View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by College of William & Mary: W&M Publish

W&M ScholarWorks

Reports

2016

Vertical One-dimensional (1-D) Simulations of Horizontal Velocity Profiles

Jerome P.Y. Maa

Jian Shen

Xiaoteng Shen

Shao Yuyang

Follow this and additional works at: https://scholarworks.wm.edu/reports

 Part of the Hydrology Commons, and the Marine Biology Commons

Recommended Citation

Maa, J. P., Shen, J., Shen, X., & Yuyang, S. (2016) Vertical One-dimensional (1-D) Simulations of Horizontal Velocity Profiles. Special scientific report (Virginia Institute of Marine Science); no. 156. Virginia Institute of Marine Science, William & Mary. https://doi.org/10.21220/V5RW2X

This Report is brought to you for free and open access by W&M ScholarWorks. It has been accepted for inclusion in Reports by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.

Vertical One-dimensional (1-D) Simulations of Horizontal Velocity Profiles

by

Jerome P.-Y. Maa, Jian Shen, Xiaoteng Shen

Virginia Institute of Marine Science
College of William and Mary
Gloucester Point, VA 23062

and

Yuyang Shao

Hohai University
Nanjing, China.

Special Scientific Report No. 156

January 2016

VIMS | WILLIAM & MARY
VIRGINIA INSTITUTE OF MARINE SCIENCE

Abstract:

Details of a vertical 1-D hydrodynamic model to simulate the horizontal velocity profiles for tidal estuarial flows with possible stratifications caused by salinity or Suspended Sediment Concentration (SSC) are presented. The standard 2nd order k-ε model was implemented to address the turbulent flow with possible stratification effects. Simulation results are verified with two field measurements for steady non-stratified flows and a field measurement for tidal estuary non-stratified flow. The stratification effect of salinity and suspended sediment concentration are also checked with the following descriptions: "Salinity stratification will change the typical logarithmic velocity profile to a linear profile for most of the water column." It appears that the possible high gradient of near-bottom (less than 0.5 m) SSC when the near-bed SSC is high only significantly alter the velocity profile when the turbulence is weak. The source codes, in FORTRAN 90, samples of the ASCII input data files, and a post process codes for plotting results using Matlab are attached for future uses.

Cite this report as:

Maa, Jerome P.Y.; Shen, Jian; Shen, Xiaoteng and Shao, Yuyang. 2015. Vertical One-dimensional (1-D) Simulations of Horizontal Velocity Profiles. Special Scientific Report No. 156. Virginia Institute of Marine Science, College of William & Mary. http://hdl.handle.net/10288/22091

Table of Contents Page

Table of Figures Page

Introduction

Simultaneous measurements of vertical profiles of flow velocity, Suspended Sediment Concentration (SSC), and salinity in estuaries are important because these data are necessary to better understand why the measured velocity profiles behave as that. Although three-dimensional (3-D) numerical models to simulate these hydrodynamic processes are available for studying the reason(s) of having the observed profiles, the cost is quite high, and also subjected to many limitations that make the use of a 3-D model impractical. On the other hand, a vertical one-dimensional (1-D) numerical hydrodynamic model is capable of simulating many observed profiles with a much better results and small cost.

Vertical 1-D models that are capable of simulating velocity profiles for steady flows, tidal flows, with or with stratification effects are needed for detail analysis of observed phenomena. Many models are available, for example, the General Ocean Turbulence Model (GOTM, Umlauf et al., 2000) developed in 1998 by two Europe scientists (M.R.Villarreal, Spain and P.-P. Mathieu, Italy), and later, with contributions from users. There is also a document available on web to explain what has been done and how to use the model. It is not easy, however, to clearly understand why and how this vertical 1-D model was implemented the way it did, especially for engineers who are not familiar with applications in oceanography. Since the governing equation for this vertical 1-D hydrodynamic model is relatively simple for coding, with the objective to examine other processes (e.g., to find the critical bed shear stress for deposition, Maa et al., 2008; to address the paradigm about erosion and deposition, Ha and Maa 2009; to estimate the settling velocity of fine sediments, Shao et al., 2011; and to study flocculation of suspended fine sediment, Shen and Maa, 2015) that are concurrently happened in this vertical 1-D system, a

vertical 1-D hydrodynamic model was first developed, verified, and documented in this report. The computer codes, written in FORTRAN 90, input data files, and post process codes, written in Matlab, to produce figures are all attached for future applications.

In this 1-D model to mimic the turbulent velocity profiles, the standard k-ε scheme was implemented. Other simplified eddy viscosity models, except the parabolic eddy viscosity model for non-stratified flows, were all omitted. A minor modification of the k-ε scheme was also included to handle the possible large shear strain near bed.

Governing Equation for Hydrodynamics

The simplified and linearized momentum equation that controls the vertical variation of horizontal velocity for turbulent flows in estuaries with the unsteady hydrostatic pressure assumption and possible stratification caused by salinity and suspended sediment is

$$\frac{\partial u(z,t)}{\partial t} = -\frac{1}{\rho(z)}\frac{\partial \rho_f g\eta(t)}{\partial x} + \frac{1}{\rho(z)}\frac{\partial}{\partial z}\left[\rho(z)\nu_t(z)\frac{\partial u}{\partial z}\right] \qquad (1)$$

where u is the time-averaged horizontal velocity, $g\rho_f \partial\eta/\partial x$ is the unsteady pressure gradient in x direction (i.e., the along channel direction), $\eta(t)$ is the fluctuation of water surface elevation, ρ_f is the density of water at surface, $\rho(z)$ is the local density of water which includes the effect of suspended sediment, salinity, and temperature, t is time, g is gravitational acceleration, υ_t is the turbulent eddy viscosity with a minimum equal to the molecular viscosity, z is the vertical coordinate with $z = 0$ at the rigid bottom and $z = h + \eta$ at the water surface, h is the mean water depth. The time-averaged variables described in this study have an average duration much

longer than the time scale for turbulence fluctuation, but still much smaller than the variation of applied forces, e.g., tidal periods. The boundary conditions are as follows:

1. At $z = 0$, $u = 0$.

2. Zero shear stress (no wind) at the free water surface, *i.e.*, $du/dz = 0$ at $z = h + \eta$.

3. The initial condition at $t = 0$ is given, or $u = 0$ if unknown.

The no-slip boundary condition used in this study indicates that this numerical model should have a high resolution grid in order to have several cells within the viscous sub-layer. This high resolution (e.g. less than 1 mm) does not necessary had the correct simulation results on turbulent velocity profiles. In this study, an attempt to address this problem is given later in the section of model verification.

For this vertical 1-D simulation, the information of $\partial\eta/\partial x$ required to solve Eq. 1 is usually not available. If data for the vertical profile of horizontal velocity, e.g., $u(z)$, are available for a steady flow, then it is possible to use the trial-and-error approach, i.e., using the simulated results to find the most possible $\partial\eta/\partial x$. Using the measured velocity profile and an empirical formulation to find shear velocity, u_*, and the energy slope is also possible, but that is just another type of trial-and-error approach. Because many uncertainty involved in the estimation of u_*, this 2nd approach is not used.

For tidal flows, it is much easier to request the time series of η at the tidal current measurement site. Thus, a change of $\partial\eta/\partial x$ in Eq. 1 to $\partial\eta/\partial t$ is a convenient option. Assuming $\eta(x) = \eta_0 e^{-\beta x} e^{-i(kx - \sigma t)}$, where η_0 is the tidal amplitude at $x = 0$, β is the tidal amplitude attenuation rate in the x direction, $i = (-1)^{1/2}$, $k = 2\pi/L$ is the tidal wave number, L is the tidal wave length, $\sigma = 2\pi/T$ and T is the tidal period. It can be shown that $\partial\eta/\partial x = \Psi\partial\eta/\partial t$, where $\Psi =$

$-1/C + i\beta/\sigma$ and $C = \sigma/k$ is tidal phase velocity. This implies that a phase lag is involved after the change from $\partial\eta/\partial x$ to $\partial\eta/\partial t$. The phase lag, $\tan^{-1}(-C\beta/\sigma)$, caused by the attenuation of tidal amplitude may be changed by channel geographic, bathymetric, and bottom friction. Since tidal waves are shallow water waves, $C = [g(h + \eta)]^{1/2}$.

The continuity equation (Eq. 2) must be satisfied at all places and all times

$$\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0 \qquad (2)$$

where w is the time averaged vertical velocity. With the bottom boundary condition $w(z = 0) = 0$, Eq. 2 can be used to estimate w at any elevation, z, in the water column as

$$w(z) = -\int_0^z \frac{\partial u}{\partial x} dz = -\Phi \int_0^z \frac{\partial u}{\partial t} dz \qquad (3)$$

The change of $\partial u/\partial x$ to $\partial u/\partial t$ is necessary because no $\partial u/\partial x$ information in this 1-D model. For periodic tidal flows, the tidal velocity can be described as $u(x, t) = u_0 e^{-\alpha x} e^{-i(kx - \sigma t)}$, where u_0 is the velocity amplitude at the beginning of the tidal channel where $x = 0$, α is a positive number that represents the attenuation rate of tidal current. Thus, there is a relationship as $\partial u/\partial x = \Phi \partial u/\partial t$, where $\Phi = -1/C + i\alpha/\sigma$,. For steady flows, Eq. 3 clearly shows that $w = 0$ in the entire water column.

In any estuary, fresh water discharge will provide a pressure gradient to push water moving toward downstream direction, and thus, the maximum ebb flow may become larger than that of maximum flood. To include this freshwater induced pressure force, a constant pressure gradient can be added to the $\partial\eta/\partial x$ term. Because the freshwater discharge may vary with time,

this added pressure gradient may also change with time, but in a much slow rate when compared with that of tidal flows. For this reason, a constant may be used, at least at a particular time.

Governing Equation for Turbulence

Although there are zero order, first-order, and one-parameter models approximations for eddy viscosity (Rodi, 1993; Wilcox, 2006), they are not used because a better second-order model already exists. The only exception is the use of the parabolic eddy viscosity model for a non-stratified, steady, turbulent flows (Henderson, 1966). The only purposes is to show that this simple parabolic eddy viscosity profile is an exceptional simple model.

The second-order, k-ε model given by Rodi (1993) was used. Turbulent eddy viscosity can be determined as $\mu_t = \rho\nu_t = \rho C_\mu k^2/\varepsilon$ where k is the total turbulent kinetic energy, $k = (u'^2 + v'^2 + w'^2)/2$, ε is the turbulent energy dissipation, and $C_\mu = 0.09$. Notice however, C_μ may not be a constant for flows with a high strain rate (Shih *et al.*, 1994). They said "…this model will become non-reliable in the case of large mean strain rate (e.g., $Sk/\varepsilon > 3.7$ …" where $S = (2S_{i,j}S_{i,j})^{1/2}$ and $S_{i,j} = (\partial u_i/\partial x_j + \partial u_j/\partial x_i)/2$. They suggested that C_μ will decrease to a minimum of 0.05 when $Sk/\varepsilon = 6$. In this vertical 1-D modeling, $S_{i,j} = 1/2(\partial u/\partial z)$ and their suggestion was implemented in the codes using a simple linear decreasing function between $Sk/\varepsilon = 3.7$ and 6.

The generation, transport, and distribution of k and ε are described by the following two equations (e.g., Rodi, 1993; Wilcox, 2006).

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho w k}{\partial z} = \frac{\partial}{\partial z}\left[\frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial z}\right] + P + G - \rho\varepsilon \qquad (4)$$

$$\frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial \rho w \varepsilon}{\partial z} = \frac{\partial}{\partial z}\left[\frac{\mu_t}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial z}\right] + C_{1\varepsilon}\frac{\varepsilon}{k}(P + C_{3\varepsilon}G) - C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} \quad (5)$$

where $P = \rho\nu_t(\frac{\partial u}{\partial z})^2$ is the production of turbulent kinetic energy, $G = \frac{g}{\rho}\frac{\mu_t}{P_r}\frac{\partial \rho}{\partial z} = \frac{\mu_t}{P_r}N^2$ is the buoyancy effect term, N is the Brunt–Väisälä frequency, $C_{1\varepsilon} = 1.44$, $C_{2\varepsilon} = 1.92$, $\sigma_k = 1$, $\sigma_\varepsilon = 1.3$ are constants, and Pr is the Prandtl Number which is the ratio of momentum eddy diffusivity and the mass/heat transfer eddy diffusivity. In general, Pr = 0.89 was suggested (Wilcox, 2006). For the coefficient $C_{3\varepsilon}$, Rodi (1980) suggested a range between 0 and 0.3. Uittenbogaard et al. (1992) have claimed that the buoyancy effect for turbulence dissipation is negligible, i.e., $C_{3\varepsilon} = 0$. This zero $C_{3\varepsilon}$ was also used by Toorman (2002) with satisfied results. For this reason, it was used in this study.

Boundary conditions of k and ε near the bottom (marked with the subscript b) are specified at an elevation, z_b, about the rigid bottom. In general, z_b is above the viscous sub-layer for a smooth bottom or at a small distance from the mean rigid bed elevation for rough beds. The bottom boundary conditions are $k_b = u_*^2/(C_\mu)^{1/2}$ and $\varepsilon_b = u_*^3/\kappa z_b$ where u_* is the shear velocity and κ is the von Karman constant. Boundary conditions at free surface (marked with the subscript s) are

$$k_s = u_{*s}^2/(C_\mu)^{1/2} \quad (6)$$

and

$$\varepsilon_s = \frac{(k_s \sqrt{C_\mu})^{3/2}}{\kappa[h + ah(1 - u_{*s}{}^2 / k_s \sqrt{C_\mu})]} \tag{7}$$

where a = 0.07 is a constant, h is the water depth, u_{*s} is the shear velocity at the surface. Obviously, if there is no shear force at the free surface, then $u_{*s} = k_s = \varepsilon_s = 0$.

The density gradient in the vertical direction caused by salinity, temperature, or/and suspended sediment difference in the water column, if exist, will produce negative buoyancy force and suppress turbulence if the density gradient is sufficiently large. In practice, the local Richardson Number, $R_i = [(g/\rho)(\partial\rho/\partial z)/(du/dz)^2]$, is used to determine the importance of buoyancy force, and when R_i is larger than 1/4, the shear force will have difficulty to overcome the buoyancy force and the flow will be stratified (Turner, 1973).

Vertical Profiles of Salinity and SSC Distributions

Because this study is a vertical 1-D model, the time history of salinity and SSC profiles for the entire simulation period must be given. These profiles serve the purpose for modifying eddy viscosity when buoyancy force (G in Eqs. 5 and 6) existed. The salinity and SSC profiles at each time step must be interpolated from the given salinity and SSC profiles.

Numerical formulations

In this modeling, u, ρ, ν_t, κ, and ε are all specified at the center of a cell, only w is specified at the bottom side of a cell (Fig. 1). Totally there are N - 1 cells to represent the entire water column.

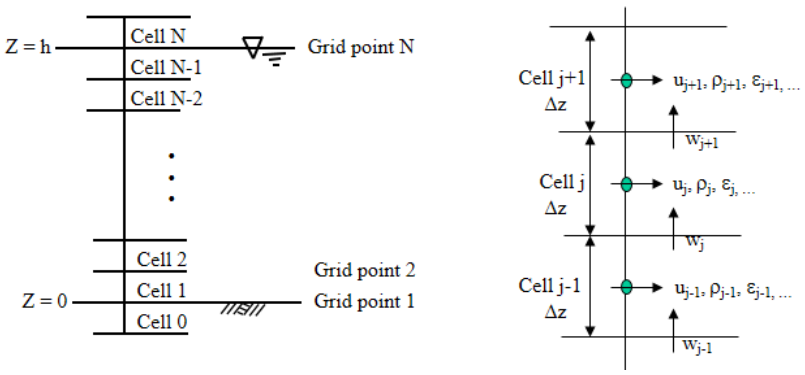

Fig. 1. The grid and variables specified in this vertical 1-D hydrodynamic model. Δz for each cell is the same, but it may change slightly with time during a tidal cycle, in order to have (N-1) Δz = h+η.

