Sparse Hydrodynamic Ocean Code

V1632

Scientific Manual

M. Herzfeld, J. Waring, J. Parslow, N. Margvelashvili,
P. Sakov and J. Andrewartha

CSIRO Marine Research
GPO Box 1538, Hobart 7001
27 July 2010

Introduction.

SHOC (Sparse Hydrodynamic Ocean Code) is a finite difference hydrodynamic model developed by the Environmental Modelling group at CSIRO (Commonwealth Scientific and Industrial Research Organization) Division of Marine Research. This model is intended to be a general purpose model applicable to scales ranging from estuaries to regional ocean domains. Outputs from the model include three dimensional distributions of velocity, temperature, salinity, density, passive tracers, mixing coefficients and sea level. Inputs required by the model include forcing due to wind, atmospheric pressure gradients, surface heat and water fluxes and open boundary conditions (e.g. tides). SHOC is based on the three dimensional equations of momentum, continuity and conservation of heat and salt, employing the hydrostatic and Boussinesq assumptions. The equations of motion are discretised on a finite difference stencil corresponding to the Arakawa C grid. The model uses a curvilinear orthogonal grid in the horizontal and a choice of fixed 'z' coordinates or terrain following σ coordinates in the vertical. The 'z' vertical system allows for wetting and drying of surface cells, useful for modelling regions such as tidal flats where large areas are periodically dry. SHOC has a free surface and uses mode splitting to separate the two dimensional (2D) mode from the three dimensional (3D) mode. This allows fast moving gravity waves to be solved independently from the slower moving internal waves allowing the 2D and 3D modes to operate on different time-steps, resulting in a considerable contribution to computational efficiency. The model uses explicit time-stepping throughout except for the vertical diffusion scheme which is implicit. The implicit scheme guarantees unconditional stability in regions of high vertical resolution. A Laplacian diffusion scheme is employed in the horizontal on geopotential surfaces. Smagorinsky mixing coefficients may be utilised in the horizontal.

SHOC can invoke several turbulence closure schemes, including k-ε, k-ω, Mellor-Yamada 2.5, 2.0 and Csanady type parameterisations. A variety of advection schemes may be used on tracers and 1^{st} or 2^{nd} order can be used for momentum. There also exists a suite of radiation, extrapolation, sponge and direct data forcing open boundary conditions. Input and output is handled through netCDF data formatted files, with the option of submitting ascii text files for simple time-series forcing. The netCDF format allows input of spatially and temporally varying forcing and initialization data in a grid and time-step independent manner. SHOC is capable of performing particle tracking and may be coupled to ecological and sediment transport models.

An generous set of diagnostics are available as output from SHOC, including contributing terms to the momentum balance, vorticity, steric height, CFL stability constraints, flushing times, mixed layer depth, turbulent mixing lengths, mean currents and fluxes of tracers.

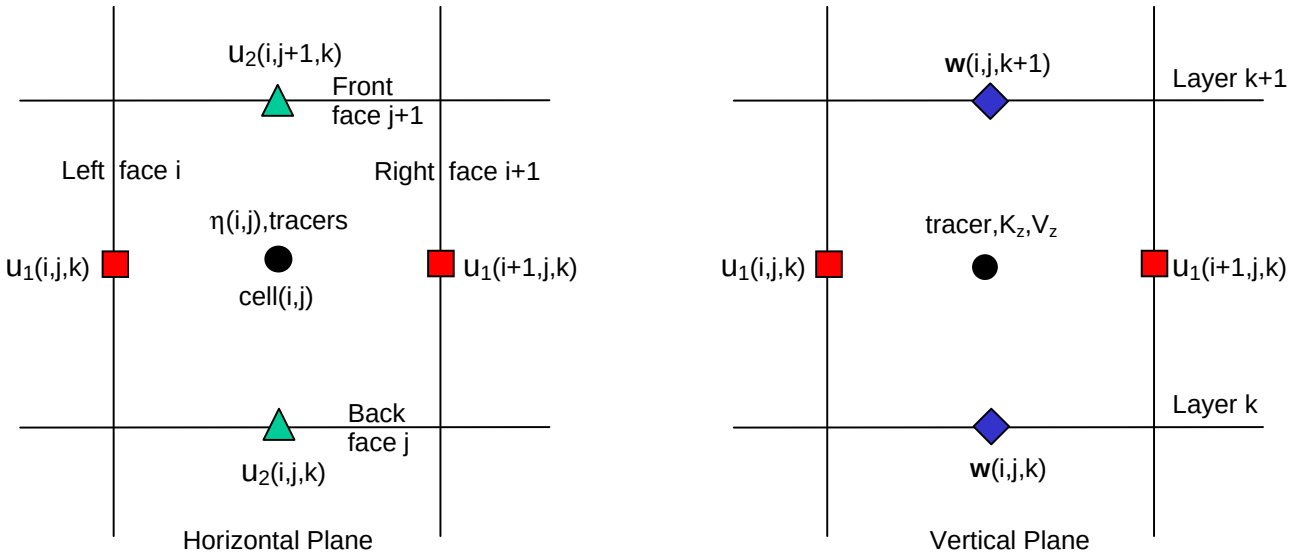
SHOC uses a sparse coordinate system which maps all cells in the grid into a 1-dimensional vector. This process effectively eliminates all land from the domain representation in computer memory. Arbitrary domain composition can be efficiently performed, allowing SHOC to operate in a true distributed processing environment. The sparse representation leads to increases in speed and simplified housekeeping allow techniques such as distributed process, 2-way nesting and hybrid physics to be performed with no overhead.

SHOC is written in C and evolved during 2002 from the hydrodynamic model MECO.

1. Model Discretisation

The equations of motion in SHOC are solved on an Arakawa C grid (Arakawa and Lamb, 1977), where velocity in the x direction (u_1), y direction (u_2) and tracers / elevation occupy different positions in the grid (Figure 1.1). Tracers, vertical velocity, mixing coefficients and elevation occupy the cell centers, u_1 velocities occupy the left faces and u_2 velocities are located on the back faces.

Figure 1.1 : Arakawa C Grid Configuration.



The horizontal finite difference mesh employed in SHOC is orthogonal curvilinear (e.g. Eringen, 1962). This means that the grid lines are not required to be straight, but may curve provided they intersect in a (nearly) orthogonal manner. This type of grid allows the spatial resolution over a domain to vary and is extremely useful in placing higher resolution in areas where it is needed and lower resolution where it is not, resulting in increased computational efficiency from fewer grid cells to process and improved memory usage. The horizontal coordinates in the orthogonal curvilinear coordinate system may be written as (ξ_1, ξ_2) and the vertical coordinate as z. Metric coefficients h_1 and h_2 are defined such that a distance increment ds satisfies:

$$ds = h_1^2 d\xi_1^2 + h_2^2 d\xi_2^2 \qquad 1.1$$

The grid arrangement for a square domain, including boundaries, is illustrated in Figure 1.2. The indexing of the grid in SHOC ranges in the ξ_1 direction from 0 to nce1-1, and an extra face exists at nce1 to accommodate the u_1 velocity on the boundary (only non-zero for open boundaries). This means that the array size in the ξ_1 direction is nce1+1 = nfe1. The same holds in the ξ_2 direction, with the grid size being nfe2.

The vertical coordinates system may either be 'z', where layers occur at fixed distances from a datum, or σ (Phillips, 1957) where the layers are scaled to the depth via:

$$\sigma = \frac{z - \eta}{H + \eta} \qquad 1.2$$

where z is the corresponding 'z' coordinate, η is the surface elevation and H is the water depth. These vertical coordinate systems are illustrated in Figure 1.3. Note that z=0 in the 'z' system

corresponds to the bottom-most layer. The sigma system allows better resolution in the bottom boundary layers over variable topography but suffers truncation error in the pressure term when bathymetric gradients become large (Mellor et al 1994, 1998). The sigma system also requires a minimum depth at the coast where the layers converge.

Figure 1.2 : Grid Arrangement in SHOC ($u=u_1$, $v=u_2$)

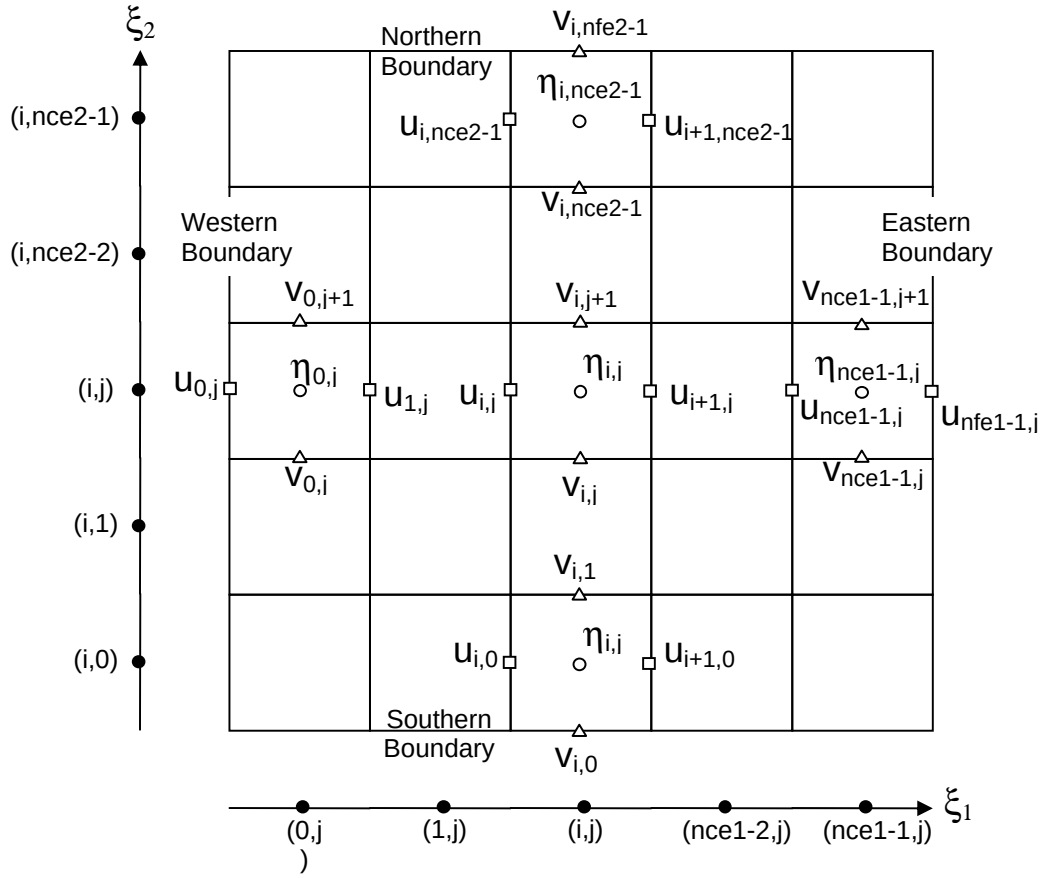
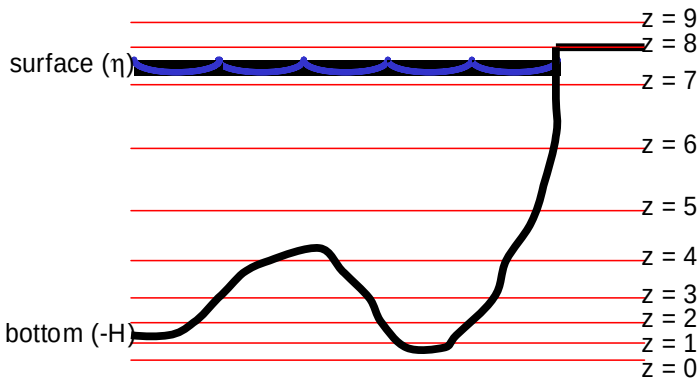
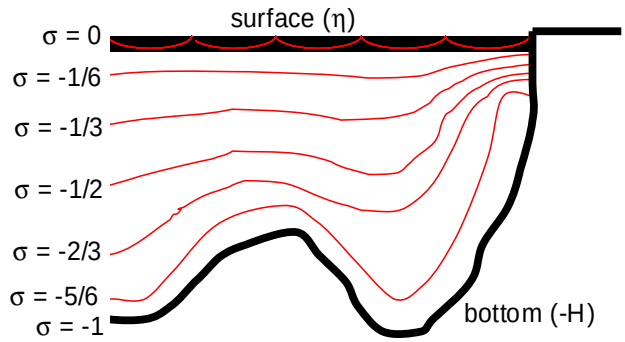


Figure 1.3 : Vertical coordinate systems



'z' Cartesian



Sigma

2. Model Equations.

2.1 Basic Equations

The equations of motion are transformed for curvilinear coordinates using Eqn. 1.1 and for the sigma system using 1.2, resulting in equations similar to those described in Blumberg and Herring (1987). The equations used in the 'z' model are presented here; the σ model follows the formulation of Blumberg and Herring (1987) and is not repeated here. All variables used in the equations of motion are listed below.

Table 2.1 : Model Variables

η	Surface elevation (m)
u_1	Velocity in the ξ_1 direction (ms^{-1})
u_2	Velocity in the ξ_2 direction (ms^{-1})
w	Velocity in the z direction (ms^{-1})
U_1	Vertically integrated velocity in the ξ_1 direction (ms^{-1})
U_2	Vertically integrated velocity in the ξ_2 direction (ms^{-1})
T	Temperature (^oC)
S	Salinity (psu)
H	Bathymetry (water depth) (m)
D	Total water depth ($H+\eta$) (m)
t	Time (s)
g	Acceleration due to gravity (ms^{-2})
f	Coriolis parameter (s^{-1})
P	Pressure (Pa)
ρ	Density (kgm^{-3})
ρ_o	Reference density (kgm^{-3})
C_D	Drag coefficient
C_{Dmin}	Minimum drag coefficient
κ	von Karman constant (0.4)
z_o	Bottom roughness length (m)
K_z	Vertical diffusivity (m^2s^{-1})
V_z	Vertical viscosity (m^2s^{-1})
V_h	Horizontal viscosity (m^2s^{-1})
z	Vertical coordinate (m)
h_1	Metric in the ξ_1 direction (m)
h_2	Metric in the ξ_2 direction (m)
τ_{sx}	Surface wind stress in the ξ_1 direction (Nm^{-2})
τ_{sy}	Surface wind stress in the ξ_2 direction (Nm^{-2})
τ_{bx}	Bottom stress in the ξ_1 direction (Nm^{-2})
τ_{by}	Bottom stress in the ξ_2 direction (Nm^{-2})
H_T	Surface heat flux ($ms^{-1}K$)
H_S	Surface salt flux ($ms^{-1}psu$)
w_{top}	Vertical velocity at the surface (ms^{-1})
u_{1top}	Velocity at the surface in the ξ_1 direction (ms^{-1})
u_{2top}	Velocity at the surface in the ξ_2 direction (ms^{-1})
w_{bot}	Vertical velocity at the bottom (ms^{-1})
u_{1bot}	Velocity at the bottom in the ξ_1 direction (ms^{-1})
u_{2bot}	Velocity at the bottom in the ξ_2 direction (ms^{-1})
ψ_1	Horizontal ξ_1 momentum diffusion term
ψ_2	Horizontal ξ_2 momentum diffusion term

Let the three dimensional vector of velocity have the components u_1 in the ξ_1 direction, u_2 in the ξ_2 direction and w in the z direction. Then:

$$u_1 = h_1 \frac{\partial \xi_1}{\partial t} \quad 2.1.1$$

$$u_2 = h_2 \frac{\partial \xi_2}{\partial t} \quad 2.1.2$$

$$w = \frac{\partial z}{\partial t} \quad 2.1.3$$

The horizontal momentum equations can be written as:

Continuity:

$$\frac{1}{h_1 h_2}\left[\frac{\partial u_1 h_2}{\partial \xi_1} + \frac{\partial u_2 h_1}{\partial \xi_2}\right] + \frac{\partial w}{\partial z} = 0 \quad 2.1.4$$

Momentum ξ_1 direction:

$$\frac{\partial u_1}{\partial t} + \frac{1}{h_1^2 h_2}\left[\frac{\partial (u_1^{\,2} h_1 h_2)}{\partial \xi_1} + \frac{\partial (u_1 u_2 h_1^{\,2})}{\partial \xi_2}\right] + \frac{\partial (w u_1)}{\partial z} = -\frac{1}{h_1 \rho}\frac{\partial P}{\partial \xi_1} + f u_2 + \frac{u_1^{\,2}}{h_1^2}\frac{\partial h_1}{\partial \xi_1} + \frac{u_2^{\,2}}{h_1 h_2}\frac{\partial h_2}{\partial \xi_1} + \psi_1 + \frac{\partial}{\partial z}\left[V_z \frac{\partial u_1}{\partial z}\right] \quad 2.1.5$$

Momentum ξ_2 direction:

$$\frac{\partial u_2}{\partial t} + \frac{1}{h_1 h_2^2}\left[\frac{\partial (u_1 u_2 h_2^2)}{\partial \xi_1} + \frac{\partial (u_2^2 h_1 h_2)}{\partial \xi_2}\right] + \frac{\partial (w u_2)}{\partial z} = -\frac{1}{h_2 \rho}\frac{\partial P}{\partial \xi_2} - f u_1 + \frac{u_1^{\,2}}{h_1 h_2}\frac{\partial h_1}{\partial \xi_2} + \frac{u_2^{\,2}}{h_2^2}\frac{\partial h_2}{\partial \xi_2} + \psi_2 + \frac{\partial}{\partial z}\left[V_z \frac{\partial u_2}{\partial z}\right] \quad 2.1.6$$

Tracers:

$$\frac{\partial T}{\partial t} + \frac{1}{h_1 h_2}\left[\frac{\partial (u_1 T h_2)}{\partial \xi_1} + \frac{\partial (u_2 T h_1)}{\partial \xi_2}\right] + \frac{\partial (w T)}{\partial z} = \frac{1}{h_1 h_2}\left[\frac{\partial}{\partial \xi_1}\frac{h_2}{h_1} V_h \frac{\partial T}{\partial \xi_1} + \frac{\partial}{\partial \xi_2}\frac{h_1}{h_2} V_h \frac{\partial T}{\partial \xi_2}\right] + \frac{\partial}{\partial z}\left[K_z \frac{\partial T}{\partial z}\right] \quad 2.1.7$$

Equation 2.1.7 is valid for temperature, salinity and any contaminants included. The vertical velocity is obtained from Eqn. 2.1.4.

2.2 Approximations

SHOC employs the hydrostatic approximation to simplify the vertical equation of motion. This relies on the fact that vertical accelerations in the ocean are generally small in comparison to the gravity acceleration and the vertical equation of motion is reduced to the hydrostatic law, Eqn 2.2.1.

$$\partial P = -\rho g \partial z \qquad 2.2.1$$

If density variations are small, the Boussinesq approximation simplifies density in the equations of motion, essentially replacing density in the equations with a mean density except for the buoyancy term (e.g. see Muller 1995, Gill 1982). Density is taken to be a constant when calculating rates of change of momentum due to accelerations, but using the actual density when calculating terms involving buoyancy. Also, it is assumed that the scale for variations of the vertical velocity are small in comparison to the scale of vertical variations in density. The scale for vertical variations in density is referred to as the scale height of the ocean, and is the depth at which density would increase by a factor of e=2.718 due to the weight of overlying water. This depth equates to approximately 200km, which is far greater than the depth of the ocean or the scale of vertical motions, therefore the density variations with depth are small; approximately 2% from a constant reference value of 1035 kgm^{-3} (Gill, 1982, p47). Assuming a constant density allows the full continuity equation;

$$\frac{\partial \rho}{\partial t} + \nabla \bullet \rho \vec{u} = \frac{D\rho}{Dt} + \rho \nabla \bullet \vec{u} = 0 \qquad 2.2.2$$

to be simplified to eqn 2.1.4, i.e. the ocean is assumed to be incompressible, where sound and shock waves are not resolved.
The equation of motion in the ξ_1 direction may be written as:

$$\rho(\frac{Du_1}{Dt} - fu_2) = \frac{\partial \rho u_1}{\partial t} + \nabla \bullet (\rho u \vec{u}) - f\rho u_2 = -\frac{1}{h_1}\frac{\partial P}{\partial \xi_1} + HOT \qquad 2.2.3$$

where HOT is the higher order frictional terms and the two equivalent terms on the left hand side are derived via eqn. 2.2.2 (e.g. Gill, 1982, p74). The Boussinesq approximation allows ρ to be replaced by the constant reference density, ρ_o, simplifying eqn. 2.2.3 to:

$$\rho_o \frac{Du_1}{Dt} - \rho_o f u_2 = -\frac{1}{h_1}\frac{\partial P}{\partial \xi_1} + HOT \qquad 2.2.4$$

2.3 Pressure Term

The hydrostatic equation, eqn. 2.2.1, can be integrated from any depth z to the surface, η, to give the pressure at the level z:

$$P = P_a + g\int_z^\eta \rho . dz \qquad 2.3.1$$

where now P refers to the pressure at level z and P_a is the atmospheric pressure. The derivative in the ξ_1 direction is then:

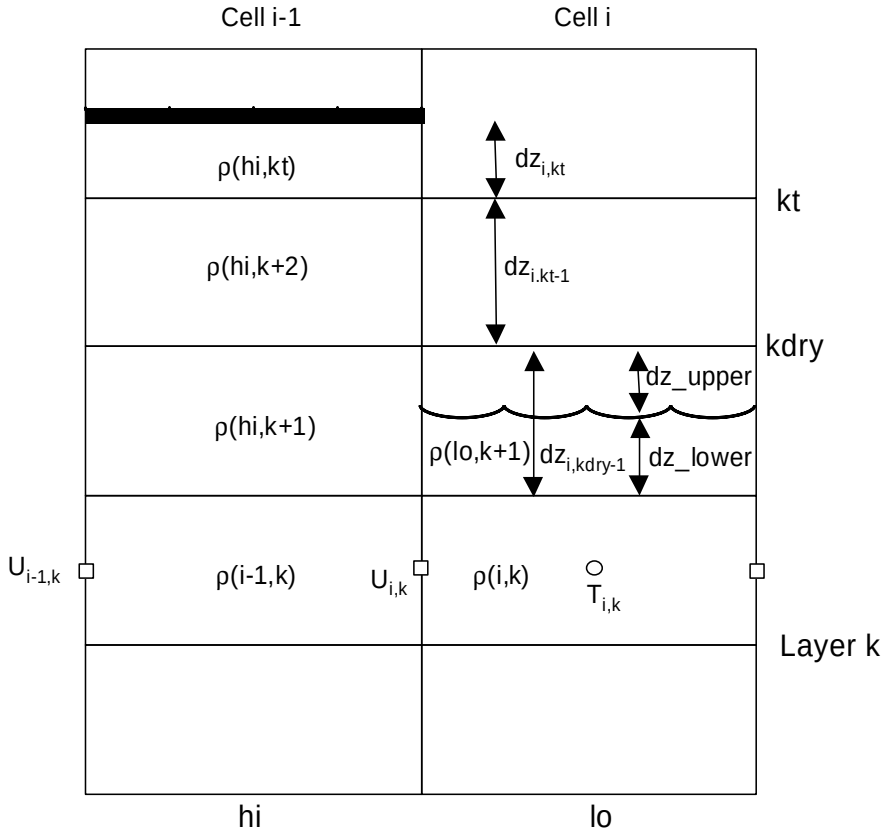
$$\frac{\partial P}{\partial \xi_1} = \frac{\partial P_a}{\partial \xi_1} + g\frac{\partial}{\partial \xi_1}\int_z^\eta \rho dz \qquad 2.3.2$$

using Leibnitz's Rule this can be expanded to:

$$\frac{\partial P}{\partial \xi_1} = \frac{\partial P_a}{\partial \xi_1} + g\rho_\eta \frac{\partial \eta}{\partial \xi_1} + g\int_z^\eta \frac{\partial \rho}{\partial \xi_1} dz \qquad 2.3.3$$

where ρ_η is the density at the surface. This is the formulation adopted in the 'z' version of SHOC, however, the surface gradient is not explicitly evaluated. Rather, this term enters the pressure balance through a summation of cell thickness above (and including) cell faces that are completely or partially dry. This approach is adopted to cope with wetting and drying in which the surface elevation in adjacent wet cells may be separated by many layers. This situation is illustrated in Figure 2.1.

Figure 2.1 : Vertical Grid Section of the Surface Layers



Where :
hi = water column containing the highest water level
lo = water column containing the lower water level
kt = layer containing the free surface of the higher level
kdry = layer above that containing the lower water level
dz_upper = distance from the lower water level to the layer above
dz_lower = distance from the lower water level to the layer below

The surface gradient for layers $kt \geq k \geq kdry$ is discretized as:

$$g\rho_\eta \frac{\partial \eta}{\partial \xi_1} = g\rho(hi,k)\sum_{kt}^{k} dz(k) \qquad 2.3.4$$

and for the layer k = kdry-1 as:

$$g\rho_\eta \frac{\partial \eta}{\partial \xi_1} = g\rho(hi,k)\sum_{kt}^{kdry} dz(k) + g\rho(hi,k)dz\_upper \qquad 2.3.5$$

For layers k < kdry the total pressure is given by;

$$g\rho_\eta \frac{\partial \eta}{\partial \xi_1} + g\int_z^\eta \frac{\partial \rho}{\partial \xi_1} dz = g\rho(hi,k)\sum_{kt}^{kdry} dz(k) + g\rho(hi,k)dz\_upper + g\sum_{kdry-1}^{k}(\rho_{i,k} - \rho_{i-1,k})dz \qquad 2.3.6$$

In the σ model the free surface term is explicitly calculated. Moreover, the density is decomposed into a constant and depth varying component:

$$\rho = \rho_o + \rho'(\xi_1, \xi_2, \sigma, t) \qquad 2.3.7$$

Substituting into 2.3.2 and transforming to σ coordinates (noting that $\partial\rho_o / \partial\xi_1 = 0$):

$$\frac{\partial P}{\partial \xi_1} = \frac{\partial P_a}{\partial \xi_1} + g\rho_o \frac{\partial \eta}{\partial \xi_1} + gD\int_\sigma^0 \frac{\partial \rho'}{\partial \xi_1} d\sigma + g\frac{\partial D}{\partial \xi_1}\int_\sigma^0 \sigma \frac{\partial \rho'}{\partial \sigma} d\sigma \qquad 2.3.8$$

where the last term takes into account the deviation of σ levels from geopotential surfaces.

2.4 Horizontal Diffusion

The horizontal diffusion terms in the momentum equations, ψ_1 and ψ_2, may be written as (see Apel (1987) p91-100 for derivation in Cartesian coordinates):

$$\psi_1 = \frac{1}{h_1 h_2}\left[\frac{\partial}{\partial \xi_1}(h_2\tau_{11}) + \frac{\partial}{\partial \xi_2}(h_1\tau_{21})\right] + \frac{1}{h_1 h_2}\left[\tau_{21}\frac{\partial h_1}{\partial \xi_2} - \tau_{22}\frac{\partial h_2}{\partial \xi_1}\right] \qquad 2.4.1$$

$$\psi_2 = \frac{1}{h_1 h_2}\left[\frac{\partial}{\partial \xi_1}(h_2\tau_{12}) + \frac{\partial}{\partial \xi_2}(h_1\tau_{22})\right] + \frac{1}{h_1 h_2}\left[\tau_{12}\frac{\partial h_2}{\partial \xi_1} - \tau_{11}\frac{\partial h_1}{\partial \xi_2}\right] \qquad 2.4.2$$

where the shear stress components are:

$$\tau_{11} = 2V_h\left[\frac{1}{h_1}\frac{\partial u_1}{\partial \xi_1} + \frac{u_2}{h_1 h_2}\frac{\partial h_1}{\partial \xi_2}\right] \qquad 2.4.3$$

$$\tau_{12} = \tau_{21} = V_h\left[\frac{h_1}{h_2}\frac{\partial}{\partial \xi_2}\left(\frac{u_1}{h_1}\right) + \frac{h_2}{h_1}\frac{\partial}{\partial \xi_1}\left(\frac{u_2}{h_2}\right)\right] \qquad 2.4.4$$

$$\tau_{22} = 2V_h\left[\frac{u_1}{h_1 h_2}\frac{\partial h_2}{\partial \xi_1} + \frac{1}{h_2}\frac{\partial u_2}{\partial \xi_2}\right] \quad 2.4.5$$

An option exists to invoke the Smagorinsky diffusion (Smagorinsky, 1963) for the horizontal viscosity and diffusivity, given by:

$$K_H = ch_1h_2\left[\tilde{\tau}_{11}^2 + \frac{\tilde{\tau}_{12}^2}{2} + \tilde{\tau}_{22}^2\right]^{\frac{1}{2}} \quad 2.4.6$$

where $\tilde{\tau}_{11} = \tau_{11}/(2V_h)$, $\tilde{\tau}_{22} = \tau_{22}/(2V_h)$, $\tilde{\tau}_{12} = \tau_{12}/V_h$ and c is a constant, typically c=0.1.

2.5 Boundary Conditions

The boundary conditions at the surface are:

$$\rho V_z \frac{\partial}{\partial z}(u_1, u_2) = (\tau_{sx}, \tau_{sy}) \quad 2.5.1$$

$$K_z \frac{\partial}{\partial z}(T, S) = (H_T, H_S) \quad 2.5.2$$

(Mellor 1996, Section 3) where expressions for H_T and H_S are given in Section 9, and

$$w_{top} = \frac{\partial \eta}{\partial t} + \frac{u_{1top}}{h_1}\frac{\partial \eta}{\partial \xi_1} + \frac{u_{2top}}{h_2}\frac{\partial \eta}{\partial \xi_2} \quad 2.5.3$$

The surface wind stress is given by:

$$\begin{aligned}\tau_{sx} &= \rho_a C_{DS}(W_1^2 + W_2^2)^{1/2} W_1 \\ \tau_{sy} &= \rho_a C_{DS}(W_1^2 + W_2^2)^{1/2} W_2\end{aligned} \quad 2.5.4$$

where ρ_a is the air density (~1.225 kgm^{-3}), W_1 and W_2 are the wind speeds in the ξ_1 and ξ_2 directions respectively and C_{DS} is the surface drag coefficient. SHOC uses the drag formulation of Large and Pond (1981) which has the form:

$$C_{DS} = \begin{cases} C_{D0} & W \le W_a \\ C_{D0} + (C_{D1} - C_{D0})\dfrac{W - W_0}{W_1 - W_0} & W_a < W < W_b \\ C_{D1} & W \ge W_b \end{cases} \quad 2.5.5$$

where C_{D0} (~0.00114), C_{D1} (~0.00218), W_a (~10.0 ms^{-1}) and W_b (~26.0 ms^{-1}) are specified parameters.

The boundary conditions at the bottom are:

$$\rho V_z \frac{\partial}{\partial z}(u_1, u_2) = (\tau_{bx}, \tau_{by}) \quad 2.5.6$$

$$\rho K_z \frac{\partial}{\partial z}(T, S) = 0 \quad 2.5.7$$

$$w_{bot} = -\frac{u_{1bot}}{h_1}\frac{\partial H}{\partial \xi_1} - \frac{u_{2bot}}{h_2}\frac{\partial H}{\partial \xi_2} \quad 2.5.8$$

The bottom stress is given by the quadratic friction law:

$$\tau_{bx} = \rho C_D (u_1^2 + u_2^2)^{1/2} u_1 \quad 2.5.9$$

$$\tau_{by} = \rho C_D (u_1^2 + u_2^2)^{1/2} u_2 \quad 2.5.10$$

where the drag coefficient is given by:

$$C_D = \max\left(\left[\frac{1}{\kappa}\ln(\frac{z + z_o}{z_o})\right]^{-2}, C_{D\min}\right) \quad 2.5.11$$

Here z denotes the distance above the sea floor; numerically this is implemented as the first grid point above the bottom. C_{Dmin} is a minimum drag coefficient (typically between 0.002 and 0.003) used when the first grid point is a long way from the bottom (i.e. for large z).

2.6 Vertically Integrated Equations

The solutions to the equations of motion contain both fast moving surface gravity waves and slower moving internal waves. It is advantageous from a computational efficiency perspective to separate these two types of motion, allowing the fast moving waves to be solved on a smaller time-step. This is accomplished by vertically integrating eqn. 2.1.5 and 2.1.6 to give equations for the velocity transport (called the 2D mode or barotropic mode) which are then used via the vertical integral of eqn. 2.1.4 to provide the surface elevation. This technique is known as mode splitting (Simons 1974).
The vertical integration of eqn. 2.1.5 and 2.1.6 removes all vertical structure and is achieved for a variable x as:

$$\overline{x} = \frac{1}{H + \eta}\int_{-H}^{\eta} x dz = \frac{1}{D}\int_{-H}^{\eta} x dz \quad 2.6.1$$

Eqn. 2.1.4 subject to vertical integration yields (using 2.5.3 and 2.5.8):

$$\frac{\partial \eta}{\partial t} + \frac{1}{h_1 h_2}\left[\frac{\partial D U_1 h_2}{\partial \xi_1} + \frac{\partial D U_2 h_1}{\partial \xi_2}\right] = 0 \quad 2.6.2$$

The vertical integrals of eqns. 2.1.5 and 2.1.6 give:

$$\frac{\partial U_1}{\partial t}+\frac{1}{Dh_1^2h_2}\left[\frac{\partial DU_1^2h_1h_2}{\partial \xi_1}+\frac{\partial DU_1U_2h_1^2}{\partial \xi_2}\right]=fU_2-\frac{1}{h_1\rho_{av}}\left[\frac{\partial P_a}{\partial \xi_1}+g\rho_{top}\frac{\partial \eta}{\partial \xi_1}\right]-\frac{1}{D}\int_{-H}^{kdry}\frac{g}{h_1\rho}\int_z^\eta\frac{\partial \rho}{\partial \xi_1}dz'dz$$
$$+\frac{\tau_{sx}}{D\rho_{top}}-\frac{\tau_{bx}}{D\rho_{bot}}+D\overline{\psi}_1$$
$$+\frac{U_1^2}{h_1^2}\frac{\partial h_1}{\partial \xi_1}+\frac{U_2^2}{h_1h_2}\frac{\partial h_2}{\partial \xi_1}-\frac{U_1}{D}\frac{\partial \eta}{\partial t}$$
$$-\frac{1}{Dh_1^2h_2}\left[\frac{\partial D\overline{u_1'^2}h_1h_2}{\partial \xi_1}+\frac{\partial D\overline{u_1'u_2'}h_1^2}{\partial \xi_2}\right]+\frac{\overline{u_1'^2}}{h_1^2}\frac{\partial h_1}{\partial \xi_1}+\frac{\overline{u_2'^2}}{h_1h_2}\frac{\partial h_2}{\partial \xi_1}$$

2.6.3

$$\frac{\partial U_2}{\partial t}+\frac{1}{Dh_2^2h_1}\left[\frac{\partial DU_1U_2h_1h_2}{\partial \xi_1}+\frac{\partial DU_2^2h_2^2}{\partial \xi_2}\right]=-fU_1-\frac{1}{h_2\rho_{av}}\left[\frac{\partial P_a}{\partial \xi_2}+g\rho_{top}\frac{\partial \eta}{\partial \xi_2}\right]-\frac{1}{D}\int_{-H}^{kdry}\frac{g}{h_2\rho}\int_z^\eta\frac{\partial \rho}{\partial \xi_2}dz'dz$$
$$+\frac{\tau_{sy}}{D\rho_{top}}-\frac{\tau_{by}}{D\rho_{bot}}+D\overline{\psi}_2$$
$$+\frac{U_1^2}{h_1h_2}\frac{\partial h_1}{\partial \xi_2}+\frac{U_2^2}{h_2^2}\frac{\partial h_2}{\partial \xi_2}-\frac{U_2}{D}\frac{\partial \eta}{\partial t}$$
$$-\frac{1}{Dh_2^2h_1}\left[\frac{\partial D\overline{u_1'u_2'}h_1h_2}{\partial \xi_1}+\frac{\partial D\overline{u_2'^2}h_2^2}{\partial \xi_2}\right]+\frac{\overline{u_1'^2}}{h_1h_2}\frac{\partial h_1}{\partial \xi_2}+\frac{\overline{u_2'^2}}{h_2^2}\frac{\partial h_2}{\partial \xi_2}$$

2.6.4

where

$$u_1'=u_1-U_1$$
$$u_2'=u_2-U_2$$

2.6.5

Note that the vertical integral of the time derivative can be written as:

$$\begin{aligned}\int_{-H}^{\eta}\frac{\partial u_1}{\partial t}dz&=\frac{\partial}{\partial t}\int_{-H}^{\eta}u_1dz-u_1(\eta)\frac{\partial \eta}{\partial t}-u_1(-H)\frac{\partial(-H)}{\partial t}\\&=\frac{\partial}{\partial t}\left[(H+\eta)U_1\right]-u_{1top}\frac{\partial \eta}{\partial t}\\&=D\frac{\partial U_1}{\partial t}+U_1\frac{\partial \eta}{\partial t}+U_1\frac{\partial H}{\partial t}-u_{1top}\frac{\partial \eta}{\partial t}\\&=D\frac{\partial U_1}{\partial t}+U_1\frac{\partial \eta}{\partial t}-u_{1top}\frac{\partial \eta}{\partial t}\end{aligned}$$

2.6.6

Also, using Leibnitz's Rule and eqn. 2.6.1 the vertical integral of the horizontal advective terms is:

$$\frac{1}{h_1^2 h_2}\left[\int_{-H}^{\eta}\frac{\partial u_1^2 h_1 h_2}{\partial \xi_1}dz+\int_{-H}^{\eta}\frac{\partial u_1 u_2 h_1^2}{\partial \xi_1}dz\right]=\frac{1}{h_1^2 h_2}\frac{\partial(D\overline{u_1^2}h_1 h_2)}{\partial \xi_1}+\frac{u_{1top}^2}{h_1}\frac{\partial \eta}{\partial \xi_1}+\frac{u_{1bot}^2}{h_1}\frac{\partial H}{\partial \xi_1}$$
$$+\frac{1}{h_1^2 h_2}\frac{\partial(D\overline{u_1 u_2}h_1^2)}{\partial \xi_2}+\frac{u_{1top}u_{2top}}{h_2}\frac{\partial \eta}{\partial \xi_2}+\frac{u_{1top}u_{2top}}{h_2}\frac{\partial H}{\partial \xi_1} \quad 2.6.7$$

and the vertical integral of the vertical advective term:

$$\int_{-H}^{\eta}\frac{\partial w u_1}{\partial z}dz=wu_1\Big|_{-H}^{\eta}=w_{top}u_{1top}-w_{bot}u_{1bot} \quad 2.6.8$$

The term $u_{1top}\partial\eta/\partial t$ in eqn. 2.6.6 combines with those on the RHS of eqn. 2.6.7 involving u_{1top} and u_{1bot} to cancel the RHS of eqn. 2.6.8 via the use of eqns. 2.5.3 and 2.5.8.
The horizontal diffusion terms, $D\overline{\psi}_1$ and $D\overline{\psi}_2$ are obtained from 2.4.1 and 2.4.2 where u_1 is substituted with DU_1 and u_2 with DU_2. Specifically:

$$\begin{aligned}D\overline{\psi}_1=&\frac{1}{h_1h_2}\frac{\partial}{\partial\xi_1}\left[\frac{2V_h h_2}{h_1}\frac{\partial(DU_1)}{\partial\xi_1}+\frac{2V_h DU_2}{h_1}\frac{\partial h_1}{\partial\xi_2}\right]\\&+\frac{1}{h_1h_2}\frac{\partial}{\partial\xi_2}\left[\frac{V_h h_1^2}{h_2}\frac{\partial}{\partial\xi_2}\left(\frac{DU_1}{h_1}\right)+V_h h_2\frac{\partial}{\partial\xi_2}\left(\frac{DU_2}{h_2}\right)\right]\\&+\frac{V_h}{h_1h_2}\frac{\partial h_1}{\partial\xi_2}\left[\frac{h_1}{h_2}\frac{\partial}{\partial\xi_2}\left(\frac{DU_1}{h_1}\right)+\frac{h_2}{h_1}\frac{\partial}{\partial\xi_1}\left(\frac{DU_2}{h_2}\right)\right]\\&-\frac{V_h}{h_1h_2}\frac{\partial h_2}{\partial\xi_1}\left[2DU_1\frac{\partial h_2}{\partial\xi_1}+\frac{1}{h_2}\frac{\partial(DU_2)}{\partial\xi_2}\right]\end{aligned} \quad 2.6.9$$

$$\begin{aligned}D\overline{\psi}_2=&\frac{1}{h_1h_2}\frac{\partial}{\partial\xi_1}\left[V_h h_1\frac{\partial(DU_1)}{\partial\xi_1}+\frac{V_h h_2^2}{h_1}\frac{\partial}{\partial\xi_1}\left(\frac{DU_2}{h_2}\right)\right]\\&+\frac{1}{h_1h_2}\frac{\partial}{\partial\xi_2}\left[\frac{2V_h DU_1}{h_2}\frac{\partial h_2}{\partial\xi_1}+\frac{2V_h h_1}{h_2}\frac{\partial(DU_2)}{\partial\xi_2}\right]\\&+\frac{V_h}{h_1h_2}\frac{\partial h_2}{\partial\xi_1}\left[\frac{h_1}{h_2}\frac{\partial}{\partial\xi_2}\left(\frac{DU_1}{h_1}\right)+\frac{h_2}{h_1}\frac{\partial}{\partial\xi_1}\left(\frac{DU_2}{h_2}\right)\right]\\&-\frac{V_h}{h_1h_2}\frac{\partial h_1}{\partial\xi_2}\left[2DU_2\frac{\partial h_1}{\partial\xi_2}+\frac{1}{h_1}\frac{\partial(DU_1)}{\partial\xi_1}\right]\end{aligned} \quad 2.6.10$$

The last line of eqns 2.6.3 and 2.6.4 represent the advection of the fluctuating velocity and are called the dispersion terms. Rather than evaluate these explicitly eqn. 2.6.5 is used and the fluctuating velocity is equal to the vertical average of the 3D velocity advection minus the 2D velocity advection.

$$\frac{\partial D\overline{u_1'^2}h_1h_2}{\partial\xi_1}+\frac{\partial D\overline{u_1'u_2'}h_1^2}{\partial\xi_2}=\frac{\partial D\overline{u_1^2}h_1h_2}{\partial\xi_1}+\frac{\partial D\overline{u_1u_2}h_1^2}{\partial\xi_2}-\frac{\partial DU_1^2h_1h_2}{\partial\xi_1}-\frac{\partial DU_1U_2h_1^2}{\partial\xi_2} \quad 2.6.11$$

The bottom stress, vertical integral of momentum advection and the pressure gradient integral are calculated prior to the 2D mode and held constant throughout the 2D time integrations. The 2D mode in turn provides the surface elevation for use in the 3D mode. After the 2D mode is complete, the 3D velocities are adjusted so that their vertical integral is equal to the 2D velocities. It is possible to execute SHOC in 2D mode only. This essentially only solves the vertically integrated equations to provide velocity transports and surface elevation, leading to large savings in execution time.

