_S} SZSP(\text{grazing}) - LZSZ(\text{grazing}) + \varepsilon^{B} SZB(\text{grazing}) - SZ(\text{mortality}) \tag{8.93}$$

$$\frac{dB}{dt}-\frac{\partial}{\partial z}(A_h\frac{\partial B}{\partial z})=DB\ (\text{decomposit ion})+BP\ (\text{uptake})-SZB\ (\text{grazing})-B\ (\text{mortality}) \tag{8.94}$$

$$\frac{dD_S}{dt}-\frac{\partial}{\partial z}(A_h\frac{\partial D_P}{\partial z})=\sigma_S LZLP(\text{grazing})+\sigma_S LP\ (\text{mortality})-DS\ (\text{remineralization}) \tag{8.95}$$

$$\begin{aligned}\frac{dD_P}{dt}-\frac{\partial}{\partial z}(A_h\frac{\partial D_P}{\partial z})=&(1-\varepsilon^{Z_L})\sigma_p LZLP\ (\text{grazing})+(1-\varepsilon^{Z_S})SZSP\ (\text{grazing})\\&+(1-\varepsilon^{B})SZB\ (\text{grazing})+(1-\varepsilon^{Z_{LS}})LZSZ\ (\text{grazing})\\&-DB\ (\text{decomposition})-DP\ (\text{sinking})-DP\ (\text{remineralization})\\&+\sigma_P LP\ (\text{mortality})+SP\ (\text{mortality})+LZ\ (\text{mortality})\\&+SZ\ (\text{mortality})+B\ (\text{mortality})\end{aligned} \tag{8.96}$$

$$\frac{dP}{dt}-\frac{\partial}{\partial z}(A_h\frac{\partial P}{\partial z})=-\sigma_p LP\ (\text{uptake})-SP\ (\text{uptake})-BP\ (\text{uptake})+DP\ (\text{remineralization})+PQ \tag{8.97}$$

$$\frac{dS_i}{dt}-\frac{\partial}{\partial z}(A_h\frac{\partial S_i}{\partial z})=-\sigma_S LP\ (\text{uptake})+DS\ (\text{remineralization})+SQ \tag{8.98}$$

where P_L, P_S, Z_L, Z_S, B, P, and S_i are the large-size phytoplankton, small-size phytoplankton, large-size zooplankton, small-size zooplankton, bacteria, phosphorus, and silicon, respectively. D_P and D_S are the phosphate- and silica-related components of detritus. A_h is the thermal diffusion coefficient that is calculated using the Mellor and Yamada level 2.5 turbulent closure scheme incorporated in the physical model. $\frac{d}{dt}=\frac{\partial}{\partial t}+u\frac{\partial}{\partial x}+v\frac{\partial}{\partial y}+w\frac{\partial}{\partial z}$ is the derivative operator; x, y and z are the eastward, northward, and vertical axes in the Cartesian coordinate frame, and *u*, *v*, and *w* are the *x*, *y*, and *z* components of the velocity. The definition of parameters ε^{Z_L}, $\varepsilon^{Z_{LS}}$, ε^{Z_S}, ε^{B}, σ_S, and σ_P are given in Table 10. PQ and SQ are the phosphorus and silicon fluxes from suspended sediments.

The mathematical formulas for *LP*(uptake), *LZLP*(grazing), *SP*(uptake), *SZSP*(grazing), *LZSZ*(grazing), *SZB*(grazing), *DB*(decomposition), *BP*(uptake), *DP*(remineralization), *DS*(remineralization), *LP*(sinking), *DP*(sinking), *LP*(mortality), *SP*(mortality), *LZ*(mortality), *SZ*(mortality), and *B*(mortality) are given as

$$LP\ (\text{uptake}) = \min\ (V_{\max}^{P_L}\ \frac{P}{k_{P_L}+P}, V_{\max}^{S}\ \frac{S}{k_S+S}) f(I) P_L \tag{8.99}$$

$$LZLP\ (\text{grazing}\ ) = G_{\max}^{Z_L}\ (1-e^{-k^{P_L}P_L}) Z_L \tag{8.100}$$

$$SP\ (\text{uptake}) = V_{\max}^{P_s}\ \frac{P}{k_{P_L}+P}\ f(I) P_s \tag{8.101}$$

$$SZSP\ (\text{grazing}) = G_{\max}^{Z_s}\ (1-e^{-k^{P_s}P_s}) Z_s \tag{8.102}$$

$$LZSZ\ (\text{grazing}\ ) = G_{\max}^{Z_{LS}}\ (1-e^{-k^{Z_{LS}}Z_s}) Z_L \tag{8.103}$$

$$SZB\ (\text{grazing}) = G_{\max}^{B}\ (1-e^{-k^{B}B}) Z_s \tag{8.104}$$

$$BP\ (\text{uptake}) = V_{\max}^{B}\ \frac{P}{k_B+P} B \tag{8.105}$$

$$DB(\text{decomposi tion}) = V_{\max}^{DOP}\ \frac{\mu_D D}{k_{DOP}+\mu_D D} \tag{8.106}$$

$$SZB\ (\text{grazing}\ ) = G_{\max}^{B}\ (1-e^{-k^{B}B}) Z_s \tag{8.107}$$

$$DP(\text{remineralization}) = e_P D_P \tag{8.108}$$

$$DS(\text{remineral ation}) = e_s D_s \tag{8.109}$$

$$LP\ (\text{sinking}) = w_{P_L}\ \frac{\partial P_L}{\partial z} \tag{8.110}$$

$$LP\ (\text{mortality}\ ) = \alpha^{P_L} P_L^2 \tag{8.111}$$

$$SP(\text{mortality}\ ) = \alpha^{P_s} P_s^2 \tag{8.112}$$

$$LZ\ (\text{mortality}\ ) = \alpha^{Z_L} Z_L^2 \tag{8.113}$$

$$SZ\ (\text{mortality}) = \alpha^{Z_s} Z_s^2 \tag{8.114}$$

$$B\ (\text{mortality}\ ) = \alpha^{B} B^2 \tag{8.115}$$

where the definition for each parameter used in the above equations is given in Table 8.10. σ_P and σ_s are the phosphorus and silica fractions of large phytoplankton (diatoms) contained in the total amount of unassimilated zooplankton grazing, respectively, with $\sigma_P + \sigma_s = 1$. The value of σ_P is made according to the observed ratios of carbon to phosphorus in general plants and diatoms and then σ_s is given directly by $1-\sigma_P$.

The dependence of the phytoplankton growth rate on incident irradiance intensity $f(I)$ is given as

$$f(I) = e^{-k_o z} \tag{8.116}$$

where k_o is the diffuse attenuation coefficient. (8.116) is normalized using the surface incident irradiance intensity. In general, the response of phytoplankton to light intensity varies according to different species. For many phytoplankton species, photosynthesis reaches its saturation level at a certain level of light intensity and then is inhibited as light intensity continues to become stronger. The linear assumption of $\ln f(I)$, used widely in previous lower trophic level food web models (Totterdell, 1992), does not consider saturation and inhibition of photosynthesis via light. It is found to be a good approximation in a vertically well-mixed region (Franks and Chen, 1996, Chen et al. 1997, and Chen et al. 1999), but users should be aware of its limitation to reproduce the vertical profile of primary production which normally exhibits a maximum value at a subsurface depth. The Steele and Henderson (1992) mortality formula is used for phytoplankton and zooplankton. This empirical formula assumes that the organism mortality was proportional to its biomass.

Bacteria assimilation of dissolved organic phosphorus (DOP) is also assumed to follow the Michaelis-Menten function, in which DOP is proportional to the total detritus. This assumption is similar to having the bacteria graze detritus directly with a half-saturation constant of k_{DOP}/μ_D. A large portion of bacterial ingestion, which may be excreted into the particular organic pool, is taken into account in our numerical experiments by assuming a larger mortality rate of bacteria (Cotner and Wetzel, 1992).

Default values and their ranges of the biological parameters are listed in Table 8.10. These values were obtained either from field measurements made in Lake Michigan or from the literature. Since the biological parameters can vary in a wide range with time and space, a sensitivity analysis in parameter space should be conducted as part of the parameter setup.

In this model, phosphorus and silicon are the two limiting nutrients that control the primary production. Nitrogen was excluded since it was always found in sufficient

concentration in the Great Lakes. This phosphorus-controlled food web model, in view of the food web system, had an advantage of avoiding the complex ammonia dynamics (Fasham *et al*., 1990).

Table 8.10: Biological Model Parameters

Parameter	Definition	Value used	Ranges	Sources
$V_{\max}^{P_L}$	Maximum growth rate for P_L	1.6 d^{-1}	0.8-6 d^{-1}	Bieman and Dolan (1981); Scavia et al. (1988)
$V_{\max}^{P_S}$	Maximum growth rate for P_S	1.2 d^{-1}	0.8-2 d^{-1}	Bieman &Dolan (1981); Scavia et al. (1988)
$V_{\max}^{S}$	Maximum Si uptake rate by P_L	1.2d^{-1}	0.8-6 d^{-1}	Various sources
V_{max}^{B}	Maximum *P* uptake rate by *B*	0.05d^{-1}	?	
V_{max}^{DOP}	Maximum *DOP* uptake rate by *B*	5d^{-1}	23-144d^{-1}	Bentzen et al. (1992)
k_{P_L}	Half-saturation constant for the *P* uptake by P_L	0.2 µmol P/l	0.07-0.4 µmol P/l	Tilman et al. (1982); Bieman& Dollan (1981)
k_{P_S}	Half-saturation constant for the *P* uptake by P_S	0.05µmol P/l	0.015-?µmol P/l	Bieman &Dollan (1981)
k_S	Half-saturation constant for the *Si* uptake by P_L	5.0 µmol Si/l	3.5-3.57 µmol Si/l	Bieman&Dollan (1981); Jorgensen et al. (1991)
k_B	Half-saturation constant for the *P* uptake by *B*	0.2µmol P/l	0.02-0.2 µmol P/l	Cotner &Wetzel (1992)
k_{DOP}	Half-saturation constant for the *DOP* uptake by *B*	0.1µmol P/l	0.005-0.02 µmol P/l	Bentzen et al. (1992)
$G_{max}^{Z_L}$	Maximum P_L grazing rate by Z_L	0.4d^{-1}	0.2-0.86d^{-1}	Jorgensen et al. (1991); Scavia et al. (1988)
$G_{\max}^{Z_S}$	Maximum P_S grazing rate by Z_S	0.2d^{-1}	0.1d^{-1}	Bieman&Dollan (1981)
$G_{\max}^{B}$	Maximum *B* grazing rate by Z_S	3.5d^{-1}	3.5d^{-1}	Hamilton&Preslan (1970)
$G_{\max}^{Z_{LS}}$	Maximum Z_S grazing rate by Z_L	0.4d^{-1}	?	
k^{Z_L}	Ivlev constant for Z_L grazing	0.06 *l*/µmol	0.001-1 *l*/µmol	Jorgensen et al.(1991); Scavia et al.(1988)

k^{Z_S}	Ivlev constant for P_s grazing by Z_s	0.02 l/μmol	0.011l/μmol	Bierman&Dollan(1981)
k^{B}	Ivlev constant for the B grazing by Z_s	0.03 l/μmol	0.022 l/μmol	Hamilton& Preslan (1970)
$k^{Z_{LS}}$	Ivlev constant for the Z_s grazing by Z_L	0.07	?	
ε^{Z_L}	Assimilation efficiency of Z_L	0.35	0.15-0.5	Jorgensen et al. (1991)
ε^{Z_S}	Assimilation efficiency of Z_s	0.3	?	
ε^{B}	Assimilation efficiency of B grazing by Z_s	0.3	?	
$\varepsilon^{Z_{LS}}$	Assimilation efficiency of the Z_s by Z_L	0.6	?	
α^{Z_L}	Mortality rate of Z_L	0.02d^{-1}	0.01-0.05 d^{-1}	Jorgensen et al. (1991); Bierman& Dollan (1981)
α^{Z_S}	Mortality rate of Z_s	0.03d^{-1}	0.1d^{-1}	Bierman&Dollan(1981)
$V_{\max}^{P_L}$	Mortality rate of B	0.5d^{-1}	0.5-5.9 d^{-1}	Jorgensen et al. (1991)
k_o	Photosynthetic attenuation coefficient	0.08 m^{-1}	0.12-0.17 m^{-1}	Scavia et al. (1986)
μ_D	Proportionality of DOP from the detrital P	0.02	0.1-0.58	Valiela (1995)
V_{max}^{B}	Sinking velocity of P_L	0.6 m d^{-1}	0.5-9 m d^{-1}	Jorgensen et al. (1991); Scavia et al. (1988)
δ_{P_s}	Sinking velocity of P_s	0.3 m d^{-1}	0.01-3 m d^{-1}	Fahnenstiel & Scavia (1987)
δ_D	Sinking velocity of D	0.6 m d^{-1}	0.5-1 m d^{-1}	Jorgensen et al.(1991)
e_P	Remineralization rate of detrital P	0.15d^{-1}	0.05 d^{-1}	Fasham *et al*. (1990)
e_S	Remineralization rate of detrital Si	0.03d^{-1}	?	
α	Temperature dependence coefficient	0.069	0.069	Parsons et al. (1984)
$\lambda_{C:Chl_i}$	Ratio of carbon (C) to chlorophyll	35	23-79	Parsons et al. (1984)

$\lambda_{C:P}$	Ratio of C to P	80	?	Parsons et al. (1984)

8.2.3. The Multi-Species NPZD Model

A lower trophic level food web model was developed for a study of the spring bloom dynamics in the Gulf of Maine/Georges Bank region (Ji, 2003). This is a 9-compartment NPZD model including 3 nutrients (nitrate, ammonia and silicate), 2 phytoplankton (large- and small-size groups), 2 zooplankton (large- and small-size groups), 1 detrital organic nitrogen component and 1 detrital organic silicon component. The schematic of the model is given in Figure 8.10. As an extension of our previous NPZ model, this NPZD model was designed to resolve the seasonal pattern of the phytoplankton, while understanding the limitations of this model in capturing the zooplankton dynamics that are linked directly to population dynamics.

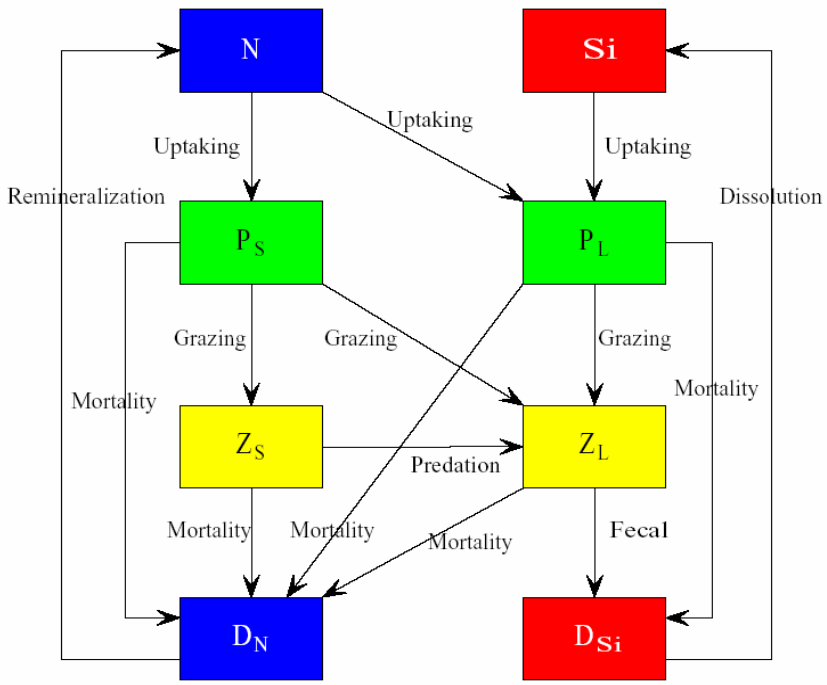


Figure 8.10: Schematic of the flow chart of the multi-species NPZD model.

The governing equations of this 9-compartment model are given as

$$\frac{dP_S}{dt} - \frac{\partial}{\partial z}\left(A_h \frac{\partial P_S}{\partial z}\right) = F3 + F4 - F5 - F6 \tag{8.117}$$

$$\frac{dP_L}{dt} - \frac{\partial}{\partial z}\left(A_h \frac{\partial P_L}{\partial z}\right) = F1 + F2 - F11 - F14 \tag{8.118}$$

$$\frac{dZ_S}{dt} - \frac{\partial}{\partial z}\left(A_h \frac{\partial Z_S}{\partial z}\right) = F5a + F9 - F7 - F8 \tag{8.119}$$

$$\frac{dZ_L}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial Z_L}{\partial z}) = F8a + F14b - F12 \quad (8.120)$$

$$\frac{dNO_3}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial NO_3}{\partial z}) = -F1 - F3 \quad (8.121)$$

$$\frac{dNH_4}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial NH_4}{\partial z}) = F10 - F2 - F4 \quad (8.122)$$

$$\frac{dD_N}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial D_N}{\partial z}) = F5b + F6 + F7 + F8b + F10 + F11 + F12 + F14a - F9 - F10 \quad (8.123)$$

$$\frac{dSi}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial S_i}{\partial z}) = F17 - F13 \quad (8.124)$$

$$\frac{dD_{Si}}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial D_{Si}}{\partial z}) = F15 + F16 - F17 \quad (8.125)$$

where P_S is the small-phytoplankton biomass (µmol N l^{1}); P_L the large-phytoplankton biomass (µmol N l^{1}); Z_S the small-zooplankton biomass (µmol N l^{-1}); Z_L the large-zooplankton biomass (µmol N l^{1}); NO_3 the nitrate concentration (µmol N l^{1}), NH_4 the ammonium concentration (µmol N l^{-1}); D_N the particulate organic nitrogen concentration (µmol N l^{-1}); *Si* the silicate concentration (µmolSi l^{-1}); and *Dsi* the particulate organic silica concentration (µmolSi l^{1}). *F*1 to *F*17 are the flux terms among the food web components defined as

*F*1: Uptake of nitrate by large phytoplankton (µmol N l^{-1}day^{-1})

*F*2: Uptake of ammonia by large phytoplankton (µmol N l^{-1}day^{-1})

*F*3: Uptake of nitrate by small phytoplankton (µmol N l^{-1}day^{-1})

*F*4: Uptake of ammonia by small phytoplankton (µmol N l^{-1}day^{-1});

*F*5: Small zooplankton grazing on small phytoplankton (µmol N l^{-1}day^{-1});

*F*5a: Assimilated part of *F*5 (µmol N l^{-1}day^{-1});

*F*5b: Un-assimilated part of *F*5 (µmol N l^{-1}day^{-1});

*F*6: Mortality of small phytoplankton (µmol N l^{-1}day^{-1});

*F*7: Mortality of large phytoplankton (µmol N l^{-1}day^{-1});

*F*8: Large zooplankton grazing on small zooplankton (µmol N l^{-1}day^{-1});

*F*8a: *Assimilated part of F*8 (µmol N l^{-1}day^{-1});

*F*8b: Un-assimilated part of *F*8 (µmol N l^{-1}day^{-1});

$F9$: Small zooplankton grazing on detritus nitrogen (μmol N $l^{-1}day^{-1}$);

$F10$: Remineralization of particulate organic nitrogen (μmol N $l^{-1}day^{-1}$);

$F11$: Mortality of large phytoplankton (in term of N) (μmol N $l^{-1}day^{-1}$);

$F12$: Mortality of large zooplankton (μmol N $l^{-1}day^{-1}$);

$F13$: Uptake of silicate by large phytoplankton (μmol Si $l^{-1}day^{-1}$);

$F14$: Large zooplankton grazing on large phytoplankton (μmol N $l^{-1}day^{-1}$);

$F14a$: Assimilated part of $F14$ (μmol N $l^{-1}day^{-1}$);

$F14b$: Un-assimilated part of $F14$ (μmol N $l^{-1}day^{-1}$);

$F15$: Mortality of large phytoplankton (in term of Si) (μmol Si $l^{-1}day^{-1}$);

$F16$: Silica rejected from large zooplankton (μmol Si $l^{-1}day^{-1}$);

$F17$: Dissolution of particulate organic silica (μmol Si $l^{-1}day^{-1}$).

The mathematical formulas for $F1$, $F3$, .. $F17$ are given in Ji (2003). Many of them are very similar to those listed in (8.99)-(8.115).

In this model, the small-size phytoplankton group represents nano- and pico-sized phytoplankton, usually flagellates. The growth of small phytoplankton is limited by nitrogen and light. The large phytoplankton size group is explicitly modeled as diatoms that are limited by nitrogen, silicon and light. For application to Georges Bank, the zooplankton are dominated in abundance and biomass by the copepods *Calanus finmarchicus*, *Pseudocalanus sp.*, *Paracalanus parvus*, *Centropages typicus*, *Centropages hamatus*, and *Olithona similes*. At any given time of the year, these six species collectively make up over 80% of the total zooplankton abundance (Davis, 1987). During the spring bloom period, the large zooplankton group in the model represents the dominant species of *Calanus* and *Pseudocalanus*. The small zooplankton group refers to micro-zooplankton with size much smaller than the large zooplankton.

Users should understand that no higher trophic level regulation on zooplankton is included in this model. Therefore, the large and small zooplankton in the model function like a flux balancing terms to maintain the stability of the lower trophic level food web.

8.2.4. The Water Quality Model

The water quality model (WQ) described here is a modified version of the EPA Water Quality Analysis Simulation Program (WASP) (Amborse et al., 1995). The benthic flux

from sediment resuspension via sedimentation processes is added at the bottom to include the impact of the nutrient fluxes from the benthic layer to the water column (Zheng et al.

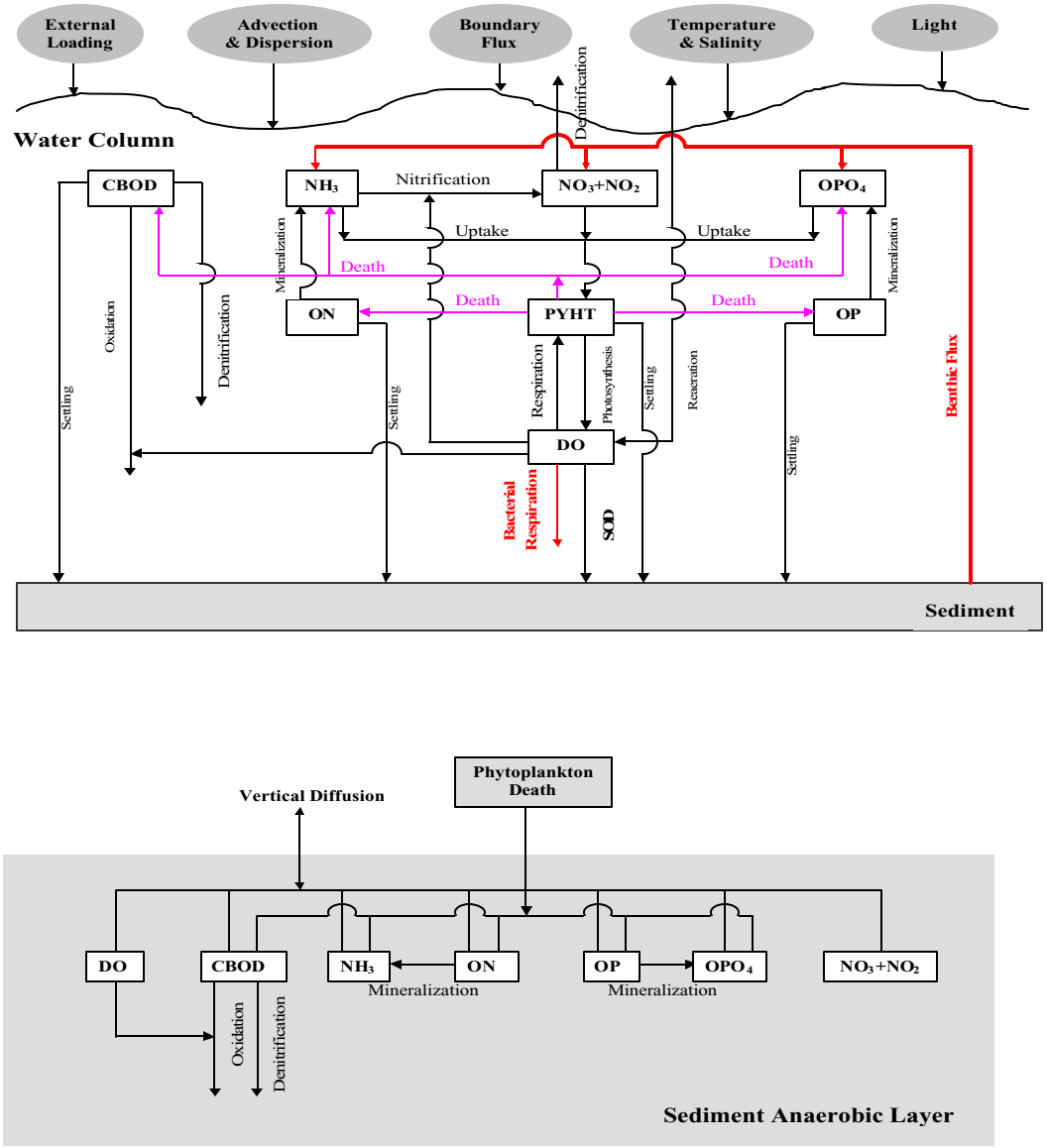


Fig. 8.11. Schematic of the standard EPA water quality model with inclusion of the benthic fluxes.

2004). The schematic of the WQ is shown in Fig. 8.11. This is a typical eutrophication model consisting of eight water quality state variables: (1) ammonia (NH_3); (2) nitrate and nitrite (NO_2 and NO_3); (3) inorganic phosphorus (OPO4); (4) organic nitrogen (ON); (5) organic phosphorus (OP); (6) phytoplankton (PHYT); (7) carbonaceous biochemical oxygen demand (CBOD); and (8) dissolved oxygen (DO). The benthic and sediment

resuspension processes are incorporated into the water model by adding a benthic layer and a sediment pool on the bed of the estuary. This biological/chemical model incorporates the basic transformation processes including photosynthesis, uptake, respiration, nitrification, denitrification, benthic flux, sediment resuspension, and external loading. The modified water quality model was developed based on the characteristics of biological and chemical processes in Georgia estuaries, with the assistance of biologists at the University of Georgia, Skidaway Institution of Oceanography, and the Environmental Protection Agency (EPA) in Athens. The key references include Ambrose et al. (1995), Di Toro et al. (1971), Thomann et al. (1974), Di Toro and Matystik (1980), Di Toro and Connolly (1980), Thomann and Fitzpatrick (1982), and Di Toro and Fitzpatrick (1993).

The governing equations of the WQ model are described in detail in Zheng et al. (2004). The model was converted to the σ-coordinate and then coupled to the ECOM-si physical model by a scientific team led by C. Chen at UGA. The unstructured-grid finite-volume code version of the WQ model was written by L. Qi and C. Chen at SMAST and parallelized by L. Qi and G. Cowles. This program is coupled to FVCOM and can be run in either online or offline mode.

The WQ model was set up in FVCOM as an independent parallelized module. This module was tested by running it for the Satilla River DO, nutrients and phytoplankton simulation. This module is released in version 2.5 of FVCOM.

Chapter 9: The Tracer-Tracking Model

The tracer-tracking equation incorporated in FVCOM is the same as the water temperature equation except with the addition of a source for the tracer. One could easily add this piece of the code by adding a source term in the temperature equation. To reduce any confusion that might arise using the same equation for temperature and tracer, we include a separate module for tracer tracking with the form

$$\frac{\partial DC}{\partial t}+\frac{\partial DuC}{\partial x}+\frac{\partial DvC}{\partial y}+\frac{\partial \omega C}{\partial \sigma}-\frac{1}{D}\frac{\partial}{\partial \sigma}(K_h\frac{\partial C}{\partial \sigma})-DF_c=DC_o(x,y,\sigma,t) \qquad (9.1)$$

where C is the concentration of the tracer, D is the total water depth , $u, v,$ and ω are the x, y and σ components of the water velocity, K_h is the vertical thermal diffusion coefficient, F_c is the horizontal diffusion term, and C_o is the concentration injected from a source point given as

$$C_o(x,y,\sigma,t)=\begin{cases}1 & t_s\le t\le t_e;\sigma_k\le\sigma\le\sigma_{k+n};x=\{x_i\};y=\{y_i\};i=1,N\\ 0 & \text{otherwise}\end{cases} \qquad (9.2)$$

where t_s and t_e are the start and end time of the tracer injection, σ_k and σ_{k+n} refer to the upper and lower-bound σ-levels in which the tracer is injected; n can be an integral from 0 to KB-1 (KB is the total number of the σ-levels specified in the model), i is the node ID, and N is the total node numbers where the tracer is injected. K_h is vertical diffusion coefficient that is calculated using the selected turbulent closure scheme in FVCOM and the horizontal diffusivity in F_c is calculated using the Smagorinsky eddy parameterization method [Smagorinsky, 1963] in FVCOM.

This tracer-tracking module was originally developed when we applied FVCOM to simulate the evolution of a patch of dye that was released near the tidal mixing front on Georges Bank and tracked by R. Houghton (LDGO) as part of the 1999 GLOBEC field program. The code was recently upgraded to run in parallel, so that it can be run simultaneously (online mode) with the hydrodynamic part of FVCOM on single or multi-processor computers or later in offline mode.

In general, after the tracer is injected into the ocean in a field experiment, time series measurements of the tracer concentration are made as the ship attempts to follow the tracer patch in time and space. We have found that in order to make accurate comparisons of the observed tracer concentration data with the model simulation, the model tracer field needs to be sampled the same as the tracer in the field, i.e., to sample the model at the same time and location as the ship sampled the ocean. To facilitate this, we have developed a model-sampling module that

allows us to output the tracer concentration at selected times and locations defined by the ship tracking surveys. This module can be operated in either online or offline mode. At present, this module works for single processor computers, and will be upgraded to parallel operation in the near future.

Chapter 10: The 3-D Lagrangian Particle Tracking

The Lagrangian particle tracking module consists of solving a nonlinear system of ordinary differential equations (ODE) as follows

$$\frac{d\vec{x}}{dt} = \vec{v}(\vec{x}(t),t) \tag{10.1}$$

where $\vec{x}$ is the particle position at a time t, $d\vec{x}/dt$ is the rate of change of the particle position in time and $\vec{v}(\vec{x},t)$ is the 3-dimensional velocity field generated by the model. This equation can be solved using any method suitable for solving coupled sets of nonlinear ODE's. One commonly used class of algorithms is the explicit Runge-Kutta (ERK) multi-step methods which are derived from solving the discrete integral:

$$\vec{x}(t) = \vec{x}(t_n) + \int_{t_n}^{t} \vec{v}(\vec{x}(t),\tau)d\tau \ . \tag{10.2}$$

Assume that $\vec{x}_n = \vec{x}(t_n)$ is the position of a particle at time $t = t_n$, then the new position $[\vec{x}_{n+1} = \vec{x}(t_{n+1})]$ of this particle at time $t = t_{n+1}(= t_n + \Delta t)$ can be determined by the 4th-order 4-stage ERK method as follows:

$$\begin{aligned}
\vec{\xi}_1 &= \vec{x}_n \\
\vec{\xi}_2 &= \vec{x}_n + \frac{1}{2}\Delta t\vec{v}(\vec{\xi}_1) \\
\vec{\xi}_3 &= \vec{x}_n + \frac{1}{2}\Delta\vec{v}(\vec{\xi}_2) \\
\vec{\xi}_4 &= \vec{x}_n + \Delta t\vec{v}(\vec{\xi}_3) \\
\vec{x}_{n+1} &= \vec{x} + \Delta t[\frac{\vec{v}(\vec{\xi}_1)}{6} + \frac{\vec{v}(\vec{\xi}_2)}{3} + \frac{\vec{v}(\vec{\xi}_3)}{3} + \frac{\vec{v}(\vec{\xi}_4)}{6}]
\end{aligned} \tag{10.3}$$

where Δt is the time step. In this calculation, the dependence of the velocity field on time has been eliminated since the velocity field is considered stationary during the tracking time interval of Δt. It is important to understand that in a multidimensional system, the local functional derivative of $\vec{v}$ must be evaluated at the correct sub-stage point $\vec{x}$ in (x, y, z) space.

On a 2-dimensional (x,y) plane, for example, a particle can be tracked by solving the x and y velocity equations given as

$$\frac{dx}{dt} = u, \quad \frac{dy}{dt} = v. \tag{10.4}$$

The 4-stage ERK algorithm can be written out as follows:

$$\begin{cases} \xi_1 = x_n \\ \eta_1 = y_n \end{cases} \tag{10.5}$$

$$\begin{cases} \xi_2 = x_n + \frac{1}{2}\Delta t u(\xi_1, \eta_1) \\ \eta_2 = y_n + \frac{1}{2}\Delta t v(\xi_1, \eta_1) \end{cases} \tag{10.6}$$

$$\begin{cases} \xi_3 = x_n + \frac{1}{2}\Delta t u(\xi_2, \eta_2) \\ \eta_3 = y_n + \frac{1}{2}\Delta t v(\xi_2, \eta_2) \end{cases} \tag{10.7}$$

$$\begin{cases} \xi_4 = x_n + \Delta t u(\xi_3, \eta_3) \\ \eta_4 = y_n + \Delta t v(\xi_3, \eta_3) \end{cases} \tag{10.8}$$

$$\begin{cases} x_{n+1} = x_n + \Delta t[\frac{u(\xi_1, \eta_1)}{6} + \frac{u(\xi_2, \eta_2)}{3} + \frac{u(\xi_3, \eta_3)}{3} + \frac{u(\xi_4, \eta_4)}{6}] \\ y_{n+1} = y_n + \Delta t[\frac{v(\xi_1, \eta_1)}{6} + \frac{v(\xi_2, \eta_2)}{3} + \frac{v(\xi_3, \eta_3)}{3} + \frac{v(\xi_4, \eta_4)}{6}] \end{cases}. \tag{10.9}$$

In the 3-dimensional (x,y,σ) space, a particle can be tracked by solving the x, y, and z velocity equations

$$\frac{dx}{dt} = u, \quad \frac{dy}{dt} = v, \quad \frac{d\sigma}{dt} = \frac{\varpi}{H + \zeta}, \tag{10.10}$$

where u, v, and ϖ are the x, y, and σ velocity components. The relation between ϖ and w is defined as

$$\varpi = w - (2 + \sigma)\frac{d\zeta}{dt} - \sigma\frac{dH}{dt}, \tag{10.11}$$

where w is the vertical velocity in the z coordinate direction. Let us rewrite $\varpi /(H + \zeta)$ as $\hat{\varpi}$. The 4-stage ERK algorithm for this case can be written out as follows:

$$\begin{cases} \xi_1 = x_n \\ \eta_1 = y_n \\ \gamma_1 = \sigma_n \end{cases} \tag{10.12}$$

$$\begin{cases} \xi_2 = x_n + \frac{1}{2}\Delta t u(\xi_1,\eta_1,\gamma_1) \\ \eta_2 = y_n + \frac{1}{2}\Delta t v(\xi_1,\eta_1,\gamma_1) \\ \gamma_2 = \sigma_n + \frac{1}{2}\Delta t \hat{\omega}(\xi_1,\eta_1,\gamma_1) \end{cases} \tag{10.13}$$

$$\begin{cases} \xi_3 = x_n + \frac{1}{2}\Delta t u(\xi_2,\eta_2,\gamma_2) \\ \eta_3 = y_n + \frac{1}{2}\Delta t v(\xi_2,\eta_2,\gamma_2) \\ \gamma_3 = \sigma_n + \frac{1}{2}\Delta t \hat{\omega}(\xi_2,\eta_2,\gamma_2) \end{cases} \tag{10.14}$$

$$\begin{cases} \xi_4 = x_n + \Delta t u(\xi_3,\eta_3,\gamma_3) \\ \eta_4 = y_n + \Delta t v(\xi_3,\eta_3,\gamma_3) \\ \gamma_4 = \sigma_n + \Delta t \hat{\omega}(\xi_3,\eta_3,\gamma_3) \end{cases} \tag{10.15}$$

$$\begin{cases} x_{n+1} = x_n + \Delta t[\frac{u(\xi_1,\eta_1,\gamma_1)}{6} + \frac{u(\xi_2,\eta_2,\gamma_2)}{3} + \frac{u(\xi_3,\eta_3,\gamma_3)}{3} + \frac{u(\xi_4,\eta_4,\gamma_4)}{6}] \\ y_{n+1} = y_n + \Delta t[\frac{v(\xi_1,\eta_1,\gamma_1)}{6} + \frac{v(\xi_2,\eta_2,\gamma_2)}{3} + \frac{v(\xi_3,\eta_3,\gamma_3)}{3} + \frac{v(\xi_4,\eta_4,\gamma_4)}{6}] \\ \sigma_{n+1} = \sigma_n + \Delta t[\frac{\hat{\omega}(\xi_1,\eta_1,\gamma_1)}{6} + \frac{\hat{\omega}(\xi_2,\eta_2,\gamma_2)}{3} + \frac{\hat{\omega}(\xi_3,\eta_3,\gamma_3)}{3} + \frac{\hat{\omega}(\xi_4,\eta_4,\gamma_4)}{6}] \end{cases} . \tag{10.16}$$

Many users have added a random walk-type process into this 3-D Lagrangian tracking code to simulate subgrid-scale turbulent variability in the velocity field. In FVCOM version 2.5, we only include the traditional tracking program as described above. The program can be run on both single and multi-processor computers. However, in the MPI parallel system, tracking many particles simultaneously with the model run on a multi-processor computer can significantly slow down computational efficiency, since particles moving from one sub-domain to another require additional information passing. For this reason, we suggest that users use the offline version of the particle tracking code. Since there are several versions of the offline particle tracking programs built for

FVCOM model output, we have not put the source code in the FVCOM website. For users who want to use our offline program, please contact C. Chen or G. Cowles.

Chapter 11: A triangular finite element sea-ice model for FVCOM

Frédéric Dupont

Quebec-Ocean,
University Laval,
Quebec, QC, Canada

Nicolai Kliem

Danish Meteorological Institute,
Copenhagen, Denmark

1 Introduction

With the advent of triangular finite element and finite volume ocean models, a need has arisen for coupling these models to a thermodynamic-dynamic sea-ice model. We hereafter present a sea-ice model and its implemetation inside FVCOM. We propose a P1-P1 formulation for the dynamics (linear basis functions for velocity and pressure) and P1 (linear) in the vertical for resolving the different layers of the sea ice and the snow cover. The model has been tested and yields reasonable results.

The work by Hibler III (1979); Zhang and Hibler III (1997); Hunke and Dukowicz (1997) are regarded as being the most influential in the sea-ice dynamic modelling community. Hibler III (1979) [with later improvements by Zhang and Hibler III (1997)] introduced a viscous-plastic rheology for the ice and developed an efficient implicit velocity solver for the ice dynamics with timestep as high as one day. On the other hand Hunke and Dukowicz (1997) relaxed the hypothesis of the viscous-plastic rheology by subcycling the viscous-plastic model by an elastic-viscous-plastic model. Although the introduction of the elastic part is somewhat artificial (the reason is only to relax the viscous-plastic rheology for convergence issues) and introduced new prognostic equations for the internal ice stress, the overall efficiency was found to be much better and this formulation tends to be now in favor by most numerical modellers. However the same authors also introduced a symmetric matrix problem to solve for the viscous-plastic solution with conjugate gradient methods. This formulation has its equivalent in finite elements (FE) using the so-called Galerkin formulation.

With respect to the thermodynamics of simple ice models (the ones coupled to ocean or atmospheric models), not much improvement has been done since the work of Semtner (1976) and Parkinson and Washington (1979), with the exception of Winton (2000) who introduced an efficient formulation with among other things a better parametrization of the

brine pockets. Hereafter, we propose a new formulation which also takes advantages of the FE element method but in the vertical direction this time.

2 Dynamic formulation

The dynamical equations are taken from Hibler III (1979); Hunke and Dukowicz (1997) with the common assumption that the momentum advection terms are negligible compared to the Coriolis force, water drag, wind stress and internal stress.

$$\begin{aligned} m\frac{\partial u}{\partial t} =& mfv - \frac{1}{2}\frac{\partial P}{\partial x} + \frac{\partial}{\partial x}\left[(\zeta+\eta)\frac{\partial u}{\partial x} + (\zeta-\eta)\frac{\partial v}{\partial y}\right] \\ &+ \frac{\partial}{\partial y}\left[\eta\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] + \tau_{ai}^{x} + \tau_{iw}^{x} - mg\frac{\partial \varepsilon}{\partial x} \end{aligned} \tag{1}$$

$$\begin{aligned} m\frac{\partial v}{\partial t} =& -mfu - \frac{1}{2}\frac{\partial P}{\partial y} + \frac{\partial}{\partial y}\left[(\zeta+\eta)\frac{\partial v}{\partial y} + (\zeta-\eta)\frac{\partial u}{\partial x}\right] \\ &+ \frac{\partial}{\partial x}\left[\eta\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] + \tau_{ai}^{y} + \tau_{iw}^{y} - mg\frac{\partial \varepsilon}{\partial y} \end{aligned} \tag{2}$$

where m is the mass per unit area, u and v are the two components of the horizontal ice velocity $\mathbf{u}$, P the internal ice pressure as defined by Hibler III (1979), ε is the surface tilt of the ocean, τ_{ai} and τ_{iw} the frictional stress between respectively the atmosphere and the ice and the ice and the ocean. The frictional stresses are expressed in the conventional manner

$$\boldsymbol{\tau}_{ai} = (\tau_{ai}^{x}, \tau_{ai}^{y}) = \rho_a\, A\, Cf_a\, \|\mathbf{u}_a\|\mathbf{u}_a \tag{3}$$

$$\boldsymbol{\tau}_{iw} = (\tau_{iw}^{x}, \tau_{iw}^{y}) = \rho_w\, A\, Cf_w\, \|\mathbf{u}_w - \mathbf{u}\|(\mathbf{u}_w - \mathbf{u}) \tag{4}$$

with A being the ice concentration. The two previous equations can be treated in a complex form, with in particular, complex frictional coefficients Cf_a and Cf_w in order to take into account the turning angle between the two considered media. The diagonal part of the stresses (terms in $\mathbf{u}$) can be moved to the left hand side for increased computational efficiency of iterative solvers (see comment below). We then use a Galerkin weighted-residual approach to solve for these equations:

$$\begin{aligned} \langle m\frac{\partial u}{\partial t}, \phi_i\rangle =& \langle mfv - \frac{1}{2}\frac{\partial P}{\partial x} + \tau_{ai}^{x} + \tau_{iw}^{x} - mg\frac{\partial \varepsilon}{\partial x}, \phi_i\rangle \\ &- \langle \mathbf{G}_x \cdot \nabla\phi_i\rangle + \oint_{\delta\Omega} \mathbf{G}_x \cdot \mathbf{n}\phi_i dl \end{aligned} \tag{5}$$

$$\begin{aligned} \langle m\frac{\partial v}{\partial t}, \phi_i\rangle =& \langle -mfu - \frac{1}{2}\frac{\partial P}{\partial y} + \tau_{ai}^{y} + \tau_{iw}^{y} - mg\frac{\partial \varepsilon}{\partial y}, \phi_i\rangle \\ &- \langle \mathbf{G}_y \cdot \nabla\phi_i\rangle + \oint_{\delta\Omega} \mathbf{G}_y \cdot \mathbf{n}\phi_i dl \end{aligned} \tag{6}$$

where we have integrated by part the viscous stress and we have defined:

$$\mathbf{G}_x = \begin{pmatrix} (\zeta+\eta)\dfrac{\partial u}{\partial x} + (\zeta-\eta)\dfrac{\partial v}{\partial y} \\ \eta\left(\dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x}\right) \end{pmatrix} \quad \text{and} \quad \mathbf{G}_y = \begin{pmatrix} \eta\left(\dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x}\right) \\ (\zeta+\eta)\dfrac{\partial v}{\partial y} + (\zeta-\eta)\dfrac{\partial u}{\partial x} \end{pmatrix} \tag{7}$$

The $\langle u, \phi_i \rangle = \int_\Omega u\phi_i dxdy$ defines the norm or the scalar product inside the domain Ω. In fact, we are going to use free-stress conditions along the open boundary of the domain and no-slip along the coastline which renders the use of the border integral useless. The latter will be removed for now on. By using finite differencing in time (implicit scheme) we can rewrite these equations as

$$\langle m\frac{u^{n+1}}{\Delta t}, \phi_i \rangle + \langle \mathbf{G}_x^{n+1} \cdot \nabla\phi_i \rangle = \langle mfv^n - \frac{\partial P^n}{\partial x} + \tau_{ai}^x + \tau_{iw}^x - mg\frac{\partial \varepsilon^n}{\partial x}, \phi_i \rangle = R_i^u \tag{8}$$

$$\langle m\frac{v^{n+1}}{\Delta t}, \phi_i \rangle + \langle \mathbf{G}_y^{n+1} \cdot \nabla\phi_i \rangle = \langle -mfu^n - \frac{\partial P^n}{\partial y} + \tau_{ai}^y + \tau_{iw}^y - mg\frac{\partial \varepsilon^n}{\partial y}, \phi_i \rangle = R_i^v \tag{9}$$

All the variables will be approximated by continuous linear piecewise basis functions:

$$u^n \approx \sum_{i=1..N} \hat{u}_i^n \phi_i(x,y) \tag{10}$$

where N is the number of nodes in the mesh. We can then show that the viscous influence matrix of velocity at node i on velocity at node j

$$\mathbf{A}_{ij} = \begin{matrix} \\ (u) \\ (v) \end{matrix} \begin{matrix} \begin{matrix} (u) & (v) \end{matrix} \\ \begin{pmatrix} \langle(\zeta^e+\eta^e)\dfrac{\partial\phi_j}{\partial x}\dfrac{\partial\phi_i}{\partial x}\rangle + \langle\eta^e\dfrac{\partial\phi_j}{\partial y}\dfrac{\partial\phi_i}{\partial y}\rangle & \langle\eta^e\dfrac{\partial\phi_j}{\partial x}\dfrac{\partial\phi_i}{\partial y}\rangle + \langle(\zeta^e-\eta^e)\dfrac{\partial\phi_j}{\partial y}\dfrac{\partial\phi_i}{\partial x}\rangle \\ \langle\eta^e\dfrac{\partial\phi_j}{\partial y}\dfrac{\partial\phi_i}{\partial x}\rangle + \langle(\zeta^e-\eta^e)\dfrac{\partial\phi_j}{\partial x}\dfrac{\partial\phi_i}{\partial y}\rangle & \langle\eta^e\dfrac{\partial\phi_j}{\partial x}\dfrac{\partial\phi_i}{\partial x}\rangle + \langle(\zeta^e+\eta^e)\dfrac{\partial\phi_j}{\partial y}\dfrac{\partial\phi_i}{\partial y}\rangle \end{pmatrix} \end{matrix} \tag{11}$$

leads to a general symmetric matrix $\mathbf{A}$ (see Appendix) of rank $2N$, i.e. twice the number of nodes in the mesh, because the two velocity components are coupled. Hence, the Galerkin FE approach to the ice viscous problem leads naturally to a symmetric matrix problem if the viscous coefficients are taken to be piecewise constant inside each element. This is to be contrasted to the method of Hunke and Dukowicz (1997) where the finite difference grid requires that each cell is cut into 4 subtriangles where the viscous stress tensor and viscosities are piecewise constant in each sub-triangle. Here we diagonalize the mass matrix (also referred as "lumping") and we can also take advantage of the stabilizing influence of friction by splitting friction between a diagonal part which can be treated implicitly and a non-diagonal part that can be left on the right hand side. The Coriolis terms are non-symmetric and are left on the right hand side. This is a severe drawback of this

method. However we do not intent to use a timestep larger than one hour and therefore the approximation is crude but not irrelevant.

For the elastic-viscous-plastic rheology, we followed the description of Hunke and Dukowicz (1997) with time-explicit (technically) integration of the equations for both the stresses and the velocity with a sub-timestep. Since the stresses are related to the viscosity, we decided to locate the stresses using a element-piece-constant basis function (just as the viscosities). The model is then initialized with no stresses and the elastic constant is set to $E = 400.0$ which yields the best results in our case. We chose to use a sub-timestep Δt_e of a hundredth of the normal timestep. The equations for the stresses are:

$$\sigma_{ij}^{e\,k+1}\left(\frac{1}{E\Delta t_e}+\frac{1}{2\eta^e}\right)+\frac{\eta^e-\zeta^e}{4\eta^e\zeta^e}(\sigma_{11}^{e\,k+1}+\sigma_{22}^{e\,k+1})\delta_{ij}=\frac{P}{4\zeta^e}\delta_{ij}+\dot{\epsilon}_{ij}^{e\,k} \tag{12}$$

$$\langle \mathbf{L}^k\frac{\mathbf{u}^{k+1}}{\Delta t},\phi_i\rangle+\langle \boldsymbol{\sigma}^{e\,k+1}\cdot\nabla\phi_i\rangle=\langle \mathbf{R}^k,\phi_i\rangle \tag{13}$$

where the superscript e denotes the elementwise value, the superscript k is the time index during elastic sub-timestepping, and $\dot{\epsilon} = 1/2(\partial u_i/\partial x_j + \partial u_j/\partial x_i)$ is the symmetric rate of deformation tensor. $\mathbf{L}$ is 2 by 2 matrix that includes Δt_e, the semi-implicit treatment of the Coriolis and frictions terms. $\mathbf{R}$ is then the right hand side including all remaining terms. For speeding-up the velocity solution in the elastic-viscous-plastic model, the mass matrix terms (first term in left-hand-side of Eq. 13) are lumped (see explanation above).

The ice concentration and ice volume are then advanced using

$$A^{n+1} = A^n - \Delta t\nabla\cdot(\mathbf{u}^n A^n) \tag{14}$$

$$V^{n+1} = V^n - \Delta t\nabla\cdot(\mathbf{u}^n V^n) \tag{15}$$

which is done using a similar finite volume formulation to what is found in FVCOM for fields located at the vertices of the triangular mesh. The mean ice thickness being given by $h = V/A$ and ice mass by $m = \rho_i V$. In computation of the mass term m that gets into the left-hand-side of Equation 1 and 2, we apply a minimum thickness of 0.1 m. The thickness, volume and mass are all expressed in terms of the local value at each node of the mesh, making this discretization similar to the so-called P1-P1 velocity-pressure FE pair. Equation 14 and 15 are advanced using either a pure upwind operator or a corrected upwind scheme (Smolarkiewicz and Szmelter, 2005) in order to minimize the diffusion nature of the upwind operator.

