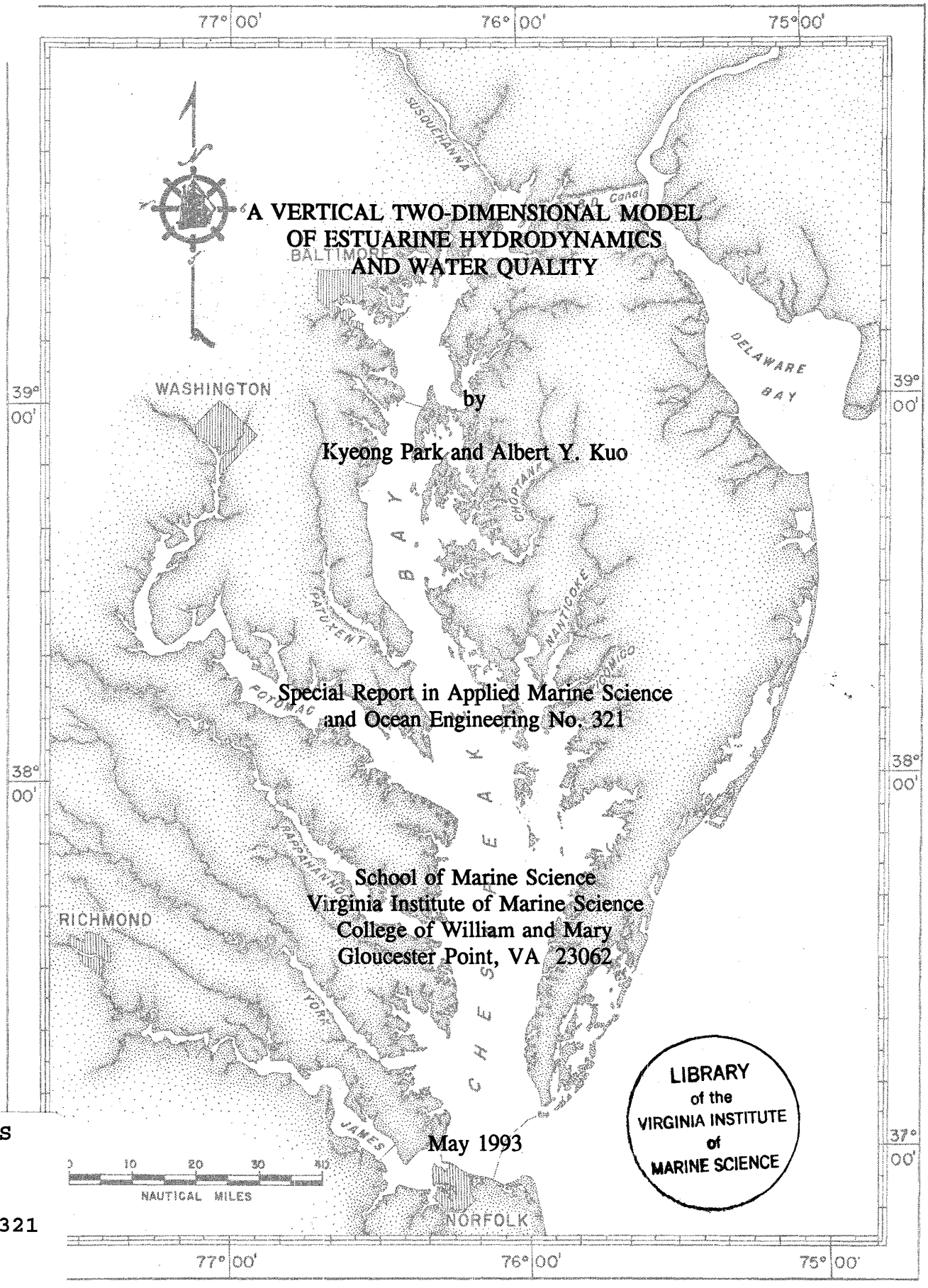
5-1-1993

A Vertical Two-Dimensional Model of Estuarine Hydrodynamics and Water Quality

Kyeong Park
Virginia Institute of Marine Science

Albert Y. Kuo
Virginia Institute of Marine Science



A VERTICAL TWO-DIMENSIONAL MODEL OF ESTUARINE HYDRODYNAMICS AND WATER QUALITY

by

Kyeong Park and Albert Y. Kuo

Special Report in Applied Marine Science
and Ocean Engineering No. 321

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

May 1993

A VERTICAL TWO-DIMENSIONAL MODEL OF ESTUARINE HYDRODYNAMICS AND WATER QUALITY

by

Kyeong Park and Albert Y. Kuo

Special Report in Applied Marine Science
and Ocean Engineering No. 321

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

May 1993

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

The mathematical model described in this report was developed as a part of the Hypoxia Project funded by the Virginia Chesapeake Bay Initiative Programs. We thank Dr. John M. Hamrick for his assistance in numerical methods, and Drs. Bruce J. Neilson and Carl F. Cerco for their discussions of water quality processes. The review of manuscript by them, and Drs. Roger L. Mann and John M. Brubaker is appreciated.

I. INTRODUCTION

Water quality problems in a tidal system generally result from a combination of physical and biogeochemical processes as human activities exert an excessive stress on the system. Therefore, water quality management in estuarine and coastal waters has received increased attention in recent years as human activities in these areas increase. Since many processes (physical transport, biogeochemical transformations, anthropogenic processes, etc) affect the water quality in the water column, it is difficult to assess the relative importance of each process. To this end, a mathematical model based on physical and biogeochemical principles is useful both to aid in understanding the system and to provide consistent, rational predictions of dynamic responses of the system to changes in specified factors. This report documents the development of a laterally integrated two-dimensional model consisting of hydrodynamic and water quality models.

Most mathematical models of water quality consist of a hydrodynamic model and a water quality model, linked either externally or internally. The hydrodynamic model generates time-series records of water velocity and surface elevation that are used in the simulation of physical transport processes in the water quality model. So far, advances in hydrodynamic modeling far exceed those in water quality modeling, mainly because the mathematical expressions describing biochemical processes, compared to those describing physical processes, are much cruder approximations of the processes being modeled, and thus subject to less accuracy. Also the field data that can be obtained from the current sampling techniques have finer spatial and temporal resolutions, and are of higher quality for the hydrodynamic parameters than the water quality ones. These differences in data quantity and quality make the calibration and verification of the hydrodynamic model easier and more reliable.

Primarily because of their simplicity, one-dimensional longitudinal models have

been widely employed for water quality management in lakes, estuaries and coastal waters (Shanahan & Harleman 1982; Thomann & Fitzpatrick 1982; Williams & Kuo 1984; Cerco et al. 1987; Ambrose et al. 1988; Kuo et al. 1991). These models are based on the one-dimensional momentum, continuity and salt balance equations and the mass balance equations for dissolved oxygen (DO) and other water quality parameters, assuming no vertical and lateral parameter variations. Despite its limitation, the one-dimensional approach has helped us to understand many aspects of the natural system and is still being employed for systems in which there is virtually no sectional variation.

The flow patterns in partially mixed estuaries, however, call for an explicit description of vertical structures of velocity and density in the numerical modeling of hydrodynamic transport (Wang et al. 1990). Besides, the vertical variations in DO often are large. These leave only two choices for the modeling of DO distribution in partially mixed estuaries: three-dimensional or laterally integrated two-dimensional models.

Estuarine flow and DO distribution are three-dimensional in nature. To simulate these completely, a three-dimensional model with time-dependent momentum and continuity equations, mass-balance equations with detailed description of the biochemical kinetics, and sources and sinks of all dissolved constituents is necessary. It seems that the state-of-the-art computer technology enables us to do three-dimensional simulations, particularly of hydrodynamics (Caponi 1977; Oey et al. 1985; Blumberg & Mellor 1987; Hamrick 1992). The current sampling capacity, however, cannot provide us with the quantity and quality of field data that are indispensable for the calibration and verification of the model, particularly the water quality model. Shanahan & Harleman (1982) have pointed out the necessity to consider both hydrodynamics and biochemistry in a compatible and even-handed fashion. In practice, the application of the three-dimensional water quality model to a natural body of water is complicated in terms of tractability and economy, and sometimes not warranted in terms of desired results.

Relatively narrow, straight estuaries are suitable for laterally integrated two-dimensional models assuming that all properties are uniformly distributed laterally across the estuary at their respective average values (Wang et al. 1990). The assumption of no lateral variation can be applied to estuaries where both tidal pumping and dispersion produced by tidal trapping are not important.

Tidal pumping refers to the situations where the flow is distributed differently in a cross-section during ebb and flood, and is principally caused by the Coriolis force and irregular bathymetry (Fischer et al. 1979). In simplifying the three-dimensional phenomena into two-dimensional frame work, some approximations are inevitable. The typical example is that a vertical two-dimensional model cannot account for the effects of the Coriolis force and irregular bathymetry.

The importance of the Coriolis force can be estimated using the nondimensional Kelvin number (K), which is $K = B/R_o$ where B is a typical width and R_o is the external Rossby radius of deformation, and $R_o = C_o/f$ where f is the Coriolis parameter and C_o is the wave celerity (Kundu 1990). The celerity (C_o) of a long surface gravity wave is equal to $(gH)^{1/2}$, where g is the gravitational acceleration and H is a typical depth. When K is of order unity or greater, Coriolis force becomes important. When K is less than 0.01, it is not important. The internal structure of the estuary can be more responsive to the Coriolis force because it is governed by motions with typically much smaller phase speeds. In this case, the internal R_o should be based on celerities of the internal waves, that depend upon the vertical structure of the stratification (Wang et al. 1990).

Tidal trapping is a term used to describe the effects of side embayments, small branching channels and shallow regions of an estuary, and occurs in almost all estuaries and in many tidal rivers (Fischer et al. 1979). A portion of water with its contained substances is stored temporarily in side channel, while the main flow proceeds along the estuary's major axis. If the recapture of the stored volume is out of phase with the main

flow, longitudinal mixing occurs when the tide reverses.

Numerous laterally integrated two-dimensional hydrodynamic models have been applied to estuaries (Bowden & Hamilton 1975; Blumberg 1975 and 1977; Hamilton 1977; Kuo et al. 1978; Wang & Kravitz 1980; Edinger & Buchak 1980 and 1981; Perrells & Karelse 1981). Among these, the one developed by Kuo et al. (1978) accounts for tidal trapping by modeling the side embayment area. This hydrodynamic model was modified and used in the present study to simulate the flow field and salinity distribution, and the corresponding water quality model was developed to simulate the distributions of DO and other related water quality parameters. The development of hydrodynamic model is described in Chapter 2, and that of water quality model in Chapter 3. The model operation, i.e., what the model actually does, is described in Chapter 4. The complete finite difference formulations are presented in Appendix A. Source programs are listed in Appendix B. The operating manual to control the input data files is provided in Appendix C.

II. DESCRIPTION OF THE HYDRODYNAMIC MODEL

2-1. Basic Equations

The laterally integrated two-dimensional hydrodynamic model developed by Kuo et al. (1978) was extensively modified and used to calculate the flow field and salinity. With a right-handed Cartesian coordinate system with the x-axis directing seaward and the z-axis directing upward (Fig. 2-1), the governing equations are,

$$\frac{\partial(uB)}{\partial x} + \frac{\partial(wB)}{\partial z} = q_p \tag{2-1}$$

$$\frac{\partial}{\partial t}(B_\eta \eta) + \frac{\partial}{\partial x}\int_{-H}^{\eta}(uB)\,dz = q \tag{2-2}$$

$$\frac{\partial(uB)}{\partial t} + \frac{\partial(uBu)}{\partial x} + \frac{\partial(uBw)}{\partial z} = -\frac{B}{\rho}\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}(A_x B\frac{\partial u}{\partial x}) + \frac{\partial}{\partial z}(A_z B\frac{\partial u}{\partial z}) \tag{2-3}$$

$$\frac{\partial p}{\partial z} = -\rho g \tag{2-4}$$

$$\frac{\partial(sB)}{\partial t} + \frac{\partial(sBu)}{\partial x} + \frac{\partial(sBw)}{\partial z} = \frac{\partial}{\partial x}(K_x B\frac{\partial s}{\partial x}) + \frac{\partial}{\partial z}(K_z B\frac{\partial s}{\partial z}) + S_o \tag{2-5}$$

where

t = time,

η = position of the free surface above mean sea level,

u & w = laterally averaged velocities in the x and z directions, respectively,

s = laterally averaged salinity,

B & B_η = river width and width at the free surface including side storage area,

H = total depth below mean sea level,

q_p = point source discharge,

q = lateral inflow including exchange with side storage area and q_p,

p & ρ = pressure and water density,

g = gravitational acceleration,

A_x & A_z = turbulent viscosities in the x and z directions, respectively,

K_x & K_z = turbulent diffusivities in the x and z directions, respectively,

S_o = source or sink of salt.

Equations 2-1 and 2-2 are the laterally and sectionally, respectively, integrated continuity equations for an incompressible flow. Equation 2-3 is the laterally integrated equation of motion for an incompressible but non-homogeneous flow, and represents the momentum balance along the longitudinal axis of an estuary. When the hydrostatic approximation, i.e., gravity is the dominant force in the vertical direction, is applied to the equation of motion in the z direction, the result is the hydrostatic equation (Eq. 2-4). Equation 2-5 is the laterally integrated mass-balance equation for salt. The density is related to the salinity by the simplified equation of state,

$$\rho = \rho_o(1 + ks) \tag{2-6}$$

where ρ_o is the density of freshwater and k is constant (7.5×10^{-4} ppt^{-1}). This equation is usually regarded as a satisfactory approximation because of the large horizontal gradients of salinity in estuaries (Hamilton 1977).

Equations 2-1 through 2-3 are solved by a finite difference method to obtain the time-varying solution of the free surface elevation (η) and the laterally averaged velocity fields (u and w). The pressure term (p) is evaluated using Eq. 2-4 with the water density (ρ) from Eq. 2-6, and salinity (s) using Eq. 2-5.

2-2. Boundary Conditions

2-2-1. Free surface

The condition of no mass flux through the free surface is effected by specifying zero vertical velocity and diffusion coefficient there. The wind stress term is used to

account for momentum introduced into the estuary. That is, at $z = \eta$,

$$\rho A_z \frac{\partial u}{\partial z} = C_D \rho_a U_w |U_w| \tag{2-7}$$

where C_D is the dimensionless drag coefficient (ca. 1.3×10^{-3}), ρ_a is the air density (1.2×10^{-3} g cm^{-3}) and U_w is the wind speed at the height of 10 m above surface.

2-2-2. Bottom

The condition of no mass flux through the bottom is effected by specifying zero vertical velocity and diffusion coefficient there. The bottom stress, which accounts for resistance by friction at the estuarine bottom, is calculated using a quadratic law. That is, at $z = -H$,

$$\tau_b = A_z \frac{\partial u}{\partial z} = C_d \cdot u_1 |u_1| \tag{2-8}$$

where τ_b is the bed shear stress, u_1 is the velocity at the first grid point above bottom and C_d is the friction coefficient on the order of 10^{-3}. Two options are provided in this model to calculate C_d, which are:

$$C_d = g\, n^2\, \Delta z^{-1/3} \tag{2-9}$$

or

$$C_d = \kappa^2 \left[\ln\left(\frac{\Delta z}{z_o}\right) - 1 \right]^{-2} \tag{2-10}$$

where n is the Manning's friction coefficient, κ is the von Karman constant (ca. 0.4), Δz is the bottom layer thickness and z_o is the bottom roughness height. Equation 2-10 is based upon the logarithmic velocity profile over the bottom layer. This velocity profile may be modified by acceleration and stratification, though the present formulation does not account for this modification. When estimating the total frictional force, Blumberg's approximation (1975), $\tau_b[1 + (\partial B/\partial z)^2]^{1/2} \approx \tau_b\, \partial B/\partial z$, is used.

2-2-3. Upstream boundary

The landward boundary (x = 0) of the model is chosen at the fall line or landward limit of tidal influence. It is assumed that the freshwater discharge and salinity are known at this boundary. They are, respectively, at x = 0,

$$u = \frac{Q(t)}{A} \quad \textit{and} \quad s = \textit{known} \tag{2-11}$$

where Q(t) is the freshwater discharge through the upstream boundary and A is the cross-sectional area there. Velocity (u) is assumed to be constant over depth at this boundary. Since the tide propagates farther upriver than the salt does, the salinity at the upstream boundary is specified to be zero.

2-2-4. Downstream boundary

The seaward boundary (x = ℓ) is located near the mouth of the estuary. The surface elevation is specified as a function of time either with harmonic functions or with the field measurements at this boundary.

In calculating velocities at the open boundary, the horizontal velocities are linearly extrapolated to a fictitious model transect outside the mouth, and the advective and diffusive terms are calculated over this fictitious model segment. An alternative approach is to assume that the dominant momentum balance in partially mixed estuaries takes place between the effects of surface slope, density gradient, and turbulent shear stress neglecting the horizontal advection and diffusion of momentum at the mouth. The difference between the results of the two methods is negligible, and the former method is adopted for this model.

Finally, a condition for the salt balance equation (Eq. 2-5) must be given. An "oceanic" or "bay" salinity is assumed to exist off the mouth of the estuary. During flood tide, the bay water is advected into the estuary, increasing the salinity at the mouth

until the bay salinity is achieved. Following the suggestion of Thatcher & Harleman (1972), a period of adjustment is allowed after the flow starts to flood and before the salinity at the mouth reaches the bay value. In the model, an input parameter is assigned for the specification of this adjustment period, and the salinity is assumed to increase linearly with time during this period. During ebb tide ($u > 0$), the horizontal salinity profile is assumed to have advected out of the mouth as a "frozen" pattern, i.e., neglecting the diffusion. That is, at $x = \ell$,

$$\frac{\partial s}{\partial t} = -\frac{\partial s}{\partial x} u \tag{2-12}$$

2-3. Turbulence Closure Model

The three-dimensional equations for volume, momentum and salt expressed in terms of the instantaneous flow field and salinity distribution, need to be averaged over a time interval longer than the turbulence time scale but much shorter than the tidal period, to derive the equations in terms of mean quantities. This time-averaging of momentum and mass balance equations gives rise to Reynolds stress and flux terms, which account for the diffusive exchange of momentum and salt, respectively, by small-scale turbulent motion. Then these time-averaged equations are laterally averaged to obtain the basic equations in Section 2-1-1. In a system with lateral inhomogeneity, the spatial-averaging in the lateral direction of the momentum and mass balance equations produces dispersion terms. These dispersion terms, owing to their similar mathematical appearance to Reynolds terms, are usually incorporated into the Reynolds terms and treated as one term to represent the mixing of momentum or salt. In Equations 2-3 and 2-5, therefore, A_x, A_z, K_x and K_z should be interpreted as lateral average values that take account of both diffusive and dispersive processes (Fischer 1967).

The system of equations would be closed only through the parameterization of

Reynolds stress and flux terms. Formulation of the Reynolds stress and flux terms mathematically, i.e., the turbulence closure model, has been, and still is, one of the most problematic steps for the laterally integrated two-dimensional or three-dimensional numerical models. The current practice ranges from a simple eddy viscosity approach to more complicated second order closure schemes (Blumberg 1986). The most reasonable way, with the current understanding of the turbulent mixing processes, is to choose cautiously the best method for an application and to calibrate it by comparison with field data (Wang et al. 1990). The oldest, yet still the most popular method of parameterizing the Reynolds terms, is the one based upon the eddy viscosity hypothesis. In Equations 2-3 and 2-5, the Reynolds terms are already expressed in terms of eddy viscosities (A_z and A_x) and diffusivities (K_z and K_x).

2-3-1. Vertical turbulent mixing coefficients

The vertical coefficients (A_z and K_z) are strongly affected by the flow velocity, the relative roughness of the flow channel and the vertical stratification. Their values can vary over several orders of magnitude at a fixed point in an estuary during a tidal cycle (Odd & Rodger 1978).

In the present model, the mixing length concept, first conceived by Prandtl, is used for A_z and K_z. For flow with two parallel plane boundaries in a wide channel of depth h, Rossby & Montgomery (1935) proposed the mixing length form:

$$A_z = \alpha Z^2 (1 - \frac{Z}{h})^2 \left| \frac{\partial u}{\partial z} \right| \phi_M(R_i) \qquad (2\text{-}13)$$

$$K_z = \alpha Z^2 (1 - \frac{Z}{h})^2 \left| \frac{\partial u}{\partial z} \right| \phi_S(R_i) \qquad (2\text{-}14)$$

where Z is the distance from the surface, α is a constant to be determined empirically, and ϕ_M and ϕ_S are the stability functions for momentum and mass, respectively. The local Richardson number (R_i), a measure of stability, is defined by,

$$R_i = -\frac{g}{\rho}\left(\frac{\partial \rho}{\partial z}\right)\left(\frac{\partial u}{\partial z}\right)^{-2} \quad (2\text{-}15)$$

The stability functions in Equations 2-13 and 2-14 account for the inhibition of the vertical exchange of momentum and mass (salt) by a stable density structure. Many studies have been performed for ϕ_M and ϕ_S, and there seem to be as many sets of formulations for them. Although the effect of the stratification on the vertical turbulent exchange is a function of R_i, the theory does not dictate what form this function should have (Perrells & Karelse 1981).

A. Munk-Anderson type formulation: A review of the various formulations for ϕ_M and ϕ_S shows the following general forms (Bowden & Hamilton 1975; Blumberg 1986),

$$\phi_M = (1 + \beta_M R_i)^{q_M} \quad (2\text{-}16)$$

$$\phi_S = (1 + \beta_S R_i)^{q_S} \quad (2\text{-}17)$$

where β_M, β_S, q_M and q_S are constants to be determined empirically. In the numerical modeling, these constants can be determined through model calibration. For a given condition, however, different sets of these constants can exist. That is, there are too many degrees of freedom. Results from previous studies were used to determine some of these constants. For q_M and q_S, this model adopts the values of Munk & Anderson (1948), $q_M = -1/2$ and $q_S = -3/2$, which indicates that a stable density gradient reduces the vertical turbulent exchange of mass more than that of momentum. In the numerical modeling, the constants including β_M and β_S are usually evaluated through comparison of model results with field measurements of a conservative substance such as salt. This practice makes the evaluation of β_M, which affects the vertical exchange of momentum, somewhat ambiguous. A priori there is no particular reason to consider that $\beta_M \neq \beta_S$. Since the difference in the effect of density structure on momentum and salt is already included in choosing the constants, q_M and q_S, it is assumed that β_M is equal to β_S.

When the wind effect is included, the resulting formulations for A_z and K_z are,

$$A_z = \alpha Z^2(1-\frac{Z}{h})^2 \left|\frac{\partial u}{\partial z}\right| (1+\beta R_i)^{-\frac{1}{2}} + \alpha_w \frac{H_w^2}{T}\exp(-\frac{2\pi}{L}Z) \quad (2\text{-}18)$$

$$K_z = \alpha Z^2(1-\frac{Z}{h})^2 \left|\frac{\partial u}{\partial z}\right| (1+\beta R_i)^{-\frac{3}{2}} + \alpha_w \frac{H_w^2}{T}\exp(-\frac{2\pi}{L}Z) \quad (2\text{-}19)$$

where H_w, T and L are the height, period and length, respectively, of wind-induced waves, and the constants (α, β and α_w) are determined through model calibration. The second terms in Equations 2-18 and 2-19 account for the turbulent mixing by wind-induced waves, and the first part, $\alpha_w \cdot H_w^2/T$, is the formulation proposed by Ichiye (1967). The depth dependence deduced by Pritchard (1960) using the James River data, consists of two parts; exponential decay with depth, $\exp(-2\pi Z/L)$, and shape function with depth, $Z(1-Z/h)$. In the study of the wind effects on turbulent mixing in lakes, Ford (1976) used the exponential decay but not the shape function. The shape function having a parabolic shape with depth renders the wind effect zero at the surface ($Z = 0$). The present model includes only the exponential decay term, which does not reduce the wind mixing just below the surface.

Pritchard (1960) also included the stability function in the wind mixing term to account for the inhibition of mixing due to stratification. The winds blowing over an estuary influence the velocity structure as well as the salinity structure. In a vertical two-dimensional model, however, the former responds only to the longitudinal component of wind, while the latter is not a function of wind direction. In the events of wind blowing across an estuary, therefore, the validity of using $\Delta u/\Delta z$ as an estimate of the velocity shear near the surface may be doubtful and so is that of using R_i calculated from $\Delta u/\Delta z$ as a measure of stability. Besides, the vertical density gradient may be negligible in the surface mixing layer that is mostly affected by wind, and thus the stability function is not applied to the wind mixing term in Equations 2-18 and 2-19.

For stable conditions ($R_i > 0$), Equations 2-18 and 2-19 are used to calculate A_z

and K_z. The unstable situation is usually taken care of by the convective adjustment process in which the unstable portion of water is neutralized by using average value (Bryan & Cox 1967; Blumberg 1986). This adjustment, if used in the model, does not allow the unstable condition to exist at all. In the present model, When $R_i < 0$, A_z and K_z are calculated with Equations 2-18 and 2-19 without including the effect of the stability functions, that is, assuming $R_i = 0$. This treatment, although it would allow the vertical instability to exist longer than it should, does allow the existence of the unstable conditions, which have been frequently observed in Virginia estuaries including James, York and Rappahannock estuaries (Brooks 1983).

B. Mellor and Yamada Level 2 turbulence model: Mellor & Yamada (1974 and 1982) developed a hierarchy of turbulence closure models depending upon the various degrees of approximations and simplifications made. The Levels 4, 3 and 2½ models require the solution of additional partial differential equations, and the simple Level 2 model gives a set of algebraic equations for A_z and K_z that are comparable to Equations 2-18 and 2-19. The present model has the Mellor and Yamada (M&Y) Level 2 turbulence model (Mellor & Yamada 1974 and 1982) as a second choice to estimate the vertical turbulent mixing coefficients.

The M&Y Level 2 model, like Equations 2-18 and 2-19, is based upon the local equilibrium, in which the turbulent kinetic energy produced by shear and buoyancy is balanced by the energy dissipation. The stability functions in the M&Y Level 2 model may be expressed as,

$$\phi_M = \left[\frac{(1 - R_{fc}^{-1} R_f)}{(1 - R_f)^{2/3}} \frac{(1 - R_{f1}^{-1} R_f)}{(1 - R_{f2}^{-1} R_f)} \right]^{\frac{3}{2}} \tag{2-20}$$

$$\phi_S = \frac{1}{P_r^N} \left[\frac{(1 - R_{fc}^{-1} R_f)^3}{(1 - R_f)^2} \frac{(1 - R_{f1}^{-1} R_f)}{(1 - R_{f2}^{-1} R_f)} \right]^{\frac{1}{2}} \tag{2-21}$$

where

$$R_{fc} = \frac{B_1 - 6A_1}{B_1 + 12A_1 + 3B_2} = \text{critical flux Richardson number} \quad (2\text{-}22)$$

$$R_{f1} = \frac{B_1(1 - 3C_1) - 6A_1}{B_1(1 - 3C_1) + 12A_1 + 9A_2} \quad (2\text{-}23)$$

$$R_{f2} = \frac{B_1 - 6A_1}{B_1 + 3A_1 + 3B_2} \quad (2\text{-}24)$$

$$R_f = \frac{1}{2}\frac{1}{P_r^N}\frac{R_{f1}}{R_{f2}}\left[R_i + P_r^N R_{f2} - \sqrt{R_i^2 + 2P_r^N\frac{R_{f2}^2}{R_{f1}}(\frac{R_{f1}}{R_{f2}} - 2)R_i + (P_r^N R_{f2})^2}\right] \quad (2\text{-}25)$$

Here, the flux Richardson number (R_f) is the ratio of the buoyant production to shear production of turbulent energy, i.e., $R_f = (\phi_S/\phi_M)\, R_i$, and P_r^N is the turbulent Prandtl number at neutral conditions. Using the constants in Mellor & Yamada (1982),

$$(A_1, B_1, A_2, B_2, C_1, P_r^N) = (0.92, 16.6, 0.74, 10.1, 0.08, 0.80) \quad (2\text{-}26)$$

the followings are obtained;

$$(R_{fc}, R_{f1}, R_{f2}, R_{ic}) = (0.191, 0.234, 0.223, 0.196) \quad (2\text{-}27)$$

$$R_f = 0.656\left(R_i + 0.178 - \sqrt{R_i^2 - 0.323R_i + 0.0318}\right) \quad (2\text{-}28)$$

where R_{ic} is the critical Richardson number above which mixing ceases to exist due to the stable stratification. From Equations 2-18 and 2-19, then, the vertical mixing coefficients, including the wind effect, are expressed as,

$$A_z = \alpha' Z^2(1 - \frac{Z}{h})^2\left|\frac{\partial u}{\partial z}\right|\left[\frac{(1 - 5.24R_f)}{(1 - R_f)^{2/3}}\frac{(1 - 4.27R_f)}{(1 - 4.48R_f)}\right]^{\frac{3}{2}} + \alpha_w\frac{H_w^2}{T}\exp(-\frac{2\pi}{L}Z) \quad (2\text{-}29)$$

$$K_z = \frac{\alpha'}{0.80}Z^2(1 - \frac{Z}{h})^2\left|\frac{\partial u}{\partial z}\right|\left[\frac{(1 - 5.24R_f)^3}{(1 - R_f)^2}\frac{(1 - 4.27R_f)}{(1 - 4.48R_f)}\right]^{\frac{1}{2}} + \alpha_w\frac{H_w^2}{T}\exp(-\frac{2\pi}{L}Z) \quad (2\text{-}30)$$

where α' and α_w are determined through model calibration. Equations 2-29 and 2-30 with R_f estimated from Eq. 2-28 are used to calculate A_z and K_z when $-10 < R_i < R_{ic} = 0.196$ (or $-11.8 < R_f < R_{fc} = 0.191$). When R_i is greater than R_{ic}, the background

value of 0.1 cm^2 sec^{-1} is used for A_z and K_z to avoid numerical instability. When $R_i < -10$, A_z and K_z are calculated with Equations 2-29 and 2-30 assuming $R_i = -10$. Note that since the stability functions (Equations 2-20 and 2-21) are defined and meaningful even for unstable condition, the R_i are allowed to be negative till -10 below which the R_i is assumed to be -10. This treatment is to prevent the calculation of mixing coefficients from overflowing in the source program.

2-3-2. Horizontal turbulent mixing coefficients

The horizontal mixing coefficients (A_x and K_x) are of the order of 10^5 of the vertical mixing coefficients (Dyer 1973). Results of diffusion measurements in English estuarine waters showed that K_x ranged from 10^4 to 10^6 cm^2 sec^{-1} (Talbot & Talbot 1974). Festa & Hansen (1976) studied the importance of exact values of A_x and K_x. Varying the momentum exchange coefficient from $A_x = A_z$ to $A_x = 10^6 A_z$ caused negligible effects on the results of their tidal average model. The change, however, in the mass exchange coefficient from $K_x = K_z$ to $K_x = 10^7 K_z$ did produce significant changes in their results.

The horizontal mixing terms, despite their relative insignificance in the momentum balance, are retained in the model for the stability consideration. The present model uses constant values for A_x and K_x and they are adjusted, within the range of 10^4 to 10^6 cm^2 sec^{-1}, through model calibration.

2-4. Method of Solution

2-4-1. Grid system and geometry

The system of equations is solved using finite difference method with a uniform grid of spatially staggered variables. The geometry of the grid system used in the model and the location of variables within the grid are shown in Fig. 2-1. The grid system has

η defined at the middle of each segment, while s, B, ρ and p at the center of the grid cell. The variables, w, A_z and K_z, are defined at the bottom face of the grid cell, while the grid containing u, A_x and K_x is staggered by half the segment length as these are defined at the grid cell walls. This staggered grid structure, also used by many other investigators, permits easy application of the boundary conditions and evaluation of the dominant pressure gradient force without interpolation or averaging (Blumberg 1977).

The geometry in the laterally integrated two-dimensional model is represented by the width at each depth at the center of each grid cell. A typical cross-section is illustrated in Fig. 2-2, which shows how the model accounts for the variation of width with depth.

2-4-2. Vertical integration

Large vertical gradients of variables (u and s) require a grid size that is much smaller in the vertical direction than in the horizontal direction. To accomplish this, the fluid motion is considered in horizontal slices with an exchange of momentum and mass between these slices. Integration over the height of the k^{th} layer can be performed by assuming that all variables are practically constant through the depth of any layer. Employing Leibnitz's rule and the boundary condition of no mass flux normal to the boundaries, the following equations can be obtained:

$$\frac{\partial \eta}{\partial t} = \frac{1}{B_1}[w_b B_b - \frac{\partial}{\partial x}(u_1 B_1 h_1) + q] \tag{2-31}$$

$$w_T = \frac{1}{B_T}[w_b B_b - \frac{\partial}{\partial x}(u_k B_k h_k) + (q_p)_k h_k] \tag{2-32}$$

$$\frac{\partial}{\partial t}(u_k B_k h_k) + \frac{\partial}{\partial x}(u_k B_k u_k h_k) + w_T u_T B_T - w_b u_b B_b$$
$$= -\frac{B_k h_k}{\rho_k}(\frac{\partial p}{\partial x})_k + \frac{\partial}{\partial x}[(A_x)_k B_k h_k (\frac{\partial u}{\partial x})_k] + (A_z B \frac{\partial u}{\partial z})_T - (A_z B \frac{\partial u}{\partial z})_b \tag{2-33}$$

$$\frac{\partial}{\partial t}(s_k B_k h_k) + \frac{\partial}{\partial x}(u_k B_k s_k h_k) + w_T s_T B_T - w_b s_b B_b$$
$$= \frac{\partial}{\partial x}[(K_x)_k B_k h_k (\frac{\partial s}{\partial x})_k] + (K_z B \frac{\partial s}{\partial z})_T - (K_z B \frac{\partial s}{\partial z})_b + S_{o,1} \quad (2\text{-}34)$$

where

u_k, B_k, h_k & $(q_p)_k$ = longitudinal velocity, width, thickness and point source discharge distributed to the k^{th} layer, respectively,

u_b, w_b & B_b = longitudinal velocity, vertical velocity and width at the bottom of a layer, respectively,

u_T, w_T & B_T = longitudinal velocity, vertical velocity and width at the top of a layer, respectively.

The term $S_{o,1}$ in Eq. 2-34 may represent the effect of the change in storage volume on the flanks (Fig. 2-2) with the change in tidal elevation. The subscript 1 means the interaction occurs only at the top layer. The storage in each segment will act as a source for the main channel on the falling tide and act as a sink on the rising tide. The salinity in the storage area remains the same on the falling tide, but changes on the rising tide because of the mixing between the incoming water from the main channel and the water in the storage area.

2-4-3. Finite difference treatment

A two-time level finite difference scheme is used to solve Equations 2-31 through 2-34. Three-time level (leapfrog) schemes are advantageous in dealing with the Coriolis term. They, however, tend to have problems of time-step splitting wherein the numerical solution alternates between physical and computational modes at each time step, even with viscous terms present (Roache 1972). The computational mode arises from using a second-order difference equation to approximate a first-order differential equation, and is a source of error (Haltiner & Williams 1980). The application of either

Euler-backward scheme (Blumberg 1977; Wang & Kravitz 1980) or weak time filter (Asselin 1972; Blumberg & Mellor 1987) has been used to eliminate the instability resulting from time-step splitting. In the present study, the vertical two-dimensional hydrodynamic model does not include the Coriolis term and thus a two-time level scheme is used to approximate the time derivative terms in Equations 2-31, 2-33 and 2-34.

Equation 2-31 is solved explicitly to obtain the free surface elevation (η), and Equations 2-33 and 2-34 are solved to get the longitudinal velocity (u) and salinity (s), respectively. Then, Eq. 2-32 is solved for the vertical velocity (w). To ensure stability,

1) the pressure gradient term in Eq. 2-33 is evaluated using η at a new time step,

2) the vertical mixing terms in Equations 2-33 and 2-34 are treated implicitly,

3) the horizontal mixing terms in Equations 2-33 and 2-34 are retained.

The implicit treatment of the vertical mixing terms results in the tri-diagonal matrix in the vertical direction, which is solved using a LINPACK subroutine (SGTSL.FOR) developed at the Argonne National Laboratory. Further development of the pressure term is given in Section 2-4-5. The complete finite difference forms of Equations 2-31 through 2-34 are presented in Appendix A.

In numerical modeling of the advection term, central and upwind (or upwind weighted) difference schemes are two routinely used ones. The upwind (or upwind weighted) difference scheme introduces the artificial numerical diffusion and thus suffers from severe inaccuracies, although the numerical diffusion makes the scheme stable and guarantees non-negative mass concentrations (Roache 1972). The use of fine grid spacing can alleviate this problem, but the consideration of computational aspects (speed and storage), particularly in the longitudinal direction, makes it impractical for the modeling of two or three-dimensional turbulent flows. On the other hand, the central difference scheme, which is second-order in accuracy and free of numerical diffusion, is

non-convergent particularly in regions where advection dominates diffusion (Roache 1972).

The unstable feature of a central difference scheme becomes more problematic in the mass balance equation (Eq. 2-34) than in the momentum balance equation (Eq. 2-33) in which the sink term (friction) tends to dissipate this oscillatory behavior. Primary dynamic balance in partially mixed estuaries is between the surface slope, density gradient and vertical gradient of turbulent shear stresses (Pritchard 1956). Since the horizontal and vertical advection terms are not important in the momentum equation, they are approximated with the central difference scheme in Eq. 2-33.

The dominant salt balance, however, takes place between horizontal advection and vertical turbulent diffusion (Pritchard 1954), making the accurate numerical treatment of horizontal advection essential to the faithful model behavior. While the relatively small vertical advection term can be treated with the central difference scheme in Eq. 2-34, the horizontal advective transport should be modeled with minimal introduction of artificial numerical oscillation or diffusion. The QUICKEST (Quadratic Upstream Interpolation for Convective Kinematics with Estimated Streaming Terms) scheme, that has been successfully applied to the modeling of the advection term (Leonard et al. 1978; Hall & Chapman 1985; Johnson et al. 1991), is used for the horizontal advection term in Eq. 2-34.

2-4-4. QUICKEST scheme

The scheme, derived by Leonard (1979), addresses the problems of estimating the concentrations at the cell walls by both central and upwind difference schemes. The QUICKEST scheme is based on a conservative control volume formulation and estimates cell wall concentrations with a quadratic interpolation using concentrations in two adjacent cells and that at the next upstream cell. This method has the properties of high

accuracy (third-order accurate in space) and significantly reduced numerical diffusion. Considering only advective transport, the estimated concentration at the right wall face of cell i (s_r^*) is,

$$s_r^* = \frac{1}{2}(s_i + s_{i+1}) - \frac{C_r}{2}(s_{i+1} - s_i) + \frac{1}{6}(C_r^2 - 1)CURV_r \tag{2-35}$$

where

$$C_r = \text{Courant number} = \frac{\Delta t}{\Delta x} u_{i+1} \tag{2-36}$$

$$CURV_r = \begin{cases} CURV_i & \text{for } u_{i+1} \geq 0 \\ CURV_{i+1} & \text{for } u_{i+1} < 0 \end{cases} \quad \text{where} \quad CURV_i = s_{i+1} - 2s_i + s_{i-1}$$

Equation 2-35 shows that the Courant number determines the contribution of each cell concentration to the cell wall concentration; the larger C_r (i.e., the faster the flow), the more contribution from the upstream cell.

2-4-5. Pressure gradient term

From the hydrostatic equation (Eq. 2-4), $\Delta p = -\rho g \Delta z$. Using this equation with the grid system,

$$p_{k-1} - p_k = -g(\rho_{k-1}\frac{h_{k-1}}{2} + \rho_k \frac{h_k}{2}) \tag{2-37}$$

which gives the pressure gradient averaged over the k^{th} layer,

$$(\frac{\partial p}{\partial x})_k = (\frac{\partial p}{\partial x})_{k-1} + \frac{g}{2}(h_{k-1}\frac{\partial \rho_{k-1}}{\partial x} + h_k \frac{\partial \rho_k}{\partial x}) \tag{2-38}$$

and this is the one appeared in Eq. 2-33. Therefore, with $(\partial p/\partial x)_1$ known, all other $(\partial p/\partial x)_k$'s can be calculated.

The pressure gradient can be decomposed into the barotropic and baroclinic terms,

$$\frac{\partial p}{\partial x} = \rho g \frac{\partial \eta}{\partial x} + g \int_z^{\eta} \frac{\partial \rho}{\partial x} dz \tag{2-39}$$

and the average pressure gradient for the top layer is,

$$(\frac{\partial p}{\partial x})_1 = g\rho_1 \frac{\partial \eta}{\partial x} + \frac{g}{2}(\eta + h_1) \frac{\partial \rho_1}{\partial x} \tag{2-40}$$

in which η^{n+1}, the surface elevation at a new time step, is used for the stability consideration.