2.7 Stability Considerations

The mode splitting approach results in separate stability constraints for the 2D and 3D modes governed by the fastest respective wave speeds of these modes. These conditions basically prohibit a perturbation moving at a speed equal to the wave speed plus the maximum advective speed from crossing one grid cell in one time-step. For the barotropic mode the Courant-Friedrichs-Levy (CFL) stability condition is:

$$\Delta t_{2D}\le\frac{1}{C_g}\left[\frac{1}{h_1^2}+\frac{1}{h_2^2}\right]^{-\frac{1}{2}} \quad 2.7.1$$

where:

$$C_g=2\sqrt{gH}+U_{\max} \quad 2.7.2$$

is the sum of the shallow water gravity wave speed plus the maximum velocity transport expected (U_{max}). It is usually good practice to reduce this time-step amount by a ‘safety factor’ of around 0.9 when setting up the model. The CFL condition for the baroclinic mode is less restrictive by virtue of the slower speeds of the internal waves, and is given by:

$$\Delta t_{3D}\le\frac{1}{C_i}\left[\frac{1}{h_1^2}+\frac{1}{h_2^2}\right]^{-\frac{1}{2}} \quad 2.7.3$$

where $C_i = 2c_i + u_{max}$. For a 2 layer system where the upper layer has a thickness of H_1 and density of ρ_1, and the lower layer thickness is H_2 with density ρ_2, the shallow water ($\Lambda/ H_1>20$ and $\Lambda/ H_2>20$; Λ = wavelength) internal wave speed can be estimated via (Pond and Pickard, 1983, p238):

$$c_i=\sqrt{\frac{g(\rho_2-\rho_1)}{\rho_2}\frac{H_1H_2}{H_1+H_2}} \quad 2.7.4$$

For the case where a thin upper layer ($H_1< \Lambda/20$) overlies a deep lower layer ($H_2>\Lambda/2$) this can be approximated as:

$$c_i=\sqrt{\frac{(\rho_2-\rho_1)}{\rho_1}gH_1} \quad 2.7.5$$

These waves are non-dispersive; deep water internal waves (Λ/ H_1<2 and Λ/ H_2<2) have speeds which are dependent on the wave number hence are dispersive, i.e.

$$c_i = \sqrt{\frac{g(\rho_2 - \rho_1)}{k(\rho_2 + \rho_1)}} \qquad 2.7.6$$

where $k = 2\pi / \Lambda$ is the radian wave number. Note that deep water internal wave speeds are independent of the water depth. The internal wave speed in a continuously stratified ocean assumes a more complicated form. Assuming that the horizontal scale is larger than the vertical scale and wave motion is confined to the oceanic waveguide (i.e. confined between the free surface and bottom boundaries) the internal wave can be decomposed into a set of discrete modes. The horizontal phase speed for the case of continuous stratification with constant Brunt-Vaisala frequency, $N = (-g\rho_o^{-1} d\rho_o / dz)^{1/2}$, in a flat bottomed ocean for long (shallow water) waves is given by (Gill, 1982, p159):

$$c_i = \frac{NH}{n\pi} = -\frac{gH}{\rho_o n\pi}\frac{\partial \rho_o}{\partial z} \quad n = 1,2,3... \qquad 2.7.7$$

The fastest typical internal wave speeds for the ocean are 2 to 3 ms^{-1}.
The horizontal diffusion term imposes a stability constraint that is typically quite unrestrictive, given by:

$$\Delta t_{diff} \le \frac{1}{4V_h}\left[\frac{1}{h_1^2} + \frac{1}{h_2^2}\right]^{-1} \qquad 2.7.8$$

Note that an implicit scheme is used to solve the vertical diffusion and these schemes are unconditionally stable. This approach overcomes stability violations that may possibly occur with the vertical diffusion if fine vertical resolution is used. Finally the Coriolis term has an associated stability constraint that is also unrestrictive in comparison to 2.7.1 and 2.7.3.

$$\Delta t_{cor} \le \frac{1}{f} \qquad 2.7.9$$

2.8 Time Stepping

SHOC uses leapfrog time stepping for momentum and Euler forward time-stepping for tracers. The leapfrog time derivative is second order accurate and (in a simple one dimensional advection equation) is approximated as:

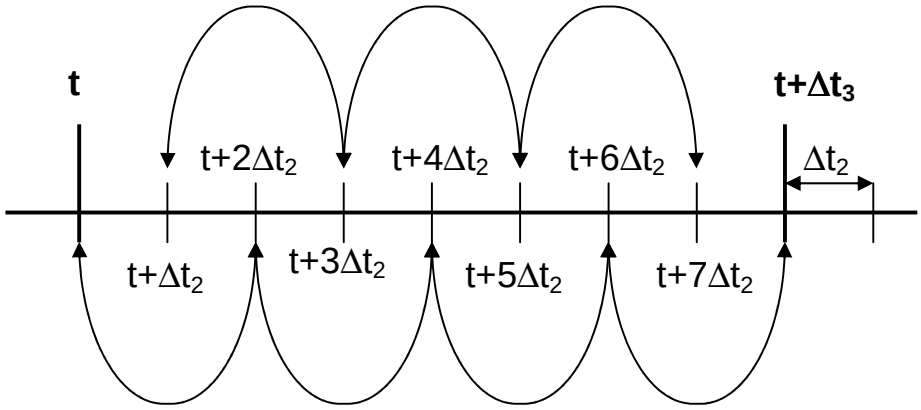
$$c^{t+1} = c^{t-1} + 2u\Delta t\frac{\partial c^t}{\partial x} \qquad 2.8.1$$

Solutions to the leapfrog scheme contain contributions representing physical (related to the differential equation solved) and computational (related to the structure of the numerical equation) modes. The latter alternates on every timestep and is an undesirable artifact of the leapfrog scheme. This computational mode can, however, be removed by the intermittent application of the Euler forward scheme or the use of a filter such as that of Asselin (1972):

$$\bar{c}^t = c^t + 0.5\upsilon(\bar{c}^{t-1} - 2c^t + c^{t+1}) \qquad 2.8.2$$

where υ is a constant (typically 0.1).
Compatibility between the 2D and 3D modes of the model, and hence conservation, is achieved by integrating the 2D fluxes over the 3D time steps and adjusting the 3D velocities once the 2D step is complete so that the vertical integral of the 3D fluxes is equal the time-integrated 2D fluxes. This ensures that the change in sea level over the 3D time-step (as computed by the 2D equations) is equal to the total divergence of 3D fluxes above any given layer (see below). The leapfrog method makes this summation over the 2D step difficult. Consider Figure 2.2 where the 3D mode operates on a time-step Δt_3 and the 2D mode has a time-step Δt_2, with $\Delta t_3 = I_R \Delta t_3$; in this case I_R=8. The leapfrog scheme advances any variable by $2\Delta t_2$ using the values of the variable at time t- Δt_2 and t according to Equation 2.8.1. This time-stepping is represented by the arrows in Figure 2.2.

Figure 2.2 : Leapfrog Time Stepping



The 2D momentum fluxes must be integrated from time t to t+ Δt_3 according to:

$$U_{flux} = \sum_{t}^{t+\Delta t_3} U(t)\left[H + \eta(t)\right] h \Delta T \qquad 2.8.3$$

where U can be U_1 or U_2, h=h_2 for U_1, h=h_1 for U_2 and ΔT is a time interval. If consistency (and thus conservation) is to be maintained between the 2D and 3D modes, then:

$$U_{flux} = u_{flux} = \int_{-H}^{\eta} u(t + \Delta t_3) h \Delta t_3 dz \qquad 2.8.4$$

where u can be u_1 or u_2. Using Eqn. 2.1.4 and 2.5.3 with Leibenitz's rule and noting $\partial z / \partial \xi = 0$;

$$\frac{\partial \eta}{\partial t} = w_z - \frac{1}{h_1 h_2}\left[\frac{\partial}{\partial \xi_1}\int_{-z}^{\eta} u_1 h_2 dz + \frac{\partial}{\partial \xi_2}\int_{-z}^{\eta} u_2 h_1 dz\right] \qquad 2.8.5$$

where w_z is the vertical velocity through layer z (i.e. the volume change in a water column over the time period Δt_3 is equal to the divergence of the vertical integral of the flux above that layer, including the contribution of vertical fluxes – the volume balance for the layer thickness). This may be expressed as:

$$\begin{aligned}[\eta(t+\Delta t_3)-\eta(t)]h_1h_2 &= h_1h_2\Delta t_3 w_z - \frac{\partial}{\partial \xi_1}\int_{-z}^{\eta} u_1 h_2 \Delta t_3 dz - \frac{\partial}{\partial \xi_2}\int_{-z}^{\eta} u_2 h_1 \Delta t_3 dz \\ &= h_1h_2\Delta t_3 w_z - \nabla\cdot\int_{-z}^{\eta}\tilde{u}_{flux}\end{aligned} \qquad 2.8.6$$

with $\tilde{u}_{flux} = u(t+\Delta t_3)h\Delta t_3 dz$. Eqn. 2.8.6 relates the vertical velocities to the change in sea level, and if this is satisfied then conservation in the tracer equations is achieved. If z = H and w_{-H}=0 (no flow through the bottom) then Eqn. 2.8.6 becomes:

$$\eta(t+\Delta t_3) = \eta(t) - \frac{\nabla\cdot u_{flux}}{h_1h_2} \qquad 2.8.7$$

If Eqn. 2.8.4 holds then Eqn. 2.8.7 reverts to the continuity equation for the 2D mode (Eqn. 2.6.2). It is clear that for Eqn. 2.8.4 to hold the 2D fluxes must be integrated in time such that $\Delta T=2\Delta t_2$ on only every 2nd step of the leapfrog (i.e. the summation of fluxes takes place on only those times depicted by the bottom sequence of arrows in Figure 2.2). This is only possible if I_R is an even number, hence the leapfrog scheme is limited by the condition that the ratio of Δt_3:Δt_2 must be even.

The Euler forward time-stepping derivative (in a simple one dimensional advection equation) is approximated as:

$$c^{t+1} = c^t + u\Delta t\frac{\partial c^t}{\partial x} \qquad 2.8.1$$

for some variable c. This is a first order Taylor approximation which is explicit in time since the function $\partial c/\partial t$ is evaluated at the time level t (as opposed to t+1). The Euler scheme is unstable (Kowalik and Murty 1993, p163) and requires the presence of some diffusion to stabilize it (Rood 1987). This scheme was chosen for tracers since an array containing tracer values at the backward time level is not required, leading to large savings in memory when many tracers are simulated.

2.9 2D Vorticity Equation

Using the continuity equation, 2.6.2, the material derivative of the vertically integrated ξ_1 equation can be written as (in Cartesian notation for convenience):

$$\begin{aligned}&\frac{\partial U_1}{\partial t} + \frac{1}{D}\left[\frac{\partial DU_1^2}{\partial x} + \frac{\partial DU_1U_2}{\partial y}\right] + \frac{U_1}{D}\frac{\partial \eta}{\partial t} \\ &\quad= \frac{\partial U_1}{\partial t} + \frac{1}{D}\left[U_1\frac{\partial DU_1}{\partial x} + DU_1\frac{\partial U_1}{\partial x} + U_1\frac{\partial DU_2}{\partial y} + DU_2\frac{\partial U_1}{\partial y}\right] + \frac{U_1}{D}\frac{\partial \eta}{\partial t} \\ &\quad= \frac{\partial U_1}{\partial t} + \frac{U_1}{D}\left[\frac{\partial \eta}{\partial t} + \frac{\partial DU_1}{\partial x} + \frac{\partial DU_2}{\partial y}\right] + U_1\frac{\partial U_1}{\partial x} + U_2\frac{\partial U_1}{\partial y} \\ &\quad= \frac{\partial U_1}{\partial t} + U_1\frac{\partial U_1}{\partial x} + U_2\frac{\partial U_1}{\partial y}\end{aligned} \qquad 2.9.1$$

The vertical integral of the pressure gradient in the x direction may be written as (Slϕrdal and Weber, 1996)

$$\int_{-H}^{\eta} \frac{\partial P}{\partial x} dz = D\frac{\partial P_a}{\partial x} + gH\frac{\partial}{\partial x}\int_{-H`}^{\eta} \rho dz + g\frac{\partial}{\partial x}\int_{-H`}^{\eta} \rho z dz \qquad 2.9.2$$

If the elevation is small in comparison to the water depth, eqn. 2.9.2 may be written as:

$$\int_{-H}^{\eta} \frac{\partial P}{\partial x} dz = D\frac{\partial P_a}{\partial x} + gH\rho_\eta \frac{\partial \eta}{\partial x} + \rho_o H\frac{\partial \phi}{\partial x} + \rho_o \frac{\partial \chi}{\partial x} \qquad 2.9.3$$

where χ is the anomaly of potential energy (e.g. Rattray, 1982, Mertz and Wright, 1992) :

$$\chi = \frac{g}{\rho_o}\int_{-H}^{0} dz \int_{-H}^{z} \rho dz = \frac{g}{\rho_o}\int_{-H}^{0} z\rho dz \qquad 2.9.4$$

and ϕ is contribution to the bottom pressure by the vertical mass distribution.

$$\phi = \frac{g}{\rho_o}\int_{-H}^{0} \rho dz \qquad 2.9.5$$

The vertically integrated equations may then be written (using the Boussinesq approximation 2.2.4):

$$\frac{\partial U_1}{\partial t} + U_1\frac{\partial U_1}{\partial x} + U_2\frac{\partial U_1}{\partial x} - fU_2 = -\frac{1}{\rho_o}\frac{\partial P_a}{\partial x} - g\frac{\partial \eta}{\partial x} - \frac{\partial \phi}{\partial x} - \frac{1}{D}\frac{\partial \chi}{\partial x} + \frac{1}{D\rho_o}(\tau_{sx} - \tau_{bx}) \qquad 2.9.6$$

$$\frac{\partial U_2}{\partial t} + U_1\frac{\partial U_2}{\partial x} + U_2\frac{\partial U_2}{\partial x} + fU_1 = -\frac{1}{\rho_o}\frac{\partial P_a}{\partial y} - g\frac{\partial \eta}{\partial y} - \frac{\partial \phi}{\partial y} - \frac{1}{D}\frac{\partial \chi}{\partial y} + \frac{1}{D\rho_o}(\tau_{sy} - \tau_{by}) \qquad 2.9.7$$

Taking the curl of 2.9.6 and 2.9.7 (i.e. $\partial(2.9.7)/\partial x - \partial(2.9.6)/\partial y$ d) yield the 2D vorticity equation. Noting that the relative vorticity of the depth averaged flow, $\zeta = \partial U_2/\partial x - \partial U_1/\partial y$, the LHS becomes:

$$LHS = \frac{\partial}{\partial x}\left[\frac{\partial U_2}{\partial t} + U_1\frac{\partial U_2}{\partial x} + U_2\frac{\partial U_2}{\partial x} + fU_1\right] - \frac{\partial}{\partial y}\left[\frac{\partial U_1}{\partial t} + U_1\frac{\partial U_1}{\partial x} + U_2\frac{\partial U_1}{\partial x} - fU_2\right]$$

$$= \frac{\partial}{\partial t}\left[\frac{\partial U_2}{\partial x} - \frac{\partial U_1}{\partial y}\right] + \frac{\partial U_1}{\partial x}\frac{\partial U_2}{\partial x} + U_1\frac{\partial^2 U_2}{\partial x^2} + \frac{\partial U_2}{\partial x}\frac{\partial U_2}{\partial y} + U_2\frac{\partial^2 U_2}{\partial x \partial y} + f\frac{\partial U_1}{\partial x}$$

$$-\frac{\partial U_1}{\partial y}\frac{\partial U_1}{\partial x} - U_1\frac{\partial^2 U_1}{\partial x \partial y} - \frac{\partial U_2}{\partial y}\frac{\partial U_1}{\partial y} - U_2\frac{\partial^2 U_1}{\partial y^2} + f\frac{\partial U_2}{\partial y} + \beta U_2$$

$$= \frac{\partial \zeta}{\partial t} + U_1\frac{\partial}{\partial x}\left[\frac{\partial U_2}{\partial x} - \frac{\partial U_1}{\partial y}\right] + U_2\frac{\partial}{\partial y}\left[\frac{\partial U_2}{\partial x} - \frac{\partial U_1}{\partial y}\right] + (\frac{\partial U_2}{\partial x} - \frac{\partial U_1}{\partial y})(\frac{\partial U_1}{\partial x} + \frac{\partial U_2}{\partial y}) + f(\frac{\partial U_1}{\partial x} + \frac{\partial U_2}{\partial y}) + \beta U_2$$

$$= \frac{\partial \zeta}{\partial t} + U_1\frac{\partial \zeta}{\partial x} + +U_2\frac{\partial \zeta}{\partial y} + (f+\zeta)(\frac{\partial U_1}{\partial x} + \frac{\partial U_2}{\partial y}) + \beta U_2$$

2.9.8

Using the continuity equation again for the divergence term:

$$LHS = \frac{\partial \zeta}{\partial t} + \vec{U} \bullet \nabla\zeta - \frac{(f+\zeta)}{D}(\frac{\partial \eta}{\partial t} + \vec{U} \bullet \nabla D) + \beta U_2 \qquad 2.9.9$$

The RHS becomes:

$$RHS = -\frac{\partial^2}{\partial x \partial y}\left[\frac{P_a}{\rho_o} + g\eta + \phi\right] - \frac{\partial}{\partial x}\left[\frac{1}{D}\frac{\partial \chi}{\partial y}\right] + \frac{\partial}{\partial x}\left[\frac{1}{D\rho_o}(\tau_{sy} - \tau_{by})\right] +$$

$$\frac{\partial^2}{\partial x \partial y}\left[\frac{P_a}{\rho_o} + g\eta + \phi\right] + \frac{\partial}{\partial y}\left[\frac{1}{D}\frac{\partial \chi}{\partial x}\right] - \frac{\partial}{\partial y}\left[\frac{1}{D\rho_o}(\tau_{sx} - \tau_{bx})\right]$$

$$= -\frac{\partial D^{-1}}{\partial x}\frac{\partial \chi}{\partial y} + \frac{\partial D^{-1}}{\partial y}\frac{\partial \chi}{\partial x} + \frac{1}{D}\left[\frac{\partial^2 \chi}{\partial y \partial x} - \frac{\partial^2 \chi}{\partial x \partial y}\right] + \frac{1}{\rho_o}\left[\frac{\partial \tau_{sy}/D}{\partial x} - \frac{\partial \tau_{sx}/D}{\partial y} - \frac{\partial \tau_{by}/D}{\partial x} + \frac{\partial \tau_{bx}/D}{\partial y}\right]$$

$$= J(\chi, D^{-1}) + \frac{1}{\rho_o} curl_z(\frac{\bar{\tau}_s}{D} - \frac{\bar{\tau}_s}{D})$$

2.9.10

where the Jacobian is defined as $J(A,B) = \frac{\partial A}{\partial x}\frac{\partial B}{\partial y} - \frac{\partial A}{\partial y}\frac{\partial B}{\partial x}$. The depth averaged vorticity equation then becomes:

$$\frac{\partial \zeta}{\partial t} + \vec{U} \bullet \nabla\zeta + \beta U_2 = \frac{(f+\zeta)}{D}(\frac{\partial \eta}{\partial t} + \vec{U} \bullet \nabla D) + J(\chi, D^{-1}) + \frac{1}{\rho_o} curl_z(\frac{\bar{\tau}_s}{D} - \frac{\bar{\tau}_s}{D}) \qquad 2.9.11$$

PRV ADV BETA TF/H JEBAR WSC BSC

Where
PRV = Rate of change of relative vorticity (production of relative vorticity),
ADV = advection of relative vorticity,

BETA = transport across contours of constant planetary vorticity,
TF/H = topographic vortex stretching, i.e. transport across f/H contours,
JEBAR = Joint Effect of Baroclinicity And Relief is the contribution of the mass field to vorticity production (Mertz and Wright, 1992),
WSC = production of vorticity due to wind stress curl and the interaction of the wind stress with the gradient of topography,
BSC = dissipation of vorticity due to bottom stress curl and the interaction of the bottom stress with the gradient of topography.
Equation 2.9.11 provides a useful diagnostic for interpreting motion in the ocean, such that forcing mechanisms for the development and decay of eddies or circulation may be investigated (e.g. Middleton et al, 2001, Dippner, 1998). Equation 2.9.11 is often expressed in terms of the barotropic streamfunction in the literature.
Note that the last line of eqn. 2.9.8 may be written:

$$\frac{\partial \zeta}{\partial t} + U_1 \frac{\partial \zeta}{\partial x} + + U_2 \frac{\partial \zeta}{\partial y} + (f + \zeta)(\frac{\partial U_1}{\partial x} + \frac{\partial U_2}{\partial y}) + \beta U_2 = \frac{D(\zeta + f)}{Dt} + (f + \zeta)(\frac{\partial U_1}{\partial x} + \frac{\partial U_2}{\partial y}) \qquad 2.9.12$$

since $\partial f / \partial t = \partial f / \partial x = 0$ and $\beta = \partial f / \partial y$. Thus for a homogeneous inviscid ocean, eqn. 2.9.11 may be written:

$$\frac{1}{(f + \zeta)} \frac{d(\zeta + f)}{dt} = -(\nabla \bullet \vec{U}) \qquad 2.9.13$$

where $d / dt = \partial / \partial t + \vec{U} \bullet \nabla$ is the material derivative. The continuity equation 2.6.2 may be written in a similar form:

$$\frac{1}{D} \frac{dD}{dt} = \frac{1}{(\eta + H)} \frac{d(\eta + H)}{dt} = -(\nabla \bullet \vec{U}) \qquad 2.9.14$$

Subtraction of 2.9.13 from 2.9.14 to remove the divergence term yields the conservation of potential vorticity equation (e.g. see Apel, 1987, p270-282, Gill, 1982, p232):

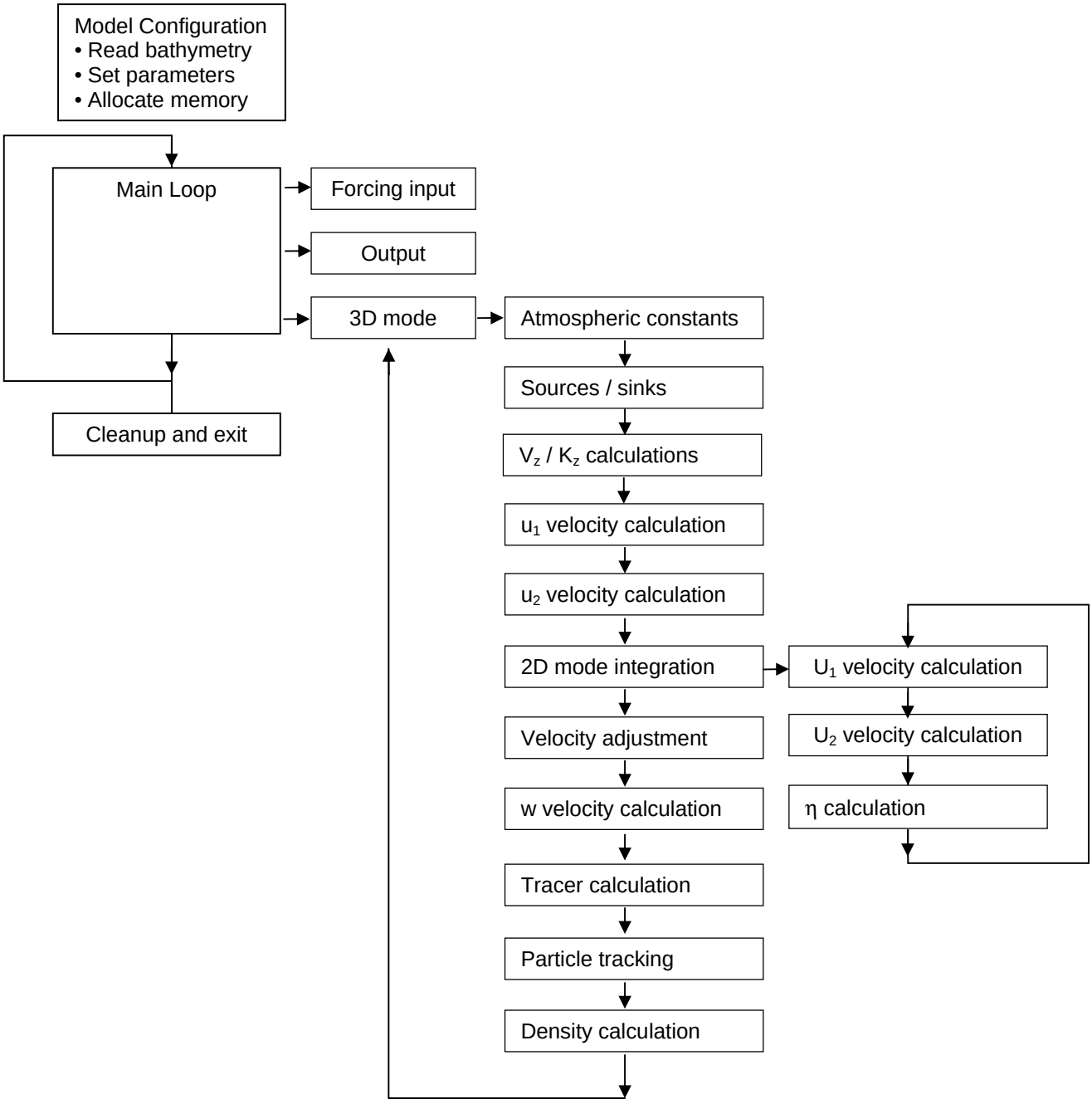
$$\frac{d}{dt}\left(\frac{f + \zeta}{D} \right) = 0 \qquad \text{or} \qquad \Pi = \frac{f + \zeta}{D} = \text{constant} \qquad 2.9.15$$

where Π is the potential vorticity.

3. Flow of Control.

A schematic of the program flow is illustrated in Figure 3.1.

Figure 3.1 : Flow Diagram of SHOC



SHOC first undergoes initialisation and configuration procedures where the input file is read and constants are initialised, arrays are allocated and initialised, forcing functions are initialised and the grid is established. The main time loop is then entered where a scheduler will spawn routines

at defined, but arbitrary, time intervals. One of these is the integration of the 3D mode (and also the 2D mode since this is embedded in the 3D mode). First wind, surface pressure and surface flux calculations are performed followed by any source or sink calculations. The turbulence closure is then invoked providing the vertical viscosity and diffusivity parameters followed by the solution to the 3D momentum equations. The 2D mode is then invoked, cycling many times on the smaller time-step to provide the velocity transports and surface elevation. The bottom stress, vertical integral of momentum advection and the pressure gradient integral are supplied from the 3D mode and held constant throughout the 2D mode loop. After the 2D mode is complete the computed surface elevation is used in subsequent tracer and 3D mode calculations. The 3D velocities are then adjusted so that vertical integrals are equal to 2D velocities. Vertical velocity is then calculated via the continuity equation followed by the solution to the conservation equations for tracers. Particle tracking is then invoked if required and finally the density is computed. The main loop is then re-entered to wait for the scheduler to spawn the next event. When no more events remain then memory is de-allocated and the program terminates.

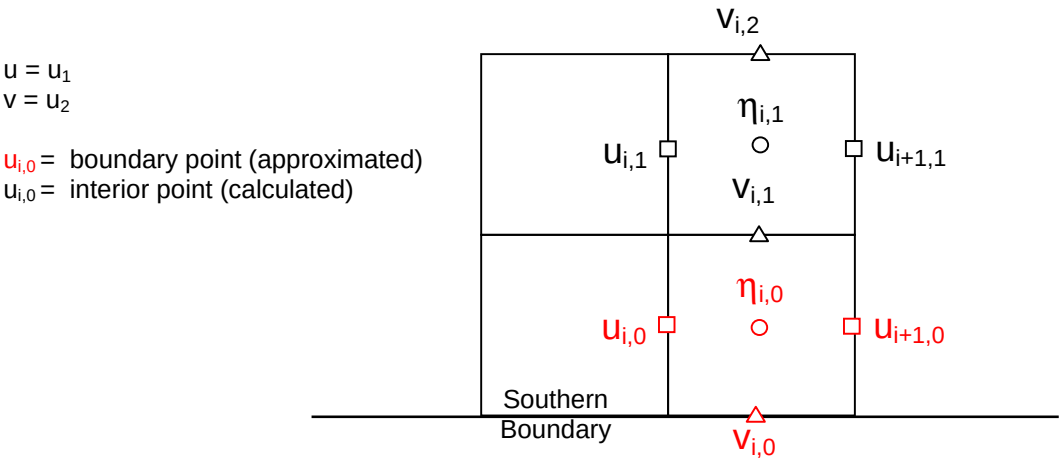
4. Open Boundary Conditions.

4.1 Introduction

Version 1 of SHOC contains a range of open boundary conditions (OBCs) including radiation, extrapolation, sponge and direct data forcing methods. Open boundaries can be passive, where any signals generated inside the domain are allowed to propagate through the open boundary (Stevens, 1990). In practice passive open boundary conditions are usually associated with some degree of reflection or distortion of the signal. Alternatively the open boundary may be active, where velocity or elevation is prescribed on the open boundary and may influence the solution in the model interior. Forcing with an elevation on the open boundary (which constitutes a clamped, or Dirichlet condition) may not be successful since this may not account for the contribution to boundary elevation from the response of the model interior (e.g. a resonant response of a domain forced with synthesized tides). Also the prescription of elevation data on the boundary does not capture the transient response of the model interior (e.g. to a time dependent wind forcing) and often a combination of passive and active open boundary conditions is required (Blumberg and Kantha, 1985).

The model requires that two components of velocity be prescribed on each open boundary (normal and tangential velocities to the boundary) for both the 3D and 2D modes. Surface elevation and the values of any tracers present are also required. The grid configuration in the vicinity of an open boundary may use the normal velocity on the outer faces, e.g. Figure 4.1 (for a southern boundary in this case) or the inner faces as in Figure 4.2.

Figure 4.1 : Southern Boundary Grid Configuration or Outer Stagger



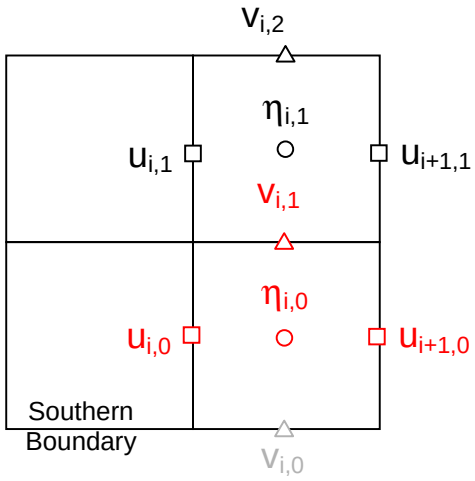
In these cases the u_2 velocity corresponds to the normal velocity and the u_1 velocity to the tangential velocity. The boundary configuration of Fig. 4.1 limits any error in the normal boundary velocity to the non-linear terms only (i.e. normal velocity boundary approximations (v_{i0}) are not used in the tangential velocity Coriolis calculation, u_{i1}). This allows the diffusive terms to damp any oscillatory behavior the normal velocity may develop and helps to maintain stability (Herzfeld and Tomczak, 1997). The velocities are fully non-linear at the boundary and can be used in both passive and active forcing capacities (for either elevation or velocity). 2D velocity open boundaries are updated on the 2D time step. This type of stagger was used by Stevens (1990). Note that this stagger requires an extra cell at the grid limits (northern and eastern boundaries) to accommodate the normal velocity at these cell faces (i.e. **v**(i,nfe2-1) and **u**(nfe1-1,j) respectively in Figure 1.3).

The inner stagger is the default configuration for POM type models. In this case normal velocity approximated on the boundary is communicated into the interior via the tangential velocity Coriolis term. Using the inner stagger the boundary elevation (e.g. $\eta_{i,0}$) is only used in the non-linear advective terms via the total depth, D=η-H. Therefore, if H >> η, any prescription of η at this boundary (e.g. tidal forcing) will not adequately propagate the prescribed sea level into the domain. If h is required to be prescribed at the boundary, then the only way to accomplish this is via a Flather condition combined with the local solution. The Flather condition is generally unsuccessful with the outer stagger (see Palma and Matano (2001, p120) and Palma and Matano (1998, p1340)), however, the outer stagger may be forced directly with η without the requirement of imposing Flather. Note that different solutions result when using the same boundary conditions applied to identical problems with outer and inner staggers, due to the terms in the momentum equations that are recipients of boundary approximations. The outer stagger is generally considered more stable.

Figure 4.2 : Southern Boundary Grid Configuration for Inner Stagger

$u = u_1$
$v = u_2$

$u_{i,0}$ = boundary point (approximated)
$u_{i,0}$ = interior point (calculated)
$v_{i,0}$ = boundary point (unused)



4.2 Boundary Condition Types

The boundary conditions available are based on a variety of approaches and are listed in Table 4.1. The name and general class of condition are listed, along with a reference to the original study if this exists. The variables the condition may be applied to are also included, where un = normal velocity, ut = tangential velocity, η = surface elevation and T = tracers.

Table 4.1 : SHOC Open Boundary Conditions

Condition name	Type	Reference	Variable
Clamped	Clamped	-	un,ut,η,T
Data prescription from file	Clamped	-	un,η,T
Custom data prescription	Clamped	-	un,η,T
Tidal synthesis	Clamped	Bye (1988)	η
3D vertical integral for 2D	Clamped	-	un,ut
No-gradient	Extrapolation	-	un,ut,η,T
Linear least squares	Extrapolation	-	un,ut,η,T
2^{nd} order polynomial	Extrapolation	-	un,ut,η,T
Cyclic	Extrapolation	-	un,ut,η,T
Statistical prescription	Extrapolation	-	T
Linear	Extrapolation	-	un,ut
Gravity wave radiation	Radiation	Sommerfeld (1949)	un,ut,η,T
Orlanski	Radiation	Orlanski (1976)	un,ut,η,T
Camerlengo and O'Brien	Radiation	Camerlengo & O'Brien (1980)	un,ut,η,T
Miller and Thorpe	Radiation	Miller and Thorpe (1981)	un,ut,η,T
Raymond and Kuo	Radiation	Raymond and Kuo (1984)	un,ut,η,T
Flather	Radiation	Flather (1976)	un (2d)
Upstream advection	Upstream advection	-	T
Tidal memory	Clamped / upstream advection	-	T
Flow relaxation	Relaxation	Martinsen & Engedahl (1987)	un,ut,η,T
Sponge	Sponge	Israeli and Orszag (1981)	un,ut

A detailed discussion of the performance of these boundary conditions is not presented here; see Chapman (1985), Roed and Cooper (1987), Palma and Matano (1998, 2001) or Jensen (1998) for evaluations of these conditions.

4.3 Relaxation to Forced Data

Many of the boundary conditions listed above are based on the Sommerfeld radiation condition, which for some variable ϕ takes the one dimensional form of:

$$\frac{\partial \phi}{\partial t} + c\frac{\partial \phi}{\partial n} = 0 \qquad 4.3.1$$

where c is the phase speed of disturbances at the boundary and n is the coordinate normal to the boundary. Many of the open boundary conditions listed above differ only in their treatment of the phase speed. Boundary data specified from a file may be combined with a radiation condition so that the transient response of the domain is transmitted through the boundary while allowing the boundary to respond to the prescribed forcing, i.e. the following is solved on the boundary (Blumberg and Kantha, 1985);

$$\frac{\partial \phi}{\partial t} + c\frac{\partial \phi}{\partial n} = -\frac{\phi - \phi_D}{T_f} \qquad 4.3.2$$

where n is the coordinate normal to the boundary, c is the phase speed of disturbances at the boundary, ϕ_D is the prescribed variable and T_f is the user defined time scale of relaxation to the prescribed values. As mentioned above, this condition is only functional using the outer stagger.

4.4 Clamped Boundary Conditions

Clamped open boundary conditions impose a specified value on the open boundary. This value may be any value deemed appropriate and may be derived from field data, analytical calculations or even zero values. Supported options are outlined below.

4.4.1 Clamped

The clamped condition requires the user to supply a single time and space independent value to be imposed on the boundary. This type of condition corresponds to c=0 in eqn 4.3.1 and is a very reflective condition. For elevation and velocities SHOC uses a clamped value of zero, and is useful if there are to be zero tangential velocities on the boundary (i.e. all flow into the domain occurs at right angles to the boundary). Tracers are clamped to a fill value specified in the parameter file.

4.4.2 Data prescription from file

This boundary condition requires the user to supply data stored in a file (usually in netCDF format) which is interpolated onto the open boundary. This allows boundary variables to vary temporally and spatially. An example of the implementation of this condition may be a nested simulation that uses output from a previous simulation of a larger domain encompassing the nested domain.

4.4.3 Custom data prescription

Additional code may be included in SHOC which is executed by specifying the custom OBC, and the clamped value is calculated from this additional code during the simulation. This is typically invoked for application specific scenarios where an analytical condition is required on the open boundary. An example of this may be a depth dependent velocity profile representing a river discharge. This type of condition may be time and space varying.

4.4.4 Tidal synthesis for elevation

The elevation at an open boundary may be synthesized from tidal harmonics following the formulation of Bye (1988):

$$\eta = \sum_k A_k \cos\frac{2\pi t}{T_k} - d_j(\alpha_k \sin\theta_k \cos\frac{2\pi t}{T_k} + A_k\beta_k \sin\phi_k \sin\frac{2\pi t}{T_k}) - d_i(\alpha_k \cos\theta_k \cos\frac{2\pi t}{T_k} + A_k\beta_k \cos\phi_k \sin\frac{2\pi t}{T_k}) \quad 4.4.1$$

where for each tidal constituent k, A is the tidal amplitude at a given location (i_c,j_c), T is the tidal period at location (i_c,j_c), α is the rate of modulation of tidal amplitude in cm/km, θ is the direction towards which the tidal amplitude is progressing in ^{o}T, β is the rate of modulation of tidal phase in

degrees/km, and ϕ is the direction towards which the tidal phase is progressing in ^{o}T. The distance from the cell (i_c,j_c) to the boundary cell at j along the ξ_2 axis is d_j and the distance from the cell (i_c,j_c) to the boundary cell at i along the ξ_1 axis is d_i. The quantities α, β, θ and ϕ can be obtained from cotidal charts which display co-phase and co-range contours for a particular constituent. Note that this specification of the tide allows the tide to be calculated over the entire boundary, thus alleviating the necessity to prescribe the tidal elevation at each boundary cell. This method is, however, only applicable when there exists a linear variation of co-range or co-phase contours over the boundary.

4.4.5 3D vertical integral for 2D velocity

This condition simply vertically integrates the 3D velocities to obtain the depth averaged velocity on the open boundary and clamps the 2D open boundary velocity values to these averages. This condition imposes a time invariant velocity field on the boundary throughout the 2D mode.

4.5 Extrapolation Conditions

Extrapolation conditions use data at cells interior to the open boundary to derive a value applicable to the boundary. Supported extrapolation OBC's are as follows.

4.5.1 No-gradient condition

The no-gradient OBC assumes that there does not exist a gradient of the variable across the open boundary and is sometimes referred to as a Neumann boundary condition. This condition corresponds to setting $c = \infty$ in eqn. 4.3.1 and is specified by setting the boundary value equal to the value immediately adjacent to the boundary in the interior, i.e.

$$\phi_B^{t+1} = \phi_{B\pm1}^{t+1} \tag{4.5.1}$$

where ϕ_B^{t+1} is the boundary value at the forward time-step, ϕ_{B+1}^{t+1} is the interior value for western and southern boundaries and ϕ_{B-1}^{t+1} is the interior value for eastern and northern boundaries.

4.5.2 Linear least squares

The boundary value is specified using a linear extrapolation from the first 4 cells interior to the boundary using a least squares method. If n=4 and the origin is assumed to be 4 cells into the interior such that $(x_i,\phi_i^{t+1}) = (1,\phi_{B-4}^{t+1})$, $(2,\phi_{B-3}^{t+1})$, $(3,\phi_{B-2}^{t+1})$ and $(4,\phi_{B-4}^{t+1})$ for i=1:4 then:

$$\phi_B^{t+1} = a_0 + 5a_1 \tag{4.5.2}$$

where:

$$a_1 = \frac{\sum_{i=1}^{n} x_i\phi_i - \frac{1}{n}\sum_{i=1}^{n} x_i \sum_{i=1}^{n} \phi_i}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)^2} \tag{4.5.3}$$

and

$$a_0 = \tfrac{1}{n}\sum_{i=1}^{n}\phi_i - \frac{a_1}{n}\sum_{i=1}^{n}x_i \qquad 4.5.4$$

4.5.3 2nd order polynomial extrapolation

This method calculates the boundary value using a unique interpolating polynomial of 2nd order based on the 3 interior cells via Neville's algorithm (Press et al 1992, p 102).

4.5.4 Cyclic

The cyclic, or periodic, OBC presumes that two opposing boundaries (e.g. north and south, or east and west) have the same dimension and effectively sets a boundary condition such that the domain becomes infinite in the direction of the opposing boundaries. This is accomplished by considering a certain boundary and prescribing the values of the opposing boundary on that boundary. Hence what goes out one end comes in the other and vice versa, i.e. the boundaries wrap around and become cyclic. This can be useful to produce an infinite coast or a domain surrounded by an infinite ocean.

4.5.5 Statistical prescription

The statistical prescription for tracers on open boundaries relies on defining a sub-region at the boundary over which the tracer is sampled and some form of statistic (e.g. mean, median, nth percentile) is calculated. The sub-region is defined on the basis of the magnitude of the normal velocity component at the boundary. Starting at the boundary and progressing into the interior, cells are checked until a cell is located where the difference in normal velocity between that cell and the boundary cell becomes more than a specified threshold. The location of this cell becomes the limit of the sub-region. The threshold is currently given as a fraction of the normal velocity at the boundary, i.e. a cell u_S is located where;

$$\left|u_S - u_B\right| > \left|uf.u_B\right| \qquad 4.5.5$$

where u_B is the normal velocity at the boundary and uf is the threshold fraction. The statistic calculated over the sub-region may be one of the following:

Last value at the limit of the sub-region (cyclic over the sub-region)
Mean value of the sub-region
Minimum value of the sub-region
Maximum value over the sub-region
Any percentile value over the subregion

The choice of uf and the sub-region statistic are typically derived on a trial and error basis to produce smooth, apparently continuous tracer distributions near open boundaries. The statistical prescription method is useful to prescribe unknown tracer values on the boundary for non-conservative tracers (i.e. tracers whose values are not dominated by advection / diffusion) and can be tailored to minimise discontinuities at the boundaries sometimes introduced by no-gradient or clamped open boundary conditions.

4.5.6 Linear conditions

It is possible to omit the advective and horizontal diffusive terms on the boundary and thus linearize the boundary momentum balance. However, for normal velocities this is not particularly useful since using the stagger depicted in Fig. 4.1 the pressure gradients for v_{i0} will be zero as there are no values of η, T or S beyond the boundaries. A better approach is to shift the stagger for normal velocities one cell into the interior and linearize the normal velocity at this location, effectively making velocity v_{i1} the boundary cell. The velocity v_{i0} is then effectively not used in calculation and may be set to anything, typically a no-gradient condition (unless no action is taken for the tangential velocity, then this point is used in non-linear terms). This conforms to the inner stagger depicted in Figure 4.2.

The non-linear terms may be further omitted for a certain number of cells interior to the open boundary to improve stability (e.g. Kowalik and Murty (1993), p186 suggest 3 cells interior to the boundary).

4.6 Radiation Conditions

These OBCs all use the formulation of eqn. 4.3.1 using different values of the phase speed, c. The implementation of most of these schemes uses an implicit approach. Supported conditions are listed below.