Coeff.	Value	units	Definition
ρ_i	900	kg/m^3	ice density
ρ_s	300	kg/m^3	snow density
ρ_a	1.3	kg/m^3	air density
ρ_w	1025	kg/m^3	water density
C_i	3000	J/kg/K	specific heat capacity of ice
C_w	4190	J/kg/K	specific heat capacity of water
C_a	1004	J/kg/K	specific heat capacity of air
T_o	273.16	K	conversion $^\circ$C $\rightarrow$ K
T_m	–	K	melting temperature of the considered medium
T_{mt}	–	K	melting temperature at $z = h$
T_{ms}	273.15	K	melting temperature of snow
T_{mi}	273.05	k	melting temperature of ice
T_{fw}	271.20	K	temperature at which sea water freezes
T_{2m}	–	K	air temperature at 2m above ground
T_w	–	K	sea surface temperature
c_h, c_e	1.2×10^{-3}	–	transfert coefficient
c_q	1.5×10^{-3}	–	transfert coefficient
ϵ_t	–	–	emissivity of the considered medium
ϵ_i	0.97	–	ice emissivity
ϵ_s	0.99	–	snow emissivity
L_{vap}	2.5×10^{6}	J/kg	latent heat of vaporization
L_{sub}	2.834×10^{6}	J/kg	latent heat of sublimation
L_{fus}	0.334×10^{6}	J/kg	latent heat of fusion
σ	5.67×10^{-8}	W/K^4	Stefan-Boltzman constant
ϵ	0.622	–	molecular weight ratio of water vapor to dry air
c	611.0	Pa	coefficient in calculating saturation vapor pressure
a	21.8746	–	coefficient in calculating saturation vapor pressure
b	265.5	K	coefficient in calculating saturation vapor pressure
α_t, α_t'	–	–	albedo of the considered medium
α_w	0.1	–	albedo of water
α_i	0.6	–	albedo of ice
α_s	0.8	–	albedo of snow
κ_i	2.04	–	ice conductivity
κ_s	0.31	–	snow conductivity
$\mathbf{u}_a$	–	m/s	atmospheric wind velocity at 10 m
$\mathbf{u}_w$	–	m/s	sea surface velocity
$\mathbf{u}$	–	m/s	sea-ice velocity
q_{2m}	–	kg/kg	specific humidity at 2 m
q_0	–	kg/kg	specific humidity at 0 m
p_a	–	Pa	atmopsheric pressure at sea level
F_b	–	W/m^2	ice/ocean heat flux positive downward
F_{ir}^u	–	W/m^2	long wave flux positive upward
F_{ir}^d	–	W/m^2	long wave flux positive downward
F_{sh}	–	W/m^2	sensible heat flux positive downward
F_{lh}	–	W/m^2	latent heat flux positive downward
F_{sw}	–	W/m^2	short wave positive downward before reflection
Cf_a	1.2×10^{-4}	–	air/ice drag coefficient
Cf_w	1.0×10^{-3}	–	water/ice drag coefficient

Table 1
Definition and value of the different coefficients used in the thermodynamics of the ice.

3 Thermodynamics

One (small) improvement here is to use a FE approximation in the vertical for solving the thermodynamic problem. Another is to consider a variable bottom ice temperature, i.e. not fixed to the congelation temperature of sea-water as done in previous models (Semtner, 1976; Winton, 2000). McPhee (1992) suggested to use a variable ice-ocean heat flux due

small scale turbulence. This allows for both the bottom ice and surface water temperatures to adjust to the ice-ocean heat exchange and we will follow this route. However we do not treat explicitly the brine pockets usually present in the ice. On the other hand, in order to mimic the effect of the melt ponds during the melt season, the snow albedo is simply decreased following an arbitrary function of air temperature. We do not take into account the effect of rain over the snow/ice by assuming that it is always snow that falls onto the surface of the snow/ice.

The thermodynamic equation is given by:

$$\rho C \frac{\partial T}{\partial t} = \frac{\partial}{\partial z}\left(\kappa \frac{\partial T}{\partial z}\right) \tag{16}$$

with boundary condition

$$\kappa \frac{\partial T}{\partial z}(z = h) = F(h) \tag{17}$$

$$\kappa \frac{\partial T}{\partial z}(z = 0) = F(0) \tag{18}$$

where $F(z)$ is the total heat flux at different depth, i.e. at $z = h$ it is the heat flux between the atmosphere and the snow/ice and at $z = 0$ it is the heat flux between the snow/ice and the ocean. Equation 16 is treated at each horizontal location as an independent problem and discretized in the vertical using a one-dimensional FE approach and linear basis functions ϕ_k. If we have K layers of ice/snow, then we need $K + 1$ basis functions. Here we will assume that the weight associated with the P1 basis functions are indexed from 0 to K:

$$T(z, t) \approx \sum_{k=0}^{K} \hat{T}_k(t)\phi_k(z) \; . \tag{19}$$

Treating the time discretization in a fully implicit way whenever possible, the weak Galerkin formulation of Equation 16 therefore becomes

$$\langle \rho^e C^e \frac{\hat{T}^{n+1} - \hat{T}^n}{\Delta t}, \phi_k \rangle + \langle \kappa^e \frac{\partial \hat{T}^{n+1}}{\partial z}, \frac{\phi_k}{\partial z} \rangle = [F(z)\ \phi_k(z)]_{z=0}^{z=h} \tag{20}$$

where $\rho^e C^e$ (the density times the heat capacity of each layer) and κ^e (the thermal conduction coefficient) have been approximated by a piecewise constant function inside each element. Following Parkinson and Washington (1979), the sensible fluxes and the outgoing long wave radiation should be treated implicitly for improved stability. Equation 20 is defined for any number of snow/ice layers. However, we will restrict ourselves here to a two layer problem consisting of one layer of snow and one layer of ice, hence $K = 2$.

Many parameters and coefficients are associated to the thermodynamics. For clarity, the definition of the symbols given below are listed and defined in Table 1.

We first deal with the expression for the radiative fluxes. For the reflection of short radiations, we added an albedo function at the top which decreases during melting. The top albedo (either ice or snow) α_t is hence given during the melt season as a function of the atmospheric temperature T_{2m} and the melting temperature of the considered medium (ice or snow) T_{mt}:

$$\text{if} \quad T_{2m} > T_{mt} \quad \alpha_t' = \min(\alpha_w, \alpha_t - 0.12 * (T_{2m} - T_{mt})) \tag{21}$$

The specific humidity at the interface air-snow, q_0, is first obtained through the value of the saturated vapor pressure e_s

$$e_s = c \ \exp\left[a \ \hat{T}_K/(\hat{T}_K + b)\right] \tag{22}$$

$$q_0 = \frac{\epsilon e_s}{p_a - (1-\epsilon)e_s} \tag{23}$$

The different radiative fluxes are given as:

$$F_b = \rho_w C_w c_h \|\mathbf{u}_w\|(\hat{T}_0^{n+1} - T_w) \tag{24}$$

$$F_{ir}^u = \epsilon_t \sigma \left(\hat{T}_K^n\right)^4 + 4\epsilon_t \sigma \left(\hat{T}_K^n\right)^3 \left(\hat{T}_K^{n+1} - \hat{T}_K^n\right) \tag{25}$$

$$F_{sh} = \rho_a C_a \|\mathbf{u}_a\| c_h (T_{2m} - \hat{T}_K^{n+1}) \tag{26}$$

$$F_{lh} = \rho_a L_{sub} \|\mathbf{u}_a\| c_q (q_{2m} - q_0) \tag{27}$$

The total fluxes are then:

$$F(h) = -F_{ir}^u + F_{ir}^d + F_{sh} + F_{lh} + (1 - \alpha_t')F_{sw} \tag{28}$$

$$F(0) = F_b \tag{29}$$

The terms in $\hat{T}_0^{n+1}$ and $\hat{T}_K^{n+1}$ are the ones treated implicitly. Therefore the diagonal terms of the matrix problem will be augmented by the coefficient in front of those terms.

3.1 Melting

The thermodynamics of the ice also involves the transformation of ice to water, snow to ice and water to ice. These transformations are modelled following again Parkinson and Washington (1979). After the energy equation is solved, the case of temperature below the melting temperature is treated. if $\hat{T}_0 > T_{mi}$ then the excess energy is used to melt a portion of the bottom ice, h_{melt}.

$$h_{melt} = 1/2\ h_1 C_i\ (\hat{T}_0 - T_{mi})/L_{fus} \tag{30}$$
$$\Delta h_1 = \Delta h_1 - h_{melt} \tag{31}$$
$$\hat{T}_0 = T_{mi} \tag{32}$$

Note that the $1/2$ ratio accounts for the linear profile of temperature in the medium. The melt water is then taken into account in the heat and freshwater budget of the top computational cell of the underlying ocean. At the surface, if $\hat{T}_K > T_{mt}$, the same treatment leads to:

$$h_{melt} = 1/2\ h_K C_i\ (\hat{T}_K - T_{mt})/L_{fus} \tag{33}$$
$$\Delta h_K = \Delta h_K - h_{melt} \tag{34}$$
$$\hat{T}_K = T_{mt} \tag{35}$$

and can be generalized at the snow/ice interface (or any ice/ice interface) by taking the maximum value of the melting temperature on each side: if $\hat{T}_k > T_m(z^-)$ in ice/snow

$$h_{melt} = 1/2\ h_{k-1} C_i\ (\hat{T}_k - T_m(z^-))/L_{fus} \tag{36}$$
$$\Delta h_{k-1} = \Delta h_{k-1} - h_{melt} \tag{37}$$
$$h_{melt} = 1/2\ h_k C_i\ (\hat{T}_k - T_m(z^-))/L_{fus} \tag{38}$$
$$\Delta h_k = \Delta h_k - h_{melt} \tag{39}$$
$$\hat{T}_k = T_m(z^-) \tag{40}$$

this operation is applied on both sides of the interface. At the surface of the snow, the precipitation is accumulated as snow. Sublimation at the surface of the snow is proportional to the latent heat.

$$\Delta h_K = \Delta h_K + \Delta t F_{lh}/(\rho^e L_{fus}) \tag{41}$$

If the summed ice and snow thickness falls below 10 cm, the ice/snow is redistributed laterally by decreasing the ice concentration and keeping the summed thickness close to 10 cm, thus reducing the extent of ice but conserving ice volume. If the snow thickness is too large, the ice floe dips into the sea by a certain amount (Archimedes's law) and an equivalent portion of the snow is transformed into solid ice.

3.2 Creation of new ice

If the water is not covered by ice and its temperature falls below freezing, the ice concentration is fixed at 97% and the ice thickness is assumed to take of minimum value of 1 mm. Then the excess sea water thermal energy $\rho_w C_w (T_{fw} - T_w)$ in the top computational cell is used to form new ice by congelation of sea water. Sea ice is supposed to be at 4 PSU salinity and this number is used to compute the brine rejection as a negative freshwater flux into the ocean.

4 Application

First, the dynamical part of the sea-ice model is applied in a two-dimensional test-case. The test-case consists on a stack of ice of 80% concentration and 0.625 m mean thickness (0.5 m of mean ice volume). The stack of ice is crushed against a wall by a 20 m/s southerly wind. The ocean is initially at rest. The input files for this test case are located in test_ice and the runfile is named ice_run.dat. FVCOM has to be compiled with the CPP-flag ICE on. The input ice concentration field is provisionnaly in a QUODDY-like format (*.s2r) as well as output field files —locatead in OUTDIR_ICE/sms/. The resulting ice concentration under three rheologies (free-drift, viscous-plastic and elastic-viscous-plastic) is shown after 6 days of simulations (Fig. 1). The west-east desymmetry is due to the Coriolis force (taken at 30° degrees of latitude). The ice is most deformed under free-drift and least deformed under viscous-plastic, the elastic-viscous-plastic rheology giving an intermediate state. This is so because we initiliaze the elastic-viscous-plastic case with no stress. The model is therefore similar to free-drift until the stresses build up and start resisting the compression. The input parameters in ice_run.dat are found provisionnaly in a namelist format until better integrated into FVCOM. The nice feature of this is that at least in absence of this namelist (and of ice) FVCOM can run normally. The namelist is composed of the following parameters and default/user defined values:

```
&ICE_PARAMETERS
DTICE   =   3600.000         ! ice dynamic timestep (s) (default=20 s)
LICE    =           2        ! number of ice/snow layer (default 2=snow+ice)
RHEOLOGY = viscplast         ! ice internal rheology (freedrift, viscplast, EVP)
                             ! (default freedrift)
PSTAR   =   5000.000         ! internal ice pressure (Pa)
CSTRENGTH       =   20.00000 ! parameter in defining internal pressure
HINIT   =   1.000000         ! default ice volume if not defined by an input file
AINIT   =  0.0000000E+00     ! default ice concentation if not defined by an input file
HSINIT  =  0.0000000E+00     ! default snow thickness if not defined by an input file
ECCENT  =   2.000000         ! eccentricity in the ice stress ellipse
ISTEP_EVP = 100              ! number of subtimestep for EVP
EE      =   400.0000         ! Elastic constant in EVP
OUTFIELD = ai   hi   ui      ! prefix names of output field (ai for concentration,
                             ! hi for volume and ui for velocity)
                             ! (default empty)
DT_CLIM =      864000        ! time relaxation (s) for enforcing open boundary conditions (not used)
ADVECTION_SCHEME = UPSTR     ! advection scheme for concentration and volume
                             ! UPSTR or UPCOR
                             ! (default UPSTR)
TYPE_ICE        = 2          ! 0=no ice, 1=static, 2=dynamic
ICEFILE = ./test_ice/ai.s2r  ! initial ice concentration. If given by "ai" prefix
                             ! a "hi" file for volume should follow
                             ! (default empty)
THETA   =  0.5000000         ! semi-implicit factor for friction
ICE_EPS =  1.0000000E-12     ! terminal value for Conjugate Gradient (viscous-plastic only)
ICE_KMAX = 200               ! max number of iteration in Conjugate Gradient (viscous-plastic only)
/
```

The default values are used if no other value is provided by the user. The default values are either explicitly listed on the left or given on the right in the explanatory note if the value on the left is not the default one.

A second application involves testing the thermodynamic part of the sea-ice model [1] . We

[1] This application was not done inside FVCOM as provision has not been made in FVCOM to provide all the necessary atmospheric forcing fields for a complete thermodynamic ice. The application (and code) is however provided for people interested. It is left to them to connect the code to the provided thermodynamic routines and make the appropriate changes in the atmospheric forcing!

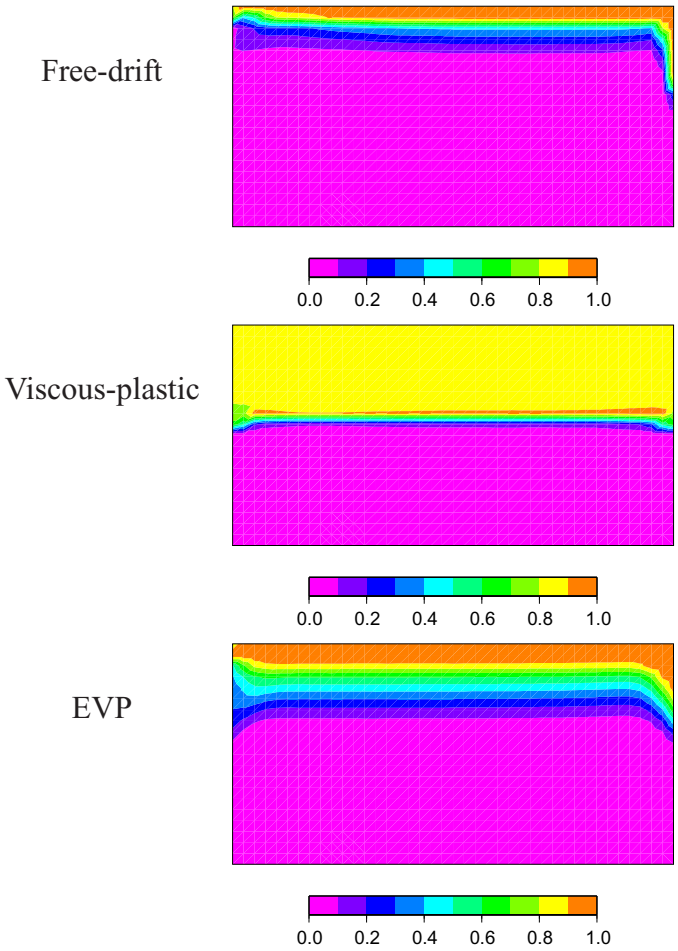


Fig. 1. Ice concentration after 6 days of crushing the ice against the northern wall.

choose here to apply the sea-ice model close to the Resolute Station in the Canadian Archipelago (110.89°W, 73.94°N) where the topographic influence of Greenland on the atmospheric fields is assumed to be less problematic. The model is forced by 6 hourly ECMWF reanalysis and coupled to a one dimensional model of the ocean with a MY2.5 (Mellor and Yamada, 1982) turbulence closure. The surface temperature from ECMWF reanalysis has been corrected for the geopotential height of the lowest level –which does not necessarily conform to the true topography or to sea level, especially around Greenland– with a lapse rate of 0.006 K/m (Greg Holloway, pers. comm., 2005). The model was started from an linear profile of temperature and salinity on September,1 1989 and no ice. The timestep was set to one hour. Figure 2 compares the model result with observations of ice thickness taken at Resolute and Alert (the observations were obtained through the Canadian Ice Service website). Overall the model seems to overpredict the ice thickness in 1991 and to underpredict it in 1997 and 1999. The fact that the interannual signal is not always well captured points to inaccuracies in the atmospheric forcing or the lateral transport of sea water properties as the possible source of discrepancy. For instance, increased snow fall enhances the insulation of ice by snow which can explain between 10 to 40 cm variation in ice thickness from year to year. Note that the model shows a significant trend of ice thinning over a decade of simulation. Similar simulations have been performed for different

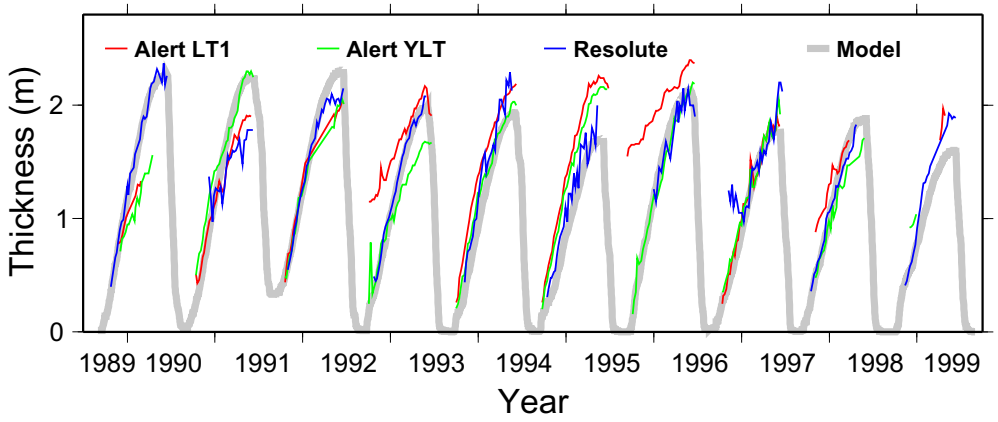


Fig. 2. Comparison between the model and observations. The model at Resolute Station is in thick gray. The ice thickness data at ALert LT1 (red) and YLT (green) and Resolute YRB (blue) was downloaded from the Canadian Ice Service (CIC) web site: http://ice-glaces.ec.gc.ca. The station codes are from CIC.

other locations and the trend seen at Resolute is also visible to the west but not to the north or to the east. One of these other locations is for instance Alert, the northermost station maintained by the Canadian services. The model (not shown) there tends to overpredict the ice, which would imply that there are either other biases in the ECMWF reanalysis due to the proximity of Greenland ice sheet, or some local factors, likely ocean currents which are bringing a substantial amount of heat. In general, the fact that the heat flux between the ocean and the ice is parametrized using a standard turbulent sensible heat flux formula (and the fact that the bottom ice temperature and surface ocean temperature adjust to this flux) enables an average total 20-25 W/m^2 to pass from the ocean to the ice –an order larger than usually expected (Parkinson and Washington, 1979) but not inconsistent with more recent estimates of Perovich and Elder (2002). This large heat flux drives higher bottom ice temperature but also drives lower top ocean temperature and therefore favors in the end thicker ice. The replacement of the one dimensional ocean model with a slab ocean with a fixed thickness tends to deteriorate the simulation by increasing on average the ice thickness by about 50 cm. The continuous warming due to the heat flux from the lower ocean seems to be necessary for a realistic simulation. Finally, the albedo parametrization (introduced in Eq. 21) has quite a major role during the melt season. Without this term, the ice does not as nearly disappear during the summer time as it should and it then actually drift to unrealistic values during the growth season up to 5 m (not shown).

5 Conclusion

We showed some initial results using a triangular finite element model for modelling the dynamics and thermodynamics of sea-ice. The model yields reasonable results. A viscous-plastic and elastic-viscous-plastic rheology were implemented. Future improvements include adding a particle in cell (PIC) method for treating the advection in a Lagrangian manner (Flato, 1993; Sayed et al., 2002). The PIC method should be of particular inter-

est in constricted ice flows. Another plan is to add the granular rheology of Tremblay and Mysak (1997) which is an extension of the cavitating ice family (Flato and Hibler III, 1992).

Final note to the users/developers: the sea-ice thermodynamic presented here is only a suggestion. Users are welcome to use their own or other thermodynamic routines as the one described here was meant to be extremely simple.

Appendix

The Matrix $\mathbf{A}$ is symmetrix if we can prove that $\mathbf{A}_{ij} = A_{ji}$. $\mathbf{A}_{ji}$ is given by

$$\mathbf{A}_{ij} = \begin{pmatrix} \langle(\zeta^e + \eta^e)\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_j}{\partial x}\rangle + \langle\eta^e\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial y}\rangle & \langle\eta^e\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_j}{\partial y}\rangle + \langle(\zeta^e - \eta^e)\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial x}\rangle \\ \langle\eta^e\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial x}\rangle + \langle(\zeta^e - \eta^e)\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_j}{\partial y}\rangle & \langle\eta^e\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_j}{\partial x}\rangle + \langle(\zeta^e + \eta^e)\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial y}\rangle \end{pmatrix} \quad (42)$$

The diagonal terms are obviously equal. For the off-diagonal terms, we need to show that $\mathbf{A}_{ij}(1,2) = A_{ji}(2,1)$. in fact, this is only a point of reordering the partial derivatives inside each term.

$$\mathbf{A}_{ji}(2,1) = \langle\eta^e\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial x}\rangle + \langle(\zeta^e - \eta^e)\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_j}{\partial y}\rangle = \langle\eta^e\frac{\partial\phi_j}{\partial x}\frac{\partial\phi_i}{\partial y}\rangle + \langle(\zeta^e - \eta^e)\frac{\partial\phi_j}{\partial y}\frac{\partial\phi_i}{\partial x}\rangle \quad (43)$$

is indeed $\mathbf{A}_{ij}(1,2)$.

Acknowledgements

Special thanks to Mohamed Sayed who provided the finite difference viscous-plastic sea-ice model. Financial support was provided by Simon Prinsenberg through a PERD grant and from Quebec-Ocean through a FQRNT-center grant.

Chapter 12: The Code Parallelization

The FVCOM code has been parallelized using a Single Processor Multiple Data (SPMD) approach. The domain is decomposed using the METIS graph partitioning libraries. The interprocessor communication is explicitly defined using Message Passing Interface (MPI) calls. The resulting implementation is highly portable and will run efficiently on a variety of parallel computer architectures including both shared and distributed memory systems. The basic elements of the parallelization are as follows.

1) *Domain Decomposition*: The domain (grid) is decomposed into *N* equal partitions (subdomains) where *N* is the number of processors to be used for the computation.

2) *Domain Setup*: Each processor sets up an FVCOM integration in its respective subdomain.

3) *Data Exchange*: During the calculation, information is exchanged between processors across subdomain boundaries to ensure correctness of the boundary fluxes.

4) *Data Collect*: Output data is collected from individual processors and reconstructed into a global array before being written to disk.

12.1. Domain Decomposition

The domain decomposition is performed using the METIS graph partitioning libraries (Karypis and Kumar, 1998). Given a list of elements, information about their connectivity and a user input desired number of partitions; METIS is tasked to assign elements to partitions under the following constraints.

1) Each partition will contain roughly the same number of elements.
2) The total length of the boundary between partitions is to be minimized.

The first constraint pertains to the concept of load balancing. In a code such as FVCOM where the computational effort is dominated by explicit integration of the primary equations, the work required is roughly proportional to the number of elements (triangles) in a domain. Thus to ensure equal workload among the processors, the decomposition must provide the same number of elements to each partition. The second constraint is introduced to reduce communication overhead. Communication of data between processors must be performed to ensure correctness of the flux at the interprocessor

boundary. This communication represents overhead in a parallel program and directly contributes to a reduction in the efficiency of the parallel implementation. Efforts must always be made to minimize it. The volume of communication (bytes/iteration) is proportional to total length of the interprocessor boundary. With the second constraint in the domain decomposition, the communication volume is minimized.

Fig. 12.1 shows a 16 way partitioning of the Gulf of Maine/Georges Bank model domain. Each color represents the subdomain assigned to a given processor. Note that it is not the geographical area but rather the number of elements that is equal in each subdomain. Partitioning is performed in the horizontal only. The implicit nature of the discretization of the vertical diffusion terms in the FVCOM model make a domain decomposition in the vertical impractical. The partitioning performed by METIS occurs at the beginning of the calculation and requires a trivial amount of time to complete. Statistics on the partitions, including load balance and the number of elements assigned to each processors are written to standard output for reference.

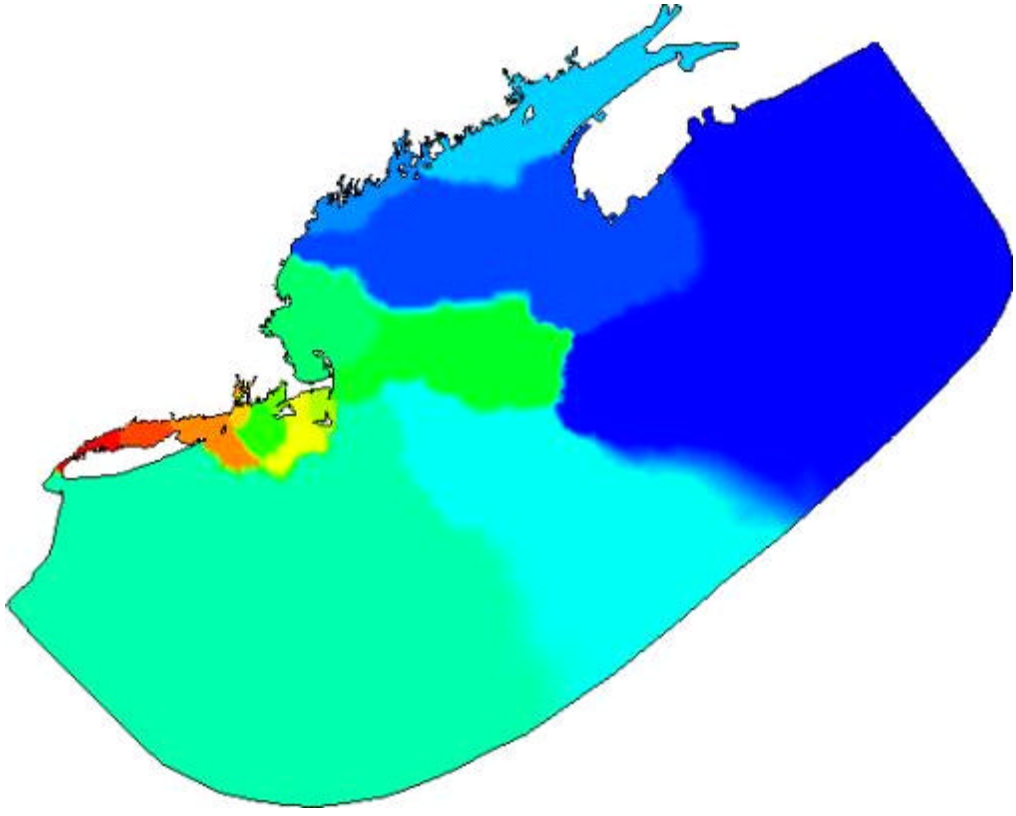


Fig.12.1: 16 way partitioning of the Georges Bank/Gulf of Maine

12.2. Domain Setup

After the domain decomposition stage, each processor has been assigned a subdomain in which to integrate FVCOM. Array mappings can then be calculated to map global indices to local and vice versa. Other maps are created to coordinate data exchange across the interprocessor boundaries. Globally referenced data such as rivers, open boundary nodes, and the grid file are decomposed into the local domains using the global to local mappings. For example, if a river runs into a given processor's subdomain, this processor will read in the river inflow data and assign it to the locally numbered node number which corresponds to the inflow river point. The other processors will ignore the data. Open boundaries and spatially varying surface forcing are treated in a similar manner. At the conclusion of this setup stage, each processor has the correct initial and boundary conditions with which to drive a full integration of its subdomain.

12.3. Data Exchange

At the boundaries between processor subdomains, data must be exchanged to preserve the correctness of the flux. The data to be exchanged is set up in a mapping procedure where interior nodes of neighboring processors along the interprocessor boundaries are mapped to the corresponding halo nodes of the exchange partner and vice-versa. For example, in Fig. 12.2, computation of the flux of element E for the edge residing on the interprocessor (dark line) boundary requires information in elements H. Elements H belong to Processor 2 (P2) but they are considered halo nodes of Processor 1 (P1). Information on the current state of flow quantities in these elements must be

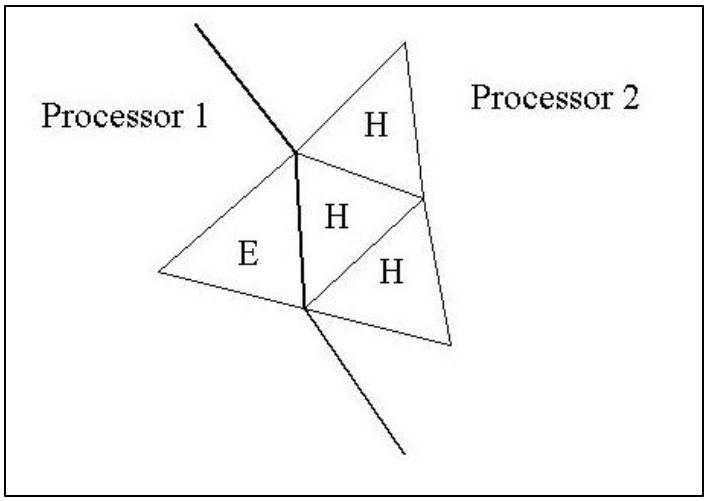


Figure 12.2: Illustration of the data exchange between two decomposition domains.

provided by P2 in order for P1 to compute and update the fluxes of E correctly. This information is provided in an explicit interprocessor communication. Interprocessor communication of data is made using standard MPI (Message Passing Interface) non-blocking send/receive calls. The exchange subroutine is generic and can be used to exchange both element-based and node-based data for both vertically-averaged and three-dimensional flow quantities.

12.4. Data Collection

Flow field output must be performed globally. Thus the data residing in the local processor subdomains must be collected to an aggregate array corresponding to the global element/node numbering defined by the grid file. In general the true values of this array are known only to the master processor that writes the data and discards the array before proceeding with the calculation. Data collection is performed using blocking MPI send/receive pairs.

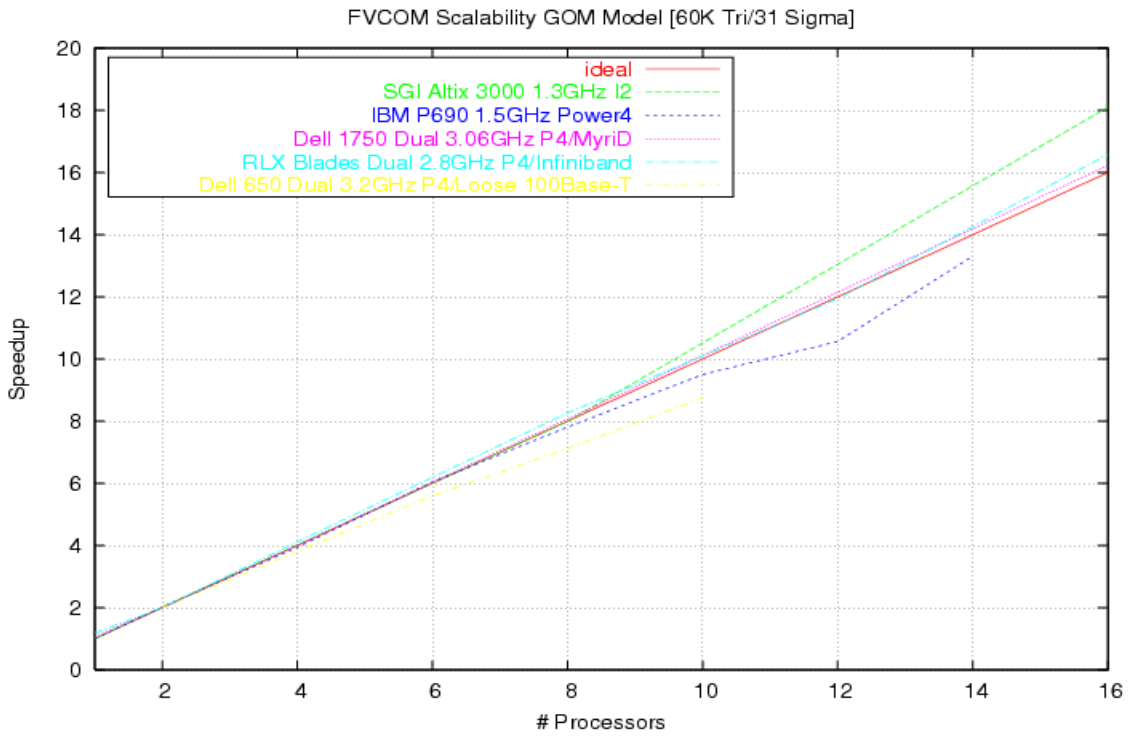


Figure 12.3: shows the speedup of FVCOM on machines with small number of processors (< 16).

12.5. Performance

The efficiency of the parallel implementation can be analyzed be evaluating the performance speedup. A given model run is performed on an increasing number of processors and the time to complete the run is recorded. Speedup (S) is defined as the

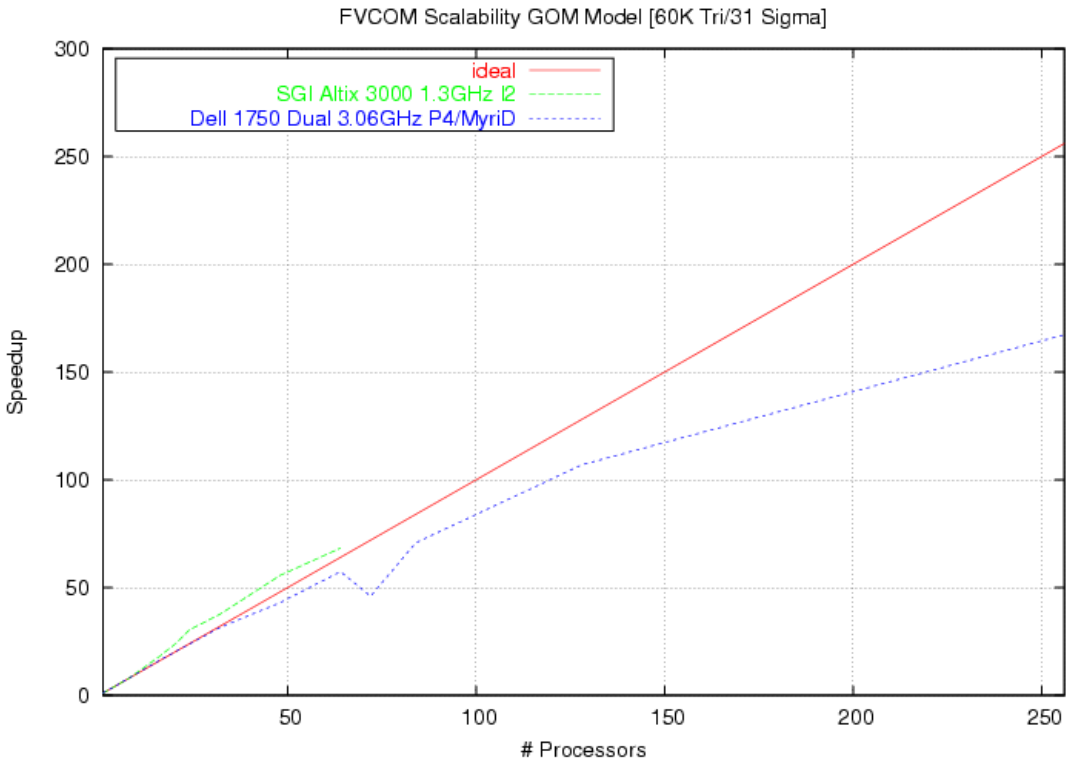


Figure 12.4: shows the speedup of FVCOM on machines with large number of processors.

ratio of time to complete a job on one processor (serial) divided by the time to complete the job on N processors. Ideally, the curve should be a straight line defined by the equation $S(N) = N$. Various factors combine to modify code efficiency, sometimes resulting in superlinear ($S(N) > N$) speedup. Fig.12.3 shows the speedup measured on systems with small (< 16) number of processors. The curves do not stray far from the ideal line. The SGI Altix exhibits superlinear speedup due to its efficient network, which reduces communication overhead, and the large cache size (3 MB) of the Itanium 2 processor that boosts performance as partition size is decreased. The yellow line corresponds with a loosely coupled cluster made of the desktop computers in the MEDM

lab at SMAST, which are connected by 100BaseT Ethernet. In this case the bandwidth and latency limitations of the interconnect generates significant interprocessor communication overhead, resulting in a significant drop in the parallel efficiency. In the case with large number of processors (Fig. 12.4): the SGI Altix maintains linear speedup up to the maximum number of tested processors (64) and the Dell 1750 series dual Pentium 4 processor nodes coupled with Myricom's Myrinet 2000/D interconnect maintains a speedup of around 100 on 128 processors and 160 on 256, the SGI Altix remains superlinear speedup, but the computational efficiency in Dell 1750 gradually reduces as more processors are added. It should be noted that speedup is a natural measurement of parallel efficiency but not the primary metric to be evaluated when ranking computers. Job throughput and machine cost are more important criteria when selecting a machine.

Chapter 13: Model Coding Description and General Information

13.1. What Users Should Know Before Using FVCOM

Many factors can affect the output of numerical circulation models, especially when they are applied to cases with complex bathymetry and realistic forcing. Ideally the model domain and forcing are tailored to the specific scientific application such that the model captures the dominant critical physical processes and the model output supports the understanding of the original scientific problem. Occasionally, there can be some (usually very subtle) mismatch between the model setup and the actual application that produces model results that may agree with some observations but actually contain significant kinematic and/or dynamical errors. As modelers, we appreciate the difficulties involved in making optimal use of the models. Thus we mention here some important guidelines which should be considered when setting up and running the model.

Users should be familiar with the inherent assumptions in FVCOM. Like most ocean models, the current version of FVCOM is formulated using the hydrostatic approximation. The model is appropriate for many coastal and regional oceanic problems, but is not applicable for the study of such processes as very small-scale convection or highly nonlinear internal gravity wave dynamics. In order to model vertical convection driven by surface cooling, FVCOM includes an adjustment option to ensure that the density profile is locally statically stable after each time step. However, the true dynamics of vertical convection are driven by non-hydrostatic processes that are not resolved in the model It should be easy to add the non-hydrostatic effect (due to the acceleration of the vertical velocity) into the horizontal pressure gradient term after the vertical velocity is determined from the incompressible continuity equation, however, this approach (so-called quasi-non-hydrostatic approximation) would result in a time lag between vertical convection and horizontal convergence/divergence and thus destroys the physics of the convection process. This is the reason why we decided not to add a quasi-non-hydrostatic version of FVCOM. An effort is being made to upgrade FVCOM to include non-hydrostatic dynamics. However, it is still in development and no source code is included in the current release version.

Users should be familiar with the important spatial scales of their application. The primary advantage of FVCOM over many coastal ocean models is that the domain is discretized using unstructured triangular meshes. To take advantage of the grid flexibility and maximize the potential of FVCOM in a specific application, it is important to estimate *a priori* the regions where increased grid density is needed. The resulting grid resolution should reflect this result. For example, if a user wants to use FVCOM to resolve the density front at the shelf break, he or she should first investigate the observational information about the cross-shelf scale(s) of this front and use this information to determine the grid density along the shelf. The cross-frontal numerical grid size should be at least 3-4 times smaller than the cross-shelf scale of the front. Users should also understand that FVCOM is a second-order approximation model. This means that the velocity in a MCE is assumed to be linearly distributed, thus setting a limit on the small-scale variation that can be resolved with a given grid resolution For example, if one wants to resolve eddies produced by small-scale topographic features (e.g., canyons) near the shelf-break, the grid in this eddy area must have the resolution to accurately capture the size and current patterns of these eddies.

Users should try to use the right approach to running the model. A good modeler always tries to find out the reason(s) why the model they are using works or fails rather than make a quick conclusion. As more people use FVCOM for different applications, a wider variety of model solutions are generated and the potential for poor model performance and/or poor model/data comparisons increases. In our experience, in many cases, the problems arise due to an inappropriate or incorrect setup of model parameters or the application of the model to a situation beyond the limitations of the model physics. Here is our suggestion: if FVCOM fails to run or produces results counter to physical intuition, please first question yourself and first examine carefully the runtime output from the code for setup errors. There is a lot of information provided in this output which is useful for finding setup errors. Second, examine if the model physics can resolve the critical physical processes inherent in your application. Third, ensure that the model temporal and spatial resolutions are sufficient for your problem application. Finding an error in numerical model experiments can require considerable skill and experience, so that we hope that users who do find errors can share their knowledge with other users.

Towards this end, we have created a users' forum on the FVCOM website to increase the communication among users.

FVCOM is in the process of continuous development. The FVCOM code is a complex suite of scientific software. Great efforts have been made to eliminate bugs from the coding. However, we cannot guarantee that it is error free. Users are strongly urged to report bugs to us through the FVCOM website. Correct findings will be credited to the discoverer. Open issues in model development and performance can be discussed within the community by initiating threads in the FVCOM bulletin board. FVCOM is intended to be a community model and thus feedback and support from the community are critical for continued model development and improvement.

13.2. The Code Structure of FVCOM

The original version of FVCOM (FVCOM 1.0) was written in Fortran 77 and has subsequently been migrated to Fortran 90 (FVCOM 2.0). The Fortran 90 version is organized in a modular fashion, which makes the coding structure clearer and provides the flexibility for users to add and remove modules according to their own research needs. In addition, all arrays are dynamically allocated at runtime so that a change in problem dimensions does not warrant a need for recompilation. Precision is selected using the *selected_real_kind* Fortran intrinsic function. The default precision is single. If a particular application proves sensitive to roundoff errors, the precision can be increased to "double"8 bytes using an option in the code makefile discussed in section 14.2b. Depending on the computer architecture and machine hardware, a change to double precision may increase the runtime by as much as 20-70%. The memory requirement will approximately double.

The present version of FVCOM includes a number of options and components as shown in Fig. 13.1 (Chen et al., 2006d). These include (1) choice of Cartesian or spherical coordinate system, (2) a mass-conservative wet/dry point treatment for the flooding/drying process simulation, (3) the General Ocean Turbulent Model (GOTM) modules (Burchard et al., 1999; Burchard, 2002) for optional vertical turbulent mixing schemes, (4) a water quality module to simulate dissolved oxygen and other environmental indicators, (5) 4-D nudging and Reduced/Ensemble Kalman Filters

(implemented in collaboration with P. Rizzoli; Zang and Rizzoli, 2003) for data assimilation, (6) a fully-nonlinear ice model (implemented by F. Dupont) for the Arctic Ocean study, (7) a 3-D sediment transport module (based on the U.S.G.S. national sediment transport model) for estuarine and near-shore applications, and (8) a flexible biological module (FBM) for food web dynamics study. FBM includes seven groups: nutrients, autotrophy, heterotrophy, detritus, dissolved organic matter, bacteria, and other. With various pre-built functions and parameters for these groups, FBM allows users to either select a pre-built biological model (such as NPZ, NPZD, etc.) or to build their own biological model using the pre-defined pool of biological variables and parameterization functions. As shown in Fig. 13.1, some components (e.g., an unstructured-grid surface wave model) still need to be added to form a complete integrated coastal ocean model system, which we hope to do in the near future.

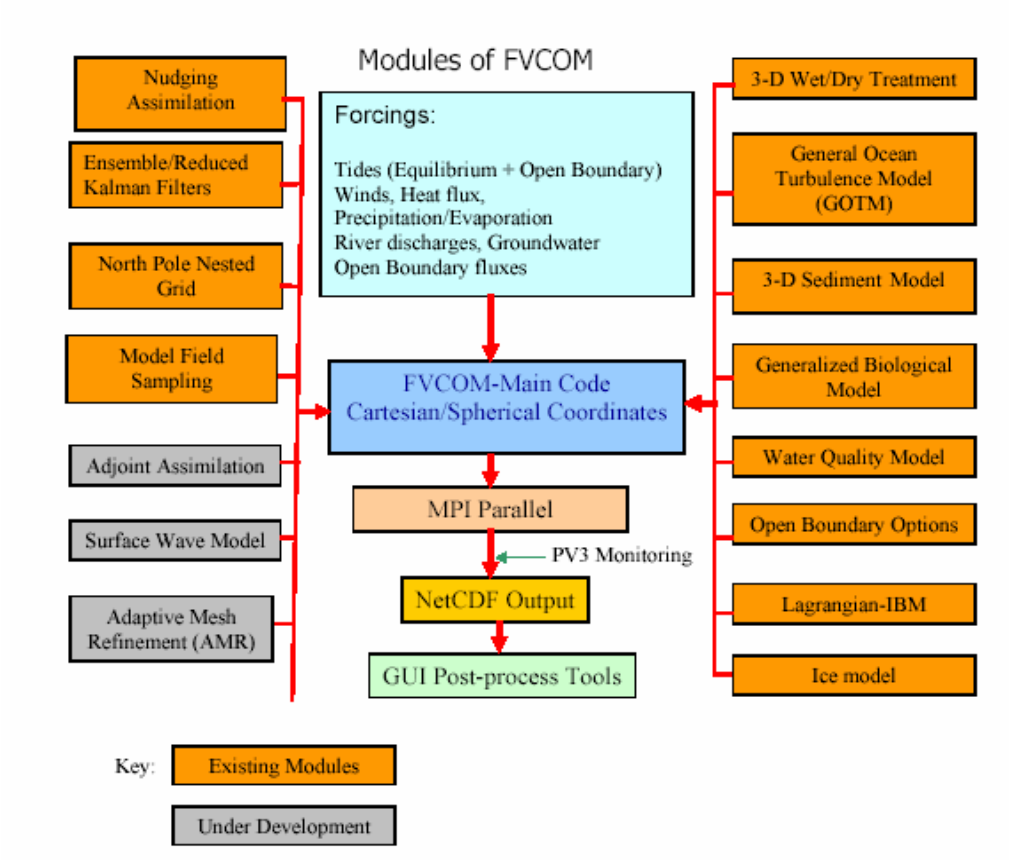


Fig. 13.1: Schematic of FVCOM module structure.

FVCOM consists of a main control program called "US_FVCOM.F" and a set of subroutines. A schematic of the flow chart of the baseline code structure is shown in Fig.

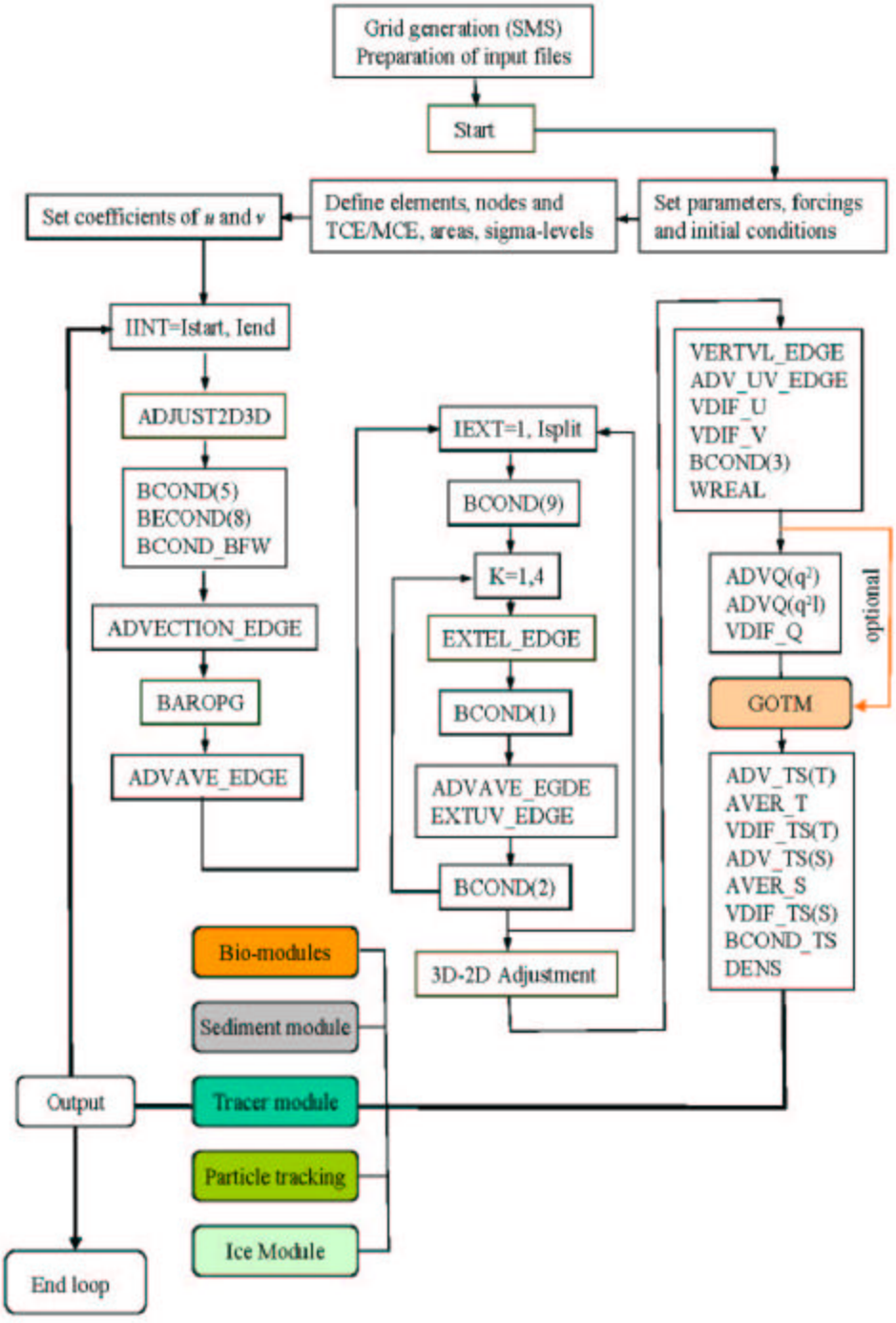


Fig.13.2: Flow chart schematic of the FVCOM.