Horizontal Velocity: The Crank-Nicolson scheme (Smith, 1978) with the implicit factor, θ, was used to formulate the implicit finite difference form of Eq. 1 as follows.

$$a_{j-1}u_{j-1}^{k+1} + a_j u_j^{k+1} + a_{j+1}u_{j+1}^{k+1} = \frac{\rho_f g}{\rho_j}\frac{\partial \eta(t)}{\partial x}\Delta t + b_{j+1}u_{j+1}^{k} + b_j u_j^{k} + b_{j-1}u_{j-1}^{k} \tag{8}$$

where k and k+1 are two consecutive time steps, u_{j-1},u_j, u_{j+1} are horizontal velocity component specified at the center of three consecutive cells, j-1, j, and j+1, respectively, a_j and b_j are coefficients given next

$$a_{j-1} = -\nu_{t,j}\frac{\theta\Delta t}{\Delta z^2} + (\rho_{j+1}\nu_{t,j+1} - \rho_{j+1}\nu_{t,j-1})\frac{\theta\Delta t}{4\rho_j\Delta z^2} \tag{9a}$$

$$a_j = 1 + 2\nu_{t,j}\frac{\theta\Delta t}{\Delta z^2} \tag{9b}$$

$$a_{j+1} = -\nu_{t,j}\frac{\theta\Delta t}{\Delta z^2} - (\rho_{j+1}\nu_{t,j+1} - \rho_{j-1}\nu_{t,j-1})\frac{\theta\Delta t}{4\rho_j\Delta z^2} \tag{9c}$$

$$b_{j-1} = \nu_{t,j}\frac{(1-\theta)\Delta t}{\Delta z^2} - (\rho_{j+1}\nu_{t,j+1} - \rho_{j-1}\nu_{t,j-1})\frac{(1-\theta)\Delta t}{4\rho_j\Delta z^2} \tag{9d}$$

$$b_j = 1 - 2\nu_{t,j}\frac{(1-\theta)\Delta t}{\Delta z^2} \tag{9e}$$

$$b_{j+1} = \nu_{t,j}\frac{(1-\theta)\Delta t}{\Delta z^2} + (\rho_{j+1}\nu_{t,j+1} - \rho_{j-1}\nu_{t,j-1})\frac{(1-\theta)\Delta t}{4\rho_j \Delta z^2} \tag{9f}$$

where the eddy viscosity, ν_t, specified in the above equations are all at time step k. When applying Eq. 8 at the top cell, j = n -1, and using the top boundary condition, $u_n = u_{n-1}$, one obtain the following

$$a_{n-2}u_{n-2}^{k+1} + (a_n + a_{n-1})u_{n-1}^{k+1} = \frac{\rho_f g}{\rho_{n-1}}\frac{\partial \eta(t)}{\partial x}\Delta t + b_{n-2}u_{n-2}^{k} + (b_n + b_{n-1})u_{n-1}^{k} \tag{10}$$

When applying Eq. 8 to the bottom cell, j = 1, the velocity at cell 0 is u_0 and it can be specified as $-u_1$, and thus, it is a known and can be moved to the right hand side of Eq. 8 and rewritten as

$$(a_1 - a_0)u_1^{k+1} + a_2 u_2^{k+1} = \frac{\rho_f g}{\rho_1}\frac{\partial \eta(t)}{\partial x}\Delta t + b_2 u_2^k + b_1 u_1^k - b_0 u_1^k \tag{11}$$

Combine Eqs. 11, 8 and 10, a tri-diagonal matrix can be established as **au = b** where **a** is a 3 x (N-1) matrix that contains all the coefficients given in the left-hand side of Eqs. 11, 8 and 10, **u** is a column matrix for the velocity at time level k+1 and **b** is also a column matrix that contains all the right-hand terms given in Eqs. 11, 8, and 10. This matrix equation was solved by using the traditional tri-diagonal matrix solver (Hildebrand, 1965). Notice that the tri-diagonal matrix equation can only be built if eddy viscosity information is available. The followings are details on solving the k-ε equation for eddy viscosity.

Turbulent kinetic energy: The same Crank-Nicolson scheme (Smith, 1978) was used to formulate an implicit finite difference form of Eq. 4 as follows.

$$p_{j-1}k_{j-1}^{k+1} + p_j k_j^{k+1} + p_{j+1}k_{j+1}^{k+1} = q_{j-1}k_{j-1}^{k} + q_j k_j^{k} + q_{j+1}k_{j+1}^{k} + R_j \tag{12}$$

where

$$p_{j-1} = -m_1\rho_j w_j + \frac{m_3}{4\sigma_k}(\rho_{j+1}\nu_{t,j+1} - \rho_{j-1}\nu_{t,j-1}) - \frac{m_3}{\sigma_k}\rho_j\nu_{t,j} \tag{13a}$$

$$p_j = \rho_j + \frac{2m_3}{\sigma_k}\rho_j\nu_{t,j} + m_1(\rho_{j+1}w_{j+1} - \rho_{j-1}w_{j-1}) \tag{13b}$$

$$p_{j+1} = m_1\rho_j w_j - \frac{m_3}{4\sigma_k}(\rho_{j+1}\nu_{t,j+1} - \rho_{j-1}\nu_{t,j-1}) - \frac{m_3}{\sigma_k}\rho_j\nu_{t,j} \tag{13c}$$

$$q_{j-1} = m_2\rho_j w_j - \frac{m_4}{4\sigma_k}(\rho_{j+1}\nu_{t,j+1} - \rho_{j-1}\nu_{t,j-1}) + \frac{m_4}{\sigma_k}\rho_j\nu_{t,j} \tag{13d}$$

$$q_j = \rho_j - \frac{2m_4}{\sigma_k}\rho_j\nu_{t,j} - m_2(\rho_{j+1}w_{j+1} - \rho_{j-1}w_{j-1}) \tag{13e}$$

$$q_{j+1} = -m_2\rho_j w_j + \frac{m_4}{4\sigma_k}(\rho_{j+1}\nu_{j+1} - \rho_{j-1}\nu_{j-1}) + \frac{m_4}{\sigma_k}\rho_j\nu_j \tag{13f}$$

$$R_j = \rho_j\nu_{t,j}\Delta t\left[\frac{u_{j+1}^k - u_{j-1}^k}{2\Delta z}\right]^2 + g\Delta t\frac{\nu_{t,j}^k}{P_r}\frac{\rho_{j+1}^k - \rho_{j-1}^k}{2\Delta z} - \rho_j\varepsilon_j^k\Delta t \tag{13g}$$

$$m_1 = \frac{\theta\Delta t}{2\Delta z},\ m_2 = \frac{(1-\theta)\Delta t}{2\Delta z},\ m_3 = \frac{\theta\Delta t}{\Delta z^2} \text{ and } m_4 = \frac{(1-\theta)\Delta t}{\Delta z^2}$$

When applying Eq. 12 at the surface cell (cell number j = n-1) with no wind stress, k at the water surface should be zero. This can be achieved by assuming there is a fictional layer above the free surface with the same layer thickness and $k_n = -k_{n-1}$. Also the density and velocity for the fictional layer (layer n) were assumed as the same as that for layer n-1. Thus, Eq. 12 becomes

$$p_{n-2}k_{n-2}^{k+1} + (p_{n-1} - p_n)k_{n-1}^{k+1} = q_{n-2}k_{n-2}^{k} + q_{n-1}k_{n-1}^{k} - q_n k_{n-1}^{k} + R_{n-1} \tag{14}$$

The bottom boundary condition for k is $k_b = u_*^2/C_\mu^{1/2}$. Notice that this condition is specified at an elevation z_b above the bed with a corresponding cell number b.

For implementing this boundary condition, near bed velocities, i.e., located from 0.1 m to 0.8 m above the bed, were used to find u_*, assuming the log velocity profile, $u = u^*/\chi \log(z/z_0)$, can be used in this range, where $\chi = 0.4$ is the von Karman Constant. For all the cells that were below cell b, the k values were linearly interpreted between 0 and k_b with $k = 0$ at $z = 0$. If applying Eq. 12 to cell b+1, then it becomes

$$p_{b+1}k_{b+1}^{k+1} + p_{b+2}k_{b+2}^{k+1} = (q_b - p_b)k_b + q_{b+1}k_{b+1}^{k} + q_{b+2}k_{b+2}^{k} + R_{b+1} \quad (15)$$

Equation 15, 12 and 14 were used to build a tri-diagonal matrix equation, $\mathbf{p}\boldsymbol{k} = \mathbf{q}$, where $\boldsymbol{k}$ is a column matrix includes k_{b+1}, k_{b+2}, …, k_{n-1} for the next time step, $\mathbf{p}$ is a tri-diagonal matrix that includes all these coefficients on the left-hand side of Eqs. 15, 12, and 14 and $\mathbf{q}$ is a column matrix that includes all terms on the right-hand side of Eqs. 15, 12, and 14. Again, this matrix equation was solved by using the tri-diagonal matrix equation solver.

Dissipation rate: Similarly, the finite difference equation for ε is

$$S_{j-1}\varepsilon_{j-1}^{k+1} + S_j\varepsilon_j^{k+1} + S_{j+1}\varepsilon_{j+1}^{k+1} = T_{j-1}\varepsilon_{j-1}^{k} + T_j\varepsilon_j^{k} + T_{j+1}\varepsilon_{j+1}^{k} \quad (16)$$

where

$$S_{j-1} = -m_1\rho_j w_j + \frac{m_3}{4\sigma_\varepsilon}(\rho_{j+1}\nu_{t,j+1} - \rho_{j-1}\nu_{t,j-1}) - \frac{m_3}{\sigma_\varepsilon}\rho_j\nu_{t,j} \quad (17a)$$

$$S_j = \rho_j + \frac{2m_3}{\sigma_\varepsilon}\rho_j \nu_{t,j} + m_1(\rho_{j+1}w_{j+1} - \rho_{j-1}w_{j-1}) + c_{2\varepsilon}\rho_j \Delta t \varepsilon_j^k / k_j - c_{1\varepsilon}\theta\, \Delta t N_j / k_j$$

$$S_{j+1} = m_1\rho_j w_j - \frac{m_3}{4\sigma_\varepsilon}(\rho_{j+1}\nu_{t,j+1} - \rho_{j-1}\nu_{t,j-1}) - \frac{m_3}{\sigma_\varepsilon}\rho_j \nu_{t,j} \quad (17c)$$

$$T_{j-1} = m_2\rho_j w_j - \frac{m_4}{4\sigma_\varepsilon}(\rho_{j+1}\nu_{t,j+1} - \rho_{j-1}\nu_{t,j-1}) + \frac{m_4}{\sigma_\varepsilon}\rho_j \nu_{t,j} \quad (17d)$$

$$T_j = \rho_j - m_2(\rho_{j+1}w_{j+1} - \rho_{j-1}w_{j-1}) - \frac{2m_4}{\sigma_\varepsilon}\rho_j \nu_{t,j} + c_{1\varepsilon}(1-\theta)\Delta t N_j / k_j \quad (17e)$$

$$T_{j+1} = -m_2\rho_j w_j + \frac{m_4}{4\sigma_\varepsilon}(\rho_{j+1}\nu_{t,j+1} - \rho_{j-1}\nu_{t,j-1}) + \frac{m_4}{\sigma_\varepsilon}\rho_j \nu_{t,j} \quad (17f)$$

and

$$N_j = \left[\rho_j \nu_{t,j}\left(\frac{u_{j+1} - u_{j-1}}{2\Delta z}\right)^2 - c_{3\varepsilon} g \frac{\nu_{t,j}}{P_r}\frac{\rho_{j+1} - \rho_{j-1}}{2\Delta z}\right] \quad (17g)$$

When applying Eq. 16 at the surface cell (cell number j = n-1) with no wind stress, ε at the water surface should be zero. This can be achieved by letting $\varepsilon_n = -\varepsilon_{n-1}$. Also because the gradient of eddy viscosity, the production and buoyant terms are all negligible small or zero at the surface, these terms were neglected. Thus, Eq. 16 becomes

$$S_{n-2}\varepsilon_{n-2}^{k+1} + (S_{n-1} - S_n)\varepsilon_{n-1}^{k+1} = T_{n-2}\varepsilon_{n-2}^k + (T_{n-1} - T_n)\varepsilon_{n-1}^k \quad (18)$$

When applying Eq. 16 at the cell b+1 and with the boundary condition $\varepsilon_b = u_*^3/\kappa z_b$, it becomes

$$S_{b+1}\varepsilon_{b+1}^{k+1} + S_{b+2}\varepsilon_{b+2}^{k+1} = (T_b - S_b)\varepsilon_b + T_{b+1}\varepsilon_{b+1}^k + T_{b+2}\varepsilon_{b+2}^k \quad (19)$$

Again, Eqs. 19, 16 and 18 were used to build a matrix equation $\mathbf{S\varepsilon} = \mathbf{T}$ for solving the $\boldsymbol{\varepsilon}$ at the next time step, where $\boldsymbol{\varepsilon}$ is a column matrix includes ε_{b+1}, ε_{b+2}, …, ε_{n-1} for the next time step, **S** is

a tri-diagonal matrix that includes all these coefficients on the left-hand side of Eqs. 19, 16, and 18 and **T** is a column matrix that includes all terms on the right-hand side of Eqs. 19, 16, and 18. Notice however, because the denominator *k* would be zero at the beginning of a simulation if started with a zero velocity profile, the term ε_j/k_j should be set to zero because it has no meaning when $k_j = 0$. The ε terms below cell b were also determined by linear interpretation between 0 and ε_b.

Vertical velocity, w: The vertical profile of w can be estimated based on the continuity equation (Eq. 3). The finite difference formulation is given next.

$$w_J = \Phi \sum_{j=1}^{J} (u_j^{k+1} - u_j^k) \Delta z / \Delta t \qquad (20)$$

where w_J is the vertical velocity at cell number J, and the boundary condition is $w_1 = 0$. Although the exact value of Φ is unknown *in prior*, but it may be assumed that it is the same as ψ.

In order to have correct tidal phase speed, the total water depth, i.e., h + η, shall be used instead of a fixed water depth. Since the time series of tidal elevation is given as input for this 1-D model, the new η is used to change the total water depth by changing Δz, but the total number of cells remains the same. Because of the large number of cells in z direction (more than 200), the change of Δz is rather small and usually do not cause stability problem with the selected Δt.

Model Validations

The following cases were selected to verify the modeling results: (1) Steady current, (2)Tidal current, (3) stratified flow caused by salinity, and (4) stratified flow caused by suspended sediment concentration.

Steady flows. For checking this 1-D model, steady open channel flows without stratification were first used. For the first verification, the simulated velocity profile by using the parabolic distribution of the eddy viscosity profile were compared with that using the k - ε model for eddy viscosity. The result (Fig. 2) indicates a small difference in the eddy viscosity profiles. This result also demonstrates that the parabolic distribution of eddy viscosity is a very good approximation for steady, non-stratified flows. The small difference in velocity profile mainly caused by the small difference of eddy viscosity near the bottom.

The model was then checked with field measurements for steady open channel flows without stratification. The SSC is also low such that the influence of SSC is neglected. Two data sets published by Gonzalez et al. (1996) were used. They used a 1200 kHz ADCP to measure the vertical profiles of horizontal velocity at the center of the Chicago Sanitary and Ship Canal. The two cases they presented have water depths of 7.45 m and 8.23 m, respectively. This canal has a rocky bottom and the width of this canal (about 49 m) is more than 5 times of the canal depth, so that this 1-D model is suitable of simulating the velocity profile at the canal center without the effect of side boundary. Although they provided estimated energy slopes

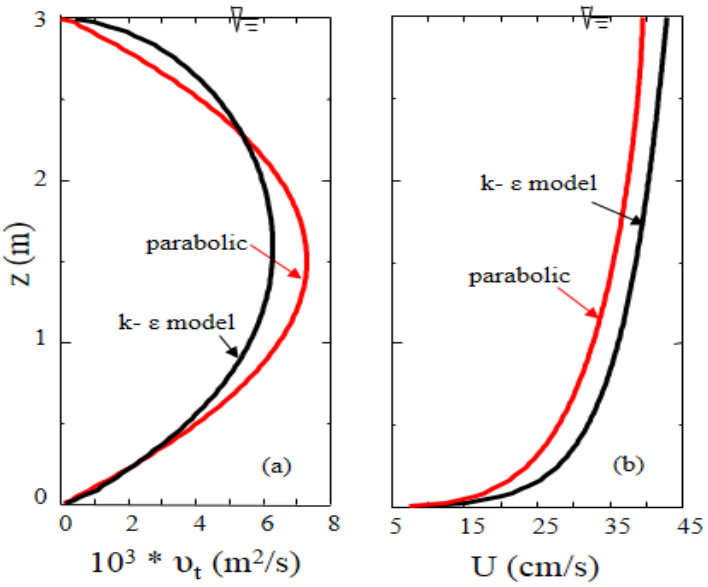

Fig. 2. Comparison of simulated steady turbulent flows without stratification. (a) Eddy viscosity profile from the parabolic eddy viscosity model versus the k-ε model. (b)The simulated velocity profiles. $d\eta/dx = -0.00002$ and $\Delta z = 0.02$ m. The only reason of selecting a negative pressure gradient is to produce positive horizontal velocities.

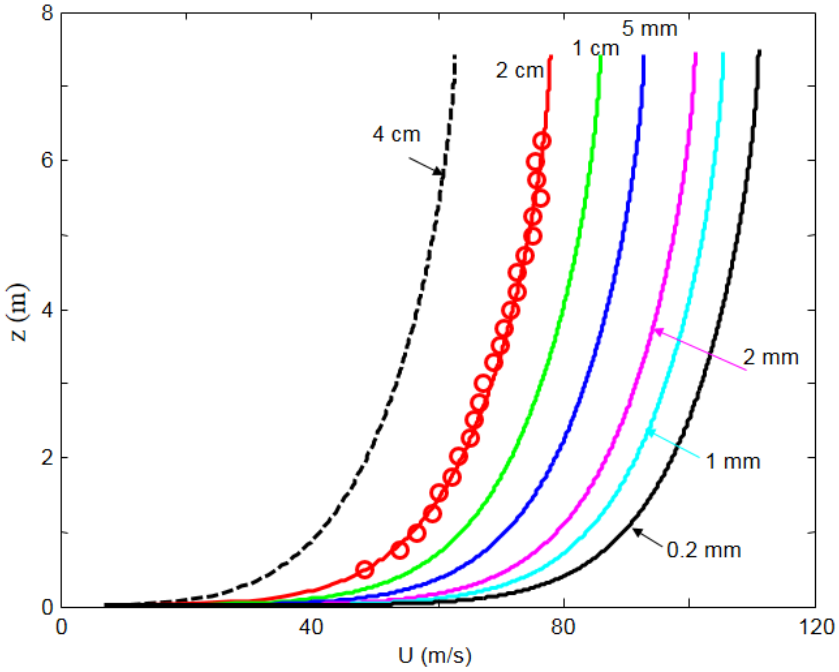

Fig. 3. Comparison of the effect of grid size on simulated velocity profiles, with the same pressure gradient, $d\eta/dx = -0.000023$, no stratification. The data (o) are coming from the first case study given by Gonzalez et al., 1996.