4.6.1 Gravity wave radiation.

This formulation uses a fixed flat bottomed barotropic shallow water wave speed as the phase velocity, i.e:

$$c = \sqrt{gD_B} \tag{4.6.1}$$

where D_B is the depth at the boundary. The OBC is implemented in an implicit form such that:

$$\phi_B^{t+1} = \frac{\phi_B^t + \mu\phi_{B\pm1}^{t+1}}{1+\mu} \tag{4.6.2}$$

where:

$$\mu = c\frac{\Delta t}{h} \tag{4.6.3}$$

The solution to eqn. 4.3.2 using the gravity wave radiation scheme in implicit form is given by:

$$\phi_B^{t+1} = \frac{\phi_B^t + \mu\phi_{B\pm1}^{t+1}}{1+\mu} + \frac{\Delta t}{1+\mu}\frac{\phi_D - \phi_B^t}{T_f} \tag{4.6.4}$$

4.6.2 Orlanski radiation

The Orlanski radiation condition computes the phase speed of disturbances approaching the boundary at every time-step from the distribution of the interior values near the boundary, e.g.

$$c = -\frac{\partial\phi/\partial t}{\partial\phi/\partial x} \tag{4.6.5}$$

The Orlanski radiation coefficient has a theoretical reflection coefficient of zero. The form employed by SHOC is the implicit formulation:

$$\phi_B^{t+1} = \frac{\phi_B^{t-1}(1-\mu) + 2\mu\phi_{B\pm1}^{t}}{1+\mu} \tag{4.6.6}$$

with:

$$\mu = \begin{cases} 1 & if \quad C \geq 1 \\ C & if \quad 0 < C < 1 \\ 0 & if \quad C \leq 0 \end{cases} \tag{4.6.7}$$

where:

$$C = \frac{\phi_{B\pm1}^{t-1} - \phi_{B\pm1}^{t+1}}{\phi_{B\pm1}^{t+1} + \phi_{B\pm1}^{t-1} - 2\phi_{B\pm2}^{t}} \tag{4.6.8}$$

4.6.3 Camerlengo and O'Brien

This OBC is a modified form of the Orlanski radiation condition where only the extreme values of the phase speed, zero or h/Δt, is used, so that;

$$\phi_B^{t+1} = \begin{cases} \phi_{B\pm1}^{n} & if \quad C > 0 \\ \phi_B^{n-1} & if \quad C \leq 0 \end{cases} \tag{4.6.9}$$

with C given by eqn 4.6.8. The solution to eqn. 4.3.2 using the Orlanski or Camerlengo and O'Brien radiation schemes in implicit form is given by:

$$\phi_B^{t+1} = \frac{\phi_B^{t-1}(1-\mu) + 2\mu\phi_{B\pm1}^{t}}{1+\mu} + \frac{2\Delta t}{1+\mu}\frac{\phi_D - \phi_B^{t}}{T_f} \tag{4.6.10}$$

4.6.4 Miller and Thorpe

The Orlanski scheme is modified here so that time differences are evaluated using a forward scheme and space differences with an upwind scheme (Miller and Thorpe 1981, eqn 15):

$$\phi_B^{t+1} = \phi_B^t - \mu(\phi_B^t - \phi_{B\pm1}^t) \qquad 6.4.11$$

with $\mu=\mu_1+\mu_2+\mu_3$ where:

$$\mu_1 = \frac{\phi_{B\pm1}^{t+1} - \phi_{B\pm1}^t}{\phi_{B\pm2}^t - \phi_{B\pm1}^t}, \qquad \mu_2 = \frac{\phi_B^t - \phi_B^{t-1}}{\phi_{B\pm1}^{t-1} - \phi_B^{t-1}}, \qquad \mu_2 = \frac{\phi_{B\pm1}^t - \phi_{B\pm1}^{t-1}}{\phi_{B\pm2}^{t-1} - \phi_{B\pm1}^{t-1}} \qquad 6.4.12$$

This scheme is implemented in explicit form. The solution to eqn. 4.3.2 using Miller and Thorpe's scheme in explicit form is given by:

$$\phi_B^{t+1} = \phi_B^t - \mu(\phi_B^t - \phi_{B\pm1}^t) + \Delta t \frac{\phi_D - \phi_B^t}{T_f} \qquad 4.6.13$$

4.6.5 Raymond and Kuo

The Raymond and Kuo (1984) scheme calculates the phase velocity for multidimensional flows using a projection of each coordinate direction, i.e. not just the normal component. The adaptive approach of Marchesiello et al (2001) is implemented in SHOC. The Sommerfeld condition takes the form:

$$\frac{\partial \phi}{\partial t} + c_x \frac{\partial \phi}{\partial x} + c_y \frac{\partial \phi}{\partial y} = 0 \qquad 4.6.14$$

where x and y are directions normal and tangential to the boundary respectively. The phase speeds c_x and c_y are projections given by:

$$\begin{aligned} c_x &= -\frac{\partial \phi}{\partial t} \frac{\partial \phi / \partial x}{(\partial \phi / \partial x)^2 + (\partial \phi / \partial y)^2} \\ c_y &= -\frac{\partial \phi}{\partial t} \frac{\partial \phi / \partial y}{(\partial \phi / \partial x)^2 + (\partial \phi / \partial y)^2} \end{aligned} \qquad 4.6.15$$

This is discretised following Marchesiello et al (2001);

$$\begin{aligned} \phi_{B,j}^{t+1} &= \frac{1}{1+r_x}\left[\phi_{B,j}^t + r_x \phi_{B-1,j}^{t+1} - r_y(\phi_{B,j}^t - \phi_{B,j-1}^t)\right] \qquad r_y > 0 \\ &= \frac{1}{1+r_x}\left[\phi_{B,j}^t + r_x \phi_{B-1,j}^{t+1} - r_y(\phi_{B,j+1}^t - \phi_{B,j}^t)\right] \qquad r_y < 0 \end{aligned} \qquad 4.6.16$$

where:

$$\begin{aligned} r_x &= -\frac{\Delta\phi_t \Delta\phi_x}{\Delta\phi_x^2 + \Delta\phi_y^2} \\ r_y &= -\frac{\Delta\phi_t \Delta\phi_y}{\Delta\phi_x^2 + \Delta\phi_y^2} \end{aligned} \qquad 4.6.17$$

$$\Delta\phi_t = \phi^{t+1}_{B-1,j} - \phi^{t}_{B-1,j}$$
$$\Delta\phi_x = \phi^{t+1}_{B-1,j} - \phi^{t+1}_{B-2,j} \qquad 4.6.18$$

$$\Delta\phi_y = \phi^{t}_{B-1,j} - \phi^{t}_{B-1,j-1} \qquad if\left[\Delta\phi_t(\phi^{t}_{B-1,j+1} - \phi^{t}_{B-1,j-1})\right] > 0$$
$$\Delta\phi_y = \phi^{t}_{B-1,j+1} - \phi^{t}_{B-1,j} \qquad if\left[\Delta\phi_t(\phi^{t}_{B-1,j+1} - \phi^{t}_{B-1,j-1})\right] < 0 \qquad 4.6.19$$

The adaptive for takes on a form similar to Eqn. 4.3.2:

$$\frac{\partial\phi}{\partial t} + c_x\frac{\partial\phi}{\partial x} + c_y\frac{\partial\phi}{\partial y} = -\frac{\phi - \phi_D}{T_f} \qquad 4.6.20$$

where $T_f = T_{out}$ if $c_x > 0$ and $T_f = T_{in}$ with $c_x = c_y = 0$ if $c_x < 0$. The relaxation time scale $T_{out} >> T_{in}$ such that during outward phase propagation a weak relaxation exists to avoid boundary values drifting excessively but also preventing problems of over-specification, while during inward phase propagation stronger relaxation is applied that avoids shock issues. SHOC uses an outward relaxation time-scale of T_{out} = 1 year.

4.6.6 Flather Radiation

This radiation condition is based on Flather (1976), used by Flather (1988) and evaluated by Palma and Matano (1998 and 2001). This condition combines the Sommerfeld radiation condition, eqn. 4.3.1, with a 1-dimensional continuity equation and applies the result to normal depth averaged velocity. Mass is conserved in the interior with this scheme and transients are allowed to propagate out of the domain with the gravity wave speed. The scheme is described by:

$$U_B = U_o(t) \pm \sqrt{\frac{g}{D_B}}\left[\eta_{BI} - \eta_o(t)\right] \qquad 4.6.21$$

where $U_o(t)$ and $\eta_o(t)$ are prescribed normal depth averaged velocity and elevation respectively and `BI=B` for west/south boundaries and `BI=B-1` for east/north boundaries (`B` = normal velocity location). This formulation allows forcing such as tides to be introduced through the boundary. If $U_o(t)$ and $\eta_o(t)$ = 0 then the condition behaves in a passive manner. This condition is most successful using an inner stagger, e.g. Palma and Matano (2001, p120) and Palma and Matano (1998, p1340). and Palma and Matano (1998) suggest gravity wave radiation conditions on tangential velocity and η when using the Flather condition.

The Flather condition is not particularly useful in many situations where the user does not have information for $U_o(t)$ with which to force the model. In these cases, a local solution may be used for $U_o(t)$ (Palma and Matano (1998, p1323), following the methodology of Roed and Smedstad (1984)). SHOC may use the linearized version of the momentum equations for the local solution.

4.7 Flow Relaxation Scheme

The flow relaxation scheme of Martinsen and Engedahl (1987) has been included to relax boundary data to interior data. This is accomplished over a region NN cells wide (typically NN=10) where the prognostic variables (η, u_1, u_2 or tracers) are updated according to:

$$\phi = \alpha_i \phi_B + (1 - \alpha_i)\phi_{B\pm i} \qquad 4.7.1$$

where ϕ_B is the boundary specified value, $\phi_{B\pm i}$ are the interior variable values and α_ι is a relaxation parameter given by:

$$\alpha_i = 1 - \tanh[(i-1)/2] \qquad i = 1,2,3....NN \qquad 4.7.2$$

Note that the flow relaxation scheme is used in conjunction with another boundary condition and ϕ_B may be obtained from any condition in Table 4.1; whatever is specified on the boundary is relaxed to the model integrated values over NN cells. If ϕ_B is equal to zero (clamped boundary condition) then this flow relaxation scheme acts as a sponge type condition.

4.8 Upstream Advection for Tracers

The upstream advection condition may be thought of as an eqn. 4.3.1 with c equal to the normal velocity at the boundary and must be used in conjunction with another boundary condition. For a southern boundary this may be discretized as:

$$\phi_B^{t+1} = \phi_B^t + \frac{\Delta t}{h_2}\left[(u_2 - |u_2|)(\phi_{B-1}^t - \phi_B^t) + (u_2 + |u_2|)(\phi_B^t - \phi_D^t)\right] \qquad 4.8.1$$

where ϕ_B is the tracer on the southern boundary, ϕ_{B-1} is the tracer one cell into the interior and ϕ_D is a tracer value that must be supplied via another boundary condition. Usually ϕ_D is supplied via data input from file, but may be any clamped or extrapolation type condition. This value is only used when flow is into the domain. If flow is out of the domain only the tracer values ϕ_B and ϕ_{B-1} are used in the boundary condition.

4.9 Tidal Memory for Tracers

The tidal memory open boundary condition is supplied for tracers that experience oscillatory behavior across the boundary, e.g. tidal forcing. When flow is directed outward across the boundary (ebb tide in an estuary) the tracer concentrations when the tide first begins to ebb and first begins to flood are ‘remembered’ and reapplied as the open boundary condition during the phase of subsequent flow inward across the boundary (flood tide). Boundary value concentrations are specified as a linear interpolation between the first and last ebb values. If the tidal excursion during the flood becomes greater than that of the previous ebb (i.e. an extrapolation is required from the first and last ebb values) then the last ebb concentration is used for the remainder of the flood. (An option exists to prescribe a user supplied tracer concentration in these circumstances). During the ebb phase an upstream advection open boundary condition is always applied. The tidal memory boundary condition is only invoked above a certain specified depth.

4.10 Sponge Schemes

A sponge condition acts to supplement an existing velocity OBC and acts to increase friction and thus damp any velocity perturbations near the boundary. These conditions are typically implemented over a number of cells (typically 10) adjacent to the boundary. Friction may be increased by linearly increasing bottom friction (Israeli and Orszag, 1981) up to 4 times the interior value, or by increasing horizontal viscosity (to the maximum value allowed to maintain stability).

4.11 Tidal Harmonics

Given a latitude and longitude on the earth's surface, SHOC can generate up to 22 tidal amplitude and phase harmonics representing long period, diurnal and semi-diurnal tidal oscillations. The tidal harmonics are generated from Geosat altimeter observations at 1^o latitude and 1.5^o longitude resolution using 'orthotide' functions (Cartwright and Ray, 1990). The elevation at latitude θ, longitude λ and time t is given by:

$$\eta(\theta,\lambda,t) = \sum_{m=1}^{2}\sum_{j=0}^{2K}\left[U_j^{(m)}(\theta,\lambda)P_j^{(m)}(t) + V_j^{(m)}(\theta,\lambda)Q_j^{(m)}(t)\right] \qquad 4.11.1$$

where K=1 or 2, m denotes the tidal 'species', $P_j^{(m)}(t)$ and $Q_j^{(m)}(t)$ are the 'orthotide' functions and $U_j^{(m)}(\theta,\lambda)$ and $V_j^{(m)}(\theta,\lambda)$ are the admittance parameters. Using the Eqn. 4.11.1 the amplitude and phase for each harmonic may be derived (e.g. Cartwright and Ray, 1990, Appendix A). The tidal elevation is then given by:

$$\eta = z_o + \sum_n H_n j_n \cos(\sigma_n t + v_n - g_n) \qquad 4.11.2$$

where z_o (m) is the mean sea level above a chart datum, H_n (m) is the amplitude for constituent n (spatially variable), g_n (^{o}) is the phase lag for constituent n at Greenwhich (spatially variable), σ_n (^{o}/s) is the frequency for constituent n (constant), t is the time (GMT, s) and f_n and v_n are nodal corrections to correct for the moons 18.613 year precession cycle (time dependent). The nodal corrections are supplied as yearly values from 1970 to 2030 (provided by Proudman Oceanographic Laboratory, UK, 1990) from which the value at time t is interpolated. The tidal phases and nodal corrections to the phase are relative to Greenwhich, hence the time used in nodal interpolations and in 4.11.2 must be converted from local time to GMT.
The tidal constituents used to construct the tide are given in Table 4.2.

Table 4.2 : Tidal Harmonics used for Tide Construction

Name	Doodson Number	Frequency (^{o}/hr)
Q1	135.655	13.40
O1	145.555	13.94
P1	163.555	14.96
S1	164.556	15.00
K1	165.555	15.04
2N2	235.755	27.90
MU2	237.555	27.97
N2	245.655	28.44
NU2	247.455	28.51
M2	255.555	28.98
L2	265.455	29.53
T2	272.556	29.96
S2	273.555	30.00
K2	275.555	30.08

5. Advection Schemes.

5.1 Advection Scheme Types

SHOC employs a variety of advection schemes for tracers and uses a 1st order upwind, 2nd order centered or Van Leer's higher order upwind scheme for momentum. The advection schemes available for tracers are:

1. 1st order upwind
2. 2nd order centered
3. QUICKEST
4. 4th order
5. Van Leer's scheme
6. Semi-Lagrangian scheme

All schemes have been implemented on the curvilinear coordinate system for non-uniform grids, and are solved using a non-splitting method (as opposed to a splitting, or fractional step method; see Yanenko, 1971). All schemes are implemented in the flux form of the advection equation, i.e. solution of the following is sought:

$$\frac{\partial T}{\partial t}+\frac{\partial (uT)}{\partial x}+\frac{\partial (vT)}{\partial y}+\frac{\partial (wT)}{\partial z}=0 \qquad 5.1.1$$

The higher order schemes have been rendered monotonic by the optional application of the ULTIMATE limiter (Leonard, 1991). The ULTIMATE filter is generally unsuccessful when applied to the lower order monotonic schemes. A review of errors asscoiated with advection schemes and various approaches to the solution of 5.1.1 can be found in Rood (1987).

The one dimensional implementation of the flux form of the advection equation can be written as:

$$\frac{T^{t+1}-T^{t}}{\Delta t}+\frac{F_{i+\frac{1}{2}}-F_{i-\frac{1}{2}}}{h}=0 \qquad 5.1.2$$

where T is a tracer value h=h_1 and i is the Cartesian coordinate in the ξ_1 direction. The flux through the cell face i, $F_{i-1/2}$, is equal to the velocity u=u_1 multiplied by an interpolation of T onto the cell face at i. This interpolation varies according to the type of scheme used. The fluxes for the schemes used in SHOC are listed below. For some of the schemes the advective form of the equation is also given to provide insight into the scheme. Stability of these schemes is usually defined in terms of the Courant number:

$$q=\frac{T|u|}{h} \qquad 5.1.3$$

5.1.1 Upwind

The upwind (or one-sided, donor cell) scheme is a first order scheme is given in its advective form by:

$$\frac{T^{t+1}-T^{t}}{\Delta t}+0.5(u-|u|)\frac{T_{i+1}^{t}-T_{i}^{t}}{h}+0.5(u+|u|)\frac{T_{i}^{t}-T_{i-1}^{t}}{h}+O(h,T)=0 \qquad 5.1.4$$

where $u=0.5(u_{i+1}+u_i)$. The fluxes for the flux form of the advection equation, eqn. 5.1.2 are therefore:

$$F_{i-\frac{1}{2}} = 0.5(u_i - |u_i|)T_i^t + 0.5(u_i + |u_i|)T_{i-1}^t \quad 5.1.5$$

This scheme suffers severe numerical error in the form of numerical diffusion, which acts to destroy frontal features in the tracer distribution. This scheme is, however, monotonic (i.e. no new maxima or minima are introduced into the solution) and positive definite (if T>0 at t=0 then T>0 for t>0). This scheme is stable for $q \leq 1$.

5.1.2 2nd Order Centered

The 2nd order centered scheme is second order accurate and is given in advective form by:

$$\frac{T^{t+1} - T^t}{\Delta t} + u\frac{T_{i+1}^t - T_{i-1}^t}{2h} + O(h^2, T^2) = 0 \quad 5.1.6$$

with corresponding fluxes through cell face i by:

$$F_{i-\frac{1}{2}} = 0.5u_i(T_i^t + T_{i-1}^t) \quad 5.1.7$$

This scheme used with Euler forward time stepping is unstable and requires diffusion to stabilize it. This means that a non-zero diffusion coefficient should be set when using the 2nd order scheme (a very rough approximation is $V_h=0.01.h^2/\Delta t$). The minimum amount of diffusion required to stabilize the 2nd order Euler forward scheme results in the Lax-Wendroff scheme:

$$\frac{T^{t+1} - T^t}{\Delta t} + u\frac{\partial T^t}{\partial x} = 0.5u^2\Delta t\frac{\partial^2 T}{\partial x^2} \quad 5.1.8$$

Note that the upwind scheme represents the minimum amount of diffusion that is required to be added to the 2nd order Euler forward scheme to render it monotonic (Rood, 1987). If a centered time stepping scheme is used (e.g. leapfrog scheme) then 5.1.6 becomes stable if $q \leq 1$. The 2nd order scheme suffers numerical error in the form of numerical dispersion. This typically leads to overshoots and undershoots in the solution in the vicinity of fronts resulting in new tracer maxima or minima. Hence this scheme is non-monotonic and not positive definite. The scheme is, however, computationally efficient.

5.1.3 2nd Order Upwind

This second order upwind approximation is described by Leonard (1994, Eqn. 34) is stable for Courant numbers < 2, and is given by:

$$T^{t+1} = T^t - q\left[\left(\frac{3-q}{2}\right)T^t - (2-q)T_{i-1}^t + \left(\frac{1-q}{2}\right)T_{i-2}^t\right] \quad 5.1.9$$

with corresponding fluxes through cell face i by:

$$F_{i-\frac{1}{2}} = \left(\frac{3-q}{2}\right)T_{i-1}^t - \left(\frac{1-q}{2}\right)T_{i-2}^t \qquad for\ 0 < q \leq 2 \quad 5.1.10$$

5.1.4 QUICKEST

The QUICKEST scheme (Quadratic Upstream Interpolation for Convective Kinematics with Estimated Streaming Term) is a 3rd order accurate upwind scheme described by Leonard (1979). This scheme has very little numerical diffusion and numerical dispersion is limited to one small ripple in the vicinity of fronts. Used in conjunction with the ULTIMATE filter the scheme is monotonic and positive definite, and is the most accurate advection scheme SHOC provides. The drawback is that it is more computationally expensive than other schemes. QUICKEST is stable for $q \leq 1$. The flux representation of the QUICKEST scheme is given by:

$$T_i^{t+1} = T_i^t - (q_{i+\frac{1}{2}} F_{i+\frac{1}{2}} - q_{i-\frac{1}{2}} F_{i-\frac{1}{2}}) \qquad 5.1.11$$

where $q_{i-\frac{1}{2}} = u_i \Delta t / h$ and $q_{i+\frac{1}{2}} = u_{i+1} \Delta t / h$ are the Courant numbers on the left and right cell faces respectively. The values $F_{i-1/2}$ and $F_{i+1/2}$ are the tracer concentrations at the left and right cell faces where;

$$F_{i-\frac{1}{2}} = 0.5(T_{i+1} + T_{i-1}) - 0.5 q_{i-\frac{1}{2}} (T_i - T_{i-1}) - \left\lfloor \tfrac{1}{6} - \tfrac{1}{6} q_{i-\frac{1}{2}}^2 \right\rfloor CURV \qquad 5.1.12$$

with:

$$\begin{aligned} CURV &= T_i - 2T_{i-1} + T_{i-2} \quad for \;\; q_{i-\frac{1}{2}} > 0 \\ CURV &= T_{i+1} - 2T_i + T_{i-1} \quad for \;\; q_{i-\frac{1}{2}} < 0 \end{aligned} \qquad 5.1.13$$

5.1.5 4th Order Scheme

The 4th order scheme in advective form is given by:

$$\frac{T^{t+1} - T^t}{\Delta t} + u \left[\frac{4}{3} \frac{T_{i+1}^t - T_{i-1}^t}{2h} - \frac{1}{3} \frac{T_{i+2}^t - T_{i-2}^t}{4h} \right] + O(h^4) = 0 \qquad 5.1.14$$

with corresponding fluxes given by:

$$F_{i-\frac{1}{2}} = u_i \left[\tfrac{7}{12} (T_i^t + T_{i-1}^t) - \tfrac{1}{12} (T_{i+1}^t + T_{i-2}^t) \right] \qquad 5.1.15$$

The 4th order scheme has error characteristics of the same type as the 2nd order scheme although less severe. This scheme is more accurate than the 2nd order scheme and when combined with the ULTIMATE filter also provides very good results. The computational effort required is larger than some of the other schemes.

5.1.6 Van Leer's Scheme

This scheme is a higher order upwind scheme, described by Van Leer (1979). This scheme is monotonic and suffers less numerical diffusion than the 1st order upwind scheme. The computational efficiency of this scheme is also fair. The fluxes for this scheme are given by:

$$F_{i-\frac{1}{2}} = u_i\left[T_{i-1}^t + 0.5(1-q_i)\Delta_{i-1}T\right] \qquad for\ u_i > 0$$
$$F_{i-\frac{1}{2}} = u_i\left[T_i^t - 0.5(1+q_i)\Delta_i T\right] \qquad for\ u_i < 0 \qquad 5.1.16$$

where ΔT is an operator (if ΔT=0 the 1st order upstream scheme is recovered). An expression that is monotonic with now numerical diffusion is given by (Allen et al, 1991):

$$\Delta_i T = \frac{2(T_i - T_{i-1)})(T_{i+1} - T_i)}{(T_{i+1} - T_{i-1})} \quad for\ (T_i - T_{i-1)})(T_{i+1} - T_i) > 0$$
$$\Delta_i T = 0 \qquad for\ (T_i - T_{i-1)})(T_{i+1} - T_i) \le 0 \qquad 5.1.17$$

5.1.7 Semi-Lagrangian Scheme

The semi-Lagrangian scheme adopts the approach of tracing back along a velocity streamline from each cell node until the location in the grid at time t-Δt is located. The value of the tracer at this location at time t is then equal to the tracer value at the origin of the streamline (cell node) at t+Δt. Generally an interpolation is required to obtain the tracer value at the end of the traced streamline. The semi-Lagrangian formulation in SHOC uses a tri-linear interpolation which results in characteristics similar to the 1st order upstream method (i.e. diffusive, monotonic, positive definite). The advantage of this scheme is that it is unconditionally stable, hence tracers may be invoked on much longer time-steps than the 3D mode. Also, the origin of the streamline only needs to be found once irrespective of the number of tracers simulated, thus this scheme becomes more computationally efficient in comparison to other schemes as the number of simulated tracers increases. The three-dimensional representation of the semi-Lagrangian scheme is (Casulli and Cheng, 1992):

$$T_{i,j,k}^{t+1} = T_{i-a,j-b,k-c}^t \qquad 5.1.18$$

where $a=u_1\Delta t/h_1$, $b=u_2\Delta t/h_2$ and $c=w\Delta t/\Delta z$ are the Courant numbers in the ξ_1, ξ_2, and z directions. The tri-linear interpolation can be written as:

$$\begin{aligned} T_{i-a,j-b,k-c}^t = {} & (1-r)(1-p)\left[(1-q)T_{i-l,j-m,k-n}^t + qT_{i-l,j-m-1,k-n}^t\right] \\ & + (1-r)p\left[(1-q)T_{i-l-1,j-m,k-n}^t + qT_{i-l-1,j-m-1,k-n}^t\right] \\ & + r(1-p)\left[(1-q)T_{i-l,j-m,k-n-1}^t + qT_{i-l,j-m-1,k-n-1}^t\right] \\ & + r.p\left[(1-q)T_{i-l-1,j-m,k-n-1}^t + qT_{i-l-1,j-m-1,k-n-1}^t\right] \end{aligned} \qquad 5.1.19$$

where for a,b,c>0, l,m,n are the integer parts and p,q,r the decimal parts of a,b,c respectively, i.e. a=l+p, b=m+q and c=n+r.

Due to the longer time-step that may be employed with this scheme it is possible to solve the tracer equation on a different (longer) time-step, Δt_t, than the momentum equations. Figure 5.1(a) shows surface and a cross-shelf section of salinity solutions using the semi-Lagrange scheme for a simulation performed on the North West Shelf of Australia using the momentum time-step Δt = 780s. Figure 5.1 (b) shows the same simulation with the semi-Lagrange scheme invoked on every 4th time-step, i.e. the tracers operate on a time-step Δt_t = 3120 s. Solutions are visually identical and significant saving is made in the computational time (Table 5.1).

Table 5.1 : Tracer time-stepping run time ratios

Time-step (s)	Mean CPU / iteration (s)	Run time ratio
780	0.561	0.00234
3120	0.494	0.00206

Figure 5.1 (a) : Salinity solutions for Δt_t = 780

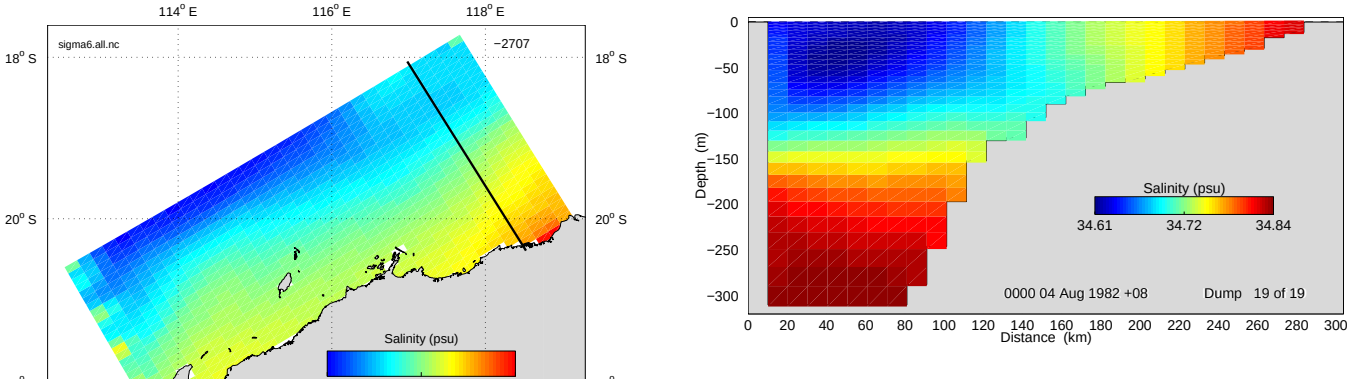
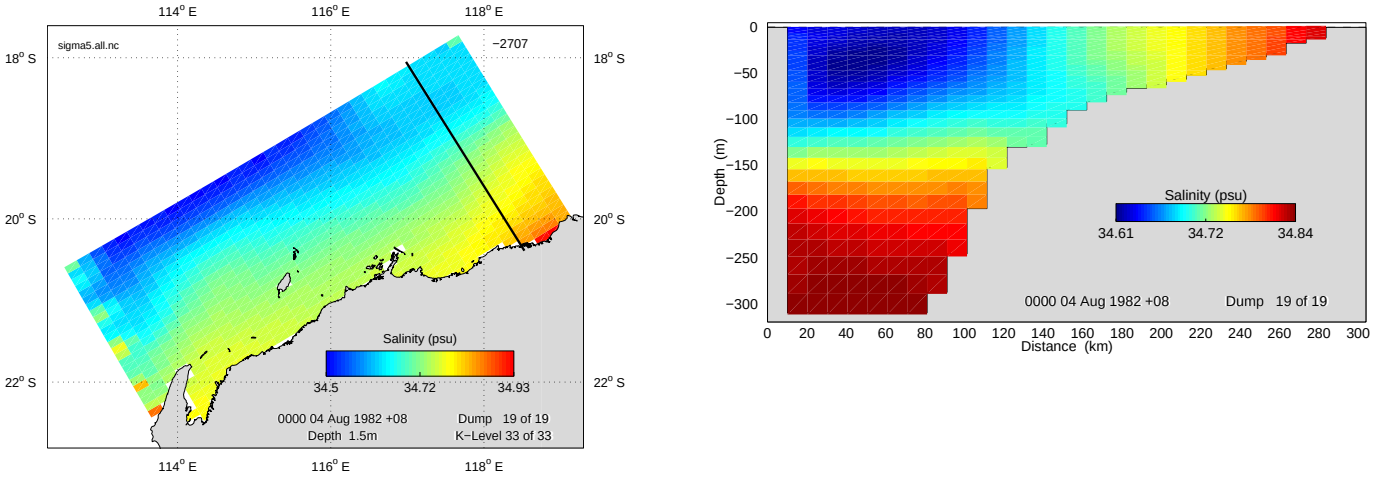


Figure 5.1 (b) : Salinity solutions for Δt_t = 3120



5.1.7 Angular Momentum Scheme

The angular derivative (see Kowalik and Murty (1993), p51, 186) is an implicit second order in space and time scheme which exhibits unconditional stability. This scheme is implemented for momentum only to better handle instances where the Courant stability criterion is violated in the momentum advection, typically due to large vertical velocities near large changes in bathymetry. The sub-stepping algorithm is not always successful in handling these violations, hence the use of the implicit angular derivative to improve stability. Substituting the 2nd order fluxes 5.1.7 into 5.1.2 gives an explicit form of the difference equation:

$$\frac{T^{t+1}-T^{t}}{\Delta t}+\frac{u_{i+1}(T_{i+1}+T_i)-u_i(T_i+T_{i-1})}{2h}=0 \qquad 5.1.20$$

The angular derivative is constructed by evaluating one of the tracer values in each of the flux terms at the forward time-step, e.g;

$$\frac{T^{t+1}-T^{t}}{\Delta t}+\frac{u_{i+1}(T_{i+1}+T_{i}^{t+1})-u_{i}(T_{i}+T_{i-1}^{t+1})}{2h}=0 \qquad 5.1.21$$

This can then be re-arranged to solve for T_i^{t+1}, e.g;

$$T^{t+1}=\frac{T^{t}-\Delta t\frac{u_{i+1}T_{i+1}-u_{i}(T_{i}+T_{i-1}^{t+1})}{2h}}{(1-\Delta t u_{i+1}\frac{T_{i}^{t+1}}{2h})} \qquad 5.1.22$$

Note that if the grid is ordered correctly (i.e. 5.1.22 is solved in order of increasing i) then the value of T_{i-1}^{t+1} is known since it was calculated one grid cell before the current grid cell. This can be extended to three dimensions, noting that at land boundaries are subject to the zero-flux condition, $T_{i-1}^{t+1}=0$ or the free-slip condition, $T_{i-1}^{t+1}=T_{i}^{t+1}$. The surface and bottom vertical fluxes are equivalent to prescribing a free-slip condition at the surface and the bottom. In three dimensions the angular derivative must start in the surface south-west corner of the grid and proceed in order of increasing k, increasing i and increasing j. Note that the implicit nature of this algorithm makes it unsuitable for use with multi-threaded or distributed processing.

5.1.8 Implicit Vertical Advection

An implicit second order vertical advection scheme is included which provides unconditional stability for vertical advection, hence improves stability in regions of large horizontal divergence. Under these circumstances vertical advection may violate the stability criterion (note that the CFL condition for the time-step does not consider vertical motion) and fuel instability in horizontal flow. The vertical advection equation is written as (discetizing the time derivative as Euler forward for simplicity):

$$\frac{u^{n+1}-u^{n}}{dt}+\frac{\partial wu^{n+1}}{\partial z}=f(u) \qquad 5.1.23$$

Where *u* is velocity in either e_1 or e_2 directions and *f(u)* is the sum of horizontal advection, diffusion, pressure and Coriolis terms. This may be written:

$$u^{n+1}=u^{n}+dt.f(u)-dt\frac{\partial wu^{n+1}}{\partial z} \qquad 5.1.24$$

Using fractional steps to split the equation; first solve for *f(u)*:

$$\tilde{u}=u^{n}+dt.f(u) \qquad 5.1.25$$

Then solve for vertical diffusion implicitly:

$$u^{n+1} = \tilde{u} - dt\frac{\partial wu^{n+1}}{\partial z} \qquad 5.1.26$$

For non-uniform vertical grid spacing, the velocity at any vertical layer face (see Fig. 1.1) is provided by a linear interpolation:

$$\hat{u}_k = df_k(u_k - u_{k-1}) + u_{k-1} \qquad \text{where} \quad df_k = \frac{dz_{k-1}}{dz_k + dz_{k-1}} \qquad 5.1.27$$

Then dropping the time superscripts and discretizing:

$$\frac{\partial wu^{n+1}}{\partial z} = (w_{k+1}\hat{u}_{k+1} - w_k\hat{u}_k)/dz_k \qquad 5.1.28$$

The vertical velocity is defined at the layer faces and includes surface and bottom boundary conditions. Incorporating this into the variable $\hat{w}$ and noting the stagger of the index k (i.e. $\hat{w}$ is defined in cell layer faces rather than centres, hence has dimension *nz+1* rather than *nz*), then in the e_1 direction;

$$\begin{aligned} \hat{w}_{ks} &= 0.5(wtop_i + wtop_{i-1}) \qquad k = k_s \\ \hat{w}_k &= 0.5(w_{i,k+1} + w_{i-1,k+1}) \qquad k_s < k \le k_b \\ \hat{w}_{kb-1} &= 0.5(wbot_i + wbot_{i-1}) \qquad k = k_b - 1 \end{aligned} \qquad 5.1.29$$

Similar velocities are defined in the e_2 direction. Using the definition of $\hat{w}$, 5.1.26 and 5.1.28, then for the mid-water layers $k_b < k < k_s$:

$$u_k = \tilde{u} - \frac{dt}{dz_k}\left(\hat{w}_k(df_{k+1}(u_{k+1} - u_k) + u_k) - \hat{w}_{k-1}(df_k(u_k - u_{k-1}) + u_{k-1})\right) \qquad 5.1.30$$

$$\Rightarrow u_k = \tilde{u} - \frac{dt}{dz_k}\left(\hat{w}_k df_{k+1}u_{k+1} - \hat{w}_k df_{k+1}u_k + \hat{w}_k u_k - \hat{w}_{k-1}df_k u_k + \hat{w}_{k-1}df_k u_{k-1} - \hat{w}_{k-1}u_{k-1}\right) \qquad 5.1.31$$

$$\Rightarrow \frac{u_k dz_k}{dt} + u_k(\hat{w}_k - \hat{w}_k df_{k+1} - \hat{w}_{k-1}df_k) + u_{k+1}\hat{w}_k df_{k+1} + u_{k-1}(\hat{w}_{k-1}df_k - \hat{w}_{k-1}) = \frac{\tilde{u}dz_k}{dt} \qquad 5.1.32$$

The vertical diffusion implicit formulation is written as (see Section 6.12):

$$u_k\left[\frac{dz_k}{dt} - C_p - C_m\right] + u_{k+1}C_p + u_{k-1}C_m = rhs \qquad 5.1.33$$

where:

$$C_p = -\frac{\upsilon_{k+1}}{dzz_{k+1}}, \quad C_m = -\frac{\upsilon_k}{dzz_k}, \quad rhs = \tilde{u}\frac{dz}{dt} \tag{5.1.34}$$

Including 5.1.32 in the definitions of C_p and C_m gives;

$$C_p = -\frac{\upsilon_{k+1}}{dzz_{k+1}} + \hat{w}_k df_{k+1} \tag{5.1.35}$$

$$C_m = -\frac{\upsilon_k}{dzz_k} + \hat{w}_{k-1} df_k - \hat{w}_{k-1} \tag{5.1.36}$$

Then:

$$u_k\left[\frac{dz_k}{dt} - C_p - C_m - \hat{w}_{k-1} + \hat{w}_k\right] + u_{k+1}C_p + u_{k-1}C_m = rhs \tag{5.1.37}$$

At the surface, $k = k_s$:

$$\frac{\partial wu}{\partial z}^{n+1} = (\hat{w}_{ks}u_k - \hat{w}_{ks-1}\hat{u}_{ks})/dz_{ks} \tag{5.1.38}$$

leading to:

$$u_k = \tilde{u} - \frac{dt}{dz_{ks}}\left(\hat{w}_{ks}u_{ks} - \hat{w}_{ks-1}df_{ks}u_{ks} + \hat{w}_{ks-1}df_{ks}u_{ks-1} - \hat{w}_{ks-1}u_{ks-1}\right) \tag{5.1.39}$$

$$\Rightarrow \frac{u_k dz_{ks}}{dt} + u_{ks}(\hat{w}_{ks} - \hat{w}_{ks-1}df_{ks}) + u_{ks-1}(\hat{w}_{ks-1}df_{ks} - \hat{w}_{ks-1}) = \frac{\tilde{u}dz_{ks}}{dt} \tag{5.1.40}$$

Then:

$$C_p = 0 \tag{5.1.41}$$

$$C_m = -\frac{\upsilon_k}{dzz_k} + \hat{w}_{ks-1}df_{ks} - \hat{w}_{ks-1} \tag{5.1.42}$$

And 5.1.37 follows. For the bottom layer, $k = k_b$:

$$\frac{\partial wu}{\partial z}^{n+1} = (\hat{w}_{kb}\hat{u}_{kb+1} - \hat{w}_{kb-1}u_{kb})/dz_{kb} \tag{5.1.43}$$

leading to:

$$u_k = \tilde{u} - \frac{dt}{dz_{kb}}\left(\hat{w}_{kb} df_{kb+1} u_{kb+1} - \hat{w}_{kb} df_{kb+1} u_{kb} + \hat{w}_{kb} u_{kb} - \hat{w}_{kb-1} u_{kb}\right) \quad 5.1.44$$

$$\Rightarrow \frac{u_{kb} dz_{kb}}{dt} + u_{kb}(\hat{w}_{kb} - \hat{w}_{kb-1} - \hat{w}_{kb} df_{kb+1}) + u_{kb+1} \hat{w}_{kb} df_{kb+1} = \frac{\tilde{u} dz_{kb}}{dt} \quad 5.1.45$$

Then:

$$C_p = -\frac{\upsilon_{k+1}}{dzz_{k+1}} + \hat{w}_{kb} df_{kb+1} \quad 5.1.46$$

$$C_m = 0 \quad 5.1.47$$

The tri-diagonal matrix is constructed using 5.1.35 and 5.1.36 for the middle layers $k_b < k < k_s$, 5.1.41 and 5.1.42 for the surface layer $k = k_s$, and 5.1.46 and 5.1.47 for the bottom layer $k = k_b$ in conjunction with 5.1.37.

5.2 The ULTIMATE Limiter

The ULTIMATE (Universal Limiter for Transient Interpolation Modelling of the Advective Transport Equations) limiter (Leonard, 1991) is a scheme which may be applied to higher order finite difference advection schemes to remove spurious oscillations and render the scheme monotonic without altering the accuracy of the original advection scheme. This scheme has been applied and evaluated with the QUICKEST algorithm by Binliang and Falconer (1997) and falls into a class of schemes knows as TVD (total variation-diminishing) schemes designed to address spurious oscillations. Normalised tracer variables are introduced at the cell faces:

$$\overline{T}_i = \frac{T_i - T_U}{T_D - T_U}, \quad \overline{T}_{i+\frac{1}{2}} = \frac{T_{i+\frac{1}{2}} - T_U}{T_D - T_U} \quad 5.2.1$$

where T_U is the tracer concentration at the upstream cell center and T_D is the tracer concentration at the downstream cell center. The monotonic solution is obtained if:

$$\overline{T}_{i+\frac{1}{2}} \le \overline{T}_i / q_{i+\frac{1}{2}} \quad for \;\; 0 < \overline{T}_i \le 1 \quad 5.2.2$$

$$\overline{T}_i \le \overline{T}_{i+\frac{1}{2}} \le 1 \quad for \;\; 0 < \overline{T}_i \le 1 \quad 5.2.3$$

$$\overline{T}_{i+\frac{1}{2}} = \overline{T}_i \quad for \;\; 0 < \overline{T}_i \; or \; \overline{T}_i > 1 \quad 5.2.4$$

5.3 Advection Scheme Characteristics

The numerical diffusion and dispersion inherent in a scheme is evaluated by setting up a test domain where a frontal feature is advected along a length of straight coastline. The conservation characteristics may be evaluated by observing the evolution of total tracer content in a closed domain subject to wind driven circulation.

5.3.1 Numerical Diffusion and Dispersion

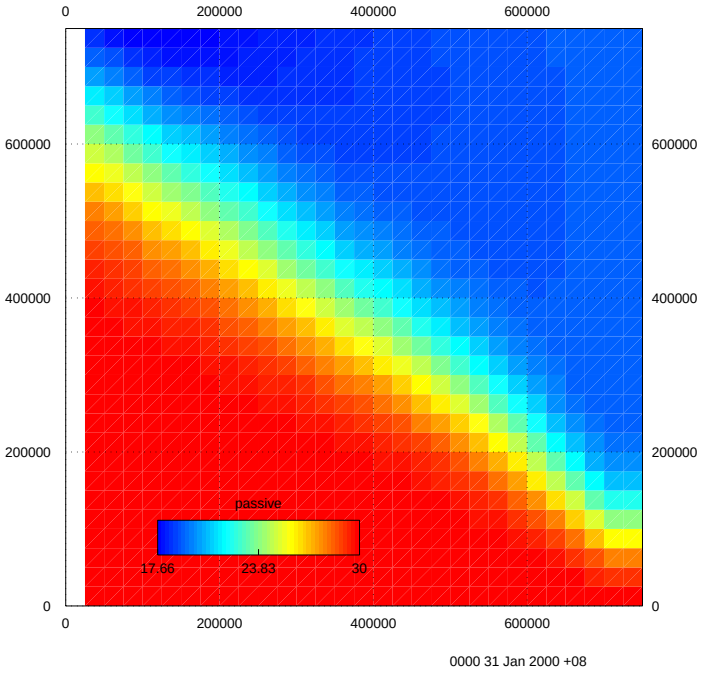
Results for the transport of a tracer front along a straight coast on the western side of a test domain surrounded by three open boundaries are shown below. The domain is forced by a southerly wind (7.5 ms^{-1}) with an initial condition for the passive tracer of 10 above 5m depth and 20 below 10m. High concentration tracer (30) is introduced along the southern boundary. The surface tracer concentration distribution and the concentration along a north-south transect midway along the domain are displayed below for each scheme. The typical diffusive characteristics of the odd ordered schemes and dispersive characteristics of even ordered schemes can be observed. A dramatic improvement in the solutions is obtained by using the ULTIMATE limiter on the higher ordered schemes. Run time characteristics are summarised in Table 5.2 (note: these are grid and processor dependent and act as a guide only).