13.2. Baseline in this case refers to the core FVCOM code which does not include data assimilation, water quality, biological, sediment, and tracer modules, MPI parallelization code structures, wet/dry point treatment processes, 3-D Lagrangian tracking, etc. The setup of these components is described in Chapter 14. Some components in Fig. 13.2 are colored, indicating that they are optional.

The FVCOM code can be run in single processor or multiple processor Linux or Unix personal computers or super computers or clusters. The type of the model run is controlled in the makefile. The SMS software is used to create the unstructured grids for FVCOM. However, users may use any two dimensional unstructured grid generation program to create unstructured grids used in FVCOM.

13.3. Criterion for Numerical Stability

The time step used in the external mode of FVCOM is bounded through the Courant-Friedrich Levy (CFL) stability criterion in the form of

$$\Delta t_E \le \frac{\Delta L}{U + \sqrt{gD}} \tag{13.1}$$

where Δt_E is the time step of the external mode, the computational length scale ΔL is the shortest edge of an individual triangular grid element, U is the magnitude of the horizontal velocity, and D is the local depth. This criterion is derived using a linearized surface gravity wave equation, in which diffusion and nonlinear advection are ignored. In most cases it produces a reasonable bound for the true nonlinear case. The time step of the internal mode is restricted by

$$\Delta t_I \le \frac{\Delta L}{C_I} \tag{13.2}$$

where C_I is the maximum phase speed of internal gravity waves. Since C_I is usually smaller than $C_E = \sqrt{gD}$, Δt_I could be much larger than Δt_E. For normal applications, we usually recommend

$$I_{split} = \frac{\Delta t_I}{\Delta t_E} \le 10. \tag{13.3}$$

A larger I_{split} could be used in realistic applications, but should be fully tested to check the numerical stability and mass conservation before it is chosen.

13.4. Subroutine and Function Descriptions

The general functionality of the major subroutines used in the FVCOM core code is listed below. Users should be aware that subroutines in modules that are currently under development may not be included in this list.

ADJUST2D3D.F	Adjust the internal mode velocity to the external mode velocity. This is done by using the defect between updated and current vertically averaged velocities.
ADJUST_TS.F	Control the mass conservation of temperature and salinity for the horizontal advection calculation at river mouths (for the case with no selection of MPDATA)
ADV_Q.F	Calculate the terms of turbulent advection, shear and buoyancy production, dissipation, and horizontal diffusion in the Mellor and Yamada level 2.5 turbulent kinetic energy (q^2) and turbulent macroscale-related (q^2l) equations.
ADV_S.F	Calculate the terms of advection and horizontal diffusion in the salinity equation.
ADV_T.F	Calculate the terms of advection and horizontal diffusion in the temperature equation.
ADV_UV_EDGE_GCN.F	Calculate the terms of advection, Coriolis, pressure gradient and horizontal diffusion in the x and y momentum equations for selection with no ghost cell boundary treatment.
ADV_UV_EDGE_GCY.F	Calculate the terms of advection, Coriolis, pressure gradient and horizontal diffusion in the x and y momentum equations for selection with ghost cell boundary treatment.
ADVAVE_EDGE_GCN.F	Calculate the fluxes of advection and diffusion in the external mode momentum equations for selection with no ghost cell boundary treatment.
ADVAVE_EDGE_GCY.F	Calculate the fluxes of advection and diffusion in the external mode momentum equations for selection with

	ghost cell boundary treatment.
ADVECTION_EDGE_GCN.F **in a specific application**	Calculate the terms of 3-D advection and horizontal diffusion that are used to compute the vertically-averaged advection and diffusion terms related to the 3-D flow field in external mode momentum equations (Gx and Gy) for selection with no ghost cell boundary treatment.
ADVECTION_EDGE_GCY.F	Calculate the terms of 3-D advection and horizontal diffusion that are used to compute the vertically-averaged advection and diffusion terms related to the 3-D flow field in external mode momentum equations (Gx and Gy) for selection with ghost cell boundary treatment.
ALLOC_VARS.F	Allocate and initialize core solver arrays.
ARCHIVE.F	Archive the model results for output.
ARCRST.F	Dump data files for restart.
ATG.F	Calculate the adiabatic temperature gradient (Deg C per Decibar).
FCT_T.F	Flux Corrected Transport Scheme used in Multidimensional Positive Definite Advection Transport Algorithm (MPDATA) for the temperature advection equation.
FCT_S.F	Flux Corrected Transport Scheme used in Multidimensional Positive Definite Advection Transport Algorithm (MPDATA) for the salinity advection equation.
BAPOPG.F	Calculate the horizontal baroclinic pressure gradient in sigma coordinate.
BCMAP.F	Map open boundary nodes to local processors.
BCOND_GCN.F	Set boundary conditions: tidal forcing at open boundaries, no-flux of external and internal mode currents on the solid wall; surface forcing for external and internal modes. GCN represents the selection with no ghost cell treatment.

BCOND_GCY.F	Set boundary conditions: tidal forcing at open boundaries, no-flux of external and internal mode currents on the solid wall; surface forcing for external and internal modes. GCY represents the selection with ghost cell treatment.
BCOND_BFW.F	Set bottom boundary condition for groundwater input.
BCOND_TS.F	Set up the open boundary conditions for temperature and salinity.
BCS_FORCE.F	Set up the bottom boundary condition for ground water (freshwater), tidal forcing (amplitudes and phases) at open boundary, freshwater river discharges, and meteorological forcing.
BRACKET.F	Determine data interval in for current time for time dependent boundary forcing.
BROUGH.F	Calculate the bottom drag coefficient.
CELL_AREA.F	Calculate the area of individual triangles as well as areas for node- and element-based flux control volumes.
CLOSEFILE.F	Close the output files.
CONV_OVER.F	Adjust for static instability in the water column.
DATA_RUN.F	Input parameters that control the model run.
DENS2.F	Calculate the in-situ density (sigma-t) based on salinity and temperature.
DENS3.F	Calculate the in-situ density based on a polynomial expression (Jackett and McDougall, 1995) in which the pressure is taken into account.
DENS.F	Calculate the potential density based on potential temperature and salinity. The pressure effects are incorporated to allow the model to run for the freshwater case with water temperature < 4 ^{o}C.
DEPTH_CHECK.F	Check the water depth to be sure it is greater than minimum depth in the case with no wet/dry point treatment. When the water depth is less than the

	minimum depth, halt the program with a warning message on the screen.
DEPTH_GRAD.F	Calculate the spatial bathymetric gradients.
DOMDEC.F	Partition the domain into elements using the METIS graph partitioning libraries.
EXTEL_EDGE.F	Calculate the advective fluxes of the vertically integrated continuity equation.
EXTELPF_EDGE.F	Calculate the subtidal surface elevation at time step n+1.
EXTUV_EDGE.F	Accumulate the fluxes of the external modes.
GENMAP.F	Define local-to-global node and element numbering maps. Set up arrays to control interprocessor data exchanges.
GETDIM.F	Determine number of elements and nodes in calculation using grid.
GHOSTUV.F	Calculate ghost cell velocity for external mode and internal mode.
HOT_START_DATA.F	Read in restart data file and restart.
INITIAL_QQL.F	Set up the initial values of turbulent kinetic energy and turbulent macroscale.
INITIAL_TS.F	Set up the initial fields of water temperature, salinity and density. Calculate the initial fields of averaged water temperature, salinity and density at the reference levels.
INITIAL_UVEL.F	Read the initial fields of the surface elevation and velocities.
IOFILES.F	Open input files for model parameters used to drive the model and output files for model results to be written.
MOD_ASSIM.F	Data assimilation modules.
1) SET_ASSIM_PARAM;	Read the parameters controlling assimilation.

2) SET_CUR_ASSIM_DATA	Set up assimilation data for current observations.
3) CURRENT_NUDGING	Merge the model-predicted velocity to observational currents at measurement sites.
4) SET_SST_ASSIM_DATA	Set up the SST observational data for assimilation.
5) SST_NUDGING	Merge the model-predicted surface water temperature to the observed SST.
6) SST_INT	Interpolate hourly averaged SST data to observational points.
7) SET_TS_ASSIM_DATA	Set up the assimilation data for temperature/salinity observations
8) T_NUDGING	Nudge model-predicted temperatures to observational values at measurement sites.
9) S_NUDGING	Nudge model-predicted salinity to observational value at measurement sites.
10) HOT_START_SST	Read the simulated model results for the data assimilation.
11) ARC_SST	Dump the data files for assimilation restart.
MOD_ATMOTIDE.F	Atmospheric tide module.
1) ALLOCATE_ATMO	Allocate arrays for calculating atmospheric tidal potential gradient.
2) ELEVATION_ATMO	Calculate the atmosphere tide-induced sea surface gradient.
MOD_BALANCE_2D.F	Output terms in 2-D momentum equations for the momentum balance analysis.

MOD_CLOCK.F	Timing module:
1) START_CLOCK	Initialize timer => TINIT
2) GETTIME	Return a time string days: hours, minutes, seconds from seconds.
3) REPORT_TIME	Report calculation speed and time to complete.
MOD_DYE.F	The dye tracking module.
1) ALLOC_VARS_DYE	Allocate arrays of the dye concentration variable.
2) SET_DYE_PARA	Specify the source area and parameters used in the dye release.
3) ADV_DYE	Calculate the advection and horizontal diffusion terns in the dye concentration equation.
4) VDIF_DYE	Calculate the vertical diffusion term in the dye concentration equation.
5) INITIAL_DYE	Initialize the dye concentration.
6) BCOND_DYE	Specify the boundary conditions (river flux) of the dye concentration.
MOD_EQUITIDE.F	Equilibrium tide module
1) ALLOCATE_EQUI	Allocate arrays for calculating equilibrium tidal potential gradient.
2) ELEVATION_EQUI	Determine the equilibrium tidal elevation to add into the surface elevation gradient terms.
MOD_GOTM.F	Module to link GOTM libraries
1) INIT_GOTM	Initialize the GOTM using GOTM libraries.
2) ADVANCE_GOTM	Advance the GOTM using GOTM libraries.
MOD_INP.F	Text input module
1) GET_VAL	Decompose input line into variable name and variable value(s).

2) SCAN_FILE	Scan an input file for a variable.
MOD_LAG.F	Lagrange particle tracking module
1) SET_LAG	Read in particle locations and initialize arrays.
2) LAG_UPDATE	Update particle location and write out data.
3) TRAJECT	Use 4-stage ERK scheme to integrate particle path.
4) INTERP_V	Linearly interpolate velocity field at particle position.
5) INTERP_ELH	Interpolate free surface elevation and bathymetry at particle position.
6) FHE_ROBUST	Find element containing particle using robust method (search from nearest to progressively further elements).
7) FHE_QUICK	Search for particle in neighboring elements of last known position.
8) ISINTRIANGLE	Returns TRUE if point (xp,yp,zp) is located in triangle defined by three vertices VX(3),VY(3)
MOD_MAIN.F	Contains primary code data.
MOD_MEANFLOW	Mean flow at open boundary module
1) READ_MEANFLOW	Read in time series of the mean flow flux (m^3/s) at open boundaries.
2) SET_BNDRY_MEANFLOW	Set metrics to add the mean flow boundary conditions.
3) BCOND_MEANFLOW	Interpolate the time series of the mean flow open boundary flux (m^3/s) at open boundaries.
MOD_NCDAVE.F	Save time averaged fields and dump to NetCDF file.
MOD_NCDIO.F	Set up NetCDF format output.
1) SET_IO_PARAM	Read in parameters controlling NetCDF output.
2) OUT_NETCDF	Define NetCDF output variables and send

	to.PUTVAR.for output.
3) PUTVAR	Write variables to NetCDF output. Collect to global before writing if running in multiprocessor mode.
MOD_NORTHPOLE.F	North pole treatment module
1) FIND_NORTHPOLE	Locate the node number of north pole and nodes and elements connected to the north pole.
2) FIND_CELLSIDE	Locate the edge number of triangles and tracer control volumes connected to the north pole for the calculation of the elevation.
3) ADVAVE_EDGE_XY	Same as ADVAVE_EDGE_GCN.F, but it is used only for the north pole treatment in the polar stereographic projection coordinates.
4) ADVECTION_EDGE_XY	Same as ADVECTION_EDGE_GCN.F, but it is used only for the north pole treatment in the polar stereographic projection coordinates.
5) ADV_UV_EDGE_XY	Same as ADV_UV_EDGE_GCN.F, but it is used only for the north pole treatment in the polar stereographic projection coordinates
6) EXTEL_EDGE_XY	Same as EXTEL_EDGE.F, but it is used only for the north pole treatment in the polar stereographic projection coordinates
7) EXTUV_EDGE_XY	Same as EXTUV_EDGE.F, but it is used only for the north pole treatment in the polar stereographic projection coordinates.
8) VERTVL_EDGE_XY	Same as VERTVL_EDGE.F, but it is used only for the north pole treatment in the polar stereographic projection coordinates
9) ADV_S_XY	Same as ADV_S.F, but it is used only for the north pole treatment in the polar stereographic projection coordinates
10) ADV_T_XY	Same as ADV_T.F, but it is used only for the north pole treatment in the polar stereographic projection coordinates.

11) BAROPG_XY	Same as BAROPG, but it is used only for the north pole treatment in the polar stereographic projection coordinates.
12) SHAPE_COEF_XY	Same as SHAPE_COEF_GCN, but it is used only for the north pole treatment in the polar stereographic projection coordinates.
13) ADV_Q_XY	Same as ADV_Q, but it is used only for the north pole treatment in the polar stereographic projection coordinates.
MOD_OBSC.F	Radiation open boundary module
1) ALLOC_OBC_DATA	Allocate and initialize arrays for the radiation open boundary variables.
2) ASSIGN_ELM1_TO_ELM2	Assign the elevations at time step N-1 to N-2 and N to N-1.
3) BCOND_ASL	Type 1-ASL radiation boundary condition for the surface elevation. The sea level is specified at the open boundary (like tidal forcing).
4) BCOND_ASL_CLP	Type-2 Clamped (ASL_CLP) condition for the surface elevation.
5) BCOND_GWI	Type-3 GWI radiation condition for the surface elevation.
6) BCOND_BKI	Type-4 BKI radiation condition for the surface elevation.
7) BCOND_ORE	Type-5: Explicit Orlanski Radiation (ORE) radiation condition for the surface elevation.
8) SEPARATE_OBC	Accumulate separately the number of nodes for 11 types of open boundaries.
9) SETUP_OBC	Locate the adjacent nodes and elements connected to the open boundary cells.
10) FLUX_OBN	Calculate the flux across the open boundary line for the section of the nonlinear treatment.
11) TSOBC_TYPE	Read in the type of the open boundary condition for

	temperature and salinity perturbations (relative to the vertical mean).
12) BCOND_T_PERTURBATION	Calculate the temperature perturbation at the OB.
13) BCOND_S_PERTURBATION	Calculate the salinity perturbation at the OB.
MOD_OBCS2.F	HCQ (Huang, Chen, and Qi) open boundary module
1) ALLOC_OBC2_DATA	Allocate the array of variables used for the open boundary treatment.
2) FIND_OBSIDE	Find the indexes of nodes and cells used for HCQ OB module.
3) BCOND_TIDE_2D	Interpolate the time series of the tidal elevation in the OBS regime to the model time step in the 2-D external mode.
4) BCOND_TIDE_3D	Interpolate the time series of the tidal elevation in the OBS regime to the model time step in the 3-D internal mode.
5) BCOND_NG_2D	Non-gradient radiation open boundary used to determine the volume flux on the open boundary line for the 2-D external mode.
6) BCOND_NG_3D	Non-gradient open radiation boundary condition used to determine the volume flux on the open boundary line for the 3-D internal mode.
7) BCOND_BKI_2D	Type-3 GWI radiation boundary condition used to determine the volume flux on the open boundary line for the 2-D external mode.
8) BCOND_BKI_3D	Type-3 GWI radiation boundary condition used to determine the volume flux on the open boundary line for the 3-D external mode.
MOD_OBCS3.F	HCQ open boundary flux module
1) ALLOC_OBC3_DATA	Allocate an array of variables that are used in the flux calculation on the open boundary line.

2) ZERO_OBC3	Initialize the open boundary array by setting to zero.
3) SETUP_OBC3	Locate the boundary line of the OB TCE cell for the calculation of the volume flux.
4) FLUX_OBN2D	Calculate the volume flux on the boundary line of the OB TCE cell for the 2-D external mode.
5) FLUX_OBC2D	Calculate the volume flux on the boundary line of the OB MCE cell for the 2-D external mode.
6) FLUX_OBC3D	Calculate the volume flux on the boundary line of the OB TCE and MCE cells for the 3-D internal mode.
MOD_PAR.F:	Parallelization module
1) INIT_MPI_ENV	Initialize MPI environment.
2) EXCHANGE	Pass element/node data between processors.
3) NODE_MATCH	Ensure consistency of values for nodes on interprocessor boundary.
4) GATHER	Gather array/vector A into global array/vector (Float).
5) IGATHER	Gather array/vector A into global array/vector (Integer).
6) SORT	Sort elements in N and return order in M.
7) GETLOC	Determine local identity of global element/node I using map.
8) PREAD	Read in real global array and decompose to local sub domains.
9) PWRITE	Gather local array to global and write to file.
MOD_PREC.F	Definite floating point precision using selected real kind intrinsic function.
MOD_PROBE.F	Set up time series printing.
1) SET_PROBES	Read in user defined probe setup files.
2) OPEN_PROBES	Write header files for probe output.

3) DUMP_PROBE_DATA	Write time series data to output files.
4) PROBE_STORE	Load flow data in temporary arrays for time series output.
MOD_SPHERICAL.F	Spherical coordinate module
1) ARC	Calculate the arc length for two given points on the spherical plane.
2) AREA	Calculate the area of a triangle on a spherical plane
3) ARCC	Calculate the mid point between two points on the spherical plane
4) ARCX	Calculate the arc length in the longitude direction between two given points on the spherical plane.
5) ALLOC_SPHERE_VARS	Allocate and initialize arrays used for spherical coordinate.
MOD_TSOBC.F	Contains the open boundary conditions of temperature and salinity time series nudging.
MOD_TYPES.F	Contains user-defined types.
MOD_UTILS.F	Utility Module
1)FOPEN	File open utility.
2) CRAY_SYSTEM	System call specific to CRAY environment.
MOD_WD.F	The wet/dry point treatment modules
1) SET_WD_PARAM	Read in parameters controlling the wet/dry treatment.
2) SET_WD_DATA	Initialize arrays used for wet/dry treatment.
3) ALLOC_WD_DATA	Allocate and initialize wet/dry treatment arrays.
4) WET_JUDGE	Determine dry or wet nodes and elements.
5) WD_UPDATE	Shift wet/dry variables to new time levels.
6) WD_DUMP	Dump wet/dry flag data for restart.

7) WD_READ	Read in wet/dry flag data for restart.
OUT_AVGE.F	Write flow field averages to assess if model has achieved quasi-periodic behavior.
OUT_BINARY.F	Write binary output data.
OUT_SMS_ONE.F	Write output files used by SMS post-processing software.
PDOMDEC.F	Set up local physical domain (connectivity/mesh/Coriolis) from global.
PHY_BAROPG.F	Calculate the baroclinic pressure gradient in the standard z-levels. The water column is divided into 600 standard levels and pressure gradient is calculated at each level. The resulting pressure gradients are converted back to sigma-levels through vertical interpolation approach. (The test case shows that the small errors near the bottom on steep bottom topography due to this calculation might grow through nonlinear interaction. Recommend to use only for comparison between the z and sigma coordinate systems, but not for realistic applications.)
REPORT.F	Report flow field statistics for runtime monitoring of solution.
SECTINF.F	Specify the output files used for the "GRI" graphics configured by the Marine Ecosystem Dynamics Laboratory at SMAST/UMASSD.
SET_BNDRY.F	Set up metrics for the boundary conditions.
SET_SIGMA.F	Set up the sigma levels in the vertical.
SHAPE_COEF_GCN.F	Set coefficients of linear regression equations for selection with no ghost cell boundary treatment.
SHAPE_COEF_GCY.F	Set coefficients of linear regression equations for selection with ghost cell boundary treatment.
SHUTDOWN_CHECK.F	Check depth array for NAN. Shutdown if found.

SINTER.F	An array interpolation and extrapolation subroutine
1) SINTER	Interpolate/extrapolate vertical hydrographic data to grid
2) SINTER_P.F	Interpolate/extrapolate pressure data for z-coordinate pressure gradient reconstruction.
STARTUP.F	Startup module
1) STARTUP	Set and begin a restart run from a specified time.
2) EXCHANGE_ALL	Exchange all variables across processor boundaries.
SVAN.F	Calculate the density anomaly.
THETA.F	Calculate the local potential temperature.
TGE.F	Define the non-overlapped, unstructured triangular meshes used for flux computations. The mesh could be created using the commercial software called "sms8.0" or other mesh generation programs. The mesh file generated by sms8.0 can be used directly with this subroutines, while the mesh file generated using other programs must be converted to meet the required format used here.
US_FVCOM.F	The main program of FVCOM
UTILITIES.F	Utility Subroutines
1) PSTOP	Stops the Fortran program (stops all processors if parallel).
2) N2E3D	Calculate variables at centroids of elements by using variables at surrounding nodes for the 3-D internal mode.
3) N2E2D	Calculate the variables at centroids of elements by using the variables at surrounding nodes for the 2-D internal mode.
VDIF_Q.F	Calculate implicitly the vertical diffusion terms of turbulent kinetic energy and macroscale equations.
VDIF_TS.F	Calculate implicitly the vertical diffusion terms of

	water temperature and salinity equations.
VDIF_UV.F	Calculate the vertical diffusion terms of the u and v momentum equations.
VERTVL_EDGE.F	Calculate the sigma coordinate vertical velocity for the 3-D mode.
WATER_DEPTH.F	Read the static water depths from nodes and use them to calculate the depth at the center of individual triangle.
WREAL.F	Calculate the Cartesian vertical velocity.

The Water Quality Module (WQM)

MOD_WQM.F	Define parameters and arrays for the water quality module: 1) Dissolved oxygen (DO) 2) Carbonaceous biochemical oxygen demand (CBOD) 3) Phytoplankton (PHYT) 4) Ammonia nitrogen (NH4) 5) Nitrate and Nitrite Nitrogen (NO3+NO2) 6) Organic Nitrogen (NH4) 7) Orthophosphorus or Inorganic phosphorus (OPO4) 8) Organic phosphorus (OP)
1) GET_WQMPAR	Read in the WQM control parameters.
2) WQMPARA	Read in the WQM biological and chemical parameters from WQM input files.
3) ALLOC_WQM_VARS	Allocate the array of the water quality state variables.
4) INITIAL_WQM	Initialize water quality state variables.
5) BCS_FORCE_WQM	Read in the WQM boundary forcing (for example, river discharge, etc).
6) ADV_WQM	Calculate the advection terms of eight component water quality equations.
7) BCOND_WQM	Specify the boundary conditions of WQM variables.

8) VDIF_WQM	Calculate the vertical diffusion term of the water quality equations.
9) EXCHANGE_WQM	Exchange water quality variables among processors.
10) WQMCONST	Calculate the coefficients used in the WQM.
11) KAHYDRA	Set empirical relation for oxygen aeration (by flow)
12) KAWIND	Set empirical relation for oxygen aeration (by wind)

The Nutrient-Phytoplankton-Zooplankton (NPZ) Model Module

ADBIO.F	Compute the advection terms of NPZ variable: 1) Nutrients 2) Phytoplankton 3) Zooplankton
PROBIOL.F	Compute the vertical diffusion terms of NPZ equations and produce the new values of NPZ variable for the next time step.
BCONBIOL.F	Specify the open boundary conditions of NPZ variables.
BIOPARAM.F	Read in biological parameters that control the NPZ food web.
BIOLINIT.F	Set up the initial values of NPZ variables.

The Nutrient-Phytoplankton-Zooplankton-Detritus-Bacteria Model (NPZDB) Module

ADBIO_GL.F	Compute the advection terms of NPZDB variables: 1) Phosphate 2) Silica 3) Small phytoplankton 4) Large phytoplankton 5) Small zooplankton 6) Large zooplankton 7) Bacteria 8) Detrital silica 9) Detrital phosphate
PROBIOL_GL.F	Compute the vertical diffusion terms of NPZDB variable and produce the new values of these variables for the next time step.

BCONBIOL_GL.F	Specify the open boundary conditions of NPZDB variables.
BIOPARAM_GL.F	Read in biological parameters controlling NPZDB model run.
BIOLINIT_GL.F	Specify the initial values of NPZDB variables.

The Nutrient-Phytoplankton-Zooplankton-Detritus (NPZD) Model Module

ADBIO_GB.F Compute the advection terms of NPZD variables:
1) Nitrate
2) Silica
3) Small phytoplankton (flagellate)
4) Large phytoplankton (diatom)
5) Small zooplankton (ciliate)
6) Large zooplankton (copepod)
7) Detrital nitrogen
8) Detrital silicate
9) Ammonia

PROBIOL_GB.F Compute vertical diffusion terms implicitly to produce the new values of NPZD variables for next time step.

BCONBIOL_GB.F Specify the open boundary conditions of NPZD variables.

BIOPARAM_GB.F Read in biological parameters.

BIOLINIT_GB.F Set up the initial values of NPZD variables.

The Sediment Module

Setup

To utilize the model, users must first modify the subroutine which provides the initial sediment fields, then compile FVCOM with inclusion of the sediment model, setup the sediment parameter input file, and optionally setup the open boundary nudging and river forcing files. These steps are outlined in detail in the following subsections.

Initial Conditions

The user must provide initial fields of sediment concentration, bed layer properties and bed layer sediment fractions. Note that if a case is "hot started", these fields are read from the restart file. The initial fields are set in the subroutine *init_sed.F* found in the *FVCOM_source* directory of your distribution. The user definition section is clearly defined within this file by comment fields. The first variables to set are the bed properties, including the age, thickness and porosity of the bed. A bed porosity of 0 is solid sediment while a bed porosity of .6 has a sediment volume fraction of .4.

bed(i,k,iaged)	Age of bed layer in seconds at node i, bed layer k
bed(i,k,ithck)	Thickness of bed layer in meters at node i, bed layer k
bed(i,k,iporo)	Porosity of bed layer at node i, bed layer k
sed(ised)%(i,k)	Fraction of bed sediment in layer k and node i composed of sediment ised
sed(ised)%(i,k)	Concentration of sediment ised in sigma layer k and node i

Compilation

To compile FVCOM with the sediment model, uncomment the SEDIMENT flag in the makefile, clean the object files ('make clean'), and recompile ('make').

Runtime Parameter File

The runtime parameter file is *casename*_sediment.inp and should be placed in the input directory INPDIR with the other model input files. Parameters and definitions are described in the table below.

Parameter	Description	Type
SED_START	Iteration number at which sediment model is initiated (allows model to reach quasi-periodic equilibrium). A value of zero will start the sediment dynamics simultaneously with the physical forcing.	integer
SED_NUDGE	Indicates that sediment concentration data at open boundary nodes will be supplied (see next subsection for details).	logical
SED_ALPHA	Nudging Relaxation Factor Range: [0-1]. Use 0 for no nudging, 1 for strong nudging.	real
SED_RAMP	Number of iterations over which to ramp sediment nudging, use 0 for no ramping.	real
N_REPORT	Iteration interval for reporting sediment statistics to screen output. Use a value of 0 to deactivate reporting.	integer

BEDLOAD	Activate bedload model.	logical
SUSLOAD	Activate suspended load model.	logical
NBED	Number of bed layers (> 0).	integer
INF_BED	True if infinite sediment supply is desired (valid fo NBED = 1).	logical
MPM_CS	Critical Shields parameter for Meyer-Peter Muller bedload formulation (suggested: 0.047).	real
MPM_GM	MPM formulation exponent (suggested: 1.5).	real
MPM_K	MPM constant (suggested: 8.0).	real
NSED	Number of sediment types.	integer

For each sediment type, NSED, sediment properties must be defined in the following order:

SED_NAME	Sediment descriptive name (sand, mud, etc).	string
SED_TYPE	Sediment type (cohesive, non-cohesive).	string
SED_SD50	Sediment mean diameter (mm).	real
SED_SRHO	Sediment dry density (kg/m^3).	float
SED_WSET	Settling velocity (mm/s).	float
SED_ERAT	Erosion Rate (kg/m^2-s).	float
SED_TAUE	Critical erosive stress (N/m^2).	float
SED_TAUD	Critical depositional stress (N/m^2).	float

Open Boundary Nudging

Values for each sediment type are prescribed at open boundary nodes and used to force the open boundary towards a known solution. Note that this nudging is optional and is activated by the SED_NUDGE flag described in the previous section. If this flag is set to true (T), the sediment open boundary data file must be provided.

Filename: casename_sed_nudge.dat

The format of the file is as follows (ascii).

Line1: N_OBC_NODES

Line2: 1 OBC_NODE_1 SED1_@OBC1 SED2_@OBC1 ….

Line3: 2 OBC_NODE_2 SED1 @OBC1 SED2_@OBC2 …

Line4: ….

The first line contains the number of open boundary nodes. This must be consistent with the number of nodes in the model. The data starts on the second line. The first integer is a counter. The second integer is the open boundary node number. The remaining data on each line are the sediment concentration (kg/m^3) at that node for each sediment type.

Point Source (River) Forcing

A time dependent point source loading of sediment for each sediment type can be supplied at the river nodes. It is activated by the SED_PTSOURCE option in the sediment parameter deck described above. If this flag is set to true (T), the point source forcing data file must be included.

Filename: casename_sed_ptsource.dat

The format of this file (ascii) is described below and is similar to the format of the river data files.

Line1: ntime, nriv

Line2: time1

Line3: sed1_@time1, sed2_@time1 …

Line4: time2

Line5: sed1_@time2, sed2_@time2 …

Line6: …

The first line contains the number of data entries and rivers. The number of rivers must be consistent with the model. After the header info comes the data. The time (in hours) is followed by the sediment concentration (kg/m^3) data for each sediment type. If the concentration is constant, supply only two data points coinciding with the time in hours for the beginning and end of the model run. Linear interpolation is used to calculate data within the observation intervals.

Chapter 14: Model Installation, Compilation, and Execution

Acquisition, installation, compilation, and execution of the model are described in this chapter. Note that most of these steps are machine dependent and some knowledge of the given computer system will be necessary. Much work has been done to make the model portable. However, FVCOM is a rapidly evolving research tool, not a release of commercial software, and thus installation and compilation of the model may not be straightforward. Section 14.1 will address downloading and unpacking the model. The model directory tree will be shown. Section 14.2 will discuss model compilation. Section 11.3 will present the normal procedure of model execution.

Windows Users: It is recommended that you download and install the latest version of *cygwin (http://www.cygwin.com)*. *Cygwin* is a bash shell/unix emulation program and contains many of the tools such as tar, gzip/gunzip, and cpp which will be useful for installation and compilation of FVCOM. Also, *cygwin* gives Windows a decent interface with which to setup and run the model. Note that it requires basic familiarity with unix commands.

14.1. Obtaining FVCOM

FVCOM can be obtained from the FVCOM community website through the code repository portal. Note that this section of the website is password protected. Usernames and passwords will be provided after the user agreement form has been signed. Once in the repository, read the package definitions and choose the appropriate version for your research needs. Download the gzipped tar file directly to your computer. This tar file will contain the source code, a makefile, and the test problems described in Chapter 15. Place the model in a suitable location and unpack with:

gunzip –c FVCOMxxx | tar xvf -

This will produce the top level directory FVCOMxxx. The directory structure is shown in Fig. 14.1.

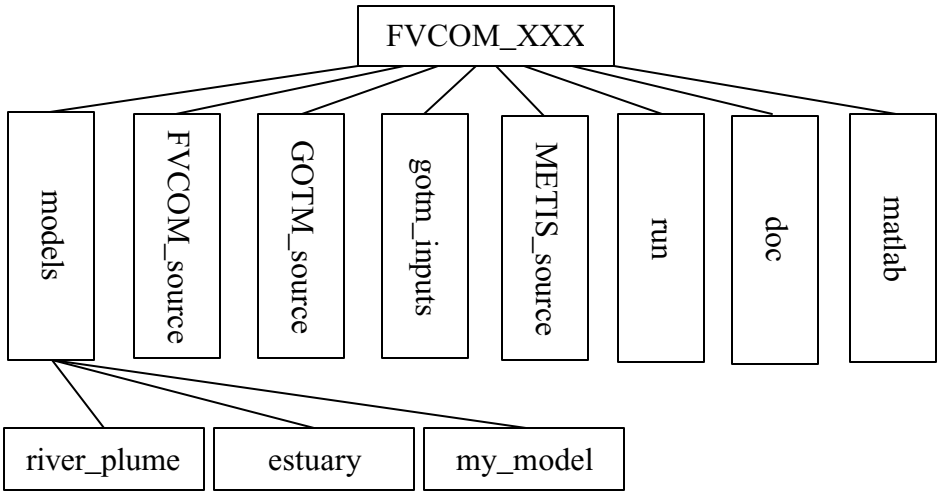


Figure 14.1: FVCOM directory structure

14.2a. Compiling METIS Libraries

(*For users intending to use the parallel code*) The METIS graph partitioning libraries are used to perform the domain decomposition portion of the FVCOM parallelization. These library routines are coded in the c language and must be compiled separately from the main coding. To compile the libraries, edit the makefile in the METIS source directory to point to your c compiler. It is advisable to use a c compiler from the same vendor as your intended Fortran compiler. This will help avoid compatibility problems when linking the libraries and the FVCOM code. To compile the libraries, use "make". A successful build will produce the file libmetis.a which will be linked during the FVCOM build. Note that if you do not intend to use FVCOM in a multiprocessor environment, it is not necessary to build these libraries.

14.2b. Compiling FVCOM

Compilation of FVCOM is done using the *makefile* included with the FVCOM source code in the FVCOM_source directory. Users must edit the makefile to select desired code modules (parallel, water quality, etc) as well as certain code options (double/single precision). Then, makefile variables must be set to point to the Fortran compiler, the

location of the c preprocessor (cpp) as well as locations for the MPI (Message Passing Interface) libraries (if intending to run FVCOM in parallel) and the NetCDF libraries (if intending to use NetCDF output). In most cases, this can be completed through the following steps and remember, the makefile is your friend.

STEP 1 Edit the *makefile* and choose from the following options by uncommenting (removal of #) the desired value for each option. (Value meanings are also outlined in the *makefile.*)

Table 14.1: Code Module Options to Select before Compilation

Option	Values	Description
PRECISION	activated	Chooses double precision for all floating point variables.
	deactivated	Chooses single precision for all floating point variables.
SPHERICAL	activated	Equations will be solved in spherical coordinates. Note that grid file must be in latitude/longitude format (see grid file description in Chapter 12).
	deactivated	Equations are solved in Cartesian coordinates.
FLOODYING/DRYING	activated	Include wet/dry treatment of domain. (It also can be activated/deactivated at run time in the input file control, see file ***_run.dat).
	deactivated	No wet/dry treatment included.
MULTI_PROCESSOR	activated	Code will include MPI based parallelization. Note that MPI libraries must be linked at compile time.
	deactivated	All parallelization will be removed from the code. No MPI libraries are necessary for compilation.
WATER_QUALITY	activated	Water Quality Model (WQM) will be included in the compilation. If WQM is turned on the runtime control file, necessary WQM setup files must be present. See Chapter 12.
	deactivated	No WQM included

NETCDF	activated	NetCDF output is included in model. Note that LIB_NETCDF and INC_NETCDF makefile variables must be set to point respectively to NetCDF libraries.he variables are located directly below the NETCDF option variable.
	deactived	Code will produce normal binary output. No NetCDF libraries necessary for compilation.
DATA_ASSIMILATION	activated	Code will include nudging-based data assimilation methods for SST, and current, salinity, and temperature data from mooring. If selected in the runtime control file, several setup files for the data assimilation must be present. See Chapter 12 for details.
	deactivated	No data assimilation is included in the model.
SOLID BOUNDARY (One of flags must be chosen, and also only one flag can be chosen).	GCN GCY1 GCY2	No ghost cells included. Ghost cells included: ghost cell is located symmetrically relative to the boundary cell edge. Ghost cells included: ghost cell is located symmetrically relative to the middle point of the boundary cell edge.
TURBULENCE MODEL	activated	Use GOTM instead of the original FVCOM Mellor-Yamada 2.5 implementation for parameterization of vertical eddy viscosity.
	deactivated	Use the original FVCOM Mellor-Yamada 2.5 implementation in FVCOM for parameterization of vertical eddy viscosity.
EQUILIBRIUM TIDE	activated	Equilibrium tides included.
	deactivated	No equilibrium tides included.
ATMOSPHERIC TIDE	activated	Atmospheric tides included.

	deactivated	No atmospheric tides included.
ARCTIC OCEAN	activated	The North Pole included. (If this flag activated, FLAG_2 should be selected first.)
	deactivated	No North Pole included.
MPDATA	activated	FCT-multidimensional positive definite advection transport algorithm included for the vertical advection term.
	deactivated	Using original algorithm in FVCOM
TWO_D_MODEL	activated	Set the model only for 2-D barotropic case.
	deactivated	Set the model for both 2-D and 3-D modes (it is default setup).
BALANCE_2D	activated	Output the terms in the 2-D momentum equations for the momentum balance analysis.
	deactivated	No selection to output the terms in the 2-D momentum equations.
TS_OBC	activated	Turn on the temperature and salinity nudging at open boundaries to force the model with observational hydrographic data.
	deactivated	No temperature and salinity nudging is specified at open boundaries.
MEAN_FLOW	activated	Mean flow transports are specified at open boundary as in or out fluxes.
	deactivated	No mean flow transports are specified at open boundaries.

TIDE_OUTPUT	activated	Turn on the output of the tidal elevation and currents at open boundary cells and cells connected to the open boundary cells. This output files include the time series of the surface elevation and currents that will be used for the case with tidal and mean flow forcing. This choice is only for the model run with only tidal forcing.
	deactivated	Turn off the output of the tidal elevation and currents at open boundary cells and cells connected to the open boundary cells. When the model is run for default setup with no HCQ open boundary treatment selected, this term should be always set to "deactivated".
DYE_RELEASE	activated	Turn on the dye release module.
	deactivated	Turn off the dye release module.

NOTE!: If you have compiled the model with a given set of options and you wish to change options you must recompile the entire code using "make clean" to remove all object files and executables and "make" to remake.

STEP 2 Set *makefile* variables for the location of your compiler and c preprocessor (cpp) as well as your compiler optimization/debugging flags. Variable description and typical values are shown in Table 14.2. Note that you must use a Fortran95 capable compiler. A listing of possible compilers and related information is provided in table 11.3. The c preprocessor (cpp) is included in almost all linux/unix distributions as part of gcc. For Windows users, if cpp is not installed on the system, it can be made available through cygwin (see note to windows users at beginning of chapter). Use "which cpp" to determine the location of cpp and supply the full path to the makefile variable CPP. The makefile currently contains several example sections for a variety of operating systems/compiler combinations. Uncomment a section if it is similar to your environment or provide values in the last section labeled "OTHER".

Table 14.2 Makefile Variables

Variable	Description	Note	Typical Value
CPP	C Preprocessor	Required	/usr/bin/cpp
CPPFLAGS	CPP Flags	Required	$(MOD_FLAGS)
FC	Fortran Compiler	Required	pgf90/xlf90/ifort/f90
OPT	Compiler Flags		Compiler + Machine Dependent
CLIB	Misc. Libraries		

STEP 3 Remove all object files and executables with "make clean". Compile code with "make". Watch screen output during compilation to ensure that the desired preprocessing flags are being utilized. Successful compilation will produce the executable *fvcom*. Copy or link the executable to the run directory. We recommend linking because code change and subsequent recompilation will automatically be reflected in the executable in the run directory.

Table 14.3 Fortran 90 Compilers for Clusters, Workstations, and Laptops

Compiler	Platforms	Type	FVCOM Tested	Notes
Intel v8	Windows/Linux	Commercial + Free Unsupported	Yes (Linux)	Outperforms Portland Group on Intel Chips
Portland Group	Windows/Linux	Commercial	Yes (Linux/Windows)	
Absoft	Windows/Linux	Commercial	No	
NAG	Windows/Linux	Commercial	No	
Lahey/Fujitsu	Windows/Linux	Commercial	No	

14.3a Running FVCOM (Serial)

The FVCOM executable "fvcom" takes only one argument, the "casename" or string used to identify a particular model (see Chapter 15: Model Setup). For example, in our Gulf of Maine model, we have chosen "gom" as our casename. To execute the model type `fvcom gom` on the command line. To run the idealized estuary case described in Chapter 16, use `fvcom tst`.

14.3b Running FVCOM (Parallel)

Parallel execution of FVCOM is machine dependent. If you are using a computer with a queuing system such as PBS or LoadLeveler, seek details from the system manager. For desktop SMP's and small laboratory clusters with commercial or open source implementations of the MPI libraries, parallel execution will most likely be performed using the mpirun command script which comes with these distributions. A typical mpirun startup of FVCOM to run the *tst* case might look like:

```
mpirun –machinefile /etc/mymachinefile –np 2 mpfvcom tst
```

Here the list of machines to be used (in order of processor allocation) is placed in the file *mymachinefile*. The flag *np* takes as an argument the number of processors to run. Finally the executable *fvcom* and argument *tst* are supplied. Generally the machinefile list is simply a list of hostnames. For dual processor machines, the hostnames can be doubled up and in some cases the hostname can be appended with ":2". Read the manual supplied with your MPI distribution for further information.

Please note: FVCOM has been explicitly parallelized using the MPI Message Passing Libraries. This is considered a low-level parallelization because it is directly programmed into the code. A high-level parallelization is also possible through the compiler. Many modern compilers come with a flag that will instruct the compiler to perform a loop-based decomposition of the code while it is compiling. Afterwards the code can be executed in parallel using compiler specific commands and environmental variables. This method of parallelization is completely transparent to the user and can be performed directly on the serial (single processor) FVCOM code. However, this method of parallelization does not in general result in good parallel efficiency and will not scale well to a large number of processors. If you have a dual processor workstation and you intend to normally execute the code on a single processor and occasionally on both processors, this may be a reasonable option because it does not require installation of the MPI libraries and parallel environment or compilation of the METIS libraries. Simply

read the documentation for your compiler for auto parallelization and compile the serial FVCOM code with the appropriate flags.

Chapter 15: Model Setup

The model must be given a case identification string, heretofore "casename", which will form the prefix for most input and output files. The string must be less than 80 characters and should not contain blanks or symbols. Models at SMAST use three letter acronyms (e.g. GOM gom = Gulf of Maine Model). The runtime control parameter file is named *casename_run.dat* and should be placed in the directory where the fvcom executable (fvcom) resides. All other input files should be placed in the directory pointed to by the variable INPDIR in *casename_run.dat* as explained in section 15.1 describing the runtime control variables. A description of all the FVCOM input files is provided in Section 15.2. Not all input files are required for a model run. Section 15.3 outlines the necessary input files for running particular cases. Section 15.4 provides detailed information on input file format for required files and Section 15.5 provides detailed information on setting up the FVCOM modules.

15.1 FVCOM Runtime Control Parameter File *casename_run.dat*

The file *casename_run.dat* contains all parameters used to control the execution of FVCOM. The file is designed to have a flexible format so that alignment of data values with particular columns is not necessary. This was done to eliminate incorrect reads of values which that can commonly occur with tab formatted input. An example control file *exa_run.dat* is included with the code distribution as well as the control files for the two included test cases. A full description of input file formatting is included in the header of these three files. A brief description follows. In the FVCOM runtime control file format, a variable is given a value by writing the variable name (capital letters) followed by one or more blanks followed by an equal sign, followed by one or more blanks and finally the value for this variable. For example the following subset of variables demonstrates each type of variable seen in the runtime control file.

```
NSTEPS = 20  !Test Case 2
BAROTROPIC = F
INPDIR = input_dir
```

```
!Below we set the external time step
DTE = 5.
DPTHSL = 0.  –10.  –20.
```

Note from the example the following rules:

1. Variables are set through the following syntax:
   PARAMETER_NAME = Value
   where PARAMETER_NAME is the exact parameter name as described in this section. Value is the user defined value for the parameter. Note that one or more spaces must separate the equal sign from PARAMETER_NAME and Value.
2. There are five parameter types: string, logical, real, integer, and vector. The logical type (example BAROTROPIC above) must be assigned the value T (true) or F (false). The real type (example DTE above) must be assigned a numeric value with decimal point included. The integer type (NSTEPS) must be given an integral numeric value. The vector type (DPTHSL) receives one or more float type values. Finally the string type (INPDIR) must be assigned a string value (note: quotation marks are not necessary).
3. Variables may be provided in any order. It is best to follow the rough ordering provided in the example file (*exa_run.dat*) or runtime control files for the test cases (*tst_run.dat* and *chn_run.dat*) because the variables are grouped according to their purpose (input/output, time step control, etc) and the ordering corresponds to the runtime control parameter description in this manual.
4. Comments may be placed between variables or following the assignment of variables if preceded by the exclamation mark (!) as shown in the example above.
5. Blank lines in the input file are allowed because they are ignored by the parser.
6. Additional variables are allowed because they are ignored by the parser. For example, the variable WQM_ON activates the water quality model at runtime. However, the water quality module may not have been included in the compilation of the code. If the WQM_ON parameter is assigned in the runtime control parameter file, it will be ignored.

7. In most cases, an incorrectly assigned or missing runtime control variable will cause the code to halt and produce information about the error.
8. Vector values may not exceed the 80th column in the input file. If the space required to specify the vector exceeds this, the return character (\\) must be placed at the end of the line. See file *exa_run.dat* for example.

Parameters Controlling Time Integration

DTE	External time step (seconds)	Real
ISPLIT	Ratio of internal mode time step to external mode time step	Integer
IRAMP	Number of iterations over which to ramp up model forcing	Integer
NSTEPS	Total number of internal time steps to integrate	Integer

Please note that logical variables must be given a value of T (true), or F (false) as BAROTROPIC is in this example. Float variable must contain a decimal point. String variables must be less than 80 characters long. The variables may be provided in any order. It is best to follow the rough ordering provided in the example files because variables are grouped according to type (input/output, time step control, etc) and ordering corresponds to the descriptions given in this manual. Comments may be placed in lines between variables if preceded by an exclamation mark (!). Comments may be placed after variable values if preceded by an exclamation mark. If a required variable is not assigned a value, the code should inform the user of the missing or incorrectly assigned variable and halt.

Parameters Controlling Input/Output

INPDIR	Path to directory containing input files	String
OUTPUT	Path to directory for run ouput	String
INFOFILE	File to dump run output. Use *screen* for screen dump.	String

IREPORT	Iteration interval for printing flow field statistics.	Integer
IRECORD	Iteration interval for printing binary output.	Integer
IDMPSMS	Iteration interval for dumping sms output.	Integer
IRESTART	Iteration interval for dumping restart files.	Integer
RESTART	Select a model start type. This is a character with three choices and each of them is related to "tidal forcing setup (S_TYPE)". *cold_start*: If S_TYPE = *non-julian*, the model will start from zero velocity field and forcings are ramped from zero to their full values over IRAMP. If S_TYPE = *julian*, the model will start from a specified flow and elevation fields determined by the Foremen's tidal forecasting model. *hot_cold_s*: The model start with a restart file from the model output for the barotropic case (usually in this case the only tidal forcing is considered) and new initial fields of water temperature and salinity. If this type is chosen, S_TYPE should use *julian*. Users can modify the code to consider a case without *julian* tidal forcing. *hot_start*: The model start with a restart file from the model output at a selected time. This is usually used for the restart case with a binary output available.	String

Parameters Controlling Ouput of Average Data Fields

AVGE_ON	Activates dumping of average data fields	Logical
INT_AVGE	Iteration interval over which to average fields	Integer
BEG_AVGE	Iteration at which to begin averaging	Integer
NUM_AVGE	Number of intervals to dump	Integer
CDF_OUT_AVE	Activates NetCDF output of average data fields	Logical
CDF_VDP_AVE	List of variables that are output to NetCDF files	Vector

Parameters Controlling Bottom Friction Calculation

BFRIC	Bottom stress drag coefficient	Real
ZOB	The bottom roughness height (meters). There is a large range for this value in coastal regions. It generally varies between 0 and 60mm but may be larger. Users may specify a horizontal variation from observational data by modifying brough.F	Real
BROUGH_TYPE	Control the up-bound limit of the bottom roughness calculated by the log layer theory.	String

Parameters Controlling Diffusivity Coefficients

HORZMIX	Controls the calculation method for the horizontal diffusion. Options: *closure*: The horizontal diffusion is calculated using the Smagorinsky's parameterization method. *constant*: The horizontal diffusion coefficient is needed to be specified in "HORCON".	String
HORCON	If *closure* is chosen, this value represents the constant multiplicative coefficient in the Smagorinsky parameterization, which is usually taken as 0.2 or smaller in FVCOM. If *constant* is chosen, this value represents the horizontal diffusion coefficient used for the model run. Note: FVCOM can run stably without horizontal diffusion terms.	Float
HPRNU	The horizontal Prandtl number defined as the ratio of the horizontal eddy viscosity to the horizontal thermal diffusivity. It is usually specified to be 1.0 or larger.	Float
VERTMIX	Controls the calculation method for the vertical eddy viscosity and thermal diffusivity (characters). There are two choices: *closure*: The vertical eddy viscosity and thermal diffusivity are calculated using a turbulent closure scheme (currently MY level 2.5); *constant*: The vertical eddy viscosity and thermal diffusivity are specified by users by	String

"UMOL"

UMOL	If *closure* is selected, UMOL represents a background mixing (m^2/s). It usually is specified to be 10^{-4}-10^{-5} or smaller; If *constant* is selected, UMOL represents a real eddy viscosity value specified by users.	Float
VPRNU	The vertical Prandtl number defined as the ratio of the vertical eddy viscosity to the vertical thermal diffusivity. It is usually specified to be 1.0 or larger.	Float

Parameters Controlling Model Run Mode

BAROTROPIC	Suppresses calculation of salinity/temperature fields. No baroclinic pressure gradient terms included.	Logical
TEMP_ON	Activates update of the temperature equation.	Logical
SALINITY_ON	Activates update of the salinity equation.	Logical

Parameters Controlling Density Calculation

C_BAROPG	The type of the baroclinic pressure gradient terms (characters). There are two choices: *sigma*: The baroclinic pressure gradient terms are calculated on the sigma-coordinate; *other*: The baroclinic pressure gradient terms are calculated on the z-coordinate. This choice is not recommended, because we found that no matter how many vertical z-levels are chosen, a small error due to the density interpolation in the element closest to the bottom on the slope can accumulate with time.	String
CTRL_DEN	Controls the method for density calculation. There are two choices: *sigma-t*: The density calculated in the model represents	String

	the sigma-t. *pdensity*: The density calculated in the model represents the potential density.	
VERT_STAB	Activates adjustment of vertical instability adjustment to mimic convective overturning.	Logical
	Parameters Controlling the Atmospheric Forcing	
H_TYPE	Selects heat flux input calculation type. Options: *body_h*: The shortwave radiation is added into the model as a body forcing in the subroutine "adv_t". *flux_h*: The shortwave radiation is added into the model in "vdif_ts" as layer fluxes.	String
M_TYPE	Selects meteorological forcing type. Options: *uniform*: The forcing is spatially uniform. *non-uniform*: The forcing is spatially variable. In this case, the meteorological forcing is input directly from the pre-created data files (for example, MM5 output files).	String
WINDTYPE	Input type of the surface wind data. Options: *stress*: The wind data input is the x and y component of wind stress. *speed*: The wind data input is the wind speed and direction.	String
ZETA1	The attenuation length for the longer wavelength component of the shortwave irradiance.	Float
ZETA2	The attenuation depth for shorter wavelength component of the shortwave irradiance.	Float
RHEAT	The fraction of the total flux associated with the longer wavelength irradiance.	Float

THOUR_HS	The starting time to begin calculating heating flux contribution (hours).	Float

Parameters Controlling Correction of Temp/Salinity Updates

TS_FCT	Activates flux correction transport algorithm of temperature and salinity.	Logical

Parameters Controlling Water Depth Adjustment

DJUST	Adjust water depth relative to low water level. Used for estuarine modeling where bathymetric data represent water depth measured relative to a low water level (e.g., relative to mean low water).	Real
MIN_DEPTH	Minimum allowable depth in calculation. Water surface elevation may be thresholded to meet this requirement,	Real

Parameters Controlling Tidal Forcing

S_TYPE	Selects Type of Tidal Forcing at Open Boundary. *julian:* Tidal forcing (amplitudes and phases) at open boundary is specified using the Foremen's tidal forecasting model with real time. It is used for hindcast, nowcast and forecast model runs. *non-julian:* Tidal forcing (amplitudes and phases) at open boundary is specified by users-given amplitudes and phases). This is usually used for process-oriented studies.	String
DELTT	Time interval (seconds) between data sets for Julian tidal forcing.	Real

!Set up the parameters for reference levels for hydrographic data input

Parameters Controlling Standard Depth Levels

KSL	Number of standard depth levels	Integer

DPTHSL	Standard depth levels (meters)	Vector

Parameters Controlling Sigma Level Distribution

P_SIGMA	Power of the formulae used to determine the sigma level: $\sigma(k) = [(k-1)/(kb-1)]^{P\_SIGMA}$ P_SIGMA=1: Uniform sigma layers; P_SIGMA=2: Layer satisfying a parabolic function with high vertical resolution near the surface and bottom.	Real
KB	Number of standard sigma levels	Integer

!Set up the logical control parameter for the Lagrangian particle tracking

LAGCTRL	The logical control parameter for particle tracking (character). Options: "on": Particle tracking is on "off": Particle tracking is off	

!Set up the parameter controlling the restart dump

IRESTART = x

!Set up the standard depth levels

KSL =	Number of standard levels (integral)	
DPTHSL =	Standard level depths (meters)	

Parameters Controlling 3-D Lagrangian Particle Tracking

LAG_ON	Activates Lagrangian tracking	Logical

Parameters Controlling Time Series Output

PROBE_ON	Time series output on	Logical

Parameters Controlling Water Quality Module

WQM_ON	Water Quality Module on [module must be included]	Logical
NB	The number of water quality variables in WQM	Integer
BENWQM_KEY	Activates the benthic process in WQM	Logical

Parameters Controlling Wet/Dry Treatment

WET_DRY_ON	Activates wet/dry treatment [module must be Included]	Logical

Parameters Controlling SST Data Assimilation

SST_ASSIM	Activates SST data assimilation	Logical
RAD_SST	Influence radius of observations on data (meters)	Real
GAMA_SST	SST data quality factor. Usually specified to be 1.0	Real
GALPHA_SST	Nudging factor. Normally scaled by slowest physical adjustment process.	Real
ASTIME_WINDOW_SST	½ width of data assimilation window	Real
IAV_DAY	Number of days over which to average computed SST for use in nudging.	Integer

!Set up parameters controlling the sigma-level distribution

P_SIGMA = Power of the formulae used to determine the sigma level (real or integral): $\sigma(k) = [(k-1)/(kb-1)]^{P-SIGMA}$

P_SIGMA=1: Uniform sigma layers;

P_SIGMA=2: Layer satisfying a parabolic function with high vertical resolution near the surface and bottom.