(2.896×10^{-5} and 4.83×10^{-6}) for these two cases, these two data are not used directly because we used the no-slip bottom boundary condition that requires a small size on vertical grid. With the same pressure gradient ($\partial\eta/\partial x = -0.000023$), the simulated velocity profiles of seven selected grid size (see Fig. 3) are very different. Even with the grid size of 0.2 mm, it appears that the profile is still changing. More important, when plot the simulated velocity profiles on semi-log scale with different and best-fitted pressure gradient (Fig. 4), it indicates that the resolution of 2 cm may be the best choice to use the no-slip boundary condition. This is because the simulated velocity profile with this grid resolution is close to a straight line in this semi-log plot. This selection is based on the understanding that for a steady turbulent, non-stratified flow, the

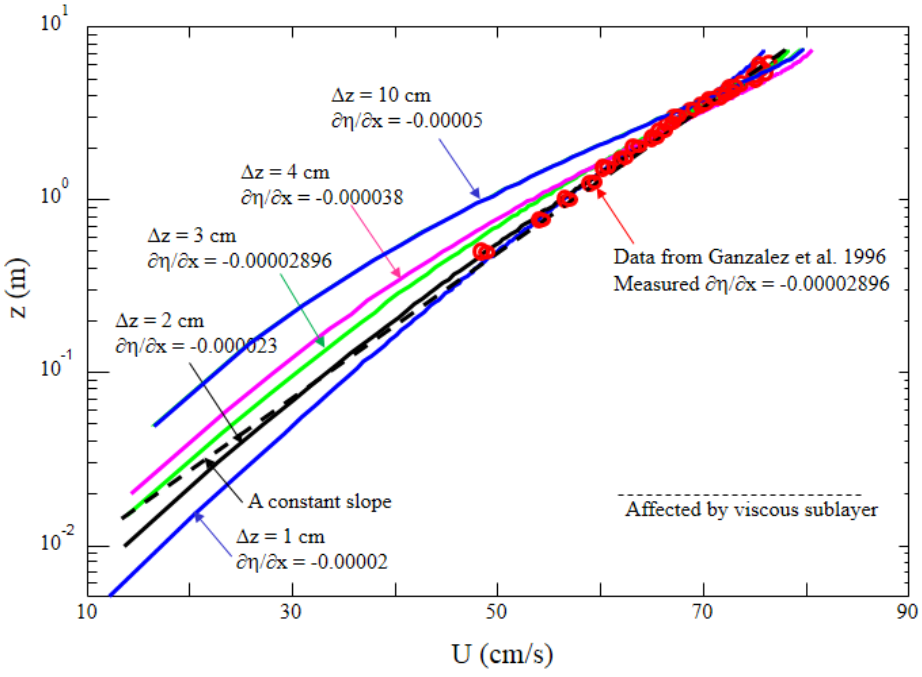

Fig. 4. Comparison of selected grid resolutions to have the best logarithm velocity profile. Pressure gradients are different for each resolution in order to best fit the data. It reveals that $\Delta z = 2$ cm is the best choice.

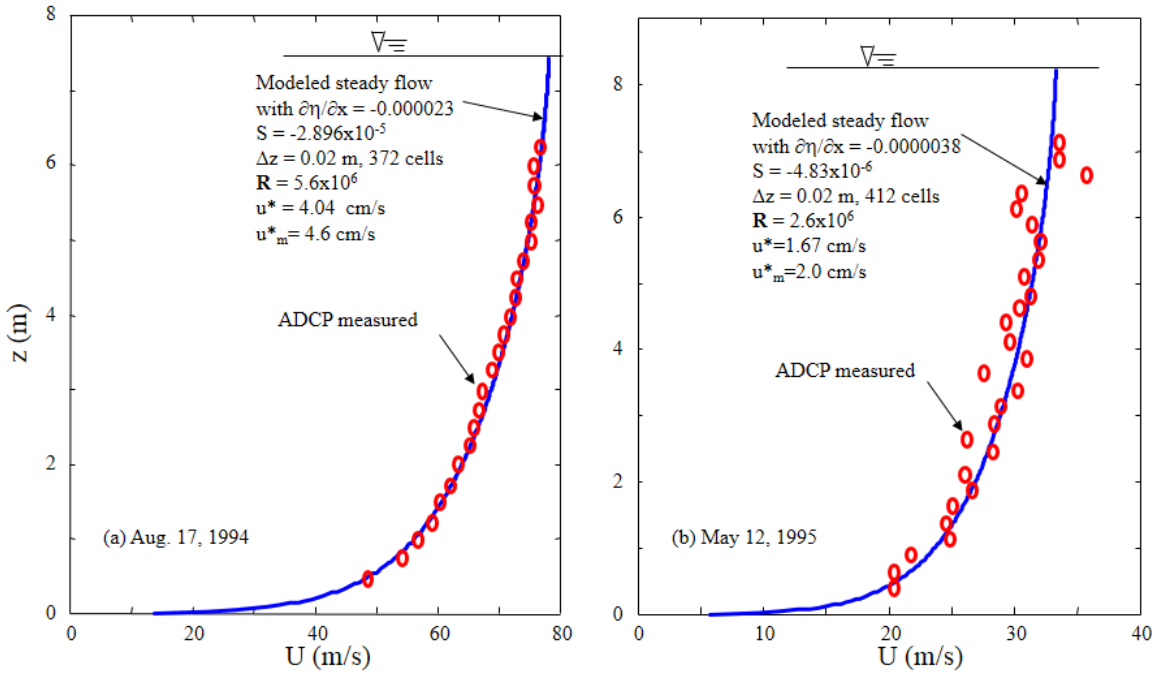

Fig. 5. Comparison of simulated and measured two velocity profiles at Chicago Sanitary and Ship Canal with 2 cm resolution. Data are from Gonzalez et al. (1996)

velocity profile is varying according to the log law. For this reason, the grid resolution of 2 cm become the first option for the following studies to find the most possible pressure gradient, and then, back to check the fit of a straight line in a semi-log plot of the velocity profile.

With the selection of $\Delta z = 0.02$ m, and using the trial-and-error approach, the best fitted driving pressure gradient can be found to produce the corresponding velocity profiles that match well with the measurements (Fig. 5). The selected $\partial\eta/\partial x$ and the model computed shear velocity, u*, versus that given by Gonzalez et al. (1996), i.e., the energy slope, S, and the estimated u^*_m, are also marked in the figure. Although these values are not exactly the same, they are close.

Tidal current. The velocity profiles measured at Menai Strait, U.K. by Rippeth et al. (2002) were used. The channel length is about 20 km with a minimum channel width about 300 m. The mean water depth is about 12.5 m at the measurement site and there is no stratification because of the well-mixed salinity. The tidal current profiles were measured by using a bottom mounted 1200 kHz ADCP, and thus, there is no data for the bottom and the top 1 m. For this case, the periodical tidal flow was generated using a sinusoidal function with a tidal amplitude of 2.25 m for the M_2 tide. The sinusoidal time series of water surface elevation were used to get $\partial\eta/\partial t$ and then transfer to $\partial\eta/\partial x$ accordingly. For this reason, a phase lag = $\tan^{-1}(-C\beta/\sigma)$ can be changed to bring the simulated velocity profile to better fit the observed profile. At this time, it is assumed that there is no constant pressure gradient along this Strait. If select a phase angle of 44 degrees, the simulation results, however, shows the maximum flood current amplitude was somewhat underestimated but the maximum ebb current amplitudes was a little overestimated (Fig. 6a). If reduce the phase lag to 40 degrees, the maximum ebb current amplitude can be matched very well, but the underestimation of maximum flood current become more severe. This situation appears showing that there is a steady flow on the flood direction, and calls for the addition of a constant pressure gradient, $\partial\eta/\partial x = -3x10^{-6}$ to the system. With this addition, the simulated results matches excellent with the observation (Fig. 7a). The good linear trend of the simulated results and the measurements on a semi-log plot indicates that it is a fully developed turbulent flow (Fig. 7b).

Although it appears that there is no difference on long-term averaged water surface elevations on both sides of the Menai Strait, it is possible that mean water level difference exists during any particular short period of time (David Bowers, personal communication).

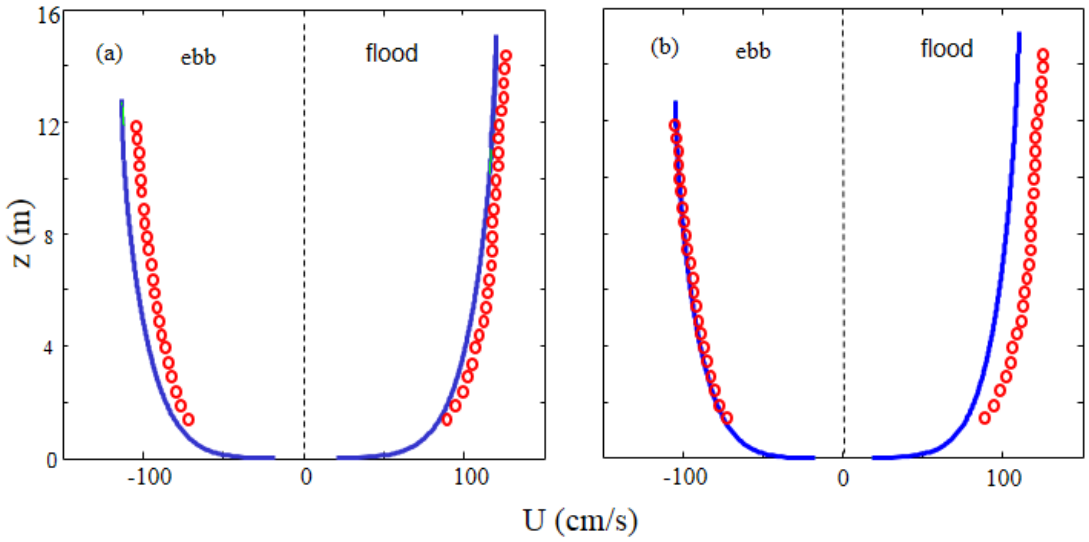

Fig. 6. Comparison of simulated tidal current amplitude profiles at the station in Menai Strait, GB with measurements from Rippeth et al., (2002). The measured tidal range is about 4.5 m with mainly M_2 tide and no stratification. A nominal grid size Δz = 2 cm is used, and zero constant pressure gradient is assumed. (a) Phase lag = 44 deg. (b) phase angle = 40 degrees.

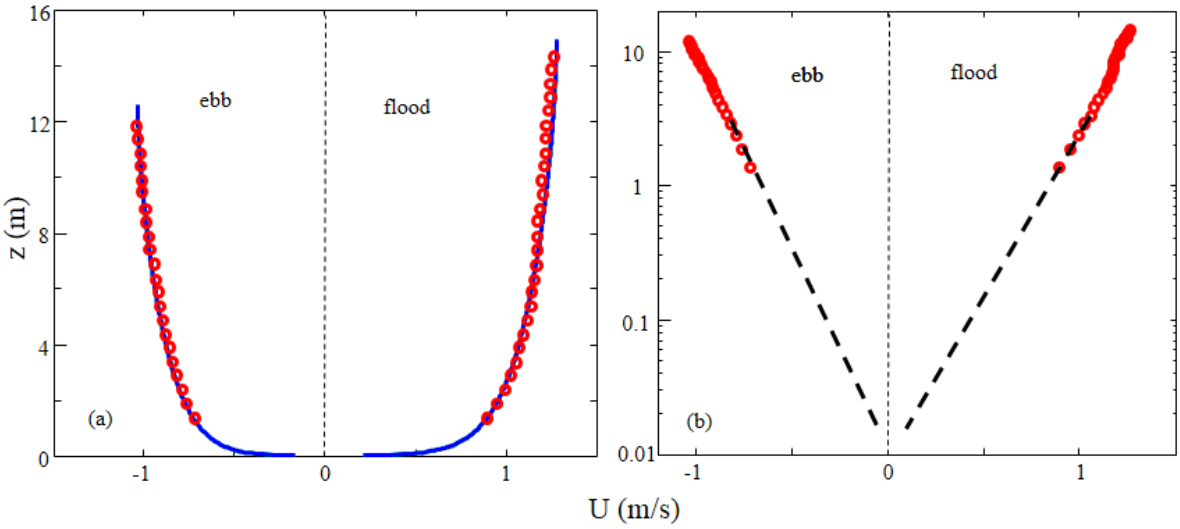

Fig. 7. Simulated tidal current amplitudes at the station in Menai Strait, GB, after adding a constant pressure gradient = -3×10^{-6} to represent the possible steady flow. (a) plot on a linear scale to show the excellent fit with data, and (b) plot on a semi-log plot to show the excellent log fit.

Salinity induced stratified flows. McCutcheon (1981) derived a modification of the logarithmic velocity profile for stratified flows which shows a linear increase of velocity for the top half of water column. He also conducted a laboratory experiment to support his solution. His lab experiment, however, was carried out in a 18 m long flume, and thus, the turbulence might not be able to be fully developed. Anwar (1983) showed that the vertical profile of horizontal velocity for a stratified estuarine flow will change from a typical logarithmic profile to a linear profile, except for the bottom 0.5 m. This change will greatly increase the near surface velocity, but not necessary near the bottom. Unfortunately, there is no data on vertical salinity distribution for model verification. Nevertheless, it is worth to check if the model simulation results also show this features. For a weakly stratified steady, open channel flow with a salinity difference of 2 psu linearly varies from top to bottom (Fig. 8c), the velocity profile indeed shows a linearly increasing trend (Fig. 8a). This is caused by a very different eddy viscosity (Fig. 8b). The velocity and eddy viscosity profiles derived by the same pressure gradient, but no salinity stratified effect are also shown in Fig. 8 for comparison.

For a stronger turbulent flow which is induced by a strong salinity gradient, the velocity profile is also shown a linearly increased trend (Fig. 9). The difference with that given in Fig. 8 is that the stratification effect is more concentrated at the middle water column, and thus, the middle portion of the water column has a linearly increased velocity profile.

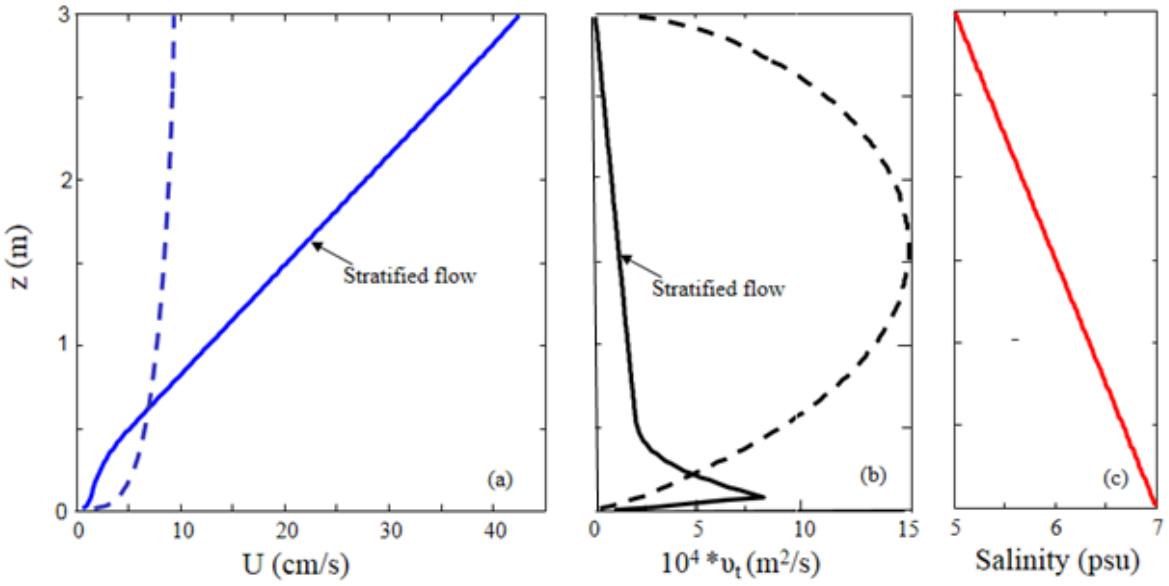

Fig. 8. Simulated velocity profiles for a steady, weak turbulent flow with (solid line) and without (dashed line) a small linear salinity gradient. (a) u, (b) eddy viscosity, and (3) salinity. The pressure gradient is $d\eta/dx = -1x10^{-6}$. Reynolds number for the non-stratification flow is around $2.5x10^{5}$.