Table 5.2 : Advection scheme run time ratios

Scheme	**Mean CPU / iteration (s)**	**Run time ratio (model:real)**	**Normalised Speed (%)**
1st order	0.196	5113:1	100
2nd order	0.201	4969:1	98
2nd order + ULTIMATE	0.250	4008:1	78
QUICKEST	0.220	4555:1	89
ULTIMATE QUICKEST	0.268	3734:1	73
4th order	0.202	4962:1	97
4th order + ULTIMATE	0.251	3988:1	78
Van Leer	0.208	4799:1	94
Semi-Lagrangian	0.255	3927:1	77

Figure 5.2 : Tracer Concentration using 1st Order Upwind Scheme

(a) Surface Distribution



(b) Surface Transect

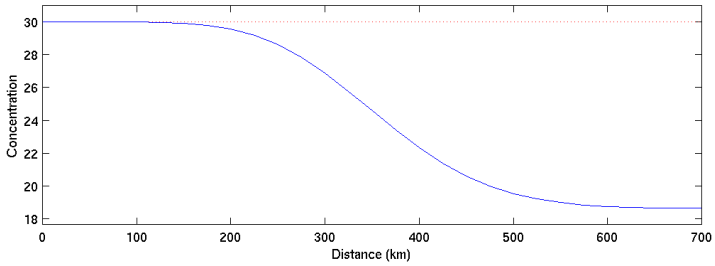
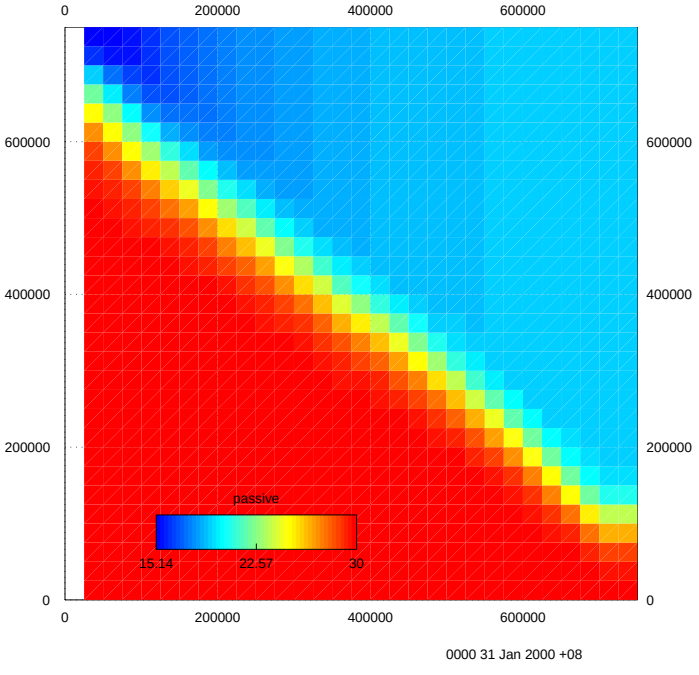


Figure 5.3 : Tracer Concentration using Van Leer's Scheme

(a) Surface Distribution



(b) Surface Transect

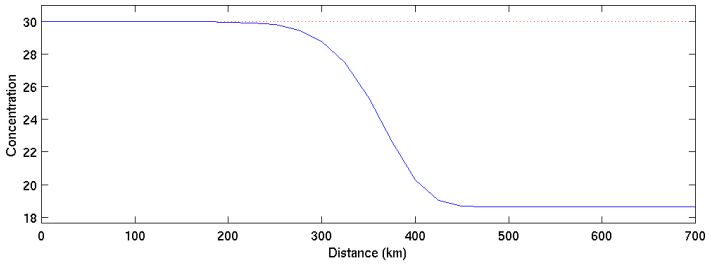
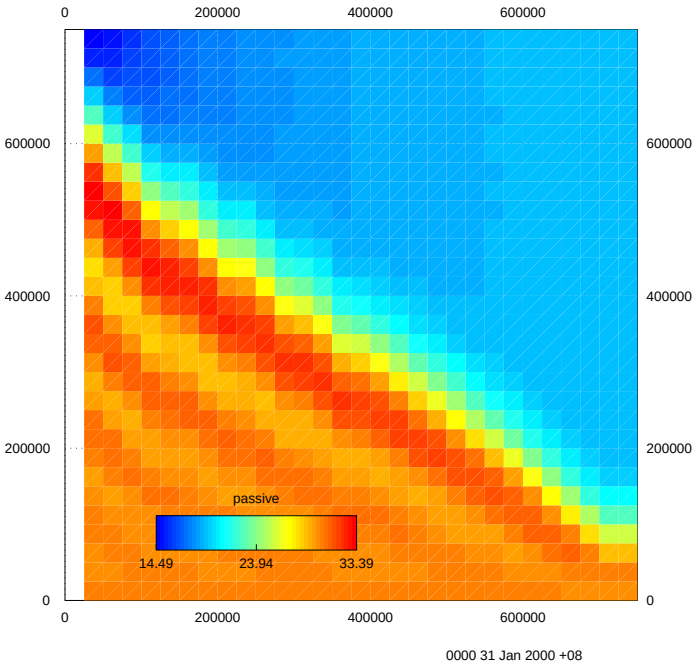


Figure 5.4 : Tracer Concentration using 2nd Order Scheme

(a) Surface Distribution



(b) Surface Transect

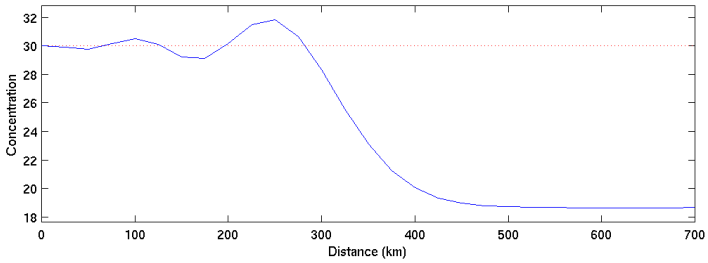
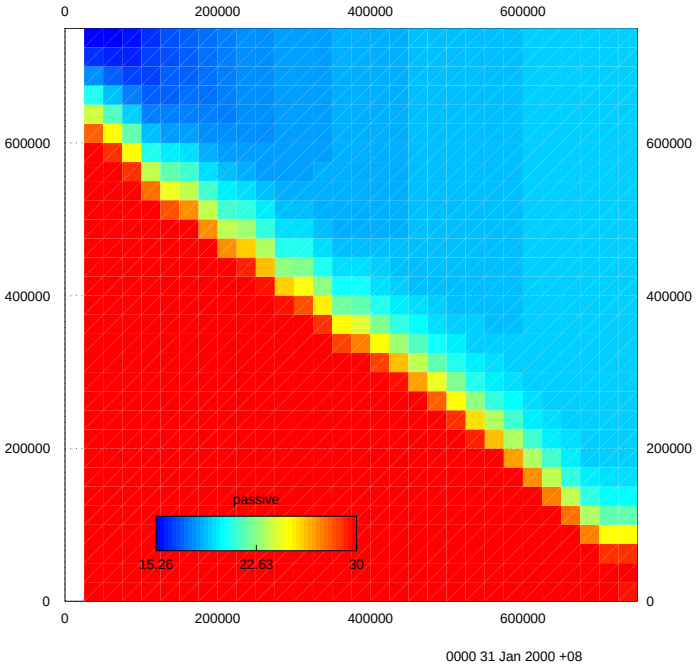


Figure 5.5 : Tracer Concentration using 2nd Order Scheme + ULTIMATE

(a) Surface Distribution



(b) Surface Transect

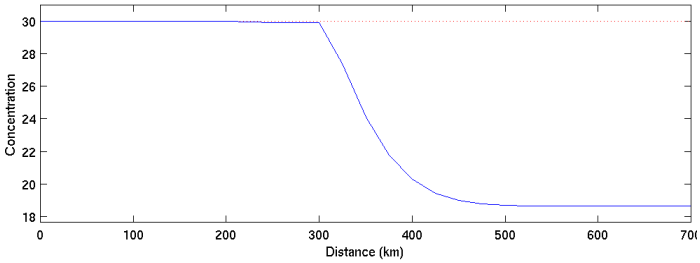
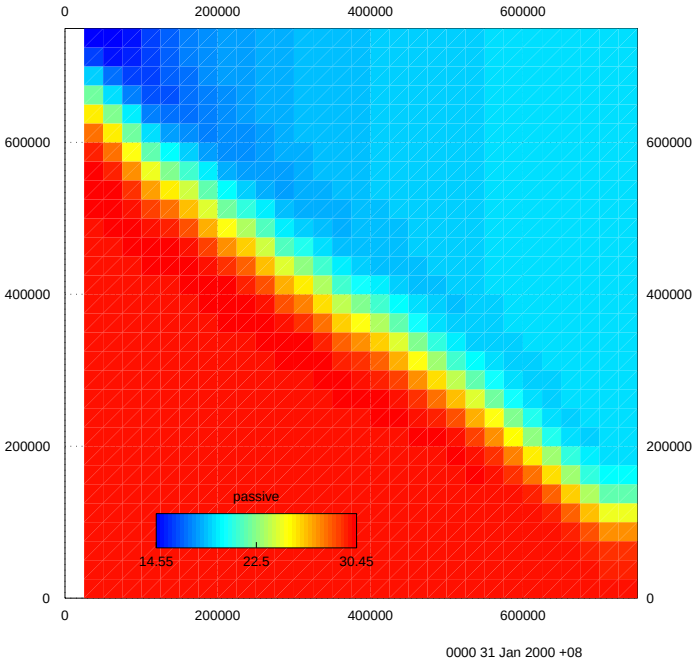


Figure 5.6 : Tracer Concentration using QUICKEST

(a) Surface Distribution



(b) Surface Transect

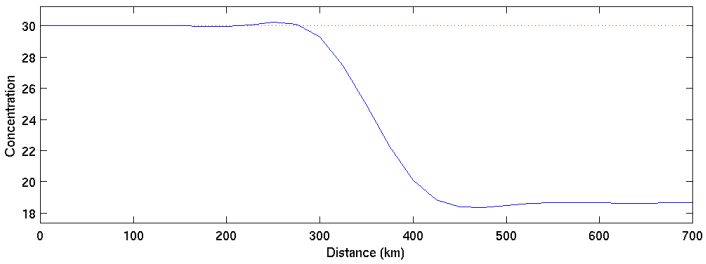
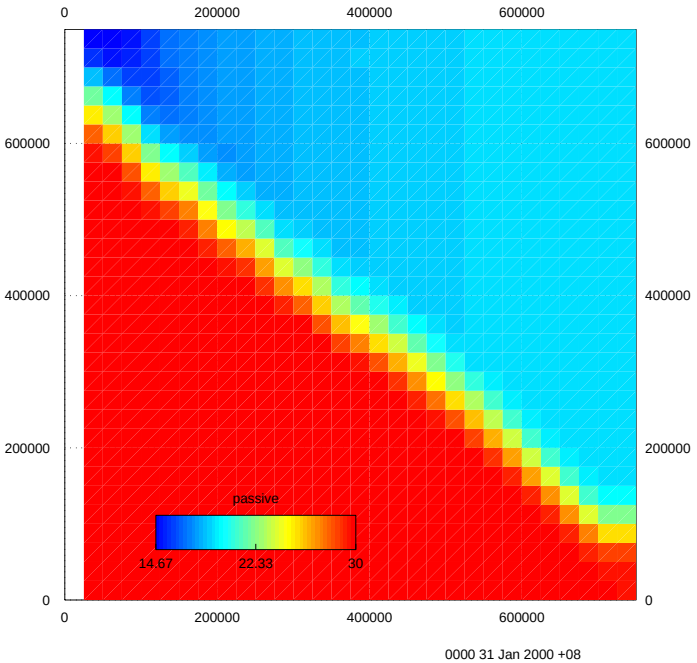


Figure 5.7 : Tracer Concentration using QUICKEST + ULTIMATE

(a) Surface Distribution



(b) Surface Transect

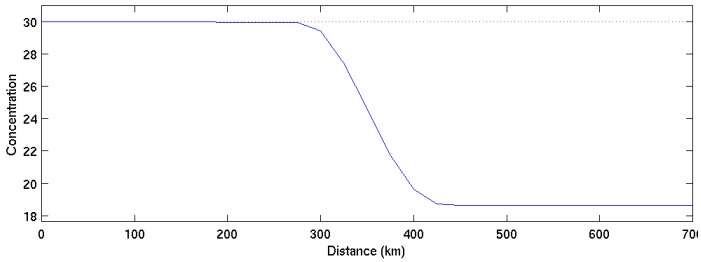
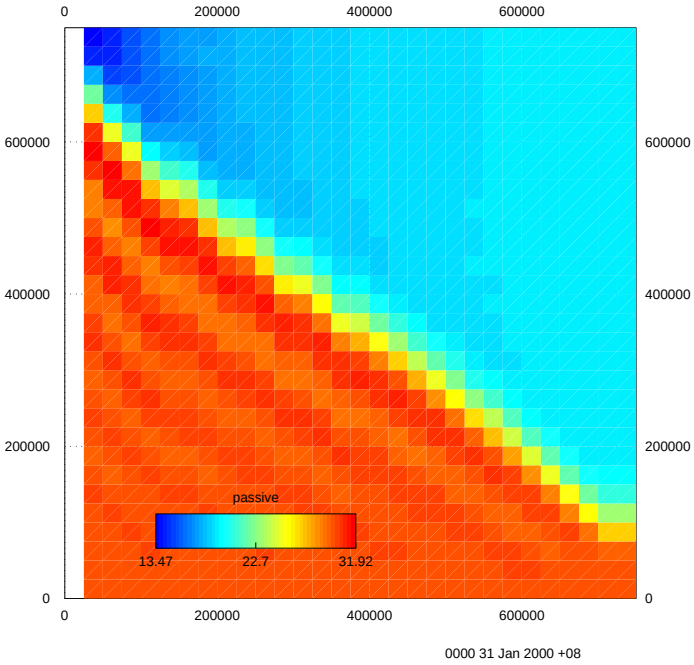


Figure 5.8 : Tracer Concentration using 4th Order Scheme

(a) Surface Distribution



(b) Surface Transect

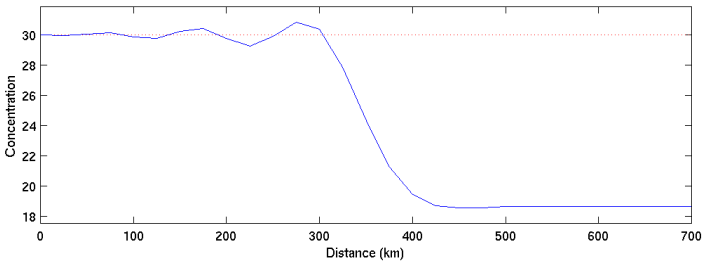
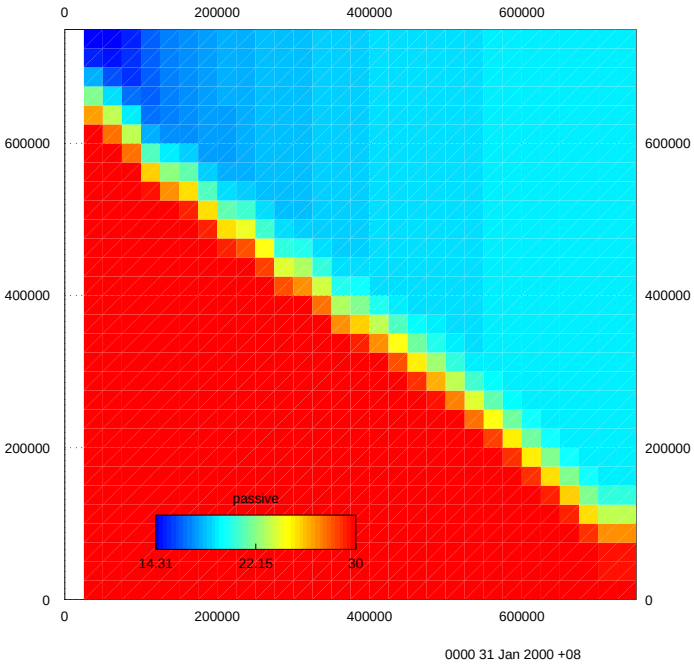


Figure 5.9 : Tracer Concentration using 4th Order Scheme + ULTIMATE

(a) Surface Distribution



(b) Surface Transect

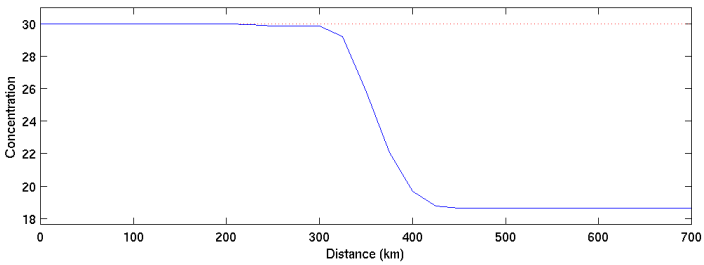
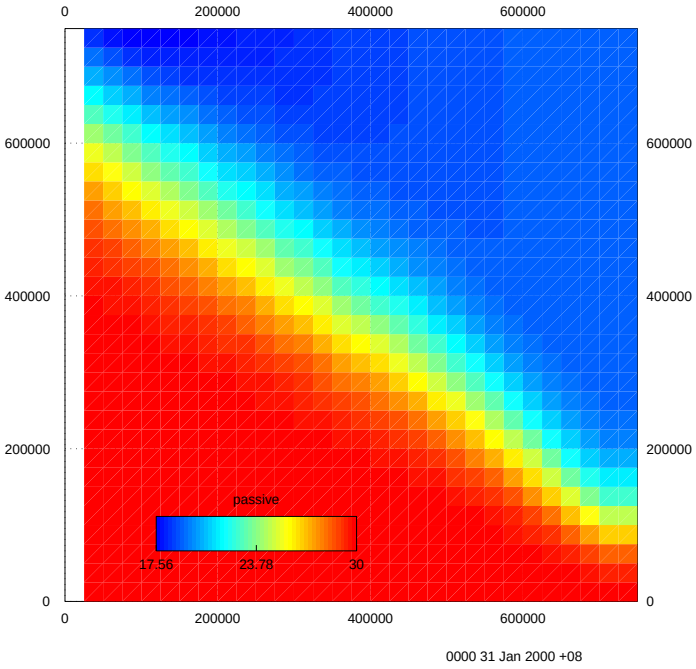
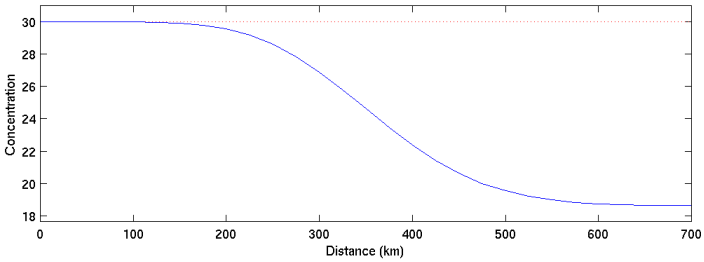


Figure 5.10 : Tracer Concentration using Semi-Lagrange

(a) Surface Distribution



(b) Surface Transect



5.3.2 Conservation Characteristics

The conservation characteristics of the advection schemes were investigated by closing the boundaries of the previous simulations and using an initial depth dependent temperature profile. The total change in heat throughout the simulation provided an indication of the scheme's conservation characteristics. All the advection schemes in flux form display excellent conservation characteristics. In the northern half of the test domain the surface temperature was set at 20°C throughout a surface layer 30m deep, and 10°C below 30m. In the southern half the temperature was 15°C throughout. This provided both vertical and horizontal fronts for the advection scheme to cope with. Results are presented after 30 days below. The conservation parameter plotted is the total heat in the domain, expressed as a % deviation from the total heat at time = 0. The total heat at time t is given by:

$$C^t = \sum_{i,j,k} T^n(i,j,k).\Delta Z^n(i,j,k).area(i,j) \qquad 5.3.1$$

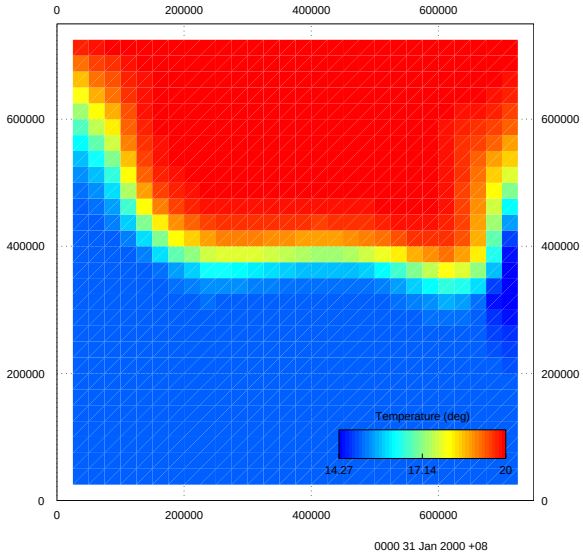
then the conservation parameter is given by:

$$conservation^t = 100\frac{(C^t - C^0)}{C^0} \qquad 5.3.2$$

The results are presented below; these results also provide additional insight into the numerical error associated with each scheme.

Figure 5.11 : Heat Conservation using Semi-Lagrange

(a) Surface Temperature Distribution (b) Heat Conservation



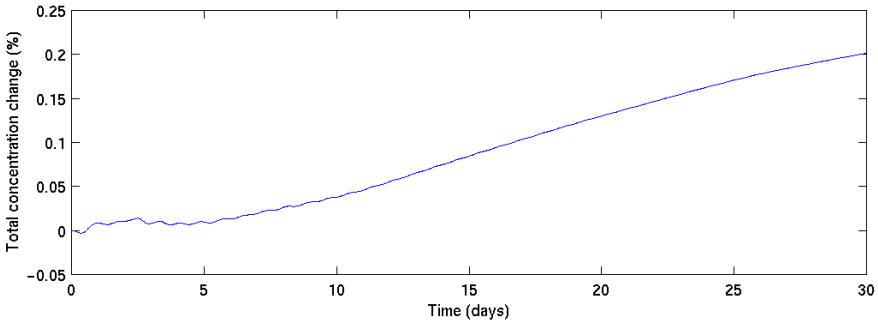
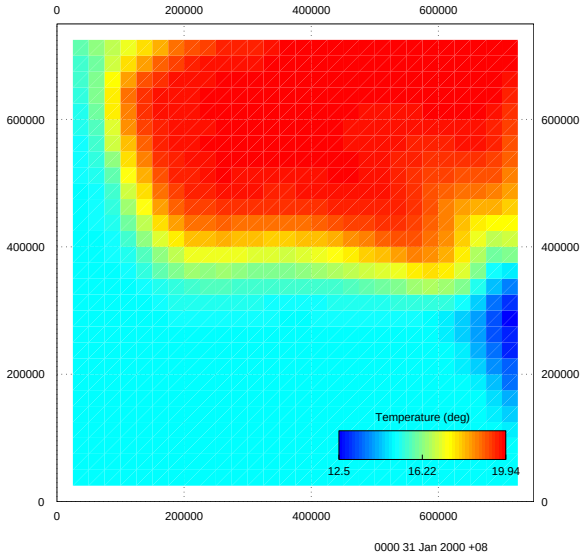


Figure 5.12 : Heat Conservation using 1st Order Upstream Scheme

(a) Surface Temperature Distribution



(b) Heat Conservation

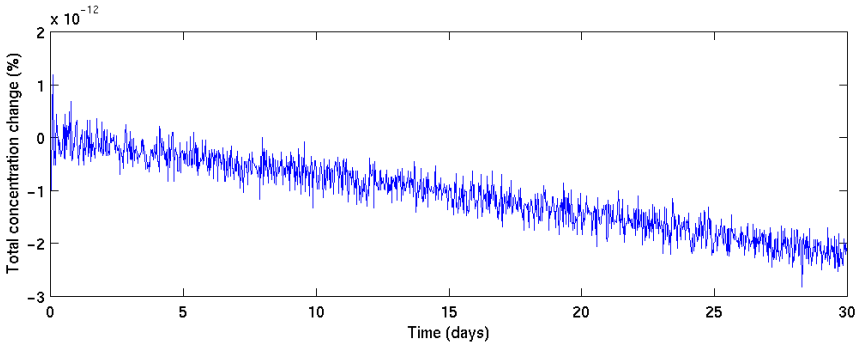
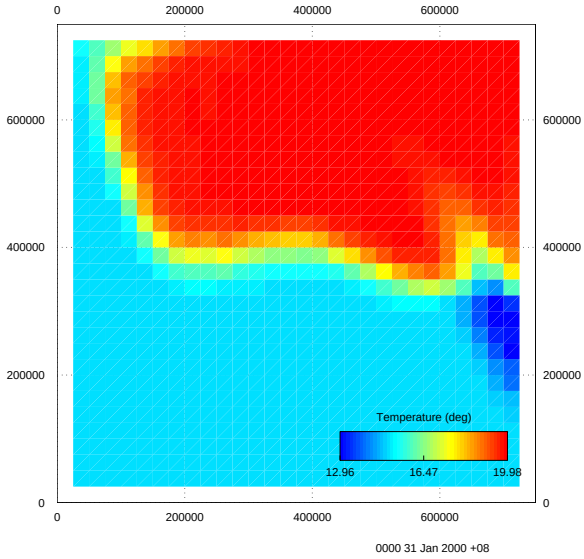


Figure 5.13 : Heat Conservation using Van Leer's Scheme

(a) Surface Temperature Distribution



(b) Heat Conservation

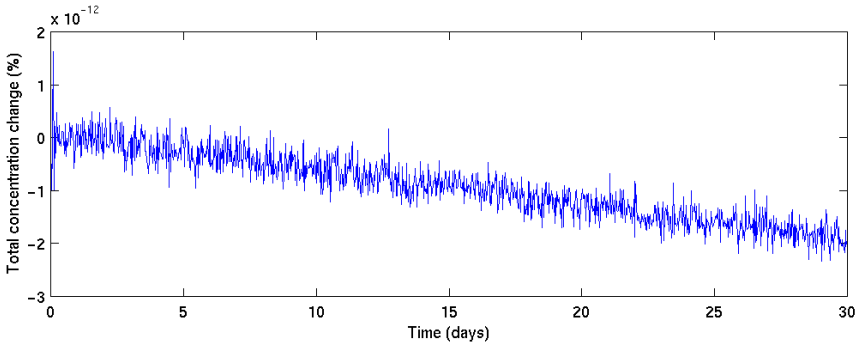
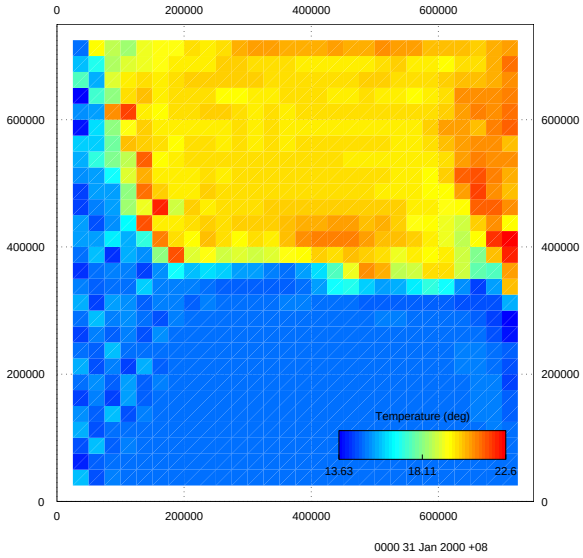


Figure 5.14 : Heat Conservation using 2nd Order Scheme

(a) Surface Temperature Distribution (b) Heat Conservation



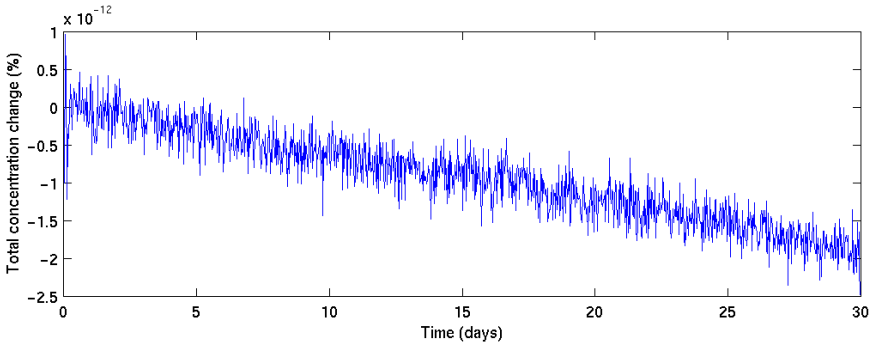
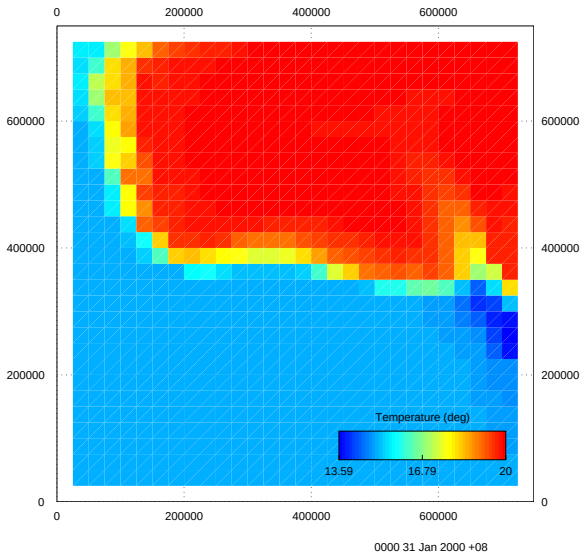


Figure 5.15 : Heat Conservation using 2nd Order Scheme + ULTIMATE

(a) Surface Temperature Distribution (b) Heat Conservation



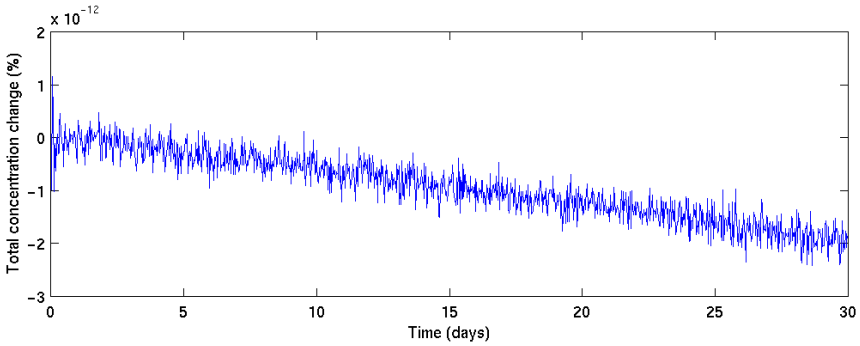
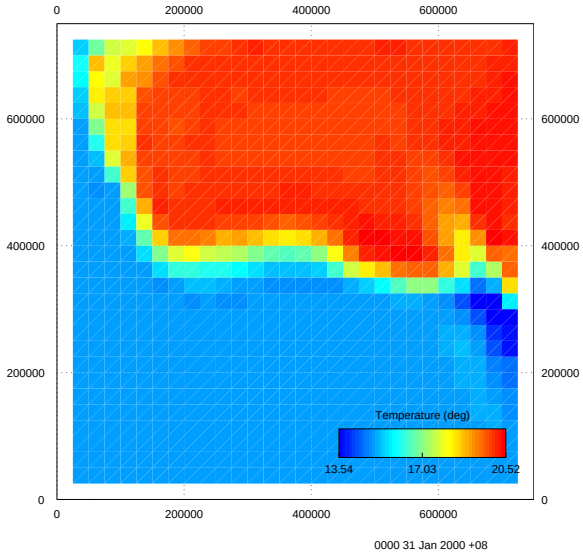


Figure 5.16 : Heat Conservation using QUICKEST

(a) Surface Temperature Distribution



(b) Heat Conservation

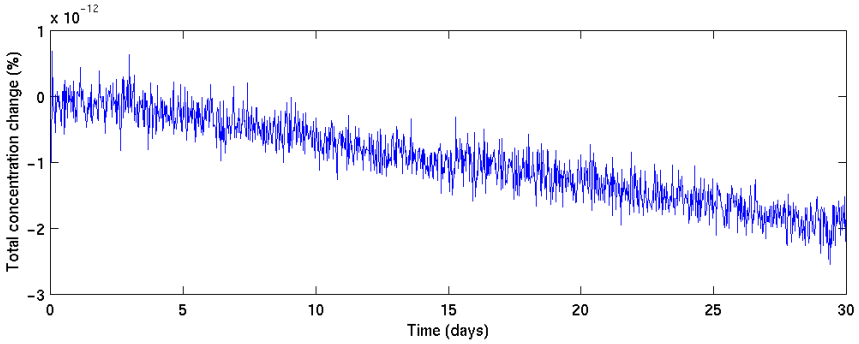
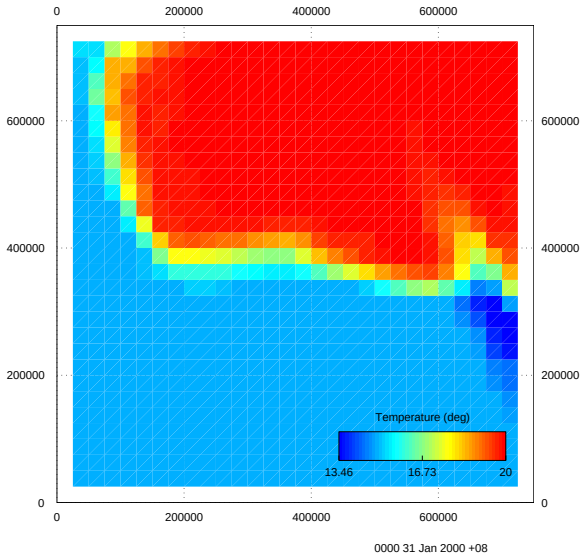


Figure 5.17 : Heat Conservation using QUICKEST + ULTIMATE

(a) Surface Temperature Distribution



(b) Heat Conservation

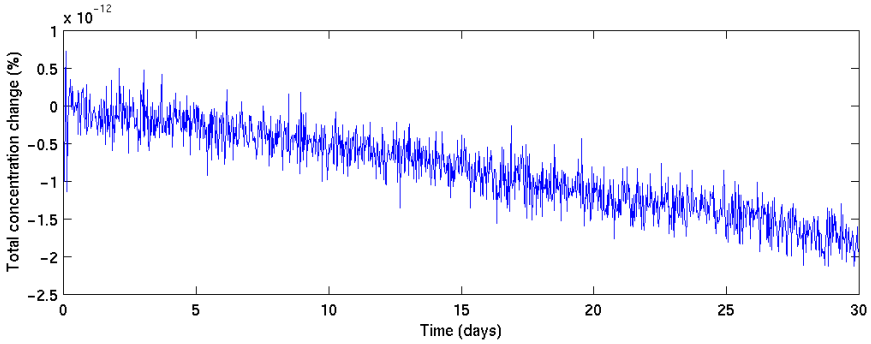
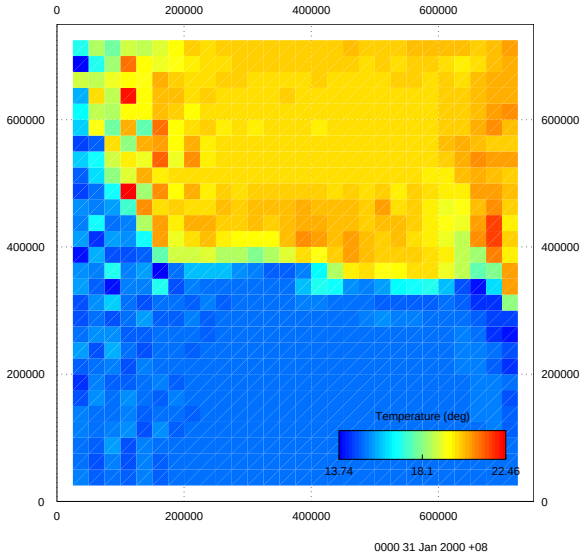


Figure 5.18 : Heat Conservation using 4th Order Scheme

(a) Surface Temperature Distribution (b) Heat Conservation



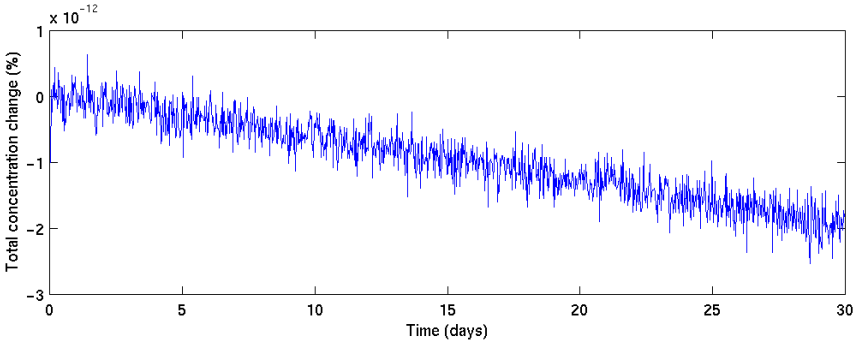
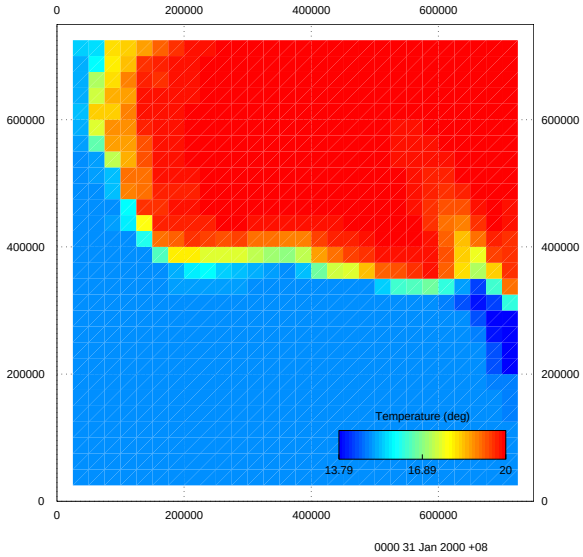
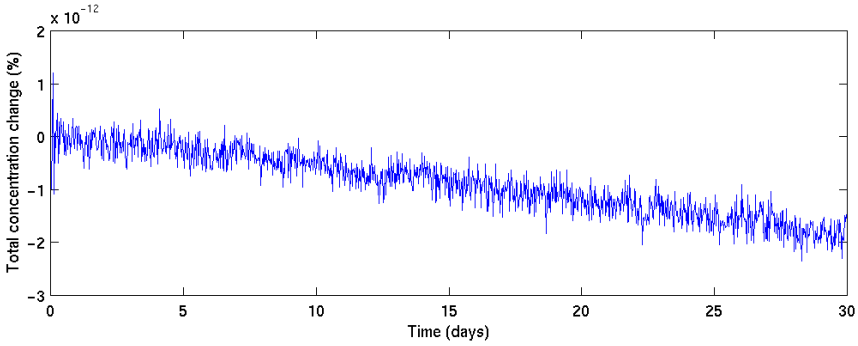


Figure 5.19 : Heat Conservation using 4th Order Scheme + ULTIMATE

(a) Surface Temperature Distribution (b) Heat Conservation





5.4 Stability sub-stepping

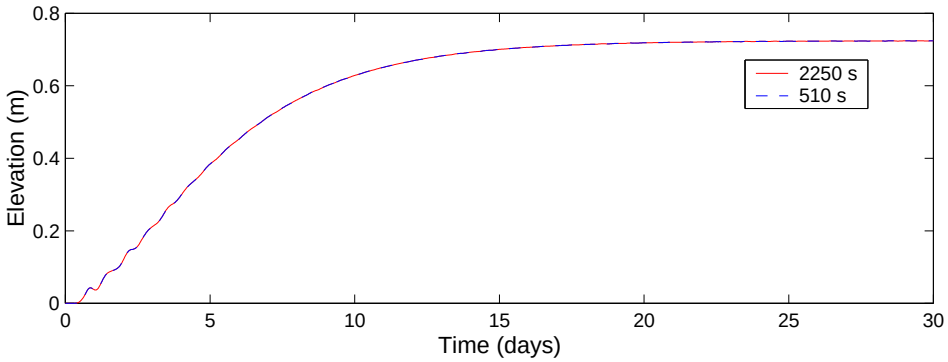
The stability criterion for the advection schemes used in SHOC dictate that the Courant number must be < 1. This may be violated locally in the model if the forcing function becomes large (e.g. for large wind stress) even if the CFL condition is satisfied. An option exists in SHOC where if the maximum Courant number in the grid becomes > 1 then the advective terms (for 2D/3D momentum and tracers) are calculated using a sub-timestep, $\Delta t_s < \Delta t$ where $q_s = \Delta t_s u / h < 1$ for all u and h in the grid. This sub-timestep is given by:

$$\Delta t_s = \frac{sf.h}{u_{\max}} \qquad 5.4.1$$

where sf is a 'safety factor, typically sf=0.9 and u_{max} is the maximum velocity in the grid at time t. The advective terms are looped on the sub-timestep until t+Δt is reached, then the remaining terms in the model equations are calculated using the original time-step, Δt. Note that the original time-step must still obey the CFL condition so as fast moving gravity and internal waves are adequately resolved. This sub-timestepping guarantees that advection schemes are always stable regardless of the forcing imposed.

A comparison of elevation at a location in the domain center of the simulations of Section 5.3.1 is made between results obtained using the CFL time-step, $\Delta t = 2250s$, and results obtained with sub-stepping on the advective terms using $\Delta t_s = 510s$. No difference is observed when sub-timestepping is invoked.

Figure 5.20 : Elevation Comparison of Sub-Timestepping Method



6. Mixing Schemes.

6.1 Mixing Scheme Types

SHOC employs a variety of mixing schemes to generate the vertical mixing coefficients, V_z and K_z. The mixing schemes available are:

1. Constant
2. Csanady type scheme
3. Mellor-Yamada level 2.5
4. Mellor-Yamada level 2.0
5. Modified Mellor-Yamada 2.0
6. k-ε
7. k-ω
8. W88

These schemes are described below.