2-4-6. Stability

The following stability criteria should be met to have a stable, convergent set of solutions,

$$\Delta t \leq \frac{\Delta x}{\sqrt{g\bar{h}}} \tag{2-41}$$

$$\Delta t \leq \frac{\Delta x}{u} \tag{2-42}$$

$$\Delta t \leq \frac{1}{2} \frac{(\Delta x)^2}{A_x} \tag{2-43}$$

where Δt is the time step and Δx is the segment length in the x direction. The Courant-Fredrick-Levy (CFL) condition (Eq. 2-41) arises from the use of η^{n+1} in evaluating the surface slope (in Eq. 2-40) in the momentum equation (Eq. 2-33). The use of η^n instead would make the numerical scheme used in this model unconditionally unstable. Equation 2-42 is the condition that the QUICKEST scheme for the horizontal advection term in the salt balance equation (Eq. 2-34) is stable up to C_r equal to one. The last diffusion condition (Eq. 2-43) is due to the explicit treatment of the horizontal mixing term in Eq. 2-33. The analogous condition for Eq. 2-34 is not necessary since A_x is always greater than, or equal to, K_x. The implicit treatment of the vertical mixing terms in Equations

2-33 and 2-34 removes the diffusion condition in the z direction. Of the three conditions, the CFL condition is the most limiting stability requirement. For $\Delta x = 2500$ m and $h = 10$ m, $\Delta t \leq 250$ sec.

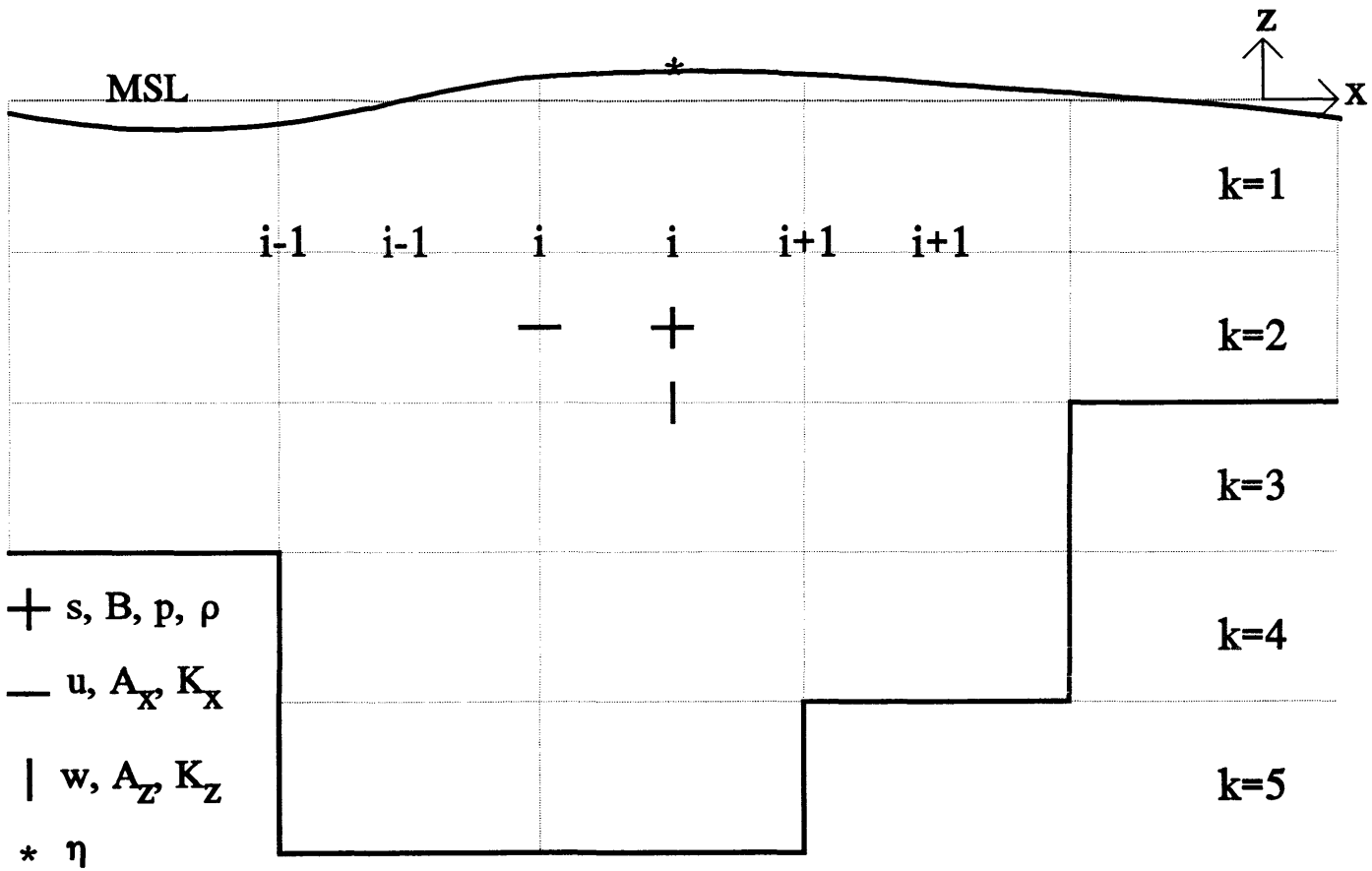


Figure 2-1. Grid pattern, location and indexing of variables.

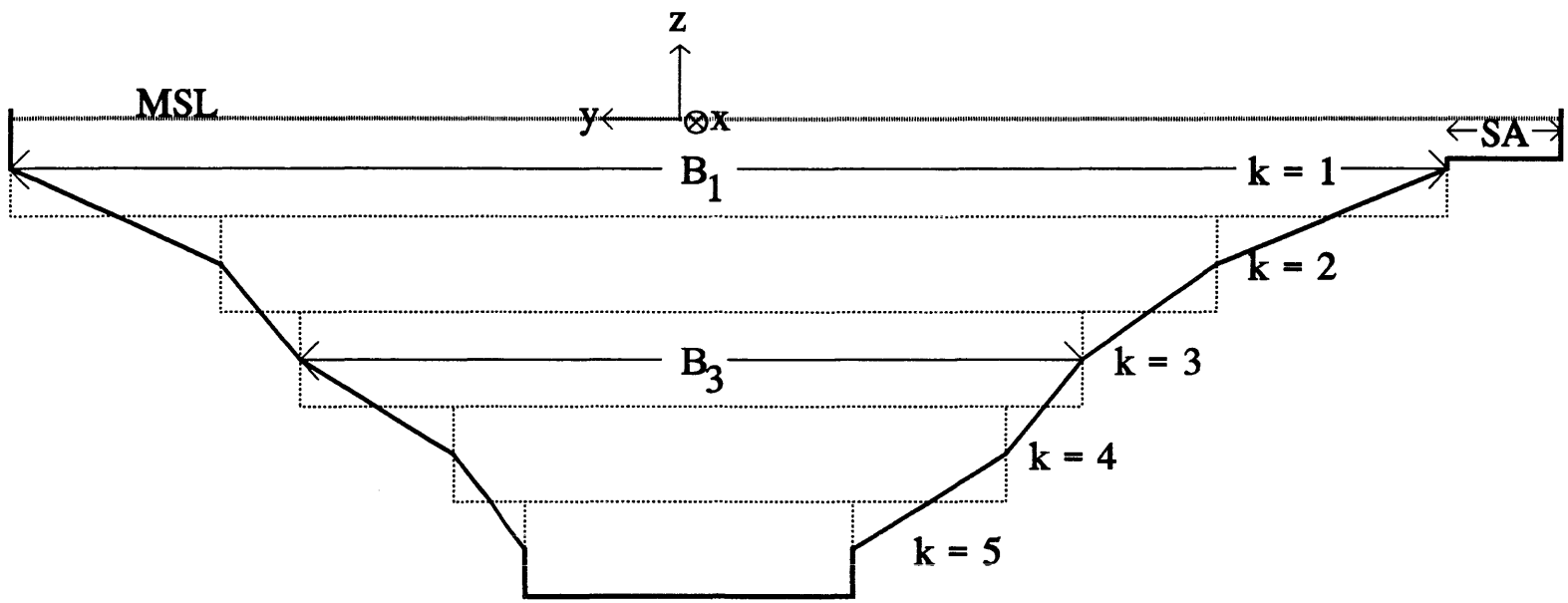


Figure 2-2. Cross-section looking downriver.

III. DESCRIPTION OF THE WATER QUALITY MODEL

3-1. Basic Equations

The water quality model is based on the equation describing the mass-balance of a dissolved or suspended substance in the water column. The equations that are solved by the finite difference method have the general form:

$$\frac{\partial(cB)}{\partial t} + \frac{\partial(cBu)}{\partial x} + \frac{\partial(cBw)}{\partial z} = \frac{\partial}{\partial x}(K_x B\frac{\partial c}{\partial x}) + \frac{\partial}{\partial z}(K_z B\frac{\partial c}{\partial z}) + BS_e + BS_i \qquad (3\text{-}1)$$

where

c = laterally averaged concentration of dissolved or suspended substance,

S_e = time rate of external addition (or withdrawal) of mass across the boundaries,

S_i = time rate of internal increase (or decrease) of mass by biochemical reaction processes.

The physical parameters (u, w, K_x, and K_z) in the physical transport terms in Eq. 3-1 are provided from the hydrodynamic model. The physical transport terms, both advective and diffusive, are identical for all dissolved and suspended substances, and treated in the same manner as those in the mass balance equation for salt (Eq. 2-5). The last two terms of Eq. 3-1 represent the external sources (or sinks) and internal sources (or sinks) due to the biochemical reactions, and differ for different substances.

In this study, the water quality model consists of eight interlinked components including phytoplankton population (Chl), organic nitrogen (N1), ammonia nitrogen (N2), nitrite-nitrate nitrogen (N3), organic phosphorus (P1), inorganic (ortho) phosphorus (P2), carbonaceous biochemical oxygen demand (CBOD) and DO (Fig. 3-1). Each of these water quality parameters can be represented by the same equation as Eq. 3-1, but with its own representations of external (S_e) and internal (S_i) source and sink

terms. Each rectangular box in Fig. 3-1 represents a component being simulated by the model. The arrows between components represent the biochemical transformation of one substance to the other. An arrow with one end unattached to a component (rectangular box) represents an internal source (or sink) due to the biochemical reaction or an external source (or sink). The mathematical expressions used in this model for the terms, S_e and S_i, for each of the eight components are the extension of the one-dimensional water quality model described in Kuo et al. (1991). They are presented in the following sections with emphases given to the modifications made.

One nutrient not included in this model is silica. Silica is a limiting nutrient only for diatoms and thus it is generally modeled only when diatoms are simulated as a separate phytoplankton group (Bowie et al. 1985). The present model uses the chlorophyll 'a' concentration to quantify the whole phytoplankton population. Besides, diatoms are not predominant in summer in the western shore tributaries and lower Chesapeake Bay (Morse 1947; Pattern 1963; Marshall 1967 and 1980; Marshall & Lacouture 1986). Since the goal of the present model is to simulate the summer conditions and to study hypoxia, the silica cycle is not included in this model.

3-2. Boundary Conditions

As in the hydrodynamic model, the boundary conditions need to be specified in the water quality model at four boundaries; free surface, bottom, upstream boundary and downstream boundary. At the free surface, the wind-induced DO reaeration is incorporated into the model using Eq. 3-9d (Banks & Herrera 1977). There is no other mass flux through the free surface. The mass fluxes at the bottom are specified specifically for each parameter by settling and benthic fluxes. The concentrations of the parameters to be modeled are assumed to be known at the upstream boundary (Eq. 2-11). The downstream boundary conditions are treated in the same manner as salinity

as described in Section 2-2-4.

3-3. Phytoplankton Population

The phytoplankton population occupies a central role in the schematic system of Fig. 3-1. It influences, to a greater or lesser extent, all of the remaining constituents. No simple aggregate measurement is entirely satisfactory to quantify the phytoplankton population (Ambrose et al. 1988). Because of the wealth of chlorophyll data available and the lack of alternative data sets, the concentration of chlorophyll 'a' is used as a measure of the phytoplankton population in this model.

The present model of chlorophyll (Chl) includes growth and nonpoint source inputs as sources, and respiration and mortality as sinks. For the settling of Chl, those settling down from the overlying layer act as a source, while those settling down to the bottom or to an underlying layer act as a sink for a layer. The mathematical representation is,

$$S_i = (G - R - P)\,Chl \tag{3-2a}$$

$$S_e = \frac{K_{Chl}}{\Delta z}(\lambda_1 Chl_{k-1} - Chl_k) + \frac{WChl}{V} \tag{3-2b}$$

where

λ_1 = 0 for k = 1 (at top layer),

1 for $2 \leq k \leq N$, and N is the number of layers at each segment,

Chl = concentration of chlorophyll 'a' ($\mu g\ l^{-1}$),

G = growth rate of phytoplankton (day^{-1}),

R = respiration rate of phytoplankton (day^{-1}),

P = mortality rate due to predation and other factors (day^{-1}),

K_{Chl} = settling rate of phytoplankton (cm day^{-1}),

Δz = layer thickness (cm),

WChl = external loading of Chl (μg day^{-1}) including nonpoint source,

V = layer volume (liter).

For the top layer, Δz is adjusted to account for the surface fluctuation.

A. Growth: Phytoplankton growth depends on nutrient availability, ambient light and temperature. A multiplicative relationship between temperature, light and nutrient limitations is assumed. The functional forms used in the model can be found in Bowie et al. (1985) and are as follow,

$$G = k_{gr}\theta_1^{T-20} L(I_a, I_s, k_e, Chl, \Delta z) \cdot N(N2, N3, P2) \quad (3\text{-}2c)$$

where

k_{gr} = optimum growth rate at 20°C (day^{-1}),

θ_1 = constant for temperature adjustment of growth rate,

T = temperature (°C),

L = attenuation of growth due to suboptimal lighting,

I_a = total daily solar radiation (langleys day^{-1}),

I_s = optimum solar radiation rate (langleys day^{-1}),

k_e = light extinction coefficient (cm^{-1}) at zero chlorophyll concentration,

N = attenuation of growth due to nutrient limitations.

N2, N3 & P2 = concentrations (mg l^{-1}) of ammonia nitrogen, nitrite-nitrate nitrogen and inorganic phosphorus, respectively,

1) Temperature: The exponential function of temperature adjustment with a reference temperature of 20°C is used for the growth. This function, based on the Arrhenius relationship, is used whenever temperature adjustment is needed in the model.

2) Light: The light effect (L) is based on the available solar energy and the attenuation of light through the water column. From the Beer-Lambert law and the Steele (1965) formulation, the light effect integrated over the layer depth may be expressed as,

$$L = \frac{e}{K_e \Delta z}\left[\exp\left\{-\frac{I_t}{I_s}\exp(-K_e[H_s+\Delta z])\right\} - \exp\left\{-\frac{I_t}{I_s}\exp(-K_e H_s)\right\}\right] \quad (3\text{-}2d)$$

$$K_e = k_e + K_{e,Chl} Chl \quad (3\text{-}2e)$$

$$I_t = I_a\left(\frac{24}{t_d - t_u}\right)\frac{\pi}{2}\sin\left(\pi\frac{t-t_u}{t_d-t_u}\right) \qquad if \quad t_u < t < t_d$$

$$= 0 \qquad if \quad t < t_u \ or \ t > t_d \quad (3\text{-}2f)$$

where

e = constant = 2.7183,

H_s = depth from the free surface to the top of the layer (cm),

K_e = light extinction coefficient (cm^{-1}) corrected for self-shading of plankton,

$K_{e,Chl}$ = light extinction due to self-shading of plankton (cm^{-1} per $\mu g\ l^{-1}$),

I_t = solar radiation at time t (langleys day^{-1}),

t = time of day (in hours),

t_u & t_d = time (in hours) of sunrise and sunset, respectively.

3) Nutrient: The nutrient effect (N) is based on the minimum limiting nutrient concept assuming nitrogen and phosphorus the major growth limiting nutrients. Using the fixed stoichiometry model based on conventional Monod or Michaelis-Menten kinetics, the nutrient effect may be expressed as,

$$N = \min\left\{\frac{N2+N3}{K_{mn}+N2+N3}, \frac{P2}{K_{mp}+P2}\right\} \quad (3\text{-}2g)$$

where

K_{mn} & K_{mp} = half-saturation concentrations (mg l^{-1}) for uptake of inorganic nitrogen and inorganic phosphorus, respectively.

B. Respiration: Respiration is the reverse of the photosynthesis process and thus

contributes to the reduction of algal biomass. The respiration rate (R) at which algae oxidize organic carbon to CO_2 is expressed as a function of temperature (Bowie et al. 1985).

$$R = R(20) \cdot \theta_2^{T-20} \tag{3-2h}$$

where

R(20) = respiration rate at 20°C (day^{-1}),

θ_2 = constant for temperature adjustment of respiration rate.

C. Mortality: The predatory mortality rate should be dependent on the time-variable herbivore population, which is in turn dependent upon the phytoplankton population. To avoid adding an additional trophic level to the model, however, both the predatory and non-predatory mortality rate (P) are combined into a single loss term, which is assumed to be a function of temperature.

$$P = P(20) \cdot \theta_3^{T-20} \tag{3-2i}$$

where

P(20) = mortality rate at 20°C (day^{-1}),

θ_3 = constant for temperature adjustment of mortality rate.

3-4. Organic Nitrogen

For the nitrogen cycle, three variables are modeled: organic nitrogen (N1), ammonia nitrogen (N2) and nitrite-nitrate nitrogen (N3). In the modeling of N1, the following processes are included; ammonification to N2, input due to algal respiration and death, settling, benthic release, and external loading. The mathematical representation is,

$$S_i = -\frac{K_{n12}N1}{K_{h12}+N1} + a_n(R+a_rP)F_nChl \quad (3\text{-}3a)$$

$$S_e = \frac{K_{n11}}{\Delta z}(\lambda_1 N1_{k-1} - N1_k) + \frac{BenN1}{\Delta z}\frac{B_k - \lambda_2 B_{k+1}}{B_k} + \frac{WN1}{V} \quad (3\text{-}3b)$$

where

λ_2 = 1 for $1 \leq k \leq N-1$,

0 for k = N (at bottom layer),

N1 = concentration of organic nitrogen (mg l^{-1}),

K_{n12} = ammonification rate of N1 to N2 (mg l^{-1} day^{-1}) = $K_{n12}(20) \cdot \theta_4^{T-20}$,

$K_{n12}(20)$ = ammonification rate at 20°C,

θ_4 = constant for temperature adjustment of ammonification rate,

K_{h12} = half-saturation concentration for ammonification (mg l^{-1}),

a_n = ratio of nitrogen to chlorophyll in phytoplankton (mg N per μg Chl),

a_r = fraction of consumed phytoplankton recycled by zooplankton,

K_{n11} = settling rate of N1 (cm day^{-1}),

F_n = fraction of metabolically produced nitrogen recycled to the organic pool,

BenN1 = benthic flux of N1 (g m^{-2} day^{-1}),

WN1 = external loading of N1 (mg day^{-1}) including point and nonpoint sources.

The model reads in the benthic fluxes of N1, N2, N3, P1, P2 and CBOD, and sediment oxygen demand (SOD) in g m^{-2} day^{-1} and multiplies them by 100 to correct for the dimensional consistency (note Δz is in cm).

3-5. Ammonia Nitrogen

The present model of N2 includes the following processes; ammonification from N1, input from algal respiration and death, benthic release and external loading as

sources, while nitrification to N3 and uptake by algae as sinks. The mathematical representation is,

$$S_i = -\frac{K_{n23}N2}{K_{h23}+N2}\frac{DO}{K_{nit}+DO} + \frac{K_{n12}N1}{K_{h12}+N1} + a_n(R+a_rP)(1-F_n)Chl - a_nG\cdot PR\cdot Chl \quad (3\text{-}4a)$$

$$S_e = \frac{BenN2}{\Delta z}\frac{B_k-\lambda_2 B_{k+1}}{B_k} + \frac{WN2}{V} \quad (3\text{-}4b)$$

where

K_{n23} = nitrification rate of N2 to N3 (mg l^{-1} day^{-1}) = $K_{n23}(20)\cdot\theta_5^{T-20}$,

$K_{n23}(20)$ = nitrification rate at 20°C,

θ_5 = constant for temperature adjustment of nitrification rate,

K_{h23} = half-saturation concentration for nitrification (mg l^{-1}),

K_{nit} = half-saturation concentration for oxygen limitation of nitrification (mg l^{-1}),

DO = concentration of dissolved oxygen (mg l^{-1}),

PR = preference of phytoplankton for N2 uptake, which is given by

$$= \frac{N2\ N3}{(K_{mn}+N2)(K_{mn}+N3)} + \frac{N2\ K_{mn}}{(N2+N3)(K_{mn}+N3)} \quad (3\text{-}4c)$$

BenN2 = benthic flux of N2 (g m^{-2} day^{-1}),

WN2 = external loading of N2 (mg day^{-1}) including point and nonpoint sources.

Nitrification by aerobic autotrophs (Nitrosomonas and Nitrobacter) requires oxic conditions, and thus the inhibition of nitrification by low DO is included (Ambrose et al. 1988). Nitrification may be formulated using first-order rate equation, which increases nitrification as ammonia concentration increases. At the high level of ammonia, however, nitrification being an enzymatic reaction by bacteria cannot proceed faster than its maximum rate and thus is zero order, i.e., independent of substrate concentration (Wild et al. 1971; Huang & Hopson 1974). Nitrification, therefore, is formulated using

Monod kinetics so as to have first-order kinetics at lower ammonia concentrations (limited by substrate availability) and to have zero-order kinetics at higher concentration (limited by bacteria availability). In the present model, this Monod or saturation-type formulation is used for all processes that transform one parameter to another including ammonification and mineralization of organic phosphorus. The fact that ammonia is preferably, to nitrate, taken up by phytoplankton for growth is incorporated into the model by using the ammonia preference term.

3-6. Nitrite-Nitrate Nitrogen

Included in the modeling of N3 are sources from nitrification, benthic release and external loading, and sinks due to algal uptake and denitrification. The mathematical representation is,

$$S_i = \frac{K_{n23} N2}{K_{h23} + N2} \frac{DO}{K_{nit} + DO} - a_n G(1 - PR) Chl \tag{3-5a}$$

$$S_e = -K_{n33} \frac{K_{h33}}{K_{h33} + DO} N3 + \frac{BenN3}{\Delta z} \frac{B_k - \lambda_2 B_{k+1}}{B_k} + \frac{WN3}{V} \tag{3-5b}$$

where

K_{n33} = denitrification rate (day^{-1}) = $K_{n33}(20) \cdot \theta_6^{T-20}$,

$K_{n33}(20)$ = denitrification rate at 20°C,

θ_6 = constant for temperature adjustment of denitrification rate,

K_{h33} = half-saturation concentration for denitrification (mg l^{-1}),

BenN3 = benthic flux of N3 (g m^{-2} day^{-1}),

WN3 = external loading of N3 (mg day^{-1}) including point and nonpoint sources.

Denitrification, which occurs only under extremely low DO conditions, is limited by DO availability. This limitation is incorporated into the model using Monod type expression

(Ambrose et al. 1988).

3-7. Organic Phosphorus

For the phosphorus cycle, two variables are modeled: organic phosphorus (P1) and inorganic phosphorus (P2). The present model of P1 includes the following processes; mineralization to P2, input due to algal respiration and death, settling, benthic release, and external loading. The mathematical representation is,

$$S_i = -\frac{K_{p12} P1}{K_{hp12} + P1} + a_p (R + a_r P) F_p Chl \tag{3-6a}$$

$$S_e = \frac{K_{p11}}{\Delta z}(\lambda_1 P1_{k-1} - P1_k) + \frac{BenP1}{\Delta z}\frac{B_k - \lambda_2 B_{k+1}}{B_k} + \frac{WP1}{V} \tag{3-6b}$$

where

P1 = concentration of organic phosphorus (mg l^{-1}),

K_{p12} = mineralization rate of P1 to P2 (mg l^{-1} day^{-1}) = $K_{p12}(20) \cdot \theta_7^{T-20}$,

$K_{p12}(20)$ = mineralization rate at 20°C,

θ_7 = constant for temperature adjustment of mineralization rate,

K_{hp12} = half-saturation concentration for mineralization (mg l^{-1}),

a_p = ratio of phosphorus to chlorophyll in phytoplankton (mg P per μg Chl),

K_{p11} = settling rate of P1 (cm day^{-1}),

F_p = fraction of metabolically produced phosphorus recycled to the organic pool,

BenP1 = benthic flux of P1 (g m^{-2} day^{-1}),

WP1 = external loading of P1 (mg day^{-1}) including point and nonpoint sources.

3-8. Inorganic (or Ortho) Phosphorus

The following processes are included to model P2; mineralization from P1, input

from algal respiration and death, uptake by algae for growth, settling, benthic release, and external loading. The mathematical representation is,

$$S_i = \frac{K_{p12} P1}{K_{hp12} + P1} + a_p (R + a_r P)(1 - F_p) Chl - a_p G \cdot Chl \quad (3\text{-}7a)$$

$$S_e = \frac{K_{p22}}{\Delta z}(\lambda_1 P2_{k-1} - P2_k) + \frac{BenP2}{\Delta z}\frac{B_k - \lambda_2 B_{k+1}}{B_k} + \frac{WP2}{V} \quad (3\text{-}7b)$$

where

K_{p22} = settling rate of P2 (cm day^{-1}),

BenP2 = benthic flux of P2 (g m^{-2} day^{-1}),

WP2 = external loading of P2 (mg day^{-1}) including point and nonpoint sources.

3-9. Carbonaceous Biochemical Oxygen Demand

In the modeling of CBOD, the followings are included; CBOD decay, input from algal death, settling, benthic release, and external loading. The mathematical representation is,

$$S_i = -K_c CBOD + a_c a_{co} (a_r P) Chl \quad (3\text{-}8a)$$

$$S_e = \frac{K_{BOD}}{\Delta z}(\lambda_1 CBOD_{k-1} - CBOD_k) + \frac{SOD}{\Delta z}\frac{K_{DO}}{K_{DO} + DO}\frac{B_k - \lambda_2 B_{k+1}}{B_k} + \frac{WBOD}{V} \quad (3\text{-}8b)$$

where

CBOD = concentration of carbonaceous biochemical oxygen demand (mg l^{-1}),

K_c = first-order decay rate of CBOD (day^{-1}) = $K_c(20) \cdot \theta_8^{T-20}$,

$K_c(20)$ = CBOD decay rate at 20°C,

θ_8 = constant for temperature adjustment of CBOD decay rate,

a_c = ratio of carbon to chlorophyll in phytoplankton (mg C per μg Chl),

a_{co} = ratio of oxygen demand to organic carbon recycled = 2.67,

K_{BOD} = settling rate of CBOD (cm day^{-1}),

SOD = sediment oxygen demand (g m^{-2} day^{-1}),

K_{DO} = half-saturation concentration for benthic flux of CBOD,

WBOD = external loading of CBOD (mg day^{-1}) including point and nonpoint sources.

Under anoxic conditions, SOD increases CBOD flux into the water column, which is incorporated into the model using Monod type expression.

3-10. Dissolved Oxygen

The present model of DO includes the following processes; sources from photosynthesis, reaeration through surface and external loading, and sinks due to decay of CBOD, nitrification, algal respiration and sediment oxygen demand (SOD). The mathematical representation is,

$$S_i = -K_c CBOD - a_{no} \frac{K_{n23} N2}{K_{h23} + N2} \frac{DO}{K_{nit} + DO} + a_c a_{co} (PQ \cdot G - \frac{R}{RQ}) Chl \qquad (3\text{-}9a)$$

$$S_e = (1 - \lambda_1) K_r (DO_s - DO) - \frac{SOD}{\Delta z} \frac{DO}{K_{DO} + DO} \frac{B_k - \lambda_2 B_{k+1}}{B_k} + \frac{WDO}{V} \qquad (3\text{-}9b)$$

where

a_{no} = ratio of oxygen consumed per unit of ammonia nitrogen nitrified = 4.57,

PQ = photosynthesis quotient (moles O_2 per mole C),

RQ = respiration quotient (moles CO_2 per mole O_2),

K_r = reaeration rate (day^{-1}),

DO_s = saturation concentration of DO (mg l^{-1}),

WDO = external loading of DO (mg day^{-1}) including point and nonpoint sources.

As a counterpart to CBOD flux term in Eq. 3-9b, SOD is inhibited by low DO

conditions.

The reaeration coefficient (K_r) includes reaeration by turbulence generated by bottom friction (O'Connor & Dobbins 1958) and that by surface wind stress (Banks & Herrera 1977), that is,

$$K_r(20) = \left[K_{ro} \sqrt{\frac{u_{eq}}{h_{eq}}} + W_{rea} \right] \frac{1}{\Delta z} \tag{3-9c}$$

where

$K_r(20)$ = reaeration rate at 20°C (day^{-1}),

K_{ro} = proportionality constant = 393.3 in CGS unit,

u_{eq} = weighted velocity over cross-section = $\Sigma(u_k B_k h_k)/\Sigma(B_k h_k)$,

h_{eq} = weighted depth over cross-section = $\Sigma(B_k h_k)/B_\eta$,

B_η = width at the free surface,

W_{rea} = wind-induced reaeration (cm day^{-1}),

$$= 72.8 U_w^{1/2} - 31.7 U_w + 3.72 U_w^2 \tag{3-9d}$$

where U_w is the wind speed (in m sec^{-1}) at the height of 10 m above surface. The reaeration rate is assumed to be temperature-dependent (Thomann & Mueller 1987),

$$K_r = K_r(20) \cdot \theta_9^{T-20} \tag{3-9e}$$

where θ_9 = constant for temperature adjustment of DO reaeration rate.

Saturated dissolved oxygen concentration (DO_s) is calculated as a function of water temperature and salinity (s in ppt) from a polynomial fitted to the tables of Green & Carritt (1967).

$$DO_s = 0.146244 \cdot 10^2 - 0.367134T + 0.4497 \cdot 10^{-2}T^2$$
$$- (0.966 \cdot 10^{-1} - 0.205 \cdot 10^{-2}T - 0.2739 \cdot 10^{-3}S)S \quad (3\text{-}9f)$$

3-11. Method of Solution

Equation 3-1 for each of eight water quality parameters is approximated with a finite difference scheme and solved for the time varying concentration field in the same way as the mass balance equation for salt (Eq. 2-5) in the hydrodynamic model. Wherever the equation of one water quality parameter involves other water quality parameters, the concentrations of the other parameters are expressed in terms of known values at the beginning of the time step. Therefore, the biochemical interaction terms in the coupled equations do not introduce additional unknowns for the finite difference equation of each individual water quality parameter over that of salt.

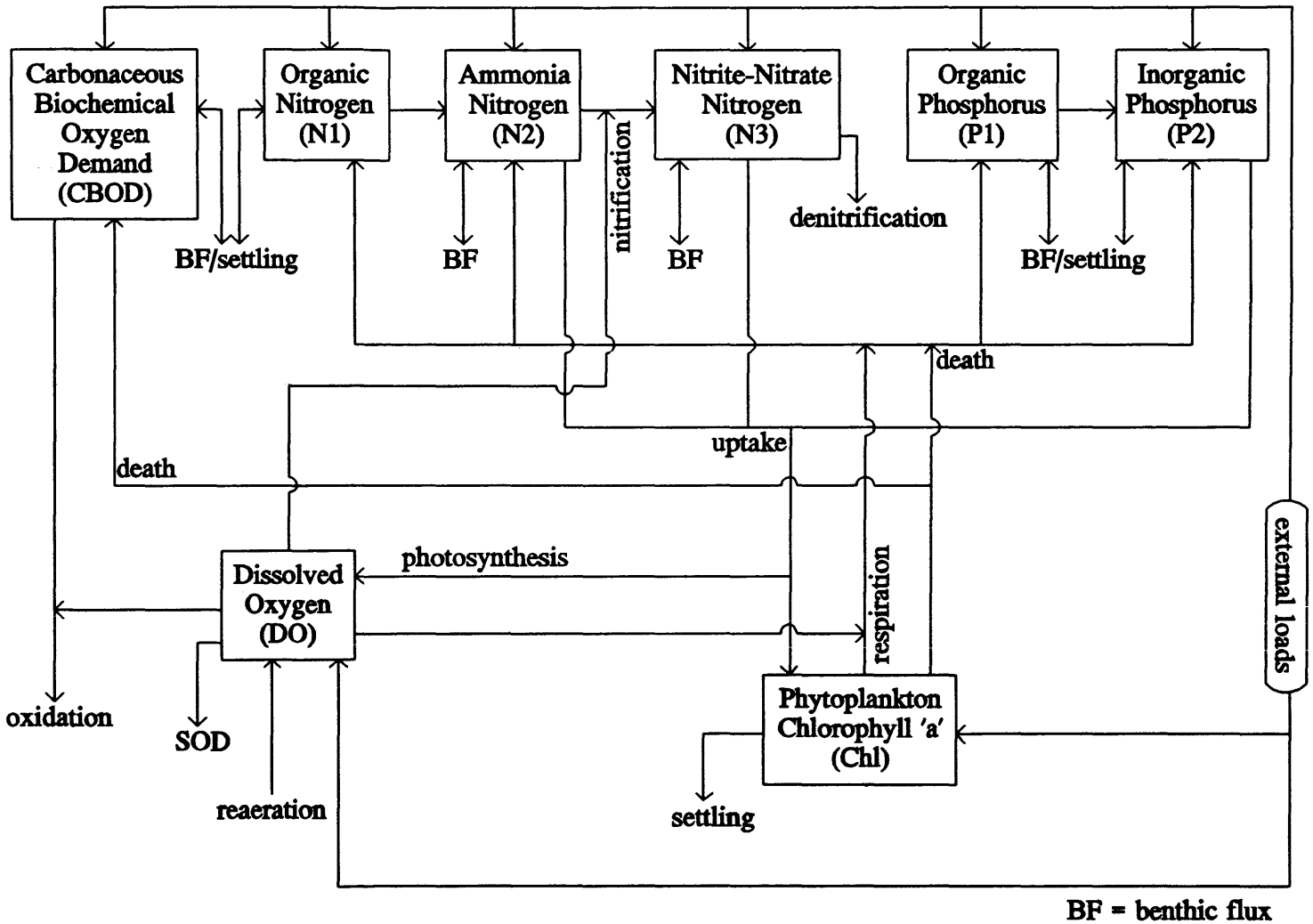


Figure 3-1. Schematic diagram of interacting water quality parameters.

IV. MODEL OPERATION (EXECUTION)

This chapter gives the general description of what the model does when one prepares the input files as described in Appendix C and runs the model. When applying this model to any system, data for geometry and calibrated constants are required. The geometric data include segmentation of the system (the number of segments and layers) and the main channel width, depth, side storage area, drainage area, etc (see Appendix C). The calibrated constants include Manning's friction coefficient (or bottom roughness height) and constants in vertical mixing coefficients (Eq. 2-18 or Eq. 2-29) in the hydrodynamic model, and a number of biogeochemical coefficients in the water quality model (see Appendix C).

In addition, for each model simulation, initial and boundary conditions are required. Initial conditions, the conditions at the beginning of a model simulation, need to be specified for velocities (u and w), surface elevation (η) and salinity (s) in the hydrodynamic model, and eight parameters (DO, Chl, CBOD, N1, N2, N3, P1 and P2) in the water quality model. To drive the model, the following external forcing variables are needed as boundary conditions:

(a) η and s at the downstream boundary,

(b) freshwater inflow at the upstream boundary (fall line),

(c) surface wind stress,

(d) concentrations for eight water quality parameters at both up and downstream boundaries,

(e) point and distributed source inputs for eight water quality parameters,

(f) benthic fluxes of nutrients and SOD,

(g) temperature,

(h) solar radiation.

When the model runs, it starts with initial conditions and calculates u, w, η, s and eight water quality parameters at each grid cell at each time step until the time that one specifies to stop the model run. It takes relatively short time for u, w and η to be stabilized, i.e., for the model solution to be independent of initial conditions. For the present model, it takes less than 10 tidal cycles for u, w and η to be stabilized. Therefore, any arbitrary conditions can be used as initial conditions for u, w and η ("cold start") as long as one does not use the model results for the first 10 tidal cycles. It, however, takes much longer time for masses to be stabilized. Considerable effort, thus, is required to specify the initial conditions for masses such as s and water quality parameters.

As the model calculation proceeds, it is constrained by the above-mentioned boundary conditions. The first three conditions (a-c) drive the hydrodynamic model to solve the continuity, momentum and salt balance equations, which gives the information of the physical transport processes. The following five conditions (d-h), with the information of the physical transport processes from the hydrodynamic model, drive the water quality model to solve eight mass balance equations, giving the distribution of eight water quality parameters. Detailed description of input data file organization is given in Appendix C.

At each time that one specifies to write the model results, the hydrodynamic model gives the following informations at every grid cell in four output files:

1) instantaneous u, w, η, s, vertical viscosity, and vertical diffusivity, and their averages over the previous tidal cycle,
2) range of surface current velocity, tidal range, and time of high tide and low tide during the previous tidal cycle,
3) time of, and salinity at, slackwaters during the previous tidal cycle.

At each time that one specifies to write the model results, the water quality model gives

the following informations at every grid cell in one output file:

1) instantaneous concentration of eight water quality parameters,

2) their daily averages, maximums and minimums over the previous day.

The times for the water quality output can be specified independent of those for the hydrodynamic output. Detailed description of output data file organization is given in Appendix C.

LITERATURE CITED

Ambrose, Jr., R.B., T.A. Wool, J.P. Connolly and R.W. Schanz. 1988. WASP4, a hydrodynamic and water quality model: model theory, user's manual and programmer's guide. EPA/600/3-87/039. Environmental Research Lab. (ERL), Office of Research and Development (ORD), US Environmental Protection Agency (EPA), Athens, GA. 297 pp.

Asselin, R. 1972. Frequency filters for time integrations. Monthly Weather Review, 100(6): 487-490.

Banks, R.B. and F.F. Herrera. 1977. Effect of wind and rain on surface reaeration. J. of the Environmental Engineering Division, American Society for Civil Engineers (ASCE), 103(EE3): 489-504.

Blumberg, A.F. 1975. A numerical investigation into the dynamics of estuarine circulation. Technical Report No. 91. Chesapeake Bay Institute, The Johns Hopkins Univ. 110 pp.

Blumberg, A.F. 1977. Numerical model of estuarine circulation. J. of the Hydraulic Division, ASCE, 103(HY3): 295-310.

Blumberg, A.F. 1986. Turbulent mixing processes in lakes, reservoirs and impoundments, pp. 79-104. In: W.G. Gray (ed.), Physics-based Modeling of Lakes, Reservoirs, and Impoundments, ASCE.

Blumberg, A.F. and G.L. Mellor. 1987. A description of a three-dimensional coastal ocean circulation model, pp. 1-16. In: N.S. Heaps (ed.), Three-dimensional Coastal Ocean Models, American Geophysical Union.

Bowden, K.F. and P. Hamilton. 1975. Some experiments with a numerical model of circulation and mixing in a tidal estuary. Estuarine and Coastal Marine Science, 3(3): 281-301

Bowie, G.L., W.B. Mills, D.B. Porcella, C.L. Campbell, J.R. Pagenkopf, G.L. Rupp, K.M. Johnson, P.W.H. Chan, S.A. Gherini and C.E. Chamberlin. 1985. Rates, constants, and kinetics formulations in surface water quality modeling (second edition). EPA/600/3-85/040. ERL, ORD, US EPA, Athens, GA. 455 pp.

Brooks, T.J. 1983. Rappahannock River slack water data report, temperature, salinity and dissolved oxygen 1970-1980. Data Report No. 18. Virginia Institute of Marine Science (VIMS), The College of William and Mary, VA. 40 pp.

Bryan, K. and M.D. Cox. 1967. A numerical investigation of the oceanic general circulation. Tellus, 19(1): 54-80.

Caponi, E.A. 1977. The simulation of estuarine circulations with a fully three-dimensional numerical model, pp. 332-346. In: M. Wiley (ed.), Estuarine Processes: Vol. II, Circulation, Sediments and Transfer of Material in the Estuary, Academic Press, Inc.

Cerco, C.F., A.Y. Kuo and P.V. Hyer. 1987. Model study of eutrophication in Virginia's Potomac embayments, pp. 119-123. In: Proceedings of the Tenth National Conference of the Coastal Society, New Orleans, LA, 1986.

Dyer, K.R. 1973. Estuaries: a physical introduction. John Wiley & Sons, Inc., NY. 140 pp.

Edinger, J.E. and E.M. Buchak. 1980. Numerical hydrodynamics of estuaries, pp. 115-146. In: P. Hamilton and K.B. Macdonald (eds.), Estuarine and Wetland Processes with Emphasis on Modeling, Plenum Press.

Edinger, J.E. and E.M. Buchak. 1981. Estuarine laterally averaged numerical dynamics: the development and testing of estuarine boundary conditions in the LARM code. Miscellaneous Paper EL-81-9. Prepared by J.E. Edinger Associates, Inc., for the US Army Engineer Waterways Experiment Station (WES), Corps of Engineers, Vicksburg, MS. 84 pp.