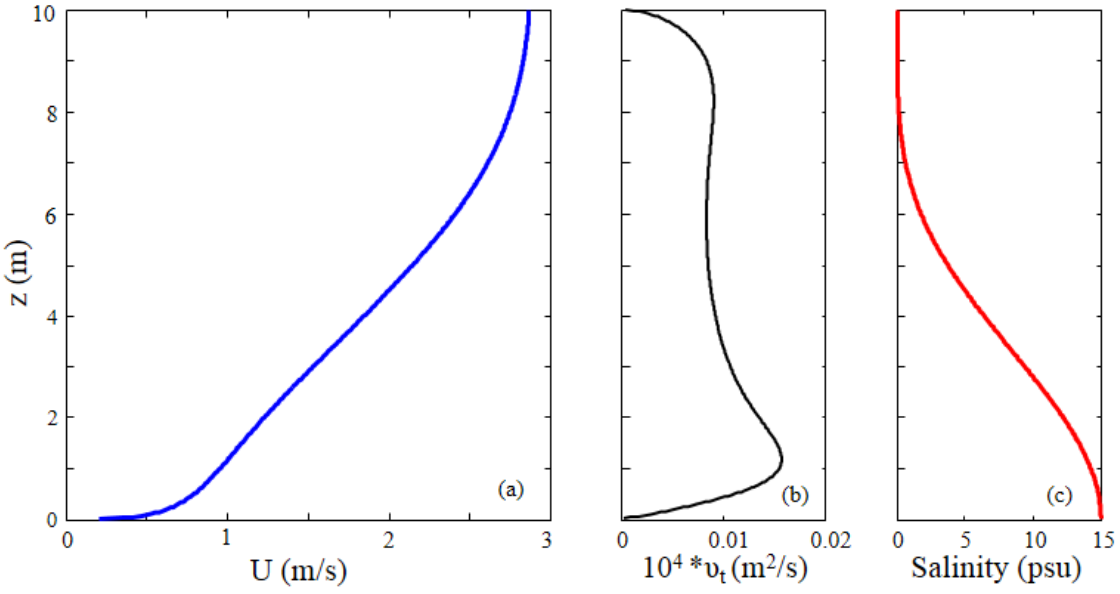

Fig. 9. Simulated velocity profiles for a steady, strong turbulent flow with a relatively strong salinity stratification at middle water depth. (a) u, (b) eddy viscosity, and (c) salinity. The pressure gradient is $d\eta/dx = -5x10^{-5}$. Reynolds number = $1.5x10^{7}$ and Richardson number ≈ 0.33.

Near bottom stratification caused by high SSC gradient. Near the bottom, suspended sediment concentration could be high under severe erosion or deposition environments. If assume a constant settling velocity, w_s, for suspended sediment and a constant near bed diffusivity, D, the SSC profile would be exponentially decreased when moving away from the bed (Henderson, 1966). With this reason, a SSC profile is selected as $C = C_o \exp(-w_s z/D)$ to check its effect on eddy viscosity, as well as the horizontal velocity profiles, where C_o is the SSC at bottom.

A reasonable turbulent flow was selected to check the effect of bottom SSC gradient, which is selected as C_o = 3 g/L and w_s/D = 1.1 as a first attempt. Although 3 g/L is not low, the effect on velocity profile is small. A maximum difference at water surface for this 10 m water depth flow is only about 10% (Fig. 10) and may be neglected.

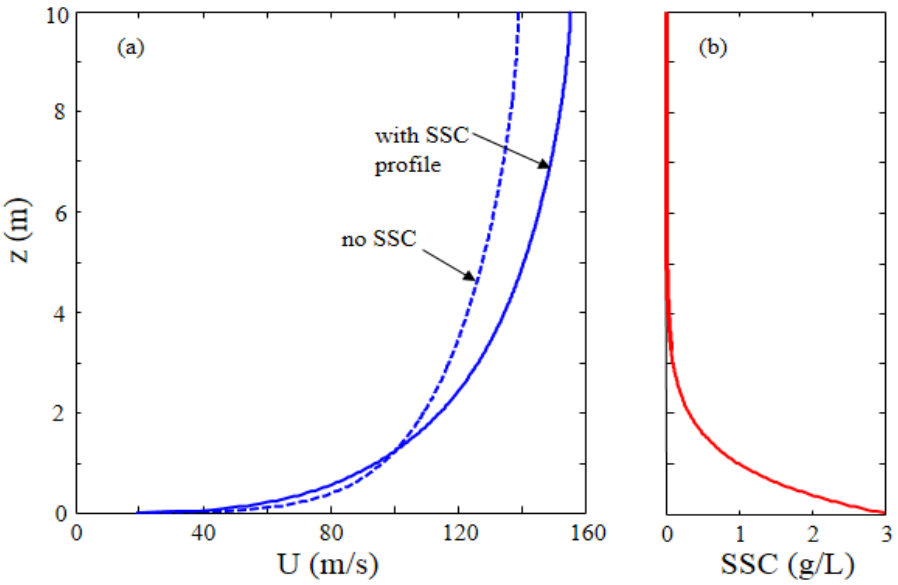

Fig. 10. Comparison of simulated velocity profiles for a steady turbulent flow with and without a gradient of suspended sediment concentration (SSC) at bottom with C_o = 3 g/L. (a) u, (b) SSC. The pressure gradient is $d\eta/dx = -5 \times 10^{-5}$. Reynolds number for the no SSC flow is around 1.2×10^{7}.

If C_o increase to 10 g/L and the turbulence is also reduced somewhat, however, the impact could be significant (Fig. 11). If the SSC has the same exponential decreasing rate (i.e., w_s and D remain the same), the maximum velocity at water surface could be 2.6 times of that without the SSC effect. Even if the SSC decreasing rate is doubled (the dotted line in Fig. 11c), the maximum velocity at water surface is still about doubled (Fig. 11a).

If the turbulence is strong, the impact of near-bed SSC gradient will be depressed (Fig. 12). Nevertheless, there is still a nearly 30% increase of the maximum velocity at water surface, if consider the SSC gradient.

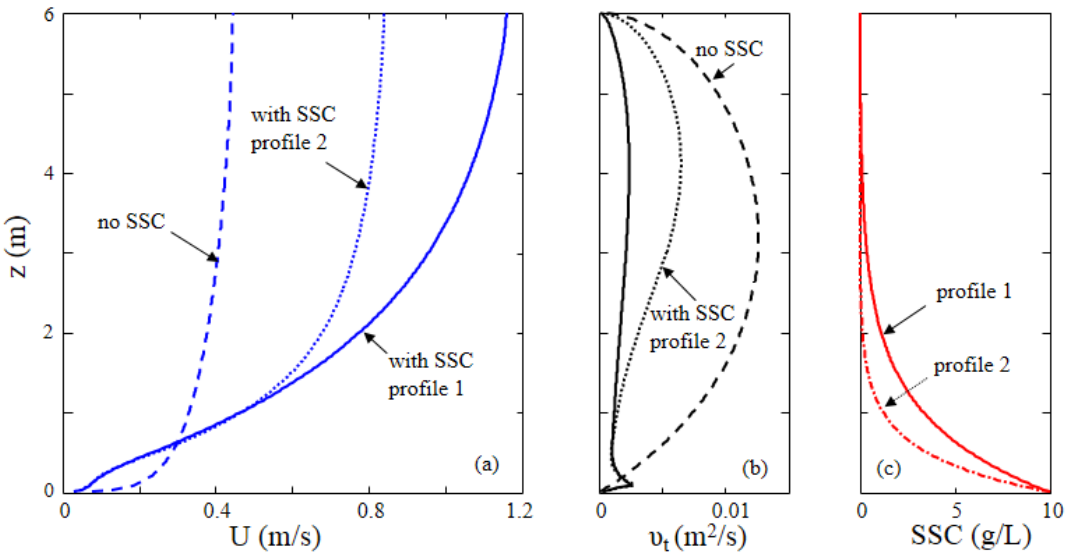

Fig. 11. Comparison of simulated velocity profiles for a relative weak steady turbulent flow with different gradient of suspended sediment concentration (SSC) at bottom with C_o = 10 g/L. (a) u, (b) eddy viscosity, and (c) SSC. The pressure gradient is $d\eta/dx = -5x10^{-5}$. Reynolds number for the no SSC flow is around $2.3x10^{6}$.

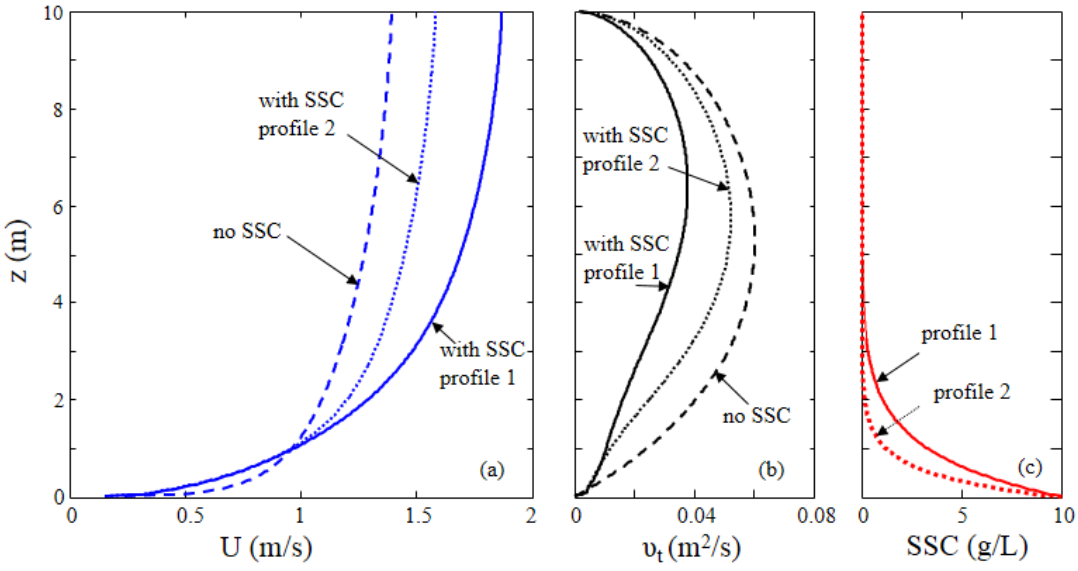

Fig. 12. Comparison of simulated velocity profiles for a steady, strong turbulent flow with different gradient of suspended sediment concentration (SSC) at bottom with C_o = 10 g/L. (a) u, (b) eddy viscosity, and (c) SSC. The pressure gradient is $d\eta/dx = -5\times10^{-5}$. Reynolds number for the no SSC flow is around 1.2×10^{7}.

The above selected cases for checking the codes demonstrate that the computer codes for simulating eddy viscosity, horizontal velocity, and stratification effects all work well.

Discussion and Conclusions

The effect of shear force acted on the water surface has not been implemented in this model yet, because there is no current need for that. But it can be done in the near future.

Although it has been checked for the selection of the best suitable grid resolution (i.e., 2 cm), this selection is good for applications with a water depth varies from a few meters up to 15 m. This resolution is not intended to address the velocity profile within the viscous sub-layer, nor for a much shallower water depth, less than 80 cm.

For most laboratory experiments with a limited length of flume, the possibility of having a fully developed turbulent flow must be checked first. If it is a fully developed lab flow, reducing the grid size should be able to simulate the lab experimental data.

Although the presented numerical scheme is a semi-implicit approximation, there is still a limitation on numerical stability. With the suggested 2 cm resolution, a time interval of 0.1 to 1 s is possible, and it may decrease somewhat with the severity of stratification.

Although tidal flow is not a steady flow, the good fit of logarithmic velocity profiles suggests that the maximum flood or maximum ebb current also a fully developed turbulent flow, even limited by the tidal period.

The salinity and SSC profiles given in the verification cases do not imply that these profiles should be available under these flow conditions. It is understood that there should be reasons to have a particular salinity or salinity profile, but these profiles used in the verification test are just assigned to check their influence on velocity. The possibility of their existence under the associated Reynolds number and Richardson number has not been checked.

Only the global Richardson number, $Ri = [(g/h)(\rho_b - \rho_t)/\rho_m]/[(u_t - u_b)/h]^2$ can be given in the figures, despite the graduate variation of water density in the water column (in other words, this is not a clearly divided two-layer flow). The subscript, t, b, and m stand for top, bottom, and mean. For a better seen of the local stratification effect, the local Richardson is a much better index.

Salinity gradient will change the eddy viscosity profile significantly. The change of velocity profile from a logarithmic distribution to a linear distribution means the surface current velocity will be greatly increased.

The near-bed gradient of SSC should be large enough to have sufficient impact on the eddy viscosity and the velocity. There are three parameters that will affect the gradient of near-bed SSC: (1) settling velocity, w_s (2) eddy diffusivity, D, and (3) C_o. This study indicates that C_o on the order of 10 g/L is necessary to produce sufficient impact. If w_s is high, e.g., for granular sediments which is relatively heavy and large, then the gradient will be large, but the affected area will be small and close to bed. If turbulence is strong, i.e., D is large, although the gradient will be smaller, but the affected near-bed area will be large. It is the combined effect of these three parameters that affects the eddy viscosity and velocity profiles. Nevertheless, the reduction of near-bed eddy viscosity stands for the reduction of the bottom friction, and that can increase the current velocity for the entire water column.

As a conclusion, a vertical 1-D hydrodynamic model has been developed that is capable of simulating stratified tidal estuarial flows. This is the bases for study other advanced topics, such as the profile of floc size distribution, settling velocity, erosion and deposition, and suspended sediment concentration in the near future.

Acknowledgments

We appreciate Ms. T.-M. Li for her comments on tidal wave propagation simulation.

References

Anwar, H.O., 1983. Turbulence measurements in stratified and well-mixed estuarine flows. *Estuarine, Coastal and Shelf Science*, 17, 243-260.

Gonzalez, J.A., C.S. Melching, and K.A. Oberg, 1996. Analysis of open-channel velocity measurements collected with an acoustic Doppler current profiler. *Proceedings of the 1st*

International Conference on New/Emerging Concepts for Rivers. International Water Resources Association.

Ha, H.K. and J. P.-Y. Maa, 2009. Evaluation of two conflicting paradigms for cohesive sediment deposition. *Marine Geology*, 265, 120-129.

Henderson, F.M., 1966. *Open Channel Flow*. MacMillan Co., New York. 522pp.

Hildebrand, F.B., 1965. *Methods of Applied Mathematics*, Prentice-Hall, New Jersey. 362 pp.

Maa, J. P.-Y., J.-I. Kwon and K.-N. Hwang, H.K. Ha, 2008. Critical bed shear stress for cohesive sediment deposition under steady flows. *Journal of Hydraulic Engineering*, 134(12), 1767-1771.

McCutcheon, S.C., 1981. Vertical velocity profiles in stratified flows. *Journal of Hydraulic Division*, ASCE, 107(HY8), 973-988.

Rippeth, T.P., E. Williams, and J.H. Simpson, 2002. Reynolds Stress and Turbulent Energy Production in a Tidal Channel. *Journal of Physical Oceanography*, 32, 1242-1251.

Rodi, W. 1993. *Turbulence models and their application in hydraulics*. International Association for Hydraulic Research, Delft, 3rd edition, Balkema

Shao, Y.Y., Y. Yan and J.P.-Y. Maa, 2011. *In-situ* measurements of settling velocity near Baimao Shoal in Changjiang Estuary. *Journal of Hydraulic Engineering*, ASCE, 137(3), 372-380.

Shen X. and Jerome P.-Y. Maa, 2015. Modeling floc size distribution of suspended cohesive sediments using quadrature method of moments. *Marine Geology*, 359, 106-119.

Shih, T.H., W.W. Liou, A. Shabbir, Z. Yang, and J. Zhu, 1994. *A new k-ε eddy viscosity model for high Reynolds number turbulent flows - Model development and validation*. Institute for Computational Mechanics in Propulsion and Center for Modeling of Turbulence and Transition. Lewis Research Center, Cleveland, Ohio, NASA, ICOMP-94-21, NASA technical memorandum 106721.

Shih, T.H., L.A. Povinelli, N.-S. Liu and M.G. Potapczuk, and J.L. Lumley, 1999. *A Generalized Wall Function*, NASA/TM-1999-209398, ICOMP-99-08, Glenn Research Center, NASA.

Toorman, E.A. 2002. Modelling of turbulent flow with suspended cohesive sediment. In: *Fine Sediment Dynamics in the Marine Environment*, Eds. J.C. Winterwerp and C. Kranenburg. Elsevier Sciences, Netherland.

Turner, J.S., 1973. *Buoyancy Effects in Fluids*, Cambridge University Press.

Uittenbogaard, R.E., avan Kester, J.A. TH and G. Stelling, 1992. *Implementation of three turbulence models in RISULA for rectangular horizontal grids*, Report Z162, Delft Hydraulics.

Wilcox, D. C., 2006. *Turbulence Modeling for CFD*, Edition 3, DCW Industries, La Canada, CA.

Umlauf, L., H. Burchard, and K. Bolding, 2000. General Ocean Turbulence Model (GOTM), Source Code and Test Case Documentation. Version 4.0.

Appendix I. Source Codes for the Vertical 1-D Hydrodynamic Model.