6.2 Constant

This scheme simply uses a time and space independent constant value for vertical viscosity and diffusivity.

$$\begin{aligned} V_z &= constant \\ K_z &= constant \end{aligned} \qquad 6.2.1$$

6.3 Csanady

The vertical viscosity and diffusivity are approximated using the approach of Csanady (1982), with the modification of Bowden and Hamilton (1975) to account for the effect of stratification, i.e.

$$V_z = V_{z0} + \alpha_V u_* H (1 + 10 R_i)^{-\frac{1}{2}} \qquad 6.3.1$$

$$K_z = K_{z0} + \alpha_K u_* H (1 + 3.33 R_i)^{-\frac{3}{2}} \qquad 6.3.2$$

where u_* is the maximum of surface and bottom friction velocities, H is the water depth, V_{z0}, K_{z0} are the background viscosities and diffusivities respectively, α_V ~ 0.0625 and α_K ~ 0.03 are constants and R_i is the gradient Richardson Number,

$$R_i = \frac{N^2}{M^2} \qquad 6.3.3$$

where N is the Brunt-Vaisala frequency:

$$N^2 \cong g\left[-\frac{1}{\rho} \frac{\partial \sigma_t}{\partial z} \right] \qquad 6.3.4$$

And M is the shear frequency:

$$M^2 = (\frac{\partial u}{\partial z})^2 + (\frac{\partial v}{\partial z})^2 \quad 6.3.5$$

6.4 Mellor-Yamada 2.5

The Mellor-Yamada mixing scheme is based on Mellor and Yamada (1982). The eddy viscosity and diffusivity are related to the product of a velocity scale and a length scale:

$$V_z = c_\mu \sqrt{k} L \quad \text{or} \quad V_z = S_M q L \quad 6.4.1$$

$$K_z = c'_\mu \sqrt{k} L \quad or \quad K_z = S_H q L \quad 6.4.2$$

where k is the turbulent kinetic energy (TKE m^2s^{-1}), L is the turbulence length scale and c_μ and c'_μ are dimensionless stability functions. Note that turbulent kinetic energy may be substituted by turbulence intensity, $q = \sqrt{2k}$ and thus $c_\mu = \sqrt{2} S_M$ and $c'_\mu = \sqrt{2} S_H$. The latter forms of equations 6.4.1 and 6.4.2 are used by Mellor and Yamada (1982). Since this closure scheme may be written in terms of k or q^2, it is sometimes referred to as the kkL closure scheme.
The length scale and TKE can be related to the normalised dissipation rate, ε, (m^2s^{-3}) via:

$$\varepsilon = (c_\mu^0)^3 \frac{k^{\frac{3}{2}}}{L} = \frac{q^3}{B_1 L} \quad 6.4.3$$

where c_μ^0=0.5562 is the momentum stability function for neutral flow and $B_1 = 2^{3/2}(C_\mu^0)^{-3} = 16.6.$

Turbulent kinetic energy is obtained from the solution to the TKE equation, written as:

$$\frac{Dk}{Dt} - \frac{\partial}{\partial z} \upsilon_k \frac{\partial k}{\partial z} = P + B - \varepsilon \quad 6.4.4$$

where $\upsilon_k = 0.283 \sqrt{k} L$ for the Mellor-Yamada scheme, P is the shear production:

$$P = V_z \left[(\frac{\partial u_1}{\partial z})^2 + (\frac{\partial u_2}{\partial z})^2 \right] \quad 6.4.5$$

and B is the buoyancy production:

$$B = -K_z N^2 \quad 6.4.6$$

where N^2 is the buoyancy frequency:

$$N^2 = -\frac{g}{\rho_o} \frac{\partial \rho}{\partial z} \quad 6.4.7$$

Boundary conditions for the surface and bottom may be specified as Dirichlet conditions;

$$k = \left[\frac{u_*^s}{c_\mu^0} \right]^2$$
$$k = \left[\frac{u_*^b}{c_\mu^0} \right]^2 \qquad 6.4.8$$

where u_*^s and u_*^b are the surface and bottom friction velocities respectively, or Neumann conditions (zero flux);

$$\upsilon_k \frac{\partial k}{\partial z} = 0 \qquad z = \eta, \quad z = -H \qquad 6.4.9$$

Alternatively, 6.4.4 can be written in the Mellor-Yamada nomenclature in terms of turbulence kinetic intensity $q^2=2k$ (Mellor and Yamada (1982), eqn. 16) :

$$\frac{Dq^2}{Dt} - \frac{\partial}{\partial z} \upsilon_q \frac{\partial q^2}{\partial z} = 2(P + B - \varepsilon) \qquad 6.4.10$$

where $\upsilon_q = S_q q L$ with S_q=0.2 (Mellor and Yamada, 1982, p862).

The length scale is derived from the q^2L (or kL) equation:

$$\frac{Dq^2 L}{Dt} - \frac{\partial}{\partial z} \upsilon_L \frac{\partial q^2 L}{\partial z} = L\left[E_1 P + E_3 B - (1 + E_2 (\frac{L}{L_z})^2) \varepsilon \right] \qquad 6.4.11$$

where $E_1=E_3$=1.8 and E_2=1.33 and again $\upsilon_L = \upsilon_q = S_q q L$. Here L_z assumes a parabolic profile:

$$L_z = \kappa \frac{d_s d_b}{d_s + d_b} \qquad 6.4.12$$

or a triangular function :

$$L_z = \kappa \min(d_s, d_b) \qquad 6.4.13$$

where where κ = 0.4 is the Von Karman constant, d_b and d_s are the distances from the bottom and surface respectively. Additional wall proximity functions are outlined in Warner et al (2005). Boundary conditions for L are given by the Dirichlet condition:

$$L = \kappa(d_b + z_o^b) \qquad z = -H$$
$$L = \kappa(d_s + z_o^s) \qquad z = \eta \qquad 6.4.14$$

where and z_o^b and z_o^s are the bottom and surface roughness lengths associated with the bottom and top boundaries respectively.
The stability functions used are given by Mellor and Yamada (1982, Eq. 41, 42):

$$S_H = 3A_2 \frac{\lambda_1 - (\lambda_1 + \lambda_2)R_f}{1 - R_f} \quad 6.4.15$$

$$S_M = S_H \frac{A_1}{A_2} \frac{B_1(\lambda_1 - C_1) - \left[B_1(\lambda_1 - C_1) + 6(A_1 + 3A_{2)}\right]R_f}{B_1\lambda_1 - \left[B_1(\lambda_1 + \lambda_2) - 3A_1\right]R_f} \quad 6.4.16$$

where R_f is the flux Richardson number:

$$R_f = \frac{-B}{P} = \frac{S_H}{S_M} R_i \quad 6.4.17$$

and:

$$\lambda_1 = \frac{1}{3} - 2\frac{A_1}{B_1}$$
$$\lambda_2 = \frac{B_2}{B_1} + 6\frac{A_1}{B_1} \quad 6.4.18$$

with A_1=0.92, A_2=0.74, B_1=16.6, B_2=10.1 and C_1=0.08.

Alternatively, denoting the non-dimensional buoyancy parameter as (Mellor, 1992, Eq. 14.2):

$$G_H = -N^2 \frac{L^2}{q^2} \quad 6.4.19$$

with $-0.28 < G_H < 0.0233$ (Galperin et. al.,1988), then:

$$S_H = A_2 \frac{1 - 6A_1 / B_1}{1 - 3A_2 G_H (6A_1 + B_2)} \quad 6.4.20$$

$$S_M = \frac{COEF3 + S_H G_H COEF4}{1 - G_H COEF5} \quad 6.4.21$$

where COEF3=$A_1(1-3C_1-6A_1/B_1)$, COEF4=$18A_1A_1+9A_1A_2$ and COEF5=$9A_1A_2$.

The mixing coefficients V_z and K_z are then obtained via 6.4.1 and 6.4.2 using 6.4.10 to get the turbulence intensity q^2, 6.4.10 and 6.4.11 to get the length scale L (i.e. $L=q^2L/q^2$), 6.4.3 to get dissipation for the loss terms and 6.4.15 and 6.4.16 for the stability functions.

6.5 Mellor-Yamada 2.0

The Level 2.0 Mellor-Yamada scheme neglects the material derivative and diffusion term in 6.4.4 (or 6.4.9) resulting in:

$$P + B = \varepsilon \quad 6.5.1$$

The turbulent length scale is approximated by a triangular analytic form, where L decreases linearly towards the surface and bottom boundaries, i.e.

$$L = \kappa \min(d_b + z_o^b, d_s + z_o^s) \quad 6.5.2$$

The turbulent length scale is modified by the local turbulence of the flow via the method of Blackadar (1962):

$$L = \left[\frac{1}{L_s} + \frac{1}{L_o} \right]^{-1} \quad 6.5.3$$

where L_s is the result of eqn. 6.5.3 above and:

$$L_o = \gamma_o \frac{\int_{-H}^{\eta} z\sqrt{k}\,dz}{\int_{-H}^{\eta} \sqrt{k}\,dz} \quad 6.5.4$$

with $0.1 \le \gamma_o \le 0.4$.

The mixing coefficients V_z and K_z are then obtained via 6.4.1 and 6.4.2 using 6.5.1 to get the turbulence intensity q (using 6.4.3 to relate ε to q, 6.4.5 for P and 6.4.6 for B), 6.5.2 and 6.5.3 for L and 6.4.15 and 6.4.16 for the stability functions. If the vertical shear becomes very small or the flux Richardson number becomes less than a certain threshold (R_{crit}~0.19) then mixing coefficients revert to the background coefficients, V_{z0}, K_{z0}.

6.6 Modified Mellor Yamada 2.0

An alternate turbulence length scale parameterisation for the Mellor-Yamada 2.0 mixing scheme is included (Burchard et al, 1999; Eifler and Schrimpf, 1992; Demirov et al., 1998). This is based on a three layer system where surface and bottom mixed layers are intersected by a stably stratified interior layer. This scheme detects the height of each mixed layer by identifying the depth where the gradient of the vertical density profile exceeds a certain threshold (currently 0.01 kgm^{-2}). Alternatively the mixed layer heights may be located by identifying the depth where the turbulent kinetic energy, k, becomes less than a threshold (10^{-5} Wkg^{-1}). The length scale in the top and bottom mixed layers is given by:

$$L = \frac{\kappa z}{1 + \frac{\kappa z}{c_2 h}} (1 - R_f)^e \quad 6.6.1$$

where z is the distance from the surface or bottom, c_2 = 0.065 is a constant κ=0.4 is the Von Karmen constant, R_f is the flux Richardson number which accounts for stratification effects in the mixed layers and e = 1 to 3 is a tuning parameter. The surface and bottom roughness lengths (z_o^b and z_o^s) are added to the turbulence length scale in the top and bottom layers. Within the stably stratified region the mixing length scale is given by:

$$L = c_i \frac{k^{0.5}}{N} \qquad 6.6.2$$

where N is the Brunt-Vaisala frequency, and c_i is determined by matching the mixing length scale at the interfaces between the top or bottom mixed layer and the stratified region, i.e.

$$c_i = \left. \frac{L_m N}{k^{0.5}} \right|_{z=h_m} \qquad 6.6.3$$

where h_m is the depth of the interface between the top or bottom mixed layer and the stratified region and L_m is the mixing length scale at the bottom/top of the surface/bottom mixed layers as given by 6.6.1. A minimum mixing length scale is imposed in the stratified region, typically L_{min}=0.01m. An upper limit on the length scale is also imposed (Galperin, 1988):

$$L^2 \leq \frac{0.56k}{N^2} \quad for \;\; N^2 > 0 \qquad 6.6.4$$

This type of scheme displays better results than Mellor-Yamada-2.0 or k-ε for estuarine applications where a three layer type system exists. A large degree of flexibility exists in tuning this mixing model, where changes to z_o^s, L_{min}, e, background mixing coefficients V_{z0}, K_{z0} and density gradient threshold may affect the solution.

6.7 k-ε

The k-ε turbulence closure scheme is based on Burchard et al (1998). This mixing scheme retains the time derivative and diffusion in the k equation so that TKE is given by the solution to 6.4.4 with $\upsilon_k = V_z$. Also, The turbulent length scale is no longer parameterised algebraically but obtained from the solution of the ε equation:

$$\frac{D\varepsilon}{Dt} - \frac{\partial}{\partial z}\upsilon_\varepsilon \frac{\partial \varepsilon}{\partial z} = \frac{\varepsilon}{k}(c_{\varepsilon 1}P + c_{\varepsilon 3}B - c_{\varepsilon 2}\varepsilon) \qquad 6.7.1$$

also using the relation 6.4.3. This necessitates the inclusion of additional tracers into the model representing TKE and ε. Here υ_ε is the eddy diffusivity of ε and:

$$\upsilon_\varepsilon = \frac{V_z}{\sigma_\varepsilon} \qquad 6.7.2$$

with the Schmidt number σ_ε = 1.08. The constants $c_{\varepsilon 1}$ = 1.44, $c_{\varepsilon 2}$ = 1.92 and $c_{\varepsilon 3}$ = -0.4 (for unstable stratification $c_{\varepsilon 3}$ = 1.0). The stability functions are those of Galperin et al (1988) and are given by:

$$c_\mu = \frac{c_\mu^0 + 2.182\alpha_N}{1 + 20.4\alpha_N + 53.12\alpha_N^2} \qquad 6.7.3$$

$$c'_\mu = \frac{0.6985}{1 + 17.34\alpha_N} \qquad 6.7.4$$

for $-0.0466 \leq \alpha_N \leq 0.56$ where α_N is the non-dimensional buoyancy parameter (note equivalence to Eq. 6.4.19: G_H=-0.5α_N):

$$\alpha_N = \frac{L^2}{k} N^2 \qquad 6.7.5$$

Dirichlet boundary conditions for ε are given by (see GOTM Manual, www.gotm.net/pages/documentation/manual/pdf/a4.pdf Eq. 134).

$$\begin{aligned} \varepsilon &= (C_\mu^0)^3 \frac{k^{3/2}}{\kappa(d_b + z_o^b)} \qquad z = -H \\ \varepsilon &= (C_\mu^0)^3 \frac{k^{3/2}}{\kappa(d_s + z_o^s)} \qquad z = \eta \end{aligned} \qquad 6.7.6$$

The eddy viscosity and diffusivity are therefore obtained from 6.4.1 and 6.4.2 with k derived from the solution to 6.4.4, L derived from the solution to 6.7.1 using 6.4.3 to relate ε to L and the stability functions given by 6.7.3 and 6.7.4.

6.8 k-ω

The k-ω closure scheme is based on Umlauf *et al* (2003). This 2-equation turbulence model is similar to the k-ε model, but uses the 'turbulence frequency', ω, rather than dissipation, ε; the relation being:

$$\varepsilon = (C_\mu^0)^4 f_{c\mu} k\omega \qquad 6.8.1$$

where

$$f_{c\mu} = \begin{cases} 1 & \chi_k \leq 0 \\ \dfrac{1+680\chi_k^2}{1+400\chi_k^2} & \chi_k > 0, \end{cases} \qquad \chi_k = \frac{1}{\omega^3} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j} \qquad 6.8.2$$

and

$$f_{c\omega} = \frac{1+70\chi_\omega}{1+80\chi_\omega}, \qquad \chi_\omega = \left| \frac{W_{ij}^* W_{jl}^* S_{li}}{(C_\mu^0)^{12} \omega^3} \right| \qquad 6.8.3$$

where S_{ij} and W_{ij}^* are defined as:

$$S_{ij} = \frac{1}{2}\left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right]$$
$$W_{ij}^{*} = W_{ij} + \varepsilon_{ilj}\Omega_l \qquad 6.8.4$$
$$W_{ij} = \frac{1}{2}\left[\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right]$$

The ω equation is given by:

$$\frac{D\omega}{Dt} - \frac{\partial}{\partial z}\upsilon_\varepsilon \frac{\partial \omega}{\partial z} = \frac{\omega}{k}(c_{\omega 1}P + c_{\omega 3}B - c_{\omega 2}\frac{f_{c\omega}}{f_{c\mu}}\varepsilon) \qquad 6.8.5$$

with the $c_{\omega 1}$ = 0.52, $c_{\omega 2}$ = 0.8 and $c_{\omega 3}$ = -0.629. The last constant is only valid using the stability functions of Canuto et al (2001) with a steady state Richardson number of 0.25 (see Umlauf et al 2003, Table 6). The Schmidt numbers σ_k and σ_ω are both equal to 2 (Umlauf et al 2003, Table 1). It should be noted that the functions $f_{c\mu}$ and $f_{c\omega}$ are included to better parameterise the effects of buoyancy and rotating flows; if $f_{c\mu}$ and $f_{c\omega}$ = 1 the model of Wilcox (1988) is recovered.
The stability functions of Canuto et al (2001), also described in Burchard and Bolding (2001) are given by:

$$c_\mu = \frac{s_0 + s_1\alpha_N + s_2\alpha_N}{d_0 + d_1\alpha_N + d_2\alpha_M + d_3\alpha_N^2 + d_4\alpha_N\alpha_M + d_5\alpha_M^2}$$
$$c'_\mu = \frac{s_4 + s_5\alpha_N + s_6\alpha_N}{d_0 + d_1\alpha_N + d_2\alpha_M + d_3\alpha_N^2 + d_4\alpha_N\alpha_M + d_5\alpha_M^2} \qquad 6.8.6$$

where:

$$s_0 = \frac{3}{2}\lambda_1\lambda_5^2$$
$$s_1 = -\lambda_4(\lambda_6 + \lambda_7) + 2\lambda_4\lambda_5(\lambda_1 - \frac{1}{3}\lambda_2 - \lambda_3) + \frac{3}{2}\lambda_1\lambda_5\lambda_8$$
$$s_2 = -\frac{3}{8}\lambda_1(\lambda_6^2 - \lambda_7^2), \quad s_4 = 2\lambda_5, \quad s_5 = 2\lambda_4 \qquad 6.8.7$$
$$s_6 = \frac{2}{3}\lambda_5(3\lambda_3^2 - \lambda_2^2) - \frac{1}{2}\lambda_5\lambda_1(3\lambda_3 - \lambda_2) + \frac{3}{4}\lambda_1(\lambda_6 - \lambda_7)$$

and;

$$
\begin{aligned}
d_0 &= 3\lambda_5^2 \\
d_1 &= \lambda_5(7\lambda_4 + 3\lambda_8) \\
d_2 &= \lambda_5^2(3\lambda_3^2 - \lambda_2^2) - \frac{3}{4}(\lambda_6^2 - \lambda_7^2) \\
d_3 &= \lambda_4(4\lambda_4 + 3\lambda_8) \\
d_4 &= \lambda_4(\lambda_2\lambda_6 - 3\lambda_3\lambda_7 - \lambda_5(\lambda_2^2 - \lambda_3^2)) + \lambda_5\lambda_8(3\lambda_3^2 - \lambda_2^2) \\
d_5 &= \frac{1}{4}(\lambda_2^2 - 3\lambda_3^2)(\lambda_6^2 - \lambda_7^2)
\end{aligned}
\qquad 6.8.8
$$

with;

$$
(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7, \lambda_8) = (\boldsymbol{0.107, 0.0032, 0.0864, 0.12, 11.9, 0.4, 0, 0.48)} \qquad 6.8.9
$$

The buoyancy and shear numbers are given by:

$$
\begin{aligned}
\alpha_N &= 4\frac{k^2}{\varepsilon^2}N^2 \\
\alpha_M &= 4\frac{k^2}{\varepsilon^2}\left[(\frac{\partial u_1}{\partial z})^2 + (\frac{\partial u_2}{\partial z})^2\right]
\end{aligned}
\qquad 6.8.10
$$

The limits given by Warner et. al (2005), 40b in conjunction with Table 4 are applied.

Dirichlet boundary conditions for ωare given by:

$$
\begin{aligned}
\omega &= \frac{k^{1/2}}{C_\mu^0 \kappa(d_b + z_o^b)} \qquad z = -H \\
\omega &= \frac{k^{1/2}}{C_\mu^0 \kappa(d_s + z_o^s)} \qquad z = \eta
\end{aligned}
\qquad 6.8.11
$$

The eddy viscosity and diffusivity are therefore obtained from 6.4.1 and 6.4.2 with k derived from the solution to 6.4.4, L derived from the solution to 6.8.5 using 6.4.3 and 6.8.1 to relate ω to ε and L and the stability functions given by 6.8.6.

6.9 W88

The W88 model of Wilcox (1988) is the same as the k-ω model, but using $f_{c\mu} = f_{c\omega} = 1.0$.

6.10 Stability functions

The default stability functions for the various 2-equation turbulence closure models are listed in Table 6.10.1.

Table 6.10.1 : Default stability functions.

Closure scheme	Stability function	Argument
MY2.5	Mellor (1992)	G_H
k-ε	Galperin et. al. (1988)	α_N
k-ω	Canuto et. al. (2001) model A	$4\alpha_N/(c_\mu^0)^6,\ 4\alpha_M/(c_\mu^0)^6$
W88	Canuto et. al. (2001) model A	$4\alpha_N/(c_\mu^0)^6,\ 4\alpha_M/(c_\mu^0)^6$

The arguments to stability functions are the non-dimensional shear and buoyancy numbers, given by Burchard et. al. (1998, Eq. 27, 28) as $\alpha_N = L^2N^2/k$ and $\alpha_M = L^2M^2/k$, with N^2 and M^2 given by eq. 6.3.4 and 6.3.5 respectively. Note the argument to the k-ω and W88 stability functions is $4k^2N^2/\varepsilon^2$ and $4k^2M^2/\varepsilon^2$ (Canuto et. al., 2001, eq. 14c), which using eq. 6.4.3 are equal to the arguments listed in Table 6.10.1.

Note the equivalence of parameters in k-ε and Mellor Yamada formulations:

$q = \sqrt{2k}$
$c_\mu = \sqrt{2}S_M$ and $c'_\mu = \sqrt{2}S_H$
$G_M = 0.5\alpha_M$ and $G_H = -0.5\alpha_N$

Alternative closure schemes may be substituted for each turbulence model. In addition to the schemes listed above, the following are available.

6.10.1 Canuto et. al. (2001) model B

These stability functions have the same form as Eq. 6.8.6, except the following constants are used:

$$(\lambda_1,\lambda_2,\lambda_3,\lambda_4,\lambda_5,\lambda_6,\lambda_7,\lambda_8) = (0.127, 0.00336, 0.0906, 0.101, 11.2, 0.4, 0, 0.318) \quad 6.10.1$$

6.10.2 Kantha and Clayson (1994)

These functions are described in Warner et. al. (2005, Eq. 30, 31), and are given by:

$$SH = \frac{A_2(1-6A_1/B_1)}{1-3A_2G_H(6A_1+B_2(1-C_3))} \quad 6.10.2$$

$$SM = \frac{B_1^{-1/3} + (18A_1A_1 + 9A_1A_2(1-C_2))S_HG_H}{1-9A_1A_2G_H} \quad 6.10.3$$

With A_1=0.92, A_2=0.74, B_1=16.6, B_2=10.1, C_2=0.7, C_3=0.2.

The limits given by Warner et. al (2005), Eq. 33 in conjunction with Table 3 are applied, i.e.

$$G_H = \frac{\tilde{G}_H - (\tilde{G}_H - G_{H\_crit})^2}{\tilde{G}_H + G_{H0} - 2G_{H\_crit}} \quad 6.10.4$$

with $G_{H\_\min} < G_H < G_{H0}$ where $\tilde{G}_H$ is given by Eq. 6.4.19 and G_{H_crit} =0.02, G_{H0} =0.0233 and $G_{H\_\min}$ =-0.28.

6.10.3 Munk and Anderson (1948)

This simple stability function uses constant coefficients, e.g., Burchard et. al. (1999. Eq. 2.73, 2.74).

$$c_\mu = c_\mu^0 \quad 6.10.5$$

$$c_\mu^{'} = c_\mu / P_r \quad 6.10.6$$

$$P_r = \begin{cases} P_r^0 \frac{(1+3.33Ri)^{3/2}}{(1+10Ri)^{1/2}} & for\ Ri \geq 0 \\ P_r^0 & for\ Ri < 0 \end{cases} \quad 6.10.7$$

where Ri is the gradient Richardson number (Eq. 6.3.3) and $P_r^0 = 0.7143$ is the neutral Prandtl number.

6.10.4 Eifler and Schrimpf (1992)

This stability function is of the form (Burchard et. al., 1999, Eq. 2.76, 2.77):

$$c_\mu = 0.5 \quad 6.10.8$$

$$c_\mu^{'} = c_\mu \frac{1}{P_r^0}(1-R_f)^{1/2} \quad 6.10.9$$

$$(1-R_f) = (\sqrt{\tilde{R}_i^2 + 1} - \tilde{R}_i)^2 \quad 6.10.10$$

with $R_i = R_i / 2P_r$ and $0.18 < c_\mu^{'} / c_\mu < 2$.

6.10.5 Schumann and Gerz (1995)

This stability function has the form (Umlauf et. al., 2003, Eq. 27):

$$c_\mu = c_\mu^0 \quad 6.10.11$$

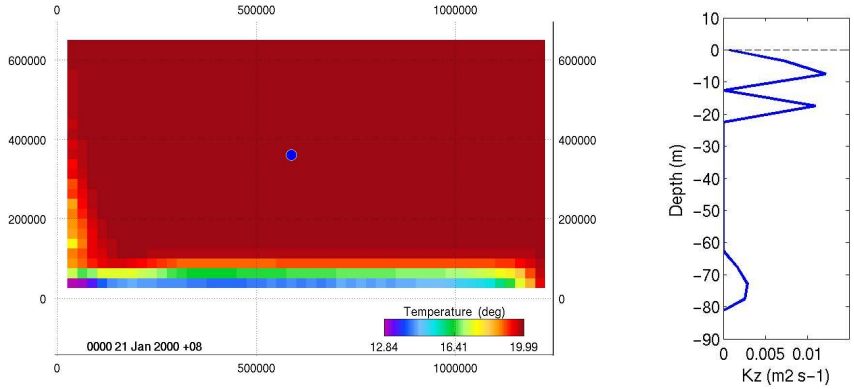
$$c_\mu^{'} = c_\mu^0 / P_{rt} \quad 6.10.12$$

$$P_{rt} = P_{rt}^0 \exp(\frac{-Ri}{P_{rt}^0 Ri^\infty}) - \frac{Ri}{Ri^\infty} \quad 6.10.13$$

where $P_{rt}^0 = 0.74$ and $Ri^\infty = 0.25$.

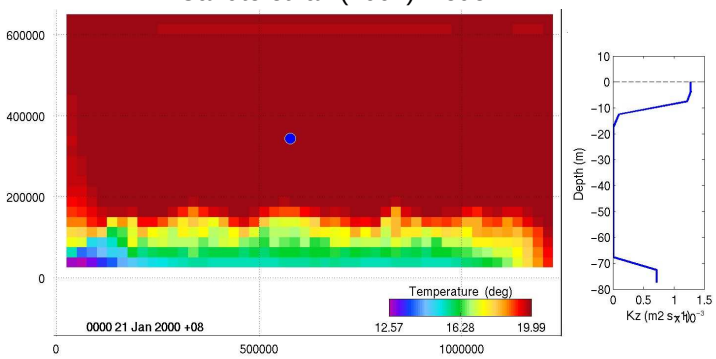
Stability functions have a significant impact on vertical mixing. Solutions in a closed basin subject to a westerly wind, with initial stratification of 20°C from 0-30 m and 10°C below 40 m are presented below for all closure schemes and stability functions. Bottom depth is 50 m on the southern coast and 110 m on the northern coast. Surface temperature and K_z profile are presented at 20 days for the various closure schemes with differing stability functions.

Mellor-Yamada 2.0.

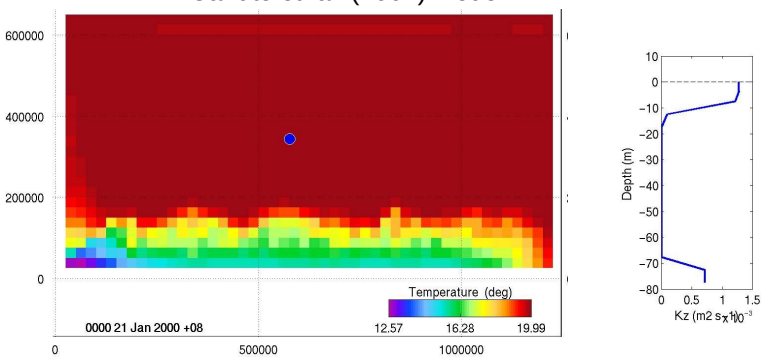


Mellor Yamada 2.5

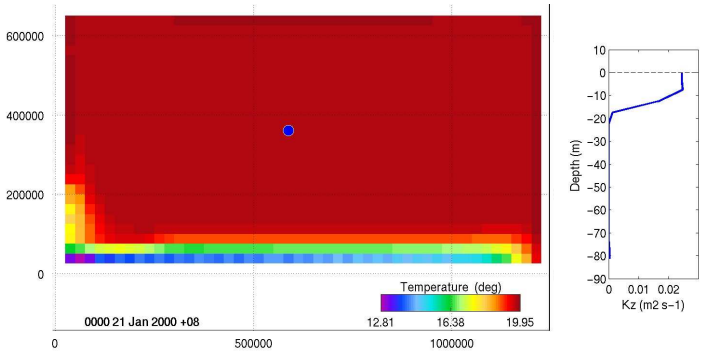
Canuto et. al. (2001) Model A



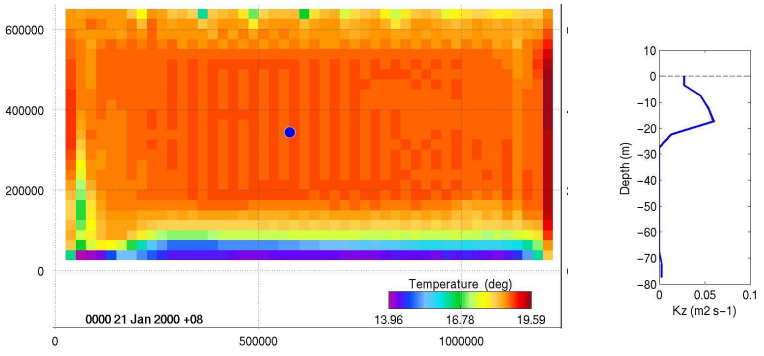
Canuto et. al. (2001) Model B



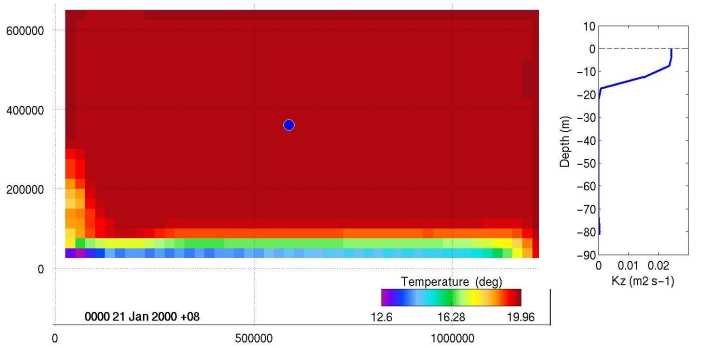
Galperin et. al. (1988)



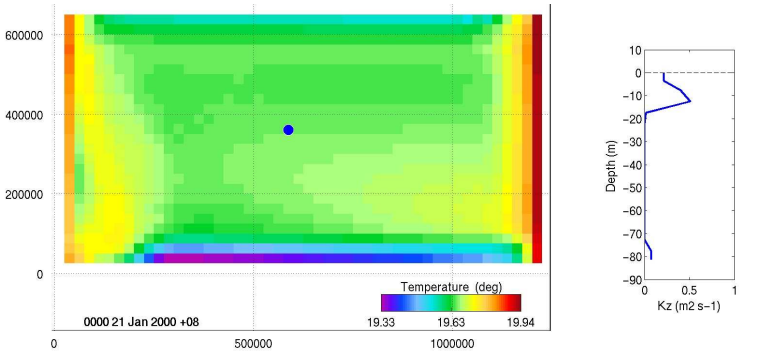
Kantha and Clayson (1994)



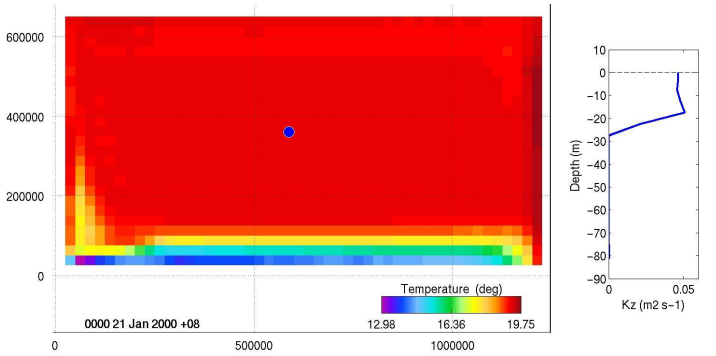
Mellor (1992)



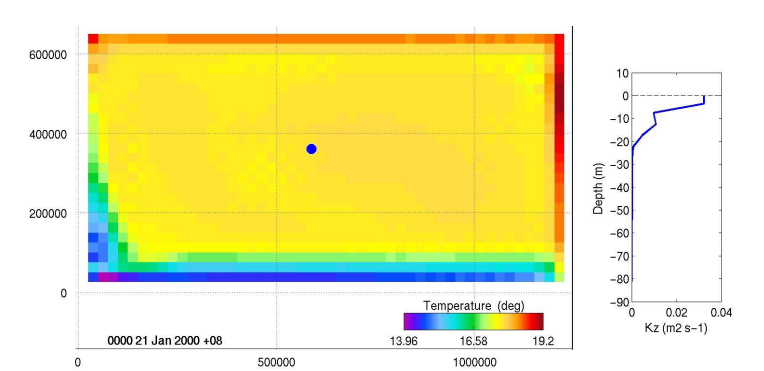
Munk and Anderson (1948)



Eifler and Schrimpf (1992)

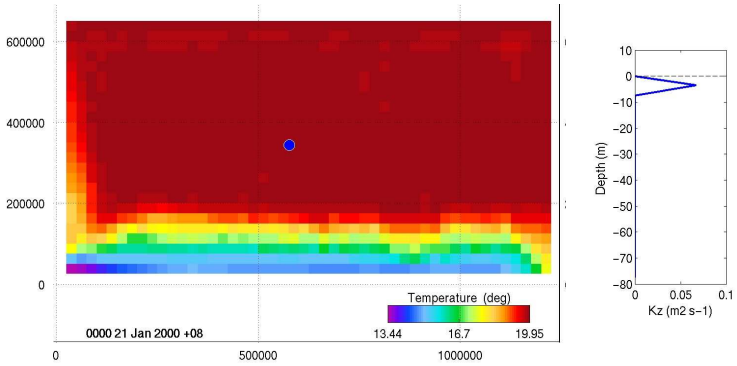


Schumann and Gerz (1995)

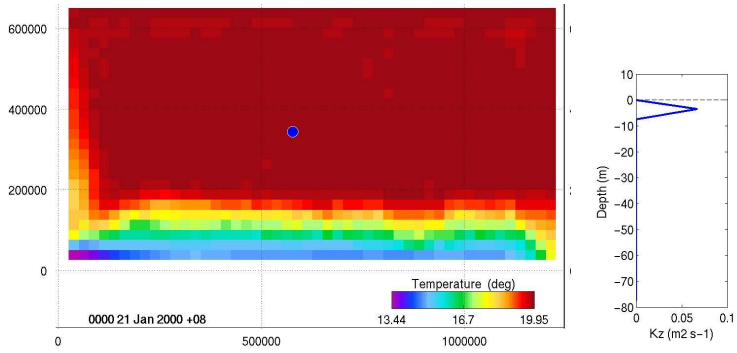


k-ε

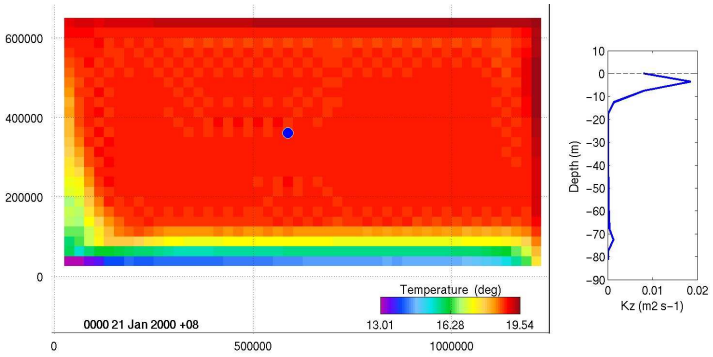
Canuto et. al. (2001) Model A



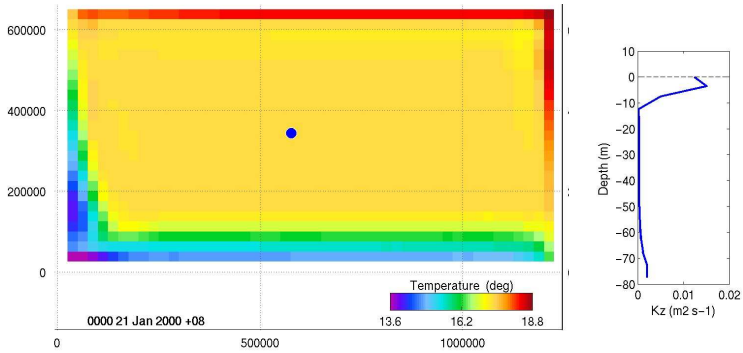
Canuto et. al. (2001) Model B



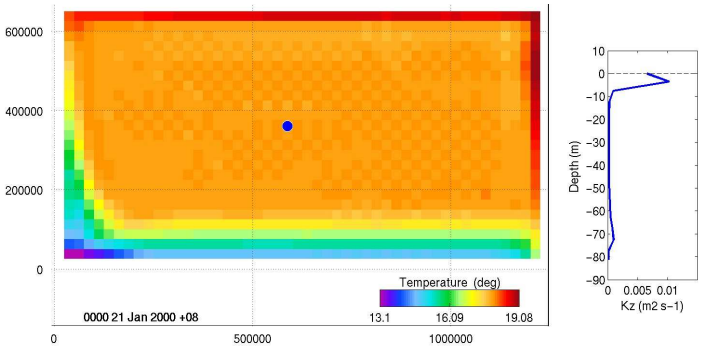
Galperin et. al. (1988)



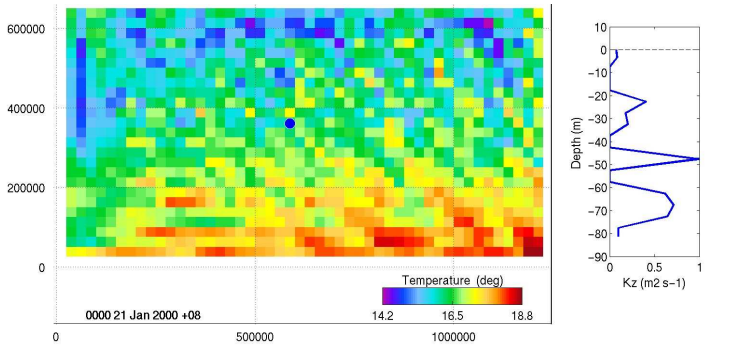
Kantha and Clayson (1994)



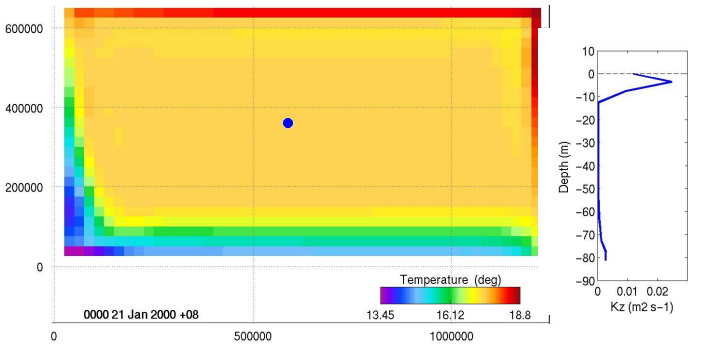
Mellor (1992)



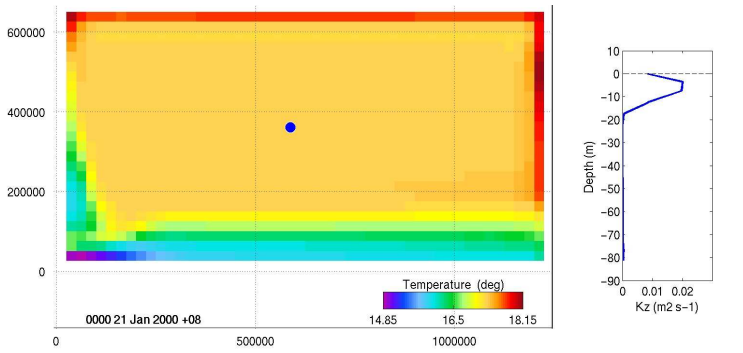
Munk and Anderson (1948)



Eifler and Schrimpf (1992)

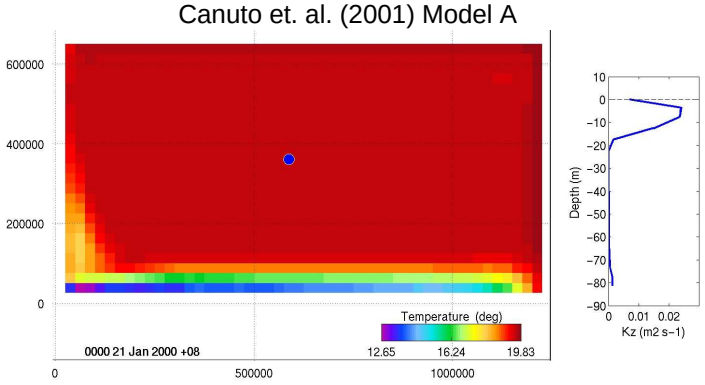


Schumann and Gerz (1995)