Festa, J.F. and D.V. Hansen. 1976. A two-dimensional numerical model of estuarine circulation: the effects of alternating depth and river discharge. Estuarine and Coastal Marine Science, 4(3): 309-323.

Fischer, H.B. 1967. The mechanics of dispersion in natural streams. J. of the Hydraulic Division, ASCE, 93(HY6): 187-216.

Fischer, H.B., E.J. List, R.C.Y. Koh and J. Imberger. 1979. Mixing in inland and coastal waters. Academic Press, Inc., NY. 483 pp.

Ford, D.E. 1976. Water temperature dynamics of Dimictic Lakes: analysis and predictions using integral energy concepts. Ph.D. Dissertation. Univ. of Minnesota, Minneapolis, Minn.

Green, E.J. and D.E. Carritt. 1967. New tables for oxygen saturation of seawater. J. of Marine Research, 25(2): 140-147.

Hall, R.W. and R.S. Chapman. 1985. Two-dimensional QUICKEST; solution of the depth-averaged transport-dispersion equation. Technical Report EL-85-3. US Army Engineer WES, Corps of Engineers, Vicksburg, MS. 29 pp.

Haltiner, G.J. and R.T. Williams. 1980. Numerical prediction and dynamic meteorology (second edition). John Wiley & Sons, Inc., NY. 477 pp.

Hamilton, P. 1977. On the numerical formulation of a time dependent multi-level model of an estuary, with particular reference to boundary conditions, pp. 347-364. In: M. Wiley (ed.), Estuarine Processes: Vol II, Circulation, Sediments and Transfer of Material in the Estuary, Academic Press, Inc.

Hamrick, J.M. 1992. A three-dimensional environmental fluid dynamics computer code: theoretical and computational aspects. Special Report in Applied Marine Science and Ocean Eng. (SRAMSOE) No. 317. VIMS, The College of William and Mary, VA. 63 pp.

Huang, C. and N.E. Hopson. 1974. Nitrification rate in biological processes. J. of the Environmental Engineering Division, ASCE, 100(EE2): 409-422.

Ichiye, T. 1967. Upper ocean boundary-layer flow determined by dye diffusion. Physics of Fluids Supplement, 10: S270-S277.

Johnson, B.H., K.W. Kim, R.H. Heath, H.L. Butler and B.B. Hsieh. 1991. User's guide for a three-dimensional numerical hydrodynamic, salinity and temperature model of Chesapeake Bay. Technical Report HL-91-20. US Army Engineer WES, Corps of Engineers, Vicksburg, MS. 41 pp.

Kundu, P.K. 1990. Fluid mechanics. Academic Press, Inc., NY. 638 pp.

Kuo, A.Y., M. Nichols and J. Lewis. 1978. Modelling sediment movement in the turbidity maximum of an estuary. Bulletin 111. Virginia Water Resources Research Center, Virginia Polytechnic Institute and State Univ., Blacksburg, VA. 76 pp.

Kuo, A.Y., B.J. Neilson and K. Park. 1991. A modelling study of the water quality of the upper tidal Rappahannock River. SRAMSOE No. 314. VIMS, The College of William and Mary, VA. 164 pp.

Leonard, B.P., G.J. Vachtsevanos and K.A. Abood. 1978. Unsteady-state, two-dimensional salinity intrusion model for an estuary, pp. 113-123. In: C. Brebbia (ed.), Applied Numerical Modelling, Pen Tech. Press.

Leonard, B.P. 1979. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. Computer Methods in Applied Mechanics and Engineering, 19: 59-98.

Marshall, H.G. 1967. Plankton in James River Estuary, Virginia. I. Phytoplankton in Willoughby Bay and Hampton Roads. Chesapeake Science, 8(2): 90-101.

Marshall, H.G. 1980. Seasonal phytoplankton composition in the lower Chesapeake Bay and Old Plantation Creek, Cape Charles, Virginia. Estuaries, 3(3): 207-216.

Marshall, H.G. and R. Lacouture. 1986. Seasonal patterns of growth and composition

of phytoplankton in the lower Chesapeake Bay and vicinity. Estuarine, Coastal and Shelf Sciences, 23(1): 115-130.

Mellor, G.L. and T. Yamada. 1974. A hierarchy of turbulence closure models for planetary boundary layers. J. of Atmospheric Sciences, 31(7): 1791-1806.

Mellor, G.L. and T. Yamada. 1982. Development of a turbulence closure model for geophysical fluid problems. Reviews of Geophysics and Space Science, 20(4): 851-875.

Morse, D.C. 1947. Some observations on seasonal variation in plankton population, Patuxent River, Maryland, 1943-1945. Publication No. 65. Chesapeake Biological Laboratory. 31 pp.

Munk, W.H. and E.R. Anderson. 1948. Notes on a theory of the thermocline. J. of Marine Research, 7(3): 276-295.

O'Connor, D.J. and W.E. Dobbins. 1958. Mechanism of reaeration in natural streams. Transactions of the American Society of Civil Engineers, 123(2934): 641-684.

Odd, N.V.M. and J.G. Rodger. 1978. Vertical mixing in stratified tidal flows. J. of the Hydraulics Division, ASCE, 104(HY3): 337-351.

Oey, L.-Y., G.L. Mellor and R.I. Hires. 1985. A three-dimensional simulation of the Hudson-Raritan estuary, Parts I-III. J. of Physical Oceanography, 15(12): 1676-1720.

Pattern, B.C. 1963. Plankton: optimum diversity structure of a summer community. Science, 140(3569): 894-898.

Perrells, P.A.J. and M. Karelse. 1981. A two-dimensional laterally averaged model for salt intrusion in estuaries, pp. 483-535. In: H.B. Fischer (ed.), Transport Models for Inland and Coastal Waters, Academic Press, Inc.

Pritchard, D.W. 1954. A study of the salt balance in a coastal plain estuary. J. of Marine Research, 13(1): 133-144.

Pritchard, D.W. 1956. The dynamic structure of a coastal plain estuary. J. of Marine Research, 15(1): 33-42.

Pritchard, D.W. 1960. The movement and mixing of contaminants in tidal estuaries, pp. 512-525. In: E.A. Pearson (ed.), Waste Disposal in the Marine Environment, Pergamon Press.

Roache, P.J. 1972. Computational fluid dynamics. Hermosa Publishers. 434 pp.

Rossby, C.G. and R.B. Montgomery. 1935. The layer of frictional influence in wind and ocean currents. Papers in Physical Oceanography and Meteorology,

3(3): 1-101.

Shanahan, P. and D.R.F. Harleman. 1982. Linked hydrodynamic and biochemical models of water quality in shallow lakes. Report No. 268. Ralph M. Parsons Lab., Aquatic Science and Environmental Eng., Dept. of Civil Eng., School of Eng., Massachusetts Institute of Technology (MIT). 279 pp.

Steele, J.H. 1965. Notes on some theoretical problems in production ecology, pp. 383-398. In: C.R. Goldman (ed.), Primary Productivity in Aquatic Environments. Mem. Ist. Ital. Idrobiol., Univ. of California Press, Berkeley.

Talbot, J.W. and G.A. Talbot. 1974. Diffusion in shallow seas and in English coastal and estuarine waters. Rapp. P.-v. Réun. Cons. int. Explor. Mer., 167: 93-110.

Thatcher, M.L. and D.R.F. Harleman. 1972. A mathematical model for the prediction of unsteady salinity intrusion in estuaries. Report No. 72-7. MIT, MA. 232 pp.

Thomann, R.V. and J.J. Fitzpatrick. 1982. Calibration and verification of a mathematical model of the eutrophication of the Potomac Estuary. HydroQual, Inc. Final Report to Dept. of Environmental Services, Washington, D.C. 500 pp.

Thomann, R.V. and J.A. Mueller. 1987. Principles of surface water quality modeling and control. Harper & Row, Publishers, NY. 644 pp.

Wang, D. and D.W. Kravitz. 1980. A semi-implicit two-dimensional model of estuarine circulation. J. of Physical Oceanography, 10(3): 441-454.

Wang, J.D., A.F. Blumberg, H.L. Butler and P. Hamilton. 1990. Transport prediction in partially stratified tidal water. J. of Hydraulics Engineering, ASCE, 116(3): 380-396.

Wild, Jr., H.E., C.N. Sawyer and T.C. McMahon. 1971. Factors affecting nitrification kinetics. J. of Water Pollution Control Federation, 43(9): 1845-1854.

Williams, S.A. and A.Y. Kuo. 1984. Water quality studies of Little Hunting Creek. SRAMSOE No. 268. VIMS, The College of William and Mary, VA. 250 pp.

APPENDIX A. FINITE DIFFERENCE EQUATIONS

In this Appendix, the finite difference formulations are given for Equations 2-31 through 2-34. The grid system is shown in Fig. 2-1 using the subscripts i and k to represent the number of intervals in the x and z directions, respectively. The subscripts 1 and 2 denote the time levels old and new, respectively.

A-1. Free Surface Elevation

Equation 2-31 is solved to get η at the top layer only, and thus k is always 1 in the following two equations,

$$\eta_{i,2} = \eta_{i,1} + \frac{\Delta t}{B_{i,k}+STB_i}\left[\frac{B_{i,k}+B_{i,k+1}}{2}w_{i,k,1} - \frac{1}{\Delta x}(TMP_{i+1}-TMP_i) + \frac{Q_{i,k}}{\Delta x}\right] \quad \text{(A-1)}$$

and

$$TMP_i = \frac{B_{i-1,k}+B_{i,k}}{2}\frac{(h_k + \eta_{i-1,1})+(h_k+\eta_{i,1})}{2}u_{i,k,1} \quad \text{(A-2)}$$

where

$Q_{i,k}$ = lateral volume inflow including the point source discharge and exchange with side storage area,

STB_i = equivalent width of the storage area = $SST_i/\Delta x$.

The inclusion of STB_i in the denominator of Eq. A-1 accounts for the effect of the side storage area, SST_i, and is explained in Section A-5.

A-2. Conservation of Longitudinal Momentum

The implicit treatment of the vertical viscosity term in Eq. 2-33 results in the equations containing the N×N tri-diagonal matrix where N is the number of layers at

each segment. The equations in the vertical direction may be expressed as,

$$[A][u] = [F] \tag{A-3}$$

That is,

$$\begin{bmatrix} D_1 & E_1 & 0 & . & . & . & 0 \\ C_2 & D_2 & E_2 & 0 & . & . & 0 \\ 0 & . & . & . & . & . & 0 \\ 0 & . & C_k & D_k & E_k & . & 0 \\ 0 & . & . & . & . & . & 0 \\ 0 & . & . & . & C_{N-1} & D_{N-1} & E_{N-1} \\ 0 & . & . & . & . & C_N & D_N \end{bmatrix} \begin{bmatrix} u_{i,1,2} \\ u_{i,2,2} \\ . \\ u_{i,k,2} \\ . \\ u_{i,N-1,2} \\ u_{i,N,2} \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ . \\ F_k \\ . \\ F_{N-1} \\ F_N \end{bmatrix} \tag{A-4}$$

and the non-zero elements, C_k, D_k, E_k and F_k, are given by,

$$E_k = -2\lambda_2 \frac{B_{i-1,k}+B_{i,k}+B_{i-1,k+1}+B_{i,k+1}}{(h_k+\lambda_1\eta_{i-1,2})+(h_k+\lambda_1\eta_{i,2})+2h_{k+1}}(A_{z(i-1,k)}+A_{z(i,k)}) \tag{A-5}$$

$$C_k = (1-\lambda_1)E_{k-1} \tag{A-6}$$

$$D_k = (B_{i-1,k}+B_{i,k})\frac{(h_k+\lambda_1\eta_{i-1,2})+(h_k+\lambda_1\eta_{i,2})}{\Delta t} - C_k - E_k \tag{A-7}$$

$$\begin{aligned} F_k = {} & (B_{i-1,k}+B_{i,k})\frac{(h_k+\lambda_1\eta_{i-1,1})+(h_k+\lambda_1\eta_{i,1})}{\Delta t}u_{i,k,1} - \frac{HAdvM_{i,k}-HAdvM_{i-1,k}}{4\Delta x} \\ & + \frac{\lambda_2}{4}(B_{i-1,k}+B_{i,k}+B_{i-1,k+1}+B_{i,k+1})(w_{i-1,k,1}+w_{i,k,1})(u_{i,k,1}+u_{i,k+1,1}) + 4\lambda_1\frac{B_{i-1,k}+B_{i,k}}{\rho_{i-1,k}+\rho_{i,k}}\tau_w \\ & - \frac{1-\lambda_1}{4}(B_{i-1,k-1}+B_{i,k-1}+B_{i-1,k}+B_{i,k})(w_{i-1,k-1,1}+w_{i,k-1,1})(u_{i,k-1,1}+u_{i,k,1}) \\ & + \frac{HDifM_{i,k}-HDifM_{i-1,k}}{2\Delta x^2} - 2\kappa u_{i,k,1}|u_{i,k,1}|(B_{i-1,k}+B_{i,k}-\lambda_2(B_{i-1,k+1}+B_{i,k+1})) \end{aligned} \tag{A-8}$$

$$- 2(B_{i-1,k}+B_{i,k})\frac{(h_k+\lambda_1\eta_{i-1,2})+(h_k+\lambda_1\eta_{i,2})}{\rho_{i-1,k}+\rho_{i,k}}\left(\frac{\partial p}{\partial x}\right)_{i,k,2} - \lambda_1 STSink_{i,k}$$

$$\begin{aligned} HAdvM_{i,k} &= \{(B_{i-1,k}+B_{i,k})(h_k+\lambda_1\eta_{i-1,1}+h_k+\lambda_1\eta_{i,1})u_{i,k,1} \\ &+ (B_{i,k}+B_{i+1,k})(h_k+\lambda_1\eta_{i,1}+h_k+\lambda_1\eta_{i+1,1})u_{i+1,k,1}\}(u_{i,k,1}+u_{i+1,k,1}) \end{aligned} \quad \text{(A-9)}$$

$$\begin{aligned} HDifM_{i,k} &= \{(B_{i-1,k}+B_{i,k})(h_k+\lambda_1\eta_{i-1,1}+h_k+\lambda_1\eta_{i,1})A_{x(i,k)} \\ &+ (B_{i,k}+B_{i+1,k})(h_k+\lambda_1\eta_{i,1}+h_k+\lambda_1\eta_{i+1,1})A_{x(i+1,k)}\}(u_{i+1,k,1}-u_{i,k,1}) \end{aligned} \quad \text{(A-10)}$$

where

τ_w = wind stress at the water-air interface,

λ_1 = 1 for k = 1 (at top layer),

0 for $2 \leq k \leq N$,

λ_2 = 1 for $1 \leq k \leq N-1$,

0 for k = N (at bottom layer).

The term $STSink_{i,k}$ in Eq. A-8 represents the sink of momentum due to the side storage area, SST_i, and is explained in Section A-5.

A-3. Conservation of Salt

The implicit treatment of the vertical diffusion term in Eq. 2-34 results in the equations containing the N×N tri-diagonal matrix in the vertical direction. In matrix notation,

$$[AS][s] = [FS] \quad \text{(A-11)}$$

which takes the same form as Eq. A-4, and the non-zero elements, CS_k, DS_k, ES_k and FS_k, are given by,

$$ES_k = -\lambda_2 \frac{B_{i,k} + B_{i,k+1}}{(h_k + \lambda_1 \eta_{i,2}) + h_{k+1}} K_{z(i,k)} \quad \text{(A-12)}$$

$$CS_k = (1 - \lambda_1) ES_{k-1} \quad \text{(A-13)}$$

$$DS_k = B_{i,k} \frac{h_k + \lambda_1 \eta_{i,2}}{\Delta t} - CS_k - ES_k \quad \text{(A-14)}$$

$$FS_k = B_{i,k} \frac{h_k + \lambda_1 \eta_{i,1}}{\Delta t} s_{i,k,1} - (HAdvS_{i,k} - HAdvS_{i-1,k}) + (HDifS_{i,k} - HDifS_{i-1,k})$$

$$+ \frac{\lambda_2}{4} (B_{i,k} + B_{i,k+1}) w_{i,k,1} (s_{i,k,1} + s_{i,k+1,1}) \quad \text{(A-15)}$$

$$- \frac{1 - \lambda_1}{4} (B_{i,k-1} + B_{i,k}) w_{i,k-1,1} (s_{i,k-1,1} + s_{i,k,1}) - \lambda_1 STSinkS_{i,k}$$

$$HAdvS_{i,k} = (B_{i,k} + B_{i+1,k}) \frac{(h_k + \lambda_1 \eta_{i,1}) + (h_k + \lambda_1 \eta_{i+1,1})}{4 \Delta x} u_{i+1,k,1} \cdot s^*_{i+1,k} \quad \text{(A-16)}$$

$$HDifS_{i,k} = (B_{i,k} + B_{i+1,k}) \frac{(h_k + \lambda_1 \eta_{i,1}) + (h_k + \lambda_1 \eta_{i+1,1})}{4 \Delta x^2} K_{x(i+1,k)} (s_{i+1,k,1} - s_{i,k,1}) \quad \text{(A-17)}$$

where $s^*_{i+1,k}$ is the salinity at the downstream wall face of cell given by Eq. 2-35. The term $STSinkS_{i,k}$ in Eq. A-15 represents the exchange of salt between the conveying main channel and the side storage area, and is explained in Section A-5.

A-4. Conservation of Water Mass

The laterally integrated continuity equation (Eq. 2-32) is solved to get w, and the finite difference formulation is,

$$w_{i,k,2} = \frac{\lambda_2}{B_{i,k}+B_{i,k+1}}\left[(B_{i,k+1}+B_{i,k+2})w_{i,k+1,2} + \frac{2}{\Delta x}(Q_p)_{i,k+1}\right.$$

$$\left. - \frac{h_{k+1}}{\Delta x}\{(B_{i,k+1}+B_{i+1,k+1})u_{i+1,k+1,2} - (B_{i-1,k+1}+B_{i,k+1})u_{i,k+1,2}\}\right] \qquad \text{(A-18)}$$

where Q_p is the volume discharge from point sources and $B_{i,k+2}$ is assumed to be $B_{i,k+1}$ for k = N-1. Equation A-18 is solved to get $w_{i,k,2}$ for $1 \leq k \leq N-1$, and the vertical velocity is zero at the bottom of the bottom layer, i.e., at k = N.

A-5. Treatment of Embayment as Storage Area

Dispersion produced by tidal trapping may be important for some parts of the estuary with substantial areas of side embayments, small branching channels, or shallow flanks. A portion of water with its contained substances is stored temporarily in side embayments, while the main flow proceeds along the estuary's major axis on the rising tide. If recapture of the stored volume is out of phase with the main flow on the falling tide, longitudinal mixing occurs when the tide reverses.

The present model calculates the longitudinal and vertical variations of the dependent variables along the conveying main channel but cannot calculate the variations in the side embayment area. It can account for the exchange of momentum and mass between the main channel and side embayment by treating the latter as temporary storage area. Since only the time variations of water surface elevation at the storage area affect the main channel conditions, and the total depth in the side embayment is usually shallow, the exchange of momentum and mass may be assumed to happen only at the top layer.

For the mass exchange of water, it is necessary to include the equivalent width of the storage area (STB_i) in the denominator of continuity equation (Eq. A-1) because the

surface elevation of the storage area rises and falls in accordance with the main channel. When the water enters into storage area (i.e., on the rising tide), the storage area in each segment will act as a sink for both momentum and mass, and the mass concentration in the storage area will change because of the mixing between the incoming water and the water in the storage area. When the water leaves the storage area (i.e., on falling tide), the storage area is assumed to act as a source for the mass only (not momentum), and the mass concentration in the storage area remains the same for conservative substances. For the non-conservative substances, the mass concentration on falling tide will be determined solely by the biochemical processes in the storage area.

For the momentum exchange, therefore, it is needed to include a sink term, $STSink_{i,1}$, in Eq. A-8 only on rising tide, that is,

$$STSink_{i,1} = 2(STB_{i-1} + STB_i)\frac{\Delta\eta}{\Delta t}u_{i,1,1}, \quad if\ \Delta\eta = \eta_{i-1,2} + \eta_{i,2} - \eta_{i-1,1} - \eta_{i,1} > 0 \qquad \text{(A-19)}$$

For the mass exchange of a dissolved substance like salt, a sink (or source) term, $STSinkS_{i,1}$, is needed to be added in Eq. A-15;

$$STSinkS_{i,1} = STB_i\frac{\eta_{i,2} - \eta_{i,1}}{\Delta t}s_{i,1,1} \qquad if\ \eta_{i,2} - \eta_{i,1} > 0 \qquad \text{(A-20a)}$$

$$STS_i = \frac{STS_i \cdot STH_{i,1} + s_{i,1,1}(\eta_{i,2} - \eta_{i,1})}{STH_{i,2}}$$

$$STSinkS_{i,1} = STB_i\frac{\eta_{i,2} - \eta_{i,1}}{\Delta t}STS_i \qquad if\ \eta_{i,2} - \eta_{i,1} < 0 \qquad \text{(A-20b)}$$

where

STS_i = salinity in the storage area,

$STH_{i,n}$ = depth at the storage area at time step n.

APPENDIX B. SOURCE PROGRAMS

The model, both hydrodynamic and water quality models, is written in FORTRAN 77 following the ANSI x3.9 (1978) except 'DO WHILE' and 'END DO'. This, however, is the extension of and thus understood by many FORTRAN 77 compilers such as "Lahey F77L-EM/32" compiler. The model source program consists of the following ten files;

1) MODEL-2D.FOR (page B-2) is the main control program,
2) HYD-2D.FOR (page B-11) has the subroutines for hydrodynamic model,
3) WQ-2D-IN.FOR (page B-30) has the subroutines for input/output of water quality model,
4) WQ-2D.FOR (page B-46) has the subroutines for the main part of water quality model,
5) SGTSL.FOR (page B-63) is a LINPACK subroutine to solve a general tri-diagonal matrix,
6) COMMON.INC (page B-65) has the common blocks for hydrodynamic model,
7) BLKDATA.INC (page B-65) has the block data for COMMON.INC,
8) COM-WQ1.INC (page B-66) has a common block for water quality model; it includes COMMON.INC,
9) COM-WQ2.INC (page B-66) has another common block for water quality model,
10) BLKD-WQ.INC (page B-66) has the block data for COM-WQ1.INC and COM-WQ2.INC.

This appendix lists these ten source programs.

B-1. MODEL-2D.FOR

```
      Program V2DModel
C: Last revised on 1/19/1993.
C: This program is to solve a vertical 2-dimensional hydrodynamic
C  (Continuity, and Momentum & Salt Balance Eqns), and water quality (Mass
C  Balance Eqns for 8 water quality parameters) models.
C: The source code consists of
C 1) MODEL-2D.FOR (this file): main control program,
C 2) HYD-2D.FOR: subroutines for hydrodynamic model (HYM),
C 3) WQ-2D-IN.FOR: subroutines for input/output of water quality model
C    (WQM),
C 4) WQ-2D.FOR: subroutines for main part of WQM,
C 5) SGTSL.FOR: a LINPACK subroutine to solve tridiagonal matrix,
C 6) COMMON.INC: common blocks for HYM,
C 7) BLKDATA.INC: block data for COMMON.INC,
C 8) COM-WQ1.INC: common block for WQM, which includes COMMON.INC,
C 9) COM-WQ2.INC: another common block for WQM,
C 10) BLKD-WQ.INC: block data for COM-WQ1.INC & COM-WQ2.INC.
C: Written in Fortran 77 (ANSI X3.9-1978) except DO WHILE & END DO, which
C  is the extension of many Fortran 77 compilers.
C: iX & iZ limit the maximum # of transect (longitudinal dir) & layer
C  (vertical dir), respectively.

      INCLUDE 'COM-WQ1.INC'
      Data IterN,NTSfM/2*0/, TinTC/0.0/, Day/1.0/

      OPEN(5, FILE='HYD.IN')
      OPEN(7, FILE='HYD.OUT')
      OPEN(12, FILE='TRANGE.OUT')
      OPEN(14, FILE='SLACK.OUT')
      OPEN(15, FILE='E.OUT')

  901 FORMAT(10I5)
      READ(5,901) iWQ,iSalt,iDisch,iTide,iSatDB,iPS,iUBC
      READ(5,901) iWave,iHAdv,iTurb,iBotSh,iSDBC
      IF (iWQ .EQ. 1)  THEN
        WRITE(7,801)'* Water quality submodel is run                  '
        OPEN(11,FILE='WQ.IN')
        OPEN(8, FILE='WQ.OUT')
       ELSE
        WRITE(7,801)'* No water quality simulation                    '
      END IF
      IF (iSalt .EQ. 1)  THEN
        WRITE(7,801)'* Salt is modeled                                '
       ELSE
        WRITE(7,801)'* Salt is not modeled                            '
      END IF
      IF (iDisch .EQ. 1)  THEN
        WRITE(7,801)'* Time-dependant freshwater discharge through UB '
        OPEN(10,FILE='FLOW.IN')
       ELSE
        WRITE(7,801)'* Steady freshwater discharge through UB         '
      END IF
      IF (iTide .EQ. 1)  THEN
        WRITE(7,801)'* Hourly tidal heights are used for DBC          '
        OPEN(6,FILE='TIDE.IN')
       ELSE
        WRITE(7,801)'* Tides generated by harmonics for DBC           '
      END IF
      IF (iSatDB .EQ. 1)  THEN
        WRITE(7,801)'* Unsteady salinity for downstream bdry condition '
        OPEN(9,FILE='S-DBC.IN')
```

```
       ELSE
        WRITE(7,801)'* Steady salinity for downstream bdry condition   '
      END IF
      IF (iPS .EQ. 1)  THEN
        WRITE(7,801)'* Daily varying point source discharges           '
        OPEN(13,FILE='PSQ.IN')
       ELSE
        WRITE(7,801)'* Timewise constant point source discharges       '
      END IF
      IF (iUBC .EQ. 1)  THEN
        WRITE(7,801)'* No momentum (only mass) flux from f.w. discharge'
       ELSE
        WRITE(7,801)'* Both momentum & mass flux from f.w. discharge   '
      END IF
      IF (iWave .EQ. 1)  THEN
        WRITE(7,801)'* Daily varying parameters for wind mixing        '
        OPEN(20,FILE='WIND-W.IN')
       ELSE
        WRITE(7,801)'* Timewise constant parameters for wind mixing    '
      END IF
      IF (iHAdv .EQ. 3)  THEN
        WRITE(7,801)'* QUICKEST scheme for horizontal advection in MBE '
       ELSE IF (iHAdv .EQ. 2)  THEN
        WRITE(7,801)'* Upwind weighted horizontal advection scheme     '
       ELSE
        STOP '** Error in input: iHAdv should be either 2 or 3 !!'
      END IF
      IF (iTurb .EQ. 3)  THEN
        WRITE(7,801)'* Munk & Anderson stability fcn for turbulence    '
       ELSE IF (iTurb .EQ. 2)  THEN
        WRITE(7,801)'* Mellor & Yamada Level 2 for turbulence          '
       ELSE
        STOP '** Error in input: iTurb should be either 2 or 3 !!'
      END IF
      IF (iBotSh .EQ. 3)  THEN
        WRITE(7,801)'* Use Manning coeff. to calc. bottom shear stress.'
       ELSE IF (iBotSh .EQ. 2)  THEN
        WRITE(7,801)'* use bottom roughness to calc. bottom shear stres'
       ELSE
        STOP '** Error in input: iBotSh should be either 2 or 3 !!'
      END IF
      IF (iSDBC .EQ. 3)  THEN
        WRITE(7,801)'* SUBR SOpBdry calculates S(MU,k,2).              '
       ELSE IF (iSDBC .EQ. 2)  THEN
        WRITE(7,801)'* S(MU,k,2) = SFLDM(k): do not use SUBR SOpBdry.  '
       ELSE
        STOP '** Error in input: iSDBC should be either 2 or 3 !!'
      END IF
  801 FORMAT(A50)
C*******************************************************************C
      CALL ReadM(DTT,rNTSiH,NTSiD,DTH)
      iTprn = NINT( Tout(IncP)/DTT )
      iTps1 = NINT( (Tout(IncP)-1.0)/DTT )
      CALL CalcGrid
      CALL ReadH(rNTSiH,Day,NS2PS)
      CALL HCalcC
      WRITE(7,'(//A31/)')'***** Hydrodynamic Output *****'
      IF (iWQ .EQ. 1)  THEN
        CALL WQC1
        CALL WQC2
        CALL WQIC
        CALL WQinput
        CALL GetWNPS
      END IF
```

```
C*********************************************************************C
C Point where iteration with time begins.
C: call WQInput at 0000 hrs of each day, ie, MOD(NTSfM,NTSiD)=0 if
C  iTdep=1.
C*********************************************************************C
      DO WHILE (IterN .LE. iTmax)
        IF (IterN .EQ. iTprn)  THEN
          CALL HOut(iTPs1)
          IncP = IncP + 1
          iTprn = NINT( Tout(IncP)/DTT )
          iTPs1 = NINT( (Tout(IncP)-1.0)/DTT )
        END IF
        IF (iWQ .EQ. 1)  THEN
          IF (IterN .EQ. iWQTP)  THEN
            CALL WOut
            IncW = IncW + 1
            iWQTP = NINT( WQout(IncW) / DTD )
            iWQTPs1 = NINT( (WQout(IncW)-1.0) / DTD )
          END IF
        END IF

        IterN = IterN + 1
        NTSfM = NTSfM + 1
        TinTC = IterN * DTT
        TinDay = IterN * DTD
        Hour = NTSfM * DTH
        IF (MOD(NTSfM,NTSiD) .EQ. 0)  THEN
          Day = TinDay + 1.0
          NTSfM = NTSfM - NTSiD
          CALL MidNite(Day,NS2PS)
          IF (iWQ .EQ. 1  .AND.  iTdep .EQ. 1)  CALL WQinput
        END IF
        IF (iDisch .EQ. 1)  THEN
          IF (NTSfM .LE. iFDTL)  THEN
            QatUB = QML1+ (QML2 - QML1)/rFDTL * REAL(NTSfM)
            CALL GetQLat
            IF (iWQ .EQ. 1)  CALL GetWNPS
          END IF
        END IF

        CALL HCalcAll(rNTSiH,IterN)
        IF (iTPs1.GE.0 .AND. IterN.GT.iTPs1)  CALL Hmean(TinTC)

        IF (iWQ .EQ. 1)  THEN
          CALL WCalcAll(IterN)
          IF (iWQTPs1 .GE. 0  .AND.  IterN .GT. iWQTPs1)  CALL WQmean
        END IF
      END DO

      CLOSE(5)
      CLOSE(6)
      CLOSE(7)
      CLOSE(8)
      CLOSE(9)
      CLOSE(10)
      CLOSE(11)
      CLOSE(12)
      CLOSE(13)
      CLOSE(14)
      CLOSE(15)
      CLOSE(16)
      CLOSE(17)
      CLOSE(18)
      CLOSE(19)
```

```
      CLOSE(20)
      STOP
      END
C***************
      SUBROUTINE MidNite(Day,NS2PS)
C*********************************************************************C
      INCLUDE 'COMMON.INC'
      Dimension xPSQ(iX)
      Data xPSQ/iX*0.0/

      IF (iTide .EQ. 1)  CALL ReadTD(Day)
      IF (iSatDB .EQ. 1)  THEN
        n = KB(MU)
        READ(9,9002) (SFLDM(k), k=1,n)
        WRITE(7,803)'* Salinity at DB at the         ', Day,
     *              ' th days:', (SFLDM(k), k=1,n)
      END IF
      IF (iPS .EQ. 1)  THEN
        WRITE(7,803)'* PS discharges (m**3/s) at the ', Day,
     *              ' th days:'
        DO 50 m=1,NS2PS
          READ(13,9003) i, xPSQ(i)
          WRITE(7,807) i, ':', xPSQ(i)
          n = KB(i)
          DO 51 k=1,n
            PSQ(i,k) = 1.0E6 * xPSQ(i) * B(i,k)/TotB(i)
 51       CONTINUE
 50     CONTINUE
      END IF
      IF (iDisch .EQ. 1)  THEN
        QML1 = QML2
        READ(10,9003) i, QML2
        WRITE(7,806)'* Discharge (m**3/s) at UB at the ', Day,
     *              ' th days = ', QML2
        QML2 = QML2 * 1.0E6
      END IF
      IF (iWave .EQ. 1)  THEN
        READ(20,9002) HtP, Wave2, WndSpd, WndDir
        WRITE(7,803)'* Wind mxing parameters at the  ', Day,
     *              ' th days:'
        WRITE(7,802)': Height**2 / Period (cm**2/s) = ', HtP,
     *              ': 2 * pi / Length (/cm)        = ', Wave2,
     *              ': Wind speed (m/s) at 10 m high= ', WndSpd,
     *              ': Wind dir (deg) ccw from east = ', WndDir
        Wave1 = CWave * HtP
        WndDir = WndDir * 1.74533E-2
        DO 60 i=ML,MU
          WS = WndSpd * COS(Paxis(i) - WndDir)
          WStrs(i) = Wdrag * WS * ABS(WS)
 60     CONTINUE
      END IF
 9002 FORMAT(9X, 10F8.3)
 9003 FORMAT(9X, I5, 9F7.3)
 9004 FORMAT(15I5)
  802 FORMAT(/, (A33, F10.3))
  803 FORMAT(/, A32, F8.3, A9, /, (9F8.3))
  806 FORMAT(/, A34, F8.3, A11, F8.3)
  807 FORMAT(I3, A1, F8.3)
      RETURN
      END
C***************
      SUBROUTINE HCalcAll(rNTSiH,IterN)
C*********************************************************************C
C Calculate all HYD variables for the (n+1)st (2 in the program) time step
```

```
C (2 time level, finite difference scheme is used).
C*************************************************************************C
      INCLUDE 'COMMON.INC'

      CALL HReset
      CALL CalcE
      CALL CEqn1(rNTSiH)
      CALL CalcP
      CALL AdvDif
      CALL MEqn
      IF (iSalt .EQ. 1)  THEN
        CALL SBEqn
        IF (iSDBC .EQ. 3)  THEN
          CALL SOpBdry
         ELSE
          n = KB(MU)
          DO 100 k=1,n
            S(MU,k,2) = SFLDM(k)
  100     CONTINUE
        END IF
      END IF
      IF (NS2Sal .EQ. 1)  THEN
        IF (IterN .EQ. iTSIC)  THEN
          print*, 'Specify salt IC at days ', TinDay, iTSIC
          DO 10 i=ML,MU
            n = KB(i)
            DO 11 k=1,n
              S(i,k,2) = SIC(i,k)
   11       CONTINUE
            StS(i) = SIC(i,1)
   10     CONTINUE
        END IF
      END IF
      CALL CEqn2
      RETURN
      END
C***************
      SUBROUTINE WCalcAll(IterN)
C*************************************************************************C
C: SUBR EqnN1, where HydN/HydNSt are calculated, be called prior to EqnN2.
C: SUBR EqnN2, where rNit/rNitSt are calculated, prior to EqnN3 & EqnDO.
C: SUBR EqnP1, where HydP/HydPSt, should be called prior to EqnP2.
C*************************************************************************C
      INCLUDE 'COM-WQ1.INC'

      CALL WReset
      CALL Phyto
      CALL EqnN1
      CALL EqnN2
      CALL EqnN3
      CALL EqnP1
      CALL EqnP2
      CALL EqnChl
      CALL EqnBOD
      CALL EqnDO
      IF (iWQDBC .EQ. 3)  THEN
        CALL WDnBdry
       ELSE
        n = KB(MU)
        DO 100 k=1,n
          N1(MU,k,2) = DnN1(k)
          N2(MU,k,2) = DnN2(k)
          N3(MU,k,2) = DnN3(k)
          P1(MU,k,2) = DnP1(k)
```

```
          P2(MU,k,2) = DnP2(k)
          Chl(MU,k,2) = DnChl(k)
          CBOD(MU,k,2) = DnBOD(k)
          DOC(MU,k,2) = DnDO(k)
100     CONTINUE
      END IF
      IF (NS2WQ .EQ. 1)  THEN
        IF (IterN .EQ. iTWQIC)  THEN
          print*, 'Specify WQ IC at days ', TinDay, iTWQIC
          DO 10 i=ML,MU
            n = KB(i)
            DO 11 k=1,n
              N1(i,k,2) = rN1IC(i,k)
              N2(i,k,2) = rN2IC(i,k)
              N3(i,k,2) = rN3IC(i,k)
              P1(i,k,2) = P1IC(i,k)
              P2(i,k,2) = P2IC(i,k)
              Chl(i,k,2) = ChlIC(i,k)
              CBOD(i,k,2) = BODIC(i,k)
              DOC(i,k,2) = DOIC(i,k)
 11         CONTINUE
            StN1(i) = rN1IC(i,1)
            StN2(i) = rN2IC(i,1)
            StN3(i) = rN3IC(i,1)
            StP1(i) = P1IC(i,1)
            StP2(i) = P2IC(i,1)
            StChl(i) = ChlIC(i,1)
            StBOD(i) = BODIC(i,1)
            StDO(i) = DOIC(i,1)
 10       CONTINUE
        END IF
      END IF
      RETURN
      END
C***************
      SUBROUTINE Hmean(TinTC)
C*********************************************************************C
C Calculate the sum of Ada, U, S, W, EPz & Ez
C and the maximum & minimum Ada and surface U over the last tidal cycle.
C*********************************************************************C
      INCLUDE 'COMMON.INC'

      AvgN = AvgN + 1.0
      DO 2100 i=ML,MU
        n = KB(i)
        Tavg(i) = Tavg(i) + Ada(i,2)
        DO 2101 k=1,n
          Uavg(i,k) = Uavg(i,k) + U(i,k,2)
          Savg(i,k) = Savg(i,k) + S(i,k,2)
          Wavg(i,k) = Wavg(i,k) + W(i,k)
          EPzAvg(i,k) = EPzAvg(i,k) + EPz(i,k)
          EzAvg(i,k) = EzAvg(i,k) + Ez(i,k)
2101    CONTINUE
        IF (Ada(i,2) .GT. Amax(i))  THEN
          Amax(i) = Ada(i,2)
          THI(i) = TinTC
        END IF
        IF (Ada(i,2) .LT. Amin(i))  THEN
          Amin(i) = Ada(i,2)
          TLO(i) = TinTC
        END IF
        IF (U(i,1,2) .GT. Umax(i))  Umax(i) = U(i,1,2)
        IF (U(i,1,2) .LT. Umin(i))  Umin(i) = U(i,1,2)
        IF (U(i,1,1) .GT. 0.0  .AND.  U(i,1,2) .LE. 0.0)  THEN
```

```
          TSBF(i) = TinDay
          n = KB(i)
          DO 2102 k=1,n
            SSBF(i,k) = S(i,k,2)
 2102     CONTINUE
        END IF
        IF (U(i,1,1) .LT. 0.0  .AND.  U(i,1,2) .GE. 0.0)  THEN
          TSBE(i) = TinDay
          n = KB(i)
          DO 2103 k=1,n
            SSBE(i,k) = S(i,k,2)
 2103     CONTINUE
        END IF
 2100 CONTINUE
      RETURN
      END
C***************
      SUBROUTINE HOut(iTPs1)
C*************************************************************************C
C Print out the HYD variables (Ada, U, S, W , EPz & Ez).
C*************************************************************************C
      INCLUDE 'COMMON.INC'

      WRITE(7,901)'* Ada (cm)                                              '
      WRITE(7,903) Tout(IncP)
      DO 1100 i=ML,MU
        WRITE(7,902) i, ':', Ada(i,2)
 1100 CONTINUE

      WRITE(7,901)'* U (cm/s)                                              '
      WRITE(7,903) Tout(IncP)
      DO 1110 i=ML,MU
        n = KBU(i)
        WRITE(7,902) i, ':', (U(i,k,2), k=1,n)
 1110 CONTINUE

      WRITE(7,901)'* S (ppt)                                               '
      WRITE(7,903) Tout(IncP)
      DO 1120 i=ML,MU
        n = KB(i)
        WRITE(7,902) i, ':', (S(i,k,2), k=1,n)
 1120 CONTINUE

      WRITE(7,901)'* W (10**-3 cm/s)                                       '
      WRITE(7,903) Tout(IncP)
      DO 1130 i=ML,MU
        n = KB(i)
        WRITE(7,902) i, ':', (W(i,k)*1.0E3, k=1,n)
 1130 CONTINUE

      WRITE(15,901)'* EPz (Diffusivity)                                    '
      WRITE(15,903) Tout(IncP)
      DO 1140 i=ML,MU
        n = KB(i)
        WRITE(15,902) i, ':', (EPz(i,k), k=1,n)
 1140 CONTINUE

      WRITE(15,901)'* Ez (Viscosity)                                       '
      WRITE(15,903) Tout(IncP)
      DO 1150 i=ML,MU
        n = KB(i)
        WRITE(15,902) i, ':', (Ez(i,k), k=1,n)
 1150 CONTINUE
  903 FORMAT('    at ', F13.8, ' tidal cycles after model starts')
```

```
 904 FORMAT('    over a cycle before ', F13.8, ' cycles after start')

     IF (iTPs1 .GE. 0)  THEN
       DO 2200 i=ML,MU
         Tavg(i) = Tavg(i) / AvgN
         n = KB(i)
         DO 2201 k=1,n
           Uavg(i,k) = Uavg(i,k) / AvgN
           Savg(i,k) = Savg(i,k) / AvgN
           Wavg(i,k) = Wavg(i,k) / AvgN * 1.0E3
           EPzAvg(i,k) = EPzAvg(i,k) / AvgN
           EzAvg(i,k) = EzAvg(i,k) / AvgN
2201     CONTINUE
2200   CONTINUE

       WRITE(7,901)'* Mean Ada (cm)                                          '
       WRITE(7,904) Tout(IncP)
       DO 2210 i=ML,MU
         WRITE(7,902) i, ':', Tavg(i)
2210   CONTINUE

       WRITE(7,901)'* Mean U (cm/s)                                          '
       WRITE(7,904) Tout(IncP)
       DO 2220 i=ML,MU
         n = KBU(i)
         WRITE(7,902) i, ':', (Uavg(i,k), k=1,n)
2220   CONTINUE

       WRITE(7,901)'* Mean S (ppt)                                           '
       WRITE(7,904) Tout(IncP)
       DO 2230 i=ML,MU
         n = KB(i)
         WRITE(7,902) i, ':', (Savg(i,k), k=1,n)
2230   CONTINUE

       WRITE(7,901)'* Mean W (in 10**-3 cm/s)                                '
       WRITE(7,904) Tout(IncP)
       DO 2240 i=ML,MU
         n = KB(i)
         WRITE(7,902) i, ':', (Wavg(i,k), k=1,n)
2240   CONTINUE

       WRITE(15,901)'* Mean EPz (diffusivity)                                '
       WRITE(15,904) Tout(IncP)
       DO 2260 i=ML,MU
         n = KB(i)
         WRITE(15,902) i, ':', (EPzAvg(i,k), k=1,n)
2260   CONTINUE

       WRITE(15,901)'* Mean Ez (viscosity)                                   '
       WRITE(15,904) Tout(IncP)
       DO 2270 i=ML,MU
         n = KB(i)
         WRITE(15,902) i, ':', (EzAvg(i,k), k=1,n)
2270   CONTINUE

       WRITE(7,901)'* DistT & Max, Min & Max-Min of tidal current            '
       WRITE(7,904) Tout(IncP)
       WRITE(7,907) (i, DistT(i), Umax(i),Umin(i), Umax(i)-Umin(i),
     *                    i=ML,MU)
       DO 10 i=ML,MU
         THI(i) = (THI(i) - THI(MU)) * 12.42
         IF (THI(i) .LT. 0.0)  THI(i) = THI(i) + 12.42
         TLO(i) = (TLO(i) - TLO(MU)) * 12.42
```

```
        IF (TLO(i) .LT. 0.0)  TLO(i) = TLO(i) + 12.42
10      CONTINUE
        WRITE(12,901)'* DistS & Max, Min, Max-Min of Ada & THI, TLO    '
        WRITE(12,904) Tout(IncP)
        WRITE(12,906) (i, DistS(i), Amax(i),Amin(i), Amax(i)-Amin(i),
     *                    THI(i),TLO(i), i=ML,MU)
        WRITE(14,904) Tout(IncP)
        WRITE(14,901)'* DistS and Time & Sal at SB Flood               '
        DO 20 i=ML,MU
          n = KB(i)
          WRITE(14,905) i, DistS(i), TSBF(i), (SSBF(i,k), k=1,n)
20      CONTINUE
        WRITE(14,901)'* DistS and Time & Sal at SB Ebb                 '
        DO 30 i=ML,MU
          n = KB(i)
          WRITE(14,905) i, DistS(i), TSBE(i), (SSBE(i,k), k=1,n)
30      CONTINUE
 905  FORMAT(I2, F6.1, F9.5, 10F7.3)
 906  FORMAT(I2, 6F12.4)
 907  FORMAT(I2, 4F12.4)

        AvgN = 0.0
        DO 2280 i=ML,MU
          Tavg(i) = 0.0
          n= KB(i)
          DO 2281 k=1,n
            Uavg(i,k) = 0.0
            Savg(i,k) = 0.0
            Wavg(i,k) = 0.0
            EPzAvg(i,k) = 0.0
            EzAvg(i,k) = 0.0
2281      CONTINUE
          Amax(i) = -1000.0
          Umax(i) = -1000.0
          Amin(i) = 1000.0
          UMin(i) = 1000.0
2280    CONTINUE
      END IF
 901  FORMAT(//, A48)
 902  FORMAT(I3, A1, 20F7.2)
      RETURN
      END
```