The codes presented here is for reference. If needed, one can request a digital copy of all the files, and others if available, given in this report. Please contact maa@vims.edu for details.

```
program V1D
!
!It simulates the velocity profiles for turbulent estuary tidal flows
! by solving the following three equations together.  Density gradient effect
! is included in the k-e equation for stratified flows
!
!  du     1    dp    d        du
! --- = ----- ---- + -[eddy(z)--]  vertical 1-D equ of momentum conservation
!  dt    L(z)  dx    d        dz
!
! vertical 1-D equation of turbulent kinematic Energy,  k equation
!
! vertical 1-D equation of energy dissipation equation,  e equation
!
!  where u  : the horizontal velocity of water motion
!        w  : vertical velocity of water motion
!     dp/dx : (change to dy/dx) the hydrostatic pressure gradient force
!      eddy : turbulent eddy viscosity, minimum = 1x10^-6 M^2/s
!        t  : time
!        z  : the vertical coordinate
!        L  : water density, a function of z
!        g  : gravitional acceleration, 9.8 m/s^2
!
!salinity & SSC profiles are interpreted at each time step among profiles or
!   generated from a constant profile internally.
!
!The input file (e.g., keflow1.dat) contains all the needed input data which
!  are described in the input data file.
!There is one output files (e.g., keflow1.lis). All the input data are copied.
!  The velocity, eddy voscosity, tke, eps, salinity, suspended sediment conc.
!  water density, and vertical velocity profiles at selected time are listed.
!There is a Matlab program (profiles_plt.m and others) to read the output
!  file and plot profiles.
!
!A fictional layer (layer nz) above top (between j = nz & nz+1) were added to
!  allow the zero shear stress at the water surface By using u(nz) = u(nz-1).
!
!A fictional layer (Layer 0) below the bottom allow the specify exactly zero
!  velocity at z = 0.
!
!Notice that u, SSC, and salinity are specified at the center of a cell,
!  between (z=j*dz & (j+1)*dz). W is specified at the lower boarder of a cell
!
!             ----------------
!  here is the top fictional layer, u(nz)=u(nz-1) for surface shear free flow
! z = h       ---------------- j=nz, top, w(nz)
!                  |  u(nz-1), C(nz-1), w(nz-1)
! z = depth-dz --------------- j=nz-1
```

```
!                 |  u(nz-2), C(nz-2), w(nz-2)
!             ---------------- j=nzm2
!                 .
!                 .
!                 .
!                 |  u(3), C(3), ws(3)
! z = 2dz    ----------------- j=3,
!                 |  u(2), C(2), ws(2)
! z = dz     ----------------- j=2,
!                 |  u(1), C(1), ws(1)
! z = 0      ----------------- j=1, bottom, w(1) = 0
!                    here is the bottom fictional layer, u(0)=-u(1)
!            -----------------
!
!The Fortran Compiler used is Lahay LF95.
!Please contact maa@vims.edu at the Virginia Institute of Marine Science
!  if you find any bug.
!The support of the sabbatic leave at HoHai University and VIMS is sincerely
!  acknowledged.                Jerome Maa, 2010
!
!This Program uses a nominal constant grid size, 0.02 m, it changes a little
!  depends on flood or edd.  For steady flow, it does not change.
!
!ng is the possible maximum number of grid cells in this 1-D model
!nt is the possible maximum number of records on tidal elevation, salinity
!      and SSC that are within one tidal cycle
!
parameter (ng=48000, nt=70, nd=0)
implicit real*8 (a-h,o-z)
integer chk, case_option, depo_optiopn, erosion_option
character*30 infile, dydx_file, eta_file, ssc_file, sal_file, outfile,
chkfile
character*18 name
character*80 title, mark
character*11 timemark

common/parame/ nz,nzm1,nzm2, dt,dz,depth, dydx_in, period,wave_amp,wavek,cgm
common/param2/ eddy_const, theta, Res_E_D
common/settli/ ssc_c1, ssc_c2, ssc_a, ssc_b, ssc_c, ssc_d, ssc_e, ws(ng)
common/array2/ z(ng),zc(ng), u(ng),dis(ng), conc(ng), cneu(ng), cl(ng), uave
common/array0/ w(ng), dydx, eta, shear(ng)
common/turbul/ tke(ng), eps(ng), eddy(ng), ustar, ustar_old, roughness
common/tide_g/ Ng_tide, hr_tide(nt),eta_given(nt), eta_old, phase_r, offset
common/salini/ ngsal,hr_sal(nt),dep_sal(nt),zc_sal(nt,ng),sal(nt,ng),
salinity(ng)
common/erodep/ erosion_option, ssc_m0, tau_cr, depo_option, tau_cd
common/sscgiv/ ngssc, hr_ssc(nt), dep_ssc(nt), zc_ssc(nt,ng), ssc(nt,ng)
common/leastq/ zb, jzb, ze, jze, iption_BC

grav=9.8
pi=3.1415926
pi2=2*pi

print*,' Select: 0. parabolic;   1.keflow1;      2.keflow2;     3.keflow3'
```

```
print*,'         4. Gonzalez_1;  5.Gonzalez_2    6.Rippeth; '
print*,'        10. sal_prpf1;  11.sal_prof2;   12.sal_prof3;'
print*,'        13. ssc_prof1;  14.ssc_prof2;   15.ssc_prof3;  16.ssc_prof4;'
print*,' Select <0, 1, 2, ...> : ';
read(*,*) case_option

select case (case_option)
case(0)
  name= 'parabolic'
case(1)
  name= 'keflow1'
case(2)
  name= 'keflow2'
case(3)
  name= 'keflow3'
case(4)
  name= 'Gonzalez_1'
case(5)
  name= 'Gonzalez_2'
case(6)
  name= 'Rippeth'
case(10)
  name= 'sal_prof1'
case(11)
  name= 'sal_prof2'
case(12)
  name= 'sal_prof3'
case(13)
  name= 'ssc_prof1'
case(14)
  name= 'ssc_prof2'
case(15)
  name= 'ssc_prof3'
case(16)
  name= 'ssc_prof4'
end select

infile=adjustl(adjustr(name) // '.dat')
print*,'input file name =', infile
open(1,file=infile, form='formatted', status='old')

outfile=adjustl(adjustr(name) // '.lis')
open(8,file=outfile,  form='formatted', status='unknown')

read(1,'(a80)') mark
read(1,'(a80)') title

write(8,'(a80)') title
write(*,'(a80)') title

do i=1,21
   read(1,'(a80)') mark
end do
read(1,*) ntotal,dt,nzm1,dz, iption_flow,iption_BC, period,wave_amp,      &
```

```
          dydx_in, ncheck, id, phase, offset, eddy_const
nz=nzm1+1
nzm2=nz-2
if (nz .ge. ng) then
   write(*,10) ng, nzm1
10 format(' Parameter ng=',i5,', is smaller than needed. Change it to ', i5)
   stop
end if

!clean the arrays
do i=1, nzm1
   z(i)=0.0
   zc(i)=0
   u(i)=0
   w(i)=0
   cl(i)=0
   cneu(i)=0
   ws(i)=0.0
   conc(i)=0.0
   tke(i)=0.0
   eps(i)=0.0
   eddy(i)=0.0
   dis(i)=0.0
end do
w(nz)=0.0

!calculate the coordinates for each cell in this vertical 1-D model
z(1)=0
zc(1)=z(1)+0.5*dz
do i=2, nzm1
   z(i)=z(i-1) + dz
   zc(i)=z(i) + 0.5*dz
end do

depth=dz*nzm1
depth_mean=depth

write(*,30)ntotal,dt,nzm1,dz,iption_flow,iption_BC,period,wave_amp,dydx_in, &
           ncheck,id, phase,offset, eddy_const
30 format(' total time steps    =',i9   /' delt t (sec)        =',f10.4/    &
          ' total z-segments    =',i6   /' cell size (m)       =',f10.3/    &
          ' option for flow dydx=',i5   /' option for Bottom BC=',i5   /    &
          ' tidal wave per.(s)  =',f10.2/' Tidal wave amp (m)  =',f10.3/    &
          ' dydx_in (Steadyflow)=',f12.7/' ncheck              =',i6/       &
          ' turbulence ID       =',i5   /' tidal phase lag(deg)=',f10.5/    &
          ' dydx offset (m/m)   =',f10.7/' the constant Eddy   =',f10.6)
!temperature effect is not included in the program yet, however, the next
!  statement is needed to calculate water density profile
temperature = 10.0

read(1, '(a80)') mark
print*, mark
if (iption_flow .eq. 3 .or. iption_flow .eq. 1) then
   read(1,'(a30)') eta_file
```

```
   print*,'input time series file for tidal elevation=', eta_file
   call tide_read(eta_file)
end if

!read in SALINITY profile data (in ppt)
read(1,'(a80)') mark
read(1,'(a80)') mark
read(1,'(a80)') mark
read(1,'(a80)') mark
read(1,*) ksalinity, sal_top, sal_bottom

!this part is for reading time series of salinity profile.
read(1,'(a80)') mark
print*,mark
if(ksalinity .eq. 3) then
   read(1,'(a30)') sal_file
   print*,'input salinity from file =', sal_file
   call salinity_read(sal_file)
else
   write(*,35) ksalinity, sal_top, sal_bottom
35 format(' Salinity option     =', i5/' Top salinity (psu)  =', f8.2/     &
          ' Bot. Salinity (pau) =',f8.2)
end if
! provide the initial salinity profile
call sal_interpretation(Ksalinity, sal_top, sal_bottom, 0.0)

!read in the information for suspended sediment concentration (in g/L)
read(1,'(a80)') mark
read(1,'(a80)') mark
read(1,'(a80)') mark
read(1,'(a80)') mark
read(1,'(a80)') mark
read(1,'(a80)') mark
read(1,*) kSSC, ssc_bottom, ssc_rate

!this part is for reading time series of SSC profiles.
read(1,'(a80)') mark
print*,mark
if(kSSC .eq. 3) then
   read(1,'(a30)') ssc_file
   print*,'input SSC profiles from file =', ssc_file
   call ssc_read(ssc_file)
else
   write(*,38) kSSC, ssc_bottom, ssc_rate
38 format(' SSC input option  =', i5/' Bottom SSC (g/L)  =', f8.2/ &
          ' exp. decrease rate=',f8.4)
end if

! provide the initial SSC profiles
call ssc_interpretation(KSSC, ssc_bottom, ssc_rate, 0.0)

close(1)
```

```
!the numerical scheme is semi-implicit, Crank-Nicolson scheme, with theta =
0.5
theta=0.5

! zb=0.1 m is selected as the nearbed elevation that TKE & eps are provided
! as the boundary conditions.  For u, it depends on the parameter iption_BC.
! If iption_BC = 1, it also calculate u_b at this level as the boundary
condition
zb=0.1
jzb=int(zb/dz)

!the next line is a function available in Lahay FORTRAN for mark the time.
call time(timemark)

if (iption_flow .eq. 0) then
   print*,'Steady flow '
   wavek=0.0
   cgm=0
end if

!calculate wave length for periodic load
if (iption_flow .eq. 2) then
!   print*,' A Periodic loading, tidal wave period = ', period
   call airy(period, depth, wL, wavekh)
   cgm=pi2/period
   wavek=pi2/wL
end if

write(8,40) ntotal,dt,nzm1,depth,iption_flow,iption_BC, period,        &
            wave_amp, dydx_in, id, phase, offset,eddy_const, zb
40 format('total time steps =',i9   /'delt t (sec)     =',f8.2/        &
          'total z-segments =',i6   /'mean water dep(m)=',f10.3/       &
          'iption flow dydx =',i5   /'iption bottom BC =',i5/          &
          'tidalwave per.(s)=',f10.2/'Tidalwave amp (m)=',f10.3/       &
          'dydx_in (Steady) =',f12.7/'turbulence ID    =',i5/          &
          'tid phaselag(deg)=',f10.5/'Const pres. grad.=',f11.7/       &
          'used const eddy  =',f9.6 /'bottom BC at z(m)=',f8.2)
phase_r=0.0174532*phase

!-********************************-
!start the time cycle
   write(*,70)
70 format('   step  depth(m)   eta(m)   dydx   ub(cm/s) uave(cm/s)  u*(m/s)
roughness(m)')

ustar_old=0.0
eta_old=0.0
do n=1, ntotal
   timet=n*dt

!  using linear interpretation to estimate the salinity and SSC profile at
next time

   call SSC_interpretation(KSSC, ssc_bottom, ssc_rate, timet)
```

```
   call sal_interpretation(Ksalinity, sal_top, sal_bottom, timet)

!Pure Water density is slightly less than 1000 kg/m^3, add the deviation for
temperature
! and salinity, as well as that from conc(i), i.e., suspended sediment
concentration,
! in unit of g/L (i.e., = kg/M^3)
   do i=1, nzm1
      cL(i)=1000.0 + sigma(temperature, salinity(i) ) + conc(i)
!     check if top layer is heavier than lower layer, more than 1kg/m^3.
      if(i .ge. 2) then
         if(  (cL(i) - cL(i-1))  .gt.  0.1 ) then
            print*,' ****warning ***'
            write(*,72) timet/3600, i, cL(i), cL(i-1)
72           format('  time=',f6.2, 'hr, cell=',i5,' cL(i)=', f8.3,' cL(i-1)=',
f8.3)
         end if
      end if
   end do

   call pressure_gradient(iption_flow, timet)

!  calculating turbulent eddy coefficient
   call current_eddy(id)

   call velocity_noslip(iption)

!update water surface elevation and each cell size if it is a tidal/wave flow
   if (iption_flow .ne. 0) then
!      depth=depth + w(nz)*dt
      depth=depth + (eta-eta_old)
      dz=depth/nzm1
      zc(1)=0.5*dz
      z(1)=0.0
      do i=2, nzm1
         z(i)=z(i-1) + dz
         zc(i)=zc(i-1) + dz
      end do
   end if

   ustar_old=ustar

!check if need to save data at selected time interval
   if (n .eq. 1 .or.  n .eq. ncheck*int(n/ncheck) ) then
!find the total mass after each time step

       write(8,70)
       write(*,75) n, depth, eta, dydx, 100*u(jzb), 100*uave, ustar,
roughness
       write(8,75) n, depth, eta, dydx, 100*u(jzb), 100*uave, ustar,
roughness
75     format(i8, 2f7.3, f12.8, 5f9.3)

       write(8,200) (100*u(i1),i1=1,nzm1)
```

```
200    format(' velocity profile (cm/s)'/(10f8.2))

       write(8,210) (10000*cneu(i1),i1=1,nzm1)
210    format(' 10000   *Total viscosity profile (m2/s)'/(10f10.3))

       write(8,215) (1000*tke(i1),i1=1,nzm1)
215    format(' 1000  * TKE profile (m2/s2)'/(10f10.3))

       write(8,220) (1000*eps(i1),i1=1,nzm1)
220    format(' 1000  * EPS profile (m2/s3)'/(10f10.3))

       write(8,225) (salinity(i1), i1=1,nzm1)
225    format(' salinity profile (ppt) '/(10f8.2))

       write(8,230) (conc(i1), i1=1,nzm1)
230    format(' sediment concentration (g/L) '/(10f9.4))

       write(8,235) (CL(i1), i1=1,nzm1)
235    format(' water density profile (kg/M3) '/(10f8.2))

       write(8,245) (100.0*w(i1), i1=1,nz)
245    format(' vertical velocity (cm/sec) '/(10f8.3))

   end if
!save the water surface elevation
   eta_old = eta

!of ntotal cycle
end do

close(8)
print*,' program stops'
print*,'Program started at ', timemark
call time(timemark)
print*,'Program   ended at ', timemark
stop
end
!
!---------------------------------------------------------------------
subroutine salinity_read(filename)
!input
! filename: the filename where measured salinity data are stored
!
!Output
! five variables are pass back by a common block /salini/
! ng_sal   : number of salinity profiles
! hr_sal   : the time (in hour) when a salinity profile are given
! dep_sal  : the water depth at the time a salinity profile is given
! sal_hr(nt,nzm1) : salinity at each cell specified at the given hour
! zc_sal(nt,nzm1) : elevation of where the salinity were specified
!
parameter (ng=48000, nt=70, nd=0)
implicit real*8 (a-h,o-z)
```

```
character*30 filename
common/parame/ nz,nzm1,nzm2, dt,dz, depth, dydx_in, period,wave_amp,
wavek,cgm
common/salini/ ng_sal, hr_sal(nt),dep_sal(nt), zc_sal(nt,ng),sal(nt,ng),
salinity(ng)

   open(3, file=filename, form='formatted', status='old')
   read(3,'(a80)') title
   write(*,'(a80)') title
   read(3,'(a80)') mark
   read(3,*) ng_sal
   if(ng_sal .gt. nt) then
      write(*,10)  ng_sal, nt
10    format('************************************'/   &
             ' Ng_sal > nt'/i6,'  ',i5/'  increase nt please')
   end if
   print*,'Number of salinity profile in the salinity time series =', ng_sal
   read(3,*) nzm1_chk
   if (nzm1_chk .ne. nzm1) then
      print*,'******************************************************'
      print*,' nzm1 <> nz when reading salinity data ',nzm1, nzm1_chk
      return
   end if
   do i=1,ng_sal
      read(3,'(a80)') mark
      read(3,*) hr_sal(i), dep_sal(i)
      read(3,'(a80)') mark
      read(3,*) ( zc_sal(i,j),j=1,nzm1)
      read(3,'(a80)') mark
      read(3,*) ( sal(i,j),j=1,nzm1)
   end do
   close(3)
   print*,'......... Completed reading salinity file'
return
end
!
!---------------------------------------------------------------------
Subroutine sal_interpretation(Ksal_code, sal_t, sal_b, timet)
!
! ksal_code = 1, a constant or linear distribution
!           = 2, a stratified s profile is generated
!           = 3, to interpret the vertical salinity profile from a given time
!                series of vertical salinity profiles. for this option, s-b
!                and s_t are not used
! sal-b   : bottom salinity (ppt)
! sal_t   : salinity (ppt) at the top of water column
!
! timet   : the time a profile is needed
!
parameter (ng=48000, nt=70, nd=0)
implicit real*8 (a-h, o-z)
common/parame/ nz,nzm1,nzm2, dt,dz, depth, dydx_in, period,wave_amp,
wavek,cgm
```

```
common/salini/ ng_sal, hr_sal(nt),dep_sal(nt),zc_sal(nt,ng),
sal(nt,ng),salinity(ng)
common/array2/ z(ng),zc(ng), u(ng),dis(ng), conc(ng), cneu(ng), cl(ng), uave

dimension zzc(ng), x(ng), ss(ng)

!linear distribution, include zero or/and constant salinity
if( ksal_code .eq. 1) then
   salinity(1)=sal_b
   salinity(nzm1)=sal_t
   do i=2,nzm2
      salinity(i)=salinity(1) - (sal_b - sal_t)*zc(i)/depth
   end do
end if

!A possible stratification, s curve shape
if( ksal_code .eq. 2) then
   pi=3.1415926
   do i=1, nzm1
      x(i)=z(i) - 0.5*depth
      x(i)=pi*x(i)/depth;
      ss(i)=sin(x(i) );
   end do

   smax=0
   do i=1,nzm1
      ss(i)=ss(i) + 1.0
      ss(i)=ss(i)**2
      if( smax .lt. ss(i) ) smax=ss(i)
   end do

   do i=1,nzm1
      ss(i)=sal_t + ss(i)*(sal_b - sal_t)/smax
   end do

   do i=1,nzm1
      k=nzm1-i+1
      salinity(k)=ss(i)
   end do
end if

if (ksal_code .eq. 3)  then
!using linear interpretation to increase resolution in time domain, find
!where the specified time is related to the two records: one behind & one
after
    time=timet
    time_hr=time/3600
    do while (time_hr .gt. hr_sal(ng_sal) )
       time_hr=time_hr - hr_sal(ng_sal)
    end do
    do k=1,ng_sal-1
       if (time_hr .ge. hr_sal(k) .and. time_hr .lt. hr_sal(k+1) )   ks=k
    end do
```

```
    do j=1,nzm1
       sal1=sal(ks,j)
       sal2=sal(ks+1,j)
       salinity(j)=sal1 + (time_hr - hr_sal(ks) )*(sal2-sal1)/( hr_sal(ks+1)-
hr_sal(ks) )
    end do
end if

return
end
!
!-------------------------------------------------------------------------
subroutine ssc_read(filename)
!input
! filename: where measured suspended sediment concentration data are stored
!
!Output
! five variables are pass back by a common block /sscgiv/
! Nsscg   : number of SSC profile
! hr_ssc  : the time (in hr) that a SSC profiles are measured
! dep_ssc : the water depth at the time that SSC profiles are made
! ssc     : ssc profiles (maximum of nt profiles, 6 points at each profile)
! ssc_temp: linear interpreted SSC profiles to increase vertical resolution
!
parameter (ng=48000, nt=70, nd=0)
implicit real*8 (a-h,o-z)

character*30 filename
common/parame/ nz,nzm1,nzm2, dt,dz, depth, dydx_in, period,wave_amp,
wavek,cgm
common/sscgiv/ ng_ssc, hr_ssc(nt), dep_ssc(nt), zc_ssc(nt,ng), ssc(nt,ng)

   open(3, file=filename, form='formatted', status='old')
   read(3,'(a80)') title
   write(*,'(a80)') title
   read(3,'(a80)') mark
   read(3,*) ng_ssc
   print*,'Number of suspended sediment concentration profiles read =',
ng_ssc
   read(3,*) nzm1_chk
   if (nzm1_chk .ne. nzm1) then
      print*,'*****************************************************'
      print*,' nzm1 <> nz when reading SSC data ', nzm1, nzm1_chk
      return
   end if
   do i=1,ng_ssc
      read(3,'(a80)') mark
      read(3,*) hr_ssc(i), dep_ssc(i)
      read(3,'(a80)') mark
      read(3,*) (zc_ssc(i,j), j=1,nzm1)
      read(3,'(a80)') mark
      read(3,*) (ssc(i,j), j=1,nzm1)
   end do
   close(3)
```

```
   print*,'......... Completed reading SSC files'
return
end
!
!-------------------------------------------------------------------------
----
Subroutine SSC_interpretation(KSSC_code, ssc_b, ssc_rate, timet)
!
! kssc_code = 1, an exponential decrease profile or a constant
!           = 2, bottom half has a constant SSC_B, zero on top half.
!           = 3, to read a time series of SSC profile
! ssc_b   : bottom SSC (g/L)
! ssc_rate: exponential decreasing rate of SSC
!
! timet   : the time a profile is needed
!
parameter (ng=48000, nt=70, nd=0)
implicit real*8 (a-h, o-z)
common/parame/ nz,nzm1,nzm2, dt,dz, depth, dydx_in, period,wave_amp,
wavek,cgm
common/array2/ z(ng),zc(ng), u(ng),dis(ng), conc(ng), cneu(ng), cl(ng), uave
common/sscgiv/ ng_ssc, hr_ssc(nt),dep_ssc(nt),zc_ssc(nt,ng), ssc(nt,ng)

!SSC profile: an exponential decreasing profile or a constant
if( kssc_code  .eq. 1) then
   conc(1)=ssc_b
   do i=2,nzm1
      conc(i)=ssc_b *exp(ssc_rate*zc(i) )
   end do
end if

!a special case of SSC profile for simulate hear transfer problem
if( kssc_code .eq. 2) then
   nhalf=nzm1/2
   do i=1,nhalf
      conc(i)=ssc_b
   end do
end if


if( kssc_code .eq. 3) then
!   to interpret the SSC profile for all cells at any time
!   where the specified time is related to the two records: one behind & one
after
    time=timet
    time_hr=time/3600
    do while (time_hr .gt. hr_ssc(ng_ssc) )
       time_hr=time_hr - hr_ssc(ng_ssc)
    end do

!find what are the two records that the current time is in between
    do k=1,ng_ssc
       if (time_hr .ge. hr_ssc(k) .and. time_hr .lt. hr_ssc(k+1) )  ks=k
    end do
```

```
    do j=1,nzm1
       ssc1=ssc(ks,j)
       ssc2=ssc(ks+1,j)
       conc(j)=ssc1 + (time_hr - hr_ssc(ks) )*(ssc2-ssc1)/( hr_ssc(ks+1)-
hr_ssc(ks) )
    end do
end if

return
end
!
!
!---------------------------------------------------------------------
subroutine tide_read(filename)
!
!this is to read eta time series as the given
!
!input
! filename: the filename where tidal elevation data are stored
!
!Output
! three variables are pass back by a common block /tide_g/
! N_tide     : number of tidal elevation data
! hr_tide    : The time that eta (or water depth) are given
! eta_given  : the recorded water depth or tidal elevation data
!
parameter (ng=48000, nt=70, nd=0)
implicit real*8 (a-h,o-z)

character*30 filename
common/parame/ nz,nzm1,nzm2, dt,dz, depth, dydx_in, period,wave_amp,
wavek,cgm
common/tide_g/ Ng_tide, hr_tide(nt), eta_given(nt), eta_old, phase_r, offset
dimension depth_given(nt)

   print*,' Openning eta time series file: ', filename
   open(2,file=filename, form='formatted', status='old')
   read(2,'(a80)') title
   write(*,'(a80)') title
   read(2,'(a80)') mark
   read(2,'(a80)') mark

   depth_mean=0
   do i=1,100
      read(2,*, END=20) day, hr, depth_given(i)
      hr_tide(i)=24*day + hr
      if (i.eq. 1) beg=hr_tide(1)
      hr_tide(i)=hr_tide(i) - beg
      depth_mean=depth_mean + depth_given(i)
   end do
20 close(2)
   ng_tide=i-1;
   depth_mean=depth_mean/ng_tide
```

```
   period=3600*hr_tide(Ng_tide)
   print*,'Period of the given tidal time series (sec)=', period
   print*,'Mean water depth (m)= ', depth_mean
   cgm=6.2832/period
   print*,'......... Completed reading tide file'
   print*,'  hr_tide   depth     eta'
   do i=1,Ng_tide