P_SIGMA can be any real number, too.

KB = Number of sigma levels (integral)

!Set up the input time interval of the water elevation data at the open boundary

DELTT = Time interval for the water elevation data input at the open boundary (unit: seconds).

!Set up parameters controlling the SST data assimilation

SST_ASSIM = The logical parameter controlling the SST data assimilation:

"F": False - the SST data assimilation is off

	"T": True - the SST data assimilation is on
RAD_SST =	The affecting radius (meters)
GAMA_SST =	The SST data quality factor (real). In real application, it usually is specified to be "1.0"
GALPHA_SST =	The nudging factor that keeps the nudging term to be scaled by the slowest physical adjustment process.
ASTIME_WINDOW_SST =	Half a time length of the SST assimilation window
IAV_DAY =	The days used to average the SST data: The time interval (days) of the SST data assimilation

Parameters Controlling Current Data Assimilation

CURRENT_ASSIM	Activates current data assimilation	Logical
RAD_CUR	Influence radius of observations on current data (meters)	Real
GAMA_CUR	Current data quality factor. Usually specified to be 1.0	Real
GALPHA_CUR	Nudging factor. Normally scaled by slowest physical adjustment process.	Real
ASTIME_WINDOW_CUR	½ width of data assimilation window	Real

!Set up parameters controlling the current data assimilation

CURRRENT_ASSIM =	The logical parameter controlling the current data assimilation:
	"F": False - the current data assimilation is off
	"T": True - the current data assimilation is on
RAD_CUR =	The affecting radius (meters) for current assimilation
GAMA_CUR =	The current data quality factor (real). In real application, it usually is specified to be "1.0"
GALPHA_CUR =	The nudging factor that keeps the nudging term to be scaled by the slowest physical adjustment process.
ASTIME_WINDOW_CUR =	Half a time length of the current assimilation window

Parameters Controlling Temp/Salinity Data Assimilation

TS_ASSIM	Activates temp/salinity data assimilation	Logical
RAD_TS	Influence radius of observations on temp/salinity data (meters)	Real
GAMA_TS	Current data quality factor. Usually specified to be 1.0	Real
GALPHA_TS	Nudging factor. Normally scaled by slowest physical adjustment process.	Real
ASTIME_WINDOW_TS	½ width of data assimilation window	Real

!Set up parameters controlling the wet/dry point treatment variables

WET_DRY_ON = The logical parameter that controlls the wet/dry model run (character):

"F": False - the wet/dry point treatment is off

"T": True - the wet/dry point treatment is on

MIN_DEPTH_WD = The thickness of the viscous layer specified at the bottom (meters)

Parameters Controlling NetCDF Output

CDF_OUT	Activates NetCDF output	Logical
CDF_INT	Iteration interval for NetCDF output	Integer
CDF_STK	Input CDF_STK outputs in each file	Integer
CDF_VDP	List of variables to output to NetCDF file	Vector
INFO1	Data Description String 1	String
INFO2	Data Description String 2	String

Parameter Controlling Richardson Number Dissipation Correction

SURFACEWAVE_MIX	True on if surface wind induced	Logical

Parameters Controlling Open Boundary Temp/Salt Nudging		
TS_NUDGING_OBC	Activates temp and salinity nudging on OBC	Logical

ALPHA_OBC	Nudging coefficient for temp and salinity on OBC.	Real

Parameters Controlling Open Boundary Temp/Salt Time Series Nudging		
TSOBC_ON	Activates temp and salinity nudging on OB	Logical
ALPHA_SERIES_OBC	Nudging coefficient for temp and salinity on OB	Real

Parameters Controlling 2-D Momentum Balance Checking Output		
OUT_BALANCE	Activates 2-D momentum balance checking output	Logical
NUM_BALANCE	The amounts of cells selected for the 2-D momentum balance output	Integer
NO_CELL	The cell ID for the 2D momentum ouput	Integer

Parameter Controlling the Type of Open Boundary Conditions for Temp/Salt

TYPE_TSOBC	Type of open boundary condition (see module MOD_OBCS)	Integer

15.2 FVCOM Input Files

FVCOM input files must be placed in the directory pointed to by the variable INPDIR in the runtime control parameter file described in section 15.1. All input files are prefixed by the string "casename" referring to the application description string chosen by the user. A description of the input files and their primary data is provided below.

1) **Casename_bfw.dat:** The bottom freshwater input values including number of sources, time, and discharge rate, etc.

2) **Casename_cor.dat:** The latitudes of triangular nodes which are used to calculate the Coriolis parameter.

3) **Casename_dep.dat:** The water depth at all node points.

4) **Casename_el_obc.dat:** Tidal amplitudes and phases at the open boundary (for the case with no Julian time).

5) **Casename_elj_obc.dat:** Tidal amplitudes at the open boundary (for the case with Julian time). In this case, the time dependent amplitudes on the open boundary are generated using Foreman's program with Julian time.

6) **Casename_el_ini.dat:** The initial field of surface elevation at individual triangular nodes calculated using the Foreman harmonic tidal analysis program. This file needs to be generated based on calibrated amplitudes and phases of major tidal constituents from the model tidal simulation.

7) **Casename_grd.dat:** Triangular meshes (numbers of individual triangle identification and its three nodes and the x and y locations of individual node points)

8) **Casename_hfx.dat**: The real-time fields of the surface heat flux and shortwave irradiance. Users can specify this input themselves. In our application for the GOM/GB region, this file is an output file from our regional MM5 model.

9) **Casename_its.dat**: Initial fields of the water temperature and salinity.

10) **Casename_jmpobc.dat**: Identification number of open boundary nodes where a frictional geostrophic inflow correction is made. This input file is usually used in a near-shore upstream open boundary where the inflow can be parameterized by the wind stress. This condition was added by J. Pringle (UNH) to add the inflow on the Scotian Shelf in the Gulf of Maine/Georges Bank model.

11) **Casename_lag_ini.dat**: The initial positions of particles to be tracked using Lagrangian tracking.

12) **Casename_mc.dat**: Meteorological forcing values: wind velocity, heat flux and precipitation/evaporation. This input file is used only for a spatially uniform meteorological forcing case.

13) **Casename_obc.dat**: Forcing values at open boundary nodes.

14) **Casename_riv.dat**: River discharge data including number of rivers, discharge volume, discharge temperature, and discharge salinity.

15**) Casename_run.dat**: Set up parameters controlling the model run.

16) **Casename_spg.dat**: Parameters for a sponge layer for damping at the open boundary.

17) **Casename_uv_ini.dat**: The initial field of currents at individual triangular centroids calculated using the Foreman harmonic tidal analysis program. This file needs to be generated based on calibrated tidal ellipses of the major tidal constituents from the model tidal simulation.

18) **Casename_wnd.dat:** The real-time field of wind velocity or wind stress used for external and internal modes. In our current experiments, it is an output file from our regional MM5 model.

19) **Casename_tsobc.dat** The time series of the temperature and salinity used for nudging on open boundary nodes. This is an input file that is usually constructed from observational data or output from a larger domain model.

20) **Casename_meanflow.dat** The mean flow volume transport specified on the open boundary. The vertical distribution of the transport can be determined like that used in the river discharge input file.

15.3 Input Files Required for Specific Setups

Required Input Files for All Setups	
Input File	Note
Casename_grid.dat	If it is created by users' own grid generation program, be sure to have the same format as FVCOM requires.
Casename_riv.dat	Can specify zero rivers

Casename_bfw.dat	Can specify zero fresh water sources
Casename_obc.dat	Can specify no open boundary nodes
Casename_cor.dat	
Casename_spg.dat	Can specify zero sponge nodes
Casename_dep.dat	

Startup Dependent Files

Startup Type	Required Files
Cold_start	Casename_el_ini.dat
	Casename_uv_ini.dat
	Casename_its.dat
Hot_start	Casename_restart.dat
Hot_start_s	Casename_its.dat
	Casename_restart.dat

Meteorological Forcing Dependent Files

Forcing Type	Input Files
Spatially uniform, temporally-varying and temporally constant	Casename_mc.dat
Spatially distributed	Casename_wnd.dat
	Casename_hfx.dat

Open Boundary Surface Elevation Specification Files

No open boundary nodes	NO FILE
No open boundary nodes with prescribed surface elevation data (all are radiative)	NO FILE
Non-Julian surface elevation specification	Casename_el_obc.dat
Julian surface elevation specification	Casename_elj_obc.dat

15.4 Input File Formats for Primary Input Files

1. Casename_bfw.dat

Comments (80A)
IBFW_GL
(NODE_BFW(I), I=1, IBFW_GL)
NBFWTIME
BFWTIME ! time: 1
(RTEMP(J), J=1, IBFW_GL)
BFWTIME ! time: 2
(RTEMP(J), J=1, IBFW_GL)
….
…..
BFWTIME ! time: NBFWTIME
(RTEMP(J), J=1, IBFW_GL)

Line 1
Line 2: IBFL_GL is an integral number of the freshwater discharge points at the bottom. This number is read in the subroutine called "getdim.f" and "in the subroutine called "bcs_force.f". If no freshwater is specified at the bottom, just put "0".

Line 3: NODE_BFW(I) - The *i*th node ID of the bottom freshwater discharge.

Line 4: NBFWTIME - The number of the input times for the bottom freshwater discharge (integral);

Line 5: BFWTIME - The time at which the freshwater discharges

Line 6: RTEMP: an array with the one dimensional size of IBFW_GL. This array includes the freshwater discharge rate (m^3/s) at nodes of NODE_BFW for the time given in BFWRIME.

The number of appearances of lines for BFWTIME and RTEMP is equal to NBFWTIME. The bottom freshwater discharge data are read in "bcs_force.f".

2. Casename_cor.dat

x (meters)	*y* (meters)	Lat (degree)
:	:	:
:	:	:
:	:	:

This is a data array with three columns. (x, y) and Lat are the location and latitude of individual node points on each triangular mesh. Total rows of this array are equal to the total number of node points. No header information is required in this input file.

3. Casename_dep.dat

x (meters)	*y* (meters)	*d* (meters)
:	:	:
:	:	:
:	:	:

This is a data array with three columns. (x, y) and d are the location and water depth of individual node points on each triangular mesh. Total rows of this array are equal to the total number of node points. The depth in FVCOM is specified at the node point. No header information is required in this input file.

For estuaries, the water depth was usually measured at the lowest water level. Therefore, the adjustment depth must be specified in "Casename_run.dat".

4. Casename_el_obc.dat

Comments (80A)
Total number of the OBS：IOBCN1

1st Node ID, Mean Level
A_S_2, A_M_2, A_N_2, A_K_1, A_P_1, A_O_1
P_S_2, P_M_2, P_N_2, P_K_1, P_P_1, P_O_1

2nd Node ID, Mean Level
A_S_2, A_M_2, A_N_2, A_K_1, A_P_1, A_O_1
P_S_2, P_M_2, P_N_2, P_K_1, P_P_1, P_O_1
……..
……
IOSCN1th Node ID, Mean level
A_S_2, A_M_2, A_N_2, A_K_1, A_P_1, A_O_1
P_S_2, P_M_2, P_N_2, P_K_1, P_P_1, P_O_1

Line 1:
Line 2

Line 3
Line 4
Line 5

Line 6
Line 7
Line 8
……..
……..
Line 3×IOBCN1
Line 3×IOBCN1+1
Line 3×IOBCN1+2

User can write any comment on the first line.

Total number of the IOSCN1 must be an integral number

A_( ): The amplitude of tidal constituent ( ).
P_( ): The phase of the tidal constituent ( ).

5. Casename_elj_obc.dat

(RTEMP1(J,I),J=1,IOBCN_GL)

RTEMP(J, I): An array including the water elevation (meters) at individual open boundary nodes.

IOBCN_GL: Total number of the open boundary node points. The order of the data written in the column of RTEMP must be the same as the order of the open boundary node points in the input file "Casename_jmpobc.dat".

I: Indicates the number of the data input. Each row of RTEMP contains the water elevation values at open boundary nodes with a time interval of "DELTT". The value of DELTT is specified in "Casename_run.dat".

Note: Users also can use the Foreman program to calculate the water elevation at the open boundary node point for a selected time. This can avoid having a big input file for water elevation at open boundary.

6. Casename_el_ini.dat

EL(1)
EL(2)
.
.
.
EL(M)

EL: The water elevation.

The data is written in a one column file with a total number of rows equal to the total number (M) of node points. No specific format is required. The unit of the water elevation is meters.

7. Casename_grd.dat

E	N_1	N_2	N_3	Type
:	:	:	:	1
:	:	:	:	1
:	:	:	:	1

The grid input file consists of two parts: 1) integral numbers identifying elements and nodes and (2) the x and y locations of individual nodes.

Column E: Identify integral number of individual

element I (SMS program will list the elements from 1, 2… to N, but FVCOM can identify a random order of the element number file).

Column N_1-N_3: Identify integral numbers of the nodes of the element I listed in the same row.

Column Type: The element type information which automatically comes out from SMS grid generation program. This information is not needed for FVCOM. For user's created grid file, this column can be removed.

N X (meters) Y (meter)
: : :
: : :
: : :

Column N: Identify integral number of individual node I.

Columns X and Y: The x and y locations of the node I shown in the same row.

Note 1: In the grid file created by SMS, there is a fourth column in this part. That information is never used in FVCOM. For user's created grid file, this column can be removed.
Note 2: When integrating FVCOM in Spherical Coordinates, X and Y are Latitude and Longitude respectively.

8. Casename_hfx.dat

TIME (1)
(RTEMP1(1, J), RTEMP2(1, J), I=1, M)
TIME (2)
(RTEMP1(2, J), RTEMP2(2, J), I=1, M)
…….
…….
TIME (NCNT)
(RTEMP1(NCNT, J), RTEMP2(NCNT, J), I=1, M)

RTEMP1: Net heat flux
RTEMP2: Shortwave flux

M: Total number of node points.

Time: hours relative to the start time of the model run.

NCNT: Total number of the times for the heat flux input.

9. Casename_its.dat

Comment (80A)
Comment (80 A)
(T(J, 1), J=1, KSL)
(S(J, 1), J =1, KSL)
(T(J, 1), J=1, KSL)
(S(J, 1), J =1, KSL)
…….
…….
(T(J, M), J=1, KSL)
(S(J, M), J =1, KSL)

Line 1: Comments with a maximum 80 characters.

Line 2: Data type (characters).
Observed: from field data.
Constant: specified by users.

KSL: Number of standard levels

M: Total number of node points.

10. Casename_jmpobc.dat

N_OBS
N_ID (1)
N_ID (2)
…..
…..
N_ID(N_OBS)

N_OBS: Total number of the open boundary node points.

N_ID: The identification number of individual open boundary node points.

This data file consists of N_OBS+1 rows. Line one indicates the total number of the nodes and then line 2 to line N_OBS+1 list the identification numbers of N_OBS nodes.

11. Casename_lag_ini.dat:

Islag, Ielag, Iolag
Pdx1(1), Pdy1(1), Pdz1(1)
Pdx1(2), Pdy1(2), Pdz1(2)
……..
……..
Pdx1(Ndrft), Pdy1(Ndrft), Pdz1(Ndrft)

Islag: The time step at which the particles are released.

Ielag: The time step at which the particle tracking is ended.

Iolag: Iteration interval for output of particle locations and velocities.

Ndrft: Total numbers of the particles.

Pdx1(i), Pdy(i), Pdz1(i): The x, y and σ locations of the ith particle at the release time. Number of rows for the location should equal to Ndrft.

12. Casename_mc.dat:

Time (1)
QPREC, QEVAP, WDS, WDD, HFLUX, HSHORT
Time (2)
QPREC, QEVAP, WDS, WDD, HFLUX, HSHORT
…….
…….
…….
…….
Time (NCNT)
QPREC, QEVAP, WDS, WDD, HFLUX, HSHORT

Time: Time (hours) is measured relative to the start time of the model run. This is the start time for a "cold start" run.

QPREC: Amount of precipitation (m/year)

QEVAP: Amount of evaporation (m/year)

WDS: Wind speed (m/s)

WDD: Wind direction in degrees from which the wind blows. It is measured clockwise from north.

HFLUX: Net surface heat flux (W/m^2)

HSHORT: Shortwave irradiation flux at the surface (W/m^2)

NCNT: If no meteorological forcing is included, this input file includes 4 rows. Time (1) = 0.0 and Time (2) = 9999999.00 (one number issecond number must be larger than the total model run hours).

13. Casename_obc.dat

Comments (80A)

No.	Node_ob (*i*)	Node _oi (*i*)	Node Type	T	S
1	x	x	x	x	x
2	x	x	x	x	x
.	.	.	.	.	.
.	.	.	.	.	.
NOB	x	x	x	x	x

No.: counting number of the open boundary node from 1 to NOB.

Node_ob (*i*): The node ID of the *i*th open boundary node.

Node_oi (*i*): The node_ID of the interior node closest to the

ith open boundary node.

Node Type: Specifies method of setting surface elevation. Use 0 for prescribed elevation (Julian/non-Julian). Use 1 for Sommerfeld radiative condition.

T: Open boundary temperature for nudging (not for the time series nudging).

S: Open boundary salinity for nudging (not for the time series nudging).

NOB: Total number of the open boundary nodes.

To make the radiation open boundary condition easy to be set up, we request that Node_oi(*i*) is parallel to Node_ob(*i*) .

14. Casename_riv.dat

Inflow_type Point_st_type
Numqbc
Node_ID (1)
Node_ID (2)
…..
Node_ID (Numqbc)
Node_ID (1), (VQDIST(1,k), k =1, kb-1)
Node_ID (2), (VQDIST(2,k), k =1, kb-1)
…….
Node_ID (Numqbc),
(VQDIST(Numqbc,k), k =1, kb-1)
Nqtime
Ttime(1)
(Qdis(j), j=1, Numqbc)
(Tdis(j), j=1, Numqbc)
(Sdis(j), j=1, Numqbc)
…….
…….

Inflow_type: The method used to add the river discharge (characters string).
There are two choices:
1) "node"- discharge water is added from node points as point sources;
2) "edge"- discharge water is added from boundary lines of triangles.
Point_st_type: The method used to calculate water salinity or temperature at the discharge point (characters). There are two choices:
1) "calculated"- the salinity or temperature at boundary nodes (where the discharge is given) is calculated using the mass conservative salinity or temperature equation;
2) "specified"- the salinity or temperature at boundary nodes is specified by

Ttime(Nqtime)
(Qdis(j), j=1, Numqbc)
(Tdis(j), j=1, Numqbc)
(Sdis(j), j=1, Numqbc)

If the inflow_type is "edge", "Node_ID (I)" is replaced by "ICELLG(I)": the *i*th cell ID number for the *i*th river.
All the others remain unchanged as the format shown above.

users.

Numqbc: Total number of river discharge points (number of rivers).

Node_ID(I): The ith node ID number for the ith river. The number of rows should equal Numqbc.

VQDIST(i,k): percentage of the river discharge in the kth sigma level at the *i*th river.

Nqtime: Total number of the river discharge records input into the model.

Ttime(i): Time (in hours) for the *i*th record.

Qdis(j): Discharge rate at the jth river.

Tdis(j), Temperature of the discharge water at the jth river;

Sdis(j): Salinity of the discharge water at the jth river.

15. Casename_spg.dat:

N_sp

Node_ID (1), R(1), F_f (1)

Node_ID (2), R(2), F_f (2)

……

……

Node_ID(N_sp), R(N_sp), Ff (N_sp)

N_sp: Total number of outer-side boundary nodes at which the friction is added.

Node_ID(i): The *i*th sponge center node ID

R(i): The *i*th sponge layer's affecting influence radius.

F_f(i): Damping coefficient of the *i*th sponge layer.

16. Casename_uv_ini.dat

(u(1,k), v(1,k), k=1, kb-1)

(u(2,k), v(2,k), k=1, kb-1)

……

……..

(u(N,k), v(N,k), k=1, kb-1)

kb: The total number of the sigma-levels in the vertical.

N: The total number of triangular cells.

Note: the order of *i*=1, N must be the same as listed on the grid input file.

17. casename_wnd.dat

Time (1)

$(dtx(i,1), dty(i,1),\ i=1, N)$

Time (2)

$(dtx(i,2), dty(i,2),\ i=1, N)$

……

……

Time (NCNT)

$(dtx(i, NCNT), dty(i, NCNT),\ i=1, N)$

N: Total number of triangular cells and the order of *i*=1, N must be the same as listed on the grid input file.

Time (i): The time (in hours) at which the *i*th wind stress is input.

NCNT: Total number of the wind input times

dtx: x-component of the wind stress

dty: y-component of the wind stress

18. casename_tsobc.dat

Time (1)

N0\LAY 1 2 … KBM1

1 T T … T

……..

IOBCN T T … T

1 S S … S

……..

IOBCN S S … S

Time(2)

……

Time (toll days)

Time (i): The time (in days) at which the *i*th temperature and salinity is input.

N0: Total node number of the open boundary

LAY: The total number of sigma levels

T: The temperature value.

S: The salinity value

……

19. casename_meanflow.dat

Nmfcell Cell_ID (1) Cell_ID (2) ….. Cell_ID (Nmfcell) Cell_ID (1), (VQDIST(1,k), k =1, kb-1) Cell_ID (2), (VQDIST(2,k), k =1, kb-1) ……. Cell_ID (Nmfcell), (VQDIST(Nmfcell,k), k =1, kb-1) Nqtime Ttime(1) (Qdis(j), j=1, Nmfcell) ……. ……. Ttime(Nqtime) (Qdis(j), j=1, Nmfcell)	Nmfcell: Total number of the open boundary cells in which the mean flow is specified. Cell_ID(I): The *i*th cell ID number for the *i*th open boundary cell. The number of rows should be Nmfcell. VQDIST(i,k): Percentage of the volume transport in the *k*th sigma level at the *i*th open boundary cell. Nqtime: Total number of mean flow volume transport records input into the model. Ttime(i): Time (in hour) for the *i*th record. Qdis(j): Volume transport rate (m^3/s) at the jth mean flow cell; positive value means flowing into the domain while negative value means flowing out of the domain

15.5 Setting up and Using FVCOM Modules

In this section, the setup for the Lagrangian particle tracking module, time series output module, NetCDF output module, data assimilation module, water quality module, and biological model module is described.

Lagrangian Particle Tracking (mod_lag.F)

The Lagrangian particle tracking module contains subroutines which trace neutral particle paths through the model domain using the 3-D velocity field. *Note that the current implementation will only work for single (serial) processor run.* A future

implementation will include particle tracking during parallel computation. Initial particle locations are set up in the file *casename_lag_ini.dat* which has the following format.

Format of Lagrangian Particle input file *casename_lag_ini.dat*

NDRFT ISLAG,IELAG,INT_LAG XP(1),YP(1),ZP(1) XP(2),YP(2),ZP(2) …. XP(NDRFT),YP(NDRFT),ZP(NDRFT)	NDRFT: Number of particles. ISLAG: The time step at which the particles are released. IELAG: The time step at which the particle tracking is ended INT_LAG: Iteration interval for writing particle location and velocities. XP(i),YP(i),ZP(i): The x, y and σ locations of the ith particle at the release time. Number of the rows for the location should equal to NDRFT

Particle tracking is activated with the runtime control parameter LAG_ON in the *casename_run.dat* file. Set LAG_ON = T to activate. While running, particle location and velocity data will be dumped every INT_LAG iterations into the out subdirectory of the output directory specified by OUTDIR in the runtime control parameter file. There are two separate files for the data and each output of the files has the following formats:

Format of Lagrangian Particle output file *casename_lag_out.dat*

Time, (xp(i),yp(i),ztrue(i),I=1,NDRFT)	NDRFT : Number of particles Time: Model Time in Days XP(i),YP(i),ZTRUE(i): The x, y and z locations of the ith particle at the release time. A new row will be written with each output.

Format of Lagrangian Particle output file *casename_lag_ouv.dat*

Do I=1,ndrft Time, xp(i),yp(i),zp(i),up(i),vp(i),wp(i),elnum(i) End do	Time: Model time in days XP(i),YP(i),ZP(i): The *x*, *y* and sigma locations of the ith particle at the release time. UP(i),YP(i): Cartesian velocities in cm/s. WP(i): omega velocity in .0001 m/s Elnum(i): Element containing particle.

where time is the model time in days, and xp,yp,ztrue are the true (Cartesian) particle locations for each particle. If the particle leaves the horizontal boundaries of the domain, it will remain at its last know position and take on the value of zero velocity. The elnum value in the *lag_ouv.dat* file will take on a value of zero.

Time Series Module (mod_probe.F)

The times series output module contains subroutines which setup the variable for time series output and dump them to a file at times specified by the user in the setup file. Time series output is activated by setting PROBE_ON = T in the runtime parameter file. Probe setup files are specified as *casename_timeseriesXX.dat* where XX is a two digit number from 01 to 99. A typical setup file is as follows:

```
!======TIME SERIES OUTPUT FOR OBSERVATION POINT 1
DLOC = 1012
D_TYP = element
D_TIT = river_mouth_u_vel
D_DES = observation data set 1
K_ONE = 1
K_TWO = 1
O_INT = 1
VAR = u
VNAME = x-velocity
```

Descriptions of the variables is as follows.

!=====…..	Comment, not used
D_LOC	Global element or node number of data location.
D_TYP	Data type (element or node)

D_TIT	Output file name
D_DES	Data description (max 80 characters)
K_ONE	Initial sigma value of data output
K_TWO	Final sigma value of data output
O_INT	Iteration interval for dumping output
VAR	Variable to dump
VNAME	String (max 80 chars) descriptor for variable

Note that the file format follows the free-format conventions of the runtime parameter control file *casename_run.dat* outlined in section 12.1. A time series must be set up separately for each data location for each variable. Note that for surface values of a 3-D variable, use a value of 1 for both K_ONE and K_TWO. Output will be placed in the file specified by the string D_TIT in the subdirectory time series of the main output directory. If a file already exists, a new file will be created with the name D_TIT-1. Some information on the time-series setup is provided in normal code output before the calculation begins and may provide information helpful for debugging. The D_DES and VNAME strings provide information to the header of the output file and may provide useful reference at a later time. If output of a variable which is not currently setup for time series, output may be added by modifying subroutine PROBE_STORE in mod_probe.F. A warning will be given if this is the case and the code will halt.

NetCDF Output (mod_ncdio.F)

NetCDF is a machine-portable, self-describing data format which is commonly used in the oceanographic and atmospheric communities. Output in this format requires calls to NetCDF libraries which must be linked to the code during compile time. To use NetCDF output, the makefile variable NETCDF must be set to *yes* and paths to the NetCDF libraries and include files must be supplied to the makefile variables LIB_NETCDF and INC_NETCDF respectively. When the code is compiled, the NetCDF output module will be included with compilation. Control of the output is specified through the runtime control parameter file *casename_run.dat*. See section 12.1 for information on the relevant variables. Variables to be dumped are specified in the

runtime parameter file. Additional variables which are not currently setup may be specified by modifying the subroutine OUT_NETCDF in the mod_ncdio.F module.

An example setup from the runtime parameter file is shown below.

```
CDF_OUT = T
CDF_INT = 90
CDF_VDP = u v ww s1 t1 rho1 kh
INFO1 = 1995 Fine Grid Gulf of Maine Run
INFO2 = Data Assimilation Included
```

The relevant control parameter variables are defined as follows (excerpt from full description of runtime control parameter variables in section 12.1).

Parameters Controlling NetCDF Output

CDF_OUT	Activates NetCDF output	Logical
CDF_INT	Iteration interval for NetCDF output	Integer
CDF_VDP	List of variables to output to NetCDF file	Vector
INFO1	Data description string 1	String
INFO2	Data description string 2	String

Data Assimilation Module (mod_assim.F)

The nudging data assimilation is an optional module incorporated in FVCOM for hindcast application. The current version of this module has been tested for the assimilation of remote-sensing-derived sea surface temperature (SST), current time series from moorings, and water temperature/salinity taken from hydrographic surveys. To help users to prepare the input data files used for nudging, we include some examples below.

a) SST Assimilation

The time interval for the SST data is usually chosen based on two factors: 1) number of clear SST images and 2) statistical confidence level. In general, the daily SST data contains many voids due to cloud cover. If the computational domain covers a large coastal ocean area, it is difficult to find continuous clear SST images on a daily base. On

the other hand, in many coastal regions, for example, in the Gulf of Maine/Georges Bank region, driven by synoptic meteorological variation, the near-surface water temperature features a low-frequency variation with a time scale of 5-7 days. It makes more sense to conduct the SST assimilation over a time scale that represents better the nature of the thermal dynamics and retains a certain statistical level. For these data, we usually conduct the SST assimilation based on the 5-day average field of the SST. The SST data assimilation is unique in that the observations are averaged over a long time period relative to the model time step. If the model is nudged at every time iteration with averaged SST data throughout the averaging period, this can introduce bias into the nudging. In FVCOM, this bias is eliminated by first integrating the model without SST nudging for the period over which observations have been averaged. During this time, the model SST is saved. After the simulation to the end of this period, the model returns to the beginning of the period and the flow field is reset with the values saved before the simulation stage began. In the second sweep, the SST data is assimilated by nudging the model SST's by the difference between the SST data and model data averaged over this period of time. The time period is selected by the variable IAV_DAY in the runtime control parameter file where the time specified is in days. The runtime control parameters relevant to SST assimilation are provided in section 12.1.

The SST data used for the nudging assimilation is prepared in an input file named "casename_sst.dat" and read in subroutine SET_SST_ASSIM_DATA in the data assimilation module. Code fragments of the Fortran program used to read these data are shown below:

```
SUBROUTINE SET_SST_ASSIM_DATA

!--Read SST Data-----------------------------------------------------!
!.....................................................................!
   READ(1,*) N_TIMES_SST

   DO I=1,N_TIMES_SST
     READ(1,*) TEMPF,N_ASSIM_SST
     DO J=1,N_ASSIM_SST
       READ(1,*)SST_OBS(J)%X,SST_OBS(J)%Y,SST_OBS(J)%SST(I)
       SST_OBS(J)%TIMES(I) = TEMPF
     END DO
   END DO
!.....................................
```

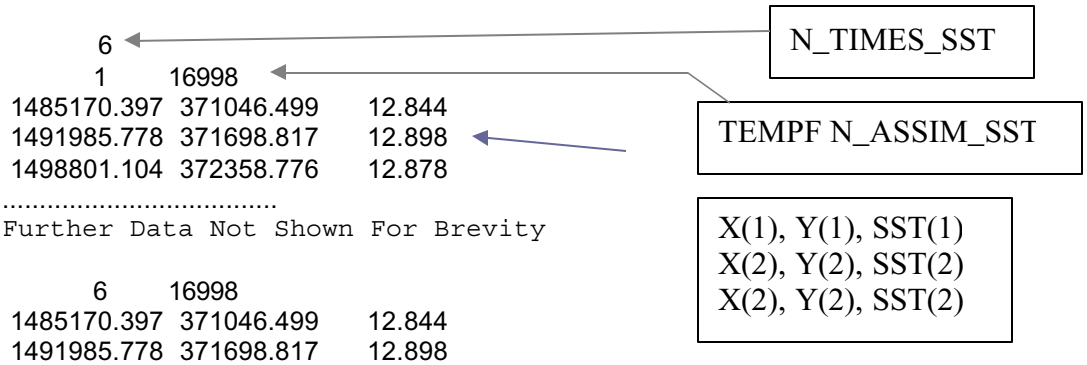
```
   RETURN
   END SET_SST_ASSIM_DATA
```

The parameters and arrays in this subroutine are defined as

N_TIMES_SST	Total times of the SST data assimilation that will be conducted during the entire period of the model run. This is an integral number.
TEMPF	The starting time of the nudging assimilation.
N_ASSIM_SST	The total number of the SST data points used for nudging assimilation at a start time of TEMPF.
SST_OBS(J)%X	The x location of the Jth data point.
SST_OBS(J)%Y	The y location of the Jth data point
SST_OBS(J)*SST(I)	The SST value at the Jth data point for the Ith nudging time.

:

An example of the SST input file used for nudging assimilation with a time interval of 5 days is given below. In this case, the total model integration time is one month and the nudging assimilation is carried out at days 1, 6, 11, 16, 21, 26 with a time integration window of 5 days.




```
       6
       1     16998
 1485170.397  371046.499     12.844
 1491985.778  371698.817     12.898
 1498801.104  372358.776     12.878
.....................................
Further Data Not Shown For Brevity

       6     16998
 1485170.397  371046.499     12.844
 1491985.778  371698.817     12.898
 1498801.104  372358.776     12.878
.....................................
Further Data Not Shown For Brevity
      11     16998
 1485170.397  371046.499     12.844
 1491985.778  371698.817     12.898
.....................................
Further Data Not Shown For Brevity

      16     16998
 1485170.397  371046.499     12.844
 1491985.778  371698.817     12.898
.....................................
Further Data Not Shown For Brevity
```

```
       21      16998
 1485170.397  371046.499     12.844
 1491985.778  371698.817     12.898
.....................................
Further Data Not Shown For Brevity

       26      16998
 1485170.397  371046.499     12.844
 1491985.778  371698.817     12.898
```

b) T/S Assimilation

The temperature/salinity (T/S) assimilation is usually conducted to merge the model-computed water temperature and salinity to observed values measured during hydrographic surveys. Users should realize that under conditions with an insufficient spatial/temporal coverage of the field measurements, this method may cause unrealistic temperature and/or salinity gradients at the boundary of the available data. For example, if the model starts with the initial conditions specified using climatologic averaged T/S fields, the model-predicted T and S may significantly differ from the actual CTD data taken in a particular time. To merge the model-predicted T/S fields to the CTD data taken in a small region in the computational domain, sharp gradients in T and/or S will likely occur around the boundary area of nudging. The gradients can be reduced by increasing the influence radius used in the nudging. However, this may decrease the effectiveness of the data assimilation by smoothing the gradients in regions where they should be large (for example, at a front) within the nudging region. Therefore, this method should be used with caution.

The T/S assimilation is a 4-D assimilation in time and space. There are two approaches coded in FVCOM. The first is carried out on scattered points in the model domain and the second is carried out for data which has been interpolated to the model grid.

The scatter point method. Two input files are required: "casename_ts.xy" and "casename_ts.dat". The "casename_ts.xy" contains 1) the total number of the scattered points included for the data assimilation, 2) the x and y locations of the measurement points, 3) the local water depth where the measurements was made, 4) the total number of measurements in the vertical, and 5) the directional difference (the angle between the tangent of the local isobath and x-coordinate) used for directionally-dependent nudging

($\Delta\theta$), and 6) the measurement depths in the vertical (a 1-D array). The "casename_tsXX.dat" is an array containing the measurement times and T/S at each depth in the vertical for each scattered-point measurement XX. These data are read in the subroutine called SET_TS_ASSIM_DATA shown below.

```
SUBROUTINE SET_TS_ASSIM_DATA

!---------------------------------------------------------------------!
!  SET UP ASSIMILATION DATA FOR TEMP/SAL OBSERVATIONS
|
!---------------------------------------------------------------------!


   FNAME = "./"//TRIM(INPDIR)//"/"//trim(casename)//"_ts.xy"
!
!.........................................
!
!--Read Number of Current Measurement Stations-----------------------!
!
   OPEN(1,FILE=TRIM(FNAME),STATUS='OLD')
   READ(1,*) N_ASSIM_TS
   ALLOCATE(TS_OBS(N_ASSIM_TS))

!
!--Read X,Y Coordinate of Measurement Stations-----------------------!
!

   DO I=1,N_ASSIM_TS
         READ(1,*)ITMP,TS_OBS(I)%X,TS_OBS(I)%Y,TS_OBS(I)%DEPTH,NLAY,
  *              TS_OBS(I)%SITA
     TS_OBS(I)%N_LAYERS = NLAY
     ALLOCATE(TS_OBS(I)%ODEPTH(NLAY))
     DO J=1,NLAY
       READ(1,*)TS_OBS(I)%ODEPTH(J)
       IF(TS_OBS(I)%ODEPTH(J) > TS_OBS(I)%DEPTH)THEN
         IF(MSR)WRITE(IPT,*)'OBSERVATION DEPTH',J,'OF
  *             TEMP/SALINITY OBS',I
         IF(MSR)WRITE(IPT,*)'EXCEEDS BATHYMETRIC DEPTH'
         IF(MSR)WRITE(IPT,*)'HALTING..........'
         CALL PSTOP
       END IF
     END DO
   END DO
   MAX_LAYER_TS = MAXVAL(TS_OBS(1:N_ASSIM_TS)%N_LAYERS)

!--Close Current Observation Global File-----------------------------!
!
   CLOSE(1)
!---------------------------------------------------------------------!
!  Open Temp/Sal Observation Files for Each Observation Point and Read
Data    !
!---------------------------------------------------------------------!
```

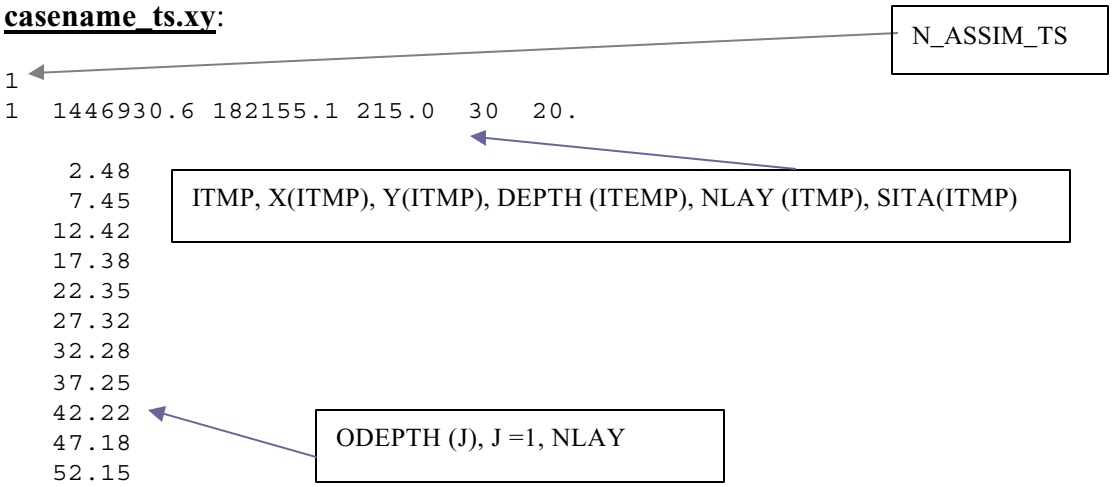
```
   DO I=1,N_ASSIM_TS
!.................................................
!----Make Sure Temperature/Salinity Observation File Exists-----------!
     WRITE(NAC,'(I2.2)')I
     ONAME =
"./"//TRIM(INPDIR)//"/"//trim(casename)//'_ts'//NAC//'.dat'
!.....................................................
!----Open Temp/Salinity Observation File for Read---------------------!
     OPEN(1,FILE=ONAME,STATUS='old')  ; REWIND(1)
!.....................................................................
        DO J=1,TS_OBS(I)%N_TIMES

READ(1,*)TS_OBS(I)%TIMES(J),(TS_OBS(I)%TEMP(J,K),TS_OBS(I)%SAL(J,K),K=1
,NLAY)
     END DO
     CLOSE(1)

   RETURN
   END SET_TS_ASSIM_DATA
```

The parameters and variables in the input file "casename_ts.xy" are defined as

N_ASSIM_TS:	Total number of scatter points (CTD measurement stations) included for data assimilation.
ITMP:	The sequence number of scatter points starting from 1 to N_ASSIM_TS.
TS_OBS(I)%X:	The x location of the scatter point corresponding to the sequence number of the scatter point shown on the first column.
TS_OBS(I)%Y:	The y location of the scatter point corresponding to the sequence number of the scatter point shown on the first column.
TS_OBS(I)%DEPTH:	The local mean water depth at the location of the scatter point shown on the first column.
NLAY:	The total number of measurement levels in the vertical at the location of the scatter point shown on the first column.
TS_OBS(I)%SITA:	The angle of the tangent of the local isobath to the x-coordinte at the location of the scatter point shown on the first column.
TS_OBS(I)%ODEPTH(J):	The measurement depth at the scatter point of ITMP. This is a 1-D array starting from 1 to NLAY. The depth is measured in meters.

Examples of the input file *casename_ts.xy* is given below:

casename_ts.xy:




```
1
1  1446930.6 182155.1 215.0  30  20.

    2.48
    7.45
   12.42
   17.38
   22.35
   27.32
   32.28
   37.25
   42.22
   47.18
   52.15
   57.12
   62.08
   67.05
   72.02
   76.98
   81.95
   86.92
   91.88
   96.85
  101.82
  106.78
  111.75
  116.72
  121.68
  126.65
  131.62
  136.58
  141.55
  146.52
```

casename_tsXX.dat contains a (TS_OBS(I)%N_TIMES, Nlay) dimensional array, where TS_OBS(I)%N_TIMES is the total number of the measurement times at a corresponding individual measurement site. Variables in this input file are defined as

TS_OBS(I)%TIMES(J):	The *j*th measurement time (in days) at the *i*th scatter point, I=1, N_ASSIM_TS and J =1, TS(OBS(I)%N_TIMES.
TS_OBS(I)%TEMP(J,K):	The water temperature value at the jth measurement time and at the kth depth level in the vertical. K=1, Nlay.
TS_OBS(I)%SAL(J,K):	The water salinity value at the jth measurement time and at the kth depth level in the vertical. K= 1, Nlay.

Example of the "casename_ts01.dat", the input file for scatter point 1: is given below:

13.615972 2.4803 30.9080 2.4800 30.8100 2.3900 30.8500 2.3423 30.8600 ……

```
30.278473 -99.0000  31.0915 -99.0000  31.1000 -99.0000  31.1000 -99.0000  31.1005 ……...
42.107639   1.0492  31.2500   1.0700  31.2500   1.1242  31.2700   1.1983  31.2900 ……
47.431248   0.7500  31.2300   0.7480  31.2320   0.7433  31.2367   0.7547  31.2400 ……..
90.829865   1.8700  31.1400   1.8800  31.1350   1.8800  31.1400   1.8700  31.1400 ……
99.075691   1.1665  31.1700   1.1300  31.1700   1.1325  31.1675   1.1100  31.1700 ……
108.215973  0.6900  30.8100   0.6940  30.8200   0.6700  30.8200   0.6800  30.8200 ……
125.329163  3.8406  31.7400   2.3295  31.5985   1.2458  31.4683   0.3152  31.5100 ……
139.830551  8.1717  30.6623   6.0275  30.7045   3.7392  31.1700   2.0327  31.5200 ……..
157.011108 11.6150  30.7800  11.4900  30.4350  11.0300  30.4550   4.8300  30.5950 ……
161.602783 11.1271  31.0394   7.5180  30.7920   2.1850  31.0400   1.8923  31.0077……
Additional data not included for document
```

This example files contains one point nudging assimilation for the entire year starting on 1 January and ending at 31 December. For times with no data at a particular depth, we fill it by using a value of "-99.0000". This data will not be used for assimilation. In this case, NLay is 30. This indicates that there should be 61 columns in each row, 1 for the measurement time and 60 for temperature and salinity. We trimmed a portion of temperature and salinity in each row to fit the size of this page.

The grid point method. In some cases, to avoid excessive smoothing due to the interpolation process, users might want to use the pre-prepared fields of T and S for nudging assimilation. FVCOM has an option to include a 3-D field of T and S at grid points. This assimilation approach is very similar to the one used for the scatter point method, except for the input file format. The T and S data used for this 3-D field are prepared in a file named *casename_ts3d.dat* and read in the subroutine shown below:

```
   SUBROUTINE SET_TS3D_ASSIM_DATA
   USE ALL_VARS

!--------------------------------------------------------------------------
--------!
!  SET UP ASSIMILATION DATA FOR TEMP/SAL OBSERVATIONS
|
!--------------------------------------------------------------------------
--------!
!
   if(iint <= 1) then
   ONAME = "./"//TRIM(INPDIR)//"/"//trim(casename)//'_ts3d.dat'
   OPEN(INATS3D,FILE=ONAME,STATUS='old')  ; REWIND(1)
   endif

   READ(INATS3D,*,IOSTAT=IOS) ASTIME_TS3D
   IF(IOS< 0) THEN
      ASTIME_TS3D=1.0E30_SP
      RETRUN
   ENDIF
   READ(INATS3D,*) KB_TS_ASSIM
```

```
   ALLOCATE(TS3D_OBS(KB_TS_ASSIM))
   DO K=1,KB_TS_ASSIM
      READ(INATS3D,*) KTMP,TS3D_OBS(K)%NNODE_TS_ASSIM
      DO I=1,TS3D_OBS(K)%NNODE_TS_ASSIM
         ALLOCATE(TS3D_OBS(K)%NODE_NNTS(TS3D_OBS(K)%NNODE_TS_ASSIM))
         ALLOCATE(TS3D_OBS(K)%T1_TS_OBS(TS3D_OBS(K)%NNODE_TS_ASSIM))
         ALLOCATE(TS3D_OBS(K)%S1_TS_OBS(TS3D_OBS(K)%NNODE_TS_ASSIM))
         READ(100,*) TS3D_OBS(K)%NODE_NNTS(I),&
                TS3D_OBS(K)%T1_TS_OBS(I),&
               TS3D_OBS(K)%S1_TS_OBS(I)
      ENDDO
   ENDDO

   RETURN
   END SET_TS3D_ASSIM_DATA
```

The variables in *casename_ts3d.dat* are defined as

ASTIME_TS3D:	The time at which the assimilation starts. This is the time relative to the start time of the model run (in hours).
KB_TS_ASSIM:	The total sigma levels at which T and S data are available for the data assimilation.
KTMP	The sequence number of the sigma level, starting at 1 and end KB_TS_ASSIM.
TS3D_OBS(K)%NNODE_TS_ASSIM	The node number at the *K*th sigma level included for the assimilation data.
TS3D_OBS(K)%NODE_NNTS(I)	Node ID number.
TS3D_OBS(K)%T1_TS_OBS(I)	Temperature at the *i*th node number and *K*th sigma level.
TS3D_OBS(K)%S1_TS_OBS(I)	Salinity at the *i*th node number and *K*th sigma level.