B-2. HYD-2D.FOR

```
C***************
      SUBROUTINE ReadM(DTT,rNTSiH,NTSiD,DTH)
C*************************************************************************C
C Read in parameters for geometry & model control from unit #5.
C*************************************************************************C
      INCLUDE 'COMMON.INC'

      WRITE(7,'(//A45/)')'***** Parameters for Geometry & Control *****'

C: KB(i) = # of layers in the ith segment.
C: Note that the BC's for Ada & S are specified at the center of the most
C  downstream segment (MU).
C: Note that all calculations will be done in CGS unit.

      READ(5,9001) ML, MU, Kmax
      READ(5,9002) Darea
      WRITE(7,804)'* Upstream boundary transect number   = ', ML,
     *            '  Downstream boundary transect number = ', MU,
     *            '  Max. # of layers in any segment     = ', Kmax
      WRITE(7,801)'* Drainage area (km**2) upstream of UB      = ',Darea
      ML1 = ML + 1
      MLs1 = ML - 1
      MU1 = MU + 1
      MUs1 = MU - 1

      READ(5,9001) (KB(i), i=ML,MU)
      READ(5,9002) (H(k), k=1,Kmax)
      WRITE(7,'(/A22/,(10X,I3,F8.1))')'** Layer thickness (m)',
     *                                (k, H(k), k=1,Kmax)
      DO 1300  k=1,Kmax
        H(k) = H(k) * 100.0
        DH(k) = 2.0 * H(k)
 1300 CONTINUE
      WRITE(7,'(/A20,20X,A5/A7,20I7)')'** Estuary width (m)',
     *        'Layer', 'Segment', (k, k=1,Kmax)
      DO 1310 i=ML,MU
        n = KB(i)
        READ(5,9002) (B(i,k), k=1,n)
        WRITE(7,'(I3,A1,20F7.1)') i, ':', (B(i,k), k=1,n)
        TotB(i) = 0.0
        DO 1311 k=1,n
          B(i,k) = B(i,k) * 100.0
          TotB(i) = TotB(i) + B(i,k)
 1311   CONTINUE
 1310 CONTINUE

C: Convert SSTm to cm after divided by DX.
C: SSTm = sfc storage area (St) at mean tide, not covered by B*DX.
C: StH = St depth (When SSTm=0, 1.0 is assigned to StH to avoid
0-division).
C: DX is also the distance bet/ the center of 2 succeissive segments.

      READ(5,9002) (SSTm(i), i=ML,MU)
      READ(5,9002) (StSlop(i), i=ML,MU)
      READ(5,9002) (StH(i), i=ML,MU)
      READ(5,9002) (ARD(i), i=ML,MU)
      READ(5,9002) DX, DmDB, DTs
      WRITE(7,801)'* Sfc storage area(in 10**6 m**2) at mean tide'
      WRITE(7,803) (SSTm(i), i=ML,MU)
      WRITE(7,801)'* Depth (in m) in surface storage area        '
      WRITE(7,803) (StH(i), i=ML,MU)
```

```
      WRITE(7,801)'* Change (in cm) in St width / cm rise in Ada '
      WRITE(7,803) (StSlop(i), i=ML,MU)
      WRITE(7,801)'* Drainage area (km**2) feeding into each seg '
      WRITE(7,803) (ARD(i), i=ML,MU)
      WRITE(7,801)'* Distance between transects (m)            = ', DX,
     *            '  Time interval (sec)                       = ', DTs,
     *            ': km from real mouth to most DB segment     = ', DmDB
        Tmp = 1.0E8 / DX
      DO 1340 i=ML,MU
        SSTm(i) = SSTm(i) * Tmp
        StH(i) = StH(i) * 100.0
 1340 CONTINUE
      DTT = DTs / 44712.0
      DTD = DTs / 86400.0
      DTH = DTs / 3600.0
      rNTSiH = 3600.0 / DTs
      NTSiD = NINT(1.0/DTD)

      DXinKM = DX / 1.0E3
      DistS(MU) = DmDB
      DistT(MU) = DmDB + DXinKM*0.5
      DO 1360 i=MUs1,ML,-1
        DistT(i) = DistT(i+1) + DXinKM
        DistS(i) = DistS(i+1) + DXinKM
 1360 CONTINUE
      DX = DX * 100.0
C*********************************************************************C
      READ(5,9003) Tmax, NP
      READ(5,9002) (Tout(m), m=1,NP)
      WRITE(7,801)'* Number of tidal cycles for model to be run =', Tmax
      WRITE(7,804)'  Number of times to print output     = ', NP
      WRITE(7,801)'* Times (in tidal cycles) to print output:    '
      WRITE(7,803) (Tout(m), m=1,NP)
      iTmax = NINT(Tmax/DTT)
 9001 FORMAT(15I5)
 9002 FORMAT(10F8.0)
 9003 FORMAT(F8.0,I8)
  801 FORMAT(/, (A46, F15.7))
  803 FORMAT((10F8.3))
  804 FORMAT(/, (A40, I5))
      RETURN
      END
C***************
      SUBROUTINE CalcGrid
C*********************************************************************C
      INCLUDE 'COMMON.INC'

C: rMD(i) = mean depth at the ith transect = Depth(i,k) + HMZ(i,k).
C: Depth(i,k) = depth below the mean surface to the bottom of the kth
C   layer at the ith transect (i.e., to the point where W is defined).
C: HMZ(i,k) = depth above the sed. bottom to the bottom of the kth layer
C   at the ith transect.
C: Note that above 3 are defined at the center of each segment, where W
C  (and thus Ez, EPz) is defined.

      DO 1400 i=ML,MU
        rMD(i) = 0.0
        n = KB(i)
        DO 1401 k=1,n
          rMD(i) = rMD(i) + H(k)
          Depth(i,k) = rMD(i)
 1401   CONTINUE
        DO 1402 k=1,n
          HMZ(i,k) = rMD(i) - Depth(i,k)
```

```
 1402   CONTINUE
 1400 CONTINUE
C*********************************************************************C
      DO 1700 i=ML,MUs1
        BT(i) = B(i,1) / DTs
        n = KB(i)
        DO 1701 k=2,n
          BHT(i,k) = B(i,k) * H(k) / DTs
 1701   CONTINUE
 1700 CONTINUE

C: Set KBU(i) = # of layers for U definition at each transect.
C: Note KBU(i) should always be .LE. KB(i) (see the Fig. 2-1 in manual).

      KBU(ML) = KB(ML)
      DO 1500 i=ML1,MU
        k = KB(i)
        KBU(i) = k
        IF (B(i-1,k) .LT. 1.0E-5)  KBU(i) = KB(i-1)
C        IF (B(i-1,k) .EQ. 0.0)  KBU(i) = KB(i-1)
 1500 CONTINUE

C Set 2 bdry transects equal at each end to calculate U & Ada gradient.

      KB(MLs1) = KB(ML)
      n = KB(ML)
      DO 1510 k=1,n
        B(MLs1,k) = B(ML,k)
 1510 CONTINUE
      KB(MU1) = KB(MU)
      n = KB(MU)
      DO 1520 k=1,n
        B(MU1,k) = B(MU,k)
 1520 CONTINUE
      KBU(MU1) = KB(MU1)

C BU/BW = addition of 2 B's of adjacent segments or layers, respectively.

      DO 1600 i=ML,MU1
        iEC(i) = KB(i) - KB(i-1)
        n = KB(i)
        DO 1601 k=1,n
          BU(i,k) = B(i-1,k) + B(i,k)
          BW(i,k) = B(i,k) + B(i,k+1)
          BU2H(i,k) = BU(i,k) * DH(k)
 1601   CONTINUE
 1600 CONTINUE
      DO 1650 i=ML,MU
        IF (iEC(i) .GT. 0  .AND.  iEC(i+1) .LT. 0)  THEN
          PRINT*, 'There is a hole at the bottom of segment ', i
          STOP '** Check bottom geometry for hole(s).'
        END IF
 1650 CONTINUE
      RETURN
      END
C***************
      SUBROUTINE ReadH(rNTSiH,Day,NS2PS)
C*********************************************************************C
C Read in parameters from unit #5.
C*********************************************************************C
      INCLUDE 'COMMON.INC'
      Common/HarTD/Ampl(9),Phase(9)
      Parameter (Cef=91.0, F=0.3333)
      Dimension  xPSQ(iX)
```

```
      Data xPSQ/iX*0.0/

      WRITE(7,'(//A40/)')'***** Parameters for Hydrodynamics *****'

C: When iWQ=1, SUBR WQInput in WQ-2D-IN.FOR may (if iTdep=1 there) read in
C  time-varying temperature (Temp in degC) from 'WQ.IN', every midnight.

      READ(5,9001) TempC
      WRITE(7,805)'* Time-constant estuarine temp (degC)     = ', TempC
C      Tmp = TempC * TempC
C      T1 = 5.89E3     + 38.0*TempC     - 0.375*Tmp
C      T2 = 5.89091E3 + 37.7752*TempC - 0.336262*Tmp
C      T3 = 1.70591    + 0.01*TempC

C: Read in parameters for BC's at the sfc.
C: WndDir (degree) = 0 to the east and increases counterclockwise.
C: Note WndDir is that to which the wind blows.

      READ(5,9001) Cstar, RhoAir
      READ(5,9001) (Paxis(i), i=ML,MU)
      Wdrag = 4.0E4 * Cstar * RhoAir
      WRITE(7,805)'* Air density (g/cm**3)                    = ',RhoAir,
     *            '  Drag coefficient (in CGS unit)           = ',Cstar,
     *            ': Wind stress (force/area/vel**2=g/cm**3) = ',Wdrag
      WRITE(7,808)'* Principal axis angle (deg) at each transect ',
     *            (Paxis(i), i=ML,MU)
      DO 10 i=ML,MU
        Paxis(i) = Paxis(i) * 1.74533E-2
 10   CONTINUE

C: Read in parameters for BC's at the bottom.
C: k (~ 2.5E-3) = Frik/2 = portion of bottom stress.  When rMan is ~0.015
C  to 0.03, g & H(k) should be in MKS unit.  Since g & H(k) are in CGS
C  unit, conversion factor (CF) of 100**(2/3)=21.544 is needed
C  (Cef=2g/CF=91, where g=980.0).

      READ(5,9001) (rMan(i), i=ML,MU)
      IF (iBotSh .EQ. 3)  THEN
        WRITE(7,804)'* Manning friction coeff. (in MKS unit)       ',
     *              (rMan(i), i=ML,MU)
        DO 1700 i=ML,MU
          n = KBU(i)
          DO 1701 k=1,n
            Frik(i,k) = Cef * rMan(i) * rMan(i) / H(k)**F
 1701     CONTINUE
 1700   CONTINUE
       ELSE IF (iBotSh .EQ. 2)  THEN
        WRITE(7,804)'* Bottom roughness (cm): about 0.1 cm         ',
     *              (rMan(i), i=ML,MU)
        c1 = 2.0 * 0.4*0.4
        DO 1705 i=ML,MU
          n = KBU(i)
          DO 1706 k=2,n
              a1 = H(k) / rMan(i)
              a2 = LOG(a1)
              a3 = a2 - 1.0
            Frik(i,k) = c1 / (a3*a3)
 1706     CONTINUE
 1705   CONTINUE
      END IF

C: Assume constant EPx & Ex.

 9000 FORMAT(4E8.3)
```

```
      READ(5,9000) rk
      WRITE(7,805)'* Proportionality constant for salinity   = ', rk
      READ(5,9000) CTurb, CRi, CWave, CHD
      WRITE(7,805)'* Constant for turbulent mixing coeff.    = ',CTurb,
     *            ': Constant for Richardson number          = ',CRi,
     *            ': Constant for wave part in mixing coeff  = ',CWave,
     *            ': Multiplier for horizontal mixing coeff  = ',CHD
      DO 2700 i=ML,MU
        n = KB(i)
        DO 2701 k=1,n
          EPx(i,k) = CHD
          Ex(i,k) = CHD
 2701   CONTINUE
 2700 CONTINUE
      IF (iUBC .EQ. 1)  THEN
        n = KB(ML)
        DO 2702 k=1,n
          EPx(ML,k) = 0.0
          EPx(ML,k) = 0.0
 2702   CONTINUE
      END IF

C Parameters used for computing E & EP caused by wind-induced waves.

      READ(5,9001) HtP, Wave2, WndSpd, WndDir
      IF (iWave .EQ. 1)  THEN
        READ(20,9002) HtP, Wave2, WndSpd, WndDir
        WRITE(7,806)'* Wind mxing parameters at the    ', Day,
     *              ' th days:  '
        WRITE(7,805)': Height**2 / Period (cm**2/s)           = ',HtP
     *             ,': 2 * pi / Length (/cm)                  = ',Wave2
     *             ,': Wind speed (m/s) at a height of 10 m   = ',WndSpd
     *             ,': Wind direction (deg) ccw from east     = ',WndDir
        Wave1 = CWave * HtP
       ELSE
        WRITE(7,804)'* Wind wave characteristics                '
        WRITE(7,805)': Height**2 / Period (cm**2/s)           = ',HtP
     *             ,': 2 * pi / Length (/cm)                  = ',Wave2
     *             ,': Wind speed (m/s) at a height of 10 m   = ',WndSpd
     *             ,': Wind direction (deg) ccw from east     = ',WndDir
        Wave1 = CWave * HtP
      END IF
      WndDir = WndDir * 1.74533E-2
      DO 60 i=ML,MU
        WS = WndSpd * COS(Paxis(i) - WndDir)
        WStrs(i) = Wdrag * WS * ABS(WS)
 60   CONTINUE

      READ(5,9001) (Alpha(i), i=ML,MU)
      WRITE(7,805)'* Weighting factor for horz mass advection  '
      WRITE(7,804)': when using upwind scheme                   ',
     *            (Alpha(i), i=ML,MU)
      DO 1800 i=ML,MU
        CAlpha(i) = 1.0 - Alpha(i)
        n = KB(i)
        DO 1801 k=1,n
          Gam(i,k) = Alpha(i)
          Del(i,k) = CAlpha(i)
 1801   CONTINUE
 1800 CONTINUE

C Initialize S by interpolating with SalUp & SalDn for the top layer (k=1)
C from i=MLs1-MU, and assign the values at k=1 to the remaining layers.
C: StS(i) = S at the storage area (ie, embayed area) at the ith transect.
```

```
      READ(5,'(2F8.0,I8)') SalUp, SalDn, NLS
      WRITE(7,801)'** Initial upstream salinity (ppt)      = ', SalUp,
     *            '   Initial downstream salinity (ppt)    = ', SalDn,
     *  '   Initial salt intrusion limit i.t.o. transect # = ', NLS
  801 FORMAT(/, 2(A42, F20.7, /), A52, I10)
      READ(5,9004) NS2Sal
      IF (NS2Sal .EQ. 1)  THEN
        READ(5,9001) TSIC
        WRITE(7,801)'* Time (in d) to specify the following S =', TSIC
        DO 110 i=ML,MU
          n = KB(i)
          READ(5,9001) (SIC(i,k), k=1,n)
          WRITE(7,807) i, ':', (SIC(i,k), k=1,n)
  110   CONTINUE
        iTSIC = NINT(TSIC/DTD)
      END IF

      DO 1710 i=MLs1,NLS
        S(i,1,1) = SalUp
 1710 CONTINUE
      DDS = (SalDn-SalUp) / REAL(MU-NLS)
      DO 1720 i=NLS,MU
        Tmp2 = REAL(MU-i)
        S(i,1,1) = SalDn - DDS*Tmp2
 1720 CONTINUE
      DO 1730 i=MLs1,MU
        S(i,1,2) = S(i,1,1)
        StS(i) = S(i,1,1)
        n = KB(i)
        DO 1731 k=2,n
          S(i,k,1) = S(i,1,1)
          S(i,k,2) = S(i,1,2)
  1731  CONTINUE
  1730 CONTINUE

C These will be used for calculating S BC's at open bdry in SUBR SOpBdry.
C: SFLDM = S in incoming water at open bdry transect during flood tide.
C: MST = TOFH in DTs scale.

      n = KB(MU)
      READ(5,9001) TOFH
      READ(5,9001) (SFLDM(k), k=1,n)
      WRITE(7,805)'* Time lapse to reach SFLDM after SBF (hr) =', TOFH
      IF (iSatDB .EQ. 1)  THEN
        READ(9,9002) (SFLDM(k), k=1,n)
        WRITE(7,803)'* Salinity (ppt) at DB at the     ', Day,
     *              ' th days:', (SFLDM(k), k=1,n)
       ELSE
        WRITE(7,804)'* Constant salinity at downstream bdry;       ',
     *              (SFLDM(k), k=1,n)
      END IF
      MST = NINT( TOFH*3600.0/DTs )
      n = KB(ML)
      READ(5,9001) (S(MLs1,k,2), k=1,n)
      DO 1990 k=1,n
        S(MLs1,k,1) = S(MLs1,k,2)
 1990 CONTINUE
      WRITE(7,804)'* Constant salinity at upstream bdry;         ',
     *            (S(MLs1,k,2), k=1,n)

C Use the FUNC HTide to initialize Ada(i,1) at open bdry (i=MU), and
C assign this value to the next time step, Ada(MU,2).
C: Note Ada's are already initialized (0ed) for i=MLs1-MUs1 in BLKDATA.INC
C: Initialize H1(i,t) = H(1)+Ada(i,t) for i=MLs1-MU.
```

```
      READ(5,9003) (Ampl(m), Phase(m), m=1,9)
      IF (iTide .EQ. 1)  THEN
        CALL ReadTD(Day)
        CALL OTide(Ada(MU,2),rNTSiH)
       ELSE
        WRITE(7,802) (Ampl(m), Phase(m), m=1,9)
        Ada(MU,2) = HTide(TinDay)
      END IF
      Ada(MU,1) = Ada(MU,2)
      DO 1740 i=MLs1,MU
        H1(i,1) = H(1) + Ada(i,1)
        H1(i,2) = H1(i,1)
 1740 CONTINUE
      DO 1750 i=ML,MU
        StH1(i,1) = StH(i) + Ada(i,1)
        StH1(i,2) = StH1(i,1)
        SST(i) = SSTm(i) + StSlop(i)*Ada(i,2)
 1750 CONTINUE
  802 FORMAT(/, '* Tidal amplitudes (cm) & phases (rad)',
     */ 5X,'M2 ',F8.2,F16.3 / 5X,'S2 ',F8.2,F16.3 / 5X,'N2 ',F8.2,F16.3,
     */ 5X,'K1 ',F8.2,F16.3 / 5X,'M4 ',F8.2,F16.3 / 5X,'O1 ',F8.2,F16.3,
     */ 5X,'MM ',F8.2,F16.3 / 5X,'SSA',F8.2,F16.3 / 5X,'SA ',F8.2,F16.3)
C*****************************************************************C
      READ(5,9004) NS2PS
      IF (iPS .EQ. 1)  THEN
        iUPS = 13
        WRITE(7,806)'* PS discharges (m**3/s) at the   ', Day,
     *              ' th days   '
       ELSE
        iUPS = 5
        WRITE(7,804)'* Timewise constant PS discharge (m**3/s);     '
      END IF
      DO 30 m=1,NS2PS
        READ(iUPS,9005) i, xPSQ(i)
        WRITE(7,807) i, ':', xPSQ(i)
        n = KB(i)
        DO 31 k=1,n
          PSQ(i,k) = 1.0E6 * xPSQ(i) * B(i,k)/TotB(i)
  31    CONTINUE
  30  CONTINUE

C: QML1/QML2 = U at the upstream bdry (i=ML) at previous/present day.

      n = KBU(ML)
      XAML = 0.0
      DO 50 k=1,n
        XAML = XAML + H(k)*B(ML,k)
  50  CONTINUE
      READ(5,9001) QML2
      IF (iDisch .EQ. 1)  THEN
        READ(10,9001) FDTL
        READ(10,9005) i, QML2
        WRITE(7,805)'* Time lag (days) to adjust f.w. discharge =', FDTL
        WRITE(7,806)'* Discharge (m**3/s) at UB at the ', Day,
     *              ' th days = ', QML2
        QML2 = QML2 * 1.0E6
        iFDTL = NINT(FDTL/DTD)
        rFDTL = REAL(iFDTL)
       ELSE
        WRITE(7,805)'* Constant f.w. discharge (m**3/s) at UB  = ', QML2
        QML2 = QML2 * 1.0E6
        QatUB = QML2
        CALL GetQLat
      END IF
```

```
      QML1 = QML2

C Initialize U - Take QML2 as U at i=ML1
C: Disch = freshwater discharge at i=ML1
C: Cross = X-sectional area at the point where U's are defined
C: Note that W's are already initialized in BLKDATA.INC (0ed).

      DO 1770 i=ML1,MU1
        n = KBU(i)
        Cross = 0.0
        DO 1771 k=1,n
          Cross = Cross + H(k)*BU(i,k)/2.0
 1771   CONTINUE
        U(i,1,1) = QML2 / Cross
        U(i,1,2) = U(i,1,1)
        DO 1772 k=2,n
          U(i,k,1) = U(i,1,1)
          U(i,k,2) = U(i,1,2)
 1772   CONTINUE
 1770 CONTINUE
 9001 FORMAT(10F8.0)
 9002 FORMAT(9X, 10F8.3)
 9003 FORMAT(2F8.0)
 9004 FORMAT(15I5)
 9005 FORMAT(9X, I5, F7.3)
  803 FORMAT(/, A34, F8.3, A9, /, (9F8.3))
  805 FORMAT(/, (A44, F15.5))
  804 FORMAT(/, A46, /, (10F7.3))
  806 FORMAT(/, A34, F8.3, A11, F8.3)
  807 FORMAT(I3, A1, 10F8.3)
  808 FORMAT(/, A46, /, (10F8.1))
      RETURN
      END
C***************
      SUBROUTINE HCalcC
C*********************************************************************C
C Calculates the invariant constants, once and for all.
C*********************************************************************C
      INCLUDE 'COMMON.INC'
      Parameter (g=980.0)

      dtx = DTs / DX
      DTi2 = 2.0 / DTs
      DTo2 = DTs / 2.0
      DX2 = 2.0 * DX
      DX4 = 4.0 * DX
      DXS2 = 2.0 * DX * DX
      DXS4 = 2.0 * DXS2
      DXi2 = 2.0 / DX
      DXiG = g / DX
      G4m = -4.0 * g
      RETURN
      END
C*************
      SUBROUTINE ReadTD(Day)
C*********************************************************************C
      Common/ObsTD/Tdata(0:25),rNTC

      rNTC = 0.0
      Tdata(0) = Tdata(24)
      READ(6,9001) (Tdata(m), m=1,24)
      DO 10 m=1,24
        Tdata(m) = Tdata(m) * 30.48
 10   CONTINUE
```

```
      WRITE(7,801)'* At the ', Day, ' th days, tidal heights (cm) are:',
     *            (Tdata(m), m=1,24)
 9001 FORMAT(7X, 12F6.2)
  801 FORMAT(/, A9, F8.3, A33, /, (9F8.2))
      RETURN
      END
C*************
      SUBROUTINE GetQLat
C*********************************************************************C
      INCLUDE 'COMMON.INC'

      QLAatUB = QatUB / Darea
      n = KBU(ML)
      IF (iUBC .EQ. 1)  THEN
        DO 10 k=1,n
          U(ML,k,2) = 0.0
 10     CONTINUE
        QLat(ML,1) = QatUB + QLAatUB*ARD(ML)
       ELSE
        U(ML,1,2) = QatUB / XAML
        DO 20 k=2,n
          U(ML,k,2) = U(ML,1,2)
 20     CONTINUE
        QLat(ML,1) = QLAatUB*ARD(ML)
      END IF
      DO 30 i=ML1,MUs1
        QLat(i,1) = QLAatUB*ARD(i)
 30   CONTINUE
      RETURN
      END
C*************
      FUNCTION HTide(Time)
C*********************************************************************C
C Calculate the tidal height at the open bdry, Ada(MU,TinDay), from tidal
C amplitudes (Ampl) & phases (Phase).
C: Sigma = angular speed in rad/d for each of M2,S2,N2,K1,M4,O1,MM,SSA,SA.
C*********************************************************************C
      Common/HarTD/Ampl(9),Phase(9)
      Dimension  Sigma(9)
      Data Sigma/12.141,12.566,11.913,6.3,24.282,5.84,0.228,0.034,0.017/

      HTide = 0.0
      DO 1200 m=1,9
        HTide = HTide + Ampl(m) * COS(Phase(m) + Time*Sigma(m))
 1200 CONTINUE
      RETURN
      END
C*************
      SUBROUTINE OTide(Y,rNTSiH)
C*********************************************************************C
      Common/ObsTD/Tdata(0:25),rNTC

      rNTC = rNTC + 1.0
      TmpDT = MOD(rNTC,rNTSiH)
      IF (TmpDT .LE. 0.01)  THEN
        m = NINT(rNTC/rNTSiH)
        Y = Tdata(m)
       ELSE
        m = rNTC/rNTSiH + 1.0
        Y = Tdata(m-1) + TmpDT * (Tdata(m) - Tdata(m-1)) / rNTSiH
      END IF
      RETURN
      END
C***************
```

```
      SUBROUTINE HReset
C*********************************************************************C
C: Reset variables for a new time step: Ada & H1 for i=MLs1-MU; S & U for
C  i=ML-MU, and extrapolate U at i=MU1 using U's at i=MU & MUs1.
C: Calculate density, Rho(i,k), for i=ML-MU.
C*********************************************************************C
      INCLUDE 'COMMON.INC'

      DO 100 i=MLs1,MU
        Ada(i,1) = Ada(i,2)
        H1(i,1) = H1(i,2)
 100  CONTINUE
      DO 110 i=ML,MU
        StH1(i,1) = StH1(i,2)
        SST(i) = SSTm(i) + StSlop(i)*Ada(i,2)
        BU2H(i,1) = BU(i,1) * (H1(i-1,1)+H1(i,1))
        n = KB(i)
        DO 111 k=1,n
          IF (iBotSh .EQ. 2)  Uo(i,k) = U(i,k,1)
          U(i,k,1) = U(i,k,2)
          S(i,k,1) = S(i,k,2)
C          Rho(i,k) = (T1 + 3.0*S(i,k,1)) / (T2 - T3*S(i,k,1))
          Rho(i,k) = 1.0 + rk*S(i,k,1)
  111   CONTINUE
 110  CONTINUE

      n = KBU(MU1)
      DO 120 k=1,n
        U(MU1,k,1) = U(MU1,k,2)
 120  CONTINUE
      RETURN
      END
C***************
      SUBROUTINE CalcP
C*********************************************************************C
C: Calculate the longitudinal pressure gradient, Press(i,k), for i=ML1-MU.
C: Note KBU(i) instead of KB(i) because we only need Press (gradient)
C  where U's are defined.  This is why mean values are used for Ada & Rho.
C*********************************************************************C
      INCLUDE 'COMMON.INC'

      DO 300 i=ML1,MU
C k=1
        Tmp = (H1(i-1,2)+H1(i,2))*0.5 * (Rho(i,1)-Rho(i-1,1))
        Press(i,1) = (Rho(i-1,1)+Rho(i,1))*(Ada(i,2)-Ada(i-1,2)) + Tmp

        n = KBU(i)
        IF (n .NE. 1)  THEN
C k=2
          Tmp1 = H(2) * (Rho(i,2) - Rho(i-1,2))
          Press(i,2) = Press(i,1) + Tmp + Tmp1
C k=3-KBU
          DO 301 k=3,n
            Tmp2 = H(k) * (Rho(i,k) - Rho(i-1,k))
            Press(i,k) = Press(i,k-1) + Tmp1 + Tmp2
            Tmp1 = Tmp2
  301     CONTINUE
        END IF
 300  CONTINUE
      RETURN
      END
C***************
      SUBROUTINE CalcE
C*********************************************************************C
```

```
C Calculate vertical eddy viscosity (Ez) & turbulent diffusivity (EPz) for
C i=ML-MU using the values at the previous time step (ie, time step 1).
C: IF DU <= 1.0E-5, then Ez/EPz are set to be very small value, say 0.1.
C: IF EPz/Ez < 0.1, then EPz/Ez = 0.1.
C: Note when KB(i)=1, EPz/Ez = 0.0 (as sepcified in BLKDATA.INC).
C*******************************************************************C
      INCLUDE 'COMMON.INC'
      Parameter (Rfc=0.191,Rf1=0.234,Rf2=0.223)

      DO 400 i=ML,MU
        n = KB(i) - 1
        TotH = rMD(i) + Ada(i,1)
          No = KB(i) - iEC(i)
          NoA = KB(i) + iEC(i+1)
        DO 401 k=1,n
          IF (iEC(i+1) .GE. 0)  THEN
            IF (k .LT. No)  THEN
              DU = U(i,k,1)+U(i+1,k,1) - (U(i,k+1,1)+U(i+1,k+1,1))
             ELSE IF (k .EQ. No)  THEN
              DU = U(i,k,1)+U(i+1,k,1) - 2.0*U(i+1,k+1,1)
             ELSE
              DU = 2.0 * (U(i+1,k,1) - U(i+1,k+1,1))
            END IF
           ELSE
            IF (k .LT. NoA)  THEN
              DU = U(i,k,1)+U(i+1,k,1) - (U(i,k+1,1)+U(i+1,k+1,1))
             ELSE IF (k .EQ. NoA)  THEN
              DU = U(i,k,1)+U(i+1,k,1) - 2.0*U(i,k+1,1)
             ELSE
              DU = 2.0 * (U(i,k,1) - U(i,k+1,1))
            END IF
          END IF
          IF (iUBC.EQ.1 .AND. i.EQ.ML) DU=2.0*(U(i+1,k,1)-U(i+1,k+1,1))
          IF (k .EQ. 1)  THEN
            Thick = H1(i,1) + H(2)
           ELSE
            Thick = H(k) + H(k+1)
          END IF
          ABSDU = ABS(DU)
          DRho = Rho(i,k) - Rho(i,k+1)
          IF (iTurb .EQ. 3)  THEN
            IF (DRho .GT. -1.0E-5)  THEN
              Ri = 0.0
             ELSE
              IF (ABSDU .GE. 1.0E-5)  THEN
                Ri = G4m * Thick * DRho/(Rho(i,k)+Rho(i,k+1)) / (DU*DU)
               ELSE
                EPz(i,k) = 0.09
                Ez(i,k) = 0.09
                GO TO 401
              END IF
            END IF
           ELSE IF (iTurb .EQ. 2)  THEN
            IF (ABSDU .GE. 1.0E-5)  THEN
              Ri = G4m * Thick * DRho/(Rho(i,k)+Rho(i,k+1)) / (DU*DU)
              IF (Ri .LT. -10.0)  Ri = -10.0
              Rf = 0.656 * (Ri + 0.178 - SQRT((Ri - 0.323)*Ri + 0.0318))
             ELSE
              EPz(i,k) = 0.09
              Ez(i,k) = 0.09
              GO TO 401
            END IF
          END IF
          Z = Depth(i,k) + Ada(i,1)
```

```
          IF (Wave1 .EQ. 0.0)  THEN
            WaveZ = 0.0
           ELSE
            WaveZ = Wave1 * EXP(-Wave2*Z)
          END IF
          rML = Z * HMZ(i,k) / TotH
          VSML = CTurb * rML*rML * ABSDU/Thick
          IF (iTurb .EQ. 3)  THEN
            Div = 1.0 + CRi*Ri
              xx = VSML / SQRT(Div)
            EPz(i,k) = xx/Div + WaveZ
            Ez(i,k)  = xx + WaveZ
            IF (EPz(i,k) .LT. 0.1)  EPz(i,k) = 0.1
            IF (Ez(i,k) .LT. 0.1)  Ez(i,k) = 0.1
           ELSE IF (iTurb .EQ. 2)  THEN
            IF (Rf .GE. Rfc)  THEN
              EPz(i,k) = 0.11 + WaveZ
              Ez(i,k) = 0.11 + WaveZ
             ELSE
               yy = 1.0 - Rf/Rfc
              Term1 = yy*SQRT(yy) / (1.0 - Rf)
              Term2 = (1.0 - Rf/Rf1) / (1.0 - Rf/Rf2)
                xx = SQRT(Term2)
              EPz(i,k) = Term1 * xx * VSML / 0.8 + WaveZ
              Ez(i,k)  = Term1 * Term2*xx  * VSML + WaveZ
              IF (EPz(i,k) .LT. 0.1)  EPz(i,k) = 0.1
              IF (Ez(i,k) .LT. 0.1)  Ez(i,k) = 0.1
            END IF
          END IF
  401   CONTINUE
  400   CONTINUE

        n = KB(ML) - 1
        DO 410 k=1,n
          EPz(MLs1,k) = EPz(ML,k)
          Ez(MLs1,k)  = Ez(ML,k)
  410   CONTINUE
        RETURN
        END
C***************
        SUBROUTINE CEqn1(rNTSiH)
C*********************************************************************C
C Solve CONTINUITY Eq to get water sfc elevation, Ada(i,2), for i=MLs1-MU.
C: Ada(MLs1,2) is extrapolated from Ada(ML,2) & Ada(ML1,2), and
C  Ada(MU,2) is from FUNCTION HTide or SUBR OTide.
C: Total width = B(i,1)+SST(i) instead of B(i,1) - see pxx in the report.
C: Note that QLat is in volume per second.
C*********************************************************************C
        INCLUDE 'COMMON.INC'

        Tmp1 = U(ML,1,1) * BU2H(ML,1)
        DO 500 i=ML,MUs1
          Tmp2 = U(i+1,1,2) * BU2H(i+1,1)
          Ada(i,2) = Ada(i,1) + DTo2/(B(i,1)+SST(i))*( BW(i,1)*W(i,1)
     *     - (Tmp2-Tmp1)/DX2 + (QLat(i,1)+PSQ(i,1))*DXi2 )
          Tmp1 = Tmp2
          H1(i,2) = H(1) + Ada(i,2)
          StH1(i,2) = StH(i) + Ada(i,2)
          DelA(i) = Ada(i,2) - Ada(i,1)
          SDoDT(i) = SST(i) * DelA(i) / DTs
  500   CONTINUE

        IF (iTide .EQ. 1)  THEN
          CALL OTide(Ada(MU,2),rNTSiH)
```

```
       ELSE
         Ada(MU,2) = HTide(TinDay)
       END IF
       H1(MU,2) = H(1) + Ada(MU,2)
       Ada(MLs1,2) = 2.0*Ada(ML,2) - Ada(ML1,2)
       H1(MLs1,2) = H(1) + Ada(MLs1,2)
       RETURN
       END
C***************
       SUBROUTINE AdvDif
C*********************************************************************C
C Calculate advective & diffusive terms in MOMENTUM, SALT BALANCE & DO
C BALANCE Eqns (for i=ML-MU in ME & for i=ML-MUs1 for MBE).
C: HAdvM/HDifM = horizontal advective/diffusive terms in ME.
C: HAdvS/HDifS =                                  transport terms in SBE.
C: HAdv/HDif = portion of horizontal advective/diffusive terms in DO BE.
C: VAdv/VDif =            vertical advective/diffusive terms in DO BE.
C*********************************************************************C
       INCLUDE 'COMMON.INC'

C1 For i=ML-MUs1
C1a top layer (k=1) : VAdv & VDif = 0 here.
C1b remaining layers (k=2 to KB(i))
C : VAdv(i,k) & VDif(i,k) = at top of the kth layer (bottom of (k-1)th).
C : Note that VDif(i,2) = VDif at bottom of the 1st layer.
C1a
         HAM1 = BU2H(ML,1) * U(ML,1,1)
         HDM1 = BU2H(ML,1) * Ex(ML,1)
       DO 600 i=ML,MUs1
           HAM2 = BU2H(i+1,1) * U(i+1,1,1)
           HDM2 = BU2H(i+1,1) * Ex(i+1,1)
         HAdv(i,1) = HAM2 / DX4
         HDif(i,1) = BU2H(i+1,1) * EPx(i+1,1) / DXS4
         HAdvM(i,1) = (HAM1 + HAM2) * (U(i,1,1) + U(i+1,1,1))
         HDifM(i,1) = (HDM1 + HDM2) * (U(i+1,1,1) - U(i,1,1))
           HAM1 = HAM2
           HDM1 = HDM2
C1b
         n = KB(i)
         DO 601 k=2,n
           HAdv(i,k) = BU2H(i+1,k) * U(i+1,k,1) / DX4
           HDif(i,k) = BU2H(i+1,k) * EPx(i+1,k) / DXS4
           VAdv(i,k) = W(i,k-1) * BW(i,k-1) / 4.0
           IF (k .EQ. 2)  THEN
             VDif(i,2) = EPz(i,1) * BW(i,1) / (H1(i,2) + H(2))
            ELSE
             VDif(i,k) = EPz(i,k-1) * BW(i,k-1) / (H(k-1) + H(k))
           END IF
           IF (k .LE. KBU(i))  THEN
             HAdvM(i,k) = (BU2H(i,k)*U(i,k,1) + BU2H(i+1,k)*U(i+1,k,1))
     *                  * (U(i,k,1) + U(i+1,k,1))
             HDifM(i,k) = (BU2H(i,k)*Ex(i,k) + BU2H(i+1,k)*Ex(i+1,k))
     *                  * (U(i+1,k,1) - U(i,k,1))
           END IF
  601    CONTINUE
 600   CONTINUE

C2 For i=MLs1

       n = KB(MLs1)
       DO 620 k=1,n
         HAdv(MLs1,k) = BU2H(ML,k) * U(ML,k,1) / DX4
         HDif(MLs1,k) = BU2H(ML,k) * EPx(ML,k) / DXS4
 620   CONTINUE
```

```
C3 For i=MU: Ada & Ex are extrapolated from values at i=MUs1 & MU1.

        Ada(MU1,1) = 2.0*Ada(MU,1) - Ada(MUs1,1)
        H1(MU1,1) = H(1) + Ada(MU1,1)
      BU2H(MU1,1) = BU(MU1,1) * (H1(MU,1)+H1(MU1,1))
      n = KBU(MU)
      DO 610 k=1,n
          Ex(MU1,k) = 2.0*Ex(MU,k) - Ex(MUs1,k)
        HAdvM(MU,k) = (BU2H(MU,k)*U(MU,k,1) + BU2H(MU1,k)*U(MU1,k,1))
     *              * (U(MU,k,1)+U(MU1,k,1))
        HDifM(MU,k) = (BU2H(MU,k)*Ex(MU,k) + BU2H(MU1,k)*Ex(MU1,k))
     *              * (U(MU1,k,1)-U(MU,k,1))
 610  CONTINUE
      RETURN
      END
C***************
      SUBROUTINE MEqn
C*********************************************************************C
C1 Solve the horizontal MOMENTUM Eqn. to get U(i,k,2) for i=ML1-MU.
C2 For i=MU1, do the extrapolation.
C: Assuming that the main conveying channel loses all the momentum of
C  water particles entering into storage area (St) during flood (when
C  DAda > 0.0), and receive no momentum (since water coming from St may
C  not have momentum in x-dir of main channel) from the water particles
C  coming from St during ebb, sink term (Sink, which is 0 during ebb) is
C  needed during flood. For the mathematical formulation, see the
C  explanation in SUBR SBEqn.
C: Since only the time variation of momentum & mass in St will affect the
C  main channel and also St is usually very shallow, the exchange of
C  momentum & mass may be assumed to occur only at the top layer (ie,
C  communication only through the sfc and no lateral diffusion of momentum
C  & mass bet/ St and main channel).
C* WUBt/WUBb = W*U*B (vertical advective term) at the top/bottom of the
C  kth layer at the ith transect.
C* SidF = friction at the bottom of each layer.
C : BdShr = bed shear in (cm/s)**2.
C : Note that kth layer has bottom area of (BU(i,k)-BU(i,k+1)) / 2.0.
C1-a top layer (k=1)
C : WUBt = 0.
C : Note that WStrs/Rho = (cm/s)**2.
C1-b for k=2 to KBU(i)-1
C : Note that there's no Sink any more.
C1-c bottom layer (k=KBU(i))
C : WUBb = 0.
C : In SidF, only BU(i,k) since there's no BU(i,k+1) when k=KBU(i).
C*********************************************************************C
      INCLUDE 'COMMON.INC'
      Dimension CCu(iZ),DDu(iZ),EEu(iZ),BBu(iZ)
      Data CCu,DDu,EEu,BBu/60*0.0/

      DO 700 i=ML1,MU
C1-a
          DAda = Ada(i-1,2)+Ada(i,2) - (Ada(i-1,1)+Ada(i,1))
          x1 = BU(i,1) + BU(i,2)
          x2 = H1(i-1,2) + H1(i,2)
        WUBb = (U(i,1,1)+U(i,2,1)) * (W(i-1,1)+W(i,1)) * x1 / 4.0
        WandP = BU(i,1) / (Rho(i-1,1)+Rho(i,1))
     *        * (WStrs(i) - DXiG*x2*Press(i,1))
        IF (iBotSh .EQ. 2)  THEN
            a1 = H1(1,1) / rMan(i)
            a2 = LOG(a1)
            a3 = a2 - 1.0
          Frik(i,1) = 0.32 / (a3*a3)
        END IF
```

```
          BdShr = Frik(i,1) * U(i,1,1) * ABS(U(i,1,1))
        SidF = BdShr * (BU(i,1)-BU(i,2))
        IF (DAda .GT. 0.0)  THEN
          Sink = U(i,1,1) * (SST(i-1)+SST(i)) * DAda * DTi2
         ELSE
          Sink = 0.0
        END IF
        CCu(1) = 0.0
        EEu(1) = -2.0 * (Ez(i-1,1)+Ez(i,1)) * x1 / (x2 + DH(2))
        DDu(1) = BU(i,1)*x2/DTs - EEu(1)
        BBu(1) = BU2H(i,1)/DTs*U(i,1,1) + WUBb - SidF - Sink + WandP
     *  + (HAdvM(i-1,1)-HAdvM(i,1))/DX4 + (HDifM(i,1)-HDifM(i-1,1))/DXS2

        n = KBU(i)
        IF (n .NE. 1)  THEN
C1-b
          IF (n .NE. 2)  THEN
            DO 701 k=2,n-1
                x3 = BU(i,k) + BU(i,k+1)
                x4 = BU2H(i,k) / DTs
              WUBt = WUBb
              WUBb = (U(i,k,1)+U(i,k+1,1)) * (W(i-1,k)+W(i,k)) * x3 /4.0
                BdShr = Frik(i,k) * U(i,k,1) * ABS(U(i,k,1))
              SidF = BdShr * (BU(i,k)-BU(i,k+1))
              CCu(k) = EEu(k-1)
              EEu(k) = -2.0*(Ez(i-1,k)+Ez(i,k)) * x3 / (DH(k)+DH(k+1))
              DDu(k) = x4 - CCu(k) - EEu(k)
              BBu(k) = x4*U(i,k,1) + (HAdvM(i-1,k)-HAdvM(i,k))/DX4
     *          + WUBb-WUBt + (HDifM(i,k)-HDifM(i-1,k))/DXS2 - SidF
     *          - DXiG*BU2H(i,k)*Press(i,k)/(Rho(i-1,k)+Rho(i,k))
  701       CONTINUE
          END IF
C1-c
            x5 = BU2H(i,n) / DTs
          WUBt = WUBb
            BdShr = Frik(i,n) * U(i,n,1) * ABS(U(i,n,1))
          SidF = BdShr * BU(i,n)
          CCu(n) = EEu(n-1)
          EEu(n) = 0.0
          DDu(n) = x5 - CCu(n)
          BBu(n) = x5*U(i,n,1) + (HAdvM(i-1,n)-HAdvM(i,n))/DX4
     *      - WUBt + (HDifM(i,n)-HDifM(i-1,n))/DXS2 - SidF
     *      - DXiG*BU2H(i,n)*Press(i,n)/(Rho(i-1,n)+Rho(i,n))
        END IF
        CALL SGTSL(n,CCu,DDu,EEu,BBu,info)
        IF (info .NE. 0)  THEN
          PRINT*, info, ' th element of the diagonal = 0.'
          STOP 'in SUBR MEqn'
        END IF
        DO 702 k=1,n
          U(i,k,2) = BBu(k)
  702   CONTINUE
  700 CONTINUE
C2
      n = KBU(MU1)
      DO 710 k=1,n
        U(MU1,k,2) = 2.0*U(MU,k,2) - U(MUs1,k,2)
  710 CONTINUE
      RETURN
      END
C***************
      SUBROUTINE SBEqn
C*********************************************************************C
C Solve the MASS BALANCE Eqn. for SALT to get S(i,k,2) for i=ML1-MUs1.
```

```
C: In St, for conservative substance like S.
C 1. During flood (DelA >= 0), water enters into St along with S.
C  a. St acts like sink for momentum & mass from main channel's viewpoint.
C   : S = Sink / (B*(H+Ada2)), where Sink = S [S in incoming water] *
C     SST*DelA/DTs [volume change per unit length in x-dir in St for DTs]
C     * DTs [time interval].
C  b. StS changes due to contribution from incoming main channel water.
C   : StS = (StS*(H+Ada1) + S*DelA) [(S in St * depth in St in time 1)
C     mix (S in incoming water * depth increase from time 1 to 2] /
C     (H+Ada2) [new St depth at time 2].
C 2. During ebb,
C  a. St provides S (but no momentum) to main channel.
C   : Since water in St flows into main channel, use StS instead of S.
C   : Note that since DelA < 0, -Sink > 0.
C  b. Note that it is assumed that StS changes only when flooding.
C* WSBt/WSBb = W*S*B (vertical advective transport term) at the top/bottom
C  of the kth layer at the ith transect.
C1 Top layer (k=1) : WSBt = 0.
C2 For k=2 to KB(i)-1 : Note that there's no Sink any more.
C3 Bottom layer (k=KB(i)) : WSBb = 0.
C**********************************************************************C
      INCLUDE 'COMMON.INC'
      Dimension BB(iZ),xC(iZ),xD(iZ),xE(iZ)
      Dimension A1(iX,iZ),A2(iX,iZ),CURV(iX,iZ)
      Data BB,xC,xD,xE, A1,A2, CURV,Adj/3661*0.0/

C QUICKEST scheme is used for the horizontal advection term.
C CURV(MU,k) = 0 by assuming S(MU1,k) = 2*S(MU,k) - S(MUs1,k)
C CURV(MLs1,k) = 0 by assuming S(MLs2) = 2*S(MLs1) - S(ML,k)

      IF (iHAdv .EQ. 3)  THEN
        DO 20 i=ML,MUs1
          n = KB(i)
          DO 24 k=1,n
            CURV(i,k) = S(i+1,k,1) - 2.0*S(i,k,1) + S(i-1,k,1)
 24       CONTINUE
          ns1 = KB(i-1) + 1
          IF (n .GE. ns1)  THEN
            DO 22 k=ns1,n
C assuming S(i-1,k,1) = S(i,k,1)
              CURV(i,k) = S(i+1,k,1) - S(i,k,1)
 22         CONTINUE
          END IF
          np1 = KB(i+1) + 1
          IF (n .GE. np1)  THEN
            DO 23 k=np1,n
C assuming S(i+1,k,2) = S(i,k,2)
              CURV(i,k) = -S(i,k,1) + S(i-1,k,1)
 23         CONTINUE
          END IF
 20     CONTINUE
      END IF
      DO 10 i=MLs1,MUs1
        n = KB(i)
        DO 11 k=1,n
          IF (iHAdv .EQ. 2)  THEN
            Gam(i+1,k) = Alpha(i+1)
            IF (U(i+1,k,1) .LT. 0.0)  Gam(i+1,k) = CAlpha(i+1)
            Del(i+1,k) = 1.0 - Gam(i+1,k)
            A1(i,k) = (Gam(i+1,k)*S(i,k,1) + Del(i+1,k)*S(i+1,k,1))
     *                * HAdv(i,k)
          ELSE IF (iHAdv .EQ. 3)  THEN
            x1 = dtx * U(i+1,k,1)
            COUR2(i,k) = x1 * 0.5
```

```
          CK(i,k) = (1.0 - x1*x1) / 6.0
          Adj = (S(i,k,1)+S(i+1,k,1))*0.5
     *          - COUR2(i,k)*(S(i+1,k,1)-S(i,k,1))
          IF (U(i+1,k,1) .GE. 0.0)  THEN
            A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURV(i,k))
           ELSE
            A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURV(i+1,k))
          END IF
        END IF
        A2(i,k) = HDif(i,k)*(S(i+1,k,1) - S(i,k,1))
 11   CONTINUE
10   CONTINUE

     DO 800 i=ML,MUs1
       Sink = 0.0
       IF (SST(i) .NE. 0.0)  THEN
         IF (StH1(i,2) .GT. 0.0)  THEN
           IF (DelA(i) .GT. 0.0)  THEN
             Sink = S(i,1,1) * SDoDT(i)
             StS(i) = (StS(i)*StH1(i,1) + S(i,1,1)*DelA(i)) / StH1(i,2)
            ELSE
             Sink = StS(i) * SDoDT(i)
           END IF
          ELSE
           StS(i) = 0.0
         END IF
       END IF
       WSBb = VAdv(i,2) * (S(i,1,1)+S(i,2,1))
       CC(i,1) = 0.0
       EE(i,1) = - VDif(i,2)
       DD(i,1) = BT(i)*H1(i,2) - EE(i,1)
       BB(1) = BT(i)*H1(i,1)*S(i,1,1) + A1(i-1,1)-A1(i,1) + WSBb
     *   + A2(i,1)-A2(i-1,1) - Sink
       xC(1) = CC(i,1)
       xD(1) = DD(i,1)
       xE(1) = EE(i,1)

       n = KB(i)
       IF (n .NE. 1)  THEN
         IF (n .NE. 2)  THEN
           DO 801 k=2,n-1
             WSBt = WSBb
             WSBb = VAdv(i,k+1) * (S(i,k,1)+S(i,k+1,1))
             CC(i,k) = EE(i,k-1)
             EE(i,k) = - VDif(i,k+1)
             DD(i,k) = BHT(i,k) - CC(i,k) - EE(i,k)
             BB(k) = BHT(i,k)*S(i,k,1) + A1(i-1,k)-A1(i,k)
     *         + WSBb-WSBt + A2(i,k)-A2(i-1,k)
             xC(k) = CC(i,k)
             xD(k) = DD(i,k)
             xE(k) = EE(i,k)
801        CONTINUE
         END IF

         WSBt = WSBb
         CC(i,n) = EE(i,n-1)
         EE(i,n) = 0.0
         DD(i,n) = BHT(i,n) - CC(i,n)
         BB(n) = BHT(i,n)*S(i,n,1) + A1(i-1,n)-A1(i,n) - WSBt
     *     + A2(i,n)-A2(i-1,n)
         xC(n) = CC(i,n)
         xD(n) = DD(i,n)
         xE(n) = EE(i,n)
       END IF
```

```
        CALL SGTSL(n,xC,xD,xE,BB,infoS)
        IF (infoS .NE. 0)  THEN
          PRINT*, infoS, ' th element of the diagonal = 0.'
          STOP 'in SUBR SBEqn'
        END IF
        DO 802 k=1,n
            S(i,k,2) = BB(k)
 802    CONTINUE
 800  CONTINUE
      RETURN
      END
C***************
      SUBROUTINE SOpBdry
C*********************************************************************C
C Calculate S(i,k,2) at the seaward bdry (i=MU).
C1. During ebb (when U >= 0.0), the horizontal S profile is assumed to be
C   advected out of mouth as a "frozen" pattern (ie, neglect diffusion).
C   S(MU,k,3) = S(MU,k,2) - (S(MU,k,2)-S(MUs1,k,2))/DX * U(MU,k,2)*DT.
C2. During flood,
C : From the beginning of flood to MST, some period of adjustment (MST
C   time steps) is allowed after the flow starts to flood and before S
C   reaches the bay value. S(MU,k,3) = S(MU,k,2) + (SFLDM-S(MU,k,2))/MST.
C : After MST, S(MU,k,3) = SFLDM(k).
C*********************************************************************C
      INCLUDE 'COMMON.INC'
      Dimension DS(iZ),Ncount(iZ)
      Save/HLocal/, DS,Ncount
      Data Ncount/15*10000/

      n = KB(MU)
      DO 900 k=1,n
        IF (U(MU,k,1) .GE. 0.0)  THEN
          Ncount(k) = 0
          Change = dtx * U(MU,k,1)
          S(MU,k,2) = S(MU,k,1)*(1.0-Change) + S(MUs1,k,1)*Change
         ELSE
          IF (Ncount(k) .GE. MST)  THEN
            S(MU,k,2) = SFLDM(k)
           ELSE
            IF (Ncount(k) .EQ. 0)  DS(k) = (SFLDM(k)-S(MU,k,1)) / MST
            Ncount(k) = Ncount(k) + 1
            S(MU,k,2) = S(MU,k,1) + DS(k)
          END IF
        END IF
 900  CONTINUE
      RETURN
      END
C***************
      SUBROUTINE CEqn2
C*********************************************************************C
C Solve CONTINUITY Eqn. to get W(i,k) for the current time step for
C i=ML-MU and k=1 to KB(i)-1.
C: W at k=KB(i) is always 0 as given by bdry condition (BC).
C: Note that all W(i,k)'s have been initialized to 0 at the BLKDATA.INC.
C: Note formulation is such that calculation is done from k=KB(i)-1 to 1.
C1 For k=KB(i)-1
C : Since there's no B(i,k+2) for this layer, B(i,k+1) is used instead.
C2 From k=KB(i)-2 to 1
C*********************************************************************C
      INCLUDE 'COMMON.INC'

      DO 1000 i=ML,MU
        n = KB(i) - 1
        IF (n .NE. 0)  THEN
```

```
C1
          W(i,n) = (  2.0*B(i,n+1)*W(i,n+1) - H(n+1)/DX*
     *         ( U(i+1,n+1,2)*BU(i+1,n+1)-U(i,n+1,2)*BU(i,n+1) )
     *       + (QLat(i,n+1)+PSQ(i,n+1))*DXi2  ) / BW(i,n)
C2
          m = n - 1
          DO 1001 k=m,1,-1
            W(i,k) = (  BW(i,k+1)*W(i,k+1) - H(k+1)/DX*
     *           ( U(i+1,k+1,2)*BU(i+1,k+1)-U(i,k+1,2)*BU(i,k+1) )
     *         + (QLat(i,k+1)+PSQ(i,k+1))*DXi2  ) / BW(i,k)
 1001     CONTINUE
        END IF
 1000 CONTINUE
      RETURN
      END
C***************
      BLOCK DATA HFirst
C*********************************************************************C
      INCLUDE 'COMMON.INC'
      INCLUDE 'BLKDATA.INC'
      END
```