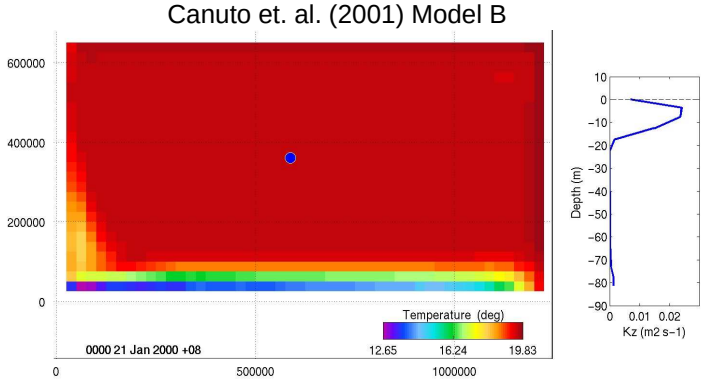


k-ω

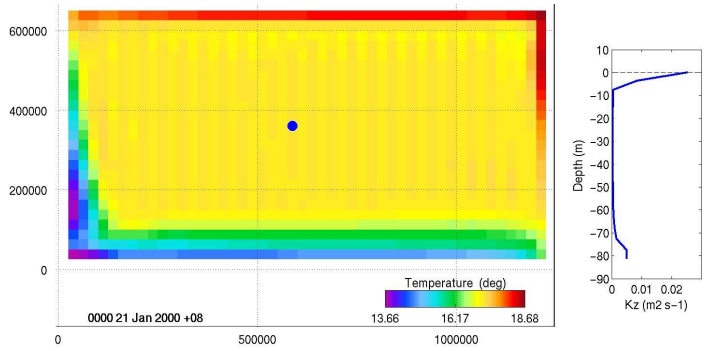
Canuto et. al. (2001) Model A



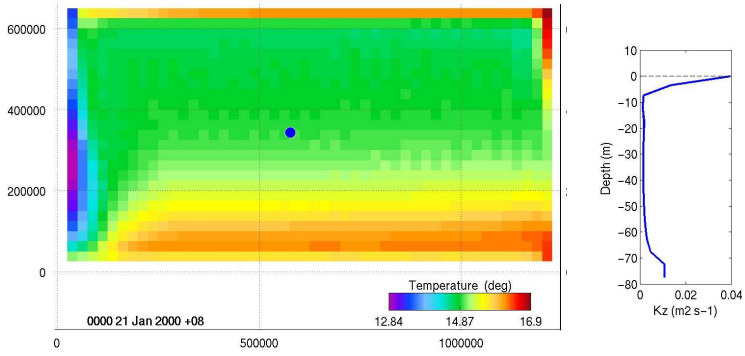
Canuto et. al. (2001) Model B



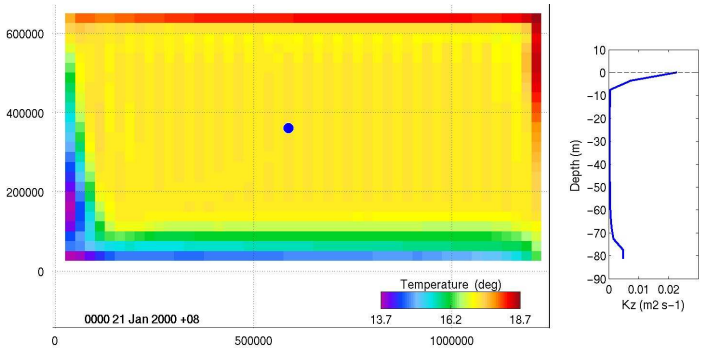
Galperin et. al. (1988)



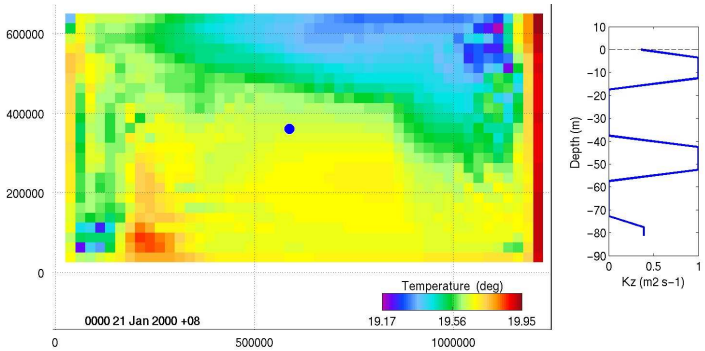
Kantha and Clayson (1994)



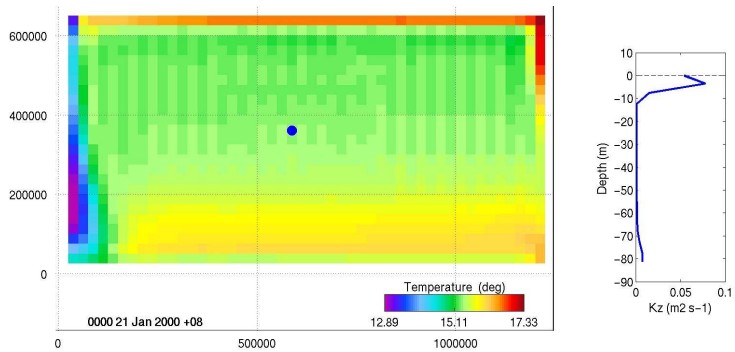
Mellor (1992)



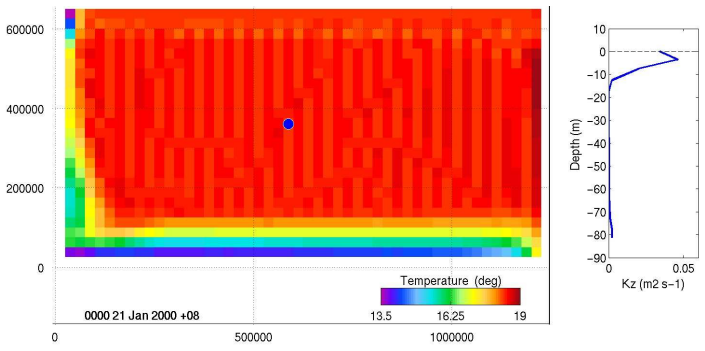
Munk and Anderson (1948)



Eifler and Schrimpf (1992)

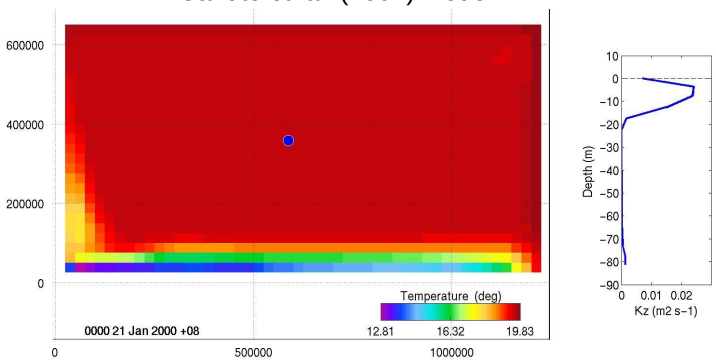


Schumann and Gerz (1995)

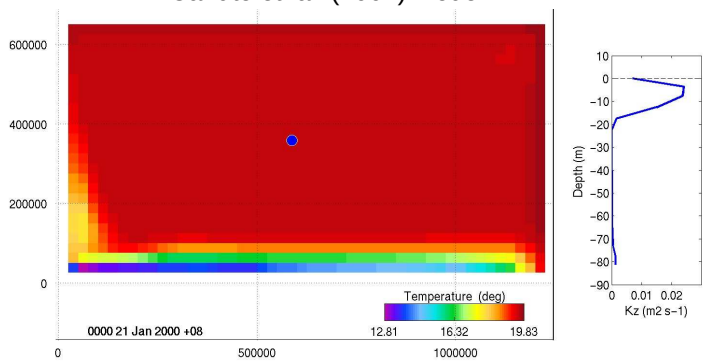


W88

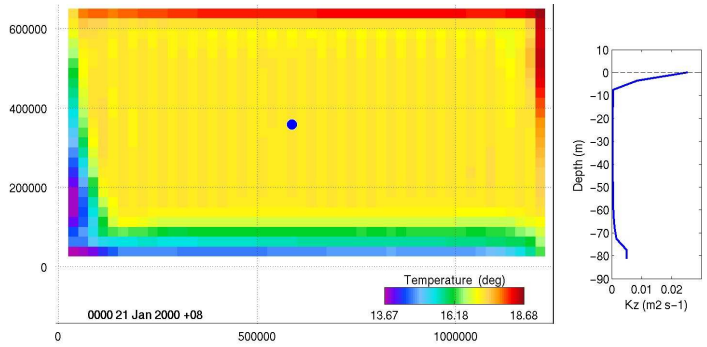
Canuto et. al. (2001) Model A



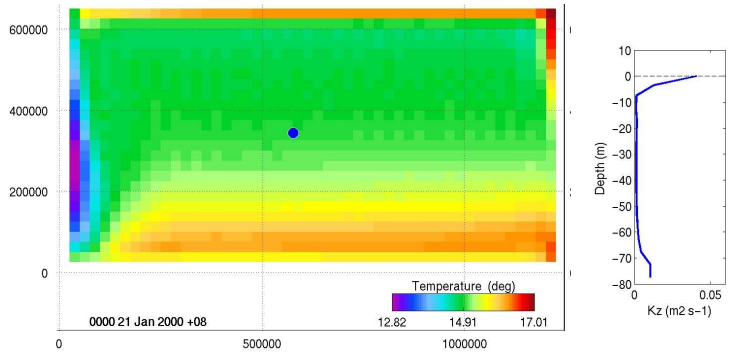
Canuto et. al. (2001) Model B



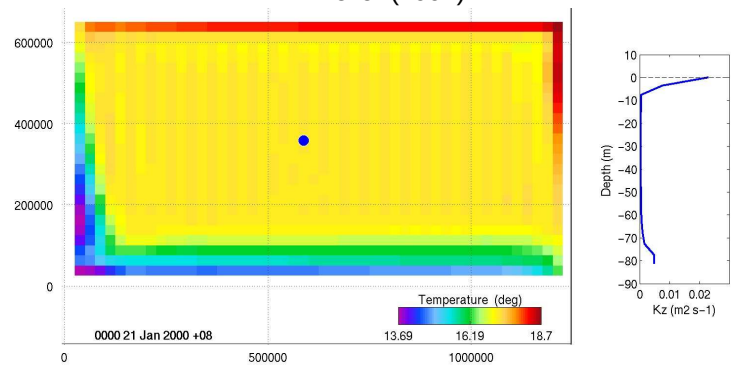
Galperin et. al. (1988)



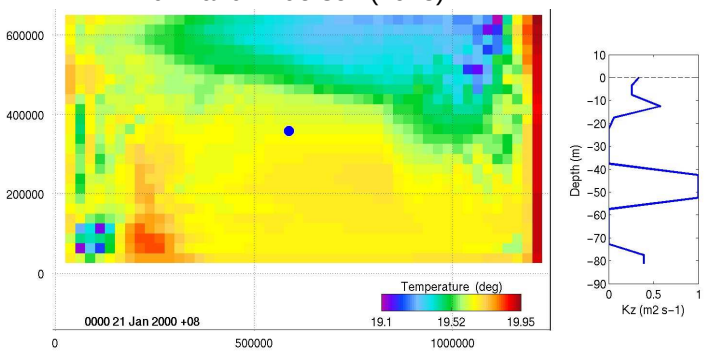
Kantha and Clayson (1994)



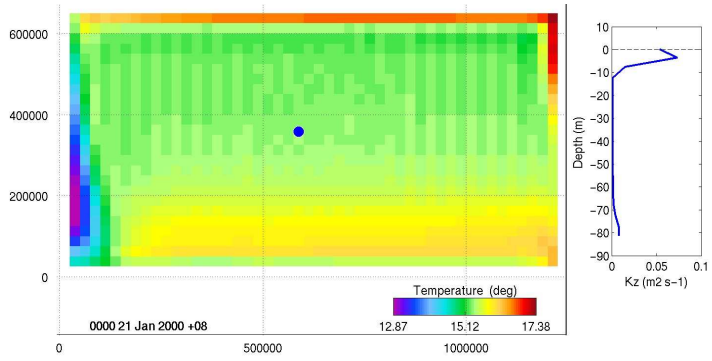
Mellor (1992)



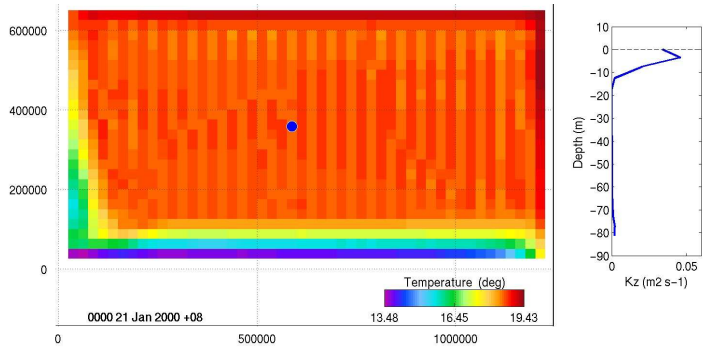
Munk and Anderson (1948)



Eifler and Schrimpf (1992)



Schumann and Gerz (1995)



6.11 Limits

The length scale is bounded by an upper limit for stable stratification (Galperin et al, 1988, eqn 22):

$$L^2 \le \frac{0.56k}{N^2} \qquad or \qquad L \le \frac{0.53q}{N} \qquad N^2 > 0 \tag{6.11.1}$$

The upper limit on L is equivalent to a lower limit on the non-dimensional buoyancy parameter, G_H. Upper and lower bounds for this parameter are given by Galperin et al (1988). Eq. 29 & 30:

$$-0.28 \le G_H \le 0.0233 \tag{6.11.1}$$

where the lower limit applies for stable stratification. The length scale limitation also translates to a lower limit for ε in the k-ε model (Burchard et al, 1998, Eq. 16) :

$$\varepsilon^2 \ge 0.045k^2N^2 \qquad N^2 > 0 \tag{6.11.2}$$

and a lower limit for ω in the k-ω model (Warner at al, 2005, Eq. 43):

$$\omega \ge \frac{N}{\sqrt{0.56}(c_\mu^0)} \qquad N^2 > 0 \tag{6.11.3}$$

The k-ε, k-ω and Mellor-Yamada 2.5 mixing schemes all require minimum values of k and ε to be specified. Various sources supply differing values, summarized in Table 6.11.1. Note that given a minimum ε, then minimum ω is computed via Eqn. 6.8.1 with $f_{c\mu}$=1 for the k-ω scheme:

$$\omega_{\min} = \frac{\varepsilon_{\min}}{(c_\mu^0)^4 k_{\min}} \tag{6.11.4}$$

and minimum length scale for MY2.5 is computed from Eqn. 6.4.3:

$$L_{\min} = \frac{(c_\mu^0)^3 k_{\min}^{3/2}}{\varepsilon_{\min}} \tag{6.11.5}$$

Minimum length scale can be over-ridden using the LMIN parameter in the input file.

Table 6.11.1 : Minimum TKE and dissipation

Scheme	Variable	Value	Reference
k-ε	**k**	**7.6×10^{-6}** m^2s^{-2}	Burchard (1998), eqn. 17, Warner (2005), Table 1
	ε	1.0×10^{-12} m^2s^{-3}	Warner (2005) Table 1, GOTM code
	ε	**5.0×10^{-10}** m^2s^{-3}	CRS code – unknown source - original MECO code
k-ω	**k**	**7.6×10^{-6}** m^2s^{-2}	Burchard (1998), eqn. 17, Warner (2005), Table 1
	ω	**1.0×10^{-12}** s^{-1}	Warner (2005) Table 1
	ε	**7.3×10^{-19}** m^2s^{-3}	Warner (2005) Table 1 and eqn 6.9.4
	ε	5.0×10^{-10} m^2s^{-3}	CRS code – unknown source - original MECO code
MY2.5	**k**	**1.0×10^{-8}** m^2s^{-2}	GOTM code
	ε	**1.0×10^{-12}** m^2s^{-3}	GOTM code
	k	7.6×10^{-6} m^2s^{-2}	Burchard (1998), eqn. 17
	ε	5.0×10^{-10} m^2s^{-3}	CRS code – unknown source - original MECO code
	k	5.0×10^{-6} m^2s^{-2}	Warner (2005) Table 1
	ε	9.6×10^{-7} m^2s^{-3}	Warner (2005) Table 1, kL=1×10^{-8} and eqn 6.4.3

Bold values are used as defaults. Default values may be over-ridden using MIN_TKE and MIN_DISS parameters in the input file.

6.12 Implicit Solution Method

The vertical diffusion equation is written as:

$$\frac{k^{n+1}-k^{n}}{dt}+adv=hdiff+\frac{\partial}{\partial z}\upsilon\frac{\partial k^{n+1}}{\partial z} \qquad 6.12.1$$

or

$$k^{n+1}=k^{n}+dt.f(k)+dt\frac{\partial}{\partial z}\upsilon\frac{\partial k^{n+1}}{\partial z} \qquad 6.12.2$$

where k may be k, q^2, q^2l, ε or ω depending on the closure scheme used. Using fractional steps to split the equation; first solve for f(k):

$$\tilde{k}=k^{n}+dt.f(k) \qquad 6.12.3$$

Then solve for vertical diffusion implicitly:

$$k^{n+1}=\tilde{k}+dt\frac{\partial}{\partial z}\upsilon\frac{\partial k^{n+1}}{\partial z} \qquad 6.12.4$$

or more generally:

$$k^{n+1}=\tilde{k}+dt\frac{\partial}{\partial z}\upsilon\frac{\partial k^{n+1}}{\partial z}+dtS_o-dtS_i \qquad 6.12.5$$

where S_o is a source term and S_i is a sink term. If the surface is denoted layer 0 and the bottom layer nz, the discretization becomes (noting positive z direction is up):

$$k^{n+1} = \tilde{k} + \frac{dt}{dz}\left[\frac{\upsilon_{k+1}}{dzz_{k+1}}(k_{k+1} - k_k) - \frac{\upsilon_k}{dzz_k}(k_k - k_{k-1})\right] + dtS_o - dtS_i$$

$$k_k \frac{dz}{dt} + dzS_i + k_k \frac{\upsilon_{k+1}}{dzz_{k+1}} + k_k \frac{\upsilon_k}{dzz_k} - k_{k+1}\frac{\upsilon_{k+1}}{dzz_{k+1}} - k_{k-1}\frac{\upsilon_k}{dzz_k} = \tilde{k}\frac{dz}{dt} + dzS_o$$

$$k_k\left[\frac{dz}{dt} + \frac{dzS_i}{k_k} + \frac{\upsilon_{k+1}}{dzz_{k+1}} + \frac{\upsilon_k}{dzz_k}\right] - k_{k+1}\frac{\upsilon_{k+1}}{dzz_{k+1}} - k_{k-1}\frac{\upsilon_k}{dzz_k} = \tilde{k}\frac{dz}{dt} + dzS_o \qquad 6.12.6$$

$$let \quad C_p = -\frac{\upsilon_{k+1}}{dzz_{k+1}}, \quad C_m = -\frac{\upsilon_k}{dzz_k}, \quad rhs = \tilde{k}\frac{dz}{dt} + dzS_o$$

$$k_k\left[\frac{dz}{dt} + \frac{dzS_i}{k_k} - C_p - C_m\right] + k_{k+1}C_p + k_{k-1}C_m = rhs$$

This forms a tri-diagonal linear system of equations:

$$\begin{bmatrix} C_0 & C_{m0} & 0 & . & 0 \\ C_{p1} & C_1 & C_{m1} & . & 0 \\ . & . & . & . & . \\ 0 & . & C_{pnz-1} & C_{nz-1} & C_{mnz-1} \\ 0 & . & 0 & C_{pnz} & C_{nz} \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ . \\ k_{nz-1} \\ k_{nz} \end{bmatrix} = \begin{bmatrix} rhs_1 - flux_{top} \\ rhs_2 \\ . \\ rhs_{nz-1} \\ rhs_{nz} + flux_{bot} \end{bmatrix} \qquad 6.12.7$$

where $C_k = \frac{dz}{dt} + \frac{dzS_i}{k_k} - C_{pk} - C_{mk}$

This system is solved by simplified Gaussian elimination (Thomas algorithm). The equations for the surface and bottom must contain flux (Neumann) or prescribed (Dirichlet) boundary conditions. For flux surface and bottom conditions C_{p0}=0 and C_{mnz}=0 respectively, and any surface or bottom fluxes are included on the rhs. Positive fluxes are out of and into the water column at the top and bottom layer respectively. Note that the sink term is multiplied by k^{n+1}/k^n for greater stability. This approach stems primarily from the turbulence diffusion equations for k, ε, q^2 or q^2l (e.g. Burchard et al, 1998, p10,547).

This method of solving the implicit vertical diffusion is also used for the solution to tracer and momentum vertical diffusion equations. Advection diffusion equations are solved using the method of fractional steps, e.g.

For any tracer, T, using forward Euler time stepping:

1. Solve advection and horizontal diffusion equation:

$$\tilde{T} = T^n + f(T)$$

2. Get intermediate solution from vertical diffusion:

$$\hat{T} = \tilde{T} + vdiff$$

$$dT = \hat{T} - \tilde{T}$$

3. Get the solution at the forward time-step

$$T^{n+1} = \tilde{T} + dT = \tilde{T} + (\hat{T} - \tilde{T}) = \hat{T}$$
$$= \tilde{T} + vdiff$$
$$= T^{n} + f(T) + \frac{\partial}{\partial z} K \frac{\partial T^{n+1}}{\partial z}$$

where f(T) is the advection and horizontal diffusion.

For momentum, u_1 or u_2 using leapfrog time-stepping:

1. Sove for advection, horizontal diffusion, pressure and Coriolis

$$\tilde{u} = u^{n-1} + f(u)$$

2. Get the intermediate solution from vertical diffusion

$$\hat{u} = u^{n-1} + vdiff$$
$$du = \hat{u} - u^{n-1}$$

3. Get the solution at the forward time-step

$$u^{n+1} = \tilde{u} + du = \tilde{u} + \hat{u} - u^{n-1}$$
$$= \tilde{u} + (u^{n-1} + vdiff) - u^{n-1} = \tilde{u} + vdiff$$
$$= u^{n-1} + f(u) + \frac{\partial}{\partial z} K \frac{\partial u^{n+1}}{\partial z}$$

Note : this could be solved using $\hat{u} = \tilde{u} + vdiff$ as is done with tracers, however, the solution appears more stable using u^{n-1} rather than $\tilde{u}$ on the RHS.

6.13 Wave enhanced mixing

Wave enhanced vertical mixing for the k-ε and k-ω schemes is included by using surface boundary conditions applicable to shear-free boundary-layers with injection of tke (e.g. Craig and Banner, 1994). The Dirichlet condition for dissipation for shear-free boundary-layers with injection of tke for the k-ε model is (GOTM Manual, Eq. 136):

$$\varepsilon = (c_\mu^0)^3 K^{3/2} L^{-1} (z + z_{os})^{3/2a-1} \qquad 6.13.1$$

and for the k-ω model (GOTM Manual, Eq. 140 with $p = n = -1$, $m = 0.5$, Table 8):

$$\omega = (c_\mu^0)^{-1} K^{1/2} L^{-1} (z + z_{os})^{1/2a-1} \qquad 6.13.2$$

where $a = -2.53$ and $L = 0.25$ for k-ω (Jones and Monosmith, 2008) and $a = -17.78$ and $L = 0.025$ for k-ε (Umlauf and Burchard, 2003, Table 6). The function *K* is given by (GOTM Manual, Eq. 128 or Jones and Monosmith. 2008, Eq. 19):

$$K = \left(-\frac{\sigma_k wf}{c_\mu aL} \right)^{2/3} \frac{1}{z_{0s}^a} \qquad 6.13.3$$

With the wave energy flux given by (Jones and Monosmith, 2008, Eq. 9):

$$wf = \alpha (u_*^2 + v_*^2)^{3/2} \qquad 6.13.4$$

where u_* and v_* are shear velocities an the x and y directions respectively and $\alpha \sim 100$ is the wave parameter. The Neumann conditions (zero flux) for tke becomes:

$$\frac{\upsilon_k}{\sigma_k} \frac{\partial k}{\partial z} = -wf \qquad z = \eta \qquad 6.13.5$$

Where $\sigma_k = 1$ for k-ε and 2 for k-ω (Umlauf and Burchard, 2003, Table 6). The parameter z_{0s} is the surface roughness including the wave amplitude, and is given by some factor of the significant wave height. Jones and Monosmith (2008) achieved best results with the k-ω scheme using $z_{0s} = 1.3 H_s$ and $\alpha = 60$. Wave parameters are summarized for the k-ε and k-ω schemes in Table 6.13.1.

Table 6.13.1: Wave Parameters

Parameter	k-ε	k-ω
a	-17.78	-2.53
L	0.025	0.25
σ_k	1	2

7. Sigma Vertical Coordinates

7.1 Introduction

The σ equations are related to the 'z' equations via the transformation:

$$(\xi_1^*, \xi_2^*, \sigma, t^*) = (\xi_1, \xi_2, \frac{z-\eta}{H+\eta}, t) \qquad 7.1.1$$

The equations for continuity, 3D momentum in the ξ_1 direction and tracers transformed to the σ system are included below (with the * dropped for clarity).

Continuity:

$$\frac{1}{h_1 h_2}\left[\frac{\partial D_s u_1 h_2}{\partial \xi_1} + \frac{\partial D_s u_2 h_1}{\partial \xi_2}\right] + \frac{\partial \omega}{\partial \sigma} = 0 \qquad 7.1.2$$

where $D_s(\xi_1,\xi_2)=D=H+\eta$ is the total depth and ω is the velocity perpendicular to σ surfaces:

$$\omega = w - \frac{1}{h_1 h_2}\left[h_2 u_1(\sigma\frac{\partial D_s}{\partial \xi_1} + \frac{\partial \eta}{\partial \xi_1}) + h_1 u_2(\sigma\frac{\partial D_s}{\partial \xi_2} + \frac{\partial \eta}{\partial \xi_2})\right] - \frac{\partial}{\partial t}(\sigma D_s + \eta) \qquad 7.1.3$$

Momentum:

$$\frac{\partial D_s u_1}{\partial t} + \frac{1}{h_1^2 h_2}\left[\frac{\partial (u_1^{\,2} D_s h_1 h_2)}{\partial \xi_1} + \frac{\partial (u_1 u_2 D_s h_1^{\,2})}{\partial \xi_2}\right] + \frac{\partial(\omega u_1)}{\partial \sigma} = -\frac{D_s}{h_1 \rho}\frac{\partial P}{\partial \xi_1} + f D_s u_2 + \frac{u_1^2 D_s}{h_1^2}\frac{\partial h_1}{\partial \xi_1} + \frac{u_2^2 D_s}{h_1 h_2}\frac{\partial h_2}{\partial \xi_1}$$
$$+ D_s \psi_1' + \frac{1}{D_s}\frac{\partial}{\partial \sigma}\left[V_z \frac{\partial u_1}{\partial \sigma}\right] \qquad 7.1.4$$

where the horizontal diffusion terms are now:

$$D\psi_1' = \frac{1}{h_1 h_2}\left[\frac{\partial}{\partial \xi_1}(h_2 D_s \tau_{11}) + \frac{\partial}{\partial \xi_2}(h_1 D_s \tau_{21})\right] + \frac{D_s}{h_1 h_2}\left[\tau_{21}\frac{\partial h_1}{\partial \xi_2} - \tau_{22}\frac{\partial h_2}{\partial \xi_1}\right] \qquad 7.1.5$$

Tracers:

$$\frac{\partial D_s T}{\partial t} + \frac{1}{h_1 h_2}\left[\frac{\partial (u_1 T D_s h_2)}{\partial \xi_1} + \frac{\partial (u_2 T D_s h_1)}{\partial \xi_2}\right] + \frac{\partial(\omega T)}{\partial \sigma} = \frac{1}{h_1 h_2}\left[\frac{\partial}{\partial \xi_1}(D_s \frac{h_2}{h_1} V_h \frac{\partial T}{\partial \xi_1}) + \frac{\partial}{\partial \xi_2}(D_s \frac{h_1}{h_2} V_h \frac{\partial T}{\partial \xi_2})\right] +$$
$$\frac{1}{D_s}\frac{\partial}{\partial \sigma}\left[K_z \frac{\partial T}{\partial \sigma}\right] \qquad 7.1.6$$

See Blumberg and Herring (1987) for the full σ equations in curvilinear coordinates. It is observed that these equations are similar to those in the 'z' system (eqns 2.1.4 to 2.1.7) except for the inclusion of the depth term, D_s. The equations in SHOC are discretized according to eqns. 7.1.2 to 7.1.6 and the model is thus configured to operate in the σ system if D_s = D, the total water depth. The 'z' model is recovered by setting D_s=1. Extra terms in the σ model pressure term (eqn

2.3.8) exist which also need to be included for the σ model. Also, the vertical indexing is modified slightly in the σ model so that vertical loops include all the σ layers (as opposed to layers encompassing the bottom to the free surface in the 'z' model).

The number of layers configured for the 'z' case is equated to the number of sigma levels, and the distribution of these layers is generated by SHOC such that a logarithmic distribution exists at the surface and bottom and a linear distribution in the interior. The netCDF input data is always on 'z' layers and is linearly interpolated onto the σ layers (a smoothing filter in the vertical is also applied). The bathymetry is checked to ensure no extreme gradients are encountered; if the bottom slope becomes greater than 0.07 then the bathymetry is smoothed (up to a maximum of 5 passes) until the gradient becomes less than 0.07 (0.1 approximates an upper limit of bottom gradients in the ocean; Mellor and Blumberg (1985)). The sigma layers converge at the coast, and this can lead to small vertical grid spacing, which in turn may lead to vertical velocity stability violations. To avoid stability violations the sigma system should always be used with time-substepping. Note also that the minimum depth at the coast may need to be increased to maintain stability when using sigma coordinates.

7.2 Numerical Sequence

Currently the sequence of calculation in SHOC is to solve the 3D equation, followed by the 2D equations and then the tracers (Figure 3.1). This is an artifact of basing the SHOC numerics on its predecessor MECO which used a first order Euler forward time-stepping scheme. The presence of the total depth term (D_s=D=η+H) in the sigma equations places some restrictions on this approach. The sigma approach in SHOC retains the MECO solutions to these restrictions, which are described below.
The Euler forward scheme requires that the depth averaged velocity is solved explicitly and elevation is solved using the velocities at the forward time-step (or conversely elevation is solved explicitly and velocities are solved using the elevation at the forward time-step) to maintain stability. This is not a problem, as velocity can be solved explicitly and elevation can be solved using the updated velocities. Also for stability reasons, this scheme requires that the u_2 Coriolis term be solved using the updated u_1 velocity. Again this is not a problem if the u_1 velocity is solved first. If a leapfrog scheme is used, then these stability restrictions are not valid.
In the sigma system the total depth at the forward time step is required to update velocity and updated velocity is required in the Euler forward scheme to keep the Coriolis term stable; this means elevation must be solved first so that total depth is obtained. Solving elevation first means that, in Euler's scheme, the velocity pressure terms (i.e. surface elevation slope and vertically integrated pressure slope) must use elevation at the forward time step to maintain stability. Hence the 3D mode solution cannot precede the 2D mode solutions, as elevation at the forward time step is not known at this point. However, the vertical integral of the 3D pressure must be calculated for use in the 2D mode. In the sigma system this is no problem since there always exists the same number of layers and one simply integrates from the surface (σ=0) to the bottom (σ=-1). Hence in sequence in the sigma system should be:

1. Calculate the vertical integral from σ=0 to σ=-1 of the 3D pressure field.
Do the 2D loop:
  2. Calculate elevation at the forward time step, η^{n+1}, using $u1av^n$ and $u2av^n$
  3. Calculate velocity using η^{n+1} ,the 3D pressure integral and using $u1av^{n+1}$ in $u2av^n$
End 2D loop
4. Calculate the 3D velocities using η^{n+1}
5. Calculate the tracers.

It is not possible to apply this sequence to the 'z' coordinate model, since the vertical integral of the pressure calculation requires the k index of the sea surface to be known at the forward time step. This means the elevation at the end of the 2D loop, η^{n+1}, must be known for step 1 above in

order to define the k level of the surface at the forward time step. However, the pressure integral cannot be calculated after the 2D loop since the 2D loop requires this pressure gradient to calculate velocity and hence elevation at the end of the 2D loop, thus closure to the sequence is lacking. These problems would be avoided using a leapfrog scheme since the pressure gradient can be calculated using elevation at the current time step, η^n.
In order to combine the sigma and 'z' models into one model and avoid this dilemma the velocity transports are solved (e.g. depth multiplied by velocity) and only converted to velocity after the 2D loop. This allows the 2D velocity to be calculated before elevation in the 2D loop, using elevations at the current time-step, η^n. The vertical integral of the pressure gradient used in the 2D velocity calculation can also use the k index corresponding to the surface at the current time-step. After the 2D loop the elevation at the forward time-step (and hence total depth at the forward time-step, D_s^{n+1}) is known and the velocity transports can be divided by D_s^{n+1} to provide the velocities, which are subsequently used to solve the tracer equation. The sequence used in both sigma and 'z' systems is therefore:

1. Calculate the 3D velocity transports using η^n and save the vertical integral (surface to bottom) of the 3D pressure field for the 2D mode.

Do the 2D loop

2. Calculate velocity transport using η^n ,the 3D pressure integral and using D_s^{n+1} u1av^{n+1} in u2av^n
3. Calculate elevation at the forward time step, η^{n+1}, using D_s^{n+1} u1av^{n+1} and D_s^{n+1} u2av^{n+1} calculated in step 2.

End 2D loop

4. 5. Get 2D and 3D velocities by dividing velocity transports by D_s^{n+1}

6. Calculate the tracers.

If the 'z' model is used, then the total depth, D_s, is set to 1.0.
The only inconsistency this approach presents is that the implicit calculation of the vertical diffusion term in the 3D velocity calculation requires the total depth at the forward time-step (which is not available in step 1 above). In the above sequence the total depth at the current time-step is used in its place (see below). It is assumed that the error introduced in this manner is small in comparison to the uncertainty involved in approximating the vertical viscosity coefficient.

7.3 Treatment of the vertical diffusion terms

The advection-diffusion equation is discretized as:

$$\frac{\partial D\phi}{\partial t} + advect(\phi) = hordiff(\phi) + \frac{\partial}{\partial \sigma}\frac{V_z}{D}\frac{\partial \phi}{\partial \sigma} \qquad 7.3.1$$

where advect(ϕ) represents the advecive terms and hordiff() represents the horizontal diffusion terms. Using the method of fractional steps this is solved in two parts, firstly the advective and horizontal diffusion by an explicit method:

$$\frac{\tilde{D}\tilde{\phi} - D^t\phi^t}{\Delta t} = hordiff(\phi) - advect(\phi) \qquad 7.3.2$$

where $\tilde{\phi}$ is a partial solution, then the vertical diffusion is solved implicitly:

$$\frac{D^{t+1}\phi^{t+1} - \tilde{D}\tilde{\phi}}{\Delta t} = \frac{\partial}{\partial\sigma}\frac{V_z}{D^{t+1}}\frac{\partial\phi^{t+1}}{\partial\sigma} \qquad 7.3.3$$

The time level of the total depth used in $\tilde{D}$ is not important provided this level is used in both 7.3.2 and 7.3.3. The time level at the current level, t, is used for momentum, and t+1 for tracers. The vertical diffusive equation for momentum in the σ system is:

$$\frac{\partial Du_1}{\partial t} = \frac{\partial}{\partial\sigma}\frac{V_z}{D}\frac{\partial u_1}{\partial\sigma} \qquad 7.3.4$$

Which is discretized as:

$$D^{t+1}u_1^{t+1} - D^t\tilde{u}_1 = \Delta t\frac{\partial}{\partial\sigma}\frac{V_z}{D^t}\frac{\partial u_1}{\partial\sigma} \qquad 7.3.5$$

where $\tilde{u}_1$ is the solution of the advective, horizontal diffusive, pressure and Coriolis terms and $\tilde{D} = D^t$. Dividing by D^t results in a form that is solved by the vertical diffusion equation:

$$\frac{D^{t+1}}{D^t}u_1^{t+1} = \tilde{u}_1 + \frac{\Delta t}{D^t}\frac{\partial}{\partial\sigma}\frac{V_z}{D^t}\frac{\partial u_1}{\partial\sigma} \qquad 7.3.6$$

The solution to the vertical diffusive equation for momentum is therefore equal to :

$$du_1 = \frac{\Delta t}{D^t}\frac{\partial}{\partial\sigma}\frac{V_z}{D^t}\frac{\partial u_1}{\partial\sigma} \qquad 7.3.7$$

The vertical diffusion algorithm is set up to solve an equation in the form of 7.3.7, but the quantity required as an end product of vertical diffusion is $D^{t+1}U^{t+1}$. Therefore $D^t.du_1$ must be added to $D^t\tilde{u}_1$ to provide $D^{t+1}u_1^{t+1}$, i.e. the updated velocity transport is given by:

$$D^{t+1}u_1^{t+1} = D^t\tilde{u}_1 + D^t du_1 \qquad 7.3.8$$

This quantity is subsequently used in the calculation of the u_2 Coriolis terms. Note that if the vertical diffusion is solved implicitly, then D^{t+1} should really be used on the rhs in equation 7.3.7
Since the total water depth at the forward time step, D^{t+1}, is available when vertical diffusion of tracers is performed, the tracer equation is discretized as:

$$D^{t+1}T^{t+1} - D^{t+1}\tilde{T} = \Delta t\frac{\partial}{\partial\sigma}\frac{K_z}{D^{t+1}}\frac{\partial T}{\partial\sigma} \qquad 7.3.9$$

where $\tilde{T}$ is the solution of the advective and horizontal diffusive terms and $\tilde{D} = D^{t+1}$. Dividing by D^{t+1} results in a form that is solved by the vertical diffusion equation:

$$T^{n+1} = \tilde{T} + \frac{\Delta t}{D^{t+1}}\frac{\partial}{\partial\sigma}\frac{K_z}{D^{t+1}}\frac{\partial T}{\partial\sigma} \qquad 7.3.10$$

The solution to the vertical diffusive equation for tracers is equal to;

$$dT = \frac{\Delta t}{D^{t+1}} \frac{\partial}{\partial \sigma} \frac{K_z}{D^{t+1}} \frac{\partial T}{\partial \sigma} \qquad 7.3.11$$

and the updated tracer concentration is then given by;

$$T^{t+1} = \tilde{T} + dT \qquad 7.3.12$$

7.4 Horizontal diffusion

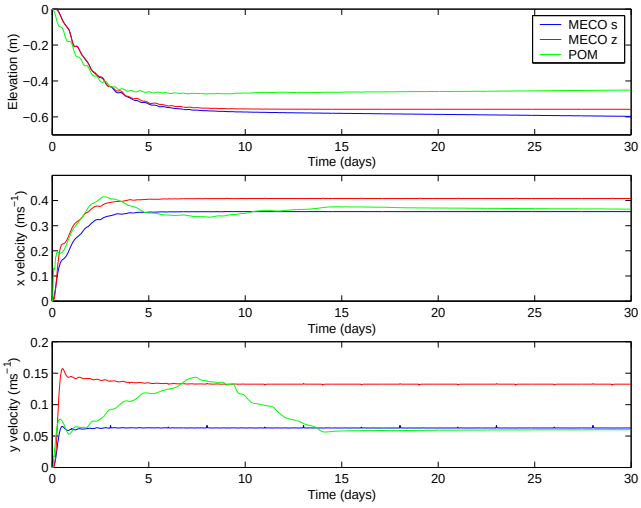
It is reccomended to use Smagorinsky diffusivity when using the sigma system so that mixing along sigma surfaces over steep bathymetry does not lead to cross-isobaric exchange in the absence of any motion.

7.5 Sigma Model performance

The sigma and 'z' versions of SHOC were compared with POM on a straight channel with open boundaries on the west and east ends and sloping bathymetry from 50m on the southern side to 100m on the northern side. The domain is forced by a westerly wind stress of 0.1 Nm^{-2}. This configuration is the same as that used by Palma and Matano (1998) in evaluating open boundary conditions using POM. The OBC's used here are cyclic on normal velocity and elevation and Orlanski radiation on tangential velocity. Temperature and salinity are held constant. Results at a location midway along the channel and 50km from the coast are summarised below.

Table 7.1 : Model Performance Comparison

Model	**Elevation** (m)	**2D x-velocity** (ms^{-1})	**2D y-velocity** (ms^{-1})	**3D x-velocity** (ms^{-1})	**3D y-velocity** (ms^{-1})
SHOC-z	-0.557	0.196	3.50×10^{-6}	0.408	0.132
SHOC-σ	-0.597	0.221	5.39×10^{-5}	0.356	0.063
POM	-0.450	0.205	3.58×10^{-4}	0.366	0.060

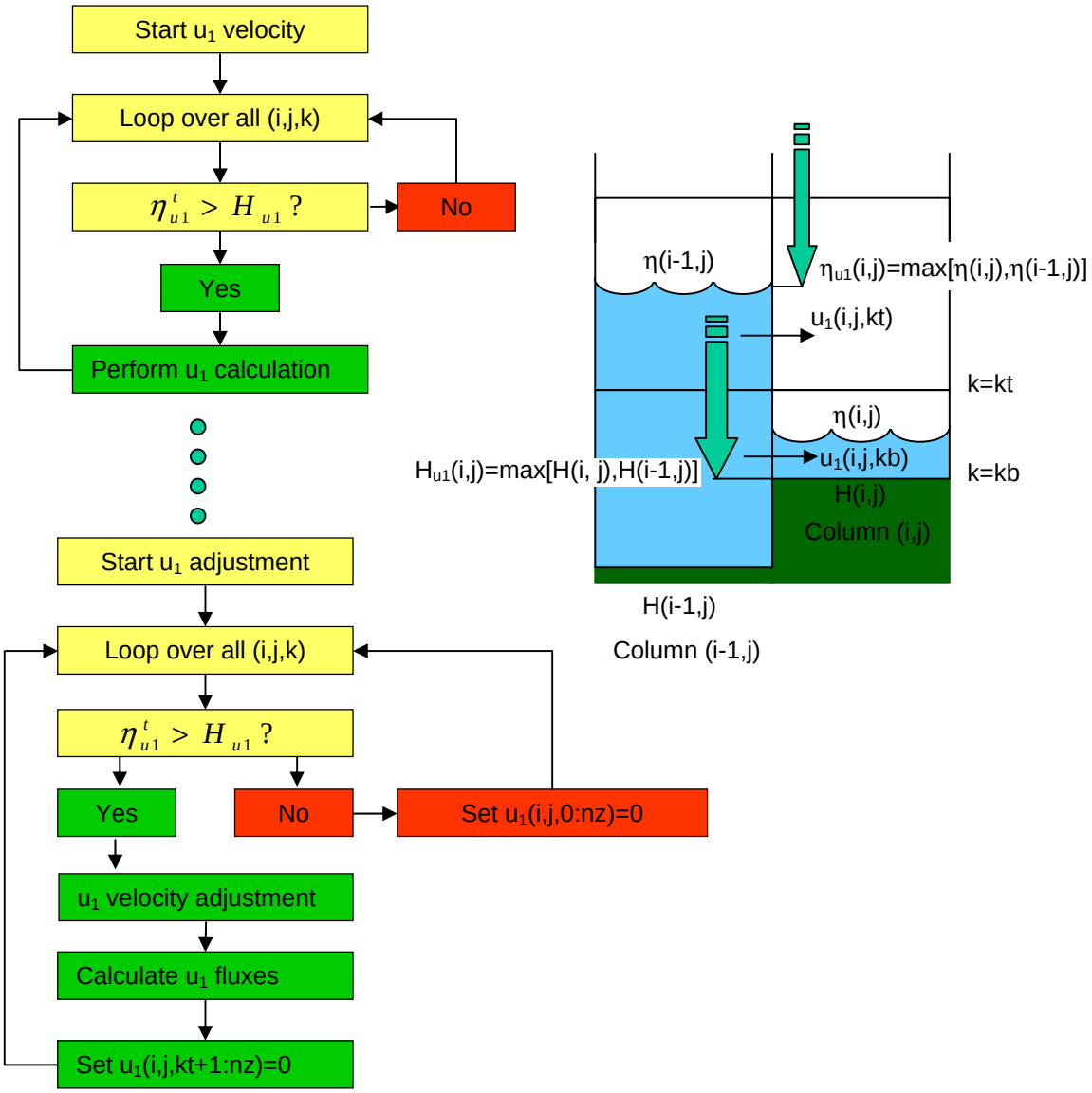


8. Wetting and Drying

SHOC possesses the capability of performing wetting and drying of cells. This involves the free surface moving through the constant 'z' layers allowing a given cell to be emptied of water and remain completely dry for one or many time integrations. A manifestation of this is the drying of a water column one cell deep such that the cell corresponds to land. Conversely, land cells may become wet if subjected to sufficiently high water level. This feature is useful for modelling regions subject to large tidal variations while maintaining adequate vertical resolution in the surface layer, or areas such as tidal flats which periodically wet and dry. An extreme example of the application of wetting and drying would be a bore propagating down a dry channel.