       eta_given(i)=depth_given(i) - depth_mean
       write(*,30) hr_tide(i), depth_given(i), eta_given(i)
30     format(3f10.3)
   end do
return
end
!
!---------------------------------------------------------------------
!A function to calculate unit weight of water for given temperature and
salinity
FUNCTION SIGMA(T,S)
real*8 S0,E1,E2,B1,B2, T,S, SIGMA
S0=((6.76786136E-6*S - 4.8249614E-4)*S+0.814876577)*S - 0.0934458632
E1=(((-1.4380306E-7*T-0.00198248399)*T-0.545939111)*T + 4.53168426)*T
B1=((-1.0843E-6*T+9.8185E-5)*T - 0.0047867)*T + 1.
B2=((1.667E-8*T-8.164E-7)*T + 1.803E-5)*T
E2=(B2*S0+B1)*S0
SIGMA=E1/(T+67.26) + E2
RETURN
END

!---------------------------------------------------------------------
subroutine pressure_gradient(iption, timet)
!
!This is a function to provide dydx information at each time step
!
!input
! iption: option for calculating dydx:
!         0. a given constant for steady flows
!         1. using a time series of d(eta)/dx record, 1 tidal wave period
long;
!         2. using a sine function
!         3. using a time series of eta record, 1 tidal wave period long;
! timet : the time for next time step, in unit of seconds.
!         this variabel is replace immediately by 'time' for local
calculation
! other input variables: N_tide and hr_tide(N_tide)
!       are pass to this subroutine by a common block /tide_g/
!
!Output
! dydx  : The water surface gradient for the next time step.  non-dimension.
!         This parameter is return to 'main' by using a common block
/array0/
!
parameter (ng=48000, nt=70, nd=0)
implicit real*8 (a-h,o-z)
complex*16 zi, zpsi, zdydx
```

```
common/parame/ nz,nzm1,nzm2, dt,dz, depth, dydx_in, period,wave_amp,
wavek,cgm
common/array0/ w(ng), dydx, eta, shear(ng)
common/tide_g/ Ng_tide, hr_tide(nt),eta_given(nt), eta_old, phase_r, offset

pi=3.1416
grav=9.8
zi=(0, 1)
x=0

!replace the global time variable for local uses
time=timet

if(iption .eq. 0) then
   dydx=dydx_in
end if

if(iption .eq. 1) then
!not done yet (probably there will be no this kind of input data)
end if

if (iption .eq. 2) then
!use a sine function to calculate tidal elevation and find d(eta)/dt first,
and then
!transfer to d(eta)/dx, add a constant p. gradient to represent freshwater
discharge
   x=0.0
   eta=wave_amp*exp(-zi*(wavek*x-cgm*time + pi/2) )
   dydt=(eta - eta_old)/dt
end if

if (iption .eq. 3) then
! use a given time series of (eta(i), i=1, Ng_tide-1) to find d(eta)/dt at
any
! particular time first, and then transfer to d(eta)/dx.
! first make sure the given time is within the range (in hr) in which the
tidal
! time series data are provided

   time_hr=time/3600
   do while (time_hr .gt. hr_tide(Ng_tide) )
      time_hr=time_hr - hr_tide(Ng_tide)
   end do

!now search which two time records that the next time step is in between
   do i=1,Ng_tide-1
      if (time_hr .ge. hr_tide(i) .and. time_hr .lt. hr_tide(i+1) )   ise=i
   end do
   dydt=(eta_given(ise+1) - eta_given(ise) )/(3600.0*(hr_tide(ise+1) -
hr_tide(ise) ) )
   eta=eta_given(ise) + dydt*3600.0*(time_hr - hr_tide(ise) )
end if

if (iption .eq. 2 .or. iption .eq. 3) then
```

```
!calculates spatial gradient, according to dy/dx=psi*dy/dt  where psi=-
1/C+i(beta/cgm)
   wave_C=sqrt(grav*depth)
   wavek=2*pi/(wave_C*period)
! beta is the attenuation rate of tidal amplitude along the tidal channel
   beta=-wavek*tan(phase_r)
   zpsi=-1/wave_C + zi*beta/cgm
   zdydx=zpsi*dydt
   rdydx=sqrt( real(zdydx)**2 + aimag(zdydx)**2)
   pdydx=atan2(aimag(zdydx), real(zdydx) )
   if (iption .eq. 3)  dydx = rdydx*sin(pdydx)
   if (iption .eq. 2)  dydx = rdydx*sin(pdydx) + offset
!   if (iption .eq. 2)  dydx = real(zdydx) + offset;  /******checked, not
work*******/
end if

return
end
!
!---------------------------------------------------------------------
subroutine velocity_noslip(iption)
!this subroutine build the tri-diagonal matrix equation and solved for
! horizontal velocity by using no-slip boundary condition.
!
! iption = 0 : for steady flow;
! iption = 1 : for wave/tidal flow with internally calculated d(eta)/dx
!        = 2 : for wave/tidal flow with input d(eta)/dx time series, not
possible
!        = 3 : for wave/tidal flow with input eta time series;
!
!It then calculate the vertical velocity w, if it is not a steady flow.

parameter (ng=48000, nt=70, nd=0)
implicit real*8 (a-h,o-z)
complex*16 zi, zpsi

common/parame/ nz,nzm1,nzm2, dt,dz, depth, dydx_in, period,wave_amp,
wavek,cgm
common/settli/ set_c1, set_c2, set_a, set_b, set_c, set_d, set_e, ws(ng)
common/array2/ z(ng),zc(ng), u(ng),dis(ng), conc(ng), cneu(ng), cL(ng), uave
common/array0/ w(ng), dydx, eta, shear(ng)
common/tide_g/ N_tide, hr_tide(nt),eta_given(nt),eta_old, phase_r,offset
common/leastq/ zb, jzb, ze, jze, iption_BC
common/param2/ eddy_const, theta, Res_E_D


dimension clow(ng), diag(ng), upper(ng), unew(ng), b(ng)
dimension dudt(ng)

if(iption_BC .eq. 1) then
   print*,'Should not use this function (velocity_noslip), iption_BC == 1'
   stop
end if
```

```
grav=9.8
pi=3.1415926
pi2=2*pi
zi=(0, 1)

!specify the implicit factor
ttzz=theta*dt/dz/dz
ttzz4=theta*dt/dz/dz/4.0
cttzz=(1.0 - theta)*dt/dz/dz
cttzz4=(1.0 - theta)*dt/dz/dz/4.0

!the properties at cell 0 (the fictional layer) is set equal to that at cell
1
   a1=-cneu(1)*ttzz + (cL(2)*cneu(2)-cL(1)*cneu(1))*ttzz4/cL(1)
   a2=1 + 2.0*cneu(1)*ttzz
   a3=-cneu(1)*ttzz - (cL(2)*cneu(2)-cL(1)*cneu(1))*ttzz4/cL(1)
   diag(1)=-a1+a2
   Upper(1)=a3

   b1=cneu(1)*cttzz - (cL(2)*cneu(2)-cL(1)*cneu(1))*cttzz4/cL(1)
   b2=1 - 2.0*cneu(1)*cttzz
   b3=cneu(1)*cttzz + (cL(2)*cneu(2)-cL(1)*cneu(1))*cttzz4/cL(1)
   bforce=-cl(nzm1)/cl(1)*grav*dydx*dt
   b(1)=bforce - b1*u(1) + b2*u(1) + b3*u(2)

!the middle cells
   do j=2,nzm2
      a1=-cneu(j)*ttzz + (cL(j+1)*cneu(j+1)-cL(j-1)*cneu(j-1))*ttzz4/cL(j)
      a2=1 + 2*cneu(j)*ttzz
      a3=-cneu(j)*ttzz - (cL(j+1)*cneu(j+1)-cL(j-1)*cneu(j-1))*ttzz4/cL(j)
      cLow(j)=a1
      diag(j)=a2
      Upper(j)=a3

      b1=cneu(j)*cttzz - (cL(j+1)*cneu(j+1)-cL(j-1)*cneu(j-1))*cttzz4/cL(j)
      b2=1 - 2.0*cneu(j)*cttzz
      b3=cneu(j)*cttzz + (cL(j+1)*cneu(j+1)-cL(j-1)*cneu(j-1))*cttzz4/cL(j)
      bforce=-cL(nzm1)/cL(j)*grav*dydx*dt
      b(j)=bforce + b1*u(j-1) + b2*u(j) + b3*u(j+1)
   end do

!the top cell
!the properties at cell nz (the fictional layer) is set equal to that at cell
nz-1
   a1=-cneu(nzm1)*ttzz + (cL(nzm1)*cneu(nzm1)-cL(nz-2)*cneu(nz-
2))*ttzz4/cL(nzm1)
   a2=1 + 2.0*cneu(nzm1)*ttzz
   a3=-cneu(nzm1)*ttzz - (cL(nzm1)*cneu(nzm1)-cL(nz-2)*cneu(nz-
2))*ttzz4/cL(nzm1)
   clow(nzm1)=a1
   diag(nzm1)=a2+a3

   b1=cneu(nzm1)*cttzz - (cL(nzm1)*cneu(nzm1)-cL(nz-2)*cneu(nz-
2))*cttzz4/cL(nzm1)
```

```
   b2=1 - 2.0*cneu(nzm1)*cttzz
   b3=cneu(nzm1)*cttzz + (cL(nzm1)*cneu(nzm1)-cL(nz-2)*cneu(nz-
2))*cttzz4/cL(nzm1)
   bforce=-grav*dydx*dt
   b(nzm1)=bforce + b1*u(nzm2) + (b2 + b3)*u(nzm1)
   j=nzm1

!now call the tridiagonal matrix solver and find the solution

   call tridiagonal(nzm1, cLow,diag,Upper, b, unew)

!calculate du/dt, update horizontal vel. component u, and then
!  calculate vertical velocity component, w
   do j=1,nzm1
      dudt(j)=(unew(j) - u(j) )/dt
      u(j)=unew(j)
   end do

return
end

!-----------------------------------------------------------------------
subroutine current_eddy(ID)
!id is an input parameter
!id = 0; a zero order estimation of eddy viscosity, i.e., constant
!id = 1; a zero order estimation of eddy viscosity, i.e., parabolic profile
!id = 2; a 2nd order estimation, using k-e model
!
parameter (ng=48000, nt=70, nd=0)
implicit real*8 (a-h, o-z)
real*8  Len, m1,m2,m3,m4, m5, m6
character*1 q

common/parame/ nz,nzm1,nzm2, dt,dz, depth, dydx_in, period,wave_amp,
wavek,cgm
common/param2/ eddy_const, theta, Res_E_D
common/settli/ set_c1, set_c2, set_a, set_b, set_c, set_d, set_e, ws(ng)
common/array2/ z(ng),zc(ng), u(ng), dis(ng), conc(ng), cneu(ng), cL(ng), uave
common/array0/ w(ng), dydx, eta, shear(ng)
common/turbul/ tke(ng), eps(ng), eddy(ng), ustar, ustar_old, roughness
common/leastq/ zb, jzb, ze, jze, iption_BC

dimension belo(ng),diag(ng),abov(ng),right(ng), dudz(ng)
dimension temp(ng)

!cneu include two parts: molecular viscosity, vis, & turbulent vis., eddy.
!stratification effects is only included in 2nd order estimation

vis=1.0e-6

if (id .eq. 0) then
!zeroth order approximation, a constnat, unit m^2/s.
   do j=1,nzm1
      cneu(j)=eddy_const
```

```
   end do
end if

if (id .eq. 1) then
!parabolic eddy viscosity approximation, zero order
!first calculate vertical gradient of horizontal velocity
   dudz(1)=abs(u(2) + u(1) ) / (2.0*dz)
   dudz(nzm1)=0
   do j=2,nzm2
      dudz(j)=abs(u(j+1) - u(j-1) ) / (2.0*dz)
   end do
!estimating the parabolic eddy viscosity profile
   do j=1,nzm1
      dum=1.0 - zc(j)/depth
      Len=0.4 * zc(j) * sqrt(dum)
      cneu(j)=Len*Len*dudz(j)
      if (cneu(j) .lt. vis) cneu(j) = vis
   end do
!The next line is not needed for this case, but provided to make the CRT
display more
!consistent with the turbulent flows
   call shearvelocity(ustar, roughness)
end if


if (id .eq. 2) then
!2nd order k-e approximation. You will need to estimate shear velocity first
!  This bottom B.C. is specified at zb(=0.1 m) above the bed. The
corresponding
!  cell number is 0.1/dz.  For these cells below this cell, The k values are
!  prorated between kb and 0 because k = 0 at z = 0,
!The correct selection of zb should be based on the flow.  If the total water
depth
!  is small, then one need to adjust this value (in the main program)
!
   call shearvelocity(ustar, roughness)
   c_meu=0.09
   tke_zb=ustar*ustar/sqrt(C_meu)
   PR=0.89
   cgm_k=1.0
   grav=9.8

   m1=theta*dt/(2.0*dz)
   m2=(1.0-theta)*dt/(2.0*dz)
   m3=theta*dt/(dz*dz)
   m4=(1.0-theta)*dt/(dz*dz)

!starts from the cell right above 'jzb'.  This is becasue of the selection of
a 0.1 m
! above bed as the elevation above viscous sublayer.
   j=jzb+1
   nj=1
   temp1=( CL(j+1)*eddy(j+1)-CL(j-1)*eddy(j-1) )/(4.0*cgm_k)
   temp2=CL(j)*eddy(j)/cgm_k
```

```
   p1=-m1*CL(j-1)*w(j-1) + m3*temp1 - m3*temp2
   diag(nj)=CL(j) + 2.0*m3*CL(j)*eddy(j)/cgm_k
   abov(nj)= m1*CL(j+1)*w(j+1) - m3*temp1 - m3*temp2

   q1= m2*CL(j-1)*w(j-1) - m4*temp1 + m4*temp2
   q2=CL(j) - 2.0*m4*CL(j)*eddy(j)/cgm_k
   q3=-m2*CL(j+1)*w(j+1) + m4*temp1 + m4*temp2
   P=dt*CL(j)*eddy(j)*( (u(j+1) - u(j-1) )/(2.0*dz) )**2
   G=grav*dt*eddy(j)*(cL(j+1) - cL(j-1))/(2.0*dz*PR)

   diss=CL(j)*eps(j)*dt
   right(nj)=(q1-p1)*tke_zb + q2*tke(j) + q3*tke(j+1) + P + G - diss

!for the internal cells
   do j=jzb+2,nzm2
      nj=nj+1
      temp1=( CL(j+1)*eddy(j+1)-CL(j-1)*eddy(j-1) )/(4.0*cgm_k)
      temp2=CL(j)*eddy(j)/cgm_k

      belo(nj)=-m1*CL(j-1)*w(j-1) + m3*temp1 -m3*temp2
      diag(nj)=CL(j) + 2.0*m3*CL(j)*eddy(j)/cgm_k
      abov(nj)= m1*CL(j+1)*w(j+1) - m3*temp1 - m3*temp2

      q1=m2*CL(j-1)*w(j-1) - m4*temp1 + m4*temp2
      q2=CL(j) - 2.0*m4*CL(j)*eddy(j)/cgm_k
      q3=-m2*CL(j+1)*w(j+1) + m4*temp1 + m4*temp2
      P=dt*CL(j)*eddy(j)*( (u(j+1) - u(j-1) )/(2.0*dz) )**2
      G=grav*dt*eddy(j)*(cL(j+1) - cL(j-1))/(2.0*dz*PR)

      diss=CL(j)*eps(j)*dt
      right(nj)=q1*tke(j-1) + q2*tke(j) + q3*tke(j+1) + P + G - diss
   end do

!for the water surface cell
   j=nzm1
   nj=nj+1
   temp1=( CL(j)*eddy(j) - CL(j-1)*eddy(j-1) )/(2.0*cgm_k)
   temp2=CL(j)*eddy(j)/cgm_k

   belo(nj)=-m1*CL(j-1)*w(j-1) + m3*temp1 - m3*temp2
   diag(nj)=CL(j) + 2.0*m3*CL(j)*eddy(j)/cgm_k
   abo = m1*CL(j)*w(j) - m3*temp2 - m3*temp2
!the next line is marked by by Li, March 26. 2011, not convergent though.
!  abo = m1*CL(j)*w(j) - m3*temp1 - m3*temp2
   diag(nj)= diag(nj) - abo

   q1=m2*CL(j-1)*w(j-1) - m4*temp1 + m4*temp2
   q2=CL(j) - 2.0*m4*CL(j)*eddy(j)/cgm_k
   q3=-m2*CL(j)*w(j)  + m4*temp1 + m4*temp2
   P=dt*CL(j)*eddy(j)*( (u(j) - u(j-1) )/(2.0*dz) )**2
!  the term u(j) may need to change to u(j+1) when there is a wind shear
   G=0
   diss=CL(j)*eps(j)*dt
```

```
   right(nj)=q1*tke(j-1) + q2*tke(j) - q3*tke(j) + P + G - diss

!now call the tridiagonal matrix solver and find the solution.  Notice that
!    only the cell between nb+1 and nzm1 are solved for

   call tridiagonal(nj, belo,diag,abov, right, temp)

!update the results
   do j=1,nj
      if (temp(j) .lt. 0.0)  temp(j) = 1.0e-8
      tke(j+jzb)= temp(j)
   end do
   tke(jzb)=tke_zb

!update the tke for the near bed zone.
   do j=1,jzb-1
      tke(j)=tke_zb*(j-0.5)*dz/((jzb-0.5)*dz)
   end do

!finished the k equation, now works on the eps equation
   cgm_e=1.3
   c1e=1.44
   c2e=1.92
!the value of c3e has a range between 0 and 0.3
   c3e=0.0
   m5=dt/(2.0*dz)
   m6=dt/(dz*dz)
   eps_zb=abs(ustar**3)/(0.41*zb)

!   if( q .eq. 'y' .and. nt .ge. ntchk)  write(8,11) ustar, eps_zb
!11 format(' Tri-diagonal matrix eq. for EPS with ustar, e_zb=
',e12.6,2x,e12.6)

!Start from the cell above zb = 10 cm
   nj=1
   j=jzb+1
   temp1= (CL(j+1)*eddy(j+1) - CL(j-1)*eddy(j-1) )/(4.0*cgm_e)
   temp2= CL(j)*eddy(j)/cgm_e
   P=CL(j)*eddy(j)*( (u(j+1) - u(j-1) )/(2.0*dz) )**2
   G=grav*eddy(j)*(cL(j+1) - cL(j-1))/(2.0*dz*PR)

   S1=-m1*CL(j-1)*w(j-1) + m3*temp1 - m3*temp2
   diag(nj)=CL(j) + 2.0*m3*temp2
   if(tke(j) .ge. 1.0e-30) then
      diag(nj)=diag(nj) + C2e*CL(j)*dt*eps(j)/tke(j) -
c1e*theta*dt*(P+c3e*G)/tke(j)
   end if
   abov(nj)=m1*CL(J+1)*w(j+1) - m3*temp1 - m3*temp2

   T1=-m2*CL(j-1)*w(j-1) - m4*temp1 + m4*temp2
   T2=CL(j) - 2.0*m4*temp2
   if(tke(j) .ge. 1.0e-30) then
      T2=T2 + c1e*(1-theta)*dt*(P + c3e*G)/tke(j)
   end if
```

```
   T3= m2*CL(j+1)*w(j+1) + m4*temp1 + m4*temp2

   right(nj)=(T1-S1)*eps_zb + T2*eps(j) + T3*eps(j+1)
!   if( q .eq. 'y' .and. nt .ge. ntchk) write(8,2)
nj,diag(nj),abov(nj),right(nj), G

!for the internal cells
   do j=jzb+2,nzm2
      nj=nj+1
      temp1= (CL(j+1)*eddy(j+1) - CL(j-1)*eddy(j-1) )/(4.0*cgm_e)
      temp2= CL(j)*eddy(j)/cgm_e
      P=CL(j)*eddy(j)*( (u(j+1) - u(j-1) )/(2.0*dz) )**2
      G=grav*eddy(j)*(cL(j+1) - cL(j-1))/(2.0*dz*PR)

      belo(nj)=-m1*CL(j-1)*w(j-1) + m3*temp1 - m3*temp2
      diag(nj)=CL(j) + 2.0*m3*temp2
      if(tke(j) .ge. 1.0e-30) then
         diag(nj)=diag(nj) + C2e*CL(j)*dt*eps(j)/tke(j) -
c1e*theta*dt*(P+c3e*G)/tke(j)
      end if
      abov(nj)=m1*CL(J+1)*w(j+1) - m3*temp1 - m3*temp2

      T1=-m2*CL(j-1)*w(j-1) - m4*temp1 + m4*temp2
      T2=CL(j) - 2.0*m4*temp2
      if(tke(j) .ge. 1.0e-30) then
         T2=T2 + c1e*(1-theta)*dt*(P + c3e*G)/tke(j)
      end if
      T3= m2*CL(j+1)*w(j+1) + m4*temp1 + m4*temp2
      right(nj)=T1*eps(j-1) + T2*eps(j) + T3*eps(j+1)
!      if(q.eq.'y' .and. nt.ge.ntchk)  write(8,3)
nj,belo(nj),diag(nj),abov(nj),right(nj),G
   end do

!for the top cell
   nj=nj+1
   j=nzm1
   temp1= (CL(j)*eddy(j) - CL(j-1)*eddy(j-1) )/(2.0*cgm_e)
   temp2= CL(j)*eddy(j)/cgm_e
   P=CL(j)*eddy(j)*( (u(j) - u(j-1) )/(2.0*dz) )**2
   G=grav*eddy(j)*(cL(j) - cL(j-1))/(2.0*dz*PR)

   belo(nj)=-m1*CL(j-1)*w(j-1) + m3*temp1 - m3*temp2
   diag(nj)=CL(j) + 2.0*m3*temp2
   if(tke(j) .ge. 1.0e-30) then
      diag(nj)=diag(nj) + C2e*CL(j)*dt*eps(j)/tke(j) -
c1e*theta*dt*(P+c3e*G)/tke(j)
   end if
   s3=m1*CL(J)*w(j) - m3*temp1 - m3*temp2
   diag(nj)=diag(nj) - S3

   T1=-m2*CL(j-1)*w(j-1) - m4*temp1 + m4*temp2
   T2=CL(j) - 2.0*m4*temp2
   if(tke(j) .ge. 1.0e-30) then
      T2=T2 + c1e*(1-theta)*dt*(P + c3e*G)/tke(j)
```

```
   end if
   T3= m2*CL(j)*w(j) + m4*temp1 + m4*temp2
   right(nj)=T1*eps(j-1) + (T2-T3)*eps(j)
!   if(q.eq.'y' .and. nt.ge.ntchk)   write(8,4) nj, belo(nj), diag(nj),
right(nj), G

!now call the tridiagonal matrix solver and find the solution
   call tridiagonal(nj, belo,diag,abov, right, temp)

!update EPS
   do j=1,nj
      if (temp(j) .lt. 0.0)  temp(j) = 1.0e-8
      eps(j+jzb)=temp(j)
   end do
   eps(jzb)=eps_zb

!estimate EPS for the bottom area
   do j=1,jzb-1
      eps(j)=eps_zb*(j-0.5)*dz/((jzb-0.5)*dz)
   end do

!complete the update of k-e, now update the eddy viscosity and cneu
   do j=1,nzm1
      if( tke(j) .lt. 1.0e-22 .or. eps(j) .lt. 1.0e-22) then
         eddy(j)=vis
      else
         eddy(j)=vis + c_meu*tke(j)*tke(j)/eps(j)
      end if
      cneu(j)=eddy(j)
!      print*,'j, tke, eps, eddy = ',j, tke(j), eps(j), eddy(j)
   end do

end if  !(for 2nd order k-e model)

return
end


!-----------------------------------------------------------------------
Subroutine Shearvelocity(ustar, roughness)
!when the depth averaged velocity is small, i.e., less than 0.1 m/s, a simple
! formula (u*=0.077U) is used.
! Otherwise, using the least squares fitting with velocity data from previous
! step that are beween 0.1 m and 0.8 m above bed to find u_star.
! The above two values should be checked before simulating a flow, especially
! when the water depth is small.
!
!velocity profile in the constant shear stress zone is
!
! u(z)=(u*/k)log(z) + b, where k  : von Karman constant;
!                              b  : -(u*/k)log(zo)
!                              u* : shear velocity
!                              zo : roughness height
! this equation is a linear equation, and can be simplified as
```

```
!     y = mx + b where   m = U*/K  and x = log(z)
! the calculated u(j) is treated as experimental data y(j), and
! u(z) in the velocity profile equation is treated as y
! the corresponding tern, log(zc(j)), is treated as x(j), experimental data.
!
parameter (ng=48000, nt=70, nd=0)
implicit real*8 (a-h, o-z)
common/parame/ nz,nzm1,nzm2, dt,dz, depth, dydx_in, period,wave_amp,
wavek,cgm
common/array2/ z(ng),zc(ng), u(ng),dis(ng), conc(ng), cneu(ng), cL(ng), uave
common/leastq/ zb1, jzb1, ze, jze, iption_BC

!change the lower elevation for interpreting shear velocity and roughness
zb=zb1
jzb=int(zb/dz)

!the following numbers must be checked for any simulation
ze=0.8
if (ze .gt. 0.3*depth) then
   print*,'************************************************************'
   print*,'Please check  ze    because  ze                             '
   print*,'  should represent a near-bed zone for an open channel flows '
   print*,'************************************************************'
end if
jze=int(ze/dz)

!find the depth-averaged velocity
   uave=0.0
   do i=1,nzm1
      uave=uave + u(i)*dz
   end do
   uave=uave/depth

if (abs(uave) .ge. 0.1) then
!uses log law to find u*
   ny=jze - jzb + 1
   if( ny .le. 2) then
      print*,'increase vertical resolution to have more points in the log
layer'
      print*,'  usually this is caused by  dz  become too big or negative.
This '
      print*,'  measns a stability problem. Reduce dt would be the first
option'
      stop
   end if
   sumxi=0
   sumyi=0
   sumxiyi=0
   sumxi2=0
   do j=jzb,jze
      sumxi=sumxi + log(zc(j))
      sumyi=sumyi + u(j)
      sumxi2=sumxi2 + log(zc(j))*log(zc(j))
      sumxiyi=sumxiyi + log(zc(j))*u(j)
```

```
   end do
   slope=(ny*sumxiyi - sumyi*sumxi)/(ny*sumxi2 - sumxi*sumxi)
   !bed shear stress
   ustar=0.41*slope
   b_sec=(sumyi - slope*sumxi)/ny
   clog_zo=-0.41*b_sec/ustar
   roughness=dexp(clog_zo)
   if (roughness .le. 0.001) roughness=0.001
   if (roughness .ge. 0.2*depth) roughness=0.2*depth

else
!uses the depth averaged velocity to find u*.  Because
Tau=1/2*density_water*Cd*U^2
! which can be translated to (u*)^2=(Cd/2)U^2. if Cd = 0.012, then u*=0.077U
! this formulation may not be accurate, but it is only used when uave is
small,
! so that the influence should be small too.
   ustar=0.077*uave
   roughness=0.001
end if

!change ustar to reflect the density effect
ave_water_density=0
do i=1, nzm1
   ave_water_density = ave_water_density + cL(i)
end do
ave_water_density=ave_water_density/nzm1
shear=ustar*ustar*ave_water_density
ustar=sqrt( shear/cL(1) )

return
end
!
!--------------------------------------------------------------------
subroutine airy(period, depth, waveL, wavekh)
!  This function calculates the wave number and wave length based on
!  small amplitude wave theory.
!  period : in second.
!  depth  : in meter.
!  wavek  : in meter^-1
!
implicit real*8 (a-h, o-y)
grav=9.8
dol0=depth/(1.56*period*period)
if (dol0 .ge. 0.7) then
   waveL=1.56*period*period
   wavekh=6.2832*depth/wavel
else
   if (dol0 .ge. 0.01) then
      cgm=6.2832/period
      y=cgm*cgm*depth/grav
      sum=1.0 + 0.6666666666*y + 0.355555555*y*y +               &
               0.1608465608*y*y*y + 0.0632098765*y*y*y*y +         &
               0.0217540484*y*y*y*y*y*y + 0.00654078*y*y*y*y*y*y*y
```

```
      wavekh=sqrt(y*y + y/sum)
      waveL=6.2832*depth/wavekh
  else
!long wave
      c=sqrt(grav*depth)
      waveL=c*period
      wavekh=6.2832*depth/wavel
  end if
end if
!
return
end
!
!-----------------------------------------------------------------
subroutine tridiagonal(n, a,d,u,e, z)

!This subroutine solves the tridiagonal matrix equation DZ=E.
! where D is stored in 3 one-dimensional arrays.
!input:
! n is the order of matrix G.
! a(i) is the lower diagonal elements, from a(2) to a(n)
! d(i) is the diagonal elements, from d(1) to d(n)
! u(i) is the upper diagonal elements, from u(1) to u(n-1)
! e(i) is a column matrix, the right hand side of Eq. DZ=E
!                            from e(1) to e(n)
!
!return:
! z(i) is the returned answer, from z(1) to z(n)
!
!The complete picture is depicted as follows:
!
!       | d1 u1  0  0  0  0  0 ... | | z1 |   | e1 |
!       | a2 d2 u2  0  0  0  0 ... | | z2 |   | e2 |
!       | 0  a3 d3 u3  0  0  0 ... | | z3 |   | e3 |
!       | 0   0 a4 d4 u4  0  0 ... | | z4 | = | e4 |
!       |         ..........       | | .  |   | .  |
!       | 0   0  0 ....   am dm um | | zm |   | em |
!       | 0   0  0  0....    an dn | | zn |   | en |
!
!where m=n-1

parameter (ng=48000)
implicit real*8 (a-h, o-z)
dimension p(ng), q(ng), y(ng)
dimension a(1), d(1), u(1), e(1), z(1)

p(1)=d(1)
q(1)=u(1)/d(1)
do i=2, n-1
   p(i)=d(i)-a(i)*q(i-1)
   q(i)=u(i)/p(i)
end do
p(n)=d(n) - a(n)*q(n-1)
```

```
y(1)=e(1)/d(1)
do i=2, n
   y(i)=(e(i)-a(i)*y(i-1) )/p(i)
end do

z(n)=y(n)
do i=n-1, 1, -1
   z(i)=y(i)-q(i)*z(i+1)
end do

return
end
```