B-3. WQ-2D-IN.FOR

```
C***************
      SUBROUTINE WQC1
C*************************************************************************C
C Read in constant parameters from 'WQ.IN'.
C: NS2 = # of segments for which parameters will be read.
C  Setting NS2=2 establishes uniform values, otherwise NS2 should = MU.
C: Since we don't have field data fine enough to see the difference
c  between each layer, generally 1-D arraies are used for WQC's.
C1 Parameters related to WQ model control parameter
C: iTdep = 1 means time-dependent water quality input conditions.
C2 Parameters related to nutrient transfer.
C3 Parameters related to CBOD decay.
C*************************************************************************C
      INCLUDE 'COM-WQ1.INC'
C1
      READ(11,9003) iTdep,iSun,iBCWQ,iNPS,iWQPS,iWQDBC
      IF (iTdep .EQ. 1)  THEN
        WRITE(8,801)'* Time-varying water quality input conditions.    '
       ELSE
        WRITE(8,801)'* Steady water quality input conditions.          '
      END IF
      IF (iSun .EQ. 1)  THEN
        OPEN(16,FILE='SOLAR.IN')
        WRITE(8,801)'* Time-varying solar radiation parameters.        '
       ELSE
        WRITE(8,801)'* Constant solar radiation parameters.            '
      END IF
      IF (iBCWQ .EQ. 1)  THEN
        OPEN(17,FILE='WQBC.IN')
        WRITE(8,801)'* Time-varying up-downstream boundary conditions. '
       ELSE
        WRITE(8,801)'* Constant up-downstream boundary conditions.     '
      END IF
      IF (iNPS .EQ. 1)  THEN
        OPEN(18,FILE='NPS.IN')
        WRITE(8,801)'* Time-varying non-point source input.            '
        READ(18,9001) xxF
        WRITE(7,804)'* Time lag (days) to adjust fw discharge =', xxF
       ELSE
        WRITE(8,801)'* Constant non-point source input.                '
      END IF
      IF (iWQPS .EQ. 1)  THEN
        OPEN(19,FILE='PS.IN')
        WRITE(8,801)'* Time-varying point source input.                '
       ELSE
        WRITE(8,801)'* Constant point source input.                    '
      END IF
      IF (iNPS.EQ.1 .OR. iWQPS.EQ.1 .OR. iSun.EQ.1 .OR. iBCWQ.EQ.1) THEN
        IF (iTdep .NE. 1)  STOP '** iTdep should be 1 (time-varying)'
      END IF
      IF (iWQDBC .EQ. 3)  THEN
        WRITE(8,801)'* SUBR WDnBdry calculates C(MU,k,2).              '
       ELSE IF (iSDBC .EQ. 2)  THEN
        WRITE(8,801)'* C(MU,k,2) = DnC(k): do not use SUBR WDnBdry.    '
       ELSE
        STOP '** Error in input: iWQDBC should be either 2 or 3 !!'
      END IF
  801 FORMAT(A50)
 9003 FORMAT(10I5)
C*************************************************************************C
      READ(11,9002) NPWQ
```

```
      READ(11,9001) (WQout(m), m=1,NPWQ)
      WRITE(8,802) NPWQ, (WQout(m), m=1,NPWQ)
      iTmp = NINT( WQout(NPWQ) / DTD )
      IF (iTmp .GT. iTmax)  STOP '** WQout(NPWQ) > max. run time'
  802 FORMAT(/, '* Ouput will be printed ', I2, ' times, at the ',
     *          'following days after model start:', /, 8F10.6)
      iWQTP = NINT( WQout(IncW) / DTD )
      iWQTPs1 = NINT( (WQout(IncW)-1.0) / DTD )
C*******************************************************************C
      DO 10 i=ML,MU
        XAm(i) = 0.0
        n = KBU(i)
        DO 11 k=2,n
          XAL(i,k) = BU(i,k) * H(k)
          XAm(i) = XAm(i) + XAL(i,k)
   11   CONTINUE
   10 CONTINUE

      DO 60 i=ML,MUs1
        n = KB(i) - 1
        DO 61 k=1,n
          Bfrac(i,k) = (B(i,k) - B(i,k+1)) / B(i,k)
   61   CONTINUE
        Bfrac(i,n+1) = 1.0
   60 CONTINUE
C2
      READ(11,9001) TheN12, TheN23, TheN33, TheP12
      WRITE(8,803)'* Exp. base for temp. dependence of rate constants'
      WRITE(8,804)'   : of N hydrolysis                    = ', TheN12,
     *            '   : of Nitrification                   = ', TheN23,
     *            '   : of Denitrification                 = ', TheN33,
     *            '   : of org. P mineralization           = ', TheP12

      READ(11,9002) NS2
      READ(11,9001) (xKn12(i), i=2,NS2)
      READ(11,9001) (xKn23(i), i=2,NS2)
      READ(11,9001) (xKn33(i), i=2,NS2)
      READ(11,9001) (xKp12(i), i=2,NS2)
      IF (NS2 .EQ. 2)  THEN
        DO 20 i=ML,MU
          xKn12(i) = xKn12(2)
          xKn23(i) = xKn23(2)
          xKn33(i) = xKn33(2)
          xKp12(i) = xKp12(2)
   20   CONTINUE
      END IF
      WRITE(8,803)'* Rate constant (mg/L/day) at 20C of                '
      WRITE(8,801)'    Kn12,      Kn23,      Kn33 (/day),      Kp12 '
      WRITE(8,805) (i, ':', xKn12(i),xKn23(i),xKn33(i),xKp12(i),i=ML,MU)
      DO 30 i=ML,MU
        xKn12(i) = xKn12(i) * DTD
        xKn23(i) = xKn23(i) * DTD
        xKn33(i) = xKn33(i) * DTD
        xKp12(i) = xKp12(i) * DTD
   30 CONTINUE
  805 FORMAT(I2, A1, 4F13.5)
      READ(11,9001) rKh12, rKh23,rKnit,rKden, rKhp12, rKDO
      WRITE(8,803)'* Half-satuaration concentration (mg/L)             '
      WRITE(8,804)'   : of N hydrolysis                    = ', rKh12,
     *            '   : of Nitrification                   = ', rKh23,
     *            '   : of Nitrification O2 limitation     = ', rKnit,
     *            '   : of Denitrification                 = ', rKden,
     *            '   : of org. P mineralization           = ', rKhp12,
     *            '   : of SOD for DO & CBOD               = ', rKDO
```

```
C3
      READ(11,9002) NS2
      READ(11,9001) (xKc(i), i=2,NS2)
      IF (NS2 .EQ. 2)  THEN
        DO 40 i=ML,MU
          xKc(i) = xKc(2)
  40    CONTINUE
      END IF
      READ(11,9001) TheKc
      READ(11,9001) TheDA, rKro
      WRITE(8,803)'* CBOD decay rate (/day) at 20C                    '
      WRITE(8,806) (i, ':', xKc(i), i=ML,MU)
      WRITE(8,804)': Exp.base. for temp.dep. for CBOD decay =', TheKc,
     *            ': Exp.base. for temp.dep. for DO reaerat =', TheDA,
     *            ': Constant for DO reaeration in CGS unit =', rKro
      rKro = rKro / SQRT(2.0)
      DO 50 i=ML,MU
        xKc(i) = xKc(i) * DTD
 50   CONTINUE
 806 FORMAT(I2, A1, F10.5)
 9001 FORMAT(10F8.0)
 9002 FORMAT(I5)
 803 FORMAT(/, A50)
 804 FORMAT(/, (A42, F10.5))
      RETURN
      END
C***************
      SUBROUTINE WQC2
C*************************************************************************C
C Read in constants from 'WQ.IN'.
C1 related to Settling rates
C2 related to Phytoplankton related coefficients
C: Since CH2O+O2 = H2O+CO2, aco (ratio of O demand to organic C recycled)
C  = O2/C = 32/14 = 2.67.
C*************************************************************************
      INCLUDE 'COM-WQ1.INC'
      Parameter  (aco=2.67)
C1
      READ(11,9002) NS2
      READ(11,9001) (rKn11(i), i=2,NS2)
      READ(11,9001) (rKp11(i), i=2,NS2)
      READ(11,9001) (rKp22(i), i=2,NS2)
      READ(11,9001) (rKchl(i), i=2,NS2)
      READ(11,9001) (rKbod(i), i=2,NS2)
      IF (NS2 .EQ. 2)  THEN
        DO 10 i=ML,MU
          rKn11(i) = rKn11(2)
          rKp11(i) = rKp11(2)
          rKp22(i) = rKp22(2)
          rKchl(i) = rKchl(2)
          rKbod(i) = rKbod(2)
  10    CONTINUE
      END IF
      WRITE(8,801)'* Settling and loss rate (cm/day) of               '
      WRITE(8,802)'      N1,       P1,       P2,         Chl,    CBOD'
      WRITE(8,803) (i, ':', rKn11(i), rKp11(i), rKp22(i), rKchl(i),
     *                      rKbod(i), i=ML,MU)
      DO 20 i=ML,MU
        rKn11(i) = rKn11(i) * DTD
        rKp11(i) = rKp11(i) * DTD
        rKp22(i) = rKp22(i) * DTD
        rKchl(i) = rKchl(i) * DTD
        rKbod(i) = rKbod(i) * DTD
 20   CONTINUE
```

```
C2
      READ(11,9001) ac,an,ap,ar, PQ,RQ, rKmn,rKmp, xKgr,rIs,
     *              Resp20,Graz20, Fn,Fp, ThetaG,ThetaR,ThetaD
      WRITE(8,801)'* Phytoplankton (pp) related coefficients          '
      WRITE(8,804)': Carbon/Chl ratio (mgC to ugChl) in pp    =',    ac,
     *             '  N/Chl ratio (mgN to ugChl) in pp        =',    an,
     *             '  org.P/Chl ratio (mgP to ugChl) in pp    =',    ap,
     *             ': Proportion of consumed pp recycled by zp =',    ar,
     *             ': Photosynthesis quotient (moles O2/mole C)=',    PQ,
     *             '  Respiration quotient (moles CO2/mole O2) =',    RQ,
     *             ': Half-sat. conc. (mg/L) for inorg.N uptake=', rKmn,
     *             '                    for orthophosphorus uptake=', rKmp,
     *             ': Optimum growth rate of pp (/day) at 20C) =', xKgr,
     *             ': Optimum solar radiation rate (langleys/d)=',  rIs,
     *             ': Endogenous resp. rate (/day) at 20C      =',Resp20,
     *             ': Grazing-other mortality rate (/d) at 20C =',Graz20,
     *             ': Fraction of made N recycled to org. pool =',    Fn,
     *             '    of metabolically produced P recycled   =',    Fp,
     *             ': Exp. base for temp. dependency of growth =',ThetaG,
     *             '                                of respiration =',ThetaR,
     *             '                  of grazing & other mortality =',ThetaD
      ac = ac * aco
      PQ = PQ * ac
      xKgr = xKgr * DTD
      Resp20 = Resp20 * DTD
      Graz20 = Graz20 * DTD
 9001 FORMAT(10F8.0)
 9002 FORMAT(I5)
  801 FORMAT(/, A50)
  802 FORMAT(A50)
  803 FORMAT(I2, A1, 5F10.5)
  804 FORMAT((2X, A44, F15.5))
      RETURN
      END
C***************
      SUBROUTINE WQIC
C*************************************************************************C
C Read in initial conditions.
C*************************************************************************C
      INCLUDE 'COM-WQ1.INC'

 9002 FORMAT(I5)
  801 FORMAT(/, A38, F10.5)
  802 FORMAT(I2, A1, 20F7.4)
  803 FORMAT(I2, A1, 20F7.3)
 9001 FORMAT(10F8.3)
      READ(11,9001) xN1,xN2,xN3,xP1,xP2,xChl,xBOD,xDO
      DO 100 i=ML,MU
        n = KB(i)
        DO 101 k=1,n
          rN1IC(i,k) = xN1
          rN2IC(i,k) = xN2
          rN3IC(i,k) = xN3
          P1IC(i,k) = xP1
          P2IC(i,k) = xP2
          ChlIC(i,k) = xChl
          BODIC(i,k) = xBOD
          DOIC(i,k) = xDO
          N1(i,k,2) = xN1
          N2(i,k,2) = xN2
          N3(i,k,2) = xN3
          P1(i,k,2) = xP1
          P2(i,k,2) = xP2
          Chl(i,k,2) = xChl
```

```
        CBOD(i,k,2) = xBOD
        DOC(i,k,2) = xDO
101     CONTINUE
        StN1(i) = xN1
        StN2(i) = xN2
        StN3(i) = xN3
        StP1(i) = xP1
        StP2(i) = xP2
        StChl(i) = xChl
        StBOD(i) = xBOD
        StDO(i) = xDO
100   CONTINUE
      READ(11,9002) NS2WQ
      IF (NS2WQ .EQ. 1)  THEN
        READ(11,9001) TWQIC
        WRITE(8,801)'* Time (in d) to specify the WQ ICs = ', TWQIC
        iTWQIC = NINT(TWQIC/DTD)
        WRITE(8,801)': Initial condition for N1            '
        DO 10 i=ML,MU
          n = KB(i)
          READ(11,9001) (rN1IC(i,k), k=1,n)
          WRITE(8,802) i, ':', (rN1IC(i,k), k=1,n)
 10     CONTINUE
        WRITE(8,801)': Initial condition for N2            '
        DO 20 i=ML,MU
          n = KB(i)
          READ(11,9001) (rN2IC(i,k), k=1,n)
          WRITE(8,802) i, ':', (rN2IC(i,k), k=1,n)
 20     CONTINUE
        WRITE(8,801)': Initial condition for N3            '
        DO 30 i=ML,MU
          n = KB(i)
          READ(11,9001) (rN3IC(i,k), k=1,n)
          WRITE(8,802) i, ':', (rN3IC(i,k), k=1,n)
 30     CONTINUE
        WRITE(8,801)': Initial condition for P1            '
        DO 40 i=ML,MU
          n = KB(i)
          READ(11,9001) (P1IC(i,k), k=1,n)
          WRITE(8,802) i, ':', (P1IC(i,k), k=1,n)
 40     CONTINUE
        WRITE(8,801)': Initial condition for P2            '
        DO 50 i=ML,MU
          n = KB(i)
          READ(11,9001) (P2IC(i,k), k=1,n)
          WRITE(8,802) i, ':', (P2IC(i,k), k=1,n)
 50     CONTINUE
        WRITE(8,801)': Initial condition for Chl           '
        DO 60 i=ML,MU
          n = KB(i)
          READ(11,9001) (ChlIC(i,k), k=1,n)
          WRITE(8,802) i, ':', (ChlIC(i,k), k=1,n)
 60     CONTINUE
        WRITE(8,801)': Initial condition for CBOD          '
        DO 70 i=ML,MU
          n = KB(i)
          READ(11,9001) (BODIC(i,k), k=1,n)
          WRITE(8,802) i, ':', (BODIC(i,k), k=1,n)
 70     CONTINUE
        WRITE(8,801)': Initial condition for DO            '
        DO 80 i=ML,MU
          n = KB(i)
          READ(11,9001) (DOIC(i,k), k=1,n)
          WRITE(8,802) i, ':', (DOIC(i,k), k=1,n)
```

```
 80     CONTINUE
      END IF

      DO 90 i=ML,MU
        n = KB(i)
        DO 91 k=1,n
          N1(i,k,2) = rN1IC(i,k)
          N2(i,k,2) = rN2IC(i,k)
          N3(i,k,2) = rN3IC(i,k)
          P1(i,k,2) = P1IC(i,k)
          P2(i,k,2) = P2IC(i,k)
          Chl(i,k,2) = ChlIC(i,k)
          CBOD(i,k,2) = BODIC(i,k)
          DOC(i,k,2) = DOIC(i,k)
 91     CONTINUE
        StN1(i) = rN1IC(i,1)
        StN2(i) = rN2IC(i,1)
        StN3(i) = rN3IC(i,1)
        StP1(i) = P1IC(i,1)
        StP2(i) = P2IC(i,1)
        StChl(i) = ChlIC(i,1)
        StBOD(i) = BODIC(i,1)
        StDO(i) = DOIC(i,1)
 90   CONTINUE
      RETURN
      END
C***************
      SUBROUTINE WQinput
C*************************************************************************C
C Read in parameters that may vary w/t time.
C*************************************************************************C
      INCLUDE 'COM-WQ1.INC'
      Character  Title*50
      Dimension xBnN1(iX,iZ),xBnN2(iX,iZ),xBnN3(iX,iZ),xBnP1(iX,iZ),
     *  xBnP2(iX,iZ),xBnBOD(iX,iZ),xBnDO(iX,iZ)
      Dimension xBN1(iX),xBN2(iX),xBN3(iX),xBP1(iX),xBP2(iX),xBBOD(iX),
     *  xBDO(iX)
      Dimension xBN1St(iX),xBN2St(iX),xBN3St(iX),xBP1St(iX),xBP2St(iX),
     *  xBBODSt(iX),xBDOSt(iX)
      Save/Local/,xBnN1,xBnN2,xBnN3,xBnP1,xBnP2,xBnBOD,xBnDO,
     *  xBN1St,xBN2St,xBN3St,xBP1St,xBP2St,xBBODSt,xBDOSt,
     *  TBN1,TBN2,TBN3,TBP1,TBP2,TBBOD,TBDO
      Data xBnN1,xBnN2,xBnN3,xBnP1,xBnP2,xBnBOD,xBnDO/8400*0.0/

      READ(11,802) Title
      WRITE(8,801) Title
      READ(11,9001) NDG, NS2

      DO WHILE (NDG .NE. 99)
C1
C Set UpConc = 0 if non-point source is specified.

      IF (NDG .EQ. 1)  THEN
        IF (iBCWQ .EQ. 1)  THEN
          iU = 17
          WRITE(8,801)'*1 Time-varying boundary conditions            '
         ELSE
          iU = 11
          WRITE(8,801)'*1 Time constant boundary conditions           '
        END IF
        n = KB(MLs1)
        READ(iU,9006) (N1(MLs1,k,2), k=1,n)
        READ(iU,9006) (N2(MLs1,k,2), k=1,n)
        READ(iU,9006) (N3(MLs1,k,2), k=1,n)
```

```
        READ(iU,9006) (P1(MLs1,k,2), k=1,n)
        READ(iU,9006) (P2(MLs1,k,2), k=1,n)
        READ(iU,9006) (Chl(MLs1,k,2), k=1,n)
        READ(iU,9006) (CBOD(MLs1,k,2), k=1,n)
        READ(iU,9006) (DOC(MLs1,k,2), k=1,n)
        DO 100 k=1,n
          N1(MLs1,k,1)   = N1(MLs1,k,2)
          N2(MLs1,k,1)   = N2(MLs1,k,2)
          N3(MLs1,k,1)   = N3(MLs1,k,2)
          P1(MLs1,k,1)   = P1(MLs1,k,2)
          P2(MLs1,k,1)   = P2(MLs1,k,2)
          Chl(MLs1,k,1)  = Chl(MLs1,k,2)
          CBOD(MLs1,k,1) = CBOD(MLs1,k,2)
          DOC(MLs1,k,1)  = DOC(MLs1,k,2)
  100   CONTINUE
        n = KB(MU)
        READ(iU,9006) (DnN1(k), k=1,n)
        READ(iU,9006) (DnN2(k), k=1,n)
        READ(iU,9006) (DnN3(k), k=1,n)
        READ(iU,9006) (DnP1(k), k=1,n)
        READ(iU,9006) (DnP2(k), k=1,n)
        READ(iU,9006) (DnChl(k), k=1,n)
        READ(iU,9006) (DnBOD(k), k=1,n)
        READ(iU,9006) (DnDO(k), k=1,n)
        WRITE(8,802)'        N1,      N2,      N3,      P1,      P2,  ',
     *              '   Chl,    CBOD,      DO                          '
        WRITE(8,801)': Upstream boundary                                '
        n = KB(MLs1)
        WRITE(8,803) (k, ':', N1(MLs1,k,2),N2(MLs1,k,2),N3(MLs1,k,2),
     *    P1(MLs1,k,2),P2(MLs1,k,2), Chl(MLs1,k,2),CBOD(MLs1,k,2),
     *    DOC(MLs1,k,2), k=1,n)
        WRITE(8,801)': Downstream (open) boundary                       '
        n = KB(MU)
        WRITE(8,803) (k, ':', DnN1(k),DnN2(k),DnN3(k), DnP1(k),DnP2(k),
     *                        DnChl(k),DnBOD(k),DnDO(k), k=1,n)
C2
C PN1 in kg/d and point source loadings (WPN1) will be divided by Vol;
C: WPN1 = PN1(kg/d) / Vol(cm**3) = C mg/L/d (C = 1.0E9)

       ELSE IF (NDG .EQ. 2)  THEN
        IF (iWQPS .EQ. 1)  THEN
          iUPS = 19
          WRITE(8,801)'*2 Daily point source input in kg/day            '
         ELSE
          iUPS = 11
          WRITE(8,801)'*2 Constant point source input in kg/day        '
        END IF
        WRITE(8,802)'  PSQ(m**3/s),  N1,      N2,      N3,      P1,   ',
     *              '   P2,  CBOD,       DO                          '
         x1 = 1.0E9 / DX * DTD
        DO 10 m=1,NS2
          READ(iUPS,9003) i, xPSQ, PN1,PN2,PN3, PP1,PP2, PBOD,PDO
          WRITE(8,803) i, ':', xPSQ, PN1,PN2,PN3, PP1,PP2, PBOD,PDO
            x2 = x1 / TotB(i)
          WPN1(i) = PN1 * x2
          WPN2(i) = PN2 * x2
          WPN3(i) = PN3 * x2
          WPP1(i) = PP1 * x2
          WPP2(i) = PP2 * x2
          WPBOD(i) = PBOD * x2
          WPDO(i) = PDO * x2
  10    CONTINUE
C3
C QLat(i,1) in cm**3/s & DSN1 in mg/L (DSChl in ug/L)
```

```
C  and non-point source loadings (WN1) will be divided by Vol;
C: WN1 = QLat(cm**3/s) * DSN1(mg/L) / Vol(cm**3) = C mg/L/d (C = 8.64E4)
C: need to /B(i,1) in GetWNPS

      ELSE IF (NDG .EQ. 3)  THEN
       IF (iNPS .EQ. 1) THEN
         iU = 18
         WRITE(8,801)'*3 Daily NPS input through UB in mg/L          '
        ELSE
         iU = 11
         WRITE(8,801)'*3 Constant NPS input through UB in mg/L       '
       END IF
       READ(iU,9003) i, DSQ,DSN1,DSN2,DSN3,DSP1,DSP2,DSChl,DSBOD,DSDO
       WRITE(8,802)'    Q(m**3/s),  N1,      N2,      N3,      P1,   ',
     *              '  P2,  Chl(ug/L),  CBOD,       DO                '
       WRITE(8,804) i,':',DSQ,DSN1,DSN2,DSN3,DSP1,DSP2,DSChl,DSBOD,DSDO
        x3 = 8.64E4 / DX * DTD
       DSN1 = DSN1 * x3
       DSN2 = DSN2 * x3
       DSN3 = DSN3 * x3
       DSP1 = DSP1 * x3
       DSP2 = DSP2 * x3
       DSChl = DSChl * x3
       DSBOD = DSBOD * x3
       DSDO = DSDO * x3
C4
C Negative values, including SOD (xBnDO), are losses to sed.
C: Ben will be divided by h & thus Ben(g/m**2/d) / h(cm) = 100 mg/L/d.

      ELSE IF (NDG .EQ. 4)  THEN
       WRITE(8,801)'*4 Benthic flux release rate (g/m**2/d) at 20C  '
       WRITE(8,802)': will be adjusted by (B(i,k)-B(i,k+1))/B(i,k)   '
       READ(11,9002) (xBN1(i), i=ML,MU)
       READ(11,9002) (xBN1St(i), i=ML,MU)
       READ(11,9002) (xBN2(i), i=ML,MU)
       READ(11,9002) (xBN2St(i), i=ML,MU)
       READ(11,9002) (xBN3(i), i=ML,MU)
       READ(11,9002) (xBN3St(i), i=ML,MU)
       READ(11,9002) (xBP1(i), i=ML,MU)
       READ(11,9002) (xBP1St(i), i=ML,MU)
       READ(11,9002) (xBP2(i), i=ML,MU)
       READ(11,9002) (xBP2St(i), i=ML,MU)
       READ(11,9002) (xBBOD(i), i=ML,MU)
       READ(11,9002) (xBBODSt(i), i=ML,MU)
       READ(11,9002) (xBDO(i), i=ML,MU)
       READ(11,9002) (xBDOSt(i), i=ML,MU)
       WRITE(8,807)'1) N1                                          '
       WRITE(8,9002) (xBN1(i), i=ML,MU)
       WRITE(8,807)'2) N2                                          '
       WRITE(8,9002) (xBN2(i), i=ML,MU)
       WRITE(8,807)'3) N3                                          '
       WRITE(8,9002) (xBN3(i), i=ML,MU)
       WRITE(8,807)'4) P1                                          '
       WRITE(8,9002) (xBP1(i), i=ML,MU)
       WRITE(8,807)'5) P2                                          '
       WRITE(8,9002) (xBP2(i), i=ML,MU)
       WRITE(8,807)'6) CBOD                                        '
       WRITE(8,9002) (xBBOD(i), i=ML,MU)
       WRITE(8,807)'7) DO                                          '
       WRITE(8,9002) (xBDO(i), i=ML,MU)
       WRITE(8,807)'1St) N1                                        '
       WRITE(8,9002) (xBN1St(i), i=ML,MU)
       WRITE(8,807)'2St) N2                                        '
       WRITE(8,9002) (xBN2St(i), i=ML,MU)
```

```
        WRITE(8,807)'3St) N3                                          '
        WRITE(8,9002) (xBN3St(i), i=ML,MU)
        WRITE(8,807)'4St) P1                                          '
        WRITE(8,9002) (xBP1St(i), i=ML,MU)
        WRITE(8,807)'5St) P2                                          '
        WRITE(8,9002) (xBP2St(i), i=ML,MU)
        WRITE(8,807)'6St) CBOD                                        '
        WRITE(8,9002) (xBBODSt(i), i=ML,MU)
        WRITE(8,807)'7St) DO                                          '
        WRITE(8,9002) (xBDOSt(i), i=ML,MU)
          xx = DTD * 100.0
        DO 19 i=ML,MUs1
          n = KB(i)
          DO 18 k=1,n
            xBnN1(i,k) = xBN1(i) * Bfrac(i,k) * xx
            xBnN2(i,k) = xBN2(i) * Bfrac(i,k) * xx
            xBnN3(i,k) = xBN3(i) * Bfrac(i,k) * xx
            xBnP1(i,k) = xBP1(i) * Bfrac(i,k) * xx
            xBnP2(i,k) = xBP2(i) * Bfrac(i,k) * xx
            xBnBOD(i,k) = xBBOD(i) * Bfrac(i,k) * xx
            xBnDO(i,k) = xBDO(i) * Bfrac(i,k) * xx
 18       CONTINUE
          xBN1St(i) = xBN1St(i) * xx
          xBN2St(i) = xBN2St(i) * xx
          xBN3St(i) = xBN3St(i) * xx
          xBP1St(i) = xBP1St(i) * xx
          xBP2St(i) = xBP2St(i) * xx
          xBBODSt(i) = xBBODSt(i) * xx
          xBDOSt(i) = xBDOSt(i) * xx
 19     CONTINUE
        READ(11,9002) TBN1, TBN2, TBN3, TBP1, TBP2, TBBOD, TBDO
        WRITE(8,802)': Exp. bases for Temp. correction for the above  '
        WRITE(8,9002) TBN1,TBN2,TBN3, TBP1,TBP2, TBBOD, TBDO
C5
C Default value for Turb is 0.01 (/cm).

       ELSE IF (NDG .EQ. 5)  THEN
        READ(11,9002) (Turb(i), i=ML,MU)
        WRITE(8,801)'*5 Light extinction coeff. (/cm) at 0 Chl conc. '
        WRITE(8,806) (i, ':', Turb(i), i=ML,MU)
C6
C Default value for WRea is 0.

       ELSE IF (NDG .EQ. 6)  THEN
        READ(11,9002) (WRea(i), i=ML,MU)
        WRITE(8,801)'*6 Wind-induced reaeration of DO (cm/day)        '
        WRITE(8,806) (i, ':', WRea(i), i=ML,MU)
C7
C Temperature should be read in after reading benthic fluxes since these
C parameters must be temperature corrected.

       ELSE IF (NDG .EQ. 7)  THEN
        READ(11,9002) Temp
        WRITE(8,807)'*7 Temperature (deg C)                    = ', Temp
          Tmp = Temp * Temp
        TDOs = 14.62 - 0.367*Temp + 4.497E-3*Tmp
        STDOs = -9.66E-2 + 2.05E-3*Temp
C        T1 = 5.89E3 + 38.0*Temp - 0.375*Tmp
C        T2 = 5.891E3 + 37.8*Temp - 0.336*Tmp
C        T3 = 1.7 + 0.01*Temp
          x20 = Temp - 20.0
          xN12 = TheN12**x20
          xN23 = TheN23**x20
          xN33 = TheN33**x20
```

```
          xP12 = TheP12**x20
          xxKc = TheKc**x20
          xxBN1 = TBN1**x20
          xxBN2 = TBN2**x20
          xxBN3 = TBN3**x20
          xxBP1 = TBP1**x20
          xxBP2 = TBP2**x20
          xxBBOD = TBBOD**x20
          xxBDO = TBDO**x20
        DO 30 i=ML,MUs1
          rKn12(i) = xKn12(i) * xN12
          rKn23(i) = xKn23(i) * xN23
          rKn33(i) = xKn33(i) * xN33
          rKp12(i) = xKp12(i) * xP12
          rKc(i) = xKc(i) * xxKc
          n = KB(i)
          DO 31 k=1,n
            BenN1(i,k) = xBnN1(i,k) * xxBN1
            BenN2(i,k) = xBnN2(i,k) * xxBN2
            BenN3(i,k) = xBnN3(i,k) * xxBN3
            BenP1(i,k) = xBnP1(i,k) * xxBP1
            BenP2(i,k) = xBnP2(i,k) * xxBP2
            BenBOD(i,k) = xBnBOD(i,k) * xxBBOD
            BenDO(i,k) = xBnDO(i,k) * xxBDO
   31     CONTINUE
          StBN1(i) = xBN1St(i) * xxBN1
          StBN2(i) = xBN2St(i) * xxBN2
          StBN3(i) = xBN3St(i) * xxBN3
          StBP1(i) = xBP1St(i) * xxBP1
          StBP2(i) = xBP2St(i) * xxBP2
          StBBOD(i) = xBBODSt(i) * xxBBOD
          StBDO(i) = xBDOSt(i) * xxBDO
   30   CONTINUE
        RespR = Resp20 * ThetaR**x20
        GrazR = Graz20 * ThetaD**x20
          xDie = RespR + ar*GrazR
        CNdie = an * xDie
        CNdie1 = CNdie * Fn
        CNdie2 = CNdie * (1.0-Fn)
        CPdie = ap * xDie
        CPdie1 = CPdie * Fp
        CPdie2 = CPdie * (1.0-Fp)
        BODdie = ac * ar * GrazR
        DOresp = ac / RQ * RespR
        rKgr = 2.718 * xKgr * ThetaG**x20
        TDOaer = TheDA**x20 * DTD
C8
      ELSE IF (NDG .EQ. 8)  THEN
        IF (iSun .EQ. 1)  THEN
          WRITE(8,801)'*8 Daily-varying solar radiation parameters     '
          IF (NS2 .EQ. 1)  THEN
            READ(16,9006) rIa, TU, TD
            WRITE(8,807)': Hours from midnight to sun rise      = ', TU,
     *                  ':                          to sun set      = ', TD,
     *                  ': Total daily radiation (langleys/day) = ', rIa
           ELSE
            READ(16,9006) rIa
            WRITE(8,807)': Total daily radiation (langleys/day) = ', rIa
          END IF
         ELSE
          WRITE(8,801)'*8 Constant solar radiation parameters          '
          READ(11,9006) rIa, TU, TD
          WRITE(8,807)': Hours from midnight to sun rise      = ', TU,
     *                ':                          to sun set      = ', TD,
```

```
     *                ': Total daily radiation (langleys/day) = ', rIa
        END IF
        PTT = 3.1416 / (TD-TU)
        rIn = - 12.0 * PTT * (rIa/rIs)
      END IF
      READ(11,9001) NDG, NS2
      END DO
 9001 FORMAT(2I5)
 9002 FORMAT(10F8.3)
 9003 FORMAT(9X, I5, 9F7.3)
 9004 FORMAT(15I5)
 9006 FORMAT(9X, 10F8.3)
  801 FORMAT(/, A48)
  802 FORMAT(2A49)
  803 FORMAT(I2, A1, 8F9.3)
  804 FORMAT(I2, A1, 9F9.3)
  805 FORMAT(I2, A1, 7F9.3)
  806 FORMAT(I2, A1, F9.3)
  807 FORMAT(/, (A41, F10.5))
  809 FORMAT(I2, A1, 10F7.3)
      RETURN
      END
C***************
      SUBROUTINE WQmean
C*********************************************************************C
      INCLUDE 'COM-WQ1.INC'
      INCLUDE 'COM-WQ2.INC'

      AvgNWQ = AvgNWQ + 1.0
      DO 2100 i=ML,MU
        n = KB(i)
        DO 2101 k=1,n
          AvgN1(i,k) = AvgN1(i,k) + N1(i,k,2)
          AvgN2(i,k) = AvgN2(i,k) + N2(i,k,2)
          AvgN3(i,k) = AvgN3(i,k) + N3(i,k,2)
          AvgP1(i,k) = AvgP1(i,k) + P1(i,k,2)
          AvgP2(i,k) = AvgP2(i,k) + P2(i,k,2)
          AvgChl(i,k) = AvgChl(i,k) + Chl(i,k,2)
          AvgBOD(i,k) = AvgBOD(i,k) + CBOD(i,k,2)
          AvgDO(i,k) = AvgDO(i,k) + DOC(i,k,2)
          IF (N1(i,k,2) .GT. rN1Max(i,k))  rN1Max(i,k) = N1(i,k,2)
          IF (N2(i,k,2) .GT. rN2Max(i,k))  rN2Max(i,k) = N2(i,k,2)
          IF (N3(i,k,2) .GT. rN3Max(i,k))  rN3Max(i,k) = N3(i,k,2)
          IF (P1(i,k,2) .GT. P1Max(i,k))  P1Max(i,k) = P1(i,k,2)
          IF (P2(i,k,2) .GT. P2Max(i,k))  P2Max(i,k) = P2(i,k,2)
          IF (Chl(i,k,2) .GT. ChlMax(i,k))  ChlMax(i,k) = Chl(i,k,2)
          IF (CBOD(i,k,2) .GT. BODMax(i,k))  BODMax(i,k) = CBOD(i,k,2)
          IF (DOC(i,k,2) .GT. DOMax(i,k))  DOMax(i,k) = DOC(i,k,2)
          IF (N1(i,k,2) .LT. rN1Min(i,k))  rN1Min(i,k) = N1(i,k,2)
          IF (N2(i,k,2) .LT. rN2Min(i,k))  rN2Min(i,k) = N2(i,k,2)
          IF (N3(i,k,2) .LT. rN3Min(i,k))  rN3Min(i,k) = N3(i,k,2)
          IF (P1(i,k,2) .LT. P1Min(i,k))  P1Min(i,k) = P1(i,k,2)
          IF (P2(i,k,2) .LT. P2Min(i,k))  P2Min(i,k) = P2(i,k,2)
          IF (Chl(i,k,2) .LT. ChlMin(i,k))  ChlMin(i,k) = Chl(i,k,2)
          IF (CBOD(i,k,2) .LT. BODMin(i,k))  BODMin(i,k) = CBOD(i,k,2)
          IF (DOC(i,k,2) .LT. DOMin(i,k))  DOMin(i,k) = DOC(i,k,2)
 2101   CONTINUE
 2100 CONTINUE
      RETURN
      END
C***************
      SUBROUTINE WOut
C*********************************************************************C
      INCLUDE 'COM-WQ1.INC'
```

```
      INCLUDE 'COM-WQ2.INC'
      Parameter (C1=1000.0, C2=-1000.0)

      WRITE(8,901)'* N1 (mg/L)          '
      WRITE(8,902) WQout(IncW)
      DO 10 i=ML,MU
        n = KB(i)
        WRITE(8,903) i, ':', (N1(i,k,2), k=1,n)
10    CONTINUE

      WRITE(8,901)'* N2 (mg/L)          '
      WRITE(8,902) WQout(IncW)
      DO 20 i=ML,MU
        n = KB(i)
        WRITE(8,990) i, ':', (N2(i,k,2), k=1,n)
20    CONTINUE

      WRITE(8,901)'* N3 (mg/L)          '
      WRITE(8,902) WQout(IncW)
      DO 30 i=ML,MU
        n = KB(i)
        WRITE(8,990) i, ':', (N3(i,k,2), k=1,n)
30    CONTINUE

      WRITE(8,901)'* P1 (mg/L)          '
      WRITE(8,902) WQout(IncW)
      DO 40 i=ML,MU
        n = KB(i)
        WRITE(8,990) i, ':', (P1(i,k,2), k=1,n)
40    CONTINUE

      WRITE(8,901)'* P2 (mg/L)          '
      WRITE(8,902) WQout(IncW)
      DO 50 i=ML,MU
        n = KB(i)
        WRITE(8,990) i, ':', (P2(i,k,2), k=1,n)
50    CONTINUE

      WRITE(8,901)'* Chl (ug/L)         '
      WRITE(8,902) WQout(IncW)
      DO 60 i=ML,MU
        n = KB(i)
        WRITE(8,904) i, ':', (Chl(i,k,2), k=1,n)
60    CONTINUE

      WRITE(8,901)'* CBOD (mg/L)        '
      WRITE(8,902) WQout(IncW)
      DO 70 i=ML,MU
        n = KB(i)
        WRITE(8,903) i, ':', (CBOD(i,k,2), k=1,n)
70    CONTINUE

      WRITE(8,901)'* DO (mg/L)          '
      WRITE(8,902) WQout(IncW)
      DO 80 i=ML,MU
        n = KB(i)
        WRITE(8,904) i, ':', (DOC(i,k,2), k=1,n)
80    CONTINUE
901   FORMAT(/, A18)
902   FORMAT('    at ', F13.8, ' days after model starts')
903   FORMAT(I2, A1, 10F7.3)
904   FORMAT(I2, A1, 10F7.2)
990   FORMAT(I2, A1, 10F7.4)
```

```
      IF (iWQTPs1 .GE. 0)  THEN
        DO 90 i=ML,MU
          n = KB(i)
          DO 91 k=1,n
            AvgN1(i,k) = AvgN1(i,k) / AvgNWQ
            AvgN2(i,k) = AvgN2(i,k) / AvgNWQ
            AvgN3(i,k) = AvgN3(i,k) / AvgNWQ
            AvgP1(i,k) = AvgP1(i,k) / AvgNWQ
            AvgP2(i,k) = AvgP2(i,k) / AvgNWQ
            AvgChl(i,k) = AvgChl(i,k) / AvgNWQ
            AvgBOD(i,k) = AvgBOD(i,k) / AvgNWQ
            AvgDO(i,k) = AvgDO(i,k) / AvgNWQ
 91       CONTINUE
 90     CONTINUE

 905 FORMAT('    over a day before ', F13.8, ' days after start')
        WRITE(8,901)'* Max N1 (mg/L)   '
        WRITE(8,905) WQout(IncW)
        DO 100 i=ML,MU
          n = KB(i)
          WRITE(8,903) i, ':', (rN1Max(i,k), k=1,n)
 100    CONTINUE
        WRITE(8,901)'* Min N1 (mg/L)   '
        WRITE(8,905) WQout(IncW)
        DO 110 i=ML,MU
          n = KB(i)
          WRITE(8,903) i, ':', (rN1Min(i,k), k=1,n)
 110    CONTINUE
        WRITE(8,901)'* Mean N1 (mg/L)  '
        WRITE(8,905) WQout(IncW)
        DO 120 i=ML,MU
          n = KB(i)
          WRITE(8,903) i, ':', (AvgN1(i,k), k=1,n)
 120    CONTINUE

        WRITE(8,901)'* Max N2 (mg/L)   '
        WRITE(8,905) WQout(IncW)
        DO 130 i=ML,MU
          n = KB(i)
          WRITE(8,990) i, ':', (rN2Max(i,k), k=1,n)
 130    CONTINUE
        WRITE(8,901)'* Min N2 (mg/L)   '
        WRITE(8,905) WQout(IncW)
        DO 140 i=ML,MU
          n = KB(i)
          WRITE(8,990) i, ':', (rN2Min(i,k), k=1,n)
 140    CONTINUE
        WRITE(8,901)'* Mean N2 (mg/L)  '
        WRITE(8,905) WQout(IncW)
        DO 150 i=ML,MU
          n = KB(i)
          WRITE(8,990) i, ':', (AvgN2(i,k), k=1,n)
 150    CONTINUE

        WRITE(8,901)'* Max N3 (mg/L)   '
        WRITE(8,905) WQout(IncW)
        DO 160 i=ML,MU
          n = KB(i)
          WRITE(8,990) i, ':', (rN3Max(i,k), k=1,n)
 160    CONTINUE
        WRITE(8,901)'* Min N3 (mg/L)   '
        WRITE(8,905) WQout(IncW)
        DO 170 i=ML,MU
          n = KB(i)
```

```
        WRITE(8,990) i, ':', (rN3Min(i,k), k=1,n)
170     CONTINUE
        WRITE(8,901)'* Mean N3 (mg/L)  '
        WRITE(8,905) WQout(IncW)
        DO 180 i=ML,MU
          n = KB(i)
          WRITE(8,990) i, ':', (AvgN3(i,k), k=1,n)
180     CONTINUE

        WRITE(8,901)'* Max P1 (mg/L)   '
        WRITE(8,905) WQout(IncW)
        DO 190 i=ML,MU
          n = KB(i)
          WRITE(8,990) i, ':', (P1Max(i,k), k=1,n)
190     CONTINUE
        WRITE(8,901)'* Min P1 (mg/L)   '
        WRITE(8,905) WQout(IncW)
        DO 200 i=ML,MU
          n = KB(i)
          WRITE(8,990) i, ':', (P1Min(i,k), k=1,n)
200     CONTINUE
        WRITE(8,901)'* Mean P1 (mg/L)  '
        WRITE(8,905) WQout(IncW)
        DO 210 i=ML,MU
          n = KB(i)
          WRITE(8,990) i, ':', (AvgP1(i,k), k=1,n)
210     CONTINUE

        WRITE(8,901)'* Max P2 (mg/L)   '
        WRITE(8,905) WQout(IncW)
        DO 220 i=ML,MU
          n = KB(i)
          WRITE(8,990) i, ':', (P2Max(i,k), k=1,n)
220     CONTINUE
        WRITE(8,901)'* Min P2 (mg/L)   '
        WRITE(8,905) WQout(IncW)
        DO 230 i=ML,MU
          n = KB(i)
          WRITE(8,990) i, ':', (P2Min(i,k), k=1,n)
230     CONTINUE
        WRITE(8,901)'* Mean P2 (mg/L)  '
        WRITE(8,905) WQout(IncW)
        DO 240 i=ML,MU
          n = KB(i)
          WRITE(8,990) i, ':', (AvgP2(i,k), k=1,n)
240     CONTINUE

        WRITE(8,901)'* Max Chl (ug/L)  '
        WRITE(8,905) WQout(IncW)
        DO 250 i=ML,MU
          n = KB(i)
          WRITE(8,904) i, ':', (ChlMax(i,k), k=1,n)
250     CONTINUE
        WRITE(8,901)'* Min Chl (ug/L)  '
        WRITE(8,905) WQout(IncW)
        DO 260 i=ML,MU
          n = KB(i)
          WRITE(8,904) i, ':', (ChlMin(i,k), k=1,n)
260     CONTINUE
        WRITE(8,901)'* Mean Chl (ug/L) '
        WRITE(8,905) WQout(IncW)
        DO 270 i=ML,MU
          n = KB(i)
          WRITE(8,904) i, ':', (AvgChl(i,k), k=1,n)
```

```
270     CONTINUE

        WRITE(8,901)'* Max CBOD (mg/L) '
        WRITE(8,905) WQout(IncW)
        DO 280 i=ML,MU
          n = KB(i)
          WRITE(8,903) i, ':', (BODMax(i,k), k=1,n)
280     CONTINUE
        WRITE(8,901)'* Min CBOD (mg/L) '
        WRITE(8,905) WQout(IncW)
        DO 290 i=ML,MU
          n = KB(i)
          WRITE(8,903) i, ':', (BODMin(i,k), k=1,n)
290     CONTINUE
        WRITE(8,901)'* Mean CBOD (mg/L)'
        WRITE(8,905) WQout(IncW)
        DO 300 i=ML,MU
          n = KB(i)
          WRITE(8,903) i, ':', (AvgBOD(i,k), k=1,n)
300     CONTINUE

        WRITE(8,901)'* Max DO (mg/L)    '
        WRITE(8,905) WQout(IncW)
        DO 310 i=ML,MU
          n = KB(i)
          WRITE(8,904) i, ':', (DOMax(i,k), k=1,n)
310     CONTINUE
        WRITE(8,901)'* Min DO (mg/L)    '
        WRITE(8,905) WQout(IncW)
        DO 320 i=ML,MU
          n = KB(i)
          WRITE(8,904) i, ':', (DOMin(i,k), k=1,n)
320     CONTINUE
        WRITE(8,901)'* Mean DO (mg/L)   '
        WRITE(8,905) WQout(IncW)
        DO 330 i=ML,MU
          n = KB(i)
          WRITE(8,904) i, ':', (AvgDO(i,k), k=1,n)
330     CONTINUE

        AvgNWQ = 0.0
        DO 340 i=ML,MU
          n= KB(i)
          DO 341 k=1,n
            AvgN1(i,k) = 0.0
            AvgN2(i,k) = 0.0
            AvgN3(i,k) = 0.0
            AvgP1(i,k) = 0.0
            AvgP2(i,k) = 0.0
            AvgChl(i,k) = 0.0
            AvgBOD(i,k) = 0.0
            AvgDO(i,k) = 0.0
            rN1Max(i,k) = C2
            rN2Max(i,k) = C2
            rN3Max(i,k) = C2
            P1Max(i,k) = C2
            P2Max(i,k) = C2
            ChlMax(i,k) = C2
            BODMax(i,k) = C2
            DOMax(i,k) = C2
            rN1Min(i,k) = C1
            rN2Min(i,k) = C1
            rN3Min(i,k) = C1
            P1Min(i,k) = C1
```

```
            P2Min(i,k) = C1
            ChlMin(i,k) = C1
            BODMin(i,k) = C1
            DOMin(i,k) = C1
  341     CONTINUE
 340    CONTINUE
      END IF
      RETURN
      END
C***************
      BLOCK DATA WData
C*******************************************************************************C
      INCLUDE 'COM-WQ1.INC'
      INCLUDE 'COM-WQ2.INC'
      INCLUDE 'BLKD-WQ.INC'
      END
```