SHOC facilitates wetting and drying by checking whether the surface elevation is greater than the bottom depth, $\eta > H$. If the condition $\eta \leq H$ arises in a certain cell then flow is disallowed in that cell for the 2D mode. The flow of control regarding wetting and drying for u_1 velocity is illustrated in Figure 8.1

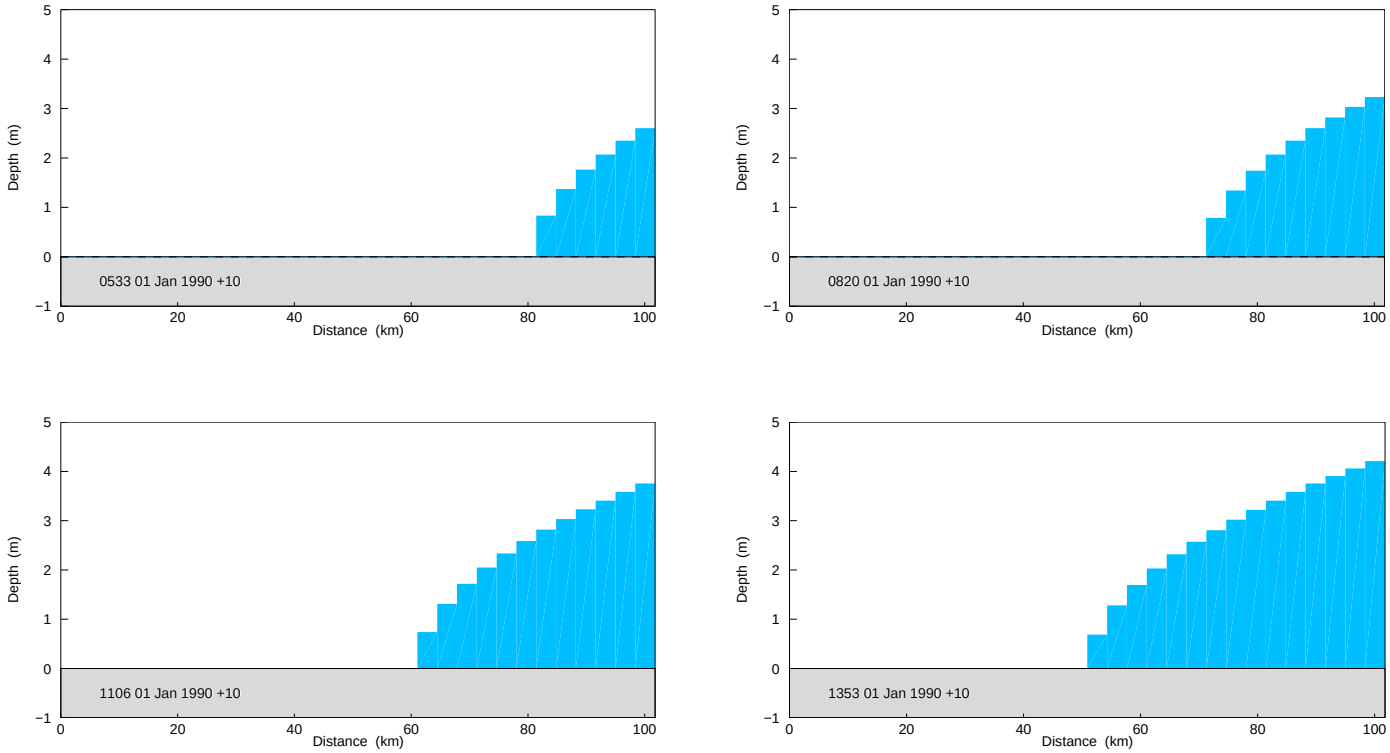
Figure 8.1 : Wetting and Drying for u_1 Velocity



SHOC allows for variable bottom topography, so the cell thickness at coordinates corresponding to the bottom, kb, may vary. The bottom depth at a velocity face is then the maximum of the depths of the cells either side of the face. Note that H(i,j) is negative so this maximum equates to the shallower of the bottom depths. The surface elevation at the face is the maximum of cell centered surface elevation either side of the face. Note again that if elevation drops below mean sea level then this will be negative. After the 3D velocities are calculated then the 2D mode is performed on all wet cells; the values of η, U_1 and U_2 in dry water columns remain unchanged from the previous time-step. When velocity adjustment occurs (ensuring the vertical integral of 3D velocities are equal to 2D velocities) dry cell faces are again identified and the velocities in the entire water column are set to zero on these faces. Velocities are also set to zero for wet cell faces above the surface cell. Finally, if the elevation has risen at the end of the 2D mode such that $\eta_{u1}^{t+1} > z_{kt+1} > \eta_{u1}^{t}$ (i.e. previously dry cells become wet) then the velocity of the newly wetted cells are set to that of the surface velocity before wetting. These procedures merely constitute housekeeping to ensure dry cells are associated with zero velocity; wetting and drying is essentially only possible through the formulation of the pressure gradient term which allows the surface to occupy different layers in adjacent cells (section 2.3).

An example of wetting and drying is illustrated in Fig. 8.2 where an initially dry channel is subjected to a non-zero velocity (1 ms^{-1}) on the right boundary resulting in the propagation of a bore along the channel. The depth (D) and length (L) of the bore is in agreement with theory ($L = gD^2 / 2C_D u^2$).

Figure 8.2 : Propagation of a bore along a dry channel.



9. Thermodynamics

9.1 Heat Flux Components

The heat flux formulation available in SHOC uses an advanced bulk parameterisation for sensible and latent heat fluxes, enhanced longwave formulation and the short wave component is calculated or supplied via time-series file. The heat flux formulation is easily accommodated using standard meteorological measurements as input. The net heat flux is computed as the sum of the components then applied as the surface boundary condition in the vertical diffusion equation (eqn. 2.5.2). Heat is input and mixed through the water column simultaneously using this approach due to the implicit nature of the vertical diffusion equation, hence avoiding undesirable skin effects when layers are thin. The components of the heat flux are described in turn.

9.1.1 Shortwave Radiation

The clear sky instantaneous short wave radiation is computed via (Zillman, 1972) :

$$S_g = S_E \sin^2(h)((\sin(h)+2.7).10^{-3}e_a + 1.085\sin(h)+0.1)^{-1} \quad 9.1.1$$

where S_E is the solar beam irradiance in free space (Wm^{-2}), with a mean of 1380 Wm^{-2} (the solar constant). The magnitude of S_E varies in a sinusoidal manner during the year as the distance between sun and earth changes. These variations constitute no more than 3.5% of the mean value. The variable h is the solar elevation (Deg) which is calculated via:

$$\sin(h) = \sin\phi\sin\delta + \cos\phi\cos\delta\cos t \quad 9.1.2$$

where ϕ is the latitude, δ is the solar declination and t is the hour angle of the sun. The solar declination is given by (Oberhuber, 1988):

$$\begin{aligned}\delta = {} & 0.006918 + 0.070257\sin\beta - 0.399912\cos\beta \\ & + 0.000907\sin(2\beta) - 0.006758\cos(2\beta) \\ & + 0.00148\sin(3\beta) - 0.002697\cos(3\beta)\end{aligned} \quad 9.1.3$$

where β is the Julian day. The clear sky irradiance is corrected for cloud cover (Reed, 1977):

$$S_{gc} = S_g(1 - 0.62C + 0.0019h_n) \quad 9.1.4$$

where C is the cloud cover in tenths and h_n is the noon solar elevation. Finally the cloudy sky irradiance is corrrected for the amount of radiation reflected at the earth surface using:

$$Q_I = S_{gc}(1-\alpha) \quad 9.1.5$$

where α is the albedo at the surface as a function of solar elevation and cloud (e.g. Zillman, 1972, Figure 3.2).
Alternatively, short wave radiation may be input to the sea surface is via file input (Wm^{-2}), subject to the albedo correction 9.1.5. Short wave radiation may be partially or completely excluded from the surface net heat flux boundary condition, and distributed with depth through the water column. The degree to which short wave radiation is excluded from the net heat flux is controlled by the transmission coefficient and is due to the preferential absorption of longer wavelengths of short

wave radiation within the first few meters (e.g. Simpson and Dickey, 1981). Transmission varies from approximately 0.42 for Jerlov class I water to 0.22 for class III water. The short wave radiation at any layer, k, is given by:

$$Q_k = R_t Q_I \exp(-\eta z) \quad 9.1.6$$

where R_t is the transmission coefficient and η is the extinction coefficient (~0.2 m^{-1}; this is the reciprocal of the *e*-folding length, ζ (m), used by Simpson and Dickey (1981), Table 1). Change in temperature within the water column is proportional to the divergence of short wave radiation:

$$\Delta T_k = \Delta t(Q_{k+1} - Q_k)/(c_v \rho \Delta z) \quad 9.1.7$$

where Δz (m) is the thickness of the surface layer, ρ (kgm^{-3}) is the density in the layer and c_v (~3990 $Jkg^{-1}K^{-1}$) is the specific heat of water. This temperature change is the result of a heat input and must be added to the water temperature in the layer k.
Alternatively the dual extinction parameterization of Paulson and Simpson (1977) may be used where the extinction coefficient for the surface 10m or so, η_1, corresponds to attenuation of the longer wavelengths and the extinction coefficient η_2 corresponds to attenuation of the shorter wavelengths at depth. The fraction R_f determines the partitioning between surface and deep attenuation.

$$Q_k = Q_I \left[R_f \exp(-\eta_1 z) + (1 - R_f)\exp(-\eta_2 z)\right] \quad 9.1.8$$

9.1.2 Longwave Radiation

The net long wave radiation at the water surface is given by the sum of the downward longwave input from the atmosphere, the amount of the downward longwave that is reflected upwards and the upward longwave radiation emitted from the surface. For clear skies this is given by (Zillman 1972, eqn. 3.1):

$$R_c = L_d - (1 - \varepsilon_s)L_d - \varepsilon_s \sigma T_s^4 \quad 9.1.8$$

where where σ = 5.67x10^{-8} $Wm^{-2}K^{-4}$ is Stefan's constant, T_s (^oK) is the temperature of the surface, ε_s (~0.985) is the emissivity of the water surface, and L_d is given by;

$$L_d = \varepsilon_a \sigma T_a^4 \quad 9.1.9$$

where with T_a is the air temperature (^oC) and ε_a is the emissivity of the atmosphere given by (Swinbank, 1963, eqn. 3.8):

$$\varepsilon_a = 0.92 \times 10^{-5} T_a^2 \quad 9.1.10$$

Note that T_s in eqn. 9.1.6 is actually the true radiative temperature of the sea surface, which may differ from the measured SST, but for practical purposes are assumed to be identical. Also, ε_s is equivalent to 1-(longwave albedo) : grey body approximation. Equation 9.1.6 may be then approximated with negligible error (Zillman, 1972, eqn. 3.4 (a)) by:

$$R_c = \sigma \varepsilon_s T_a^4 (\varepsilon_a - 1) - 4\varepsilon_s \sigma T_a^3 (T_s - T_a) \quad 9.1.11$$

The clear sky longwave radiation is corrected for cloud cover via (Budyko, 1963, eqn. 3.9):

$$Q_L = R_c(1-\beta C) \quad 9.1.12$$

where β=0.63 at latitude 30°S and C is the fractional cloud cover.

9.1.3 Sensible and Latent Heat Fluxes

The sensible heat flux is derived via the bulk formulation (Gill, 1982, p30):

$$Q_H = \rho_a c_p c_H W(T_s - T_a) = \rho_a c_p \overline{w'T_a'} \quad 9.1.13$$

where ρ_a is the density of air (~1.22 kgm^{-3}), c_p is the specific heat of air at constant pressure (~1005 $Jkg^{-1}K^{-1}$ in the absence of humidity effects), c_H is the bulk transfer coefficient (Stanton number), W is the windspeed (ms^{-1}) at 10m height and T_a is the air temperature at 10m height. The turbulent fluctuations of wind and air temperature are w' and T_a'. The air temperature should be supplied via file, if no data is present a default value of 15°C is used. Q_H is also an upward flux and must be subtracted from the surface layer temperature. An equivalent formulation is used for the latent heat flux:

$$Q_E = \rho_a L_v c_E W(q_s - q_a) \quad 9.1.14$$

where c_E is the bulk transfer coefficient for latent heat (Dalton Number), q_s ($kgkg^{-1}$) is the specific humidity at the sea surface, q_a is the specific humidity at the reference height (10m) and L_v (Jkg^{-1}) is the latent heat of vaporization given by (Gill, 1982, p607):

$$L_v = 2.5008\times10^6 - 2.3\times10^3 T \quad 9.1.15$$

where T is the surface water temperature (°C) or by;

$$L_v = 4.1868(597.31 - 0.56525T_a)\times10^3 \quad 9.1.16$$

The density of moist air may be more accurately represented by:

$$\rho_a = \frac{3.4838\times10^{-3}P_a}{273.16+T_v} \quad 9.1.17$$

where P_a is the atmospheric pressure (Pa) and T_v (°C) is the virtual temperature:

$$T_v = \left[(T_a + 273.16)(1+0.6089q_a)\right] - 273.16 \quad 9.1.18$$

Furthermore, the specific heat at constant pressure is given for moist air by:

$$c_p = 1.004(1+0.9q_a)\times10^3 \quad 9.1.19$$

The specific humidity is the mass of water vapour in a unit mass of moist air and is given by:

$$q = \frac{0.622 e_a}{(P_a - 0.378 e_a)} \quad 9.1.20$$

The vapour pressure, e_a (Hpa), is estimated by:

$$e_a = e_s - (6.6\times10^{-4}(1+1.5\times10^{-3}T_w)(T_a - T_w)P_a \quad 9.1.21$$

with P_a in (HPa) and T_w (^{o}C) is the wet bulb temperature at the standard height. The saturation vapour pressure, e_s (HPa), is given by:

$$\begin{aligned} e_s = T_w(T_w(T_w(T_w(T_w 6.136821\times10^{-11}) \\ + 3.03124\times10^{-6}) + 2.650648\times10^{-4}) \\ + 1.428946\times10^{-2}) + 0.4436519) + 6.1078 \end{aligned} \quad 9.1.22$$

Saturation vapour pressure is corrected for salinity with:

$$e_s = e_s(1 - 0.000537S) \quad 9.1.23$$

where S is the salinity. Alternatively the vapour pressure may be calculated from the dew point temperature:

$$e_a = P_a a_1^{b_3} \times 10^{a_2 b_4 + a_4 b_5 + a_5 b_6} \quad 9.1.24$$

where $a_1 = 373.16/(T_{dew} + 273.16)$ where T_{dew} is the dew point temperature (^{o}C), b_1=-3.49149, b_2=11.344, b_3=5.02808, b_4=-7.90298, b_5=8.1328x10^{-3}, b_6=-1.3816x10$_{-7}$, a_2=a_1-1, a_3=1-1/a_1, $a_4 = (10^{a_2 b_1}) - 1$ and $a_4 = (10^{a_3 b_2}) - 1$ and P_a in (HPa).
The specific humidity over water is calculated using 9.1.20 with e_a=e_s from 9.1.22 where it is assumed the wet bulb temperature = SST and the air is 100% saturated, thus e_a in this case represents the maximum amount of water the air can accommodate.

The bulk transfer coefficients, c_E and c_H, have been estimated in numerous studies. Blanc (1985) reviews ten studies providing estimates of these coefficients and concludes that each scheme provided different results when applied to the same data, highlighting the uncertainty inherent in the bulk method. The discrepancies were attributed to the use of indirect flux measurements in the studies, increased error in flux measurement at low wind-speed and lack of data at high windspeeds. The bulk scheme employed in SHOC is that of Kondo (1975), chosen for the wide range of air-sea temperature differences (20^{o}C) and wind-speeds (0.3 to 50 ms^{-1}) the scheme is applicable to. Bulk parameters are also corrected for stability (i.e. situations where the sea is warmer than the air contributing enhanced air-sea exchange or where the sea is cooler than the air suppressing exchange). The bulk transfer coefficients derived by Kondo (1975) are given by:

$$\begin{aligned} c_{D10} &= (a_d + b_d W_{10}^{p_d})\times10^{-3} \\ c_{H10} &= (a_h + b_h W_{10}^{p_h} + c_h(W-8)^2)\times10^{-3} \\ c_{E10} &= (a_e + b_e W_{10}^{p_e} + c_e(W-8)^2)\times10^{-3} \end{aligned} \quad 9.1.25$$

where c_D is the drag coefficient, W (ms^{-1}) is the windspeed at 10m height and the coefficients are all applicable to a standard height of 10m. The constants a,b,c and p are given by:

Table 9.1 (a) : Bulk Transfer Coefficient Parameters

W_{10}	a_d	a_h	a_e	b_d	b_h	b_e
0.3 – 2.2	0	0	0	1.08	1.185	1.23
2.2 – 5	0.771	0.927	0.969	0.0858	0.0546	0.0521
5 –8	0.867	1.15	1.18	0.0667	0.01	0.01
8 – 25	1.2	1.17	1.196	0.025	0.0075	0.008
25 – 50	0	1.652	0.073	0.073	-0.017	-0.016

Table 9.1 (b) : Bulk Transfer Coefficient Parameters

W_{10}	c_h	c_e	p_d	p_h	p_e
0.3 – 2.2	0	0	-0.15	-0.157	-0.16
2.2 – 5	0	0	1	1	1
5 –8	0	0	1	1	1
8 – 25	-0.00045	-0.0004	1	1	1
25 – 50	0	0	1	1	1

The bulk coefficients at 10m height can be scaled to any reference height z_r by:

$$\begin{aligned} c_{Dz} &= k^2\left[kc_{D10}^{-0.5} - \ln(z_{10}/z_r)\right]^{-2} \\ c_{Hz} &= kc_{Dz}^{0.5}\left[kc_{D10}^{0.5}c_{H10}^{-1} + \ln(z_r/z_{10})\right]^{-1} \\ c_{Ez} &= kc_{Dz}^{0.5}\left[kc_{D10}^{0.5}c_{E10}^{-1} + \ln(z_r/z_{10})\right]^{-1} \end{aligned} \qquad 9.1.26$$

where k=0.4 is the von Karman constant and z_{10}=10m. Wind is scaled to the reference height by:

$$W_z = W_{10}\ln(\frac{z_r}{z_0})\ln(\frac{z_{10}}{z_0}) \qquad 9.1.27$$

where

$$z_0 = \exp(\ln(z_{10}) - kc_{D10}^{-0.5}) \qquad 9.1.28$$

The stability parameter at height z_r is defined as:

$$s = s_0\left[\frac{|s_0|}{|s_0|+0.01}\right] \quad where \quad s_0 = (T_s - T_a)W_z^{-2}(1+\log_{10}(\frac{z_{10}}{z_r}))^{-2} \qquad 9.1.29$$

Kondo (1975) suggests that the influence of water vapour upon stability should be taken into account, and (T_s-T_a) in eqn. 9.1.24 should be replaced with :

$$T_s - T_a \rightarrow (T_s - T_a) + 0.61\theta(q_s - q) \qquad 9.1.30$$

where $\theta = T+\Gamma z$ (^oC) is the potential temperature with Γ=0.0098 $^oCm^{-1}$ the adiabatic lapse rate.

Under for stable conditions ($T_a > T_s$) :

$$\begin{aligned} &c \approx c_z(0.1 + 0.03s + 0.9\exp(4.8s)) \quad for \quad -3.3 < s < 0 \\ &c \approx 0 \quad for\ S \le -3.3 \end{aligned} \qquad 9.1.31$$

where c can be c_D, c_H or c_E. For unstable conditions ($T_a < T_s$) :

$$c \approx c_z(1 + 0.63s^{0.5}) \quad 9.1.32$$

The bulk transfer coefficients vary from zero for stable, low wind-speed conditions to > $2x10^{-3}$ for unstable, low wind-speed conditions. For high wind-speed conditions the bulk parameters converge towards ~$1.25x10^{-3}$ irrespective of stability conditions.
The net heat flux, Q_N (Wm^{-2}) is then:

$$Q_N = Q_I + Q_L + Q_H + Q_E \quad 9.1.33$$

and the surface heat flux applied as the surface boundary condition, H_T ($ms^{-1}K$) in Eqn. 2.5.2 is:

$$H_T = \frac{Q_N}{c_v \rho} \quad 9.1.34$$

where c_v=$4x10^3$ ($Jkg^{-1}K^{-1}$) is the specific heat of water at constant volume and ρ (kgm^{-3}) is the water density.
The bulk schemes of Large and Pond (1982), Kitaigorodskii et al (1973) and Masagutov (1981) are also coded into SHOC; the reader is referred to these references for details or the study of Blanc (1985) in which a comparison is made.

9.1.4 Advection Correction

Air temperature and humidity measurements are typically taken at terrestrial sites, which may differ significantly from measurements taken over water. Under offshore wind conditions, air temperature and humidity transfer across the air-sea interface via turbulent exchange may cause these variables to converge towards the sea surface values with increasing distance offshore. This results in smaller air-sea gradients, which consequently reduces the sensible and latent heat fluxes. Assuming that the turbulent exchange of heat and water vapour is restricted to the vertical direction, then the steady state advection diffusion equation (for air temperature, T_a, in this case) is written as:

$$W \frac{\partial T_a}{\partial n} = K_z \frac{\partial^2 T_a}{\partial n^2} \quad 9.1.35$$

where n is the direction normal to the coast, W is the windspeed and K_z is the vertical eddy diffusivity in the boundary layer above the sea surface (assumed to be independent of height). Boundary conditions are:

$$\begin{aligned} T_a &= T_s \quad on\ z = 0 \ \ when \ \ n > 0 \\ T_a &= T_{ai} \quad at\ n = 0, z > 0 \end{aligned} \quad 9.1.36$$

where T_{ai} is the air temperature at the coast. The solution to 9.1.35 is given as (Sutton, 1953):

$$T_a = T_s + (T_{ai} - T_s) erf\left[\frac{z\sqrt{W}}{\sqrt{4K_z n}}\right] \quad 9.1.37$$

where erf() is the error function:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \qquad 9.1.38$$

which is approximated by (Hastings, 1955):

$$erf(x) = 1 - (a_1\gamma + a_2\gamma^2 + a_3\gamma^3 + a_4\gamma^4 + a_5\gamma^5)e^{-x^2} \qquad 9.1.39$$

with $\gamma = 1/(1+px)$, p=0.3275911, a_1=0.254829592, a_2=-0.284496736, a_3=1.421413741, a_4=-1.453152027 and a_5=1.061405429.
The turbulent flux of air temperature (or humidity) is related to the mean vertical gradient, i.e.

$$\overline{w'T_a'} = -K_z \frac{\partial \overline{T_a}}{\partial z} \qquad 9.1.40$$

From Eqn. 9.1.13 and Eqn. 9.1.40 it follows that:

$$\overline{w'T_a'} = -c_H W(T_a - T_s) = -K_z \frac{\partial \overline{T_a}}{\partial z} \approx -K_z \frac{T_a - T_s}{\Delta z} \qquad 9.1.41$$

Therefore $K_z \sim c_H W \Delta z$ ($m^2 s^{-1}$) for temperature and $K_z \sim c_E W \Delta z$ ($m^2 s^{-1}$) for water vapour where Δz is the height between the sea surface and the reference level. Using the vertical diffusivity from Eqn. 9.1.41 the temperature/humidity at any distance n offshore is calculated via Eqn. 9.1.37 to provide air temperatures and humidities over water to be input into the bulk formulae. In practice this procedure is performed if the wind is offshore and terrestrial measurements are expected to differ significantly from measurements over water.

9.1.5 Precipitation

The change in the surface temperature due to rain entering the surface layer (at a temperature equivalent to the air temperature) is given by:

$$\Delta T_s = T_a \Delta t P / \Delta z \qquad 9.1.42$$

where P is the precipitation rate (ms^{-1}).

9.2 Surface forcing

A simple method of implementing a heatflux is by relaxation of the surface layer temperature to some prescribed, possibly time varying field, e.g. Haney (1971). The relaxation used in this method is of the form:

$$T_0^{t+1} = T_D + (T_0^t - T_D)e^{-T_C dt} \qquad 9.2.1$$

Where T_0 is the surface temperature, T_D is the prescribed temperature and T_C is the relaxation time constant. Note that this implies that long time constants result in strong relaxation to the

prescribed data, and short time constants result in weak relaxation, hence small deviations from T_0.

9.3 Prescribed heat flux

The net heat flux applied as the surface boundary condition may be applied directly in eqn. 2.5.2 via time-series file. The net heat fluxes are supplied in Wm^{-2} in the time-series file and must undergo the scaling outlined in eqn. 9.1.28 before being applied as the surface boundary condition. An input of heat (leading to warming of the sea surface) is given a positive sign by convention.

9.4 Inverse estimation

Neglecting advective and horizontal diffusive effects, the conservation equation for heat may be vertically integrated from some depth –h to the undisturbed surface:

$$\int_{-h}^{0} \frac{\partial T}{\partial t} dz = -\int_{-h}^{0} \overline{w'T'} dz \qquad 9.4.1$$

where $-\overline{w'T'}$ is the vertical flux of heat. Assuming that the depth –h is invariant with time and there exists no flux of heat through this depth (i.e. –h is the mixed layer depth), then eqn. 9.4.1 may be simplified to:

$$\frac{\partial}{\partial t} \int_{-h}^{0} T dz = H_T = \frac{Q_N}{c_v \rho} \qquad 9.4.2$$

Therefore, assuming that T is well mixed throughout the depth h (i.e. independent of depth over the range 0 to –h), the temperature change, ΔT, over a time period Δt throughout depth h due to the heat flux Q_N is given by eqn. 9.1.1, where $h=\Delta z$. Alternatively, if a temperature change is known then the amount of heat required to achieve this change over a time period Δt may be calculated:

$$Q_N = \frac{\Delta T c_v \rho h}{\Delta t} \qquad 9.4.3$$

Using $\Delta T=T_{mod}-T_{obs}$ where T_{mod} is the modelled SST and T_{obs} is an observed SST, Eqn. 9.4.3 forms the basis of the inverse heat flux calculation. This may be used a first estimate of the surface heat flux given a time-series of SST observations. Obviously the time-scale Δt has great impact on the calculated flux, since a larger flux is required to converge measured and modelled SST in a shorter time. Generally Δt is of the order of the sampling frequency of the observations.

9.5 Salt Flux

A salt (or freshwater flux), H_S (ms^{-1}psu) may be applied as a surface boundary condition to the salinity vertical diffusion equation 2.5.2. The salt flux is given simply by:

$$H_S = S_o(E - P) \qquad 9.5.1$$

where S_o is the surface salinity expressed as a fraction (e.g. 35 psu = 0.035), E is the evaporation (ms^{-1}) and P is the precipitation (ms^{-1}) (e.g. Mellor 1996, Eqn 3.3). E and P are input into SHOC as timeseries files in units of mm day^{-1}.
Note that the latent heat of evaporation, Eqn 9.1.14, may be used to estimate the evaporation rate by dividing by the latent heat of vaporization;

$$E = \frac{Q_E}{L_v \rho_w} \quad (ms^{-1}) \qquad 9.5.2$$

where Q_E (Wm^{-2}) is the latent heat of evaporation and ρ_w ~ 999 kgm^{-3} is the density of fresh water.

10. Waves.

The influence of waves is handled via the wave module, and may be coupled to the hydrodynamics. This coupling includes the forcing of 2D velocity by tangential radiation stresses and enhanced drag due to wave enhanced bottom friction.

10.1 Tangential Radiation Stresses

The impact of waves on currents has been included in SHOC via tangential radiation stresses. The formulation follows that of Bye (1977a) and is outlined below. Radiation stresses are included in the 2D mode through the terms:

$$-\frac{\partial S_{xx}}{\partial x}-\frac{\partial S_{xy}}{\partial x}=-\frac{\partial}{\partial x}\left[\int_{-H}^{\eta}(\overline{u'^2}-\overline{w'^2})dz+\tfrac{1}{2}g\overline{\eta'^2}\right]-\frac{\partial}{\partial y}\left[\int_{-H}^{\eta}\overline{u'v'}dz\right] \qquad 10.1$$

in the e_1 direction (for Eqn. 2.6.3), and;

$$-\frac{\partial S_{yx}}{\partial x}-\frac{\partial S_{yy}}{\partial x}=-\frac{\partial}{\partial x}\left[\int_{-H}^{\eta}\overline{u'v'}dz\right]-\frac{\partial}{\partial y}\left[\int_{-H}^{\eta}(\overline{v'^2}-\overline{w'^2})dz+\tfrac{1}{2}g\overline{\eta'^2}\right] \qquad 10.2$$

in the e_2 direction (for Eqn. 2.6.4), where (u',v',w') are the velocity fluctuations due to the orbital motion of the wave and η' is the elevation fluctuation due to the wave. The normal components of radiation stresses, S_{xx} and S_{yy}, generate a setup or setdown, and the tangential components, S_{xy} or S_{yx}, generate a long-shore circulation. The flow resulting from radiation stresses is simplified to include the tangential components only adjacent to a solid coast. A further approximation is made to relate wave refraction to the angle of approach of the wave train to the coastal boundary (see Bye, 1977a). The tangential radiation stresses are given by (Longuet-Higgins, 1970):

$$S_{xy}=S_{yx}=\tfrac{1}{4}ga^2\sin\phi\cos\phi \qquad 10.3$$

where *a* is the wave amplitude and ϕ is the angle of propagation relative to the x-axis. As the wave refracts on its approach to the coast it is assumed that the angle of incidence at breaking is proportional to ϕ, so that:

$$S_{xy}=S_{yx}=\delta\frac{1}{4}\frac{ga^2\alpha\theta}{\pi/2} \qquad 10.4$$

where α ~ 0.1 is a refraction coefficient, $0\le\theta\le\pi/2$ is the angle of the wave train relative to the coast and δ = 1 for eastern/northern coasts and δ = -1 for western/southern coasts.

10.2 Wave Amplitude Approximations

The wave module is capable of approximating wave amplitude and period if these quantities are not explicitly supplied via file input. If a wave period, T (s), only is supplied, the wave amplitude is approximated first using an estimation for the wavenumber:

$$er_n=\ fabs(a\tanh(wn_nD)(\tfrac{2\pi}{T})^2/(wn_ng)\qquad n=1,2,3 \qquad 10.5$$

where:

$$wn_1 = \frac{(\frac{2\pi}{T})}{\sqrt{gD}}, \quad wn_2 = \frac{(\frac{2\pi}{T})^2}{g}, \quad wn_3 = wn_1 + wn_2 \tag{10.6}$$

Then the wavenumber, k (m^{-1}), is:

$$\begin{aligned} &k = wn_2, \quad er_1 = er_2 \qquad if\ er_2 < er_1 \\ &k = wn_1, \quad er_2 = er_1 \qquad if\ er_2 \geq er_1 \\ &k = wn_3 \quad if\ er_3 < er_1 \end{aligned} \tag{10.7}$$

The wave amplitude, a (m), is then given by (Tang and Grimshaw, 1996):

$$a = k^{-1}\left[\frac{0.2.0.001}{W_{diss}}\left|28\frac{(\tau_{sx}^2+\tau_{sy}^2)^{1/4}}{2\pi/kT}-1\right|\right]^{\frac{1}{4}} \qquad for \quad 28(\tau_{sx}^2+\tau_{sy}^2)^{1/4} > 2\pi/kT \tag{10.8}$$

where W_{diss}=100 is the wave dissipation constant.
Alternatively, if both wave period and amplitude are not supplied via file input, they may be estimated using (Toba, 1978):

$$a = 0.5\tilde{H}\frac{u_*^2}{g} \tag{10.9}$$

where $u_* = (\sqrt{\tau_{sx}^2+\tau_{sy}^2}\ /\ \rho_{air})^{1/2}$ is the air friction velocity, F (m) is the fetch and:

$$\tilde{H} = 0.05\left[\frac{gF}{u_*^2}\right]^{\frac{1}{2}} \tag{10.10}$$

The wave period is given by:

$$T = 6.4\tilde{H}^{2/3}\frac{u_*}{g} \tag{10.11}$$

Alternative approximations for wave period and amplitude using the wind speed at 10 height (w_{10}) are (e.g. Hipsey et al, 2006):

$$T = 1.2.2\pi\frac{w_{10}}{g}\tanh(\varphi)\tanh\left(\frac{0.0379(gF/w_{10}^2)^{1/3}}{\tanh(\varphi)}\right), \qquad \varphi = 0.833(gD/w_{10}^2)^{3/8} \tag{10.12}$$

$$a = \frac{0.283}{2}\frac{w_{10}^2}{g}\tanh(\phi)\tanh\left(\frac{0.00565(gD/w_{10}^2)^{1/2}}{\tanh(\phi)}\right), \qquad \phi = 0.53(gD/w_{10}^2)^{3/4} \tag{10.13}$$

Near bottom orbital velocities, u_b (ms^{-1}), are estimated using:

$$u_b = \frac{0.5a(\pi / T)}{\sinh(kD)} \qquad 10.14$$

11. Generic Storm Systems

Wind stress may be applied as a forcing parameter corresponding to the passage of cyclonic or anticyclonic pressure systems. These systems are defined by a location of the center of the system (x_o,y_o), radius R, pressure gradient $\partial P/\partial n$, eccentricity e (0<e<1 and $b^2 = a^2(1-e^2)$ where a and b are the major and minor axes of the ellipse respectively) and rotation θ to a circle of latitude. The wind speed at any point in the domain obeys the gradient wind equation:

$$c = -\frac{rf}{2} \pm \sqrt{(\frac{rf}{2})^2 + \frac{r}{\rho_a}\frac{\partial P}{\partial n}} \qquad 11.1$$

where f is the Coriolis parameter, r is the distance from the center and ρ_a is the air density. For an anticyclonic system the pressure gradient is negative. For the system to remain stable it follows that:

$$r \geq -\frac{4}{\rho_a f^2}\frac{\partial P}{\partial n} \qquad 11.2$$

If eqn. 11.2 does not hold a wind speed of c=rf/2 is assumed. A pressure gradient distribution is assumed where $\partial P/\partial n = 0$ at the center of the system with a parabolic distribution from 0<r< r_1 and r_2<r< R where 0< r_1< r_2<R. The parabolic distribution is given by:

$$\frac{\partial P}{\partial n} = \frac{4(\partial P/\partial n)_{\max}}{R^2} r(R-r) \qquad r_1 > r > r_2 \qquad 11.3$$

where $(\partial P/\partial n)_{\max}$ is a pre-defined maximum pressure gradient. The wind stress is then calculated via:

$$\tau_s = \rho_a C_{DS} c^2 \qquad 11.4$$

where the wind drag coefficient is given by eqn. 2.5.5.
The wind stress vector lies on an ellipse with center (x_o,y_o), having an equation:

$$(x-x_o)^2 + \frac{(y-y_o)^2}{(1-e^2)} = a^2 \qquad 11.4$$

If $a \leq R$ then any point (x,y) lies within the radius R and $\tau_s \neq 0$. Let $(x'+x_o, y'+y_o)$ be the coordinates of point (x,y) rotated about (x_o,y_o) by – θ (i.e. point (x,y) lies on the ellipse passing through (x',y') rotated by θ, then;

$$\begin{aligned} x' &= (x-x_o)\cos\theta + (y-y_o)\sin\theta \\ y' &= (y-y_o)\cos\theta - (x-x_o)\sin\theta \end{aligned} \qquad 11.5$$

and

$$d = \sqrt{(x')^2 + \frac{(y')^2}{(1-e^2)}} \qquad 11.6$$

is the length of the semi-major axis of the ellipse with center (x_o,y_o) rotated by θ passing through (x,y) (i.e. d is the semi-major axis of the ellipse passing through (x',y') subject to no rotation) and if d<R then the wind stress at (x,y) is non-zero. The slope, α, of the tangent to the ellipse at (x,y) is determined by calculating $\partial y / \partial x$ of the ellipse equation in rotated coordinates, i.e.

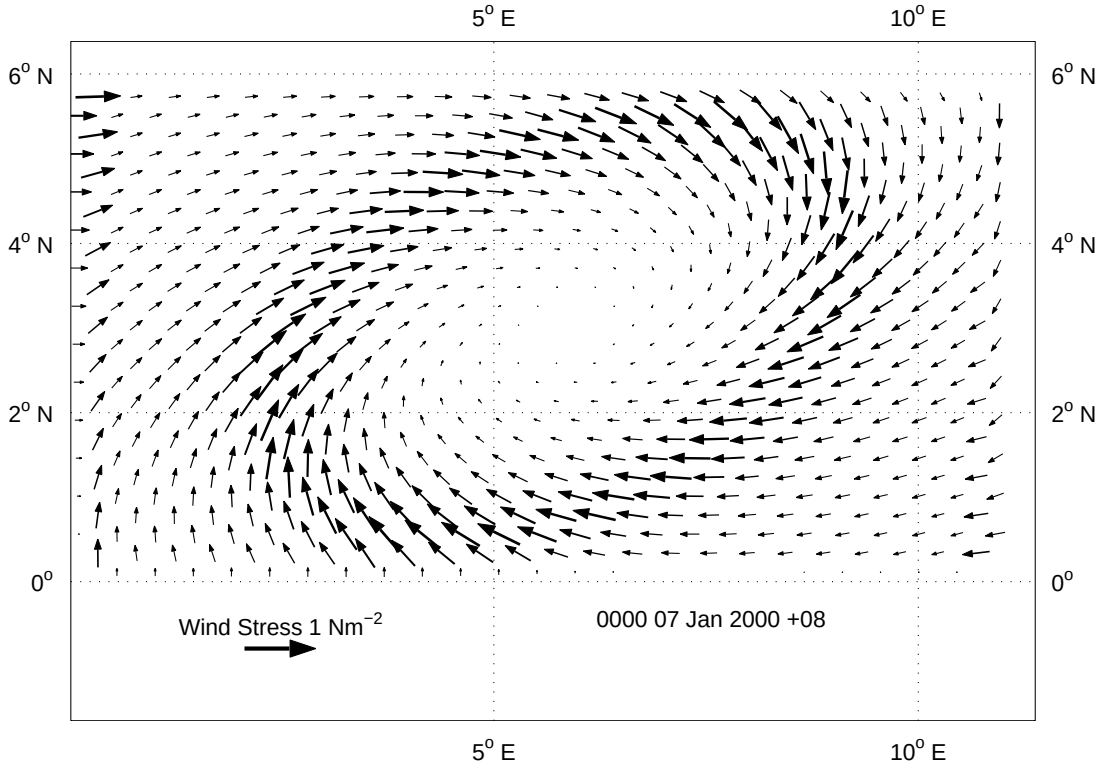
$$\alpha = \frac{-(1-e^2)\left[(x-x_o)\cos^2\theta + (y-y_o)\cos\theta\sin\theta\right] + (y-y_o)\cos\theta\sin\theta - (x-x_o)\sin^2\theta}{(1-e^2)\left[(x-x_o)\cos\theta\sin\theta + (y-y_o)\sin^2\theta\right] + (y-y_o)\cos^2\theta - (x-x_o)\cos\theta\sin\theta} \qquad 11.7$$

The wind stress vector may then be resolved into its x and y components:

$$\begin{aligned} \tau_{sx} &= \tau_s \cos\alpha \\ \tau_{sy} &= \tau_s \sin\alpha \end{aligned} \qquad 11.8$$

This type of stress formulation was used in Herzfeld and Tomczak (1997) to investigate the wind stress forcing in an idealized bay with sloping bottom. Typically synopyic weater systems are of size 3000 – 5000km with maximum pressure gradients ~7×10^{-4} HPa/km (2 HPa in 2.5° (280km)). An example of the resulting wind stress distribution is given in Figure 11.1.

Figure 11.1 : Example Storm System : R=2000km, $(\partial P/\partial n)_{\max}$ =$7e^{-4}$, θ=25°, e=0.8



12. Sparse System

The sparse coordinate system represents the physical grid configuration of the domain as a 1D vector in memory. Only cells that may potentially contain water are included in this vector; dry land locations are omitted, generally leading to large savings in memory. The first land location and bottom sediment layer are included in the sparse grid in order to establish boundary conditions. These cells are referred to as ghost or phantom points since they are technically not part of the wet domain. In some situations (e.g. headlands or outside corners) a ghost point may not be unique and several ghost points occupy the same physical point in space. These points are referred to as virtual or multiple ghost points, and are easily accounted for in the sparse matrix since there is no dependence of the organization of the sparse vector on physical space. Maps are generated which provide the location of the neighbours of a certain sparse point, and these are used in the finite difference approximations to the equations of motion. The construction of these maps in the sparse system makes it possible for several cells to have the same neighbour or several virtual points to map to the same point. Any map is not unique and may be altered when desired. Vector subscripts can easily be generated to isolate a sub-region (e.g. a particular layer, all ghost points, all wet cells interior to ghost points, open boundary points) of the domain. This simplifies the ability to access a certain set of points without having to individually test the status of the cell. A map also has the ability to act as a vector subscript and behave as a 'dynamic' vector subscript; i.e. a vector subscript that may alter itself. These characteristics of the sparse system can be exploited, especially in a pre-processing stage, to generate very efficient code that may easily be parallel processed or vectorized.
Issues arise as to how the sparse vector is generated, e.g. map (x,y) space first then z space or (z,x,y), place wet points first followed by ghost points or keep ghost points with the relevant wet points etc. These issues may impact on the versatility of the sparse vector (e.g. using 2D maps as a subset of 3D maps, using vertical maps as neighbors in the sparse vector). SHOC employs a sparse map where (x,y) locations are included in the vector first, followed by the ghost points for that layer. This is done on a layer by layer basis, starting with the surface layer. The 2D maps then become a subset of the 3D maps, and map duplication is avoided.

The SHOC code is written such that there exists no checking of the status of a cell and no dynamic memory allocation. These operations are performed at an initialisation, or preprocessing, stage. The preprocessing of cell status requires that vectors are generated which contain information regarding the sparse locations of cells requiring frequent access, i.e.

1. The points to process at every time-step.
These must be supplied for tracers, x velocity and y velocity since these state variables occupy different locations on the Arakawa C grid and hence the same cell may contain both land and water points depending on whether the cell center or face is considered. These vectors contain all points that may potentially be wet; at every time-step these vectors must be packed to eliminate cells which have become dry.

2. Ghost points for tracers, x and y velocity.
These sparse locations are used to set the lateral (land) boundary condition. Generally for tracers this is a no-normal flux (i.e. no-gradient) condition and for velocity may be no-slip, free-slip or partial-slip. Again vectors are required for each cell face and cell center (3 vectors total). Note that there exists an additional ghost layer above the surface and below the bottom so as to define the surface and bottom boundary conditions. These ghost cells are not included in the lateral ghost vectors but form a subset of the sparse system proper and are accessed via the spatial maps.

3. Interior points to ghost points.
These sparse locations represent the first interior cell (wet cell) to the ghost points and are required in order to assign the values at the lateral boundaries.

4. Open boundary points.
The cells for each open boundary are stored in a vector, which in turn is stored in a data structure containing all additional data defining that boundary. This would include the boundary condition type and a pointer to a spatial map that provides interior sparse locations to the open boundary.