Example of the input file *casename_3dts.dat* is given below:

```
1255.21667  !model hour from 1999-04-01:00 (It is a relative time)
    30        !number of sigma levels
     1   329  !Sequence number of the sigma level and total node number
  6780    7.49080   32.40919       Node number, T and S
  7053    7.85366   32.47128
.....................................
 12098    7.06421   32.30460
     2   329  !No of sigma and sum of node
  6780    7.49535   32.40990
  7053    7.85481   32.47145
.....................................
```

```
 12098    7.05733   32.30354
.....................................
.....................................
    30   329  !No of sigma and sum of node
  6780    7.45696   32.40549
  7053    7.94684   32.48793
.....................................
 12098    6.55552   32.34052
  1267.40002  !model hour from 1999-04-01:00
    30        !number of sigma level
     1   493  !No of sigma and sum of node
  6504    7.92841   32.47926
  6505    7.92186   32.47792
.....................................
 12824    7.02454   32.30564
.....................................
.....................................
    30   493  !No of sigma and sum of node
  6504    8.09501   32.52108
  6505    8.08489   32.51897
.....................................
 12824    6.59087   32.29760
.....................................
```

b) Current Assimilation

The current nudging assimilation is carried out using a similar approach as that discussed for the 3-D scatter point assimilation method. Two types of input files are required: 1) *casename_current.xy*, which contains number, locations, local water depths, total measurement levels in the vertical, angle of the local isobath relative to the x-coordinate, and the depths of the current measurements, and 2) files for each current measurement station containing the actual data (name *casename_currXX.dat*). The key means to avoid an unrealistic localized response of the model-computed current to the current nudging at a scatter point is to make sure that an appropriate affecting radius is selected based on the coherence scale of currents in the nudging region. In most applications in the coastal ocean, the coherence scale of the current is greater along the local isobath than across the local isobath. For this reasons, we have introduced an asymmetric influence radius that is heavily weighted in the along-isobath direction. If the value of the radius is appropriately selected based on the coherence scale, the nudging method should function like an optimal interpolation method.

The current data used for nudging assimilation is read by the subroutine called "SET_CUR_ASSIM_DATA". A portion of this subroutine is listed below to provide users with a direct view of the format of the input files.

```
!---------------------------------------------------------------------!
!  Read Number of Current Observations and Coordinates of Each
!
!---------------------------------------------------------------------!

   FNAME = "./"//TRIM(INPDIR)//"/"//trim(casename)//"_current.xy"
!
   OPEN(1,FILE=TRIM(FNAME),STATUS='OLD')
   READ(1,*) N_ASSIM_CUR
   ALLOCATE(CUR_OBS(N_ASSIM_CUR))

!
!--Read X,Y Coordinates of Measurement Stations-----------------------
!
   DO I=1,N_ASSIM_CUR
   READ(1,*)ITMP,CUR_OBS(I)%X,CUR_OBS(I)%Y,CUR_OBS(I)%DEPTH,NLAY,
  *           CUR_OBS(I)%SITA
   CUR_OBS(I)%N_LAYERS = NLAY
   ALLOCATE(CUR_OBS(I)%ODEPTH(NLAY))
   DO J=1,NLAY
    READ(1,*)CUR_OBS(I)%ODEPTH(J)
    IF(CUR_OBS(I)%ODEPTH(J) > CUR_OBS(I)%DEPTH)THEN
    IF(MSR)WRITE(IPT,*)'OBSERVATION DEPTH',J,'OF CURRENT MOORING',I
         IF(MSR)WRITE(IPT,*)'EXCEEDS BATHYMETRIC DEPTH'
         IF(MSR)WRITE(IPT,*)'HALTING..........'
         CALL PSTOP
       END IF
     END DO
   END DO

!
   CLOSE(1)

!---------------------------------------------------------------------!
!  Open Current Observational Files for Each Observational Point and  !
!  Read Data                                                          !
!---------------------------------------------------------------------!

  DO I=1,N_ASSIM_CUR
!----Make Sure Current Observation File Exists----------------------!
  WRITE(NAC,'(I2.2)')I
  ONAME = "./"//TRIM(INPDIR)//"/"//trim(casename)//'_curr'//NAC//'.dat'
     INQUIRE(FILE=TRIM(ONAME),EXIST=FEXIST)
     IF(MSR .AND. .NOT.FEXIST)THEN
       WRITE(IPT,*)'CURRENT OBSERVATION FILE: ',ONAME,' DOES NOT EXIST'
       WRITE(IPT,*)'HALTING.....'
       CALL PSTOP
     END IF

     OPEN(1,FILE=ONAME,STATUS='old')  ; REWIND(1)
     NCNT = 0

!----Count Number of Data Entries for Observation I-----------------!
     DO WHILE(.TRUE.)
       READ(1,*,IOSTAT=IOS)
       IF(IOS < 0)EXIT
```

```
        NCNT = NCNT + 1
      END DO
      CUR_OBS(I)%N_TIMES = NCNT
      REWIND(1)
      IF(NCNT == 0)THEN
        IF(MSR)WRITE(IPT,*)'NO DATA FOR CURRENT OBSERVATION',I
        CALL PSTOP
      END IF

!----Allocate Arrays to Hold Current (UA,VA) and Time (TIME)----------!
    ALLOCATE(CUR_OBS(I)%TIMES(CUR_OBS(I)%N_TIMES))
    ALLOCATE(CUR_OBS(I)%UO( CUR_OBS(I)%N_TIMES , CUR_OBS(I)%N_LAYERS ))
    ALLOCATE(CUR_OBS(I)%VO( CUR_OBS(I)%N_TIMES , CUR_OBS(I)%N_LAYERS ))

!----Read in Current Data for Observation I--------------------------!
     NLAY = CUR_OBS(I)%N_LAYERS
     DO J=1,CUR_OBS(I)%N_TIMES

READ(1,*)CUR_OBS(I)%TIMES(J),(CUR_OBS(I)%UO(J,K),CUR_OBS(I)%VO(J,K),K=1
,NLAY)
     END DO
     CLOSE(1)
```

Parameters and variables in *casename_current.xy* are defined as

N_ASSIM_C	Total number of the current measurement stations
ITMP	The sequence number of the current stations starting from 1 to N_ASSIM_C
CUR_OBS(I)%X	The *x* coordinate of the current station with the sequence number of ITMP
CUR_OBS(I)%Y	The *y* coordinate of the current station with the sequence number of ITMP
CUR_OBS(I)%DEPTH	The local water depth where the current station (number: ITMP) is located
NLAY	Total current measurement levels in the vertical for the current station listed in the first column
CUR_OBS(I)%SITA	The angle of the local isobath (where the current station is located) relative to the x-coordinate;
CUR_OBS(I)%ODEPTH(J)	The measurement depths of the current station numbered "ITMP". J =1, NLAY.

Variables in *casename_currXX. dat* are defined as

CUR_OBS(I)_N_Times	Total number of measurement times for the *i*th current station, where I = 1, N_ASSIM_CUR

CUR_OBS(I)%Times(J)	The *j*th measurement time for the *i*th current station, where J = 1, CUR_OBS(I)_N_Times, and time is measured in days
CUR_OBS(I)%UO(J,K)	The *x*-component velocity (*u*) at the *j*th measurement time and *k*th level in the vertical for the *i*th current station meter, where I=1, N_ASSIM_CUR J=1, CUR_OBS(I)_N_Times K =1, NLAY
CUR_OBS(I)%VO(J,K)	The *y*-component velocity (*v*) at the *j*th measurement time and *k*th level in the vertical for the *i*th current station, where I=1, N_ASSIM_CUR J=1, CUR_OBS(I)_N_Times K =1, NLAY

An example of *casename_current.xy* :

```
7
1 1265398.619439  -44588.061306 122.0  1  -61. BBIA
   16.0
2 1265174.556436  -45045.401758 120.0  3  -61. BBI
   30.0
   50.0
  100.0
3 1279722.186281  -63838.759316 118.0  1  -13. BBOA
   14.0
4 1279757.900111  -64193.413867 120.0  3  -13. BBO
   30.0
   50.0
  100.0
5 1256814.907779  -50634.279550 216.0  1  -50. NECEA
   24.0
6 1256452.199341  -50819.826317 215.0  4  -50. NECE
   35.0
   55.0
  105.0
  155.0
7 1243386.759030  -70126.152969 213.0  4  146. NECW
   33.0
   53.0
  103.0
  153.0
```

Number of the current meters

Column 1: Current meter 1;
Column 2: The x location
Column 3: The y location
Column 4: Local water depth
Column 5: number of measurement levels in the vertical
Column 6: Angle of the tangent line of the local isobath

The measurement depths of current station 4

An example of one of the current meter (casename_cur01.dat) data file:

```
   1.000000   31.8412   76.1077
   1.041667   81.6033   39.3607
   1.083333   99.0310  -18.8016
   1.125000   88.5113  -59.9447
   1.166667   74.5097  -95.0839
   1.208333   45.8404 -108.2978
   1.250000    8.9615  -98.0915
   1.291667  -46.5618  -85.4359
   1.333333  -91.8395  -62.1636
   1.375000 -118.5701  -23.1035
   1.416667 -125.9544   26.4417
   1.458333 -122.4551   56.3542
   1.500000  -87.3755   89.2498
   1.541667  -47.5856   88.5204
   1.583333    0.0000   47.3000
   1.625000   28.7474   12.3806
   1.666667   56.2508  -25.5947
   1.708333   51.3846  -50.8617
   1.750000   29.3176  -62.5723
   1.791667   -8.5477  -67.3598
   1.833333  -34.7950  -53.5959
   1.875000  -51.2704  -24.6770
   1.916667  -55.0411   19.2190
   1.958333  -46.6193   55.2623
   2.000000  -15.1039   62.9123
   2.041667    1.6301   28.6537
  ............
Further Data Not Shown For Brevity
```

Column 1: Measurement time (days)
Column 2: u value
Column 3: v value

Note: In "casename_current.xy", the current measurement on current station 1 was made at a single depth, so that NLAY =1.

In general, the number of columns should be equal to 2NLAY+1.

Biological Module

Creating a finite-volume code for a biological model is very easy because the governing equations for individual biological quantities are tracer equations with the addition of source or sink terms related to biological transformations. Thus, implementation of the additional equations only requires minor modifications of the salinity or temperature equations. Once a biological model is in place, it can be easily extended to more complex systems. For instance, an NPZ model can easily be modified to a multi-species NPZD or NPZDB model.

The biological models can run either online or off-line. The word "online" means here that the biological model is coupled and run simultaneously with the FVCOM physical model, while the word "offline" refers to the method in which the biological model is run separately within the physical environment previously predicted or

computed by the FVCOM physical model. Because of the size limitations of older hard disks, we have used the first method in many of our previous studies. Now that data storage is less of an issue, the second method is preferred because it can significantly reduce computational time and allows the flexibility to conduct process-oriented studies using different biological parameters.

The setup of the biological model run is straightforward. In addition to physical parameters and input files, the biological model run is controlled by two input files: 1) initial conditions and 2) biological parameters. Unlike physical fields, biological measurements are usually made at scatter points that are not sufficient to build an initial field for the modeling effort. For this reason, the initial fields of the NPZ, NPZD, NPZDB and water quality models are specified by either the steady state solutions of the governing equations or 1-D (vertical) profiles derived from available observations. Unless the spatial (horizontal) distributions of the biological variables are considered as part of the initial conditions, the initial values of these variables can be directly specified or calculated in a subroutine called "Biolinit".

The biological parameters are also read in "Biolinit". The parameters are prepared in the input file named "casename_bioparam.data". The content of this input file differs in the different biological models and can be easily modified for a new biological model that users may like to develop. To provide some information about the format and content of this input file, we include three examples for the NPZ and ZPZD models here.

1. The NPZ Model:

Seven biological parameters must be specified for the 3-component NPZ model. They are: 1) γ - the fraction of ingested phytoplankton unassimilated by zooplankton; 2) V_m - the maximum phytoplankton growth rate; 3) k_s - the half-saturation constant for phytoplankton growth; 4) R_m - the maximum grazing rate of phytoplankton by zooplankton; 5) λ - the grazing efficiency of phytoplankton by zooplankton; 6) g - the zooplankton death rate; and 7) ε - the phytoplankton death rate. The physical parameter is the light attenuation coefficient called "kert". To run the model with an initial field specified by a steady state solution, the total nitrogen concentration (Ntot) needs to be specified. The model also includes an option to add or remove the nutrient flux from the bottom. The initial file given below is an example of "GB_bioparam.dat" used for a

process-oriented study of the phytoplankton dynamics on Georges Bank conducted by Franks and Chen (1996).

GB_bioparam.dat:

```
0.3   !Gamma
2.0   !Vmax
1.0   !Ks
0.5   !Rmax
0.2   !Lambda
0.2   !Gz
0.1   !Epsilon
7.0   !Ntot;   was 20.0 in first runs
0.1 1.0 !kext usually 0.08; if light is on, pflag=1
 off  !including the bottom nutrient flux. If not, use "off"
0.0 ! sinking at 1 m day-1: -1.157e-5
0.31e-5 40 100 !0.31e-5 is maximum flux (m mole/m^2 s) 40 dtop,100 dbot
```

2. The NPZD Model

This is a multi-nutrient and species lower trophic level food web model controlled by 36 biological parameters. For example, the initial file for the Gulf of Maine/Georges Bank model run used in R. Ji's Ph.D thesis (Ji, 2004) is given below.

GOM_bioparam.dat:

```
3.0 !Vpl: Maximum growth rate of P_L (1/d)
2.0 !Vps: Maximum growth rate of P_S (1/d)
1.0 !Kpl1: Half-saturation constant for NO3 uptake by P_L(umN/l)
0.2 !Kpl2: Half-saturation constant for NH4 uptake by P_L(umN/l)
0.5 !Kps1: Half-saturation constant for NO3 uptake by P_S(umN/l)
0.05 !Kps2: Half-saturation constant for NH4 uptake by P_S(umN/l)
2.0 !Ksl: Half-saturation constant for silicate uptake by P_L(umSi/l)
0.3 !Rzl: Maximum grazing rate of Z_L (on P_L:1/d)
0.5 !Rzs: Maximum grazing rate of Z_S (on Z_S:1/d)
0.5 !Rsd: Maximum grazing rate of Z_S (on D:1/d)
0.3 !Rls: Maximum grazing rate of Z_L (on Z_S:1/d)
1.0 !Szl: Ivlev const for Z_L grazing (1/concentration: l/um)
0.5 !Szs: Ivlev const for Z_S grazing on P_S (1/concentration:1/um)
1.0 !Ssd: Ivlev const for Z_S grazing on D (1/concentration:1/um)
1.0 !Sls: Ivlev const for Z_L grazing on Z_S(1/concentration: l/um)
0.35 !Gzl: Efficiency of Z_L on P_L (nondimensional)
0.4  !Gzs: Efficiency of Z_S on P_S (nondimensional)
0.2  !Gsd: Efficiency of Z_S on D (nondimensional)
0.4  !Gls: Efficiency of Z_L on Z_S(nondimensional)
0.1  !Epl: The mortality rate of P_L (1/d)
0.2  !Eps: The mortality rate of P_S (1/d)
0.2  !Ezl: The mortality rate of Z_L (1/d/umol)
0.2  !Ezs: The mortality rate of Z_S (1/d/umol)
0.06 !Edn: Decomposition rate of detrital N (1/day)
0.03 !Eds:  Decomposition rate of detrital Si(1/day)
1.5  !Rns: N:Si in P_L (mole ratio)
0.1  !Kext: light attenuation coefficient (1/m)
0.03  !Xkself: Self-shading coefficient (1/uMN m)
0.069 !Afa: Index for T(°C) dependence of all biological processes.
```

```
1.0   !Rchn: The ratio of Chrolophyll versus nitrogen(ugchl-a/umolN)
0.4   !Betal: NH4 inhibition for P_L (umol l^-1)
0.2   !Betas: NH4 inhibition for P_S (umol l^-1)
0.03  !Enit:  nitrification rate(1/day)
0.2   !wsk3: sinking velocity of P_S
1.5   !wsk4: sinking velocity of P_L
5.0   !wsk7: sinking velocity of Dn and Dsi
```

Sediment Module

The setup of the sediment module is described in Chapter 13.

Ice Module

This module was developed and tested by F. Dupont (give affiliation). For detailed information about the setup of this module, please contact Dupont directly at (give email address). We will re-run Dupont's validation experiments at SMAST to gain experience with this module and ensure that it will run efficiently with the present version of FVCOM. After that, we will work together with Dupont to add an example of ice model setup into the user manual.

Chapter 16: FVCOM Test Cases

To help users learn how to use FVCOM, we have included two simple examples here. Case 1: Tidally-driven flooding/drying process in a semi-enclosed channel. Case 2: Freshwater discharge on an idealized continental shelf.

Case 1: Tidally-driven flooding/drying process in a semi-enclosed channel

This case was selected from the Chen et al. (2006c) recent manuscript entitled "A 3-Dimensional, Unstructured Grid, Finite-Volume Wet/Dry Point Treatment Method for FVCOM". The water is homogeneous in this case, with no freshwater riverine or groundwater input and no surface atmospheric forcing.

a) Design of the numerical experiment

Numerical experiments were conducted for an idealized semi-enclosed channel with a width of 3 km at the bottom, a length of 30 km, a constant depth of 10 m and a lateral slope of about 0.033 (Figure 16.1). This channel is oriented east to west, with connection to a relatively wide and flat-bottom shelf to the east and inter-tidal zones on the northern and southern channel edges. The inter-tidal zone is distributed symmetrically to the channel axis with a constant slope of α and a width of 2 km.

The computational domain was configured with unstructured triangular grids in the horizontal and σ-levels in the vertical. Numerical experiments were conducted for cases with different horizontal and vertical resolutions. The comparison between these cases was made based on differences from a standard run with a horizontal resolution of about 500 m in the channel, 600 to 1000 m on the shelf, and 600 to 900 m over the inter-tidal zone (Figure 16.1).

The model was forced by an M_2 tidal oscillation with amplitude ζ_o at the open boundary of the outer shelf. This oscillation creates a surface gravity wave which propagates into the channel and reflects back after it reaches the solid wall at the upstream end. For a given tidal elevation at this open boundary, the velocity in triangular elements connected to the boundary and water transport flowing out of the computational domain is determined through the incompressible continuity equation.

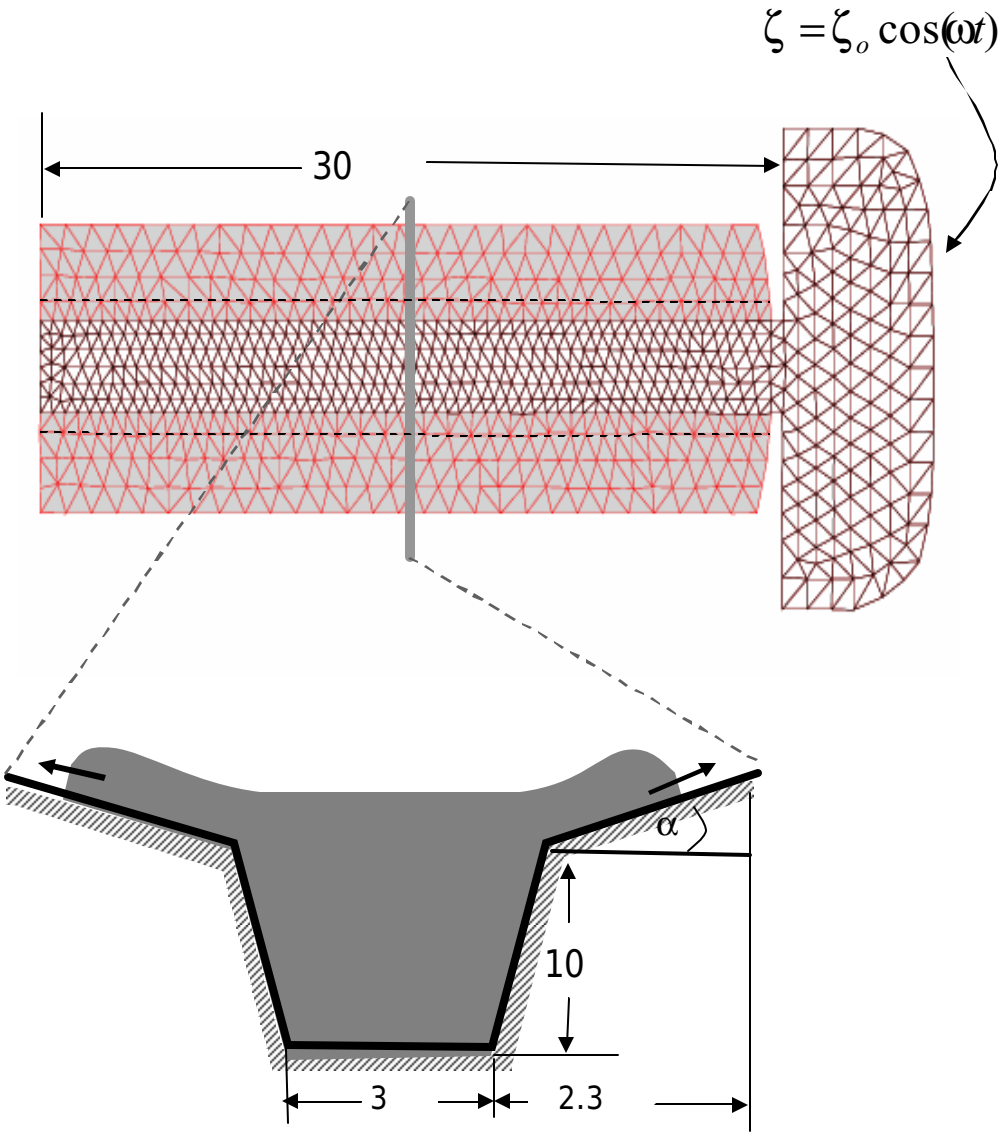


Figure 16.1: Unstructured triangular grid for the standard run plus a view of the cross-channel section. Dashed line along the channel indicates the edge of the channel connected to the inter-tidal zone.

b) The input files

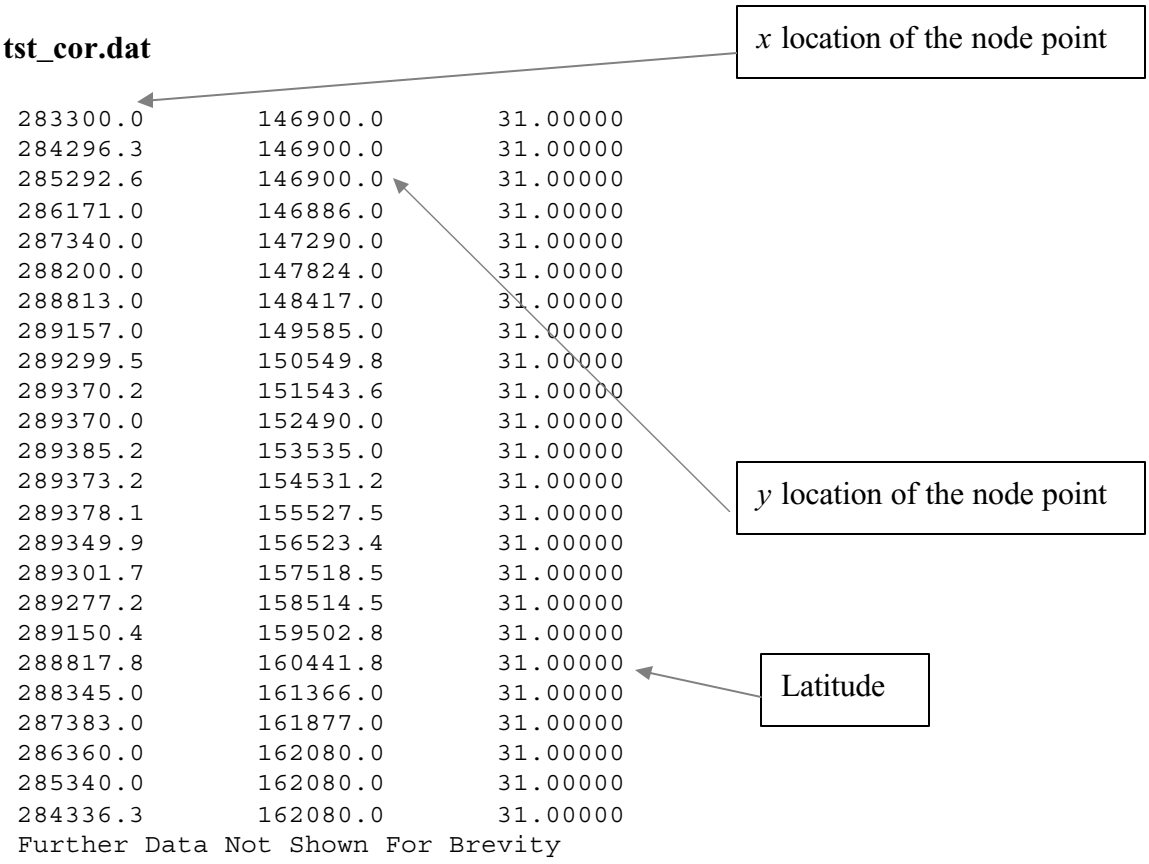
1) tst_bfw.dat

Comments

```
Number of bottom fresh water input points
0
```

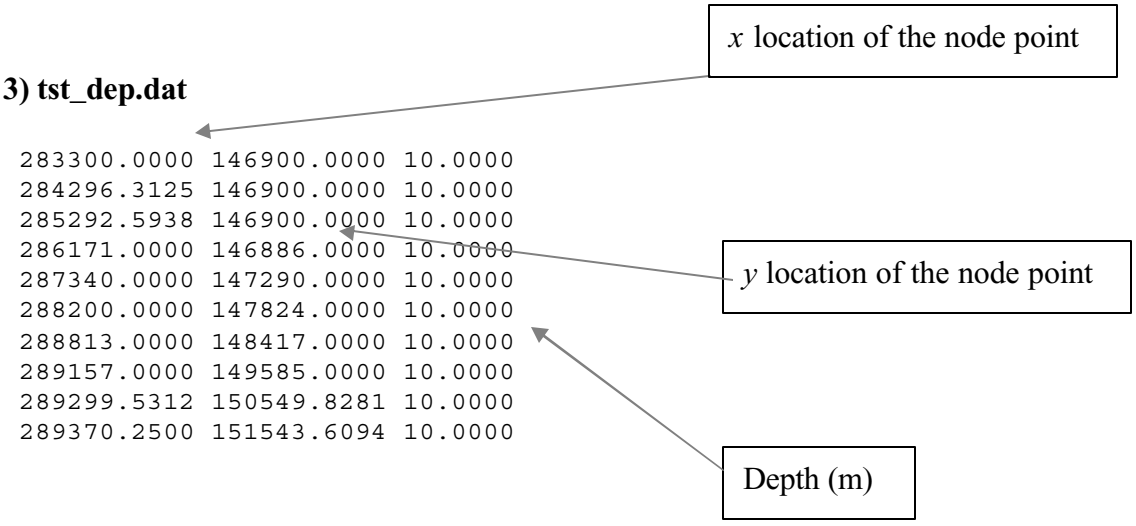
No groundwater flux

2) tst_cor.dat



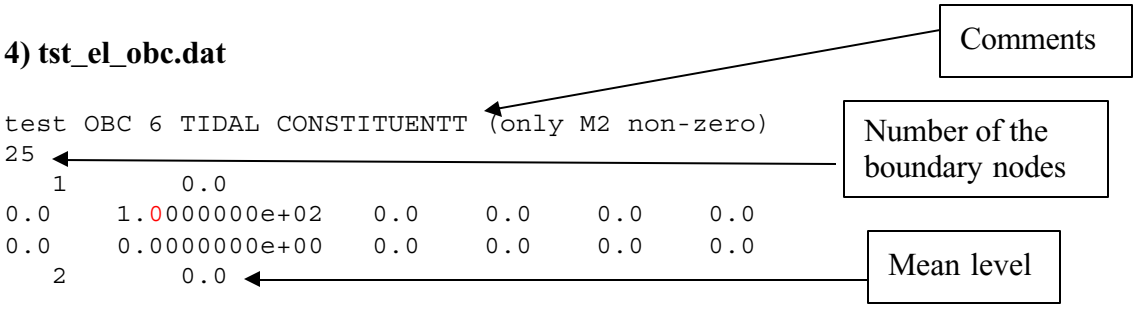

```
283300.0        146900.0        31.00000
284296.3        146900.0        31.00000
285292.6        146900.0        31.00000
286171.0        146886.0        31.00000
287340.0        147290.0        31.00000
288200.0        147824.0        31.00000
288813.0        148417.0        31.00000
289157.0        149585.0        31.00000
289299.5        150549.8        31.00000
289370.2        151543.6        31.00000
289370.0        152490.0        31.00000
289385.2        153535.0        31.00000
289373.2        154531.2        31.00000
289378.1        155527.5        31.00000
289349.9        156523.4        31.00000
289301.7        157518.5        31.00000
289277.2        158514.5        31.00000
289150.4        159502.8        31.00000
288817.8        160441.8        31.00000
288345.0        161366.0        31.00000
287383.0        161877.0        31.00000
286360.0        162080.0        31.00000
285340.0        162080.0        31.00000
284336.3        162080.0        31.00000
Further Data Not Shown For Brevity
```

3) tst_dep.dat



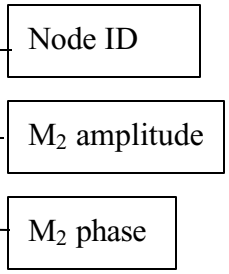

```
283300.0000 146900.0000 10.0000
284296.3125 146900.0000 10.0000
285292.5938 146900.0000 10.0000
286171.0000 146886.0000 10.0000
287340.0000 147290.0000 10.0000
288200.0000 147824.0000 10.0000
288813.0000 148417.0000 10.0000
289157.0000 149585.0000 10.0000
289299.5312 150549.8281 10.0000
289370.2500 151543.6094 10.0000
```

4) tst_el_obc.dat



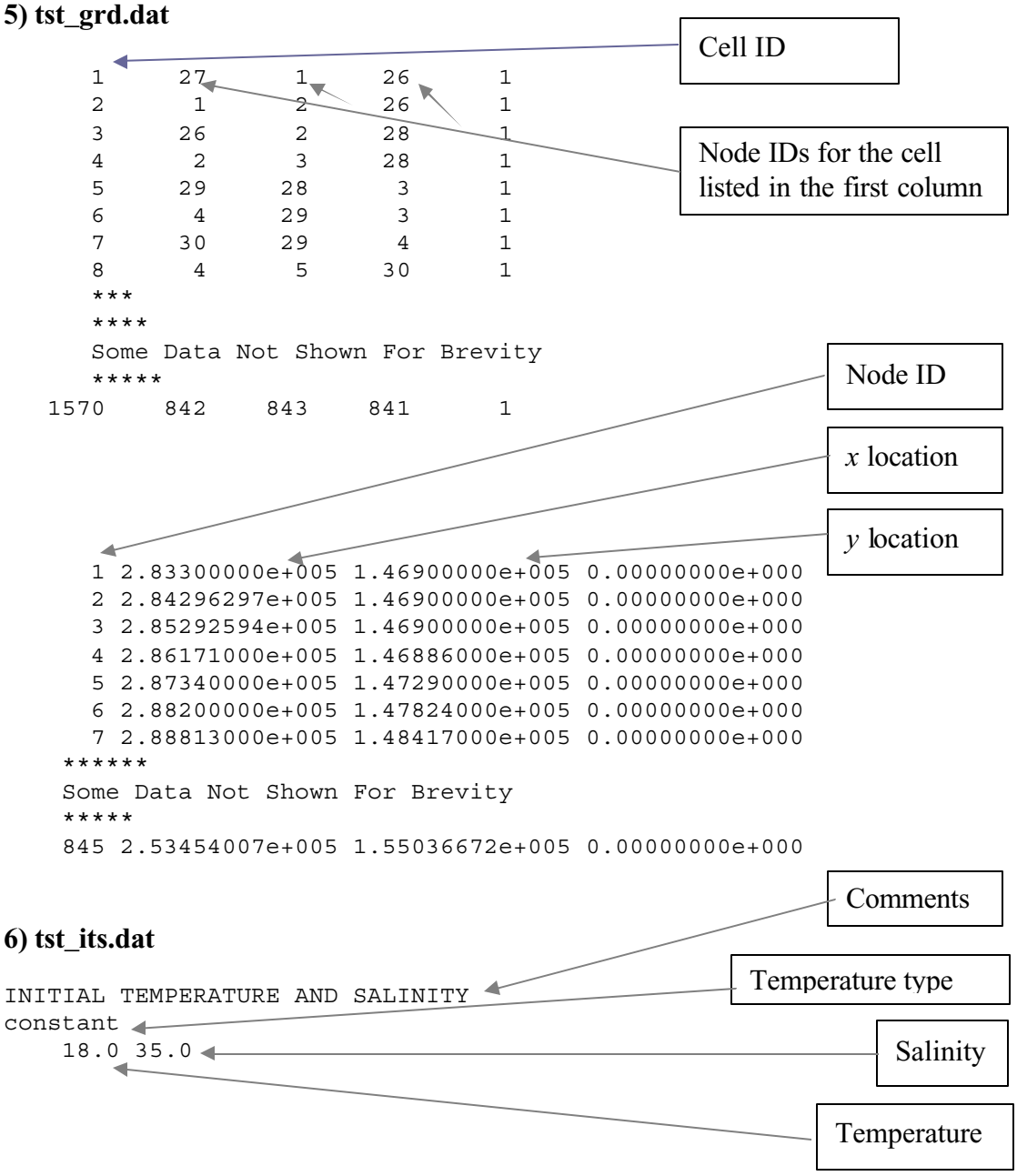

```
test OBC 6 TIDAL CONSTITUENTT (only M2 non-zero)
25
   1        0.0
0.0     1.0000000e+02    0.0     0.0     0.0     0.0
0.0     0.0000000e+00    0.0     0.0     0.0     0.0
   2        0.0
```

```
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
   3       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
   4       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
   5       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
   6       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
   7       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
   8       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
   9       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
  10       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
  11       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
  12       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
  13       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
  14       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
  15       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
  16       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
  17       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
  18       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
  19       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
  20       0.0
0.0    1.0000000e+02   0.0     0.0     0.0     0.0
0.0    0.0000000e+00   0.0     0.0     0.0     0.0
  21       0.0
```



```
0.0    1.0000000e+02   0.0    0.0    0.0    0.0
0.0    0.0000000e+00   0.0    0.0    0.0    0.0
  22       0.0
0.0    1.0000000e+02   0.0    0.0    0.0    0.0
0.0    0.0000000e+00   0.0    0.0    0.0    0.0
  23       0.0
0.0    1.0000000e+02   0.0    0.0    0.0    0.0
0.0    0.0000000e+00   0.0    0.0    0.0    0.0
  24       0.0
0.0    1.0000000e+02   0.0    0.0    0.0    0.0
0.0    0.0000000e+00   0.0    0.0    0.0    0.0
  25       0.0
0.0    1.0000000e+02   0.0    0.0    0.0    0.0
0.0    0.0000000e+00   0.0    0.0    0.0    0.0
```

5) tst_grd.dat

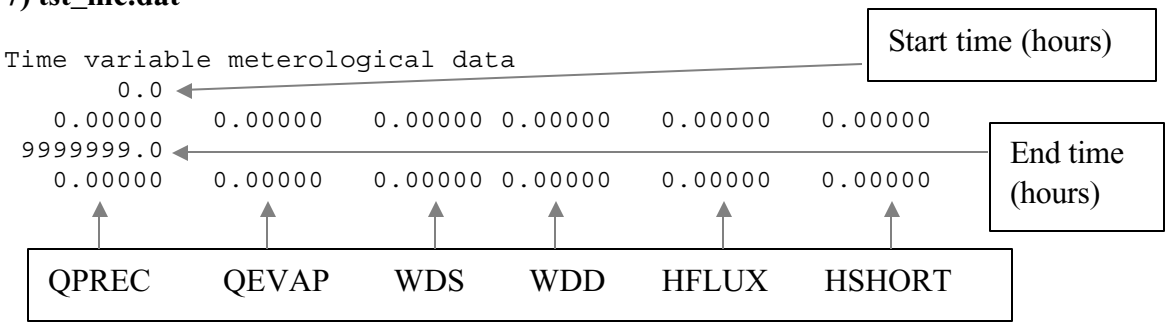



```
      1      27       1      26       1
      2       1       2      26       1
      3      26       2      28       1
      4       2       3      28       1
      5      29      28       3       1
      6       4      29       3       1
      7      30      29       4       1
      8       4       5      30       1
      ***
      ****
      Some Data Not Shown For Brevity
      *****
   1570    842    843    841       1

      1 2.83300000e+005 1.46900000e+005 0.00000000e+000
      2 2.84296297e+005 1.46900000e+005 0.00000000e+000
      3 2.85292594e+005 1.46900000e+005 0.00000000e+000
      4 2.86171000e+005 1.46886000e+005 0.00000000e+000
      5 2.87340000e+005 1.47290000e+005 0.00000000e+000
      6 2.88200000e+005 1.47824000e+005 0.00000000e+000
      7 2.88813000e+005 1.48417000e+005 0.00000000e+000
    ******
    Some Data Not Shown For Brevity
    *****
    845 2.53454007e+005 1.55036672e+005 0.00000000e+000
```

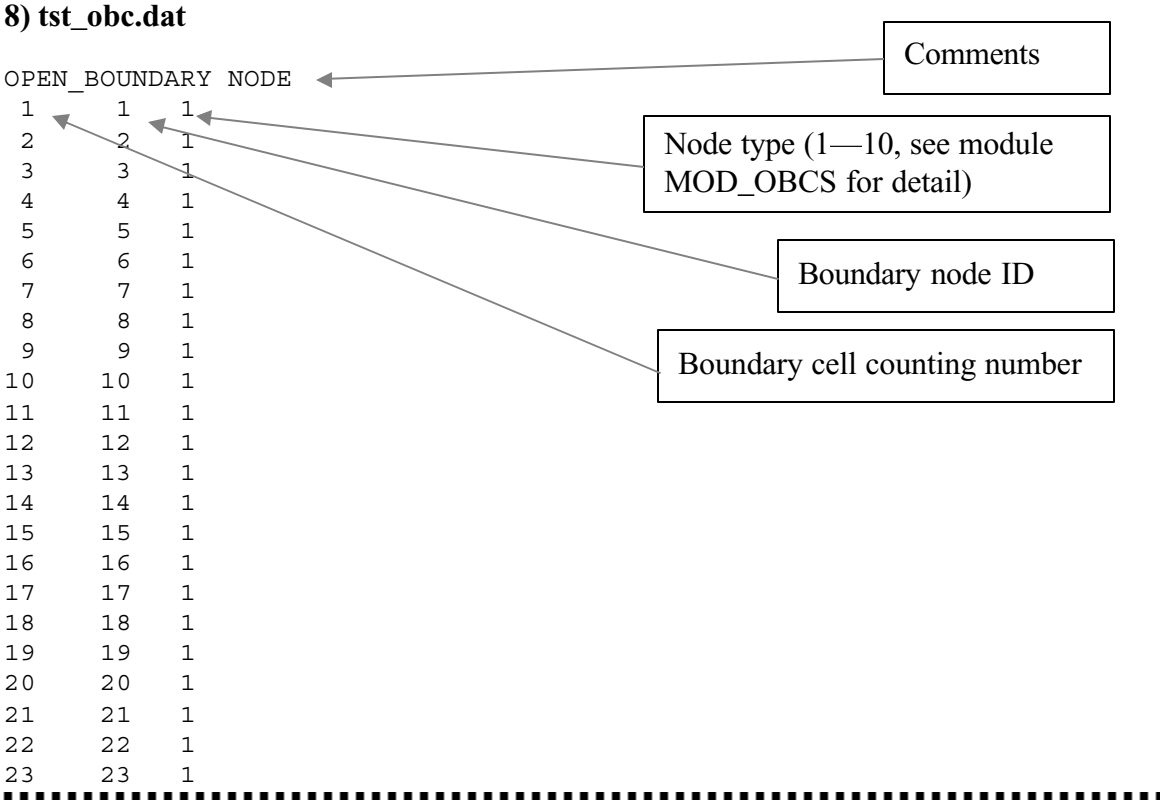
6) tst_its.dat


```
INITIAL TEMPERATURE AND SALINITY
constant
    18.0 35.0
```

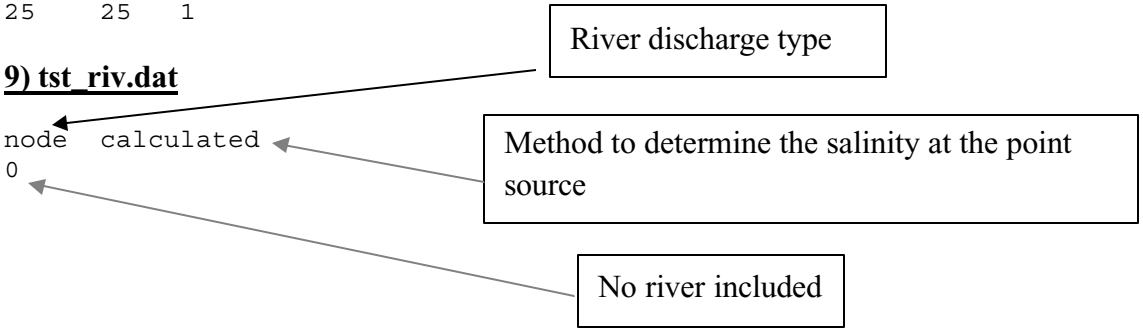
7) tst_mc.dat



8) tst_obc.dat



9) tst_riv.dat



10) tst_spg.dat

0 ← No sponge layer is included

11) tst_run.dat

```
!========================================================================!
!   INPUT FILE FOR PARAMETERS CONTROLLING EXECUTION OF FVCOM             !
!   DESCRIPTION OF VARIABLES AND SUGGESTED PARAMETERS CAN BE FOUND AT BOTTOM   !
!                                                                        !
!        FORMAT:                                                  !
!       1.) VARIABLE  = VALUE  (EQUAL SIGN MUST BE USED)                 !
!       2.) FLOATING POINT VARIABLES MUST CONTAIN A PERIOD "." EX: 1.3, 2.,etc !
!       3.) BLANK LINES ARE IGNORED AS ARE LINES BEGINNING WITH ! (F90 COMMENT)!
!       4.) COMMENTS CAN FOLLOW VALUES IF MARKED BY !                    !
!       5.) ORDER OF VARIABLES IS NOT IMPORTANT                          !
!       6.) FOR MULTIPLE VALUE VARIABLES FIRST ENTRY IS NUMBER OF VARIABLES    !
!           TO FOLLOW (OR 0 IF NONE)                                     !
!       7.) DO NOT USE COMMAS TO SEPARATE VARIABLES                      !
!       8.) DO NOT EXCEED EIGHTY CHARACTERS PER LINE                     !
!       9.) FOR LINE CONTINUATION ADD \\ TO END OF LINE TO FORCE CONTINUE      !
!           TO NEXT LINE.  MAXIMUM 4 CONTINUATIONS                       !
!       10.) TRUE = T, FALSE = F                                         !
!                                                                        !
!  THE PREVIOUS FORMAT OF "VARIABLE: VALUE" IS NO LONGER VALID           !
!  THE MORE ATTRACTIVE " = " FORMAT WAS SUGGESTED BY Hernan G. Arango    !
!    AND SUBSEQUENTLY ADOPTED                                            !
!============================================================!


!============ Case Title=========================================

CASETITLE = WET/DRY TEST

!=========Parameters Controlling Time Integration==================
DTE    = 6.21
ISPLIT = 10
IRAMP  = 720
NSTEPS = 7200

!=========Parameters Controlling Input/Output=====================

INPDIR = ./IDEAL_ESTUARY
OUTDIR = ./OUTDIR_TST
INFOFILE = screen
IREPORT = 50
IRECORD = 12
IDMPSMS = 0
IRESTART = 720
RESTART = cold_start


!=========Parameters Controlling Output of Average Data Fields=====

AVGE_ON  = F
INT_AVGE = 5
BEG_AVGE = 1
NUM_AVGE = 10
```

```
CDF_OUT_AVE = F
CDF_VDP_AVE = u v el

!=========Parameters Controlling Bottom Friction=================

BFRIC = .0025
Z0B    = .001
BROUGH_TYPE = orig

!=========Parameters Controlling Diffusivity Coefficients=========

HORZMIX = closure
HORCON  = 2.000E-1
HPRNU   = 1.000E+0
VERTMIX = closure
UMOL    = 1.000E-4
VPRNU   = 1.000E+0

!=========Parameters Controlling Model Equations==================

BAROTROPIC  = F
SALINITY_ON = T
TEMP_ON     = T

!=========Parameters Controlling Density Calculation=============

C_BAROPG  = sigma
CTRL_DEN  = sigma-t
VERT_STAB = T

!=========Parameters Controlling Atmospheric Forcing============

H_TYPE = flux_h
M_TYPE = uniform
WINDTYPE = stress
ZETA1 = 1.4
ZETA2 = 6.3
RHEAT = .78
THOUR_HS = 124.
!=========Parameters Controlling Temp/Salinity Adjustments===========

TS_FCT = T

!=========Parameters Controlling Depth Adjustments===============

DJUST = 0.0
MIN_DEPTH = 0.05

!=========Parameters Controlling Tidal Forcing====================

S_TYPE = non-julian
DELTT   = 720.

!=========Standard Depth Levels===================================

KSL    =   7
DPTHSL =   0.   -2.   -4.    -6.   -8.   -9.   -10.
```

```
!============ Parameters Controlling Sigma Distribution==========

P_SIGMA = 1.0
KB      = 6

!============ Parameters Controlling Lagrangian Tracking==========

LAG_ON = F

!============ Parameters Controlling Time Series Output===========

PROBE_ON = F

!============ Parameters Controlling Water Quality Module==========

WQM_ON = F
NB     = 8
BENWQM_KEY = F

!============ Parameters Controlling Wetting/Drying================

WET_DRY_ON = T

!============ Parmaeters Controlling SST Data Assimilation==========

SST_ASSIM = F
RAD_SST = 10000.
GAMA_SST = 1.0
GALPHA_SST = 3.e-3
ASTIME_WINDOW_SST = .50
IAV_DAY = 5

!====CURRENT DATA ASSIMILATION VARIABLES==========================

CURRENT_ASSIM = F
RAD_CUR = 20000.
GAMA_CUR = 1.0
GALPHA_CUR = 8.3e-3
ASTIME_WINDOW_CUR = .50

!====TEMPERATURE/SALINITY DATA ASSIMILATION VARIABLES=============

TS_ASSIM = F
RAD_TS = 30000.
GAMA_TS = 1.0
GALPHA_TS = 8.3e-3
ASTIME_WINDOW_TS = 72.

!====VARIABLES CONTROLLING NETCDF OUTPUT======================
CDF_OUT = F
CDF_INT = 90
CDF_STK = 0
CDF_VDP = u v s1 t1
INFO1   = 1995 Fine Grid Nov Output
INFO2   = More Information Here
```

```
!==== Parameter Controlling Richardson # dep. Dissipation
correction====
SURFACEWAVE_MIX = F

!==== Parameter Controlling Open Boundary Temp/Salt Nudging============
TS_NUDGING_OBC = F
ALPHA_OBC = .5

!==== Parameter controlling Open Boundary Temp/Salt Series
Nudging===========
TSOBC_ON = T
ALPHA_SERIES_OBC = .0014

!=====VARIABLES CONTROLLING 2D MOMENTUM BALANCE CHECKING OUTOUT=======
OUT_BALANCE = F
NUM_BALANCE = 4             !!sum of cell for 2d momentum balance output
NO_CELL  =  11446  11212  11213  11447

!=====PARAMETER CONTROLLING THE TYPE OF Temp/Salt OBC=======
TYPE_TSOBC = 3

!==== Parameter controlling Tidal Open Boundary Output===========
TIDE_INITIAL  = 14400
TIDE_INTERVAL = 6
```

12) makefile

```
#----------BEGIN MAKEFILE-----------------------------------------------
            SHELL         = /bin/sh
            DEF_FLAGS     = -P -C -traditional
            EXEC          = fvcom
#============================================================================
#  BEGIN USER DEFINITION SECTION
#============================================================================
#        SELECT MODEL OPTIONS
#          SELECT FROM THE FOLLOWING OPTIONS BEFORE COMPILING CODE
#          SELECT/UNSELECT BY COMMENTING/UNCOMMENTING LINE (#)
#          CODE MUST BE CLEANED (with "make clean") AND
#          RECOMPILED IF NEW SET OF OPTIONS IS DESIRED
#----------------------------------------------------------------------------

#----------------------------------------------------------------------------
#        PRECISION          DEFAULT PRECISION: SINGLE
#                           UNCOMMENT TO SELECT DOUBLE PRECISION
#----------------------------------------------------------------------------

#            FLAG_1 =  -DDOUBLE_PRECISION

#----------------------------------------------------------------------------
#        SPHERICAL          SELECT SPHERICAL COORDINATES FOR INTEGRATION
#                           DEFAULT: CARTESIAN
#                           UNCOMMENT TO SELECT SPHERICAL COORDINATES
#----------------------------------------------------------------------------

#             FLAG_2 = -DSPHERICAL

#----------------------------------------------------------------------------
#        FLOODYING/DRYING   INCLUDE WET/DRY TREATMENT OF DOMAIN
```

```
#                           CAN BE ACTIVATED/DEACTIVATED AT RUN TIME WITH
#                           INPUT FILE CONTROL.  (SEE exa_run.dat) FILE
#                           DEFAULT: NO FLOODYING/DRYING INCLUDED
#                           UNCOMMENT TO INCLUDE FLOODYING/DRYING
#---------------------------------------------------------------------------

             FLAG_3 = -DWET_DRY

#---------------------------------------------------------------------------
#        MULTI_PROCESSOR    INCLUDES PARALLELIZATION WITH MPI
#                           REQUIRES LINKING MPI LIBRARIES OR COMPILING
#                           WITH A PRELINKED SCRIPT (mpif90/mpf90/etc)
#                           DEFAULT: NO PARALLEL CAPABILITY
#                           UNCOMMENT TO INCLUDE MPI PARALLEL CAPABILITY
#---------------------------------------------------------------------------

              FLAG_4 = -DMULTIPROCESSOR
              PARLIB = -L../METIS_source -lmetis

#---------------------------------------------------------------------------
#        WATER_QUALITY      INCLUDE EPA WATER QUALITY MOD
#                           CAN BE ACTIVATED/DEACTIVATED AT RUN TIME WITH
#                           VARIABLE WQM_ON IN INPUT FILE
#                           DEFAULT: NO WATER QUALITY MODEL
#                           UNCOMMENT TO INCLUDE WATER QUALITY MODEL
#---------------------------------------------------------------------------

#             FLAG_5 = -DWATER_QUALITY

#---------------------------------------------------------------------------
#        NETCDF OUTPUT      DUMP OUTPUT INTO NETCDF FILES (yes/no)
#                           REQUIRES SYSTEM DEPENDENT NETCDF LIBRARIES
#                           COMPILED WITH SAME F90 COMPILER
#                           SET PATH TO LIBRARIES WITH IOLIBS
#                           SET PATH TO INCLUDE FILES (netcdf.mod) WITH IOINCS
#                           DEFAULT: NO NETCDF OUTPUT
#                           UNCOMMENT TO INCLUDE NETCDF OUTPUT CAPABILITY
#---------------------------------------------------------------------------
#           FLAG_6       =  -DNETCDF_IO
#         IOLIBS       =  -L/usr/medm/lib  -lnetcdf
#           IOINCS       =  -I/usr/medm/include
#---------------------------------------------------------------------------
#        DATA_ASSIMILATION  INCLUDE NUDGING BASED DATA ASSIMILATION FOR
#                           CURRENT/TEMP/SALINITY/SST
#                           CAN BE ACTIVATED/DEACTIVATED AT RUN TIME WITH
#                           INPUT FILE CONTROL.  (SEE exa_run.dat) FILE
#                           DEFAULT: NO DATA ASSIMILATION INCLUDED
#                           UNCOMMENT TO INCLUDE DATA ASSIMILATION
#---------------------------------------------------------------------------

#             FLAG_7 = -DDATA_ASSIM

#---------------------------------------------------------------------------
#       REMOVE SYSTEM CALLS IF RUNNING IN A DOS/WINDOWS ENVIRONMENT
#                           OR IF COMPILER FAILS WHEN COMPILING SYSTEM
#                           CALLS, TURN ON THIS OPTION.
#                           THIS WILL DEACTIVATE CERTAIN COMMANDS WHICH
#                           SETUP THE OUTPUT DIRECTORY STRUCTURE
#                           NOTE THAT THE USER WILL HAVE TO CREATE THE
#                           OUTPUT DIRECTORIES MANUALLY
#                           SEE THE README FILE INCLUDED WITH THE
#                           DISTRIBUTION
#                           DEFAULT: INCLUDE SYSTEM CALLS
```

```
#                           UNCOMMENT TO SUPRESS SYSTEM CALLS
#---------------------------------------------------------------------------

#             FLAG_8 += -DDOS

#---------------------------------------------------------------------------
#       SOLID BOUNDARY     IF GCN, NO GHOST CELL
#                          IF GCY1, GHOST CELL IS SYMMETRIC RELATIVE TO
BOUNDARY
#                          CELL EDGE
#                          IF GCY2, GHOST CELL IS SYMMETRIC RELATIVE TO MIDDLE
#                          POINT OF THE BOUNDARY CELL EDGE
#   !!!!!! ONLY ONE OF THE FLAGS BELOW CAN BE CHOSEN
#---------------------------------------------------------------------------

             FLAG_9  = -DGCN
#             FLAG_10  = -DGCY1
#             FLAG_11 = -DGCY2

#---------------------------------------------------------------------------
#        TURBULENCE MODEL   USE GOTM TURBULENCE MODEL INSTEAD OF THE ORIGINAL
#                           FVCOM MELLOR-YAMADA 2.5 IMPLEMENTATION
#                           UNCOMMENT TO USE GOTM TURBULENCE MODEL
#---------------------------------------------------------------------------

#            FLAG_12 = -DGOTM
#            GOTMLIB       = -L../GOTM_source -lturbulence -lutil
#            GOTMINCS      = -I../GOTM_source

#---------------------------------------------------------------------------
#        pV3 Realtime Visualization Server (MEDM GROUP ONLY)
#---------------------------------------------------------------------------

#            FLAG_13 = -DPV3
#            PV3LIB   = -L/usr/medm/lib -lpV3 -lgpvm3 -lpvm3 -lfpvm3 -lpgftnrtl
-lpgc

#---------------------------------------------------------------------------
#       EQUILIBRIUM TIDE
#---------------------------------------------------------------------------

#            FLAG_14 = -DEQUI_TIDE

#---------------------------------------------------------------------------
#       ATMOSPHERIC TIDE
#---------------------------------------------------------------------------

#            FLAG_15 = -DATMO_TIDE

#---------------------------------------------------------------------------
#       ARCTIC OCEAN INCLUDED (If you chose this flag, FLAG_2 should be
#                              selected)
#---------------------------------------------------------------------------

#             FLAG_16 = -DNORTHPOLE

#---------------------------------------------------------------------------
#        Using A fully multidimensional positive definite advection
#        transport algorithm with small implicit diffusion.
#        Based on Smolarkiewicz, P. K; Journal of Computational
#        Physics, 54, 325-362, 1984
#---------------------------------------------------------------------------
```

```
#            FLAG_17 = -DMPDATA

#------------------------------------------------------------------------------
#         Run Two-D Barotropic Mode Only
#------------------------------------------------------------------------------

#             FLAG_18 = -DTWO_D_MODEL

#------------------------------------------------------------------------------
#         Output 2-D Momentum Balance Checking
#------------------------------------------------------------------------------

#            FLAG_19 = -DBALANCE_2D

#------------------------------------------------------------------------------
#        Open boundary T/S time series nudging
#------------------------------------------------------------------------------

#            FLAG_20 = -DTS_OBC

#------------------------------------------------------------------------------
#           OPEN BOUNDARY FORCING TYPE
#           DEFAULT: OPEN BOUNDARY NODE WATER ELEVATION FORCING
#           UNCOMMENT TO SELECT BOTH OPEN BOUNDARY NODE WATER ELEVATION
#           FORCING AND OPEN BOUNDARY VOLUME TRANSPORT FORCING
#-------------------------------------------------------------------------------

#            FLAG_21 = -DMEAN_FLOW

#------------------------------------------------------------------------------
#           OUTPUT TIDAL INFORMATION AT NTIDENODE and NTIDECELL
#           FOR MEANFLOW CALCULATION.
#-------------------------------------------------------------------------------

#            FLAG_22 = -DTIDE_OUTPUT

#------------------------------------------------------------------------------
#           dye release
#-------------------------------------------------------------------------------

            FLAG_23 = -DDYE_RELEASE

#------------------------------------------------------------------------------
#        SELECT COMPILER/PLATFORM SPECIFIC DEFINITIONS
#          SELECT FROM THE FOLLOWING PLATFORMS OR USE "OTHER" TO DEFINE
#          THE FOLLOWING VARIABLES:
#          CPP:  PATH TO C PREPROCESSOR
#           FC:  PATH TO FORTRAN COMPILER (OR MPI COMPILE SCRIPT)
#          OPT:  COMPILER OPTIONS
#       MPILIB:  PATH TO MPI LIBRARIES (IF NOT LINKED THROUGH COMPILE SCRIPT)
#------------------------------------------------------------------------------

.
.
.
```

Case 2: Freshwater discharge over an idealized continental shelf

a) Design of the numerical experiment

Consider the linear sloping continental shelf shown in Figure 16.2. The freshwater is discharged onto the shelf from a point source at a distance of 200 km from the origin.