B-4. WQ-2D.FOR

```
C***************
      SUBROUTINE WReset
C**********************************************************************C
C Reset WQ variables (WQC) for a new time step for i=ML to MU.
C**********************************************************************C
      INCLUDE 'COM-WQ1.INC'

      DO 110 i=MLs1,MU
      n = KB(i)
      DO 111 k=1,n
        N1(i,k,1) = N1(i,k,2)
        N2(i,k,1) = N2(i,k,2)
        N3(i,k,1) = N3(i,k,2)
        P1(i,k,1) = P1(i,k,2)
        P2(i,k,1) = P2(i,k,2)
        Chl(i,k,1) = Chl(i,k,2)
        CBOD(i,k,1) = CBOD(i,k,2)
        DOC(i,k,1) = DOC(i,k,2)
  111 CONTINUE
  110 CONTINUE
      RETURN
      END
C***************
      SUBROUTINE GetWNPS
C**********************************************************************C
C: QLat(i,1) from SUBR GetQLat
C: Assume (conc in lateral inflow into segment i) = (conc at fall line).
C**********************************************************************C
      INCLUDE 'COM-WQ1.INC'

      DO 10 i=ML,MUs1
        WN1(i) = QLat(i,1) * DSN1 / B(i,1)
        WN2(i) = QLat(i,1) * DSN2 / B(i,1)
        WN3(i) = QLat(i,1) * DSN3 / B(i,1)
        WP1(i) = QLat(i,1) * DSP1 / B(i,1)
        WP2(i) = QLat(i,1) * DSP2 / B(i,1)
        WChl(i) = QLat(i,1) * DSChl / B(i,1)
        WBOD(i) = QLat(i,1) * DSBOD / B(i,1)
        WDO(i) = QLat(i,1) * DSDO / B(i,1)
  10    CONTINUE
      RETURN
      END
C***************
      SUBROUTINE Phyto
C**********************************************************************C
C Calculate for i=ML-MUs1 pp growth term (/day) in main channel (G) & in
C sfc storage area (StG), and preference of pp for N2 (PR2, StPR2) & N3
C (PR3, StPR3) : m = main channel and sa = storage area.
C: G = StG = 0 during night.
C: Since PR/StPR occurs only with G/StG, don't care about these at night.
C**********************************************************************C
      INCLUDE 'COM-WQ1.INC'
      Parameter  (C1=1.8E-4)

      IF (Hour .GT. TU  .AND.  Hour .LT. TD)  THEN
        rLite = rIn * SIN( PTT * (Hour-TU) )
        ELite = EXP(rLite)
        DO 10 i=ML,MUs1
C-m
C For k=1, Hs=0 & TLite = EXP(rLite * EXP(-Turb(i) * Hs)) = ELite
           TLite = ELite
```

```
            n = KB(i)
            DO 11 k=1,n
              Hb = Depth(i,k) + Ada(i,1)
              IF (k .EQ. 1)  THEN
                dz = Hb
               ELSE
                dz = H(k)
              END IF
              rKe = Turb(i)
              IF (Chl(i,k,1) .GT. 0.0)  rKe = rKe + C1*Chl(i,k,1)
              BLite = EXP(rLite / EXP(rKe*Hb))
              rKe = rKe * dz

              IF (N2(i,k,1) .GT. 0.0)  THEN
                IF (N3(i,k,1) .GT. 0.0)  THEN
                  xN = N2(i,k,1) + N3(i,k,1)
                  rNutN = xN / (xN + rKmn)
                  PR2(i,k) = ( N3(i,k,1)/(N2(i,k,1)+rKmn) + rKmn/xN )
     *                     * N2(i,k,1) / (N3(i,k,1)+rKmn)
                 ELSE
                  rNutN = N2(i,k,1) / (N2(i,k,1) + rKmn)
                  PR2(i,k) = 1.0
                END IF
               ELSE
                PR2(i,k) = 0.0
                IF (N3(i,k,1) .GT. 0.0)  THEN
                  rNutN = N3(i,k,1) / (N3(i,k,1) + rKmn)
                 ELSE
                  rNutN = 0.0
                END IF
              END IF
              PR3(i,k) = 1.0 - PR2(i,k)
              IF (P2(i,k,1) .GT. 0.0)  THEN
                rNutP = P2(i,k,1) / (P2(i,k,1) + rKmp)
               ELSE
                rNutP = 0.0
              END IF
              G(i,k) = rKgr/rKe * (BLite-TLite) * AMIN1(rNutN,rNutP)
              TLite = BLite
   11       CONTINUE
C-St
            IF (SST(i) .NE. 0.0  .AND.  StH1(i,2) .GT. 0.0)  THEN
              IF (StN2(i) .GT. 0.0)  THEN
                IF (StN3(i) .GT. 0.0)  THEN
                  xStN = StN2(i) + StN3(i)
                  SNutN = xStN / (xStN + rKmn)
                  StPR2(i) = ( StN3(i)/(StN2(i)+rKmn) + rKmn/xStN )
     *                     * StN2(i) / (StN3(i)+rKmn)
                 ELSE
                  SNutN = StN2(i) / (StN2(i) + rKmn)
                  StPR2(i) = 1.0
                END IF
               ELSE
                StPR2(i) = 0.0
                IF (StN3(i) .GT. 0.0)  THEN
                  SNutN = StN3(i) / (StN3(i) + rKmn)
                 ELSE
                  SNutN = 0.0
                END IF
              END IF
              StPR3(i) = 1.0 - StPR2(i)
              IF (StP2(i) .GT. 0.0)  THEN
                SNutP = StP2(i) / (StP2(i) + rKmp)
               ELSE
```

```
            SNutP = 0.0
          END IF

          StKe = Turb(i) * StH1(i,1)
          IF (StChl(i) .GT. 0.0) StKe = StKe + C1*StChl(i)*StH1(i,1)
          StBL = EXP( rLite / EXP(StKe) )
          StG(i) = rKgr/StKe * (StBL-ELite) * AMIN1(SNutN,SNutP)
        END IF
 10   CONTINUE
      ELSE
       DO 20 i=ML,MUs1
         StG(i) = 0.0
         n = KB(i)
         DO 21 k=1,n
           G(i,k) = 0.0
 21      CONTINUE
 20    CONTINUE
     END IF
     RETURN
     END
C***************
     SUBROUTINE EqnN1
C*************************************************************************C
C Solve the MASS BALANCE Eqn. for N1 to get N1(i,k,2) for i=ML-MUs1.
C: Since Hyd is calculated here, this SUBR be called prior to EqnN2.
C*************************************************************************C
     INCLUDE 'COM-WQ1.INC'
     Dimension A1(iX,iZ),A2(iX,iZ), BB(iZ),xC(iZ),xD(iZ),xE(iZ)
     Data A1,A2, BB,xC,xD,xE, Adj/2461*0.0/

     IF (iHAdv .EQ. 3)  THEN
       DO 20 i=ML,MUs1
         n = KB(i)
         DO 24 k=1,n
           CURVN1(i,k) = N1(i+1,k,1)-2.0*N1(i,k,1)+N1(i-1,k,1)
           CURVN2(i,k) = N2(i+1,k,1)-2.0*N2(i,k,1)+N2(i-1,k,1)
           CURVN3(i,k) = N3(i+1,k,1)-2.0*N3(i,k,1)+N3(i-1,k,1)
           CURVP1(i,k) = P1(i+1,k,1)-2.0*P1(i,k,1)+P1(i-1,k,1)
           CURVP2(i,k) = P2(i+1,k,1)-2.0*P2(i,k,1)+P2(i-1,k,1)
           CURVChl(i,k) = Chl(i+1,k,1)-2.0*Chl(i,k,1)+Chl(i-1,k,1)
           CURVBOD(i,k) = CBOD(i+1,k,1)-2.0*CBOD(i,k,1)+CBOD(i-1,k,1)
           CURVDO(i,k) = DOC(i+1,k,1)-2.0*DOC(i,k,1)+DOC(i-1,k,1)
 24      CONTINUE
         ns1 = KB(i-1) + 1
         IF (n .GE. ns1)  THEN
           DO 22 k=ns1,n
             CURVN1(i,k) = N1(i+1,k,1)-N1(i,k,1)
             CURVN2(i,k) = N2(i+1,k,1)-N2(i,k,1)
             CURVN3(i,k) = N3(i+1,k,1)-N3(i,k,1)
             CURVP1(i,k) = P1(i+1,k,1)-P1(i,k,1)
             CURVP2(i,k) = P2(i+1,k,1)-P2(i,k,1)
             CURVChl(i,k) = Chl(i+1,k,1)-Chl(i,k,1)
             CURVBOD(i,k) = CBOD(i+1,k,1)-CBOD(i,k,1)
             CURVDO(i,k) = DOC(i+1,k,1)-DOC(i,k,1)
 22        CONTINUE
         END IF
         np1 = KB(i+1) + 1
         IF (n .GE. np1)  THEN
           DO 23 k=np1,n
             CURVN1(i,k) = -N1(i,k,1)+N1(i-1,k,1)
             CURVN2(i,k) = -N2(i,k,1)+N2(i-1,k,1)
             CURVN3(i,k) = -N3(i,k,1)+N3(i-1,k,1)
             CURVP1(i,k) = -P1(i,k,1)+P1(i-1,k,1)
             CURVP2(i,k) = -P2(i,k,1)+P2(i-1,k,1)
```

```
            CURVChl(i,k) = -Chl(i,k,1)+Chl(i-1,k,1)
            CURVBOD(i,k) = -CBOD(i,k,1)+CBOD(i-1,k,1)
            CURVDO(i,k) = -DOC(i,k,1)+DOC(i-1,k,1)
 23       CONTINUE
        END IF
20    CONTINUE
     END IF
     DO 10 i=MLs1,MUs1
       n = KB(i)
       DO 11 k=1,n
         IF (iHAdv .EQ. 2)  THEN
           A1(i,k) = (Gam(i+1,k)*N1(i,k,1) + Del(i+1,k)*N1(i+1,k,1))
    *              * HAdv(i,k)
          ELSE IF (iHAdv .EQ. 3)  THEN
           Adj = (N1(i,k,1)+N1(i+1,k,1))*0.5
    *          - COUR2(i,k)*(N1(i+1,k,1)-N1(i,k,1))
           IF (U(i+1,k,1) .GE. 0.0)  THEN
             A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVN1(i,k))
            ELSE
             A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVN1(i+1,k))
           END IF
         END IF
         A2(i,k) = HDif(i,k)*(N1(i+1,k,1) - N1(i,k,1))
 11    CONTINUE
10   CONTINUE

     DO 800 i=ML,MUs1
       Sink = 0.0
       IF (SST(i) .NE. 0.0)  THEN
         IF (StH1(i,2) .GT. 0.0)  THEN
           HydNSt(i) = rKn12(i)*StN1(i) / (rKh12+StN1(i))
           Bio = CNdie1*StChl(i) - HydNSt(i)
    *        + (StBN1(i) - rKn11(i)*StN1(i))/StH1(i,1)
           IF (DelA(i) .GT. 0.0)  THEN
             Sink = N1(i,1,1) * SDoDT(i)
             Phys = (StN1(i)*StH1(i,1) + N1(i,1,1)*DelA(i))/StH1(i,2)
             StN1(i) = Phys + Bio
            ELSE
             Sink = StN1(i) * SDoDT(i)
             StN1(i) = StN1(i) + Bio
           END IF
          ELSE
           StN1(i) = 0.0
         END IF
       END IF
       HydN(i,1) = rKn12(i)*N1(i,1,1) / (rKh12+N1(i,1,1))
       SoN1 = CNdie1*Chl(i,1,1) - HydN(i,1) + ( BenN1(i,1)
    *    - rKn11(i)*N1(i,1,1) + WN1(i)+WPN1(i) )/H1(i,1)
       WCBb = VAdv(i,2) * (N1(i,1,1)+N1(i,2,1))
       BB(1) = BT(i)*H1(i,1)*(N1(i,1,1) + SoN1) + A1(i-1,1)-A1(i,1)
    *    + WCBb + A2(i,1)-A2(i-1,1) - Sink
       IF (iSalt .NE. 1)  THEN
         CC(i,1) = 0.0
         EE(i,1) = - VDif(i,2)
         DD(i,1) = BT(i)*H1(i,2) - EE(i,1)
       END IF
       xC(1) = CC(i,1)
       xD(1) = DD(i,1)
       xE(1) = EE(i,1)

       n = KB(i)
       IF (n .NE. 1)  THEN
         IF (n .NE. 2)  THEN
           DO 801 k=2,n-1
```

```
              HydN(i,k) = rKn12(i)*N1(i,k,1) / (rKh12+N1(i,k,1))
              SoN1 = CNdie1*Chl(i,k,1) - HydN(i,k) + ( BenN1(i,k)
     *          + rKn11(i)*(N1(i,k-1,1)-N1(i,k,1)) + WPN1(i) )/H(k)
              WCBt = WCBb
              WCBb = VAdv(i,k+1) * (N1(i,k,1)+N1(i,k+1,1))
              BB(k) = BHT(i,k)*(N1(i,k,1) + SoN1) + A1(i-1,k)-A1(i,k)
     *          + WCBb-WCBt + A2(i,k)-A2(i-1,k)
              IF (iSalt .NE. 1)  THEN
                CC(i,k) = EE(i,k-1)
                EE(i,k) = - VDif(i,k+1)
                DD(i,k) = BHT(i,k) - CC(i,k) - EE(i,k)
              END IF
              xC(k) = CC(i,k)
              xD(k) = DD(i,k)
              xE(k) = EE(i,k)
 801        CONTINUE
          END IF

          HydN(i,n) = rKn12(i)*N1(i,n,1) / (rKh12+N1(i,n,1))
          SoN1 = CNdie1*Chl(i,n,1) - HydN(i,n) + ( BenN1(i,n)
     *      + rKn11(i)*(N1(i,n-1,1)-N1(i,n,1)) + WPN1(i) )/H(n)
          WCBt = WCBb
          BB(n) = BHT(i,n)*(N1(i,n,1) + SoN1) + A1(i-1,n)-A1(i,n) - WCBt
     *      + A2(i,n)-A2(i-1,n)
          IF (iSalt .NE. 1)  THEN
            CC(i,n) = EE(i,n-1)
            EE(i,n) = 0.0
            DD(i,n) = BHT(i,n) - CC(i,n)
          END IF
          xC(n) = CC(i,n)
          xD(n) = DD(i,n)
          xE(n) = EE(i,n)
        END IF
        CALL SGTSL(n,xC,xD,xE,BB,infoN1)
        IF (infoN1 .NE. 0)  THEN
          PRINT*, i, infoN1, ' th element of the diagonal = 0.'
          STOP 'in SUBR EqnN1'
        END IF
        DO 802 k=1,n
          N1(i,k,2) = BB(k)
 802    CONTINUE
 800  CONTINUE
      RETURN
      END
C***************
      SUBROUTINE EqnN2
C*********************************************************************C
C Solve the MASS BALANCE Eqn. for N2 to get N2(i,k,2) for i=ML-MUs1.
C: Since rNit calculated here, this SUBR be called prior to EqnN3 & EqnDO.
C*********************************************************************C
      INCLUDE 'COM-WQ1.INC'
      Dimension A1(iX,iZ),A2(iX,iZ), BB(iZ),xC(iZ),xD(iZ),xE(iZ)
      Data A1,A2, BB,xC,xD,xE, Adj/2461*0.0/

      DO 10 i=MLs1,MUs1
        n = KB(i)
        DO 11 k=1,n
          IF (iHAdv .EQ. 2)  THEN
            A1(i,k) = (Gam(i+1,k)*N2(i,k,1) + Del(i+1,k)*N2(i+1,k,1))
     *              * HAdv(i,k)
          ELSE IF (iHAdv .EQ. 3)  THEN
            Adj = (N2(i,k,1)+N2(i+1,k,1))*0.5
     *          - COUR2(i,k)*(N2(i+1,k,1)-N2(i,k,1))
            IF (U(i+1,k,1) .GE. 0.0)  THEN
```

```
            A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVN2(i,k))
           ELSE
            A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVN2(i+1,k))
          END IF
        END IF
        A2(i,k) = HDif(i,k) * (N2(i+1,k,1) - N2(i,k,1))
 11   CONTINUE
 10  CONTINUE

     DO 800 i=ML,MUs1
       Sink = 0.0
       IF (SST(i) .NE. 0.0)  THEN
         IF (StH1(i,2) .GT. 0.0)  THEN
           StNit(i) = rKn23(i)*StN2(i) / (rKh23+StN2(i))
     *               * StDO(i) / (rKnit+StDO(i))
           Bio = (CNdie2 - an*StPR2(i)*StG(i))*StChl(i) - StNit(i)
     *       + HydNSt(i) + StBN2(i)/StH1(i,1)
           IF (DelA(i) .GT. 0.0)  THEN
             Sink = N2(i,1,1) * SDoDT(i)
             Phys = (StN2(i)*StH1(i,1) + N2(i,1,1)*DelA(i))/StH1(i,2)
             StN2(i) = Phys + Bio
            ELSE
             Sink = StN2(i) * SDoDT(i)
             StN2(i) = StN2(i) + Bio
           END IF
          ELSE
           StN2(i) = 0.0
         END IF
       END IF
       rNit(i,1) = rKn23(i)*N2(i,1,1) / (rKh23+N2(i,1,1))
     *           * DOC(i,1,1) / (rKnit+DOC(i,1,1))
       SoN2 = (CNdie2 - an*PR2(i,1)*G(i,1))*Chl(i,1,1) - rNit(i,1)
     *   + HydN(i,1) + (BenN2(i,1) + WN2(i)+WPN2(i))/H1(i,1)
       WCBb = VAdv(i,2) * (N2(i,1,1)+N2(i,2,1))
       BB(1) = BT(i)*H1(i,1)*(N2(i,1,1) + SoN2) + A1(i-1,1)-A1(i,1)
     *   + WCBb + A2(i,1)-A2(i-1,1) - Sink
       xC(1) = CC(i,1)
       xD(1) = DD(i,1)
       xE(1) = EE(i,1)

       n = KB(i)
       IF (n .NE. 1)  THEN
         IF (n .NE. 2)  THEN
           DO 801 k=2,n-1
             rNit(i,k) = rKn23(i)*N2(i,k,1) / (rKh23+N2(i,k,1))
     *                 * DOC(i,k,1) / (rKnit+DOC(i,k,1))
             SoN2= (CNdie2 - an*PR2(i,k)*G(i,k))*Chl(i,k,1) - rNit(i,k)
     *         + HydN(i,k) + ( BenN2(i,k) + WPN2(i) )/H(k)
             WCBt = WCBb
             WCBb = VAdv(i,k+1) * (N2(i,k,1)+N2(i,k+1,1))
             BB(k) = BHT(i,k)*(N2(i,k,1) + SoN2) + A1(i-1,k)-A1(i,k)
     *         + WCBb-WCBt + A2(i,k)-A2(i-1,k)
             xC(k) = CC(i,k)
             xD(k) = DD(i,k)
             xE(k) = EE(i,k)
801        CONTINUE
         END IF

         rNit(i,n) = rKn23(i)*N2(i,n,1) / (rKh23+N2(i,n,1))
     *             * DOC(i,n,1) / (rKnit+DOC(i,n,1))
         SoN2 = (CNdie2 - an*PR2(i,n)*G(i,n))*Chl(i,n,1) - rNit(i,n)
     *     + HydN(i,n) + (BenN2(i,n) + WPN2(i))/H(n)
         WCBt = WCBb
         BB(n) = BHT(i,n)*(N2(i,n,1) + SoN2) + A1(i-1,n)-A1(i,n) - WCBt
```

```
     *       + A2(i,n)-A2(i-1,n)
             xC(n) = CC(i,n)
             xD(n) = DD(i,n)
             xE(n) = EE(i,n)
           END IF
           CALL SGTSL(n,xC,xD,xE,BB,infoN2)
           IF (infoN2 .NE. 0)  THEN
             PRINT*, i, infoN2, ' th element of the diagonal = 0.'
             STOP 'in SUBR EqnN2'
           END IF
           DO 802 k=1,n
             N2(i,k,2) = BB(k)
  802    CONTINUE
  800  CONTINUE
       RETURN
       END
C***************
       SUBROUTINE EqnN3
C*************************************************************************C
C Solve the MASS BALANCE Eqn. for N3 to get N3(i,k,2) for i=ML-MUs1.
C*************************************************************************C
       INCLUDE 'COM-WQ1.INC'
       Dimension A1(iX,iZ),A2(iX,iZ), BB(iZ),xC(iZ),xD(iZ),xE(iZ)
       Data A1,A2, BB,xC,xD,xE, Adj/2461*0.0/

       DO 10 i=MLs1,MUs1
         n = KB(i)
         DO 11 k=1,n
           IF (iHAdv .EQ. 2)  THEN
             A1(i,k) = (Gam(i+1,k)*N3(i,k,1) + Del(i+1,k)*N3(i+1,k,1))
     *               * HAdv(i,k)
            ELSE IF (iHAdv .EQ. 3)  THEN
             Adj = (N3(i,k,1)+N3(i+1,k,1))*0.5
     *           - COUR2(i,k)*(N3(i+1,k,1)-N3(i,k,1))
             IF (U(i+1,k,1) .GE. 0.0)  THEN
               A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVN3(i,k))
              ELSE
               A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVN3(i+1,k))
             END IF
           END IF
           A2(i,k) = HDif(i,k) * (N3(i+1,k,1) - N3(i,k,1))
   11    CONTINUE
  10   CONTINUE

       DO 800 i=ML,MUs1
         Sink = 0.0
         IF (SST(i) .NE. 0.0)  THEN
           IF (StH1(i,2) .GT. 0.0)  THEN
             Bio = StNit(i) - an*StPR3(i)*StG(i)*StChl(i) + StBN3(i)/
     *         StH1(i,1) - rKn33(i)*rKden/(rKden+StDO(i))*StN3(i)
             IF (DelA(i) .GT. 0.0)  THEN
               Sink = N3(i,1,1) * SDoDT(i)
               Phys = (StN3(i)*StH1(i,1) + N3(i,1,1)*DelA(i))/StH1(i,2)
               StN3(i) = Phys + Bio
              ELSE
               Sink = StN3(i) * SDoDT(i)
               StN3(i) = StN3(i) + Bio
             END IF
            ELSE
             StN3(i) = 0.0
           END IF
         END IF
         SoN3 = rNit(i,1) - an*PR3(i,1)*G(i,1)*Chl(i,1,1)
     *     - rKn33(i)*rKden/(rKden+DOC(i,1,1))*N3(i,1,1)
```

```
     *     + ( BenN3(i,1) + WN3(i)+WPN3(i) )/H1(i,1)
           WCBb = VAdv(i,2) * (N3(i,1,1)+N3(i,2,1))
           BB(1) = BT(i)*H1(i,1)*(N3(i,1,1) + SoN3) + A1(i-1,1)-A1(i,1)
     *     + WCBb + A2(i,1)-A2(i-1,1) - Sink
           xC(1) = CC(i,1)
           xD(1) = DD(i,1)
           xE(1) = EE(i,1)

           n = KB(i)
           IF (n .NE. 1)  THEN
             IF (n .NE. 2)  THEN
               DO 801 k=2,n-1
                 SoN3 = rNit(i,k) - an*PR3(i,k)*G(i,k)*Chl(i,k,1)
     *             - rKn33(i)*rKden/(rKden+DOC(i,k,1))*N3(i,k,1)
     *             + (BenN3(i,k) + WPN3(i))/H(k)
                 WCBt = WCBb
                 WCBb = VAdv(i,k+1) * (N3(i,k,1)+N3(i,k+1,1))
                 BB(k) = BHT(i,k)*(N3(i,k,1) + SoN3) + A1(i-1,k)-A1(i,k)
     *             + WCBb-WCBt + A2(i,k)-A2(i-1,k)
                 xC(k) = CC(i,k)
                 xD(k) = DD(i,k)
                 xE(k) = EE(i,k)
  801          CONTINUE
             END IF

             SoN3 = rNit(i,n) - an*G(i,n)*PR3(i,n)*Chl(i,n,1) + (BenN3(i,n)
     *        + WPN3(i))/H(n) - rKn33(i)*rKden/(rKden+DOC(i,n,1))*N3(i,n,1)
             WCBt = WCBb
             BB(n) = BHT(i,n)*(N3(i,n,1) + SoN3) + A1(i-1,n)-A1(i,n) - WCBt
     *         + A2(i,n)-A2(i-1,n)
             xC(n) = CC(i,n)
             xD(n) = DD(i,n)
             xE(n) = EE(i,n)
           END IF
           CALL SGTSL(n,xC,xD,xE,BB,infoN3)
           IF (infoN3 .NE. 0)  THEN
             PRINT*, i, infoN3, ' th element of the diagonal = 0.'
             STOP 'in SUBR EqnN3'
           END IF
           DO 802 k=1,n
             N3(i,k,2) = BB(k)
   802     CONTINUE
   800   CONTINUE
         RETURN
         END
C***************
         SUBROUTINE EqnP1
C*********************************************************************C
C Solve the MASS BALANCE Eqn. for P1 to get P1(i,k,2) for i=ML-MUs1.
C: Since HydP is calculated here, this SUBR be called prior to EqnP2.
C*********************************************************************C
         INCLUDE 'COM-WQ1.INC'
         Dimension A1(iX,iZ),A2(iX,iZ), BB(iZ),xC(iZ),xD(iZ),xE(iZ)
         Data A1,A2, BB,xC,xD,xE, Adj/2461*0.0/

         DO 10 i=MLs1,MUs1
           n = KB(i)
           DO 11 k=1,n
             IF (iHAdv .EQ. 2)  THEN
               A1(i,k) = (Gam(i+1,k)*P1(i,k,1) + Del(i+1,k)*P1(i+1,k,1))
     *                 * HAdv(i,k)
              ELSE IF (iHAdv .EQ. 3)  THEN
               Adj = (P1(i,k,1)+P1(i+1,k,1))*0.5
     *             - COUR2(i,k)*(P1(i+1,k,1)-P1(i,k,1))
```

```
          IF (U(i+1,k,1) .GE. 0.0)  THEN
            A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVP1(i,k))
           ELSE
            A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVP1(i+1,k))
          END IF
        END IF
        A2(i,k) = HDif(i,k) * (P1(i+1,k,1) - P1(i,k,1))
 11   CONTINUE
 10  CONTINUE

     DO 800 i=ML,MUs1
       Sink = 0.0
       IF (SST(i) .NE. 0.0)  THEN
         IF (StH1(i,2) .GT. 0.0)  THEN
           HydPSt(i) = rKp12(i)*StP1(i) / (rKhp12+StP1(i))
           Bio = CPdie1*StChl(i) - HydPSt(i)
     *       + (StBP1(i) - rKp11(i)*StP1(i))/StH1(i,1)
           IF (DelA(i) .GT. 0.0)  THEN
             Sink = P1(i,1,1) * SDoDT(i)
             Phys = (StP1(i)*StH1(i,1) + P1(i,1,1)*DelA(i))/StH1(i,2)
             StP1(i) = Phys + Bio
            ELSE
             Sink = StP1(i) * SDoDT(i)
             StP1(i) = StP1(i) + Bio
           END IF
          ELSE
           StP1(i) = 0.0
         END IF
       END IF
       HydP(i,1) = rKp12(i)*P1(i,1,1) / (rKhp12+P1(i,1,1))
       SoP1 = CPdie1*Chl(i,1,1) - HydP(i,1) + ( BenP1(i,1)
     *   - rKp11(i)*P1(i,1,1) + WP1(i)+WPP1(i) )/H1(i,1)
       WCBb = VAdv(i,2) * (P1(i,1,1)+P1(i,2,1))
       BB(1) = BT(i)*H1(i,1)*(P1(i,1,1) + SoP1) + A1(i-1,1)-A1(i,1)
     *   + WCBb + A2(i,1)-A2(i-1,1) - Sink
       xC(1) = CC(i,1)
       xD(1) = DD(i,1)
       xE(1) = EE(i,1)

       n = KB(i)
       IF (n .NE. 1)  THEN
         IF (n .NE. 2)  THEN
           DO 801 k=2,n-1
             HydP(i,k) = rKp12(i)*P1(i,k,1) / (rKhp12+P1(i,k,1))
             SoP1 = CPdie1*Chl(i,k,1) - HydP(i,k) + ( BenP1(i,k)
     *         + rKp11(i)*(P1(i,k-1,1)-P1(i,k,1)) + WPP1(i) )/H(k)
             WCBt = WCBb
             WCBb = VAdv(i,k+1) * (P1(i,k,1)+P1(i,k+1,1))
             BB(k) = BHT(i,k)*(P1(i,k,1) + SoP1) + A1(i-1,k)-A1(i,k)
     *         + WCBb-WCBt + A2(i,k)-A2(i-1,k)
             xC(k) = CC(i,k)
             xD(k) = DD(i,k)
             xE(k) = EE(i,k)
801        CONTINUE
         END IF

         HydP(i,n) = rKp12(i)*P1(i,n,1) / (rKhp12+P1(i,n,1))
         SoP1 = CPdie1*Chl(i,n,1) - HydP(i,n) + ( BenP1(i,n)
     *     + rKp11(i)*(P1(i,n-1,1)-P1(i,n,1)) + WPP1(i) )/H(n)
         WCBt = WCBb
         BB(n) = BHT(i,n)*(P1(i,n,1) + SoP1) + A1(i-1,n)-A1(i,n) - WCBt
     *     + A2(i,n)-A2(i-1,n)
         xC(n) = CC(i,n)
         xD(n) = DD(i,n)
```

```
          xE(n) = EE(i,n)
        END IF
        CALL SGTSL(n,xC,xD,xE,BB,infoP1)
        IF (infoP1 .NE. 0)  THEN
          PRINT*, i, infoP1, ' th element of the diagonal = 0.'
          STOP 'in SUBR EqnP1'
        END IF
        DO 802 k=1,n
          P1(i,k,2) = BB(k)
  802   CONTINUE
  800 CONTINUE
      RETURN
      END
C***************
      SUBROUTINE EqnP2
C*********************************************************************C
C Solve the MASS BALANCE Eqn. for P2 to get P2(i,k,2) for i=ML1-MUs1.
C*********************************************************************C
      INCLUDE 'COM-WQ1.INC'
      Dimension A1(iX,iZ),A2(iX,iZ), BB(iZ),xC(iZ),xD(iZ),xE(iZ)
      Data A1,A2, BB,xC,xD,xE, Adj/2461*0.0/

      DO 10 i=MLs1,MUs1
        n = KB(i)
        DO 11 k=1,n
          IF (iHAdv .EQ. 2)  THEN
            A1(i,k) = (Gam(i+1,k)*P2(i,k,1) + Del(i+1,k)*P2(i+1,k,1))
     *                * HAdv(i,k)
          ELSE IF (iHAdv .EQ. 3)  THEN
            Adj = (P2(i,k,1)+P2(i+1,k,1))*0.5
     *          - COUR2(i,k)*(P2(i+1,k,1)-P2(i,k,1))
            IF (U(i+1,k,1) .GE. 0.0)  THEN
              A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVP2(i,k))
            ELSE
              A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVP2(i+1,k))
            END IF
          END IF
          A2(i,k) = HDif(i,k) * (P2(i+1,k,1) - P2(i,k,1))
   11   CONTINUE
   10 CONTINUE

      DO 800 i=ML,MUs1
        Sink = 0.0
        IF (SST(i) .NE. 0.0)  THEN
          IF (StH1(i,2) .GT. 0.0)  THEN
            Bio = (CPdie2 - ap*StG(i))*StChl(i) + HydPSt(i)
     *        + (StBP2(i) - rKp22(i)*StP2(i))/StH1(i,1)
            IF (DelA(i) .GT. 0.0)  THEN
              Sink = P2(i,1,1) * SDoDT(i)
              Phys = (StP2(i)*StH1(i,1) + P2(i,1,1)*DelA(i))/StH1(i,2)
              StP2(i) = Phys + Bio
            ELSE
              Sink = StP2(i) * SDoDT(i)
              StP2(i) = StP2(i) + Bio
            END IF
          ELSE
            StP2(i) = 0.0
          END IF
        END IF
        SoP2 = (CPdie2 - ap*G(i,1))*Chl(i,1,1) + HydP(i,1) + (BenP2(i,1)
     *    - rKp22(i)*P2(i,1,1) + WP2(i)+WPP2(i))/H1(i,1)
        WCBb = VAdv(i,2) * (P2(i,1,1)+P2(i,2,1))
        BB(1) = BT(i)*H1(i,1)*(P2(i,1,1) + SoP2) + A1(i-1,1)-A1(i,1)
     *    + WCBb + A2(i,1)-A2(i-1,1) - Sink
```

```
        xC(1) = CC(i,1)
        xD(1) = DD(i,1)
        xE(1) = EE(i,1)

        n = KB(i)
        IF (n .NE. 1)  THEN
          IF (n .NE. 2)  THEN
            DO 801 k=2,n-1
              SoP2 = (CPdie2 - ap*G(i,k))*Chl(i,k,1) + HydP(i,k)
     *          + ( BenP2(i,k) + rKp22(i)*(P2(i,k-1,1)-P2(i,k,1))
     *             + WPP2(i) )/H(k)
              WCBt = WCBb
              WCBb = VAdv(i,k+1) * (P2(i,k,1)+P2(i,k+1,1))
              BB(k) = BHT(i,k)*(P2(i,k,1) + SoP2) + A1(i-1,k)-A1(i,k)
     *          + WCBb-WCBt + A2(i,k)-A2(i-1,k)
              xC(k) = CC(i,k)
              xD(k) = DD(i,k)
              xE(k) = EE(i,k)
 801        CONTINUE
          END IF

          SoP2=(CPdie2 - ap*G(i,n))*Chl(i,n,1) + HydP(i,n) + (BenP2(i,n)
     *      + rKp22(i)*(P2(i,n-1,1)-P2(i,n,1)) + WPP2(i))/H(n)
          WCBt = WCBb
          BB(n) = BHT(i,n)*(P2(i,n,1) + SoP2) + A1(i-1,n)-A1(i,n) - WCBt
     *      + A2(i,n)-A2(i-1,n)
          xC(n) = CC(i,n)
          xD(n) = DD(i,n)
          xE(n) = EE(i,n)
        END IF
        CALL SGTSL(n,xC,xD,xE,BB,infoP2)
        IF (infoP2 .NE. 0)  THEN
          PRINT*, i, infoP2, ' th element of the diagonal = 0.'
          STOP 'in SUBR EqnP2'
        END IF
        DO 802 k=1,n
          P2(i,k,2) = BB(k)
 802    CONTINUE
 800  CONTINUE
      RETURN
      END
C***************
      SUBROUTINE EqnChl
C*********************************************************************C
C Solve the MASS BALANCE Eqn. for Chl to get Chl(i,k,2) for i=ML1-MUs1.
C: Note that among WQC, only Chl is in ug/L & SoChl is in ug cm/L/day.
C*********************************************************************C
      INCLUDE 'COM-WQ1.INC'
      Dimension A1(iX,iZ),A2(iX,iZ), BB(iZ),xC(iZ),xD(iZ),xE(iZ)
      Data A1,A2, BB,xC,xD,xE, Adj/2461*0.0/

      DO 10 i=MLs1,MUs1
        n = KB(i)
        DO 11 k=1,n
          IF (iHAdv .EQ. 2)  THEN
            A1(i,k) = (Gam(i+1,k)*Chl(i,k,1) + Del(i+1,k)*Chl(i+1,k,1))
     *                * HAdv(i,k)
          ELSE IF (iHAdv .EQ. 3)  THEN
            Adj = (Chl(i,k,1)+Chl(i+1,k,1))*0.5
     *          - COUR2(i,k)*(Chl(i+1,k,1)-Chl(i,k,1))
            IF (U(i+1,k,1) .GE. 0.0)  THEN
              A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVChl(i,k))
            ELSE
              A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVChl(i+1,k))
```

```
          END IF
        END IF
        A2(i,k) = HDif(i,k) * (Chl(i+1,k,1) - Chl(i,k,1))
 11   CONTINUE
 10  CONTINUE

     DO 800 i=ML,MUs1
       Sink = 0.0
       IF (SST(i) .NE. 0.0)  THEN
         IF (StH1(i,2) .GT. 0.0)  THEN
           Bio = (StG(i) - RespR - GrazR - rKchl(i)/StH1(i,1))*StChl(i)
           IF (DelA(i) .GT. 0.0)  THEN
             Sink = Chl(i,1,1) * SDoDT(i)
             Phys = (StChl(i)*StH1(i,1) + Chl(i,1,1)*DelA(i))/StH1(i,2)
             StChl(i) = Phys + Bio
            ELSE
             Sink = StChl(i) * SDoDT(i)
             StChl(i) = StChl(i) + Bio
           END IF
          ELSE
           StChl(i) = 0.0
         END IF
       END IF
       SoChl = (G(i,1) - RespR - GrazR)*Chl(i,1,1)
     *   + ( WChl(i) - rKchl(i)*Chl(i,1,1) )/H1(i,1)
       WCBb = VAdv(i,2) * (Chl(i,1,1)+Chl(i,2,1))
       BB(1) = BT(i)*H1(i,1)*(Chl(i,1,1) + SoChl) + A1(i-1,1)-A1(i,1)
     *   + WCBb + A2(i,1)-A2(i-1,1) - Sink
       xC(1) = CC(i,1)
       xD(1) = DD(i,1)
       xE(1) = EE(i,1)

       n = KB(i)
       IF (n .NE. 1)  THEN
         IF (n .NE. 2)  THEN
           DO 801 k=2,n-1
             SoChl = (G(i,k) - RespR - GrazR)*Chl(i,k,1)
     *         + rKchl(i)*(Chl(i,k-1,1)-Chl(i,k,1))/H(k)
             WCBt = WCBb
             WCBb = VAdv(i,k+1) * (Chl(i,k,1)+Chl(i,k+1,1))
             BB(k) = BHT(i,k)*(Chl(i,k,1) + SoChl) + A1(i-1,k)-A1(i,k)
     *         + WCBb-WCBt + A2(i,k)-A2(i-1,k)
             xC(k) = CC(i,k)
             xD(k) = DD(i,k)
             xE(k) = EE(i,k)
801        CONTINUE
         END IF

         SoChl = (G(i,n) - RespR - GrazR)*Chl(i,n,1)
     *     + rKchl(i)*(Chl(i,n-1,1)-Chl(i,n,1))/H(n)
         WCBt = WCBb
         BB(n) = BHT(i,n)*(Chl(i,n,1) + SoChl) + A1(i-1,n)-A1(i,n)
     *     - WCBt + A2(i,n)-A2(i-1,n)
         xC(n) = CC(i,n)
         xD(n) = DD(i,n)
         xE(n) = EE(i,n)
       END IF
       CALL SGTSL(n,xC,xD,xE,BB,infoChl)
       IF (infoChl .NE. 0)  THEN
         PRINT*, i, infoChl, ' th element of the diagonal = 0.'
         STOP 'in SUBR EqnChl'
       END IF
       DO 802 k=1,n
         Chl(i,k,2) = BB(k)
```

```
  802   CONTINUE
 800  CONTINUE
      RETURN
      END
C***************
      SUBROUTINE EqnBOD
C*************************************************************************C
C Solve the MASS BALANCE Eqn. for CBOD to get CBOD(i,k,2) for i=ML1-MUs1.
C*************************************************************************C
      INCLUDE 'COM-WQ1.INC'
      Dimension A1(iX,iZ),A2(iX,iZ), BB(iZ),xC(iZ),xD(iZ),xE(iZ)
      Data A1,A2, BB,xC,xD,xE, Adj/2461*0.0/

      DO 10 i=MLs1,MUs1
        n = KB(i)
        DO 11 k=1,n
          IF (iHAdv .EQ. 2)  THEN
            A1(i,k) = (Gam(i+1,k)*CBOD(i,k,1)+ Del(i+1,k)*CBOD(i+1,k,1))
     *              * HAdv(i,k)
           ELSE IF (iHAdv .EQ. 3)  THEN
            Adj = (CBOD(i,k,1)+CBOD(i+1,k,1))*0.5
     *          - COUR2(i,k)*(CBOD(i+1,k,1)-CBOD(i,k,1))
            IF (U(i+1,k,1) .GE. 0.0)  THEN
              A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVBOD(i,k))
             ELSE
              A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVBOD(i+1,k))
            END IF
          END IF
          A2(i,k) = HDif(i,k) * (CBOD(i+1,k,1) - CBOD(i,k,1))
   11   CONTINUE
  10  CONTINUE

      DO 800 i=ML,MUs1
        Sink = 0.0
        IF (SST(i) .NE. 0.0)  THEN
          IF (StH1(i,2) .GT. 0.0)  THEN
            Bio = BODdie*StChl(i) - rKc(i)*StBOD(i) + ( StBBOD(i) -
     *        StBDO(i)*rKDO/(rKDO+StDO(i)) - rKbod(i)*StBOD(i) )
     *        /StH1(i,1)
            IF (DelA(i) .GT. 0.0)  THEN
              Sink = CBOD(i,1,1) * SDoDT(i)
              Phys =(StBOD(i)*StH1(i,1) + CBOD(i,1,1)*DelA(i))/StH1(i,2)
              StBOD(i) = Phys + Bio
             ELSE
              Sink = StBOD(i) * SDoDT(i)
              StBOD(i) = StBOD(i) + Bio
            END IF
           ELSE
            StBOD(i) = 0.0
          END IF
        END IF
        So = BODdie*Chl(i,1,1) - rKc(i)*CBOD(i,1,1) + ( BenBOD(i,1)
     *    - BenDO(i,1)*rKDO/(rKDO+DOC(i,1,1)) - rKbod(i)*CBOD(i,1,1)
     *    + WBOD(i)+WPBOD(i) )/H1(i,1)
        WCBb = VAdv(i,2) * (CBOD(i,1,1)+CBOD(i,2,1))
        BB(1) = BT(i)*H1(i,1)*(CBOD(i,1,1) + So) + A1(i-1,1)-A1(i,1)
     *    + WCBb + A2(i,1)-A2(i-1,1) - Sink
        xC(1) = CC(i,1)
        xD(1) = DD(i,1)
        xE(1) = EE(i,1)

        n = KB(i)
        IF (n .NE. 1)  THEN
          IF (n .NE. 2)  THEN
```

```
            DO 801 k=2,n-1
              So = BODdie*Chl(i,k,1) - rKc(i)*CBOD(i,k,1) +( BenBOD(i,k)
     *          - BenDO(i,k)*rKDO/(rKDO+DOC(i,k,1)) + WPBOD(i)
     *          + rKbod(i)*(CBOD(i,k-1,1)-CBOD(i,k,1)) )/H(k)
              WCBt = WCBb
              WCBb = VAdv(i,k+1) * (CBOD(i,k,1)+CBOD(i,k+1,1))
              BB(k) = BHT(i,k)*(CBOD(i,k,1) + So) + A1(i-1,k)-A1(i,k)
     *          + WCBb-WCBt + A2(i,k)-A2(i-1,k)
              xC(k) = CC(i,k)
              xD(k) = DD(i,k)
              xE(k) = EE(i,k)
 801        CONTINUE
          END IF

          So = BODdie*Chl(i,n,1) - rKc(i)*CBOD(i,n,1) + ( BenBOD(i,n)
     *      - BenDO(i,n)*rKDO/(rKDO+DOC(i,n,1)) + WPBOD(i)
     *      + rKbod(i)*(CBOD(i,n-1,1)-CBOD(i,n,1)) )/H(n)
          WCBt = WCBb
          BB(n) = BHT(i,n)*(CBOD(i,n,1) + So) + A1(i-1,n)-A1(i,n) - WCBt
     *      + A2(i,n)-A2(i-1,n)
          xC(n) = CC(i,n)
          xD(n) = DD(i,n)
          xE(n) = EE(i,n)
        END IF
        CALL SGTSL(n,xC,xD,xE,BB,infoBOD)
        IF (infoBOD .NE. 0)  THEN
          PRINT*, i, infoBOD, ' th element of the diagonal = 0.'
          STOP 'in SUBR EqnBOD'
        END IF
        DO 802 k=1,n
          CBOD(i,k,2) = BB(k)
 802    CONTINUE
 800  CONTINUE
      RETURN
      END
C***************
      SUBROUTINE EqnDO
C*********************************************************************C
C Solve the MASS BALANCE Eqn. for DO to get DOC(i,k,2) for i=ML1-MUs1.
C: DO reaeration occurs only at the top layer (for the remaining layers,
C  vertical diffusion will account for the replenishment of DO).
C: Assume St is always DO saturated.
C*********************************************************************C
      INCLUDE 'COM-WQ1.INC'
      Parameter  (ano=4.57)
      Dimension A1(iX,iZ),A2(iX,iZ), BB(iZ),xC(iZ),xD(iZ),xE(iZ)
      Data A1,A2, BB,xC,xD,xE, Adj/2461*0.0/

      DO 10 i=MLs1,MUs1
        n = KB(i)
        DO 11 k=1,n
          IF (iHAdv .EQ. 2)  THEN
            A1(i,k) = (Gam(i+1,k)*DOC(i,k,1) + Del(i+1,k)*DOC(i+1,k,1))
     *              * HAdv(i,k)
          ELSE IF (iHAdv .EQ. 3)  THEN
            Adj = (DOC(i,k,1)+DOC(i+1,k,1))*0.5
     *          - COUR2(i,k)*(DOC(i+1,k,1)-DOC(i,k,1))
            IF (U(i+1,k,1) .GE. 0.0)  THEN
              A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVDO(i,k))
            ELSE
              A1(i,k) = HAdv(i,k)*(Adj - CK(i,k)*CURVDO(i+1,k))
            END IF
          END IF
          A2(i,k) = HDif(i,k) * (DOC(i+1,k,1) - DOC(i,k,1))
```

```
 11   CONTINUE
 10  CONTINUE

       XAL(ML,1) = BU(ML,1) * H1(ML,1)
       TotXA = XAL(ML,1) + XAm(ML)
       UXA1 = XAL(ML,1) * ABS(U(ML,1,1))
       n = KBU(ML)
       DO 799 k=2,n
         UXA1 = UXA1 + XAL(ML,k)*ABS(U(ML,k,1))
799    CONTINUE
       VS1 = UXA1 / (TotXA*TotXA) * BU(ML,1)
     DO 800 i=ML,MUs1
       Sink = 0.0
       IF (SST(i) .NE. 0.0)  THEN
         IF (StH1(i,2) .GT. 0.0)  THEN
           StDO(i) = TDOs + (STDOs + 2.739E-4*StS(i))*StS(i)
           IF (DelA(i) .GT. 0.0)  THEN
             Sink = DOC(i,1,1) * SDoDT(i)
            ELSE
             Sink = StDO(i) * SDoDT(i)
           END IF
          ELSE
           StDO(i) = 0.0
         END IF
       END IF
       XAL(i+1,1) = BU2H(i+1,1) * 0.5
       TotXA = XAL(i+1,1) + XAm(i+1)
       UXA2 = XAL(i+1,1) * ABS(U(i+1,1,1))
       n = KBU(i+1)
       DO 811 k=2,n
         UXA2 = UXA2 + XAL(i+1,k)*ABS(U(i+1,k,1))
811    CONTINUE
       VS2 = UXA2 / (TotXA*TotXA) * BU(i+1,1)
       rKr = TDOaer * (rKro*SQRT(VS1+VS2) + WRea(i)) / H1(i,1)
         VS1 = VS2
       Dev = TDOs + (STDOs + 2.739E-4*S(i,1,1))*S(i,1,1) - DOC(i,1,1)
       SoDO = (PQ*G(i,1) - DOresp)*Chl(i,1,1) - rKc(i)*CBOD(i,1,1) -
     *   ano*rNit(i,1) + rKr*Dev + ( WDO(i) + WPDO(i)
     *     + BenDO(i,1)*DOC(i,1,1)/(rKDO+DOC(i,1,1)) )/H1(i,1)
       WCBb = VAdv(i,2) * (DOC(i,1,1)+DOC(i,2,1))
       BB(1) = BT(i)*H1(i,1)*(DOC(i,1,1) + SoDO) + A1(i-1,1)-A1(i,1)
     *   + WCBb + A2(i,1)-A2(i-1,1) - Sink
       xC(1) = CC(i,1)
       xD(1) = DD(i,1)
       xE(1) = EE(i,1)

       n = KB(i)
       IF (n .NE. 1)  THEN
         IF (n .NE. 2)  THEN
           DO 801 k=2,n-1
             SoDO = (PQ*G(i,k) - DOresp)*Chl(i,k,1) - ano*rNit(i,k)
     *         - rKc(i)*CBOD(i,k,1) + ( WPDO(i)
     *           + BenDO(i,k)*DOC(i,k,1)/(rKDO+DOC(i,k,1)) )/H(k)
             WCBt = WCBb
             WCBb = VAdv(i,k+1) * (DOC(i,k,1)+DOC(i,k+1,1))
             BB(k) = BHT(i,k)*(DOC(i,k,1) + SoDO) + A1(i-1,k)-A1(i,k)
     *         + WCBb-WCBt + A2(i,k)-A2(i-1,k)
             xC(k) = CC(i,k)
             xD(k) = DD(i,k)
             xE(k) = EE(i,k)
801        CONTINUE
         END IF

         SoDO = (PQ*G(i,n) - DOresp)*Chl(i,n,1) - rKc(i)*CBOD(i,n,1)
```

```
     *       - ano*rNit(i,n) + ( WPDO(i)
     *         + BenDO(i,n)*DOC(i,n,1)/(rKDO+DOC(i,n,1)) )/H(n)
             WCBt = WCBb
             BB(n) = BHT(i,n)*(DOC(i,n,1) + SoDO) + A1(i-1,n)-A1(i,n)
     *         - WCBt + A2(i,n)-A2(i-1,n)
             xC(n) = CC(i,n)
             xD(n) = DD(i,n)
             xE(n) = EE(i,n)
           END IF
           CALL SGTSL(n,xC,xD,xE,BB,infoDO)
           IF (infoDO .NE. 0)  THEN
             PRINT*, i, infoDO, ' th element of the diagonal = 0.'
             STOP 'in SUBR EqnDO'
           END IF
           DO 802 k=1,n
             IF (BB(k) .GT. 0.0)  THEN
               DOC(i,k,2) = BB(k)
              ELSE
               DOC(i,k,2) = 0.0
             END IF
  802      CONTINUE
  800    CONTINUE
         RETURN
         END
C***************
         SUBROUTINE WDnBdry
C*********************************************************************C
C Calculate C(i,k,2) at the downstream bdry (i=MU).
C1 During ebb (when U >= 0.0), C(MU,k,2) is estimated using C(MUs1,k,1),
C   C(MU,k,1), U(MU,k,1), DTs & DX.
C2 During flood,
C : From the beginning of flood to MST, C(MU,k,2) is estimted using
C    DnC(k), C(MU,k,1) & MST.
C : After MST, C(MU,k,2) = DnC(k).
C*********************************************************************C
         INCLUDE 'COM-WQ1.INC'
         Dimension DBN1(iZ),DBN2(iZ),DBN3(iZ),DBP1(iZ),DBP2(iZ),DBChl(iZ),
        *          DBBOD(iZ),DBDO(iZ),NWQc(iZ)
         Save/WLocal/,DBN1,DBN2,DBN3,DBP1,DBP2,DBChl,DBBOD,DBDO,NWQc
         Data NWQc/15*10000/

         n = KB(MU)
         DO 900 k=1,n
           IF (U(MU,k,1) .GE. 0.0)  THEN
             NWQc(k) = 0
             Change = dtx * U(MU,k,1)
             Tmp = 1.0 - Change
             N1(MU,k,2) = N1(MU,k,1)*Tmp + N1(MUs1,k,1)*Change
             N2(MU,k,2) = N2(MU,k,1)*Tmp + N2(MUs1,k,1)*Change
             N3(MU,k,2) = N3(MU,k,1)*Tmp + N3(MUs1,k,1)*Change
             P1(MU,k,2) = P1(MU,k,1)*Tmp + P1(MUs1,k,1)*Change
             P2(MU,k,2) = P2(MU,k,1)*Tmp + P2(MUs1,k,1)*Change
             Chl(MU,k,2) = Chl(MU,k,1)*Tmp + Chl(MUs1,k,1)*Change
             CBOD(MU,k,2) = CBOD(MU,k,1)*Tmp + CBOD(MUs1,k,1)*Change
             DOC(MU,k,2) = DOC(MU,k,1)*Tmp + DOC(MUs1,k,1)*Change
            ELSE
             IF (NWQc(k) .GE. MST)  THEN
               N1(MU,k,2) = DnN1(k)
               N2(MU,k,2) = DnN2(k)
               N3(MU,k,2) = DnN3(k)
               P1(MU,k,2) = DnP1(k)
               P2(MU,k,2) = DnP2(k)
               Chl(MU,k,2) = DnChl(k)
               CBOD(MU,k,2) = DnBOD(k)
```

```
            DOC(MU,k,2) = DnDO(k)
           ELSE
            IF (NWQc(k) .EQ. 0)  THEN
              DBN1(k) = (DnN1(k) - N1(MU,k,1)) / MST
              DBN2(k) = (DnN2(k) - N2(MU,k,1)) / MST
              DBN3(k) = (DnN3(k) - N3(MU,k,1)) / MST
              DBP1(k) = (DnP1(k) - P1(MU,k,1)) / MST
              DBP2(k) = (DnP2(k) - P2(MU,k,1)) / MST
              DBChl(k) = (DnChl(k) - Chl(MU,k,1)) / MST
              DBBOD(k) = (DnBOD(k) - CBOD(MU,k,1)) / MST
              DBDO(k)  = (DnDO(k) - DOC(MU,k,1)) / MST
            END IF
            NWQc(k) = NWQc(k) + 1
            N1(MU,k,2) = N1(MU,k,1) + DBN1(k)
            N2(MU,k,2) = N2(MU,k,1) + DBN2(k)
            N3(MU,k,2) = N3(MU,k,1) + DBN3(k)
            P1(MU,k,2) = P1(MU,k,1) + DBP1(k)
            P2(MU,k,2) = P2(MU,k,1) + DBP2(k)
            Chl(MU,k,2) = Chl(MU,k,1) + DBChl(k)
            CBOD(MU,k,2) = CBOD(MU,k,1) + DBBOD(k)
            DOC(MU,k,2) = DOC(MU,k,1) + DBDO(k)
           END IF
         END IF
900   CONTINUE
      RETURN
      END
```