Appendix II. Selected Input Data Files for the 1-D Hydrodynamic Model.

Case 1.
I.project title
vertical 1-D strong turbulent flows, use k-e model, for steady non-stratified flows
II. total: total # of iteration
dt : size of time increment, in second.
nzm1 : # of grid cells in the vertical direction
dz : cell size in the vertical direction, in meter
iption_flow: 0, for Steady flow;
1, for wave/tidal flow with input d(eta)/dx time series;
2, for wave/tidal flow with internally calculated d(eta)/dx
3, for wave/tidal flow with input eta time series;
iption_BC: 0 for using no-slip boundary condition;
: 1 for using log velocity profile to estimate the velocity
an elevation z_b about the bottom as the bottom B.C.
period: wave period, if not a wave then use 0.0
wave_amp: wave amplitude, inmaterial if not for wave
dydx : water surface slope, for steady flow
nob_rec : to skip this # of time steps before saving everything once
ID : 0, use constant eddy viscosity,
1, use parabolic eddy viscosity
2, use k-e model to calculate eddy viscosity
phase : the phase difference caused by friction (degree)
offset: tidal amplitude difference between flood and ebb (m)
eddy_const: a constant used to simulate flow with constant viscosity for ID=0
100000 0.5 500 0.02 0 0 0.0 0.0 -0.00005 5000 2 0.0 0.0 0.0
III.input filename where the time series of water surface elevation are stored
IV.salinity dist. option: followed by salinity_top and Salinity_bottom.
1.Linear;
2.fixed s shape profile;
3.prepared input file; for option 3, given any two numbers
1 0.0 0.0
V.the input filename where the time series of salinity distribution are stored
VI.initial suspended sediment concentration profile (g/L).
1.a fixed exponential decreasing profile, include uniform profile.
ssc(z) = ssc_b * exp(ssc_rate*z). ssc_rate must be a negative number
2.a constant ssc (ssc_b) for the bottom half, and zero for the top half
3.prepared input file; For option 2, give zero for SSC_rate
For option 3, give zeros for SSC_b & SSC_rate
1 0.0 0.0
VII.the input filename where time series of SSC distributions are stored