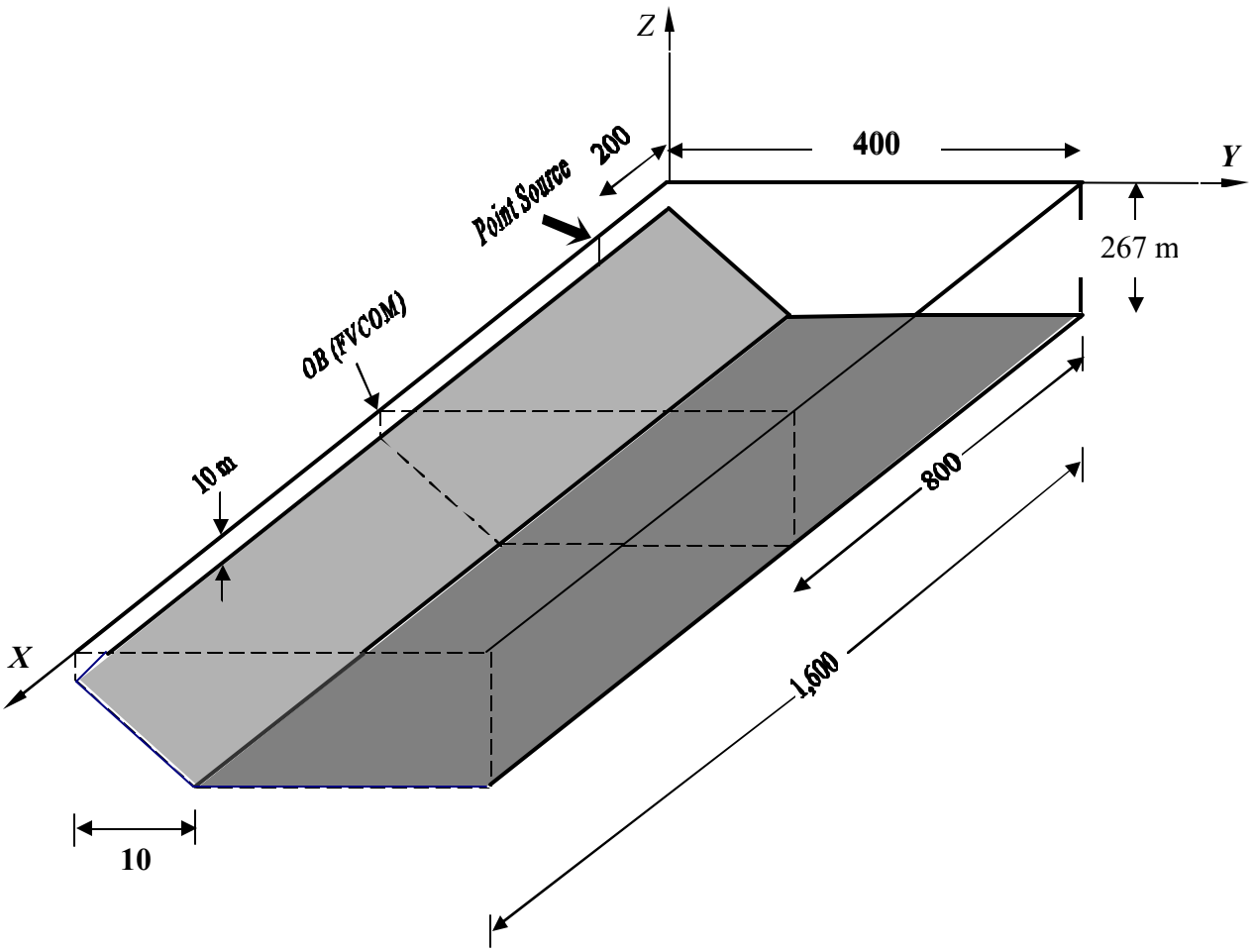


Fig. 16.2: An idealized linear slope continental shelf with a freshwater discharge at a point source.

The freshwater discharge rate Q is specified as 1000 m^3/s and the background salinity S = 30 PSU. The atmospheric surface forcing and boundary tidal forcing are both set to zero.

The numerical domain is configured with unstructured triangular grids with a resolution of 20 km (Fig. 16.3). Ten sigma levels are used in the vertical. The open boundary is located at 800 km downstream from the origin, at which a gravity wave radiation boundary condition is specified to allow the wave energy to propagate out of the computational domain with minimum reflection. The numerical grid is shown below:

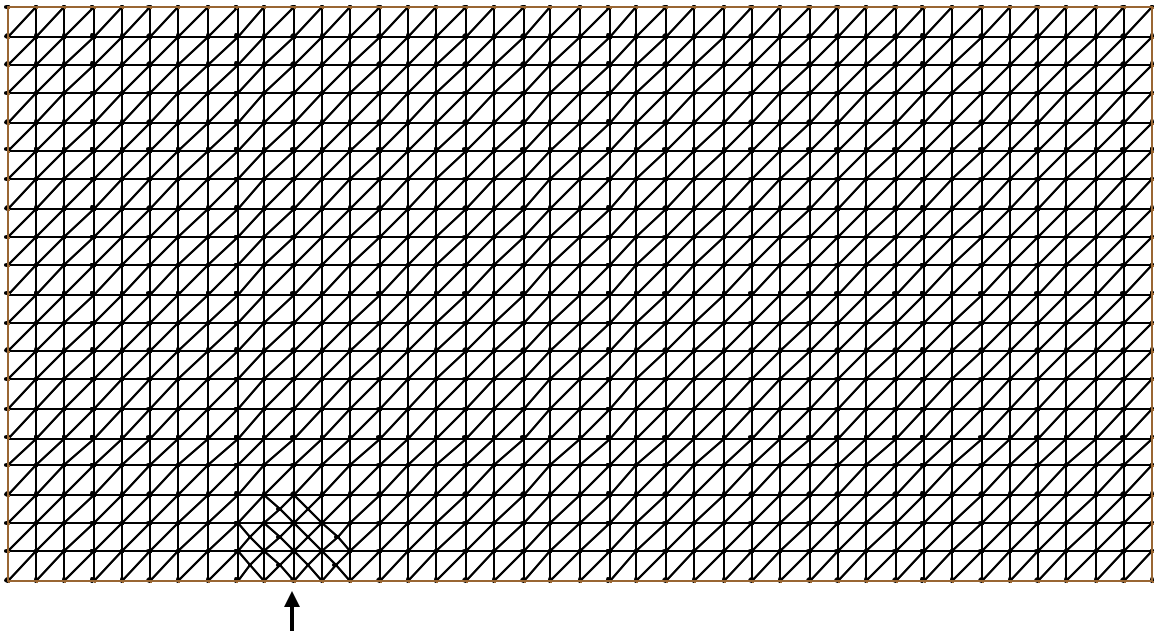


Fig. 16.3: Unstructured triangular grid for the case 2 experiment. The arrow indicates the location of the freshwater discharge.

b) The input files

1) chn_bfw.dat

```
Number of bottom fresh water input point
   0
```

2) chn_cor.dat

```
 0.8000000E+06  0.0000000E+00     30.00
 0.8000000E+06  0.2000000E+05     30.00
**
***
****
```

```
    0.0000000E+00  0.4000000E+06     30.00
```

3) chn_dep.dat

```
    800000.0000          0.0000 10.0000
    800000.0000 20000.0000 28.0000
    800000.0000 40000.0000 46.0000
    800000.0000 60000.0000 64.0000
   **
   ***
    0.0000 380000.0000 100.0000
    0.0000 400000.0000 100.0000
```

4) chn_el_obc.dat

```
Channel BC for tidal constituents (no tidal forcing)
  0
    1  0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    2  0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    3  0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    4 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    5 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    6 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    7 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    8 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
```

```
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    9 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    10 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    11 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    12 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    13 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    14 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    15 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    16 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    17 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    18 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    19 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
```

```
    20 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
    21 0.0
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
   0.0000000E+00   0.0000000E+03   0.0000000E+00   0.0000000E+00
0.0000000E+00   0.0000000E+00
```

5) chn_grd.dat

```
      1      22      23       1      1
      2       2      22       1      1
      3       2       3      22      1
      4      22       3      24      1
      **
      ***
   1620     871     870     850      1
      1 8.00000000e+005 0.00000000e+000 0.00000000e+000
      2 8.00000000e+005 2.00000000e+004 0.00000000e+000
      3 8.00000000e+005 4.00000000e+004 0.00000000e+000
      4 8.00000000e+005 6.00000000e+004 0.00000000e+000
      **
      ***
    870 0.00000000e+000 3.80000000e+005 0.00000000e+000
    871 0.00000000e+000 4.00000000e+005 0.00000000e+000
```

6) chn_ts.dat

```
Input temperature and salinity
constant
20. 30.
```

7) chn_mc.dat

```
Time variable meterological data
       0.0
   0.00000   0.00000   0.00000 180.00000   0.00000   0.00000
 9999999.0
   0.00000   0.00000   0.00000 180.00000   0.00000   0.00000
```

8) chn_obc.dat

```
OPEN_BOUNDARY NODE
     1           1       8
     2           2       8
     3           3       8
     4           4       8
     5           5       8
     6           6       8
     7           7       8
     8           8       8
     9           9       8
    10          10       8
```

```
    11          11       8
    12          12       8
    13          13       8
    14          14       8
    15          15       8
    16          16       8
    17          17       8
    18          18       8
    19          19       8
    20          20       8
    21          21       8
```

9) chn_riv.dat

```
node  calculated
1
634
634 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
   2
  0.
  1000.
  20.
  0.
  9999999.
  1000.
  20.
  0.
```

10) chn_spg.dat

```
0
```

11) chn_run.dat

```
!==============================================================================
!
!   INPUT FILE FOR PARAMETERS CONTROLLING EXECUTION OF FVCOM
!
!   DESCRIPTION OF VARIABLES AND SUGGESTED PARAMETERS CAN BE FOUND AT BOTTOM
!
!
!
!        FORMAT:                                                          !
!       1.) VARIABLE  = VALUE  (EQUAL SIGN MUST BE USED)
!
!       2.) FLOATING POINT VARIABLES MUST CONTAIN A PERIOD "." EX: 1.3, 2.,etc
!
!       3.) BLANK LINES ARE IGNORED AS ARE LINES BEGINNING WITH ! (F90
COMMENT)!
!       4.) COMMENTS CAN FOLLOW VALUES IF MARKED BY !
!
!       5.) ORDER OF VARIABLES IS NOT IMPORTANT
!
!       6.) FOR MULTIPLE VALUE VARIABLES FIRST ENTRY IS NUMBER OF VARIABLES
!
!           TO FOLLOW (OR 0 IF NONE)
!
```

```
!       7.) DO NOT USE COMMAS TO SEPARATE VARIABLES
!
!       8.) DO NOT EXCEED EIGHTY CHARACTERS PER LINE
!
!       9.) FOR LINE CONTINUATION ADD \\ TO END OF LINE TO FORCE CONTINUE
!
!           TO NEXT LINE.  MAXIMUM 4 CONTINUATIONS
!
!       10.) TRUE = T, FALSE = F
!
!
!
!  THE PREVIOUS FORMAT OF "VARIABLE: VALUE" IS NO LONGER VALID
!
!  THE MORE ATTRACTIVE " = " FORMAT WAS SUGGESTED BY Hernan G. Arango
!
!    AND SUBSEQUENTLY ADOPTED
!
!==============================================================================
!


!============ Case Title=============================================

CASETITLE = FRESH WATER CHANNEL

!=========Parameters Controlling Time Integration=====================

 DTE    = 20.00
 ISPLIT = 10
 IRAMP  = 000
 NSTEPS = 4320

!=========Parameters Controlling Input/Output=======================

 INPDIR = ./CHANNEL
 OUTDIR = ./OUTDIR_CHAN
 INFOFILE = screen
 IREPORT = 100
 IRECORD = 432
 IDMPSMS = 432
 IRESTART = 0
 RESTART = cold_start

!=========Parameters Controlling Output of Average Data Fields====

AVGE_ON  = F
INT_AVGE = 5
BEG_AVGE = 1
NUM_AVGE = 10
CDF_OUT_AVE = F
CDF_VDP_AVE = u v el

!=========Parameters Controlling Bottom Friction=====================

BFRIC = .0025
Z0B    = .001
BROUGH_TYPE = orig
```

```
!=========Parameters Controlling Diffusivity Coefficients==============

HORZMIX = closure
HORCON  = 4.000E-1
HPRNU   = 1.000E+0
VERTMIX = closure
UMOL    = 1.000E-6
VPRNU   = 1.000E+0

!=========Parameters Controlling Model Equations=======================

BAROTROPIC = F
SALINITY_ON = T
TEMP_ON = F

!=========Parameters Controlling Density Calculation===============

C_BAROPG  = sigma
CTRL_DEN  = sigma-t
VERT_STAB = T

!=========Parameters Controlling Atmospheric Forcing==============

H_TYPE = flux_h
M_TYPE = uniform
WINDTYPE = stress
ZETA1 = 1.4
ZETA2 = 6.3
RHEAT = .78
THOUR_HS = 124.

!=========Parameters Controlling Temp/Salinity Adjustments=========

TS_FCT = T

!=========Parameters Controlling Depth Adjustments================

DJUST = 0.0
MIN_DEPTH = 0.05

!=========Parameters Controlling Tidal Forcing=======================

S_TYPE = non-julian
DELTT   = 720.

!=========Standard Depth Levels=====================================

KSL    =   11
DPTHSL =   -0. -10. -20. -30. -40. -50. -60. -70. -80. -90. -100.

!============ Parameters Controlling Sigma Distribution==============

P_SIGMA = 1.0
KB      = 11

!============ Parameters Controlling Lagrangian Tracking==============
```

```
LAG_ON = F

!============ Parameters Controlling Time Series Output==========

PROBE_ON = F

!============ Parameters Controlling Water Quality Module=============

WQM_ON = F
NB     = 8
BENWQM_KEY = F

!============ Parameters Controlling Wetting/Drying====================

WET_DRY_ON = T

!============ Parmaeters Controlling SST Data Assimilation=============

SST_ASSIM = F
RAD_SST = 10000.
GAMA_SST = 1.0
GALPHA_SST = 3.e-3
ASTIME_WINDOW_SST = .50
IAV_DAY = 5

!====CURRENT DATA ASSIMILATION VARIABLES==============================

CURRENT_ASSIM = F
RAD_CUR = 20000.
GAMA_CUR = 1.0
GALPHA_CUR = 8.3e-3
ASTIME_WINDOW_CUR = .50

!====TEMPERATURE/SALINITY DATA ASSIMILATION VARIABLES==================

TS_ASSIM = F
RAD_TS = 30000.
GAMA_TS = 1.0
GALPHA_TS = 8.3e-3
ASTIME_WINDOW_TS = 72.

!====VARIABLES CONTROLLING NETCDF OUTPUT==============================
CDF_OUT = F
CDF_INT = 90
CDF_STK = 0
CDF_VDP = u v t1 s1 el
INFO1   = Dave Test
INFO2   = More Information Here

 !==== Parameter Controlling Richardson # dep. dissipation
 correction============
 SURFACEWAVE_MIX = F

 !==== Parameter Controlling Open Boundary Temp/Salt
 Nudging=====================
 TS_NUDGING_OBC = F
 ALPHA_OBC = .1
```

```
!==== Parameter controlling Open Boundary Temp/Salt Series Nudging=====
TSOBC_ON = T
ALPHA_SERIES_OBC = .0014

!=====VARIABLES CONTROLLING 2D MOMENTUM BALANCE CHECKING
OUTPUT=================
OUT_BALANCE = F
NUM_BALANCE = 4
NO_CELL     = 11446 11212 11213 11447

!=====PARAMETER CONTROLLING THE TYPE OF Temp/Salt OBC=======
TYPE_TSOBC = 3

!==== Parameter controlling Tidal Open Boundary Output==========
TIDE_INITIAL  = 14400
TIDE_INTERVAL = 6
```

12) makefile

```
#-----------BEGIN MAKEFILE-----------------------------------------------
----
            SHELL         = /bin/sh
            DEF_FLAGS     = -P -C -traditional
            EXEC          = fvcom
#=================================================================
====
#  BEGIN USER DEFINITION SECTION
#=================================================================
====
#        SELECT MODEL OPTIONS
#          SELECT FROM THE FOLLOWING OPTIONS BEFORE COMPILING CODE
#          SELECT/UNSELECT BY COMMENTING/UNCOMMENTING LINE (#)
#          CODE MUST BE CLEANED (with "make clean") AND
#          RECOMPILED IF NEW SET OF OPTIONS IS DESIRED
#-----------------------------------------------------------------
----

#-----------------------------------------------------------------
----
#        PRECISION          DEFAULT PRECISION: SINGLE
#                           UNCOMMENT TO SELECT DOUBLE PRECISION
#-----------------------------------------------------------------
----

#            FLAG_1 =  -DDOUBLE_PRECISION

#-----------------------------------------------------------------
----
#        SPHERICAL          SELECT SPHERICAL COORDINATES FOR
INTEGRATION
#                           DEFAULT: CARTESIAN
#                           UNCOMMENT TO SELECT SPHERICAL COORDINATES
#-----------------------------------------------------------------
----
```

```
#             FLAG_2 = -DSPHERICAL

#------------------------------------------------------------------
----
#        FLOODYING/DRYING   INCLUDE WET/DRY TREATMENT OF DOMAIN
#                           CAN BE ACTIVATED/DEACTIVATED AT RUN TIME
WITH
#                           INPUT FILE CONTROL.  (SEE exa_run.dat) FILE
#                           DEFAULT: NO FLOODYING/DRYING INCLUDED
#                           UNCOMMENT TO INCLUDE FLOODYING/DRYING
#------------------------------------------------------------------
----

#             FLAG_3 = -DWET_DRY

#------------------------------------------------------------------
----
#        MULTI_PROCESSOR    INCLUDES PARALLELIZATION WITH MPI
#                           REQUIRES LINKING MPI LIBRARIES OR COMPILING
#                           WITH A PRELINKED SCRIPT (mpif90/mpf90/etc)
#                           DEFAULT: NO PARALLEL CAPABILITY
#                           UNCOMMENT TO INCLUDE MPI PARALLEL
CAPABILITY
#------------------------------------------------------------------
----

             FLAG_4 = -DMULTIPROCESSOR
             PARLIB = -L../METIS_source -lmetis

#------------------------------------------------------------------
----
#        WATER_QUALITY      INCLUDE EPA WATER QUALITY MOD
#                           CAN BE ACTIVATED/DEACTIVATED AT RUN TIME
WITH
#                           VARIABLE WQM_ON IN INPUT FILE
#                           DEFAULT: NO WATER QUALITY MODEL
#                           UNCOMMENT TO INCLUDE WATER QUALITY MODEL
#------------------------------------------------------------------
----

#             FLAG_5 = -DWATER_QUALITY

#------------------------------------------------------------------
----
#        NETCDF OUTPUT      DUMP OUTPUT INTO NETCDF FILES (yes/no)
#                           REQUIRES SYSTEM DEPENDENT NETCDF LIBRARIES
#                           COMPILED WITH SAME F90 COMPILER
#                           SET PATH TO LIBRARIES WITH IOLIBS
#                           SET PATH TO INCLUDE FILES (netcdf.mod) WITH
IOINCS
#                           DEFAULT: NO NETCDF OUTPUT
#                           UNCOMMENT TO INCLUDE NETCDF OUTPUT
CAPABILITY
#------------------------------------------------------------------
----
#           FLAG_6       =  -DNETCDF_IO
#        IOLIBS       =  -L/usr/medm/lib  -lnetcdf
```

```
#           IOINCS       =  -I/usr/medm/include
#---------------------------------------------------------------------
----
#        DATA_ASSIMILATION  INCLUDE NUDGING BASED DATA ASSIMILATION FOR
#                           CURRENT/TEMP/SALINITY/SST
#                           CAN BE ACTIVATED/DEACTIVATED AT RUN TIME
WITH
#                           INPUT FILE CONTROL.  (SEE exa_run.dat) FILE
#                           DEFAULT: NO DATA ASSIMILATION INCLUDED
#                           UNCOMMENT TO INCLUDE DATA ASSIMILATION
#---------------------------------------------------------------------
----

#             FLAG_7 = -DDATA_ASSIM

#---------------------------------------------------------------------
----
#       REMOVE SYSTEM CALLS IF RUNNING IN A DOS/WINDOWS ENVIRONMENT
#                           OR IF COMPILER FAILS WHEN COMPILING SYSTEM
#                           CALLS, TURN ON THIS OPTION.
#                           THIS WILL DEACTIVATE CERTAIN COMMANDS WHICH
#                           SETUP THE OUTPUT DIRECTORY STRUCTURE
#                           NOTE THAT THE USER WILL HAVE TO CREATE THE
#                           OUTPUT DIRECTORIES MANUALLY
#                           SEE THE README FILE INCLUDED WITH THE
#                           DISTRIBUTION
#                           DEFAULT: INCLUDE SYSTEM CALLS
#                           UNCOMMENT TO SUPRESS SYSTEM CALLS
#---------------------------------------------------------------------
----

#             FLAG_8 += -DDOS

#---------------------------------------------------------------------
-----
#       SOLID BOUNDARY     IF GCN, NO GHOST CELL
#                          IF GCY1, GHOST CELL IS SYMMETRIC RELATIVE TO
BOUNDARY
#                          CELL EDGE
#                          IF GCY2, GHOST CELL IS SYMMETRIC RELATIVE TO
MIDDLE
#                          POINT OF THE BOUNDARY CELL EDGE
#   !!!!!! ONLY ONE OF THE FLAGS BELOW CAN BE CHOSEN
#---------------------------------------------------------------------
-----

#             FLAG_9  = -DGCN
             FLAG_10  = -DGCY1
#             FLAG_11 = -DGCY2

#---------------------------------------------------------------------
----
#        TURBULENCE MODEL   USE GOTM TURBULENCE MODEL INSTEAD OF THE
ORIGINAL
#                           FVCOM MELLOR-YAMADA 2.5 IMPLEMENTATION
#                           UNCOMMENT TO USE GOTM TURBULENCE MODEL
```

```
#---------------------------------------------------------------------
----

#            FLAG_12 = -DGOTM
#            GOTMLIB       = -L../GOTM_source -lturbulence -lutil
#            GOTMINCS      = -I../GOTM_source

#---------------------------------------------------------------------
----
#        pV3 Realtime Visualization Server (MEDM GROUP ONLY)
#---------------------------------------------------------------------
----

#            FLAG_13 = -DPV3
#            PV3LIB   = -L/usr/medm/lib -lpV3 -lgpvm3 -lpvm3 -lfpvm3 -
lpgftnrtl  -lpgc

#---------------------------------------------------------------------
----
#       EQUILIBRIUM TIDE
#---------------------------------------------------------------------
----

#            FLAG_14 = -DEQUI_TIDE

#---------------------------------------------------------------------
----
#       ATMOSPHERIC TIDE
#---------------------------------------------------------------------
----

#            FLAG_15 = -DATMO_TIDE

#---------------------------------------------------------------------
----
#       ARCTIC OCEAN INCLUDED (If you chose this flag, FLAG_2 should be
#                              selected)
#---------------------------------------------------------------------
----

#             FLAG_16 = -DNORTHPOLE

#---------------------------------------------------------------------
----
#        Using A fully multidimensional positive definite advection
#        transport algorithm with small implicit diffusion.
#        Based on Smolarkiewicz, P. K; Journal of Computational
#        Physics, 54, 325-362, 1984
#---------------------------------------------------------------------
----

#            FLAG_17 = -DMPDATA

#---------------------------------------------------------------------
----
#         Run Two-D Barotropic Mode Only
```

```
#------------------------------------------------------------------------
----

#              FLAG_18 = -DTWO_D_MODEL

#------------------------------------------------------------------------
----
#          Output 2-D Momentum Balance Checking
#------------------------------------------------------------------------
----

#              FLAG_19 = -DBALANCE_2D

#-------------------------------------------------------------------------
#        Open boundary T/S time series nudging
#-------------------------------------------------------------------------

#              FLAG_20 = -DTS_OBC

#------------------------------------------------------------------------
----
#            OPEN BOUNDARY FORCING TYPE
#            DEFAULT: OPEN BOUNDARY NODE WATER ELEVATION FORCING
#            UNCOMMENT TO SELECT BOTH OPEN BOUNDARY NODE WATER ELEVATION
#            FORCING AND OPEN BOUNDARY VOLUME TRANSPORT FORCING
#------------------------------------------------------------------------
-----

#              FLAG_21 = -DMEAN_FLOW

#------------------------------------------------------------------------
----
#            OUTPUT TIDAL INFORMATION AT NTIDENODE and NTIDECELL
#            FOR MEANFLOW CALCULATION.
#------------------------------------------------------------------------
-----

#              FLAG_22 = -DTIDE_OUTPUT

#------------------------------------------------------------------------
----
#            dye release
#------------------------------------------------------------------------
-----

             FLAG_23 = -DDYE_RELEASE  .
.
.
.
```

Chapter 17: Unstructured Triangular Mesh Generation

FVCOM uses unstructured triangular grids and since no automatic mesh generator is supplied with the coding, users must use alternate software to build the mesh. There are a multitude of mesh generators available. Some are open source and several are commercial. Any software which is capable of generating two-dimensional, unstructured triangular grids would work for FVCOM. However, high powered meshing software designed for complex 3-D grids, such as Gridgen, ICEM, or GAMBIT are expensive, difficult to learn, and are unnecessarily powerful. We have listed some recommended meshes on the FVCOM users website.

At MEDM/SMAST, we primarily use the SMS (Surface Water Model System) to generate unstructured grids in FVCOM. SMS is module-based commercial software that can be purchased in whole or by module. Only three modules are necessary for mesh generation. They are the Mesh Module, the Map Module, and the Scatter Module. The beta version can be downloaded from the SMS website:

http://www.ems-i.com/SMS/SMS_Overview/sms_overview.html/

In this document, we will guide you through an example mesh generation using SMS to teach the basic procedure of generating meshes for FVCOM.

17.1. Data Preparation

To generate an unstructured grid, users need to have first acquired relevant data including: 1) coastlines and 2) bathymetry.

a) Coastline preparation

FVCOM can be run in either spherical or Cartesian coordinates. For applications in the coastal ocean, we recommend that users use Cartesian coordinates. In general, the coastline data are in a geographic format of longitude and latitude. A projection program is needed to convert the longitude and latitude to the Cartesian coordinates relative to a selected reference point. Procedures to prepare the coastline data are given below.

Step 1: Download the coastline data

Recommended website: http://www.ngdc.noaa.gov/mgg/geodas/geodas.html.

NOAA has a GEODAS CD available with GEODAS coastline extractor.

The format of the download coastline data is as follows:

```
1   41.5756   -70.5038
2   41.5756   -70.5041
2   41.5757   -70.5043
2   41.5760   -70.5043
2   41.5761   -70.5042
2   41.5761   -70.5039
2   41.5760   -70.5036
2   41.5757   -70.5036
3   41.5756   -70.5038
```

Column 1: the data type: 1-start point; 2-points between start and end points; 3-end point.

Column 2: Latitude

Column 3: Longitude

Step 2: Convert the longitude and latitude to Cartesian x-y coordinates

- Download the Cartographic Projection Tool called "**Proj**" from website http://proj.maptools.org/ and install it.

- Use "Proj" to convert the longitude and latitude format data to the x-y coordinate.

The format of the output data file looks like

```
871884.3374     -139646.0237
871859.3184     -139645.9259
871842.6826     -139634.7546
871842.8129     -139601.4361
871851.1959     -139590.3625
871876.2146     -139590.4603
871901.19       -139601.6641
871901.06       -139634.9826
871884.3374     -139646.0237
```

Column 1: *x* Column 2: *y*

Note: when "Proj" is used, be sure to keep the x and y locations of the reference point used for the projection. These will be required if you wish to convert the x and y locations back to Lat/Lon.

3. Insert the 1st column (data type) from the original coastline data (latitude and longitude format) into converted x-y data file.

"Proj" can only run with an input file of longitude and latitude data, so after the projection is completed, we need to add the 1st column of the original data into the projected data to keep the identification of the data type.

4. Run "ReadCST.f" to create the coastline file format compatible with SMS.

We have a simple Fortran 77 program called "readcst_new.f" on FVCOM user website. Users can use it or can write a very simple Matlab program to do it. The format of the resulting output file is:

```
COAST
   11 0.0
    9   0
          871884.3125        -139646.0313  0.0
          871859.3125        -139645.9219  0.0
          871842.6875        -139634.7500  0.0
          871842.8125        -139601.4375  0.0
          871851.1875        -139590.3594  0.0
          871876.1875        -139590.4531  0.0
          871901.1875        -139601.6719  0.0
          871901.0625        -139634.9844  0.0
          871884.3125        -139646.0313  0.0
   11   0
          871775.5625        -139734.4531  0.0
          871742.2500        -139723.2188  0.0
          871717.3125        -139712.0000  0.0
          871700.6875        -139689.7344  0.0
          871700.7500        -139667.5156  0.0
          871784.1875        -139667.8438  0.0
          871825.8125        -139679.1094  0.0
          871842.4375        -139701.3906  0.0
          871834.0000        -139723.5781  0.0
          871817.2500        -139734.6094  0.0
          871775.5625        -139734.4531  0.0
```

b) Bathymetric data preparation

The bathymetric data can be downloaded from the USGS bathymetric database or obtained from other data sources. In general, the data format is:

Latitude, Longitude, Depth

The same procedure described in the coastline preparation needs to be used here to convert the bathymetric data into the Cartesian x and y coordinates. Final data for SMS should look like:

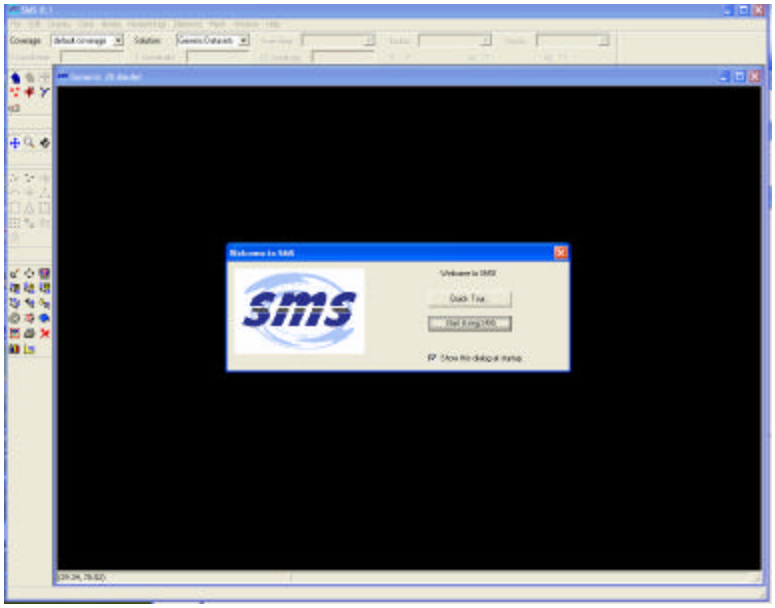
```
870118.000 -139242.750 1.770
870812.875 -139245.594 0.500
870116.063 -139705.547 1.880
870811.000 -139708.391 0.500
871853.500 -139712.547 0.710
868029.063 -140159.375 0.160
870114.125 -140168.328 1.900
870809.125 -140171.188 0.630
871504.188 -140173.969 0.880
872199.188 -140176.688 0.890
868027.000 -140622.063 0.030
870112.250 -140631.016 1.910
```

Column 1: *x* (meters) ; Column 2: *y* (meters); Column 3: depth (meters).

Note: FVCOM requires that the water depth is positive for wet points.

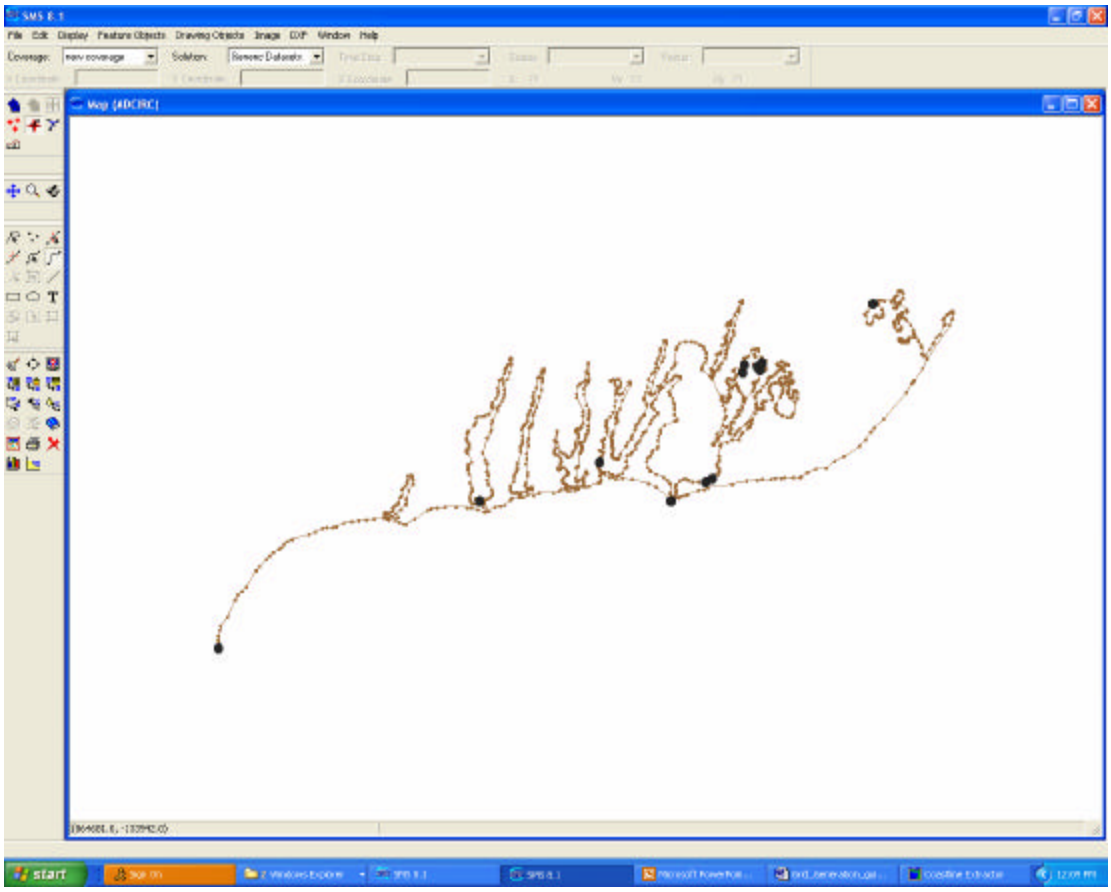
Now, you are ready to run "SMS".

17.2. Grid Generation



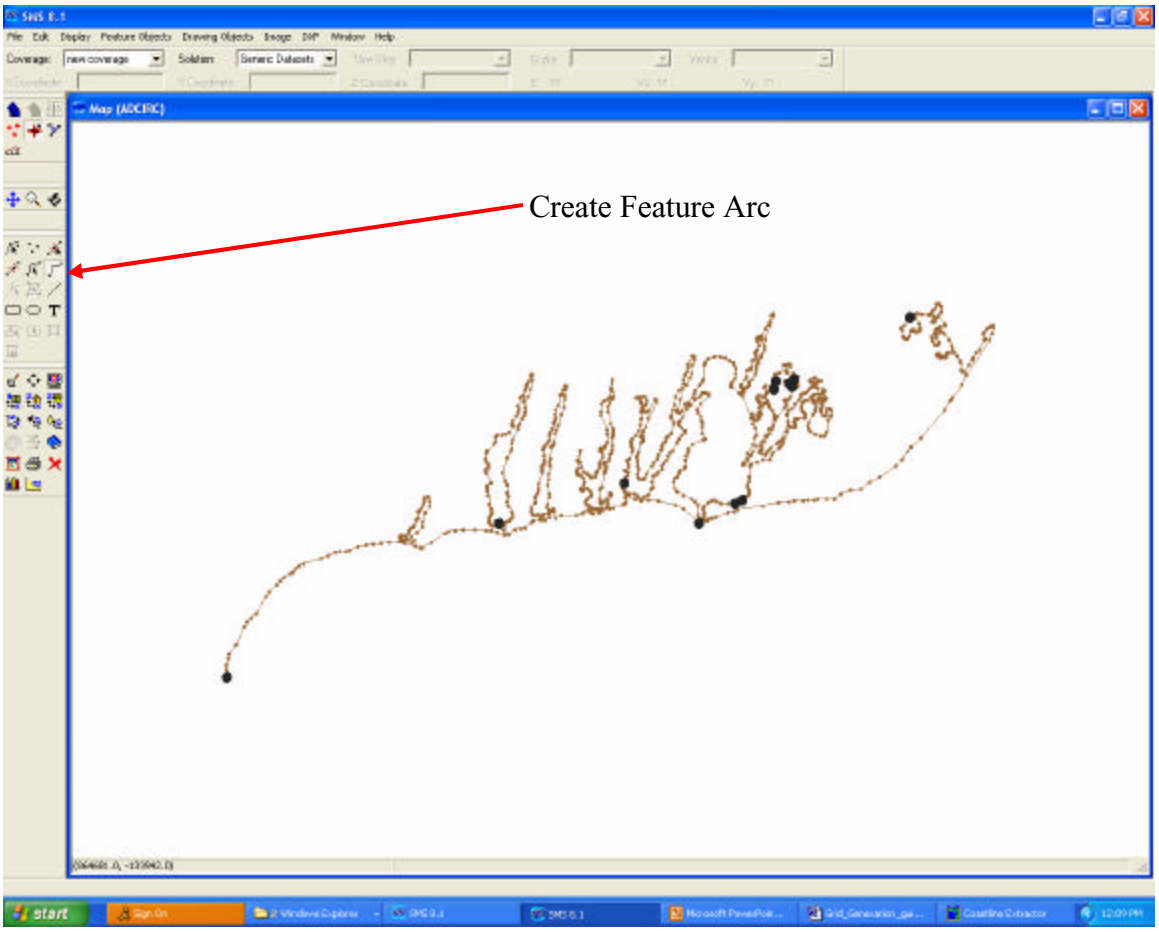
1. Click "SMS", select "Start Using SMS"

2. Go to “File” at the upper left corner. Select “Open” and then find the *.cst” file in your computer and open it. Select “ADCIRC” and click on it. Your coastline data should appear on the screen. This is an example of Waquoit Bay on the southern coast of Cape Cod, Massachusetts.

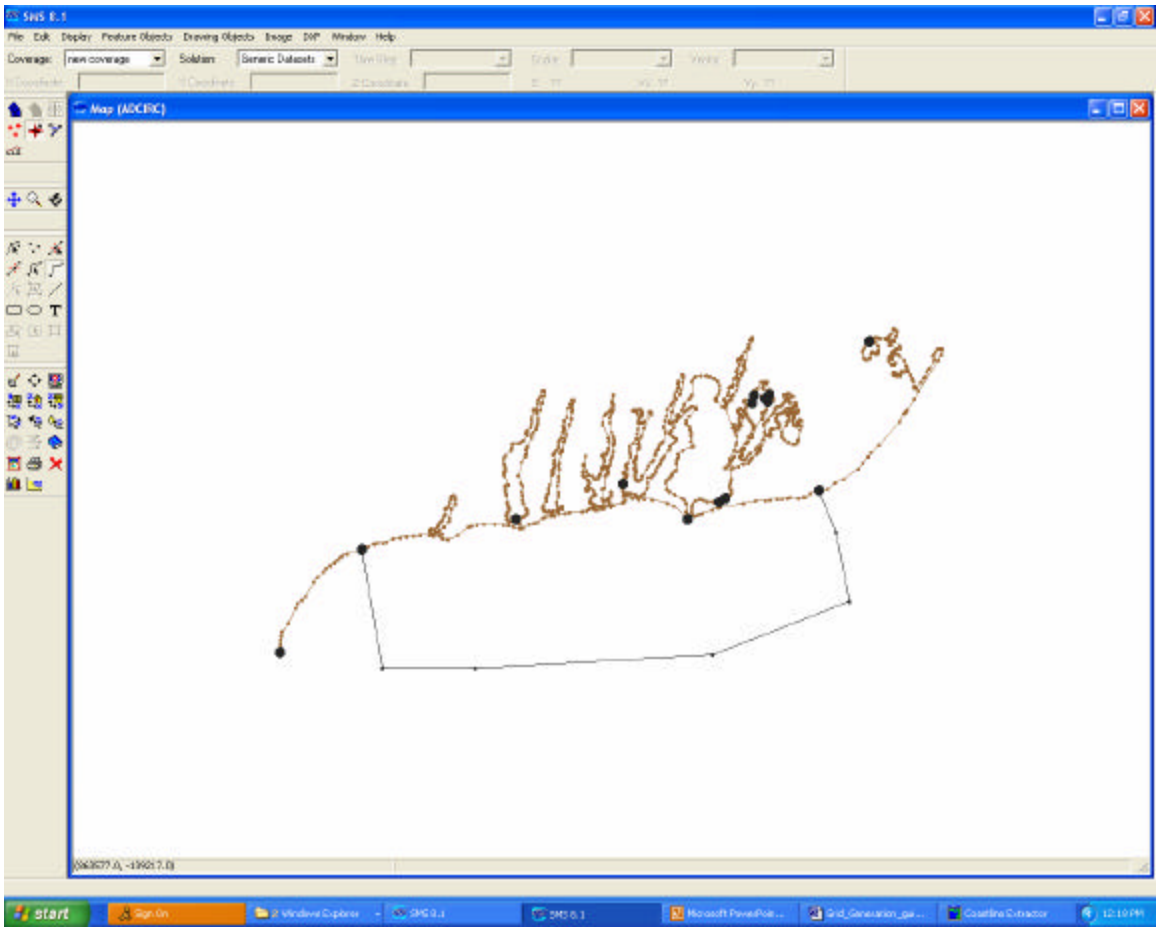


3. Create an initial open boundary

Specify the open boundary line user “Create Feature Arc”.



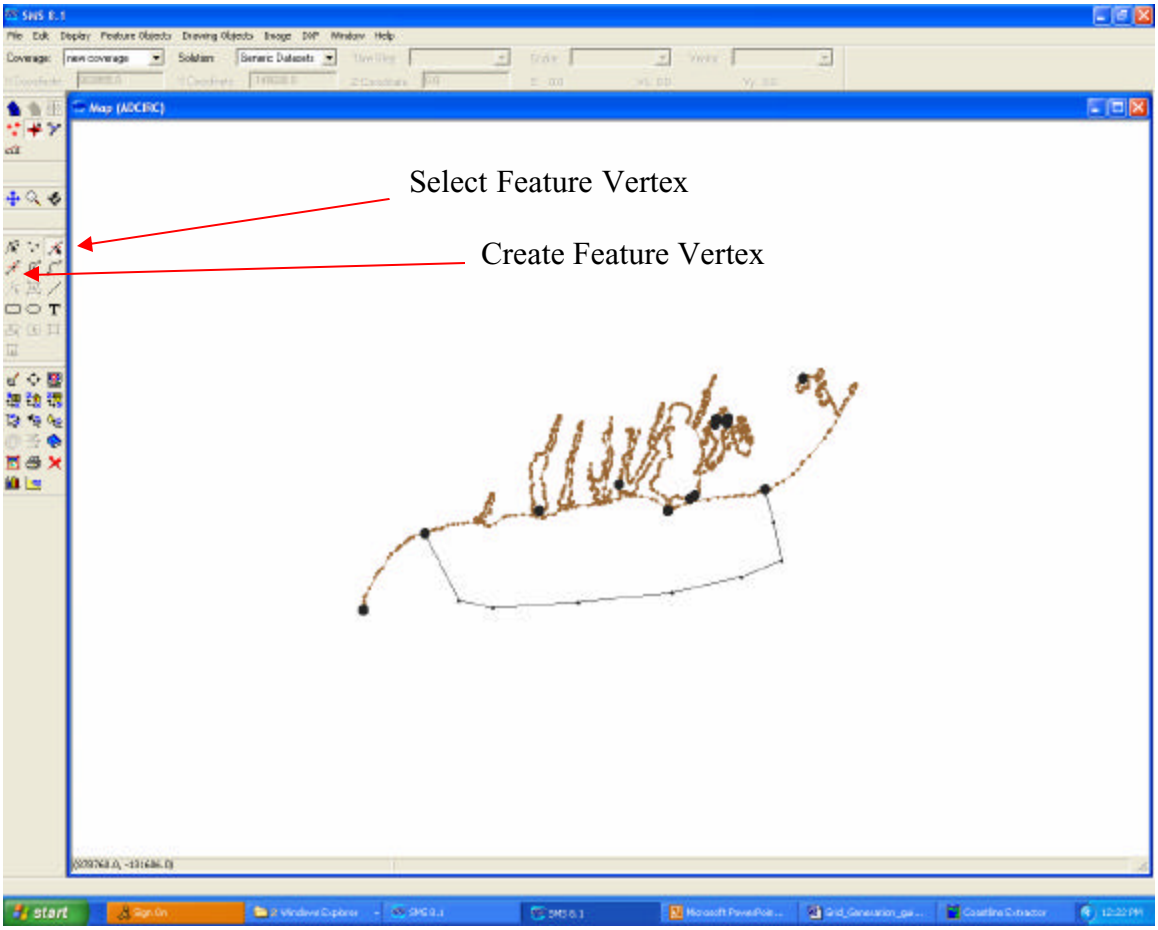
Start at a selected point at the coastline with one click, and then hold “shift”, click points to build an open boundary. Be sure to double click at the end point.