B-5. SGTSL.FOR

```
      subroutine sgtsl(n,c,d,e,b,info)
      integer n,info
      real c(1),d(1),e(1),b(1)
c
c     sgtsl given a general tridiagonal matrix and a right hand
c     side will find the solution.
c
c     on entry
c
c        n        integer
c                 is the order of the tridiagonal matrix.
c
c        c        real(n)
c                 is the subdiagonal of the tridiagonal matrix.
c                 c(2) through c(n) should contain the subdiagonal.
c                 on output c is destroyed.
c
c        d        real(n)
c                 is the diagonal of the tridiagonal matrix.
c                 on output d is destroyed.
c
c        e        real(n)
c                 is the superdiagonal of the tridiagonal matrix.
c                 e(1) through e(n-1) should contain the superdiagonal.
c                 on output e is destroyed.
c
c        b        real(n)
c                 is the right hand side vector.
c
c     on return
c
c        b        is the solution vector.
c
c        info     integer
c                 = 0 normal value.
c                 = k if the k-th element of the diagonal becomes
c                     exactly zero.  the subroutine returns when
c                     this is detected.
c
c     linpack. this version dated 08/14/78 .
c     jack dongarra, argonne national laboratory.
c
c     no externals
c     fortran abs
c
c     internal variables
c
      integer k,kb,kpl,nml,nm2
      real t
c     begin block permitting ...exits to 100
c
         info = 0
         c(1) = d(1)
         nml = n - 1
         if (nml .lt. 1) go to 40
            d(1) = e(1)
            e(1) = 0.0e0
            e(n) = 0.0e0
c
            do 30 k = 1, nml
               kpl = k + 1
```

```
c
c              find the largest of the two rows
c
               if (abs(c(kp1)) .lt. abs(c(k))) go to 10
c
c                 interchange row
c
                  t = c(kp1)
                  c(kp1) = c(k)
                  c(k) = t
                  t = d(kp1)
                  d(kp1) = d(k)
                  d(k) = t
                  t = e(kp1)
                  e(kp1) = e(k)
                  e(k) = t
                  t = b(kp1)
                  b(kp1) = b(k)
                  b(k) = t
   10          continue
c
c              zero elements
c
               if (c(k) .ne. 0.0e0) go to 20
                  info = k
c     .............exit
                  go to 100
   20          continue
               t = -c(kp1)/c(k)
               c(kp1) = d(kp1) + t*d(k)
               d(kp1) = e(kp1) + t*e(k)
               e(kp1) = 0.0e0
               b(kp1) = b(kp1) + t*b(k)
   30       continue
   40    continue
         if (c(n) .ne. 0.0e0) go to 50
            info = n
         go to 90
   50    continue
c
c           back solve
c
            nm2 = n - 2
            b(n) = b(n)/c(n)
            if (n .eq. 1) go to 80
               b(nm1) = (b(nm1) - d(nm1)*b(n))/c(nm1)
               if (nm2 .lt. 1) go to 70
               do 60 kb = 1, nm2
                  k = nm2 - kb + 1
                  b(k) = (b(k) - d(k)*b(k+1) - e(k)*b(k+2))/c(k)
   60          continue
   70          continue
   80       continue
   90    continue
  100 continue
c
      return
      end
```