Case 2.
I.project title
Vertical 1-D turbulent tidal flows, at Menai Strait, G.B., By Rippet et al., 2002
II.total: total # of iteration
dt : size of time increment, in second.
nzm1 : # of grid cells in the vertical direction
dz : cell size in the vertical direction, in meter
iption_flow: 0, for Steady flow;
1, for wave/tidal flow with input d(eta)/dx time series;
2, for wave/tidal flow with internally calculated d(eta)/dx
3, for wave/tidal flow with input eta time series;
iption_BC: 0 for using no-slip boundary condition;
: 1 for using log velocity profile to estimate the velocity
an elevation z_b about the bottom as the bottom B.C.
period: wave period, if not a wave then use 99
wave_h: wave height, in material if not for wave
dydx : water surface slope, for steady flow
nob_rec: to skip this # of time steps before saving
ID : 0, use constant eddy viscosity,
1, use parabolic eddy viscosity
2, use k-e model to calculate eddy viscosity
phase : the phase difference caused by wave attenuation (degree)
offset: constant pressure gradient caused by freshwater discharge
eddy_const: a constant used to simulate flow with constant viscosity for ID=0
600000 0.5 687 0.02 2 0 44640.0 2.25 0.000 4000 2 44.0 -0.000003 0.0
III.input filename for storing the time series of water surface elevation
IV.salinity dist. option: followed by salinity_top and Salinity_bottom
1.Linear;
2.fixed S shape profile;
3.prepared input file; For option 3, give any two numbers
1 0.0 0.0
V.the input filename for storing time series of salinity distribution
VI.options on SSC profile (g/L), followed by ssc_b for bottom SSC & ssc_rate.
1.a fixed exponential decreasing profile (include uniform profile,
ssc(z) = ssc_b * exp(ssc_rate*z). ssc_rate must be a negative number
2.a constant ssc (ssc_b) for the bottom half, and zero for the top half
3.prepared input file; For option 2, give zero for SSC_rate
For option 3, give zeros for SSC_b & SSC_rate
1 0.0 0.0
VII.the input filename where time series of SSC distributions are stored

Case 3.
I.project title
 vertical 1-D turbulent flows, use k-e model, salinity caused stratified flows
II. total: total # of iteration
     dt   : size of time increment, in second.
     nzm1 : # of grid cells in the vertical direction
     dz   : cell size in the vertical direction, in meter
iption_flow: 0, for Steady flow;
           1, for wave/tidal flow with input d(eta)/dx time series;
           2, for wave/tidal flow with internally calculated d(eta)/dx
           3, for wave/tidal flow with input eta time series;
 iption_BC: 0 for using no-slip boundary condition;
         : 1 for using log velocity profile to estimate the velocity
             an elevation z_b about the bottom as the bottom B.C.
    period: wave period, if not a wave then use 0.0
  wave_amp: wave amplitude, inmaterial if not for wave
     dydx : water surface slope, for steady flow
  nob_rec : to skip this # of time steps before saving everything once
     ID   : 0, use constant eddy viscosity,
           1, use parabolic eddy viscosity
           2, use k-e model to calculate eddy viscosity
    phase : the phase difference caused by friction (degree)
    offset: tidal amplitude difference between flood and ebb (m)
eddy_const: a constant used to simulate flow with constant viscosity for ID=0
 195000 1.0 150 0.02 0 0   0.0   0.0 -0.000001   5000   2   0.0   0.0   0.0
III.input filename where the time series of water surface elevation are stored
IV.salinity dist. option: followed by salinity_top and Salinity_bottom.
    1.Linear;
    2.fixed s shape profile;
    3.prepared input file;   for option 3, given any two numbers
1 5.0 7.0
V.the input filename where the time series of salinity distribution are stored
VI.initial suspended sediment concentration profile (g/L).
    1.a fixed exponential decreasing profile, include uniform profile.
      ssc(z) = ssc_b * exp(ssc_rate*z). ssc_rate must be a negative number
    2.a constant ssc (ssc_b) for the bottom half, and zero for the top half
    3.prepared input file; For option 2, give zero for SSC_rate
                          For option 3, give zeros for SSC_b & SSC_rate
 1 0.0 0.0
VII.the input filename where time series of SSC distributions are stored

Case 4.
I.project title
 vertical 1-D steady turbulent flows, use k-e model, SSC caused stratified flows
II. total: total # of iteration
     dt   : size of time increment, in second.
     nzm1 : # of grid cells in the vertical direction
     dz   : cell size in the vertical direction, in meter
   iption : 0, for Steady flow;
            1, for wave/tidal flow with input d(eta)/dx time series;
            2, for wave/tidal flow with internally calculated d(eta)/dx
            3, for wave/tidal flow with input eta time series;
 iption_BC: 0 for using no-slip boundary condition;
          : 1 for using log velocity profile to estimate the velocity
              an elevation z_b about the bottom as the bottom B.C.
    period: wave period, inmaterial if iption = 0 or 2
  wave_amp: wave height, inmaterial if not for wave flow
     dydx : water surface slope, for steady flow
   no_rec : to skip this # of time steps before saving once
     ID   : 0, use constant eddy viscosity,
            1, use parabolic eddy viscosity
            2, use k-e model to calculate eddy viscosity
    phase : the phase difference caused by friction (degree)
    offset: tidal amplitude difference between flood and ebb (m)
    eddy : the constant eddy viscosity, give a number even if ID <> 0
  125000 0.50 500 0.020   0 0 0.0   0.0 -0.00005   5000 2   0.0 0.0 0.0
III.input filename for storing the time series if water surface elevation
IV.options for salinity info: followed by Salinity_top and salinity_bottom.
    1. Linear;
    2. fixed s shape profile;
    3. prepared input file;   For option 3, give any two numbers
1 0.0 0.0
V.the input filename for storing time series of salinity distribution
VI.initial suspended sediment concentration profile (g/L).
    1.a fixed exponential decreasing profile (include uniform profile,
      ssc(z) = ssc_b * exp(ssc_rate*z). ssc_rate must be a negative number
    2.a constant ssc (ssc_b) for the bottom half, and zero for the top half
    3.prepared input file; For option 2, give zero for SSC_rate
                        For option 3, give zeros for SSC_b & SSC_rate
 1 3.0 -1.1
VII.the input filename where time series of SSC distributions are stored

Appendix III. Source Codes for Plotting the Simulation Results

```
%Program: profiles_plt.m
%MathLab M file for plot the results of program V1D.F90
%
% Program developed by Jerome P.-Y. Maa at
%       Virginia Institute of Marine Science, June 2009
clear all;  close all;  clc;

[name,path]=uigetfile('c:\1-d\eord\*.lis','Available Data files');
file=[path,name];
[fid, message]=fopen(file);
if fid == -1      disp([ message,' for', file]);   end

    % read data file name from the log file
    header=fgets(fid);
    header=fgets(fid);   ns=length(header);  nt_s=header(19:ns-1);
    nt=str2num(nt_s);
    header=fgets(fid);   ns=length(header);  dt_s=header(19:ns-1);
    dt= str2num(dt_s);

    header=fgets(fid);   ns=length(header);  nz_s=header(19:ns-1);
    nz=str2num(nz_s);
    header=fgets(fid);   ns=length(header);  depth_s=header(19:ns-1);
    depth= str2num(depth_s);

    header=fgets(fid);   ns=length(header);  Iption_s=header(19:ns-1);
    Iption_flow= str2num(Iption_s);
    header=fgets(fid);   ns=length(header);  iption_BC_s=header(19:ns-1);
    iption_BC= str2num(iption_BC_s);

    header=fgets(fid);   ns=length(header);  period_s=header(19:ns-1);
    period= str2num(period_s);
    header=fgets(fid);   ns=length(header);  waveH_s=header(19:ns-1);
    waveH= str2num(waveH_s);
    header=fgets(fid);   ns=length(header);  dydx_s=header(19:ns-1);
    dydx= str2num(dydx_s);

    header=fgets(fid);   ns=length(header);  ID_s=header(19:ns-1);
    ID= str2num(ID_s);
    header=fgets(fid);   ns=length(header);  phase_s=header(19:ns-1);
    phase= str2num(phase_s);
    header=fgets(fid);   ns=length(header);  offset_s=header(19:ns-1);
    offset= str2num(offset_s);
    header=fgets(fid);   ns=length(header);  eddy_const_s=header(19:ns-1);
    eddy_const= str2num(eddy_const_s);

    header=fgets(fid);   ns=length(header);  zb_s=header(19:ns-1);
    zb= str2num(zb_s);
    if iption_BC == 0    zb=0;   end
    disp(['iption_bd= ', iption_BC_s]);
    disp(['z_d for U= ', num2str(zb),' m']);
```

```
    k=0;

    figure(1);  clf;
    while ~feof(fid)
        k=k+1;
        fgets(fid);
        a=fscanf(fid,'%f %f %f %f %f %f %f %f', 8);    fgets(fid);
        step(k)=a(1);     %time step
        time_chk=a(1)*dt; %time
        depth=a(2);                dz=100*depth/nz;
        eta(k)=a(3);

        %read horizontal velocity
        fgets(fid);
        u=fscanf(fid,'%f', nz);      fgets(fid);    %in cm/s
        y=dz*(0:1:nz-1) + 0.5*dz;    y=0.01*y;

        disp(['time = ',num2str(time_chk),'s., dep=', num2str(depth)]);
        plot(u, y, '-g', 'linewidth', 1);  hold on;

        %read total viscosity
        header=fgets(fid);  coe_s=header(1:6);    coe1=str2num(coe_s);
        cneu=fscanf(fid,'%f', nz);     fgets(fid); cneu=cneu/coe1;

        %read tke for current
        header=fgets(fid);  coe_s=header(1:6);    coe2=str2num(coe_s);
        tke=fscanf(fid,'%f', nz);     fgets(fid);  tke=tke/coe2;

        %read eps for current
        header=fgets(fid);  coe_s=header(1:6);    coe3=str2num(coe_s);
        eps=fscanf(fid,'%f', nz);     fgets(fid);  eps=eps/coe3;

        %read salinity
        header=fgets(fid);
        sal=fscanf(fid,'%f', nz);     fgets(fid);

        %read suspended sediment concentration profiles
        header=fgets(fid);
        ssc=fscanf(fid,'%f', nz);     fgets(fid);

        %read water density
        header=fgets(fid);
        water_density=fscanf(fid,'%f', nz);     fgets(fid);

        %read vertical velocity profile
        header=fgets(fid);
        w=fscanf(fid,'%f', nz+1);     fgets(fid);
        % do next time step
    end
plot(u, y, '-b', 'linewidth', 2);  hold on;
title(name);
```

```
figure(2);  clf;
subplot(1, 2, 1);
plot(tke, y, '-r', 'linewidth', 2);  hold on;
text(0.5*max(tke), 0.5*depth,'r: tke');
plot(eps, y, '--k', 'linewidth', 2);  hold on;
text(0.4*max(tke), 0.4*depth,'k: eps');
xlabel('TKE (m/s)2 EPS');  ylabel('z (m)');
subplot(1, 2, 2);
plot(cneu, y, '-k', 'linewidth', 2);  hold on;
xlabel('cneu (m2/s)');  ylabel('z (m)');
title(name);

figure(3);  clf;
subplot(1, 2, 1);
plot(ssc, y, '-k', 'linewidth', 2);  hold on;
xlabel('SSC (g/L)');  ylabel('z (m)');
subplot(1, 2, 2);
plot(sal, y, '-k', 'linewidth', 2);  hold on;
xlabel('Salinity (psu)');
title(name);
```