The dots are the points you click.

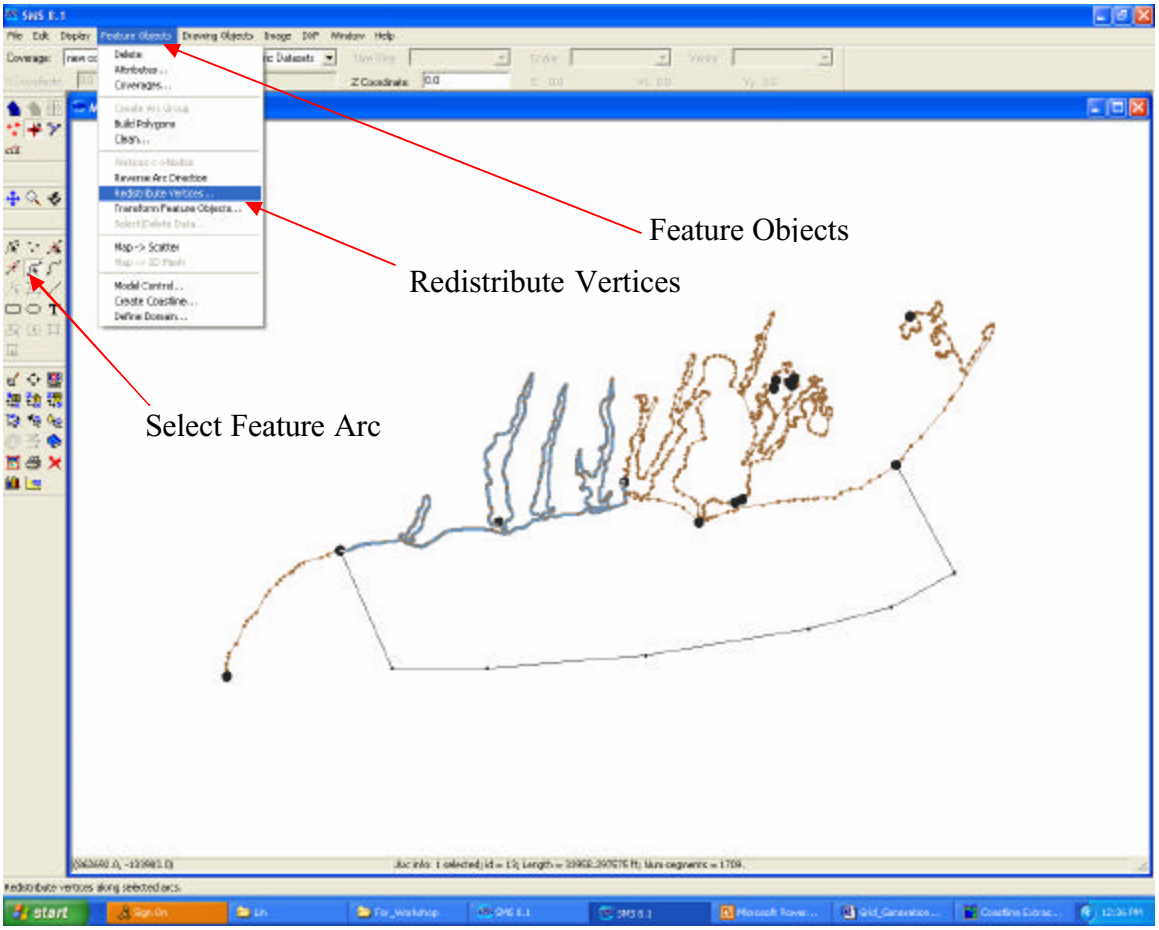
4. Smooth the open boundary line

Select "Select Feature Vertex", then use mouse to move the line to get the shape you like. You can add additional points by selecting "Create Feature Vertex"

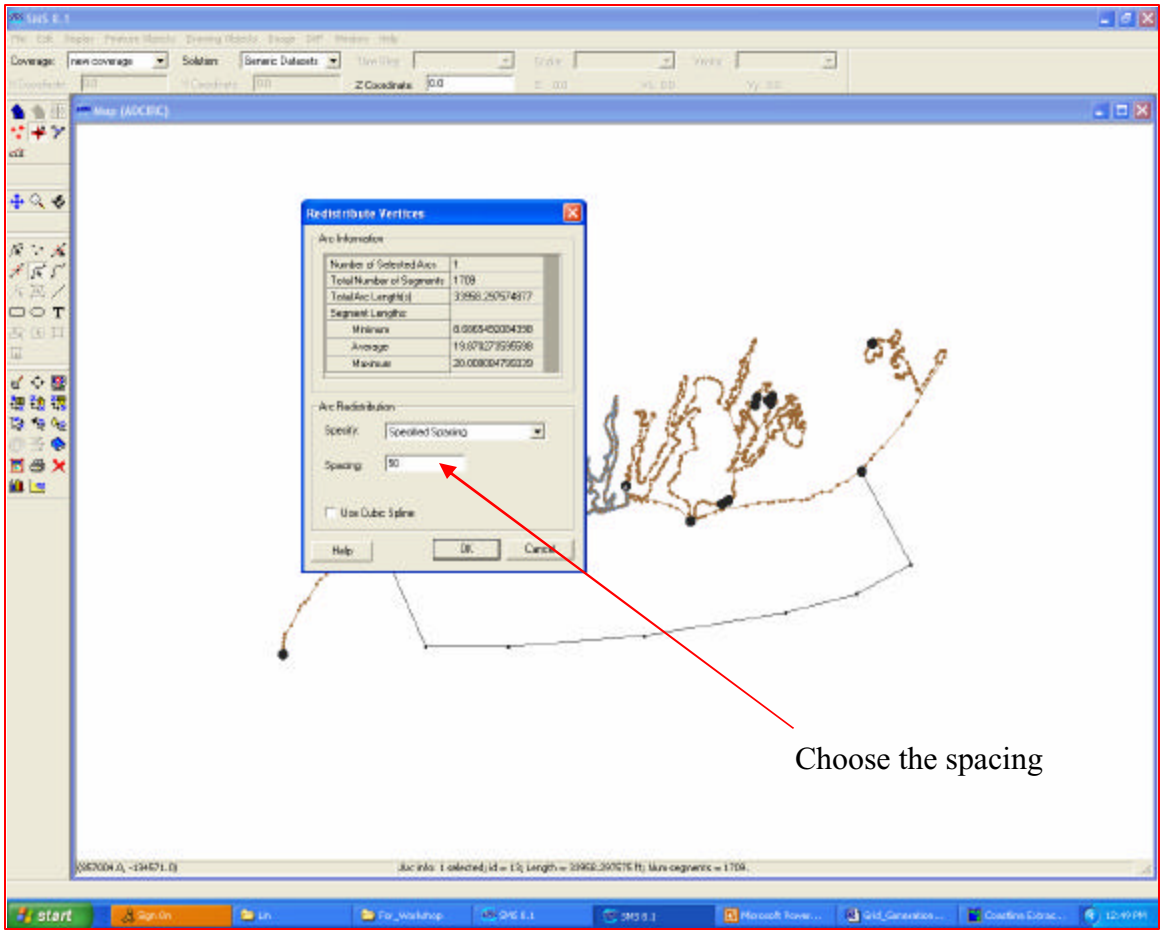


5. Choose the horizontal resolution

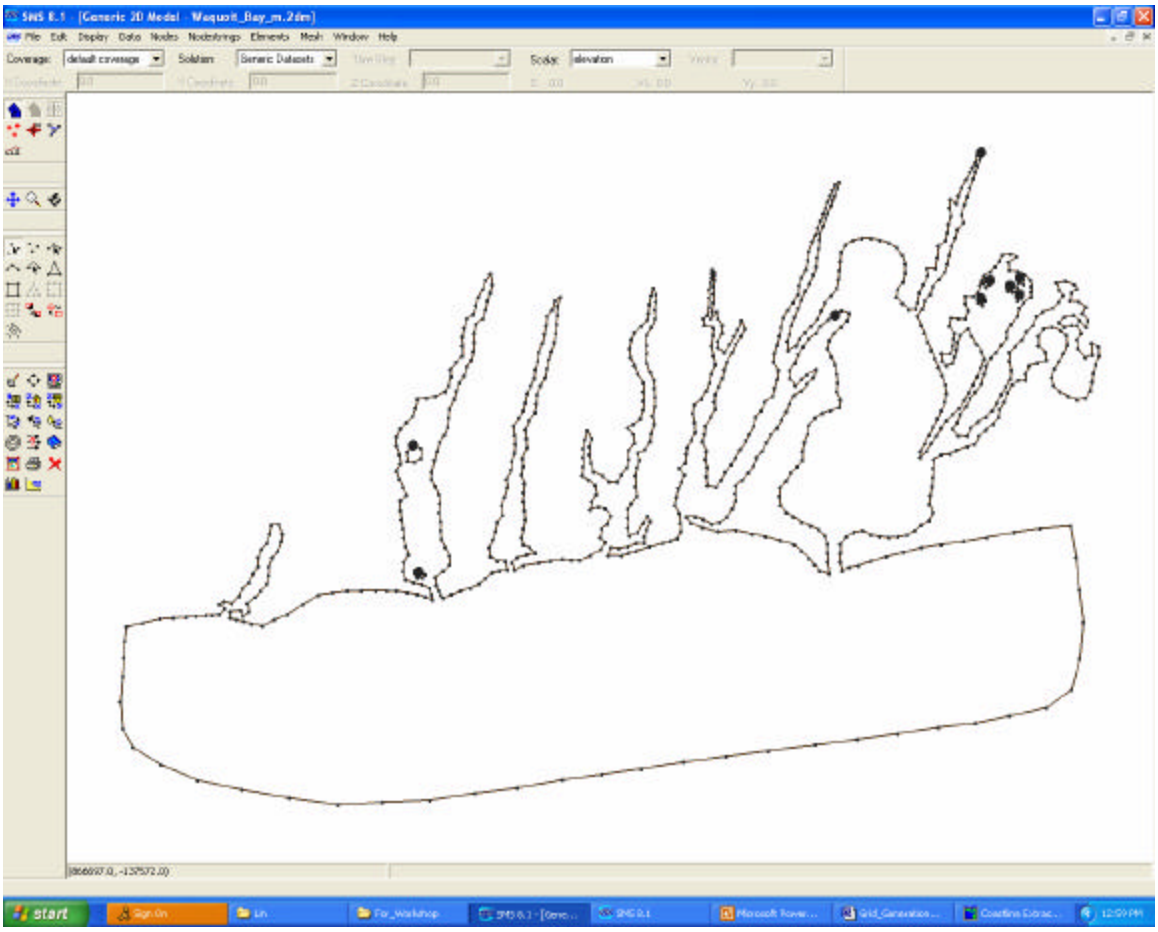
Click "Select Feature Arc", then move the mouse to the part of the coastline (the line with gray color is the part of the coastline you select); go to the upper manual line to select "Feature Objects" to select "Redistribute Vertices"



Click "Redistribute Vertices". In the sub-window of "distribute vertices", you can select the spacing interval. The unit of the length is the same as your input data. For example, we type "50" in "spacing", it means we want to have a horizontal resolution of about 50 m in the segments we choose.



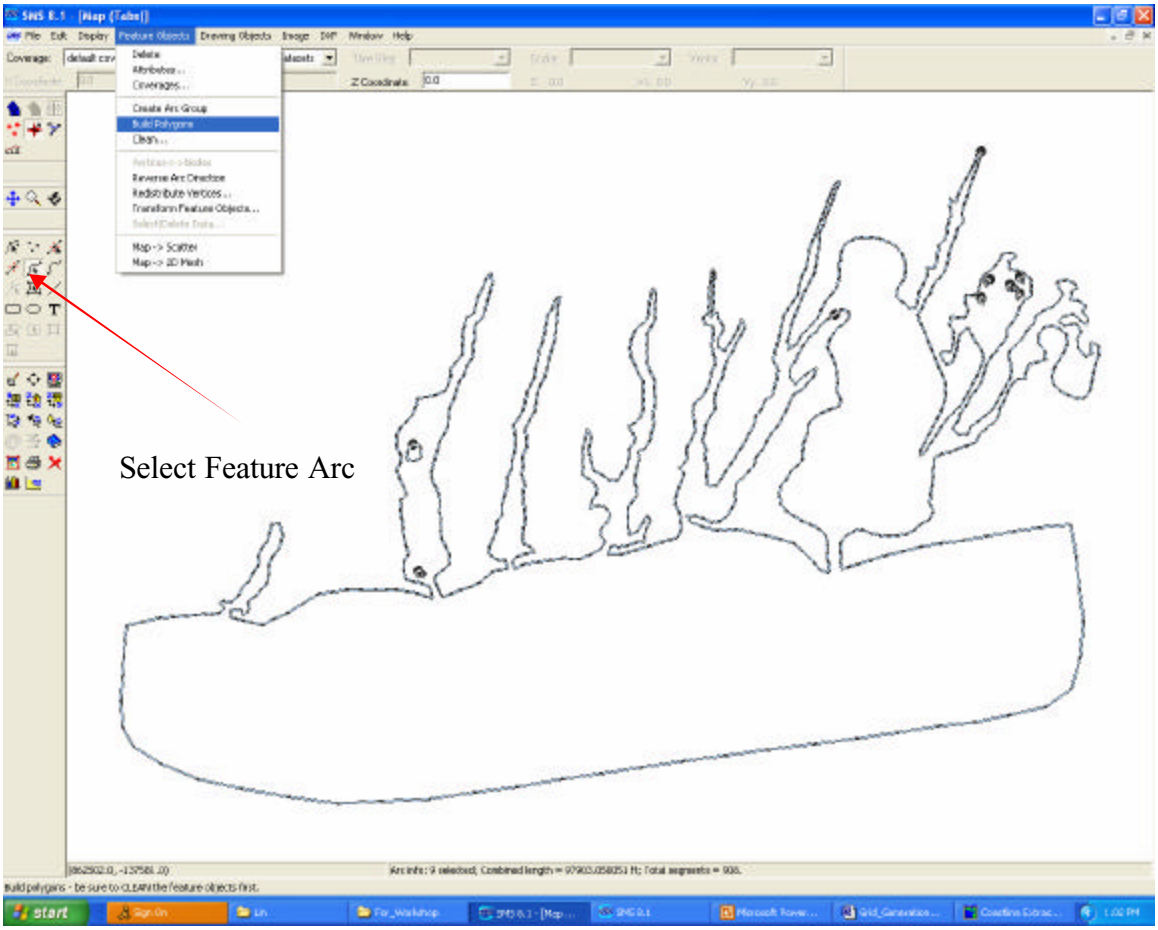
Repeat this exercise until you select all segments of the coastline and open boundary line. Be sure to save your map file. SMS does not save as your proceed and you cannot go backwards. After all these are completed, the screen should look like:



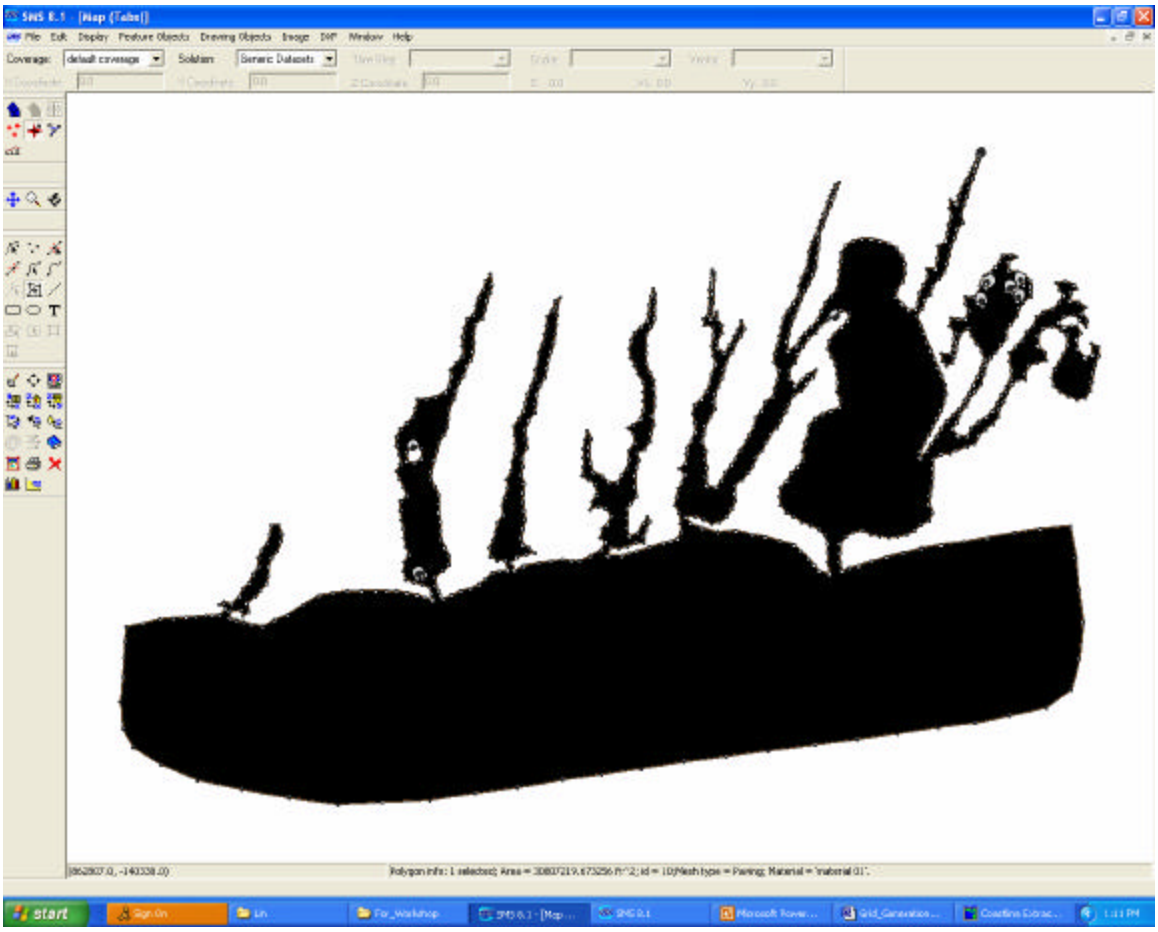
Next step is to build mesh!

6. Build the mesh

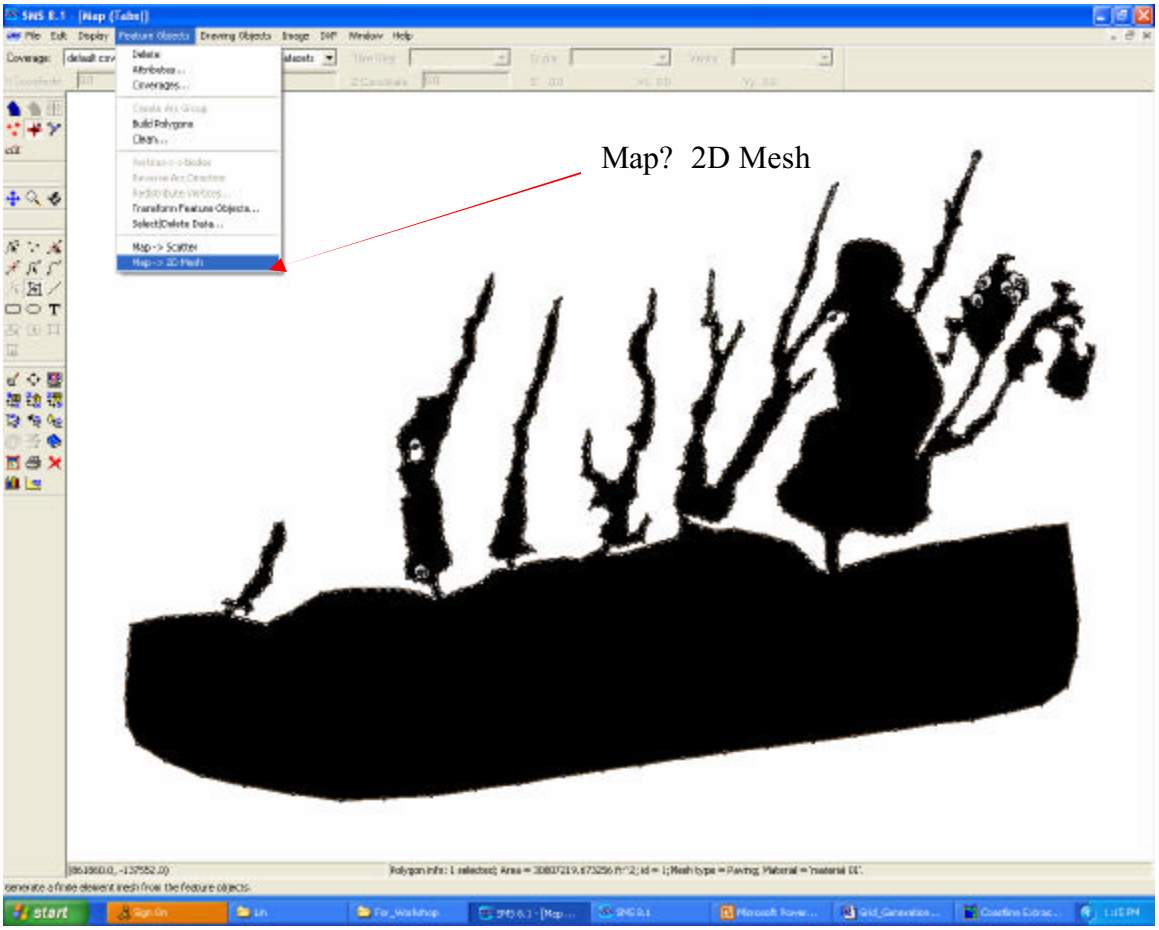
Click “Select Feature Arc”. Use your mouse to select all lines in the screen you want to include in building meshes. Then go to ’Feature Objects’ menu and select ‘Build polygons’.



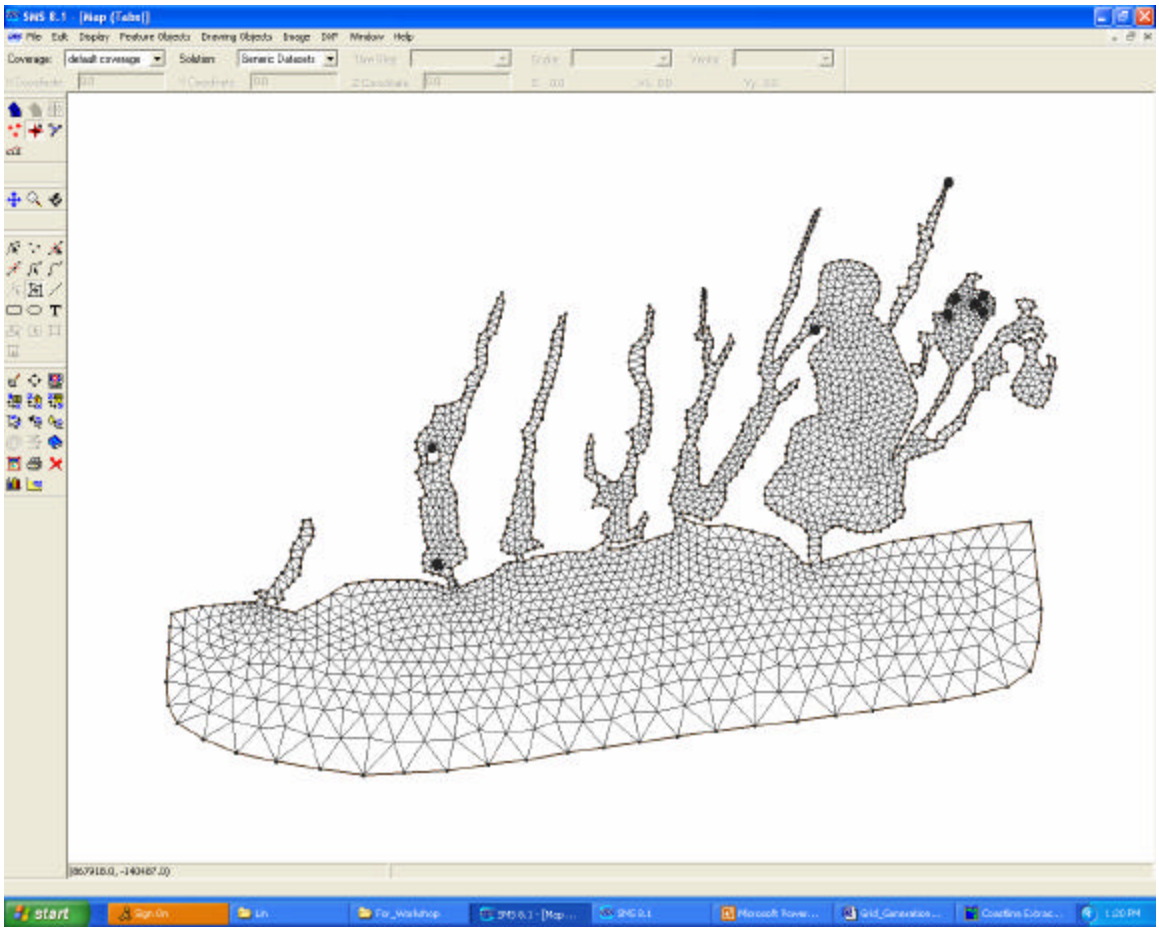
Click “Select Feature Polygons” on the left menu bar, then move the mouse back to the mesh domain, hold the shift key, and click it, the entire domain should become black. If you find some regions are not colored black, it means that your previous steps are not done correctly, either the line is not closed or perhaps another issue. Be sure to go back to fix it. When correct, your screen should look like this:



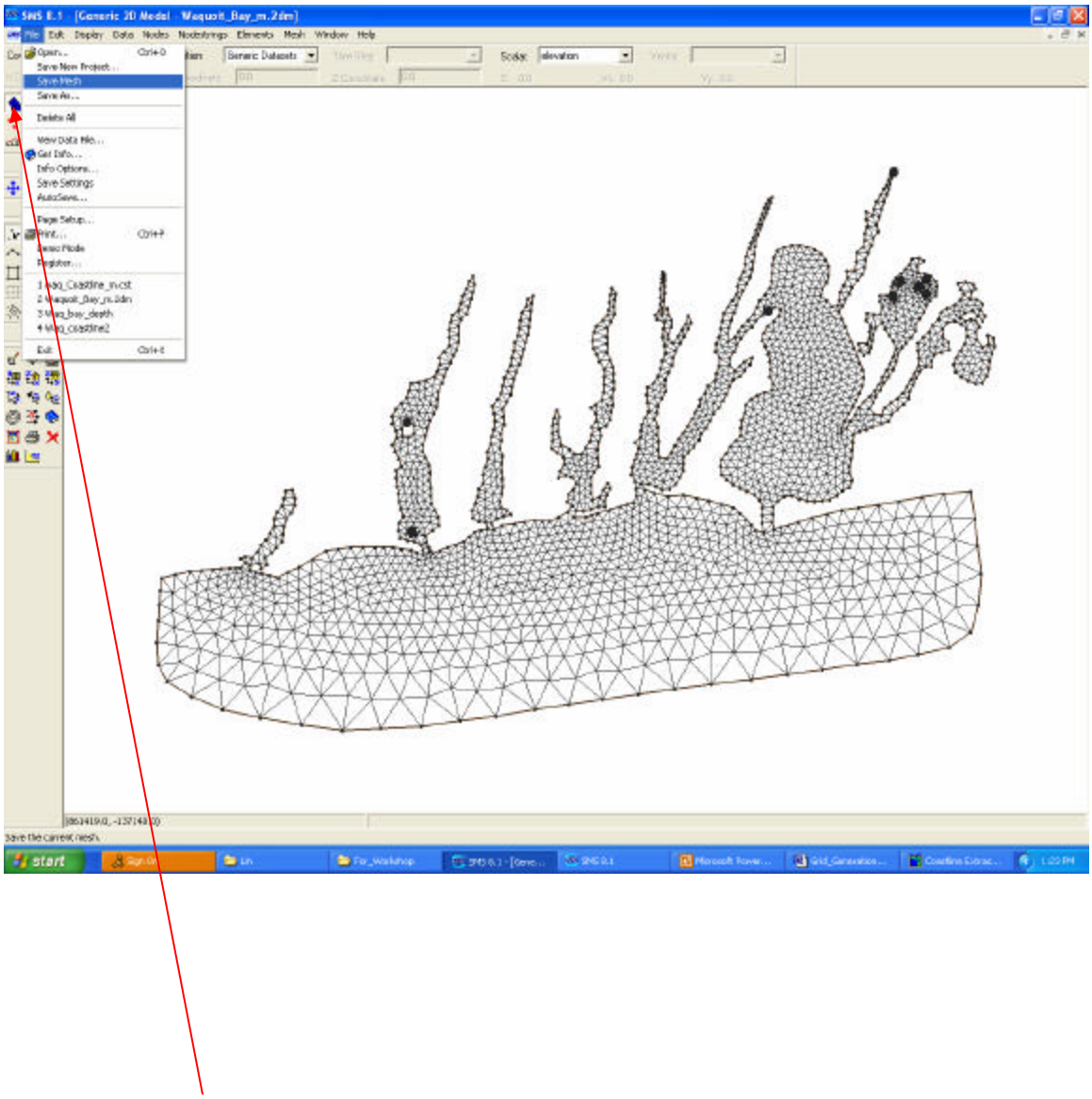
Go to the upper menu bar to and select "Map→2D Mesh" from the "Feature Objects'" pull-down menu. Click on it to start building your mesh.



Now, the mesh is built. See below for the result.



Be sure to save this mesh First, click ”Mesh Module” on the left side manual bar, then go to the upper menu “File” to select “Save Mesh”. Click on it, your mesh is saved.

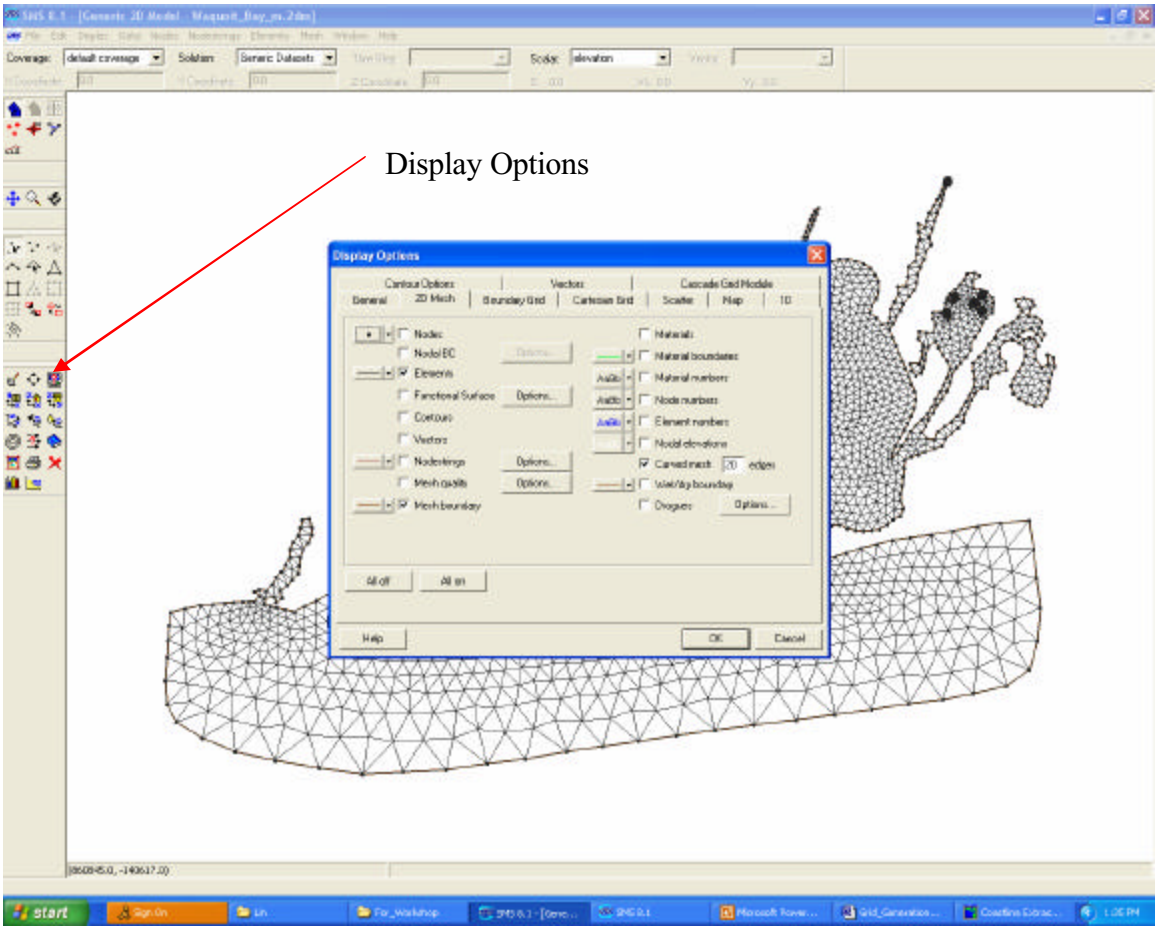


Mesh Module

7. Mesh Quality Control

Step 1: unlock the mesh. Remain in "Mesh Module", go to the upper menu to select "Nodes". In the sub-window of "Nodes", you will find "locked" box which will be checked. That means that the mesh is locked. Click "locked" to unlock the mesh.

Step 2: go to the left menu bar to select "Display Options". Click on it, you should see a screen like this:



Step 3: check the box "Mesh quality", click "options" next to "Mesh quality", an element quality checks screen appears. Users need to input their quality requirement parameters into each row. Below are our recommendations:

Minimum interior angle: 30.0

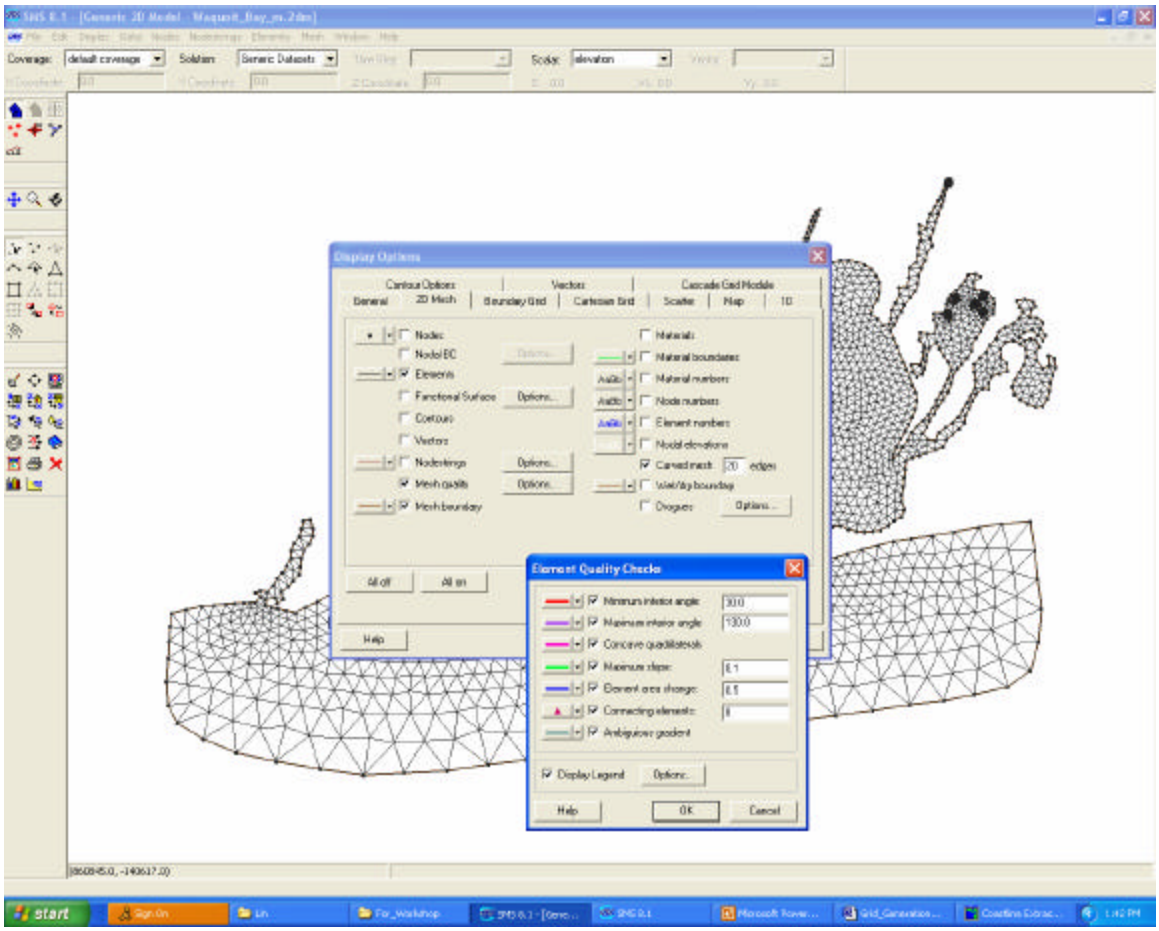
Maximum interior angle: 130.0

Maximum slope: 0.1

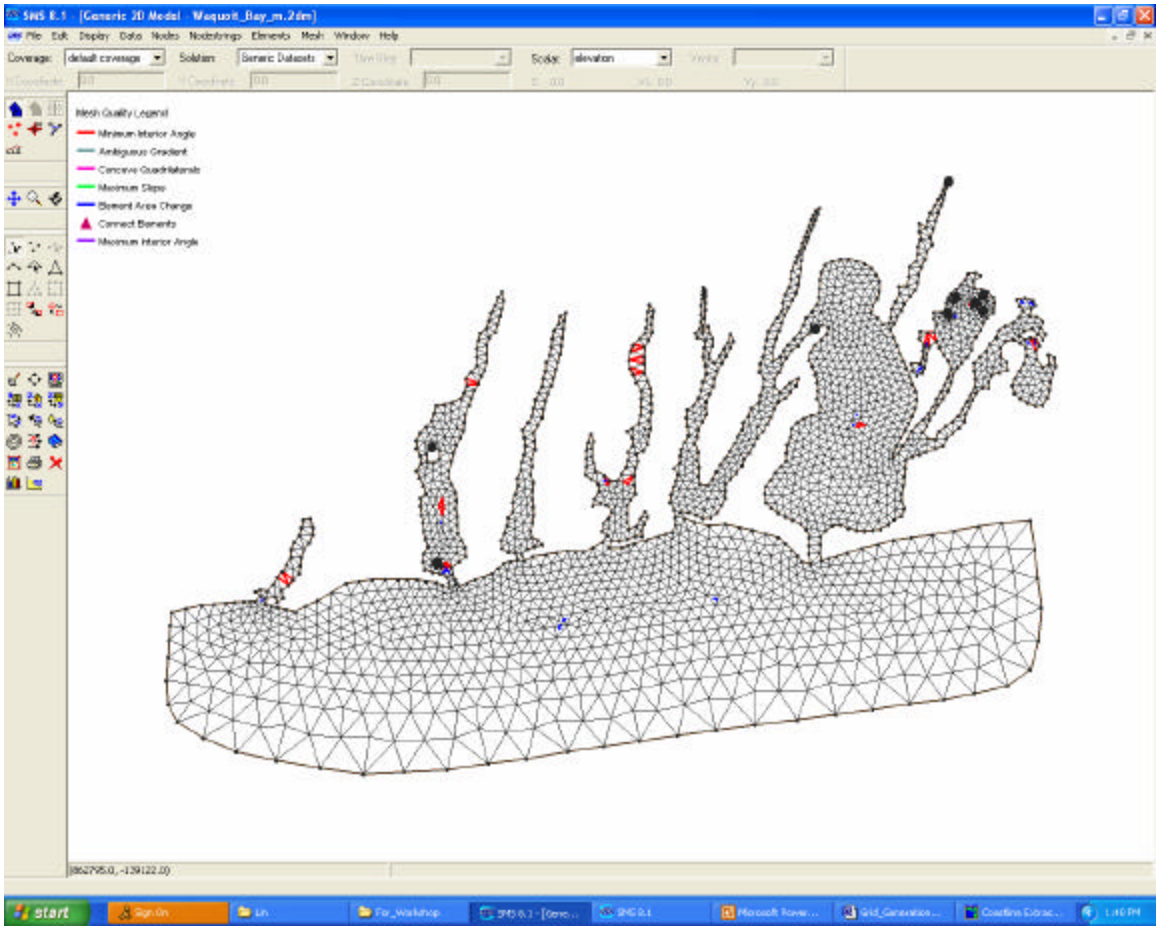
Element area change: 0.5

Connecting elements: 8 or less (8 is the maximum elements allowed for FVCOM).

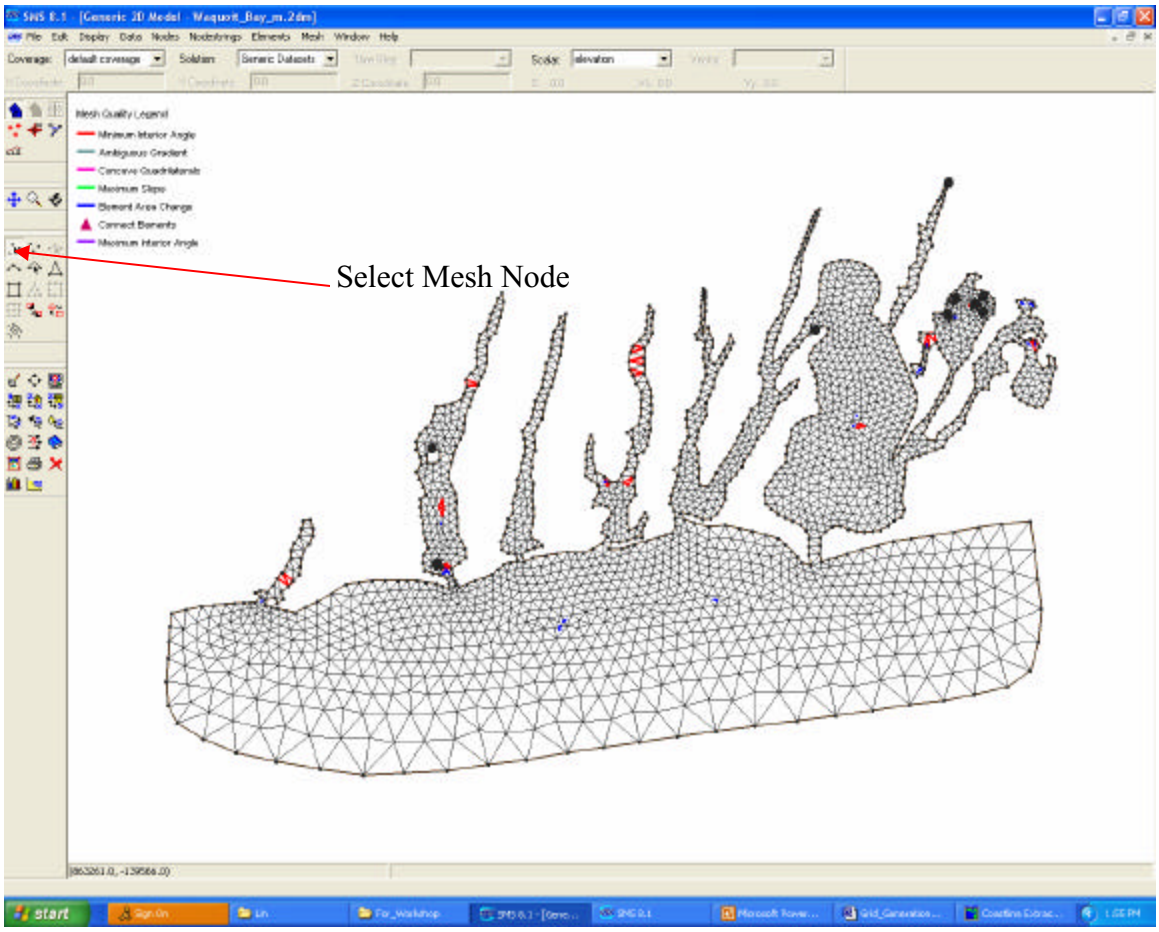
After all selections, click "ok".



Below is the result of the quality selection. Red colors show that the angle of an element is out of the range we specified, and blue colors show that the element area change of those triangles are out of the range we expected.



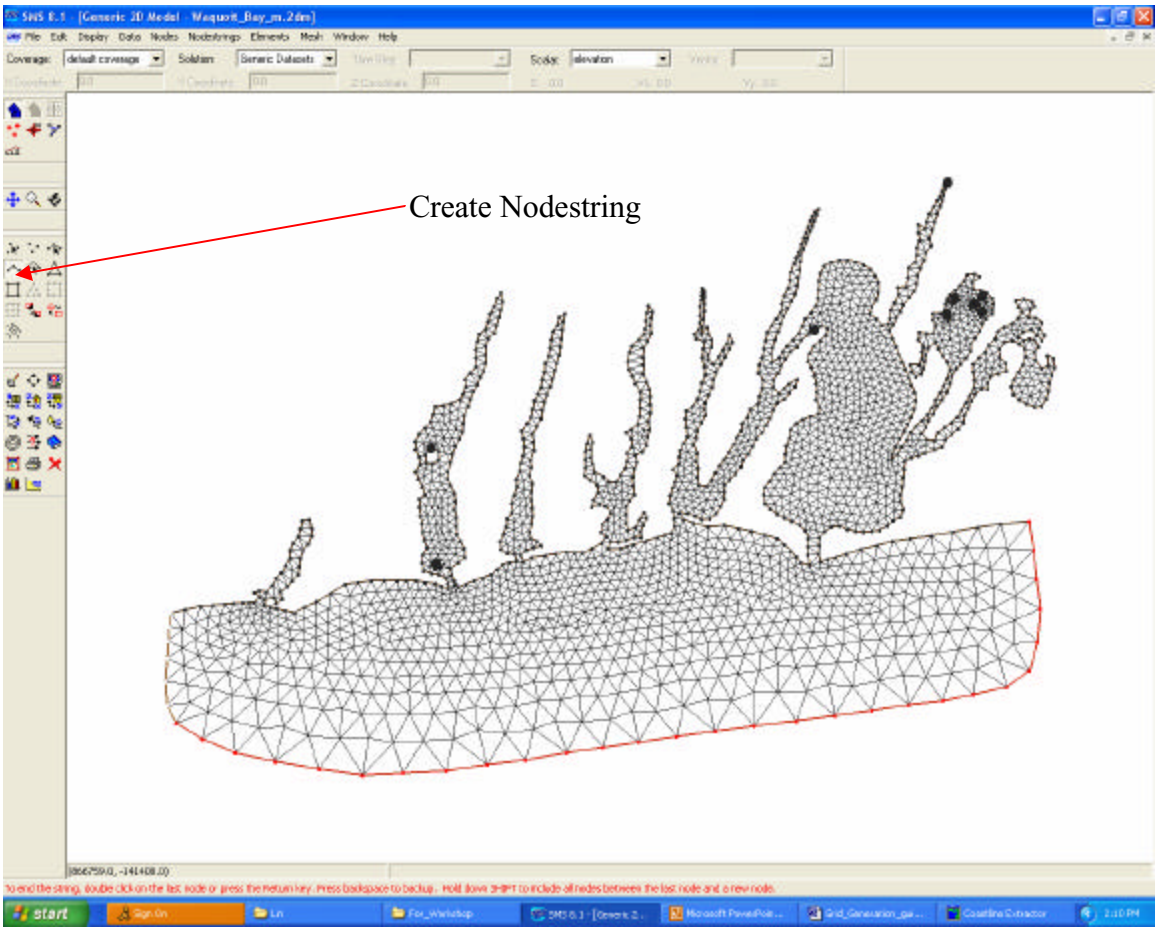
Step 4: start adjusting the mesh angles and area. The simplest way is to adjust using the mouse. Go to the left manual bar to click on "Select Mesh Node". Use the mouse to go to the colored area, and move the nodes until the color disappears. Repeat for all colored regions.



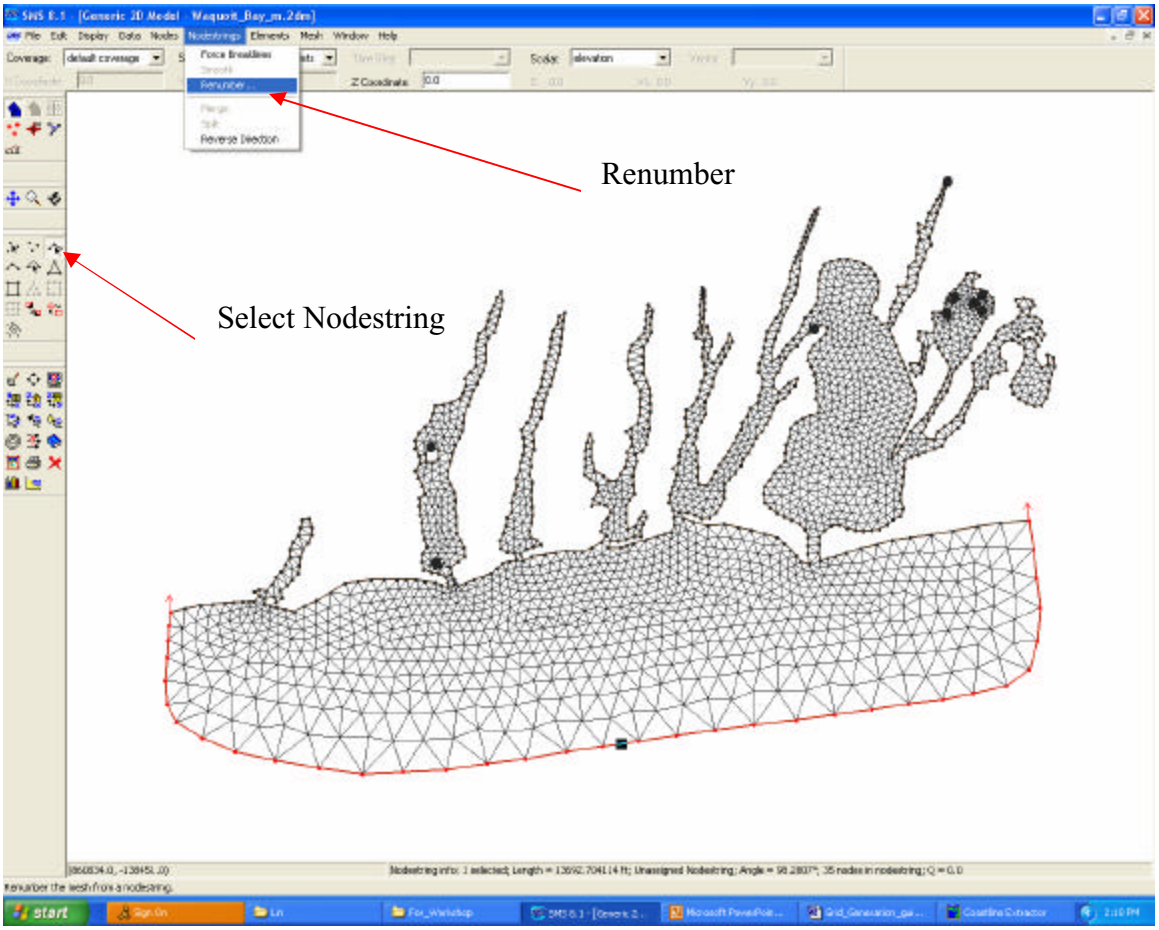
Note: To make the radiation open boundary condition work properly, we suggest that users make one of the interior edges of an open boundary triangle normal to the open boundary. FVCOM will run without this restriction, but with increased numerical noise due to the high frequency wave reflection from the open boundary.