B-6. COMMON.INC

```
      Parameter  (iX=80,iZ=15,iT=2)
      Common/X1/U(iX,iZ,iT),S(iX,iZ,iT),HAdvM(iX,iZ),HDifM(iX,iZ),
     *  HAdv(iX,iZ),VAdv(iX,iZ),HDif(iX,iZ),VDif(iX,iZ),W(iX,iZ),
     *  EPz(iX,iZ),EPx(iX,iZ),Ez(iX,iZ),Ex(iX,iZ),Press(iX,iZ),
     *  Rho(iX,iZ),Frik(iX,iZ),B(iX,iZ),BU(iX,iZ),BW(iX,iZ),
     *  Depth(iX,iZ),HMZ(iX,iZ),QLat(iX,iZ),PSQ(iX,iZ),SIC(iX,iZ),
     *  Ada(iX,iT),H1(iX,iT),StH1(iX,iT),StS(iX),rMD(iX),SST(iX),
     *  SSTm(iX),DelA(iX),StH(iX),StSlop(iX),KB(iX),KBU(iX),iEC(iX),
     *  ARD(iX),H(iZ),DH(iZ),SFLDM(iZ),QML2,QML1,XAML,QatUB,Darea,
     *  MST,Wave1,Wave2,iWave,CWave,CTurb,CRi,ML,ML1,MLs1,MU,MU1,MUs1,
     *  iWQ,iSalt,iDisch,iTide,iSatDB,iPS,iUBC,iTdep,DX,DTs,DTD,dtx,
     *  Tout(30),iTmax,IncP,Hour,rFDTL,iFDTL,NS2Sal,iTSIC,Wdrag,rk,
     *  WQout(30),IncW,iWQTP,iWQTPs1,TinDay,SDoDT(iX),TotB(iX),iTurb,
     *  BT(iX),BHT(iX,iZ),BU2H(iX,iZ),COUR2(iX,iZ),CK(iX,iZ),CC(iX,iZ),
     *  DD(iX,iZ),EE(iX,iZ),DTi2,DTo2,DX2,DX4,DXS2,DXS4,DXiG,DXi2,G4m,
     *  Alpha(iX),CAlpha(iX),Gam(iX,iZ),Del(iX,iZ),T1,T2,T3,dtxx,iHAdv,
     *  iBotSh,Frik2(iX,iZ),Uo(iX,iZ),rMan(iX),Paxis(iX),WStrs(iX),
     *  iSDBC,iWQDBC
      Common/X2/Savg(iX,iZ),Uavg(iX,iZ),Wavg(iX,iZ),EPzAvg(iX,iZ),
     *  EzAvg(iX,iZ),SSBF(iX,iZ),SSBE(iX,iZ),Tavg(iX),Amax(iX),Umax(iX),
     *  Amin(iX),Umin(iX),TLO(iX),THI(iX),TSBF(iX),TSBE(iX),AvgN,
     *  DistT(iX),DistS(iX)
```

B-7. BLKDATA.INC

```
      Data U,S/2400*0.0,2400*30.0/, HAdvM,HDifM/2400*0.0/
      Data HAdv,VAdv,HDif,VDif,W,EPz,EPx,Ez,Ex,Frik,B,BU,BW/15600*0.0/
      Data QLat,PSQ,Ada,H1,StH1/2880*0.0/, iEC/80*0/
      Data Wave1,Wave2/2*0.0/, IncP/1/, Hour,TinDay/2*0.0/
      Data IncW/1/, COUR2,CK/2400*0.0/, CC,DD,EE/3600*0.0/
      Data Frik2,Uo/2400*0.0/
      Data Savg,Uavg,Wavg,EPzAvg,EzAVG,SSBF,SSBE,Tavg/8480*0.0/
      Data AMax,Umax/160*-1.0E5/, AMin,Umin/160*1.0E5/
      Data TSBF,TSBE,AvgN/161*0.0/
```

B-8. COM-WQ1.INC

```
      INCLUDE 'COMMON.INC'
      Common/WQV/N1(iX,iZ,iT),HydN(iX,iZ),BenN1(iX,iZ),WPN1(iX),
     *  rN1IC(iX,iZ),WN1(iX),rKn12(iX),xKn12(iX),rKn11(iX),StN1(iX),
     *  HydNSt(iX),TheN12,rKh12,CNdie1,DSN1,
     * N2(iX,iZ,iT),rNit(iX,iZ),PR2(iX,iZ),BenN2(iX,iZ),WPN2(iX),
     *  rN2IC(iX,iZ),WN2(iX),rKn23(iX),xKn23(iX),StN2(iX),StNit(iX),
     *  StPR2(iX),TheN23,rKh23,CNdie2,DSN2,rKnit,rKden,
     * N3(iX,iZ,iT),PR3(iX,iZ),BenN3(iX,iZ),WPN3(iX),rN3IC(iX,iZ),
     *  WN3(iX),rKn33(iX),xKn33(iX),StN3(iX),StPR3(iX),DSN3,TheN33,
     * P1(iX,iZ,iT),HydP(iX,iZ),BenP1(iX,iZ),WPP1(iX),P1IC(iX,iZ),
     *  WP1(iX),rKp12(iX),xKp12(iX),rKp11(iX),StP1(iX),HydPSt(iX),
     *  TheP12,rKhp12,CPdie1,DSP1,
     * P2(iX,iZ,iT),BenP2(iX,iZ),WPP2(iX),P2IC(iX,iZ),WP2(iX),
     *  rKp22(iX),StP2(iX),CPdie2,DSP2,
     * Chl(iX,iZ,iT),G(iX,iZ),ChlIC(iX,iZ),WChl(iX),Turb(iX),rKchl(iX),
     *  StChl(iX),StG(iX),rKgr,xKgr,ThetaG,rIs,TU,TD,rIn,PTT,rKmn,
     *  rKmp,RespR,Resp20,ThetaR,GrazR,Graz20,ThetaD,DSChl,
     * CBOD(iX,iZ,iT),BenBOD(iX,iZ),WPBOD(iX),BODIC(iX,iZ),WBOD(iX),
     *  rKc(iX),xKc(iX),rKbod(iX),StBOD(iX),TheKc,BODdie,DSBOD,
     * DOC(iX,iZ,iT),BenDO(iX,iZ),WPDO(iX),DOIC(iX,iZ),WDO(iX),
     *  WRea(iX),StDO(iX),PQ,RQ,DOresp,TDOs,STDOs,TDOaer,TheDA,DSDO,
     * DnN1(iZ),DnN2(iZ),DnN3(iZ),DnP1(iZ),DnP2(iZ),DnChl(iZ),
     *  DnBOD(iZ),DnDO(iZ), StBN1(iX),StBN2(iX),StBN3(iX),StBP1(iX),
     *  StBP2(iX),StBBOD(iX),StBDO(iX),
     * CURVN1(iX,iZ),CURVN2(iX,iZ),CURVN3(iX,iZ),CURVP1(iX,iZ),
     *  CURVP2(iX,iZ),CURVChl(iX,iZ),CURVBOD(iX,iZ),CURVDO(iX,iZ),
     * iSun,iBCWQ,an,ar,ap,ac,Fn,Fp,XAL(iX,iZ),XAm(iX),NS2WQ,
     *  iTWQIC,Bfrac(iX,iZ),iNPS,iWQPS,rKro, rKDO
      Real N1,N2,N3
```

B-9. COM-WQ2.INC

```
      Common/WQPrt/AvgN1(iX,iZ),AvgN2(iX,iZ),AvgN3(iX,iZ),AvgP1(iX,iZ),
     *  AvgP2(iX,iZ),AvgChl(iX,iZ),AvgBOD(iX,iZ),AvgDO(iX,iZ),
     * rN1Max(iX,iZ),rN2Max(iX,iZ),rN3Max(iX,iZ),P1Max(iX,iZ),
     *  P2Max(iX,iZ),ChlMax(iX,iZ),BODMax(iX,iZ),DOMax(iX,iZ),
     * rN1Min(iX,iZ),rN2Min(iX,iZ),rN3Min(iX,iZ),P1Min(iX,iZ),
     *  P2Min(iX,iZ),ChlMin(iX,iZ),BODMin(iX,iZ),DOMin(iX,iZ),AvgNWQ
```

B-10. BLKD-WQ.INC

```
      Data HydN,BenN1,WPN1,WN1,rKn12,HydNSt,DSN1/2721*0.0/
      Data rNit/1200*0.0/, PR2/1200*1.0/, BenN2,WPN2,
     *     WN2,rKn23,StNit/1520*0.0/, StPR2/80*1.0/, DSN2/0.0/
      Data PR3,BenN3,WPN3,WN3,StPR3,DSN3/2641*0.0/
      Data HydP,BenP1,WPP1,WP1,rKp12,HydPSt,DSP1/2721*0.0/
      Data BenP2,WPP2,WP2,DSP2/1361*0.0/
      Data G,WChl/1280*0.0/, Turb/80*1.0/, StG,DSChl/81*0.0/
      Data BenBOD,WPBOD,WBOD,rKc,DSBOD/1441*0.0/
      Data BenDO,WPDO,WDO,WRea,DSDO/1441*0.0/
      Data CURVN1,CURVN2,CURVN3,CURVP1,CURVP2,CURVChl,CURVBOD,
     *     CURVDO/9600*0.0/
      Data AvgN1,AvgN2,AvgN3,AvgP1,AvgP2,AvgChl,AvgBOD,AvgDO/9600*0.0/
      Data rN1Max,rN2Max,rN3Max,P1Max,P2Max,ChlMax,BODMax,
     *     DOMax/9600*-1000.0/
      Data rN1Min,rN2Min,rN3Min,P1Min,P2Min,ChlMin,BODMin,
     *     DOMin/9600*1000.0/, AvgNWQ/0/
```

APPENDIX C. GENERAL DESCRIPTION AND OPERATING MANUAL

C-1. Hardware/Software Requirements

The model has been developed using an IBM 486 PC with the "Lahey F77L-EM/32" compiler. To compile and run the model, the following hardware is required (based up iX = 80 and iZ = 15). The source programs occupy ca. 0.2 MB of hard disk space, the executable file ca. 1 MB and the input files ca. 0.2 MB. Storage requirement for output files is determined by the number of times to print out the model results: ca. 0.5 MB of hard disk space is needed for printing out the model results once. The "Lahey F77L-EM/32" compiler requires ca. 5 MB of hard disk space and minimum 2 MB of available memory.

C-2. Program Organization

Program input and output require several data files, each of which is specified in the program by a logical unit number. Tables C-1 and C-2 list the logical unit numbers for hydrodynamic and water quality models, respectively. You can run only the hydrodynamic model. However, the water quality model, supplied with the information concerning physical transport processes from the hydrodynamic model, should be run with the hydrodynamic model. Therefore, all input and output units for water quality model are optional.

C-3. Data Input Description for Hydrodynamic Model

C-3-1. Main input data file (logical unit **5**)

VARIABLE (FORMAT)

iWQ, iSalt, iDisch, iTide, iSatDB, iPS, iUBC (7I5)

: iWQ = 1 causes to run the water quality model (open unit **11** and **8**). Otherwise no

water quality simulation.

: iSalt = 1 if salinity is to be modeled, otherwise salinity not modeled.

: iDisch = 1 if daily varying values are used for freshwater inflow through the upstream boundary (read from unit **10**). Otherwise a constant value is read in from unit **5** and kept throughout the model run.

: iTide = 1 if hourly observations of tidal height at the mouth are used for the downstream boundary conditions (read from unit **6**). Otherwise nine tidal amplitudes and phases are read in from unit **5** to generate tidal heights by harmonics.

: iSatDB = 1 if daily varying salinities are used for the downstream boundary conditions (read from unit **9**). Otherwise constant values are read in from unit **5** and kept throughout the model run.

: iPS = 1 if daily varying values are used for point source discharge (read from unit **13**). Otherwise constant values are read in from unit **5** and kept throughout the model run.

: iUBC = 1 causes that only mass (no momentum) is input from freshwater discharge at the upstream boundary; this is so, when the fall line is well developed to create supercritical flow at the upstream boundary. Otherwise both mass and momentum are input from freshwater discharge.

iWave, iHAdv, iTurb, iBotSh, iSDBC (5I5)

: iWave = 1 if daily varying values are used for wind-induced mixing and momentum input (read from unit **20**). Otherwise constant values (including 0) are read in from unit **5** and kept throughout the model run.

: iHAdv should be either 3 or 2. The horizontal advection term in mass balance equations is solved using the QUICKEST scheme if iHAdv = 3 or the upwind-weighted difference scheme if iHAdv = 2.

: iTurb should be either 3 or 2. The vertical mixing terms are evaluated using the

Munk-Anderson type stability functions (Equations 2-18 and 2-19) if iTurb = 3 or the Mellor and Yamada Level 2 turbulence scheme (Equations 2-29 and 2-30) if iTurb = 2.

: iBotSh should be either 3 or 2. The bottom shear stress is calculated using the Manning's friction coefficient (Eq. 2-9) if iBotSh = 3 or the bottom roughness height (Eq. 2-10) if iBotSh = 2.

: iSDBC should be either 3 or 2. If iSDBC = 3, the salinity at the downstream boundary is estimated using the subroutine 'SOpBdry' in 'HYD-2D.FOR'. If iSDBC = 2, the salinity at this boundary is independent of tidal phase and the values may be read from either unit **5** (constant throughout simulation) or **9** (daily varying).

At this point, control passes to the subroutine 'ReadM' in 'HYD-2D.FOR'.

ML, MU, Kmax (3I5)

: ML = upstream boundary segment number. For proper operation, ML $\geq$ 2.

: MU = downstream boundary segment number.

: Kmax = maximum number of layers in any segment.

Note that common and dimension statements presently implemented limit MU to a maximum value of 80 and Kmax of 15.

Darea (F8.0)

: drainage area (km^2) upstream of the fall line, i.e., at the gauging station at the upstream boundary.

KB(i), i=ML,MU (15I5)

: number of layers at the segment i.

H(k), k=1,Kmax (10F8.0)

: layer thickness (m) at the layer k.

(B(i,k), k=1,KB(i)) i=ML,MU (10F8.0)

: main channel width (m) at the segment i and the layer k.

SSTm(i), i=ML,MU (10F8.0)

: side storage area (km^2) at mean tide at the segment i. Since SSTm(i) is defined at the center of each segment, SSTm(MU) is not needed, which can be set to any number, let say, SSTm(MU) = 0.

StSlop(i), i=ML,MU (10F8.0)

: increase in width (in cm) of side storage area for each centimeter increase in surface elevation at the segment i; StSlop(MU) = 0.

StH(i), i=ML,MU (10F8.0)

: mean depth (m) of side storage area at mean tide at the segment i; StH(MU) = 0.

ARD(i), i=ML,MU (10F8.0)

: land drainage area (km^2) feeding into the segment i, i.e., between the transect i and i+1; ARD(MU) = 0.

DX, DmDB, DTs (3F8.0)

: DX = Δx = distance (m) between two neighboring transects = distance between the centers of two neighboring segments.

: DmDB = distance (km) from the river mouth to the center of the most downstream segment.

: DTs = Δt = time increment in seconds.

Tmax, NP (F8.0, I8)

: Tmax = duration of model run in tidal cycles.

: NP = number of times during model run to print out the results.

Tout(m), m=1,NP (10F8.0)

: At the time of Tout(m) tidal cycles from model start, the subroutine 'HOut' in 'MODEL-2D.FOR' prints out

to 'HYD.OUT' (unit **7**),

- η, u, w and s at Tout(m) tidal cycles,
- their averages over the last tidal cycle between Tout(m)-1 and Tout(m), and
- range of surface current velocity over the last tidal cycle,

to 'TRANGE.OUT' (unit **12**),

- tidal range and time of high tide and low tide during the last tidal cycle,

to 'SLACK.OUT' (unit **14**),

- time of, and salinity at, slackwaters during the last tidal cycle,

to 'E.OUT' (unit **15**)

- the vertical viscosity and diffusivity at Tout(m) tidal cycles, and
- their averages over the last tidal cycle,

Note that the present model implementation allows the maximum of 30 times of output printing, and that $0 \leq$ Tout(m) $\leq$ Tmax.

At this point, control passes to the main program and hence to the subroutine 'ReadH' in 'HYD-2D.FOR'.

TempC (F8.0)

: constant estuarine temperature (°C). In water quality model, daily varying temperature, if desired, can be used.

Cstar, RhoAir (2F8.0)

: Cstar = dimensionless drag coefficient (ca. 1.3×10^{-3}) = C_D in Eq. 2-7.

: RhoAir = air density (ca. 1.2×10^{-3} g cm^{-3}) = ρ_a in Eq. 2-7.

Paxis(i), i=ML,MU (10F8.0)

: principal axis angle at the transect i. Paxis(i) = 0° to the east and increases counterclockwise direction.

rMan(i), i=ML,MU (10F8.0)

: Manning's friction coefficient at the transect i if iBotSh = 3

: bottom roughness height (cm) at the transect i if iBotSh = 2.

The bottom shear stress is calculated using the Manning's friction coefficient (Eq. 2-9) if iBotSh = 3 or the bottom roughness height (Eq. 2-10) if iBotSh = 2.

rk (E8.3)

: proportionality constant relating water density to salinity (7.5×10^{-4} ppt^{-1}) = k in Eq. 2-6.

CTurb, CRi, CWave, CHD (4E8.3)

: CTurb = constant for turbulent mixing coefficients = α in Eq. 2-18.

: CRi = constant for Richardson number = β in Eq. 2-18.

: CWave = constant for wind-induced mixing = α_w in Eq. 2-18.

: CHD = timewise constant horizontal mixing coefficient (cm^2 sec^{-1}).

HtP, Wave2, WndSpd, WndDir (4F8.0)

: HtP = H_w^2/T (cm^2 sec^{-1}) in Eq. 2-18.

: Wave2 = $2\pi/L$ (cm^{-1}) in Eq. 2-18.

: WndSpd = wind speed (m sec^{-1}) at a height of 10 m = U_w in Eq. 2-7.

: WndDir = wind direction (degree) to which the wind blows.

The height (H_w), period (T) and length (L) of the wind-induced waves can be evaluated, for example, using the Sverdrup-Munk-Bretschneider forecasting curves (U.S. Army Coastal Engineering Research Center 1973). Note that, like Paxis(i), WndDir = 0° to the east and increases counterclockwise direction. Regardless of iWave, this line is read in from unit **5** but will not be used in model calculation if iWave = 1.

At this point, if iWave = 1, a line of HtP, Wave2, WndSpd, WndDir is read in from unit **20**, which is explained in Section C-3-2.

Alpha(i), i=ML,MU (10F8.0)

: weighting factor for advective mass transport at the transect i. Regardless of iHAdv, these factors are read in from unit **5** but will not be used if iHAdv = 3.

SalUp, SalDn, NLS (2F8.0, I8)

: Salinity field is initialized using linear interpolation of SalDn (salinity at the downstream boundary) and SalUp (salinity at the segment NLS, usually 0), where NLS is the segment number to which salt is initially intruded.

NS2Sal (I5)

: NS2Sal = 1 if one likes to specify the salinity distribution at any time. Otherwise salinity is initialized using SalUp, SalDn and NLS at time zero, i.e., when model starts.

TSIC (F8.0) if NS2Sal = 1
(SIC(i,k), k=1,KB(i)) i=ML,MU (10F8.0) if NS2Sal = 1

: TSIC is the time (in days) to specify the read-in salinity distribution, SIC(i,k).

TOFH (F8.0)
SFLDM(k), k=1,KB(i) (10F8.0)

: TOFH is the time (in hours) required for the salinity at the downstream boundary to reach the downstream boundary condition, SFLDM(k), after SBF. TOFH also is used for eight water quality parameters. Regardless of iSatDB, SFLDM(k) is read in from unit **5** but will not be used in model calculation if iSatDB = 1.

At this point, if iSatDB = 1, SFLDM(k) is read in from unit **9**, which is explained in Section C-3-3.

S(MLs1,k,2), k=1,KB(MLs1) (10F8.0)

: constant salinity at the upstream boundary (usually 0). Segment MLs1 = segment (ML-1).

Ampl(m), Phase(m), m=1,9 (2F8.0)

: amplitudes (cm) and phases (radian) for the nine most important tidal components in the following order: M2, S2, N2, K1, M4, O1, MM, SSA, SA. Regardless of iTide, these nine lines of data are read in from unit **5** but will not be used in model calculation if iTide = 1.

At this point, if iTide = 1, control passes to the subroutine 'ReadTD' in 'HYD-2D.FOR', which is explained in Section C-3-4.

NS2PS (I5)

: number of segments, which point source discharge feeds into.

i, xPSQ(i), m=1,NS2PS (9X, I5, F7.3) if iPS $\neq$ 1

: i = segment number; ML $\leq$ i < MU.

: xPSQ(i) = constant point source discharge (m^3 sec^{-1}) feeding into segment i.

At this point, if iPS = 1, NS2PS lines are read in from unit **13**, which is explained in Section C-3-5.

QML2 (F8.0)

: constant freshwater discharge rate (m^3 sec^{-1}) through the transect ML (i.e., through the upstream boundary transect). Regardless of iDisch, this line is read in from unit **5** but will not be used in model calculation if iDisch = 1.

At this point, if iDisch = 1, two lines (FDTL and QML2) are read in from unit **10**, which is explained in Section C-3-6.

End of logical unit 5.

At this point, control comes back to the main program. If iWQ = 1, control passes to the subroutine 'WQC1' in 'WQ-2D-IN.FOR', which is explained in Section C-4.

C-3-2. Time-varying wind parameters (logical unit **20**)

HtP, Wave2, WndSpd, WndDir (9X, 4F8.3)

: If iWave = 1, the daily varying four parameters are read in from a separate file (unit **20**). The format is that the first 9 columns are used for the sampling date and ignored when being read, and each line has four values (4F8.3). One line is read in every midnight.

End of logical unit 20.

C-3-3. Time-varying salinity at downstream boundary (logical unit **9**)

SFLDM(k), k=1,KB(i) (9X, 10F8.3)

: If iSatDB = 1, the daily varying SFLDM(k) is read in from a separate file (unit **9**). The format is that the first 9 columns are used for identifying the date and ignored when being read, and the next 10 columns for SFLDM(k). One line is read in every midnight.

End of logical unit 9.

C-3-4. Time-varying surface elevation at downstream boundary (logical unit **6**)

Tdata(m), m=1,24 (7X, 12F6.2)

: If iTide = 1, the subroutine 'ReadTD' in 'HYD-2D.FOR' reads in Tdata(m), for the boundary tidal heights at the segment MU, from a separate file (unit **6**), which contains hourly-measured tidal heights (in ft). The format is that the first 7 columns are used for the sampling date and ignored when being read, and every line has 12 values (12F6.2). Two lines are read in every midnight. The read-in Tdata(m) is converted into cm and used in the subroutine 'OTide' in 'HYD-2D.FOR' to calculate the tidal heights

at the downstream boundary at each time step.

End of logical unit 6.

C-3-5. Time-varying point source discharges (logical unit **13**)

i, xPSQ(i), m=1,NS2PS (9X, I5, F7.3)

: If iPS = 1, the daily varying xPSQ(i) is read in from a separate file (unit **13**). The format is that the first 9 columns are used for the sampling date and ignored when being read, and the next 5 columns are for i (segment number into which xPSQ feeds) and the next 7 columns for xPSQ(i). NS2PS lines are read in every midnight.

End of logical unit 13.

C-3-6. Time-varying freshwater inflow through upstream boundary (logical unit **10**)

FDTL (F8.0)
i, QML2 (9X, I5, F7.3)

: If iDisch = 1, the daily varying QML2 into the transect i (= ML) is read in from another separate file (unit **10**). FDTL, which is the time lag (in days) to adjust for varying freshwater discharge, occupies 8 columns of the first line, and the rest of the file is for QML2, the format of which is the same as that of logical unit **13**.

End of logical unit 10.

C-4. Data Input Description for Water Quality Model

As mentioned at the end of Section C-3-1, control passes to the subroutine 'WQC1' in 'WQ-2D-IN.FOR' if iWQ = 1.

VARIABLE (FORMAT)

iTdep, iSun, iBCWQ, iNPS, iWQPS, iWQDBC (6I5)

: iTdep = 1 causes time-varying water quality input conditions so that the subroutine 'WQInput' in 'WQ-2D-IN.FOR' will be called every midnight. If any of iSun, iBCWQ, iNPS, iWQPS equals 1, iTdep must be 1. Otherwise constant water quality conditions.

: iSun = 1 causes the daily varying parameters for solar radiation to be read in from unit **16**. Otherwise constant parameters are read in from unit **11** and kept throughout the model run.

: iBCWQ = 1 causes the daily varying boundary conditions for water quality parameters to be read in from unit **17**. Otherwise constant conditions are read in from unit **11** and kept throughout the model run.

: iNPS = 1 causes the daily varying nonpoint source loadings to be read in from unit **18**. Otherwise constant loadings are read in from unit **11** and kept throughout the model run. At this point, if iNPS = 1, a line is read in for FDTL from unit **18**.

: iWQPS = 1 causes the daily varying point source loadings to be read from unit **19**. Otherwise constant loadings are read in from unit **11** and kept throughout the model run.

: iWQDBC should be either 3 or 2. If iWQDBC = 3, the downstream boundary conditions are estimated using the subroutine 'WDnBdry' in 'WQ-2D.FOR'. If iWQDBC = 2, the boundary conditions are independent of tidal phases and the values may be read from either unit **11** (constant throughout simulation) or **17** (daily varying).

NPWQ (I5)

: NPWQ = number of times to print out the results.

WQout(m), m=1,NPWQ (10F8.0)

: At the time of WQout(m) days from model start, the subroutine 'WOut' in 'WQ-2D-IN.FOR' prints out to 'WQ.OUT' (unit **8**),

- N1, N2, N3, P1, P2, Chl, CBOD, DO at WQout(m) days, and
- their daily averages, maximums and minimums over WQout(m)-1 and WQout(m)

days.

Note that the present model implementation allows the maximum of 30 times of output printing, and that $0 \leq$ WQout(m) $\leq$ Tmax.

TheN12, TheN23, TheN33, TheP12 (4F8.0)

: constants for temperature adjustment of ammonification, nitrification, denitrification and organic phosphorus mineralization rates, respectively.

NS2 (I5)

: number of segments for which nutrient transfer coefficients will be read. Setting NS2 = 2 establishes uniform values, otherwise NS2 $\geq$ MU.

xKn12(i), i=2,NS2 (10F8.0)
xKn23(i), i=2,NS2 (10F8.0)
xKn33(i), i=2,NS2 (10F8.0)
xKp12(i), i=2,NS2 (10F8.0)

: rate in mg l^{-1} day^{-1} for ammonification, nitrification, denitrification (day^{-1}) and organic phosphorus mineralization at 20°C, respectively.

rKh12, rKh23, rKnit, rKden, rKhp12 (5F8.0)

: half-saturation concentrations in mg l^{-1} for ammonification, nitrification, oxygen limitation of nitrification, denitrification and organic phosphorus mineralization, respectively.

NS2 (I5)

: number of segments for which CBOD decay rates will be read. Setting NS2 = 2 establishes uniform values, otherwise NS2 $\geq$ MU.

xKc(i), i=2,NS2 (10F8.0)

: CBOD decay rate at 20°C in day^{-1}.

TheKc (F8.0)

: constant for temperature adjustment of CBOD decay rate.

TheDA, rKro (2F8.0)

: TheDA = constant for temperature adjustment of DO reaeration rate.

: rKro = proportionality constant for DO reaeration (393.3 in CGS unit).

At this point, control comes back to the main program and thence to the subroutine 'WQC2' in 'WQ-2D-IN.FOR'.

NS2 (I5)

: number of segments for which settling rates will be specified. Setting NS2 = 2 establishes uniform values, otherwise NS2 ≥ MU.

rKn11(i), i=2,NS2 (10F8.0)
rKp11(i), i=2,NS2 (10F8.0)
rKp22(i), i=2,NS2 (10F8.0)
rKchl(i), i=2,NS2 (10F8.0)
rKbod(i), i=2,NS2 (10F8.0)

: settling rates of organic nitrogen, organic phosphorus, inorganic phosphorus, chlorophyll and CBOD, respectively, in cm day^{-1}. Note that the present model implementation uses the same settling rates in the main channel and the storage area.

ac, an, ap, ar, PQ, RQ, rKmn, rKmp, xKgr, rIs,
Resp20, Graz20, Fn, Fp, ThetaG, ThetaR, ThetaD (10F8.0)

: ac, an, ap = carbon/chlorophyll (mg C per μg Chl), nitrogen/chlorophyll (mg N per μg Chl), phosphorus/chlorophyll (mg P per μg Chl) ratios, respectively, in algal biomass (typically 0.05, 0.007 and 0.001, respectively).

: ar = proportion of consumed phytoplankton recycled by zooplankton. To conserve nitrogen and phosphorus, ar = 1.

: PQ = ratio of oxygen produced to carbon fixed (moles O_2 per mole C).

: RQ = ratio of CO_2 liberated to oxygen consumed (moles CO_2 per mole O_2).

: rKmn, rKmp = half-saturation concentrations for growth limitation due to nitrogen and phosphorus, respectively, in mg l^{-1}.

: xKgr = optimum growth rate at 20°C in day^{-1}.

: rIs = saturation light intensity in langleys day^{-1}.

: Resp20 = respiration rate at 20°C in day^{-1}.

: Graz20 = predatory and other mortality rate at 20°C in day^{-1}.

: Fn, Fp = fraction of metabolically produced nitrogen and phosphorus, respectively, recycled to the organic pool.

: ThetaG, ThetaR, ThetaD = constants for temperature adjustment of growth rate, respiration rate, and grazing and other mortality rate, respectively.

At this point, control comes back to the main program and thence to the subroutine 'WQIC' in 'WQ-2D-IN.FOR'.

xN1, xN2, xN3, xP1, xP2, xChl, xBOD, xDO (8F8.0)

: spacewise constant initial concentrations in mg l^{-1} (except xChl in μg l^{-1}). These values are used if NS2WQ $\neq$ 1.

NS2WQ (I5)

: NS2WQ = 1 causes to specify the initial conditions at any time. Otherwise the spacewise constant values (xN1,xN2,xN3,xP1,xP2,xChl,xBOD,xDO) are used for initial conditions at time zero, i.e., when model starts.

TWQIC (F8.0) if NS2WQ = 1

: TWQIC is the time (in days) to specify the following initial conditions in mg l^{-1} (except ChlIC in μg l^{-1}).

(rN1IC(i,k), k=1,KB(i)) i=ML,MU (10F8.0) if NS2WQ = 1

(rN2IC(i,k), k=1,KB(i)) i=ML,MU (10F8.0) if NS2WQ = 1
(rN3IC(i,k), k=1,KB(i)) i=ML,MU (10F8.0) if NS2WQ = 1
(P1IC(i,k), k=1,KB(i)) i=ML,MU (10F8.0) if NS2WQ = 1
(P2IC(i,k), k=1,KB(i)) i=ML,MU (10F8.0) if NS2WQ = 1
(ChlIC(i,k), k=1,KB(i)) i=ML,MU (10F8.0) if NS2WQ = 1
(BODIC(i), k=1,KB(i)) i=ML,MU (10F8.0) if NS2WQ = 1
(DOIC(i,k), k=1,KB(i)) i=ML,MU (10F8.0) if NS2WQ = 1

Control comes back to the main program and thence to the subroutine 'WQInput' in 'WQ-2D-IN.FOR'.

Title (A49)

: alpha information describing the input data that follow. The date at which the data are read in, can be an example of Title.

NDG, NS2 (2I5)

: NDG = integer identifying a particular data group; the present model implementation has eight possible data groups.

: NS2 = number of values to be read in each data group.

1) If **NDG = 1**, then boundary conditions

- from unit **17** if iBCWQ = 1 (daily varying conditions), or
- from unit **11** if iBCWQ ≠ 1 (constant conditions).

N1(MLs1,k,2), k=1,KB(MLs1) (9X, 10F8.3)
N2(MLs1,k,2), k=1,KB(MLs1) (9X, 10F8.3)
N3(MLs1,k,2), k=1,KB(MLs1) (9X, 10F8.3)
P1(MLs1,k,2), k=1,KB(MLs1) (9X, 10F8.3)
P2(MLs1,k,2), k=1,KB(MLs1) (9X, 10F8.3)
Chl(MLs1,k,2), k=1,KB(MLs1) (9X, 10F8.3)
CBOD(MLs1,k,2), k=1,KB(MLs1) (9X, 10F8.3)
DOC(MLs1,k,2), k=1,KB(MLs1) (9X, 10F8.3)

: upstream boundary conditions in mg l^{-1} (except Chl in μg l^{-1}).

DnN1(k), k=1,KB(MU) (9X, 10F8.3)
DnN2(k), k=1,KB(MU) (9X, 10F8.3)
DnN3(k), k=1,KB(MU) (9X, 10F8.3)
DnP1(k), k=1,KB(MU) (9X, 10F8.3)
DnP2(k), k=1,KB(MU) (9X, 10F8.3)
DnChl(k), k=1,KB(MU) (9X, 10F8.3)
DnBOD(k), k=1,KB(MU) (9X, 10F8.3)
DnDO(k), k=1,KB(MU) (9X, 10F8.3)

: downstream boundary conditions in mg l^{-1} (except Chl in $\mu g\ l^{-1}$).

Note that

: NS2 for this group can be any arbitrary number.

: Even if iBCWQ = 1, the general input file (unit **11**) should have a line containing NDG and NS2 whenever boundary conditions need to be updated. Actual values for both downstream and upstream conditions are read in every midnight from a separate file (unit **17**).

: For the freshwater discharge through the upstream boundary, the present model implementation can take care of two different cases; only the mass input (iUBC = 1) or both momentum and mass fluxes (iUBC $\neq$ 1) from discharge. When iUBC = 1, the mass input from freshwater discharge will be incorporated by being treated as nonpoint source and by not including the advective and diffusive transport through the upstream boundary transect. This can be done by setting EPx(ML,k) and Ex(ML,k) in the subroutine 'ReadH' and U(ML,k,2) in 'GetQLat' to be zero if iUBC = 1. The upstream boundary conditions and Alpha(ML) can be any values. When iUBC $\neq$ 1, the freshwater discharge will be treated as the advective flux through the upstream boundary, and thus Alpha(ML) should be 1 if iHAdv = 2 (i.e., upwind-weighted difference scheme).

2) If **NDG = 2**, then point source input

- from unit **19** if iWQPS = 1 (daily varying), or

- from unit **11** if iWQPS $\neq$ 1 (constant).

i, xPSQ,PN1,PN2,PN3,PP1,PP2,PBOD,PDO, m=1,NS2 (9X, I5, 8F7.3)

: i = segment number into which the point source input feeds; ML $\leq$ i < MU.

: xPSQ = point source discharge rate (m^3 sec^{-1}).

: PN1 to PDO = waste loadings (in kg day^{-1}) from point source(s).

Note that

: Even if iWQPS = 1, the general input file (unit **11**) should have a line containing NDG and NS2 whenever point source data need to be updated. NS2 lines of actual values are read in every midnight from unit **19**.

: The discharges (xPSQ) read in here are not used in model calculation but the values read in hydrodynamic model from unit **13** (time-varying) or **5** (constant) are used.

3) If **NDG = 3**, then nonpoint source input

- from unit **18** if iNPS = 1 (daily varying), or
- from unit **11** if iNPS $\neq$ 1 (constant).

i, DSQ,DSN1,DSN2,DSN3,DSP1,DSP2,DSChl,DSBOD,DSDO (9X, I5, 9F7.3)

: i = segment number into which DSQ feeds (usually i = ML).

: DSQ = nonpoint source discharge in m^3 sec^{-1}.

: DSN1 to DSDO = concentrations in mg l^{-1} (except DSChl in μg l^{-1}).

Note that

: NS2 for this group can be any arbitrary number.

: Even if iNPS = 1, the general input file (unit **11**) should have a line containing NDG and NS2 whenever nonpoint source data need to be updated. One line of actual values is read in every midnight from unit **18**.

: The discharges (DSQ) read in here are not used in model calculation but the values read in hydrodynamic model from unit **10** (time-varying) or **5** (constant) are used.

4) If **NDG = 4**, then benthic fluxes

xBN1(i), i=ML,MU (10F8.0)
xBN1St(i), i=ML,MU (10F8.0)
xBN2(i), i=ML,MU (10F8.0)
xBN2St(i), i=ML,MU (10F8.0)
xBN3(i), i=ML,MU (10F8.0)
xBN3St(i), i=ML,MU (10F8.0)
xBP1(i), i=ML,MU (10F8.0)
xBP1St(i), i=ML,MU (10F8.0)
xBP2(i), i=ML,MU (10F8.0)
xBP2St(i), i=ML,MU (10F8.0)
xBBOD(i), i=ML,MU (10F8.0)
xBBODSt(i), i=ML,MU (10F8.0)
xBDO(i), i=ML,MU (10F8.0)
xBDOSt(i), i=ML,MU (10F8.0)
TBN1,TBN2,TBN3,TBP1,TBP2,TBBOD,TBDO (10F8.0)

: xBN1(i) to xBDO(i) = benthic exchange rates in the main channel of the segment i at 20°C in g m^{-2} day^{-1}.

: xBN1St(i) to xBDOSt(i) = those in the side storage area.

: TBN1 to TBDO = constants for temperature adjustment of exchange rates.

Note that

: NS2 for this group can be any arbitrary number.

: Negative values are losses to sediment and positive values sources from sediment.

: Whenever updating the benthic exchange rates, since they need to be adjusted for temperature, the data group 7 (temperature) should also be read in.

5) If **NDG = 5**, then turbidity

Turb(i), i=ML,MU (10F8.0)

: light extinction coefficient (cm^{-1}). NS2 for this group is arbitrary.

6) If **NDG = 6**, then wind reaeration

WRea(i), i=ML,MU 10F8.0)

: wind-induced reaeration (cm day^{-1}) estimated using Eq. 3-9d. NS2 for this group is arbitrary.

7) If **NDG = 7**, then temperature

Temp (F8.0)

: temperature (°C). Note that this data group must be read after data group 4 (bottom exchange rates including SOD) and CBOD decay rate in the subroutine 'WQC1' in 'WQ-2D-IN.FOR', since these parameters must be temperature-adjusted. NS2 for this group can be any arbitrary number.

8) If **NDG = 8**, then sunlight parameters

- from unit **16** if iSun = 1 (daily varying), or
- from unit **11** if iSun ≠ 1 (constant).

rIa, TU, TD **(9X, 3F8.3)** if iNS2 = 1
rIa **(9X, F8.3)** if iNS2 ≠ 1

: rIa = total solar radiation over one day (langleys day^{-1}).

: TU, TD = hours from midnight to sunrise and sunset, respectively.

Note that

: Even if iSun = 1, the general input file (unit **11**) should have a line containing NDG and NS2 whenever solar parameters need to be updated. One line of actual values is read in every midnight from unit **16**.

: NS2 = 1 means that all 3 parameters will be updated, whereas only rIa will be read in if NS2 ≠ 1. Regardless of iSun, NS2 should be 1 at the first call of the subroutine 'WQInput' since TU and TD as well as rIa are needed.

If **NDG = 99**, exit from the subroutine 'WQInput'.

Note that if iTdep = 1, the subroutine 'WQInput' will be called every midnight.

Table C-1. Data file organization for hydrodynamic model.

DESCRIPTION	LOGICAL UNIT	READ or WRITE	EXAMPLE (REMARK)
general input including geometry, program control parameters and hydrodynamic data	5	read	HYD.IN
daily varying freshwater discharge rate at the upstream boundary transect	10	read	FLOW.IN (optional)
time-varying tidal height at the downstream boundary segment	6	read	TIDE.IN (optional)
daily varying salinity condition at the downstream boundary segment	9	read	S-DBC.IN (optional)
daily varying point source discharge rate	13	read	PSQ.IN (optional)
daily varying wind condition	20	read	WIND-W.IN (optional)
general model output	7	write	HYD.OUT
output having tidal range, and time of high water and low water	12	write	TRANGE.OUT
output having time and salinity at slackwater	14	write	SLACK.OUT
output having vertical viscosity and diffusivity	15	write	E.OUT

Table C-2. Data file organization for water quality model.

DESCRIPTION	LOGICAL UNIT	READ or WRITE	EXAMPLE (REMARK)
general input including program control parameters and water quality data	11	read	WQ.IN (optional)
daily varying solar radiation, and time of sunrise and sunset	16	read	SOLAR.IN (optional)
daily varying boundary condition for water quality parameters	17	read	WQBC.IN (optional)
daily varying nonpoint source loading	18	read	NPS.IN (optional)
daily varying point source loading	19	read	PS.IN (optional)
general model output	8	write	WQ.OUT (optional)