8. Select the open boundary nodes to build an input file for the open boundary condition treatment.

Step 1: Go to the left manual bar to click "Create Nodestring". Move your mouse to the first node of the open boundary line connected to the coastline. Hold "shift" and then continue to click the open boundary nodes. When you get to the last end node point connected to the other side of the coastline, double click it. Check the screen to make sure all boundary nodes are selected.



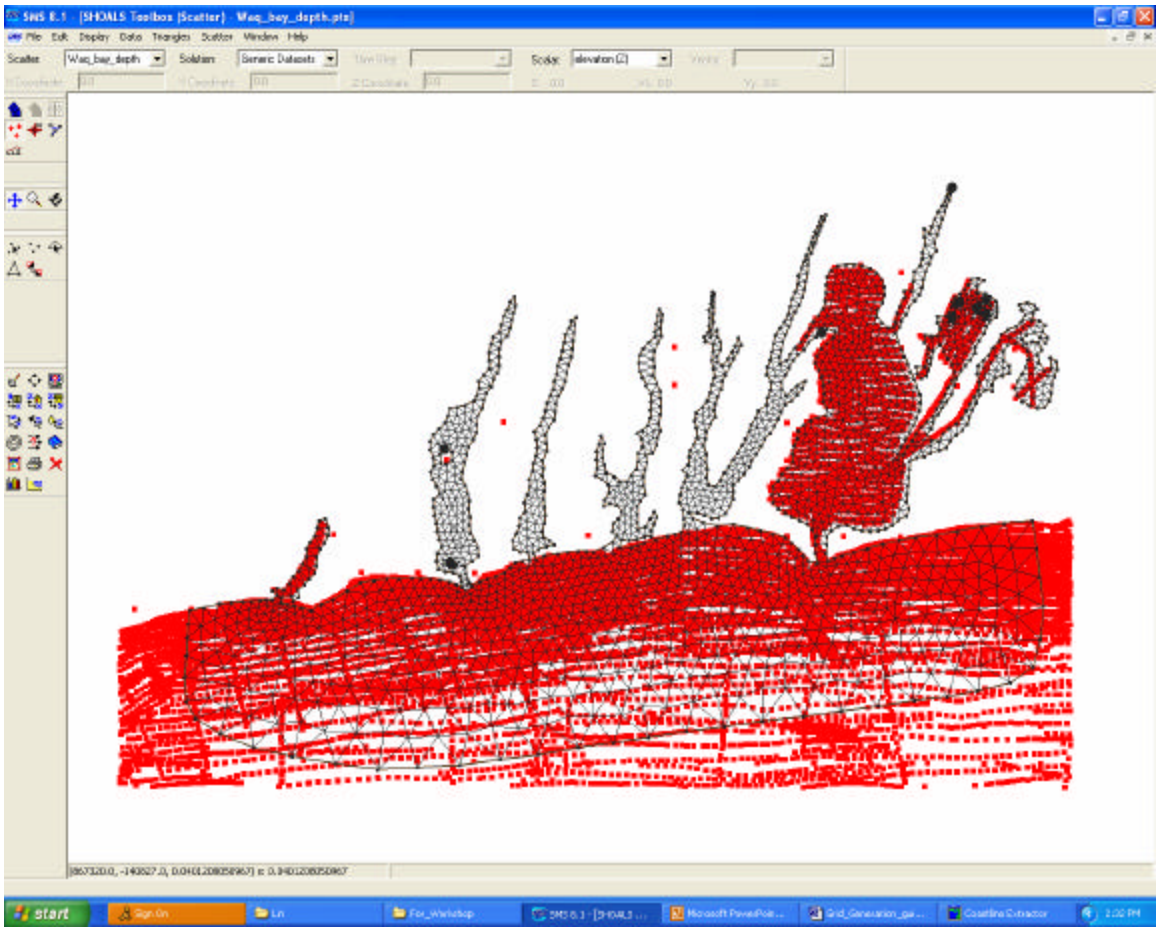
Step 2: Re-number the open boundary nodes. Go to the left manual bar to click on "Select Nodestring". A small box will appear on the open boundary line. Click this box, all the open boundary nodes will be colored. Go to the upper manual bar to select "Nodestring". On that sub-window, click on "renumber". All nodes on the open boundary will be renumbered.



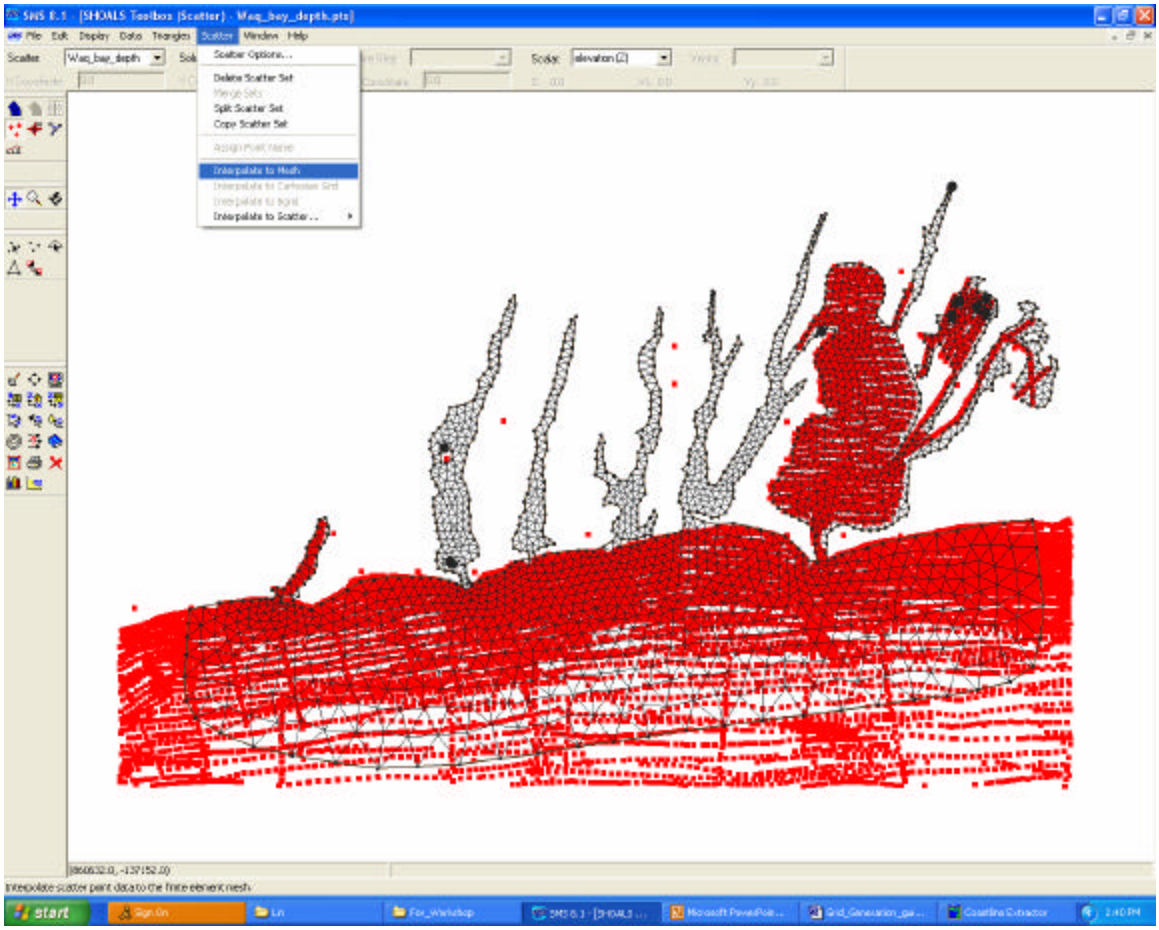
After you have finished this step, be sure to go to "File" to click "**Save Mesh**", so that the mesh file will be updated with the new numbering of the open boundary nodes. This numbering will facilitate specifying the open boundary forcing in later model setup stages.

9. Interpolation of bathymetric data into the mesh

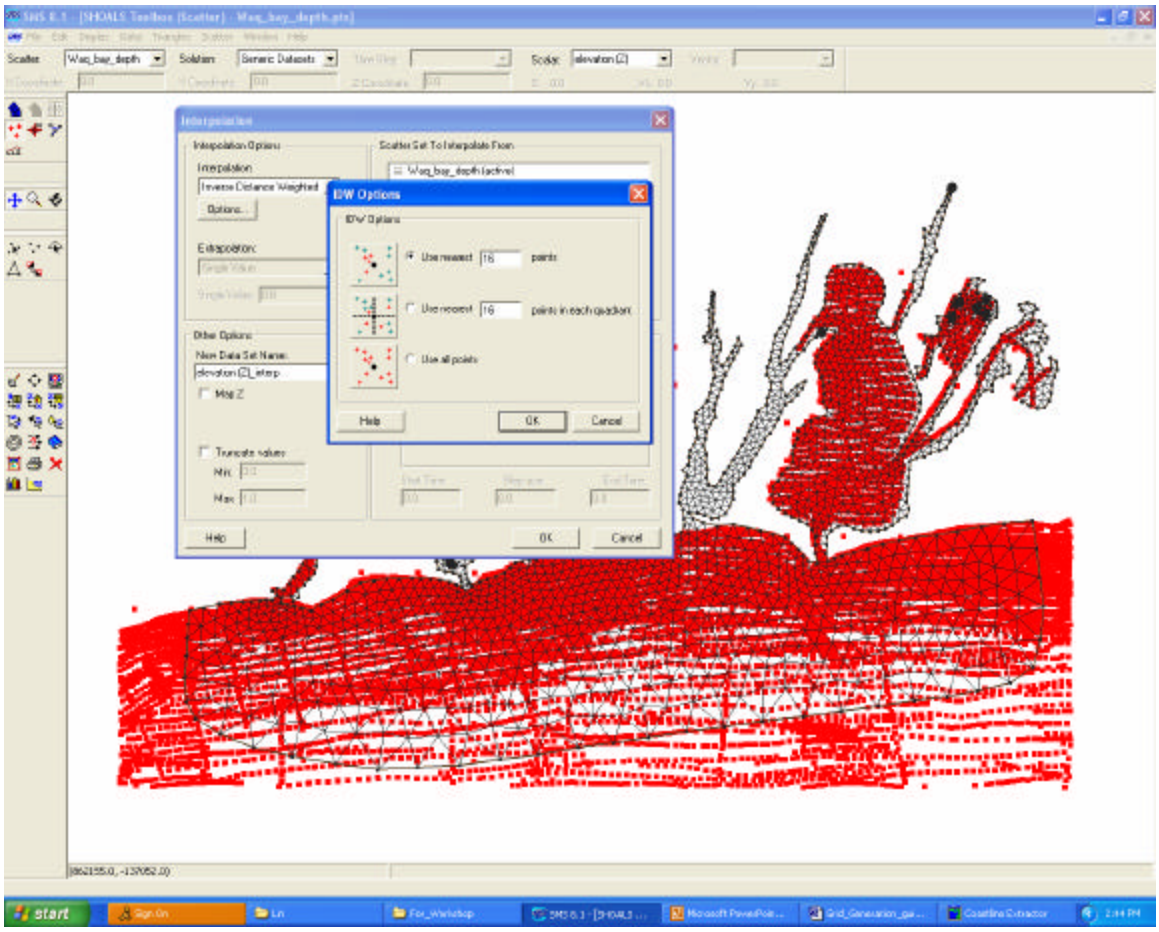
Step 1: Go to the upper manual to "File" and select "Open". Find the bathymetric file in your computer, then click it. Keep clicking "next" in the GUI selection. All the data will appear on the screen, overlapping over the mesh. An example is given below. Note: this is just used to demonstrate how we can interpolate the bathymetric data into the mesh, so we did not use the complete bathymetric data set here.



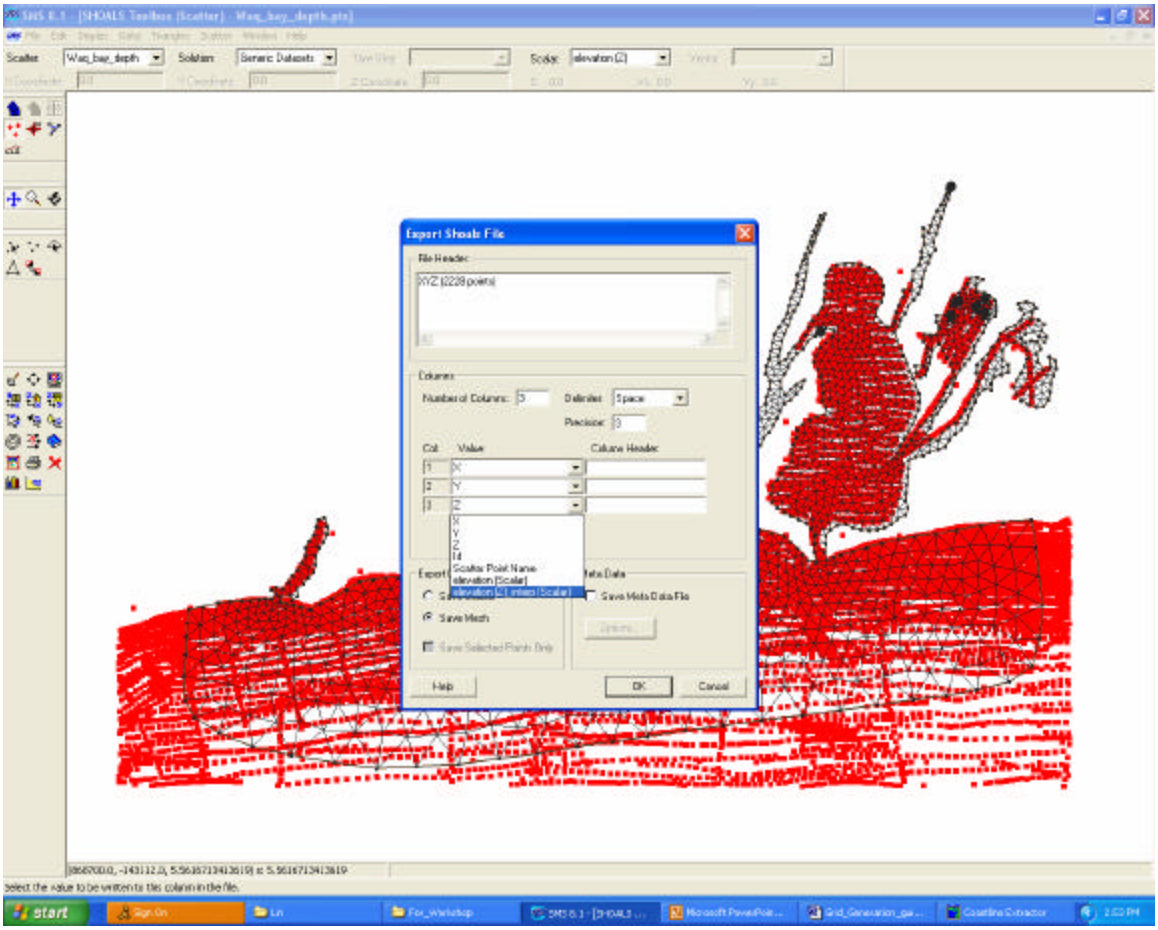
Step 2: Go to the upper manual bar, select "Scatter". In the sub-window manual, select "Interpolate to Mesh". See the screen below for example.


Scatter Options...
Delete Scatter Set
Split Scatter Set
Copy Scatter Set
Interpolate to Mesh
Interpolate to Scatter...
start

Step 3: Click "Interpolate to Mesh" You can select different interpolation methods in the box of "interpolation". After your make your selection, click "ok". The bathymetric data will be interpolated to nodes of the mesh.

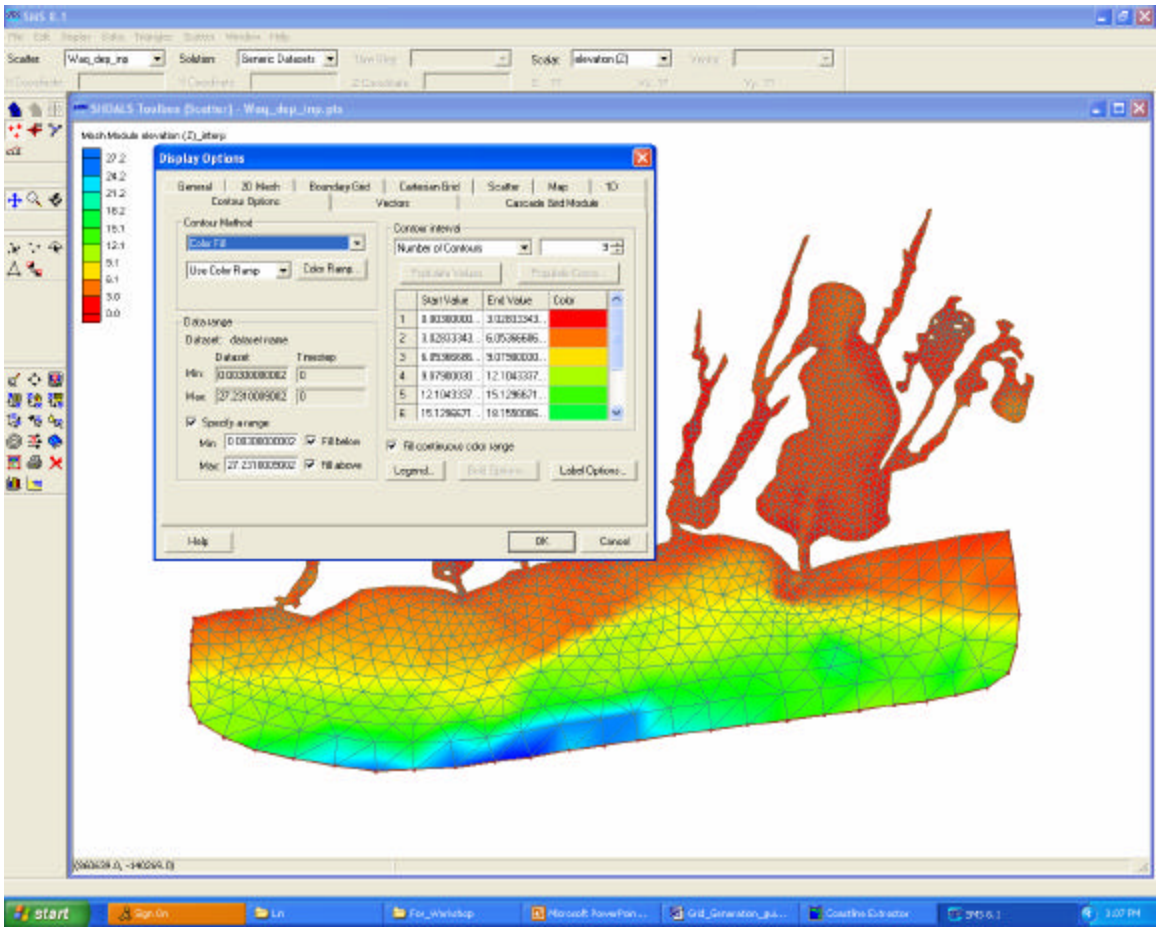


Step 4: Select "save as" from the file menu and create an output file name with the extension "pts" (example: myfile.pts). Click "Save", a sub-window titled "Export Shoal File" screen will appear. From here, you must do two things: 1) in "Export Option" check "Save Mesh"; and 2) go to "Column Header", select "Z" row and then choose the last option "elevation (Z) inter(Scalar)". After you complete these two steps, click "ok" and your bathymetric data file is saved.



10. Displaying the Mesh and Interpolated Bathymetry

Users can go to "Display Options" on the left manual bar to see different methods of displaying the mesh and interpolated bathymetry. Shown below is an example of the bathymetry image and mesh produced by selecting "2D Mesh" and "Contour Options".



Acknowledgements

The FVCOM development team at UMASS-D would like to thank the Georgia Sea Grant College Program for supporting the initial effort of the code development. We greatly appreciate all support and encouragement from Dr. Mac Rawson, Georgia Sea Grant Program Manager during the initial development years at University of Georgia. Without his support, it would have been impossible for FVCOM to have reached its current level. We also want to thank Dr. Brian Rothschild, Director of the School for Marine Sciences and Technology (SMAST), UMASS-D for providing significant personnel and equipment support and encouragement for FVCOM module development. The SMAST Fishery Program has been a key source of support for the model development and the primary motivation for the expansion of FVCOM's capabilities. Mr. Joe Deck, retired Deputy Director of SMAST, is a person we will always remember. His kind assistance in dealing with a multitude of administrational issues gave us the time and freedom to concentrate on model development. The FVCOM team also receives funding from NOAA, NSF, ONR and Sea Grants. These funds help the team apply FVCOM to realistic ocean applications. FVCOM has been developed in a team effort jointly by UMASS-D and WHOI. We want to express our thanks to all research staff in the MEDM/SMAST laboratory for their hard work and scientists at WHOI for their encouragement and support.

References

Ambrose Jr, R. B., T. A. Wool, and J. L. Martin, 1993. The water quality analysis simulation program, WASP5, Part A: Model documentation. U. S. Environmental Protection Agency, Athens, Georgia, 202 pp.

Anderson, L.A., A.R. Robinson and C.J. Lozano 2000. Physical and biological modeling in the Gulf Stream region: I. Data assimilation methodology. Deep-Sea Research I 47, 1787-1827.

Anderson, T.R. and P.J. le B. Williams 1999. A one-dimensional model of dissolved organic carbon cycling in the water column incorporating combined biological-photochemical decomposition. Global Biogeochemical Cycles 13, 337-349.

Austria, P. M. and A. A. Aldama. 1990. Adaptive mesh scheme for free surface flows with moving boundaries, p. 456-460. In G. Gambolati, A. Rinaldo, C. A. Brebbiam, W. G. Fray, and G. F. Pinder (eds.), Computational methods in Surface Hydrology, Springer-Verlag, New York.

Bergamasco, A., P. Malanotte-Rizzoli, W. C. Thacker and R. B. Long, 1993. The seasonal steady circulation of the Eastern Mediterranean determined with the adjoint method. *Deep-Sea Research*, 40, 1269–1298.

Bentzen, E., Taylor, W.D., Millard, E.S., 1992. The importance of dissolved organic phosphorus to phosphorus uptake by limnetic plankton. Limnol.&Oceanogr., 37: 217-231.

Bieman, V.J., Dolan, D.M., 1981. Modeling of phytoplankton-nutrient dynamics in Saginaw bay, Lake Huron. J. Great Lake Res., 7(4): 409-439.

Bissett, W.P., J.J. Walsh, D.A. Dieterle and K.L. Carter 1999. Carbon cycling in the upper waters of the Sargasso Sea: I. Numerical simulation of differential carbon and nitrogen fluxes. Deep-Sea Research I 46, 205-269.

Blanchet, I., C. Frankignoul and M. A. Cane, 1997. A comparison of adaptive Kalman filter for a Tropical Pacific Ocean model. *Monthly Weather Review*, 125, 40–58

Blumberg, A.F., Mellor, G.L., 1987. A description of a three-dimensional coastal ocean circulation model. In Three-dimensional Coastal Ocean Model, N.S. Heaps, Ed., Coastal. Estuar. Sci., **4**, 1-6.

Blumberg, A. F., A primer for Ecom-si. *Technical Report of HydroQual, Inc.*, 66 pp, 1994.

Bryant, A.D., M.Heath, W.S.G. Gurney, D.J. Beare and W. Robertson 1997. The seasonal dynamics of Calanus finmarchicus: development of a three-dimensional structured population model and application to the northern North Sea. Journal of Sea Research 38, 361-379.

Burchard, H., 2001. Simulating the wave-enhanced layer under breaking surface waves with two-equation turbulence models. *J. Phys. Oceanogr.*, 31, 3133-3145.

Burchard, H., 2002. Applied turbulence modeling in marine waters. *Springer:Berlin-Heidelberg-New York-Barcelona-Hong Kong-London-Milan Paris-Tokyo*, 215pp.

Burchard, H. and H. Baumert, 1995. On the performance of a mixed-layer model based on the $k-\varepsilon$ turbulence closure. J. Geophys. Res., 100 (C5), 8523-8540.

Canuto, V. M., A. Howard, Y. Cheng and M. S. Dubovikov, 2001. Ocean turbulence. Part I: one point closure model-momentum and heat vertical diffusivities. *J. Phys. Oceanogr.*, 31, 1413-1246.

Casulli, V. and E. Cattani. 1994. Stability, accuracy and efficiency of a semi-implicit method for three-dimensional shallow water flow. *Computers and Mathematics with Application*, 27: 99-112.

Caperon, J. and J. Meyer 1972. Nitrogen-limited growth of marine phytoplankton II. Uptake kinetics and their role in nutrient limited growth of phytoplankton. Deep-Sea Research 19, 619-632.

Carlotti, F., J. Giske and F. Werner 2000. Modeling zooplankton dynamics. Chapter 12, In: R.P. Harris, P.H. Wiebe, J. Lenz, H.R. Skjoldal and M. Huntley (eds.) *ICES Zooplankton Methodology Mannual.* Academic Press, pp. 571-667.

Carlson, C.A. and H.W. Ducklow 1995. Dissolved organic carbon in the upper ocean of the central equatorial Pacific Ocean, 1992: Daily and finescale vertical variations. Deep-Sea Research II 42, 639-656.

Casulli, V. and R. T. Cheng. 1991. A semi-implicit finite-difference model for three-dimensional tidal circulation, p. 620-631. In M. Spaulding et al. (eds.), *Proceeding of 2nd International Conference on Estuarine and Coastal Modeling*, ASCE, Tampa, Florida.

Casulli, V. and R. T. Cheng. 1992. Semi-implicit finite-difference methods for three-dimensional shallow-water flow. *International Journal for Numerical Methods in Fluids*, 15: 629-648.

Chapman, D.C., 1985. Numerical treatment of cross-shelf open boundaries in a barotropic coastal ocean model. *J. Phys. Oceanogr.*, 15, 1060-1075.

Chen, C., 1992. Variability of currents in Great South Channel and over Georges Bank. Ph.D Thesis, MIT/WHOI Joint Program, Technical Report WHOI-92-20, 283pp.

Chen, C., D. Wiesenburg, and L. Xie. 1997. Influences of river discharges on biological production over the inner shelf: a coupled biological and physical model of the Louisiana-Texas shelf. *Journal of Marine Research*, 55, 293-320.

Chen, C. R. Ji, L. Zheng, M. Zhu, and M. Rawson, 1999. Influences of physical processes on ecosystem in Jiaozhou Bay: A coupled physical and biological model experiment. *Journal of Geophysical Research*, 104 (C12), 29,925-29, 949.

Chen, C., J. Zhu, E. Ralph, S. A. Green, and J. Budd, 2001. Prognostic modeling studies of the Keweenaw current in Lake Superior. Part I: formation and evolution. *J. Phys. Oceanogr.*, 31, 379-395.

Chen, C. R. Ji, D. Schwab, D. Beletsky, D. Fahanenstial, M. Jiang, T. H. Johengen, H. Lavrentyev, B. Eadie, J. W. Budd, M. Bundy, W. Gardner, J Cotner, and P. J. Lavrentyer, 2002. A coupled biological and physical model study of the ecosystem in Lake Michigan Part I: A 1-D experiment. *Ecological Modeling*, 152, 145-168.

Chen, C. H. Liu, R. C. Beardsley, 2003a. An unstructured, finite-volume, three-dimensional, primitive equation ocean model: application to coastal ocean and estuaries. *J. Atm. &Oceanic Tech.*, 20, 159-186.

Chen, C, R. C. Beardsley, and P. J. S. Franks, 2003b. J. V. Keuren, Influences of the diurnally varying heat flux on stratification and residual circulation on Georges Bank. *J. Geophys. Res.*, 108(C11), 8008, DOI 10.1029/2001JC001245.

Chen, C. J. Zhu, L. Zheng, E. Ralph and J. W. Budd, 2004a. A non-orthogonal primitive equation coastal ocean circulation model: application to Lake Superior. *J. Great Lakes Res.*,30, 41-54.

Chen, C, G. Cowles and R. C. Beardsley, 2004b. An unstructured grid, finite-volume coastal ocean model: FVCOM User Manual. SMAST/UMASSD Technical Report-04-0601, pp183.

Chen, C, R. C. Beardsley, S. Hu, Q. Xu, and H. Lin, 2005. Using MM5 to hindcast the ocean surface forcing fields over the Gulf of Maine and Georges Bank region. Journal of Atmospheric and Oceanic Technology, 22(2), 131-145.

Chen, C., H. Huang, R. C. Beardsley, H. Liu, Q. Xu, and G. Cowles, 2006a. A finite-volume numerical approach for coastal ocean circulation studies: comparisons with finite-difference models. *Journal of Geophysical Research*, in press.

Chen, C., R. C. Beardsley, H. Huang. J. Qi, G. Cowles, G. Gao and H. Lin, 2006b. A spherical coordinate version of FVCOM: validation and application. Journal of Atmospheric and Ocean Technology, to be submitted.

Chen, C., J. Qi, R. C. Beardsley, H. Liu, H. Lin, and G. Cowles, 2006c. A 3-dimensional, unstructured grid, finite-volume wet/dry point treatment method for FVCOM. *Ocean Modeling*, submitted.

Chen, C, R. C. Beardsley and G. Cowles, 2006d. An unstructured grid, finite-volume coastal ocean model (FVCOM) system. Special Issue entitled "Advance in Computational Oceanography", *Oceanography*, 19(1), 78-89.

Cheng, R. T., V. Casulli, and J. W. Gartner. 1993. Tidal, residual, intertidal mudflat (TRIM) model and its applications to San Francisco Bay, California. *Estuarine*, *Coastal* and *Shelf Science*, 36: 235-280.

Clarke, A. 1987. Temperature, latitude and reproductive effort. Marine Ecology Progress Series 38, 89-99.

Claustre, H., Kerherve, P,, Marty, J.C., Prieur, L., Videau, C., Hecq, J-H. 1994. Phytoplankton dynamics associated with a geostrophic front: Ecological and biogeochemical implications. Journal of Marine Research 52, 711-742, 1994.

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

Cotner, J.B., Wetzel, R.G., 1992. Uptake of dissolved inorganic and organic phosphorus compounds by phytoplankton and bacterioplankton. *Limnol.&Oceanogr*., **37**: 232-243.

Cullen, J.J. 1993. Towards a general description of phytoplankton growth for biogeochemical models. In: G.T. Evans and M.J.R. FAsham (eds.) Towards a Model of Ocean Biogeochemical Processes. Springer-Verlag, N.Y., pp. 153-176.

Davis, C. S., 1987. Zooplankton life cycles. In Backus R. H. and D. W. Bourne, editors, *Georges Bank*, pages 256-267. MIT Press, Cambridge, Massachusetts.

Davis, A. M., J. E. Jones, and J. Xing. 1997. Review of recent developments in tidal hydrodynamic modeling, I: Spectral models. *J. Hydraulic Eng.*, ASCE, 123: 278-292.

Delille, D. and E. Perret 1989. Influence of temperature on the growth potential of Southern Polar Marine Bacteria. Microbial Ecology 18, 117-123.

Di Toro, D. M., D. J. O'Connor, and R. V. Thomann, 1971. A dynamic model of the phytoplankton population in the Sacramento-San Joaquin Delta. In: Nonequilibrium System in Natural Water Chemistry, Adv. Chem. Ser. 106, Amer. Chem. Soc., Washington, DC, 131-180.

Di Toro, D. M., J. J. Fitzpatrick, 1993. Chesapeake Bay sediment flux model. HydroQual, Inc., Mahwah, New Jesery.

Di Toro, D. M., W. F. Matystik, 1980. Mathematical models of water quality in large Lakes. Part I: Lake Huron and Saginaw Bay, EPA-600/3-80-056, 28-30.

Di Toro, D. M., J. P. Connolly, 1980. Mathematical models of water quality in large Lakes. Part II: Lake Erie, EPA-600/3-80-056, 90-101.

Donelan, M. A., 1990. Air-sea interaction. *The Sea*. B. LeNehaute and D. M. Hanes, Eds., *Ocean Engineering Science*, **9**, Wiley and Sons, 239-292.

Droop, M.R. 1983. 25 years of algal growth kinetics: A personal view. Botanica Marina 26, 99-112.

Edwards, A.M. and A. Yool 2000. The role of higher predation in plankton population models. Journal of Plankton Research 22, 1085-1112.

Evensen, G., 1992. Using the extended Kalman filter with a multilayer quasi-geostrophic ocean model. *J. Geophys. Res.*, 97(C11), 17905–19724.

Evensen, G., 1993, Open boundary condition for the extended Kalman filter with a quasi-geostrophic ocean model. *J. Geophys. Res.*, 98(C9), 16529–16546.

Fahnenstiel, G.L. and D. Scavia, 1987. Dynamics of Lake Michigan phytoplankton: Recent changes in surface and deep communities. *Can. J. Fish Aquat. Sci*., 44: 509-514.

Fasham, M.J.R , Duklow, H.W., Mckelvie, S.M., 1990. A nitrogen-based model of plankton dynamics in the oceanic mixed layer. *J. Mar. Res*., 48, 591-639.

Flather, R. A. and N. S. Heaps. 1975. Tidal computations for Morecambe Bay. *Geophys. J. Roy. Astr. Soc.*, 42:489-517.

Flato, G. M., 1993. A particle-in-cell sea-ice model. *Atmosphere-Ocean* 31, 339–358.

Flato, G. M., Hibler III, W. D., 1992. Modeling pack ice as a cavitating fluid. *J. Phys. Oceanogr*. 22, 626–651.

Foreman, M. G. G., 1978. Manual for tidal analysis and prediction. Pacific Marine Science Rep. 78-6, Institute of Ocean Sciences, Patrcia Bay, Sydney, British Columbia, Canada, 70pp.

Franks, P. J. S., J. S. Wroblewski and G. R. Flierl, 1986. Behavior of a simple plankton model with food-level acclimation by herbivores. *Mar. Biol.*, *91*, 121-129.

Franks, P. J. S. and C. Chen, 1996. Plankton production in tidal fronts: a model of Georges Bank in summer. *J. Mar. Res*., **54**, 631-651.

Franks, P. J. S., and C. Chen, 2001. A 3-D prognostic numerical model study of the Georges Bank ecosystem. Part II: biological-physical model. *Deep Sea Res*. II, **48**, 457-482.

Galperin, B., L. H. Kantha, S. Hassid, and A. Rosati, 1988. A quasi-equilibrium turbulent energy model for geophysical flows. *J. Atmos. Sci*., 45, 55-62.

Geider, R.J., MacIntyre, H.L., Kana, T.M. 1996. A dynamic model of photoadaptation in phytoplankton. Limnology and Oceanography 41, 1-15.

Geider, R.J., MacIntyre, H.L. and Kana, T.M. 1997. Dynamic model of phytoplankton growth and acclimation: responses of the balanced growth rate and the chlorophyll a:carbon ratio to light, nutrient-limitation and temperature. Marine Ecology Progress Series 148, 187-200.

Gismervik, I. and T. Andersen. 1997. Prey switching by *Acartia clausi*: experimental evidence and implications of intraguild predation assessed by a model. Marine Ecology Progress Series 157, 247-259.

Haidvogel, D. B, H. G. Arango, K. Hedstrom, A. Beckmann, P. M. Rizzoli, A. F. Schepetkin, 2000. Model evaluation experiments in the North Atlantic Basin, Simulation in nonlinear terrain-following coordinates. *Dyn. Atmo.Oceans*., 32, 239-281.

Hamilton, R.D., Preslon, J. E., 1970. Observations on the continuous culture of a planktonic phagotrophic protozoan. J. Exp. Mar. Biol., **5**: 94-104.

Haney, R. L., On the Pressure Gradient Force over Steep Topography in Sigma Coordinate Ocean Models. *J. Phys. Oceanogr.*, 21:610-619, 1991.

Hervouet, J. M. and J. M. Janin. 1994. Finite-element algorithms for modeling flood propagation, p. 102-113. In P. Molinaro and L. Natale (Eds.), *Modeling of Flood Propagation over Initially Dry Areas*, ASCE, New York

Hibler III, W. D., 1979. A dynamic thermodynamic sea ice model. *J. Phys. Oceanogr.* 9, 815–843.

Hubbard, M. E., Multidimensional slope limiters for MUSCL-type finite volume schemes on unstructured grids, *J. Comput. Phys*., 155, 54-74, 1999.

Hunk, E. C., J. K. Dukowicz, 1997. An elastic-viscous-plastic model for sea-ice dynam-dynamics. *J. Phys. Oceanogr*. 27, 1849–1867.

Janssen, P. A. E. M., 2001. Reply. *J. Phys. Oceanogr*., 31, 2532-2544.

Ip, J. T. C., D. R. Lynch, and C. T. Friedrichs. 1998. Simulation of estuarine flooding and dewatering with application to Great Bay, New Hampshire. *Estuarine, Coastal and Shelf Science*, 47:119-141.

Ji, R., 2003. Biological and physical processes controlling the spring phytoplankton bloom dynamics on Georges Bank. Ph.D. Thesis. The University of Georgia, 216pp.

Jorgensen, S.E., Nielsen, S.N., Jorge, L.A., 1991. Handbook of Ecological Parameters and Ecotoxicology. Elsevier.

Kantha, L.H. 2004. A general ecosystem model for applications to primary production and carbon cycle studies in the global oceans. Ocean Modelling 6, 285-334.

Kantha, L. and C. A. Clayson, 1994. An improved mixed layer model for geophysical applications. *J. Geophys. Res.*, 99 (C12), 25,235-25,266.

Kantha, L. and C. A. Clayson, 2000. Numerical models of oceans and oceanic processes. *International Geophysics*, 66, Academic Press, San Diego, CA 2000. 976pp.

Karypis, G. and Kumar, V. 1998. METIS* A Software Package for Partitioning Unstructured Graphs, partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 4.0. University of Minnesota, Department of Computer Science/ Army HPC Research Center Minneapolis.

Kawamiya, M.., M.. J. Kishi, Y. Yamanaka and N. Suginohara, 1995. An ecological-physical coupled model applied to station Papa. *Journal of Oceanography*, *51*, 635-664.

Keil, R.G., Kirchman, D.L., 1994. Abiotic transformation of labile protein to refractory protein in sea water. Marine Chemistry 45, 187–196.

Kobayashi, M. H., J. M. C. Pereira and J. C. F. Pereira, A conservative finite-volume second-order-accurate projection method on hybrid unstructured grids. *J. Comput. Phys*., 150, 40-45, 1999.

Kraus, E. B., 1972: Atmosphere-Ocean Interaction. *Clarendon Press*, 275pp.

Lancelot, C., J. Staneva, D. van Eeckhout, J.M. Beckers and E. Stanev 2002 Modelling the Danube influenced North-western continental shelf of the Black Sea II: Ecosystem

response to changes in nutrient delivery by the Danube River after its damming in 1972. Deep-Sea Research I 54, 473-499.

Landry, M.R., 1993. Predicting excretion rates of microzooplankton from carbon metabolism and elemental ratios. Limnology and Oceanography 38, 468–472.

Le Dimet, F., and O. Talagrand, 1986. Variational algorithm for analysis and assimilation of meteoro- logical observations: Theoretical aspects. *Tellus*, 38A, 97–110.

Leendertse, J. J. 1970. A water-quality simulation model for well-mixed estuaries and coastal seas: principles of computation, Rand Corporation Report RM-6230-RC, Vol. I.

Leendertse, J. J. 1987. Aspects of SIMSYS2D, a system for two-dimensional flow computation. Rand Corporation Report R-3572-USGS.

Lorenc, A. C., 1981. A global three-dimensional multi-variated statistical interpolation scheme. *Monthly Weather Review*, 109, 701–721.

Lynch, D. R. and W. G. Gray. 1980. Finite element simulation of flow in deforming regions. *J. Computational Phys.*, 36: 135-153

Lynch, D. R., and C. E. Naimie, 1993. The M_2 tide and its residual on the outer banks of the Gulf of Maine, *J. Phys. Oceanogr*., 23, 2222-2253.

Maday,Y. and A. T. Patera, 1988. Spectral element methods for the incompressible Navier-Stokes equations. In: State-of-the art surveys in computational mechanics, A. K. Noor, editor, ASME, New York.

Madin, L.P. and D. Deibel 1997. Feeding and energies of Thaliacea. In: Q. Bone (ed.) The Biology of Pelagic Tunicate, Oxford University Press, N.Y., pp. 81-103.

Mellor, G. L., T.Ezer, and Oey, L.-Y., The Pressure Gradient Conondrum of Sigma Coordinate Ocean Models. *J. Ocean. Atmos. Tech.*, 11:1126-1134, 1994.

Mellor, G. L. and T. Yamada, Development of a turbulence closure model for geophysical fluid problem. *Rev. Geophys. Space. Phys*., 20, 851-875, 1982.

Mellor, G. L. and A. Blumberg, 2004. Wave breaking and ocean surface layer thermal response. J. Phys. Oceanogr., 34, 693-698.

Morrow, R. A., and P. De Mey, 1995. Four-dimensional assimilation of altimetric and cruise data in the Azores current in 1992–1993. *J. Geophys. Res.*, 100(C12), 25,007–25,025.

Naimie, C. E., 1996. Georges Bank residual circulation during weak and strong stratification periods: prognostic numerical model results. *J. Geophys. Res*. 101(C3), 6469-6486.

Olson, R.J., 1981. 15N tracer studies of the primary nitrite maximum. Journal of Marine Research 39, 203–226.

Paasche, E. 1973. Silicon and the ecology of marine plankton diatoms. II. Silicate uptake kinetics of five diatom species. Marine Biology 19, 262-269.

Parker, R.A., 1993. Dynamic models for ammonium inhibition of nitrate uptake by phytoplankton. *Ecological Modelling* 66, 113-120.

Parkinson, C. L., W. M. Washington, 1979. A large-scale numerical model of sea ice. *J. Geophys. Res*. 84, 311–337.

Parsons, T.R., Takahashi, M., Hargrave, B., 1984. *Biological Oceanographic Processes*, 3rd ed. Pergamon Press.

Pedlosky, J., Longshore currents, upwelling and bottom topography. *J. Phys. Oceanogr.*, 4, 214-226, 1974.

Reed, W. H., and T. R. Hill, 1973. Triangular and methods for the neutron transport equation, Tech. Report LA-UR-73-479, Los Alamos Scientific Laboratory.

Peeters, J.C.H. and P. Eilers 1978. The relationship between light intensity and photosynthesis, a simple mathematical model. Hydrobiological Bulletin 12, 134-136.

Perovich, D. K., B. Elder, 2002. Sestimates of ocean heat flux at sheba. *Geophys.l Res. Let*. 29(9), 10.1029/2001GL014171.

Platt, T., C. L. Gallegos, and W. G. Harrison. 1980. Photoinhibition of photosynthesis in natural assemblages of marine phytoplankton. J. Mar. Res. 38, 687-701.

Redalje, D.G. and E.A. Laws. 1983. The effects of environmental factors on growth and the chemical and biochemical composition of marine diatoms I. light and temperature effects. Journal of Experimental Biology and Ecology 68, 59-79.

Rhee, G.Y. and I.J. Gotham 1981. The effect of environmental factors on phytoplankton growth: light and the interactions of light with nitrate limitation. Limnology and Oceanography 26, 649-659.

Rodi, W., 1980. Turbulence models and their application in hydraulics. Report., Int. Assoc. for Hydraul. Res., Delft, Netherlands.

Sayed, M., T. Carrieres, H.Tran, S. B. Savage, 2002. Development of an operational ice dynamics model for the Canadian Ice Service. In: Proc. of the 12th Int. Of Offshore and Polar Eng. Conf. Kitakyushu, Japan, pp. 841–848.

Scavia, D., Fahnenstiel, G.L., Evans, M.S., Jude, D.L., Lehman, J.T., 1986. Influence of salmonine predation and weather on long-term water quality trends in Lake Michigan. *Can. J. Fish. Aquat. Sci*., 43: 435-443.

Scavia, D. and G. L. Fahnenstiel, 1987. Dynamics of Lake Michigan phytoplankton: mechanisms controlling epilimnetic communities. J. Great Lake Res., 13(2): 103-120.

Scavia, D., Lang, G.A, Kitchell, J.F., 1988. Dynamics of Lake Michigan plankton: a model evaluation of nutrient loading, competition, and predation. *Can. J. Fish. Aquat. Sci*., 45: 165-177.

Semtner, A. J., 1976. A model for the thermodynamic growth of sea ice in numerical investigations of climate. *J. Phys. Oceanogr*. 6, 379–389.

Sidén, G. L. D. and D. R. Lynch. 1988. Wave equation hydrodynamics on deforming elements. *International J. Num. Meth.* in *Fluids*, 8: 1071-1093.

Simpson, J. J. and T. D. Dickey, 1981a: The relationship between downward irradiance and upper ocean structure. *J. Phys. Oceanogr*., 11, 309-323.

Simpson, J. J. and T. D. Dickey, 1981b: Alternative parameterizations of downward irradiance and their dynamical significance. *J. Phys. Oceanogr.*, 11, 876-882.

Smagorinsky, J., General circulation experiments with the primitive equations, I. The basic experiment. *Monthly Weather Review*, 91:99-164, 1963.

Smolarkiewicz, P. K., J. Szmelter, 2005. Multidimensional positive definite advection transport algorithm (MPD MPDATA): A): an edge-based unstructured-data formulation. *Int. J. for Num. Methods in Fluids* 47, 1293–1299.

Smith, S. D., R. J. Anderson, W. A. Oost, C. Kraan, N. Maat, J. DeCosmo, K. B. Katsaros, K. L. Davidson, K. Bumke, L. Hasse, and H. M. Chadwich, 1992. Sea Surface Wind Stress and Drag Coefficients: The HEXOS Results. *Boundary-Layer Meteorol.*, 60, 109-142.

Spitz, Y.H., Moisan, J.R. and Abbott, M.R. 2001. Configuring an ecosystem model using data from the Bermuda Atlantic Time Series (BATS). Deep-Sea Research Part II 48, 1733-1768.

Stacey, M. W., 1999. Simulation of the wind-forced near-surfacee circulation in Knight Inlet: a parameterization of the roughness length. J. Phys. Oceanogr., 29, 1363-1367.

Steele, J. H. and E. W. Henderson., 1992. The role of predation in plankton models. *J. Plankton Res*., *14*, 157-172.

Terray, E. A., M. A. Donelan, Y. C. Agrawal, W. M. Drnnan, K. K. Kahma, A. J. Williams, P. A. Hwang, and S. A. Kitaigorodski, 1996. Estimates of kinetic energy dissipation under breaking waves. *J. Phys. Oceanogr*., 26, 792-807.

Terray, E. A., M. A. Donelan, Y. C. Agrawal, W. M. Drnnan, K. K. Kahma, A. J. Williams, P. A. Hwang, and S. A. Kitaigorodski, 1997. Reply. *J. Phys. Oceanogr*., 27, 2308-2309.

Terray, E. A., W. M. Donelan, and M. A. Donelan, 2000. The vertical structure of shear and dissipation in the ocean surface layer. Proc. Symp. On Air-Sea Interaction Sdyney, Australia, University of New South Wales, 239-245.

Thacker, W. C. and R. B. Long, Fitting dynamics to data, 1988. *J. Geophys. Res.*, 93(C2), 1227–1240.

Thomann, R. V., D. M. Di. Toro, and D. J. O'Connor, 1974. Preliminary model of Potomac estuary phytoplankton, J. Env. Engr. Div. ASCE 100: 699-708.

Thomann, R. V. and J. J. Fitzpatrick, 1982. Calibration and verification of a mathematical model of the eutrophication of the Potomac estuary. HydroQual, Inc., Mahwah, New Jersey.

Tian, R.C., Vézina, A.F., Starr, M., Saucier, F., 2001. Seasonal dynamics of coastal ecosystems and export production at high latitudes: a modeling study. Limnology and Oceanography 46, 1845-1859.

Tilman, D., S. Kilham and P. Kilham, 1982. Phytoplankton community ecology: the role of limiting nutrients. *Annu. Rev. Ecol. Syst*., 13: 349-372.

Tremblay, B. and L. A. Mysak, 1997. Modeling sea ice as a granular material, including the dilatancy effect. *J. Phys. Oceanogr*. 27, 2342–2360.

Tziperman, E. and W. C. Thacker, 1989. An optimal-control/adjoint-equations approach to studying the oceanic general circulation. Jo*urnal of Physical Oceanography*, 19, 1471–1485.

Valiela, I., 1995. Marine Ecological Processes. Springer-Verlag New York. 686pp.

Wilcox, D, 2000. Turbulence modeling for CFD, DCW Industries, Inc. 540pp.

Winton, M., 2000. A reformulated three-layer sea ice model. *J. Atmos. Ocean Tech*. 17, 525–531.

Zhang, J. and W. D. Hibler III, 1997. On an efficient numerical method for modeling sea ice dynamics. *J. Geophys. Res.* 102, 8691–8702.

Zheng, L, C. Chen, and H. Liu, 2003a. A modeling study of the Satilla River Estuary, Georgia. Part I: flooding/drying process and water exchange over the salt marsh-estuary-shelf complex. *Estuaries*, 26 (3), 651-669.

Zheng, L. C. Chen, M. Alber, and H. Liu, 2003b. A modeling study of the Satilla River Estuary, Georgia. II: Suspended sediment. Estuaries, 26(3), 670-679.

Zheng, L. C. Chen, and F. Zhang, 2004. Development of water quality model in the Satilla River estuary, Georgia. *Ecological Modeling*, 178, 457-482.