Additionally the sparse array is generated at the pre-processing stage and all maps are defined. These maps are 'self-pointing', meaning that a map on a boundary that would map outside the domain will map to itself. This removes the necessity to test for proximity to the boundary and invoke alternate practices when higher order numerics are employed near the boundaries. The maps consist of;

The sparse array is generated by assessing the status of every cell in the Cartesian grid. The cell status will fall into one of three categories, vis;

1. A cell that may potentially contain water (wet cell)
2. A cell that will never contain water (dry cell)
3. The first dry cell adjacent to a wet cell (ghost cell)

The wet cells can easily be identified on the basis of the value of the bathymetry, i.e. those cells whose bathymetry value is less than the maximum expected elevation value are considered wet. Also, any cell below the sea floor is considered dry. The generation of the sparse array therefore requires two two-dimensional arrays as input. One contains the bathymetry value, where a land mask value is included. The arrays used in a 3D model must contain vertical discretization, and a second array, kbot[i][j], should exist containing the k index of the bottom-most vertical layer given any (i,j) location. These are then used to generate a three-dimensional array, cell(i,j,k), where if k>kbot[i][j] cell(i,j,k)=0 and if (i,j) corresponds to a land location cell(i,j,k)=0. For all other cells cell(i,j,k)=wet $\neq$ 0.

12.1 Counting the wet cells

The first task in generating the sparse grid is to count the number of wet cells. A Cartesian to sparse map is introduced where every wet cell in the Cartesian grid is given a unique sparse coordinate, e.g. if (i,j,k) are the Cartesian indicies in the x,y and z directions respectively, and c is the sparse coordinate, then:

Map[i][j][k] = c

The Map is then populated with wet sparse locations;

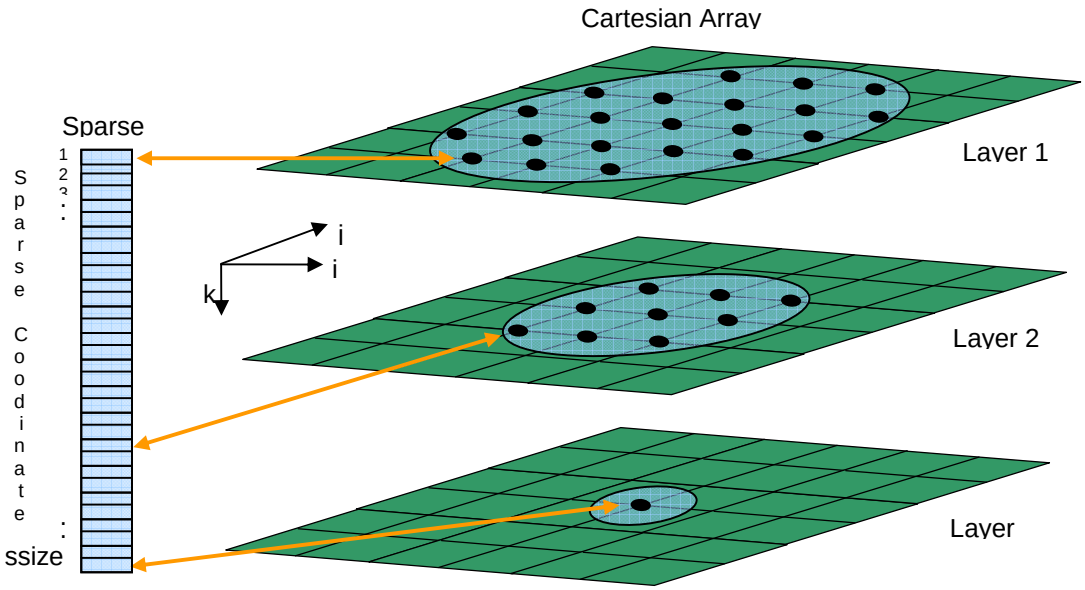
Loop 1

```
c=1;
for(k=0; k<zsize; k++)
  for(j=0; j<ysize; j++)
      for(i=0; i<xsize; i++) {
          if(cell (i,j,k) == wet) {
              Map[i][j][k] = c;
              c++;
          }
          else
              Map[i][j][k] = 0;
      }
```

where xsize, ysize and zsize are the (x,y,z) dimensions of the grid. The order that the sparse array is filled is arbitrary; here it is chosen to fill the sparse array with locations from horizontal layers first, for every layer. The sparse array then gets filled with wet points as depicted in Figure 12.1. The number of wet points in each layer and the sparse location of the last wet point in each

layer is stored to a buffer. These quantities are used at a later stage to insert the ghost cells into the sparse array.

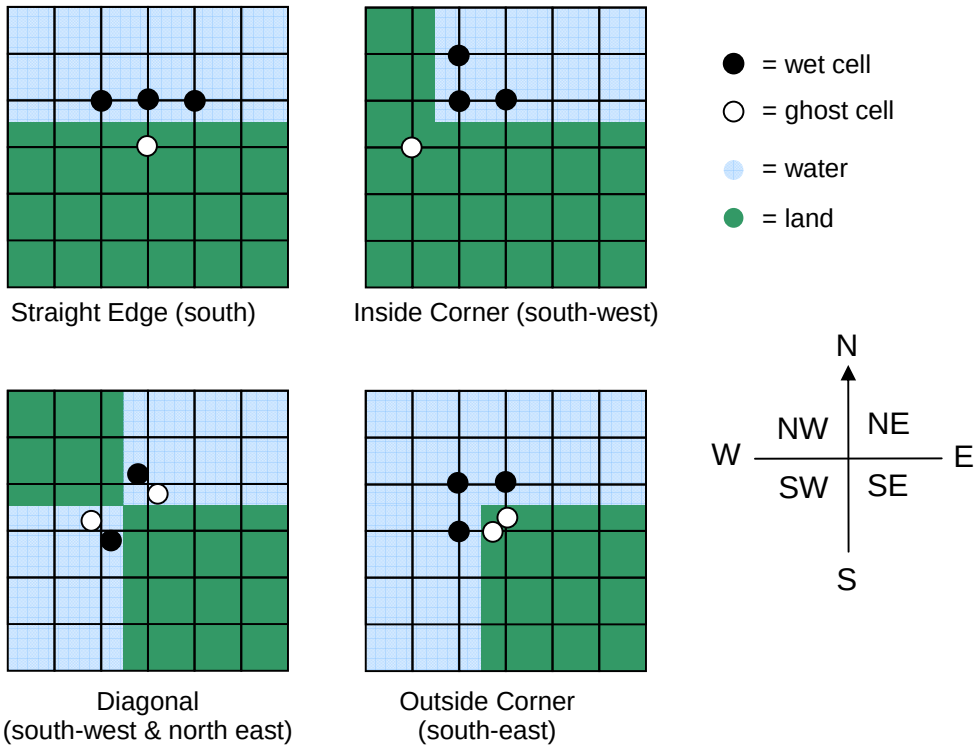
Figure 12.1 : Mapping of the Cartesian array to the sparse array. Only wet points (i.e. the cells enclosed by the ellipses) are mapped. Cells are mapped to the sparse array from the i direction first, followed by the j and k directions. The size of the sparse array is ssize.



12.2 Counting the ghost cells

The ghost cells to be represented in the sparse array are then counted. Ghost cells are defined as in Figure 12.2. The outside corner case requires two ghost cells to represent the wet cells, since two wet cells have the same adjacent land cell. The identification of an outside corner falls under the straight edge definition of a ghost cell, and since an outside corner comprises two straight edges, then two ghost cells are identified. In the case of a headland one cell wide it is possible for three ghost cells to occupy the same Cartesian location. These ghost cells are referred to as multiple ghost cells. In the diagonal case, the ghost cell must share the same Cartesian location with a wet cell. These multiple cells do not present a problem when constructing the sparse array – multiple cells are simply identified at this stage irrespective of their position in the Cartesian array and inserted in the sparse array later. Diagonal and inside corner ghost cells are generally only required if some form of interpolation is required over the grid (e.g. a semi-Lagrangian scheme is used to solve for momentum or tracer advection, or 2-way nesting is performed where a coarse grid needs to be populated). In order to prescribe the bottom boundary conditions, a ‘sediment’ ghost layer is included at the end of the sparse array. The number of these sediment cells is equal to the number of wet cells in the surface layer, denoted num_sed.

Figure 12.2 : Classification of ghost cells. Cells are placed at the cell center here; note that the land boundary passes through the cell faces.



Ghost cells are counted according to:

Loop 2

```
num_gc=0;
for(k=0; k<zsize; k++)
  for(j=0; j<ysize; j++)
      for(i=0; i<xsize; i++) {
          c = Map[i][j][k];
          if(se_oc(c))num_gc++;
          if(di_ic(c))num_gc++;
      }
```

where num_gc is the number of ghost cells, se_oc(c) represents the straight edge/outside corner ghost cell scenario and di_ic(c) represents the diagonal/inside corner ghost cell scenario illustrated in Figure 12.2 and defined below.

Denoting :
c = Map[i][j][k]
xp1 = Map[i+1][j][k]
xm1 = Map[i-1][j][k]
yp1 = Map[i][j+1][k]
ym1 = Map[i][j-1][k]

then se_oc(c) is defined as non-zero if:

West straight edges : c ≠ 0 && xm1 == 0
East straight edges : c ≠ 0 && xp1 == 0
South straight edges : c ≠ 0 && ym1 == 0
North straight edges : c ≠ 0 && yp1 == 0

South-west outside corners : c ≠ 0 && xm1 != 0 && ym1 ≠ 0 && Map[i-1][j-1][k] == 0
South-east outside corners : c ≠ 0 && xp1 != 0 && ym1 ≠ 0 && Map[i+1][j-1][k] == 0
North-west outside corners : c ≠ 0 && xm1 != 0 && yp1 ≠ 0 && Map[i-1][j+1][k] == 0
North-east outside corners : c ≠ 0 && xp1 != 0 && yp1 ≠ 0 && Map[i+1][j+1][k] == 0

And di_ic(c) is defined as non-zero if:

South-west diagonals : c ≠ 0 && xm1 == 0 && ym1 == 0 && Map[i-1][j-1][k] ≠ 0
South-east diagonals : c ≠ 0 && xp1 == 0 && ym1 == 0 && Map[i+1][j-1][k] ≠ 0
North-west diagonals : c ≠ 0 && xm1 == 0 && yp1 == 0 && Map[i-1][j+1][k] ≠ 0
North-east diagonals : c ≠ 0 && xp1 == 0 && yp1 == 0 && Map[i+1][j+1][k] ≠ 0

South-west inside corners : c ≠ 0 && xm1 == 0 && ym1 == 0 && Map[i-1][j-1][k] == 0
South-east inside corners : c ≠ 0 && xp1 == 0 && ym1 == 0 && Map[i+1][j-1][k] == 0
North-west inside corners : c ≠ 0 && xm1 == 0 && yp1 == 0 && Map[i-1][j+1][k] == 0
North-east inside corners : c ≠ 0 && xp1 == 0 && yp1 == 0 && Map[i+1][j+1][k] == 0

The routines se_oc() and di_ic() should return a unique code for each situation. It is useful for these codes to be powers of 2 so that several configurations can be represented in the same returned code. For example, setting:

SW_IC=1, SE_IC=2, NW_IC=4, NW_IC=8

Then the statement:

if(di_ic(c) & NW_IC)

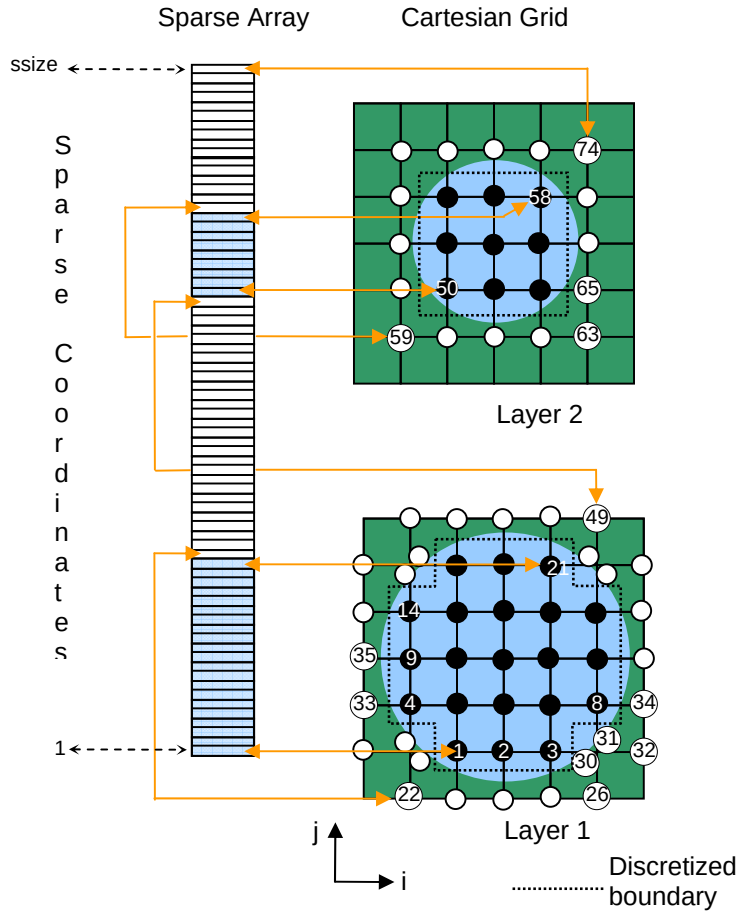
indicates that the sparse cell c constitutes a north west inside corner.

12.3 Re-ordering the sparse coordinates

Once the number of ghost cells has been established, the sparse array is reordered so that for each layer the wet cells sequentially precede the ghost cells (see Figure 12.3). This re-arrangement is not necessary, but allows the sparse coordinates for the 2D mode to become a subset of the 3D sparse array, hence any 2D sparse arrays or maps need not be explicitly generated. The number of wet points in each layer and the sparse location of the last wet point in each layer stored earlier are used to re-define the mapping array, Map[][][], so that wet cells are assigned to the re-ordered sparse locations. Cells corresponding to ghost cells in Map[] remain zero valued at this stage. The sparse location of the last wet cell in each layer is re-calculated and stored in an array, e.g. with reference to Figure 12.3;
last_wet[1] = 21
last_wet[2] = 58

Generally : last_wet[k] = cc

Figure 12.3 : Re-ordering of the sparse array. The wet and ghost cells in the Cartesian grid are numbered with their corresponding sparse array coordinate.



12.4 Wet – wet cell spatial maps

At this stage the sparse array is complete, and the spatial maps which define the sparse locations of any cell's neighbours in Cartesian space must be established. For wet cells that map to other wet cells, these maps are defined as;

xp1(c) = Map[i+1][j][k]
xm1(c) = Map[i-1][j][k]
yp1(c) = Map[i][j+1][k]
ym1(c) = Map[i][j-1][k]
zp1(c) = Map[i][j][k+1]
zm1(c) = Map[i][j][k-1]

These maps are initialised to map to themselves, i.e. xp1(c) = c, c=1,ssize. Any cell which is not subsequently assigned an alternate sparse coordinate will remain 'self-mapping', e.g. with reference to Figure 3.3, xm1[33] = 33 or xp1[34] = 34. This practice effectively sets a no-gradient condition beyond the limits of the domain and allows higher order schemes to be used at the boundaries without having to test if there are enough wet cells present to accommodate the scheme.
The spatial maps of wet cells that map to other wet cells are easily established, e.g;

Loop 3

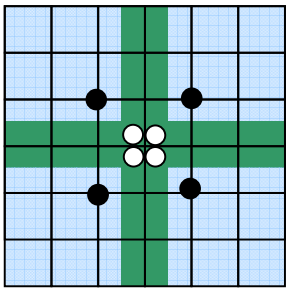
```
for(k=0; k<zsize; k++)
  for(j=0; j<ysize; j++)
      for(i=0; i<xsize; i++) {
          c = Map[i][j][k];
          if(c != 0 && Map[neighbour] != 0) {
              wet_wet_map(c) = Map[neighbour];
          }
      }
```

where wet_wet_map(c) is the relevant spatial map that maps the wet cell to another wet cell and Map[neighbour] is the value of Map[][][] one cell in any direction (e.g. Map[i+1][j][k], Map[i][j-1][k], Map[i][j][k+1] etc.).

12.5 Wet - ghost cell spatial maps

Next the spatial maps for wet cells which map to ghost cells and ghost cells which map to wet cells are assigned. Wet cells corresponding to diagonals, straight edges, outside/inside corners are identified in Map[] on the basis of their neighbour's value (zero = ghost, non-zero = wet) as illustrated in Figure 12.2. For each layer, a ghost cell counter is initialised to cc = last_wet[k]+1, then looping through the grid in the correct order, when a wet cell is identified neighbouring a ghost cell, the relevant maps are assigned to cc and cc is incremented. This process will capture all the straight edge and outside corner ghost - wet cell maps.
At this stage four new arrays are introduced which are referred to as the corner arrays (sw_c[i][j][k], se_c[i][j][k], nw_c[i][j][k] and ne_c[i][j][k]). These arrays are three dimensional arrays covering the entire Cartesian grid and are used to store ghost cell sparse locations for inside corners and diagonals. There must exist four of these arrays since it is possible for a Cartesian location to be part of multiple inside corners (a maximum of four, e.g. Figure 12.4). The sparse coordinates in the corner arrays are used to establish the ghost – ghost spatial maps involving inside corners and diagonals at a later stage. These arrays are initialised to zero.

Figure 12.4 : Multiple ghost cells at inside corners.



Loop 4

```
for(k=0; k<zsize; k++)
  cc = last_wet[k] + 1;
  for(j=0; j<ysize; j++)
      for(i=0; i<xsize; i++) {
          c = Map[i][j][k];
          if(c != 0 && se_oc(c) ) {
              wet_ghost_map(c) = cc;
              ghost_wet_map(cc) = c;
              if(Map[orthogonal][k] != 0)orthogonal_ghost_wet_map(cc)  = Map[orthogonal][k];
              cc++;
```

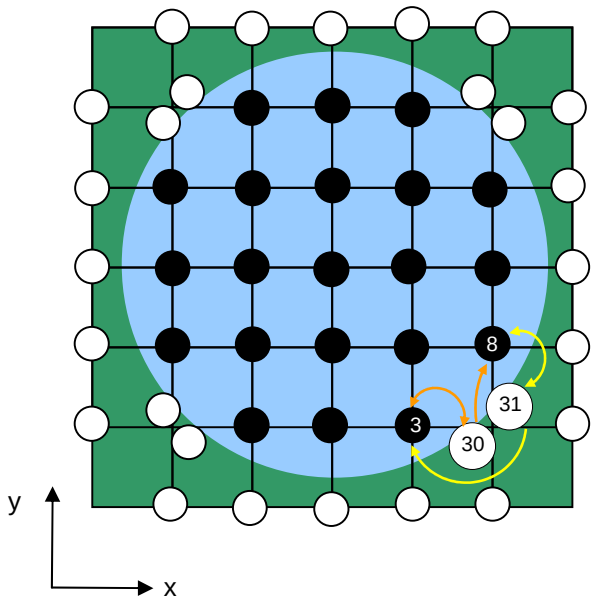
```
        }
        if(c !=0 and di_ic(c) ) {
            corner[i][j][k] = cc;
            cc++;
        }
    }
```

where wet_ghost_map(c) is the relevant spatial map that maps the wet cell to the ghost cell and ghost_wet_map(c) is the relevant spatial map that maps the ghost cell to the wet cell. With reference to Figure 12.3, examples of these maps are:

wet_ghost_map : xm1[9] = 35 or xp1[8] = 34
ghost_wet_map : xp1[33] = 4 or xm1[30] = 3

Note that these loops are written in condensed form, and should be expanded to test the edges/corners/diagonals for each compass direction. Hence corner[][][] refers to each individual corner array sw_c[][][], se_c[][][] etc. depending on whether di_ic() is true for that particular corner/diagonal.
If a straight edge map exists that is part of an outside corner, then an additional map from the ghost cell to the wet cell must be established in a direction orthogonal to the initial straight edge map (i.e. orthogonal_ghost_wet_map(cc) above). In this case Map[orthogonal][k] denoted above represents the sparse coordinate of the neighbouring wet cell to the outside corner ghost cell in a direction orthogonal to the original straight edge map. This is illustrated in Figure 12.5 for a bottom right outside corner.

Figure 12.5 : Spatial maps from bottom right outside corners.
xp1[3] = 30 and xm1[30] = 3 with the additional y direction map yp1[30] = 8
with c = 3, cc = 30, Map[orthogonal][1] = 8
wet_ghost_map = xp1, ghost_wet_map = xm1 and
orthogonal_ghost_wet_map = yp1.
ym1[8] = 31 and yp1[31] = 8 with the additional x direction map xm1[31] = 3
with c = 8, cc = 31, Map[orthogonal][1] = 3.
wet_ghost_map = ym1, ghost_wet_map = yp1 and
orthogonal_ghost_wet_map = xm1.



Note that the additional map from the outside corner ghost cell only maps one way to the wet cell; the reverse map from the wet cell maps to the other multiple ghost cell constituting the outside corner and is established via a straight edge map in the orthogonal direction.

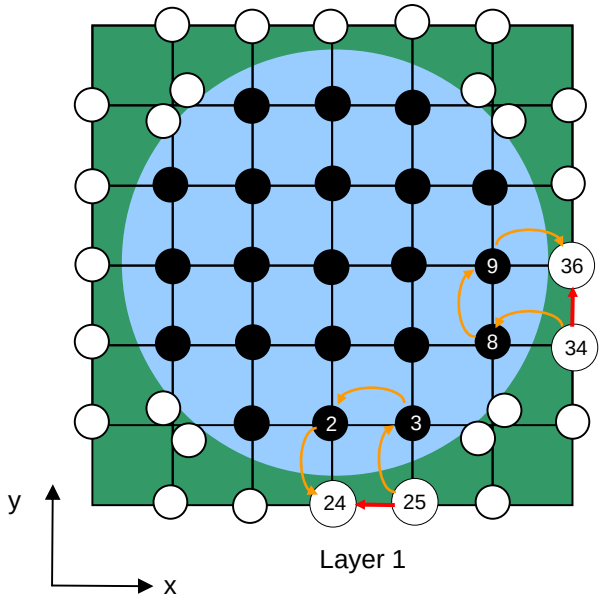
All ghost coordinates are now defined in terms of their position relative to their neighbours; the straight edge/outside corner ghost cells are defined via the wet – ghost spatial maps and the diagonal/inside corner ghost cells are stored in the corner arrays.

12.6 Ghost – ghost cell spatial maps : straight edges and outside corners

The maps from ghost cells to other ghost cells must now be assigned to complete the spatial maps.
Straight edges and outside corner ghost cells cannot be directly accessed via an (i,j,k) coordinate, but only accessed via a wet_ghost map. In this case a wet cells is checked to see if it's neighbour is a dry ghost cell. If so, then the map from this ghost cell to other neighbouring ghost cells (if any exist) is made by approaching the neighboring ghost cells via their corresponding wet cell. This process is illustrated in Figure 12.6. Here a wet cell is identified at sparse coordinate 3, i.e. (i,j,k) coordinates exist where c = Map[i][j][k] = 3. In addition, Map[i-1][j][k] = 2 is also wet. The wet ghost map established in Loop 4 is used to get the sparse location of the ghost cell associated with sparse coordinate 3, i.e. cc = ym1(c) = 25. The maps from cell coordinate 25 to other ghost cells (viz. ghost cell 24) is established via:

If c = 3 and cc = 25 : xm1[cc] = ym1[Map[i-1][j][k] = ym1[2] = 24
If c = 8 and cc = 34 : yp1[cc] = xp1[Map[i][j+1][k]] = xp1[9] = 36

Figure 12.6 : Ghost – ghost cell maps for straight edges and outside corners.



This approach is favoured in preference to using an equivalent array to corner[][][] to store straight edge/outside corner ghost cell locations due to the fact that multiple ghost cells on outside corners will result in non-unique ghost cell locations for any (i,j,k) outside corner Cartesian location.
The ghost_ghost mapping for straight edges and outside corners is formalized below.

Loop 5

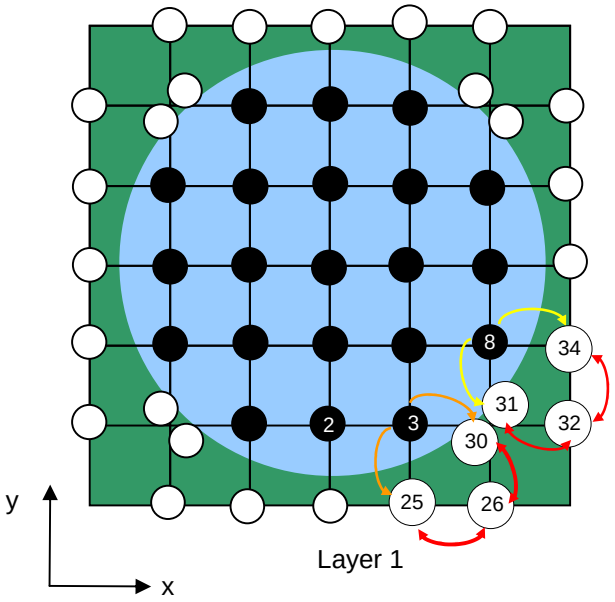
```
for(k=0; k<zsize; k++)
  for(j=0; j<ysize; j++)
      for(i=0; i<xsize; i++) {
          c = Map[i][j][k];
          if(c != 0 and se_oc(c) ) {
              cc = wet_ghost_map(c);
              ghost_ghost_map(cc) = wet_ghost_map(Map[neighbour]);
          }
      }
```

12.7 Ghost – ghost cell spatial maps : diagonals and inside corners

In order to retrieve the sparse coordinates of ghost cells, the corner[] arrays must be used. Inside corners are defined as in Figure 12.2, where the diagonal cell from a wet cell is dry and the neighbours in the x and y directions are also dry. The sparse locations of the ghost cells neighbouring the inside corner cell can be obtained via a wet_ghost map established in Loop 4. The relevant spatial maps from these ghost cell locations to the inside corner is defined as the value of corner[][][] array at the (i,j,k) Cartesian location of the inside corner. This same strategy is applied to ghost_ghost maps at diagonal ghost cells. This process is illustrated in Figure 12.7 and Loop 6. Here a wet cell is identified at sparse coordinate 3, i.e. (i,j,k) coordinates exist where c = Map[i][j][k] = 3. In addition, Map[i+1][j][k], Map[i][j-1][k] and Map[i+1][j-1][k] are dry (i.e. equal to zero). The wet ghost map established in Loop 4 is used to get the sparse location of the ghost cell associated with sparse coordinate 3, i.e. cc = ym1(c) = 25. The maps from cell coordinate 25 to the inside corner (viz. ghost cell 26) is established via:

If c = 3 and cc = 25 : xp1[cc] = se_c[i+1][j-1][k] = 26
If c = 3 and cc = 30 : ym1[cc] = se_c[i+1][j-1][k] = 26
If c = 3 and cc = 25 : xm1[se_c[i+1][j-1][k]] = ym1[c] = 25
If c = 3 and cc = 30 : yp1[se_c[i+1][j-1][k]] = xp1[c] = 30

Figure 12.7 : Maps from straight edges and outside corners to inside corners



Loop 6

```
for(k=0; k<zsize; k++)
```

```
for(j=0; j<ysize; j++)
    for(i=0; i<xsize; i++) {
        c = Map[i][j][k];
        if(c != 0 and di_ic(c)) {
            cc = wet_ghost_map(c);
            ghost_ghost_map1(cc) = corner[diagonal];
            ghost_ghost_map2(corner[diagonal]) = cc;
        }
    }
```

where corner[diagonal] is the value of corner[][][] diagonally opposite cell (i,j,k) (e.g. se_c[i+1][j-1][k], ne_c[i+1][j+1][k], nw_c[i-1][j+1][k+1] or sw_c[i-1][j-1][k]). Again the corner[] array used (se_c, sw_c etc.) should depend on the code returned by di_ic(), i.e. the above code segment should be expanded to test every compass orientation of corners/diagonals. Here ghost_ghost_map1 maps *to* the inside corner and ghost_ghost_map2 maps *from* the inside corner.

12.8 Vertical maps

Spatial maps in the vertical direction are straightforwardly established. For wet_wet spatial maps, this becomes:

```
c=Map[i][j][k];
  if(c != 0) {
      zp1(c) = Map[i][j][k+1];
      zm1(c) = Map[i][j][k-1];
  }
```

For ghost_ghost vertical spatial maps on inside corners and diagonals the corner[] array is used:

```
c=Map[i][j][k];
  if(c != 0 and di_ic(c) == 0) {
      cc = corner[i][j][k];
      zp1(cc) = corner[i][j][k+1];
      zm1(cc) = corner[i][j][k-1];
  }
```

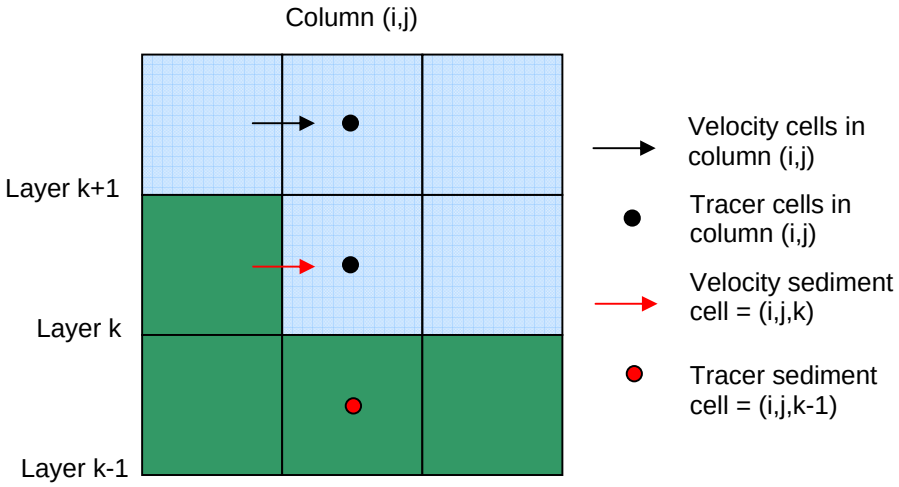
For ghost_ghost spatial maps on straight edges and outside corners the ghost cells are again defined via the wet cells, e.g;

```
c=Map[i][j][k];
  if(c != 0 and se_oc(c) == 0) {
      cc = wet_ghost_map(c);
      zp1(cc) = wet_ghost_map(Map[i][j][k+1]);
      zm1(cc) = wet_ghost_map(Map[i][j][k-1]);
  }
```

12.9 Sediment maps

Vertical maps to the sediment ghost cells must be established to complete the sparse mapping functions. These are easily established given the vertical coordinate of the bottom cell in the water column, kbot[i][j]. Note that the sediment ghost cells lie at the very end of the sparse array. There must exist separate sediment layer maps for tracers, u1 and u2 velocities since the sediment may fall on different layers depending on whether the cell face or center is considered (see Figure 12.8).

Figure 12.8 : Side view example of sediment locations corresponding to a given water column at (i,j). Cell face sediment locations differ to cell center sediment locations.



Loop 7

```
cc = ssize – num_sed + 1;
for(j=0; j<ysize; j++)
  for(i=0; i<xsize; i++) {
      k = kbot[i][j];

      c = Map[i][j][k];
      if(not a solid cell) {
          zm1[c] = cc;
          zp1[cc] = c;
          cc++;
      }
  }
```

12.10 Lateral boundary vectors

A vector containing all the ghost cell locations in the sparse array is useful to generate. This vector has size num_gc and is easily filled in Loop 4. In addition, an array may be established which contains the nearest wet neighbour to the boundary cells. This is useful for establishing no-flux boundary conditions across the boundary. If bca[] is the ghost vector, or boundary cell array, and bin[] is an array of wet neighbours to bca[], then Loop 4 may be modified:

Loop 8

```
lc=1;
for(k=0; k<zsize; k++)
  cc = last_wet[k] + 1;
  for(j=0; j<ysize; j++)
      for(i=0; i<xsize; i++) {
          c = Map[i][j][k];
          if(c != 0 && se_oc(c) ) {
              bca[lc] = cc;
              bin[lc] = c;
              lc++;
```

```
            wet_ghost_map(c) = cc;
            ghost_wet_map(cc) = c;
            if(Map[orthogonal][k] != 0)orthogonal_ghost_wet_map(cc)  = Map[orthogonal][k];
            cc++;
        }
        if(c !=0 and di_ic(c) ) {
            bca[lc] = cc;
            bin[lc] = c;
            lc++;
            corner[i][j][k] = cc;
            cc++;
        }
    }
```

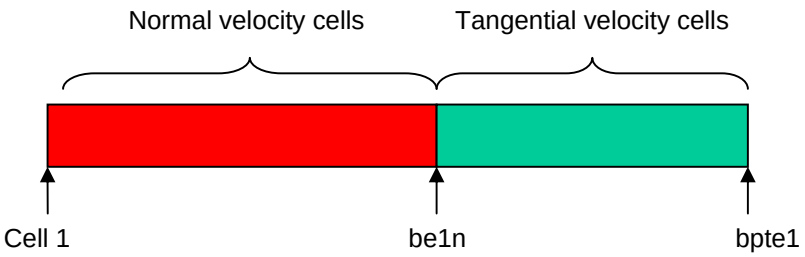
The Arakawa C grid defines velocity at different locations than elevation/tracers (i.e. cell faces as opposed to cell centers). This means that the u1 velocity on western edges and u2 velocity on southern edges actually occupy a land boundary. Since flow is not allowed through the coast, these cells must be set to zero velocity, i.e. they become the ghost cells for normal velocity components. On eastern and northern boundaries the normal velocity ghost cells remain the same as the cell centers. Western and northern edges may be determined on the basis of the inside cell to a ghost cell mapping to that ghost cell. Therefore, the normal velocity boundary arrays are constructed from the cell centered boundary arrays thus:

e.g. for e1 velocity:

```
c1=1;
for(cc=1; cc<=size(bca); cc++) {
  c=bca[cc];
  lc=bin[cc];
  if(west_map(lc)==c)
      bca_e1[c1]=east_map[c];
  else
      bca_e1[c1]=c;
  c1++;
}
```

The velocities tangential to solid boundaries may occupy different locations in the grid than normal velocities (always at the cell centered ghost cell) and different boundary conditions are applied to these cells. The boundary arrays for velocity are organized so the first be1n cells contain the normal velocity boundary locations followed by the tangential velocity locations (Figure 12.9.). Separate vectors are generated for 2D and 3D velocity arrays.

Figure 12.9 : Lateral velocity boundary vector configuration

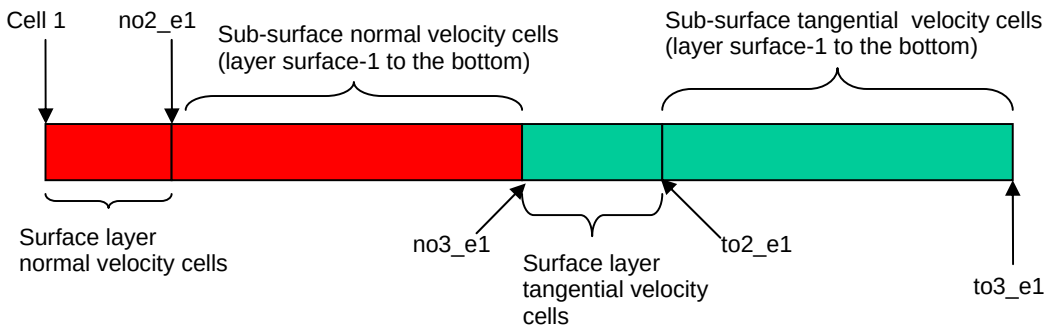


Additional arrays may be established to contain sparse locations of open boundary cells, the sparse cells in each layer, the sparse cells of the free surface, bottom layer or sediment layer, or any sub-region of the grid that may simplify computations during the integration.

12.11 Open boundary vectors

The locations of cells on the open boundaries which need to be overwritten at each time-step with elevation tracer and velocity information are stored in separate vectors. Due to the stagger imposed by the Arakawa C grid, elevation/tracers, u_1 and u_2 velocities on the open boundaries may occupy different locations in the grid, hence these components each have their own open boundary vectors.
The open boundary vectors for tracers and elevation are arranged so that the first no2_t cells contain the surface open boundary locations, followed by no3_t-no2_t sub-surface OBC cells. The open boundary velocity (obc_e1 and obc_e2) are arranged so that the first no2 cells contain the surface normal velocity locations, followed by no3-no2 sub-surface normal velocity locations. Following this the surface tangential velocity locations exist, followed by the sub-surface tangential velocity locations. If the surface layer tangential cells are to be accessed, for example, the loop would run from no3_e1+1 to to2_e1. This is illustrated in Figure 12.10. Additional vectors may be established which contain locations one and two cells interior to the OBC cell, and an array of locations applicable to cyclic type boundary conditions.

Figure 12.10 : Open boundary vectors for u_1 velocity



12.12 Surface and bottom vectors

Vectors may be established which contain the sparse location of the surface cells and the bottom cells. The sediment cells are then easily identified as the cells beneath the bottom cells. The surface cells are simply the cells corresponding to the 2D grid, however, these cells may need to be recalculated at each time-step to allow for movement of the free surface. The bottom cells have the same size as the surface sparse array, and is simply the sparse location of the bottom location, kbot[][], for cell centers, e.g.

```
c1 = 1;
for(j=0; j<ysize; j++)
  for(i=0; i<xsize; i++) {
      k = kbot[i][j];
      c = Map[i][j][k];
      if(not a solid cell) {
          botm[c1] = c;
          c1++;
      }
  }
```

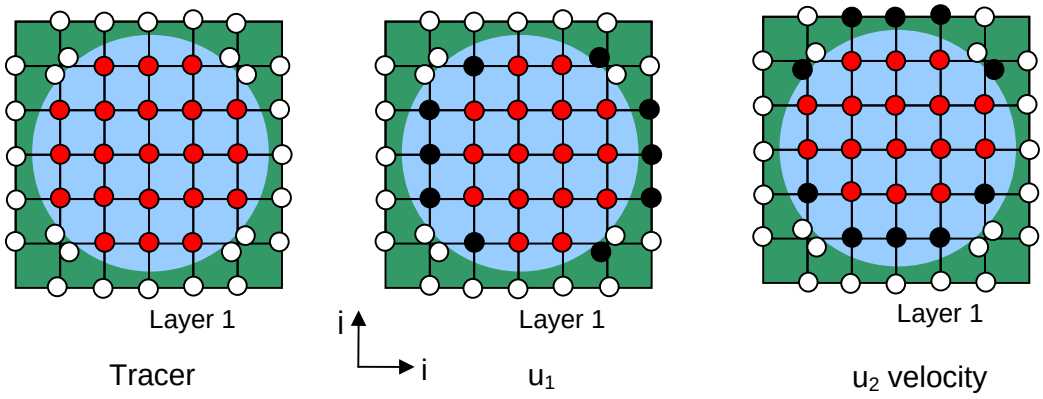
For velocity, as noted above for the sediment maps, the bottom location may occupy a different location to the cell center, and the bottom is identified via:

```
c1=1;
for(j=0; j<ysize; j++)
  for(i=0; i<xsize; i++)
      for(k=0; k<zsize; k++) {
      c = Map[i][j][k];
      if(not a solid cell) {
          if(cell k==(not a solid face) && cell k-1 == (solid face)) {
              botm_e1[c1]=c;
              c1++;
          }
      }
  }
```

12.13 Cells to process vectors

The cells that are required to be updated at every time-step are stored in vectors for each window. Again these vectors differ for u1, u2 velocity and tracer since solid boundaries differ depending on whether the cell face or center is considered. Once these vectors are established the domain may be updated without any cell status checking within the code. These vectors are required for u1, u2 velocity, tracers, u1av, u2av velocity and elevation. This is illustrated in Figure 12.11.

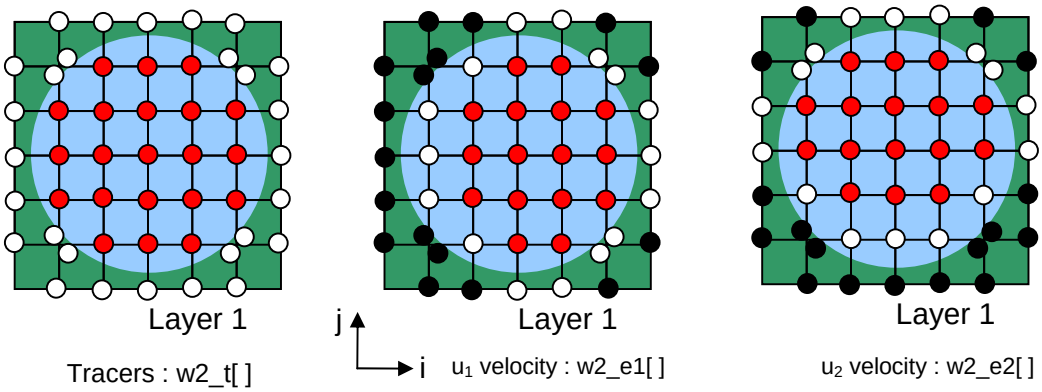
Figure 12.11 : Cells required to be updated (red circles) for tracers, u1 and u2 velocity. The black circles correspond to cells which enter the numerical computations and must assume zero velocity.



The solution of the advection and diffusion equations in flux form requires that fluxes be calculated at half cell distances (e.g. on the cell faces for cell centered variables). The differences of these fluxes then yields the updated solution. This means that there must exist a vector of cells to process for calculating the fluxes and a vector for updating the variable. Generally the cells to process for fluxes include one extra cell beyond the boundary of the wet domain. If the numerical technique employed uses a flux form of advection/diffusion schemes, the cells to process vectors may contain cells depicted in Figure 12.12. The exact format of these vectors is generally dependent on the type of numerical schemes employed.

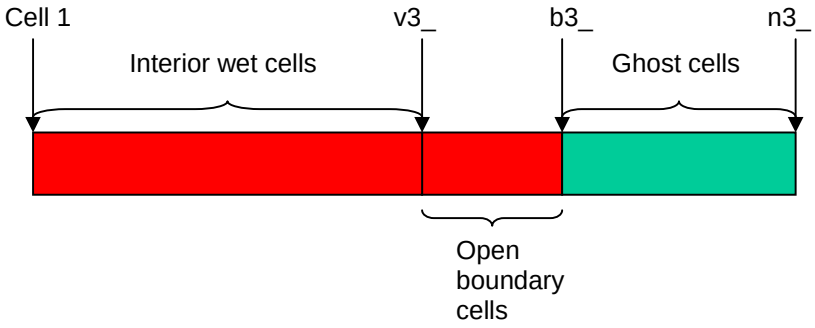
Figure 12.12 : Cells to process vectors required for the flux form of advection / diffusion schemes. Red locations are the cells which are updated, red & white locations are cells which store fluxes and black locations do not enter the finite

difference computations and are not required to be included in the cells to process vectors.



Rather than create two vectors containing cells to process for fluxes and updates, one vector exists where a given number of the first consecutive cells corresponds to the wet cells to update and the entire vector corresponds to the wet cells for fluxes. This amalgamation of the processing vectors into one vector means that the 2D vectors cannot exist as the first cells in the 3D vectors and consequently separate processing vectors exist for the 2D variables (e.g. w3_ for 3D variables, w2_ for 2D variables). The cells to process vectors are organized as illustrated in Figure 12.13.

Figure 12.13 :Cells to process vectors for 3D variables.



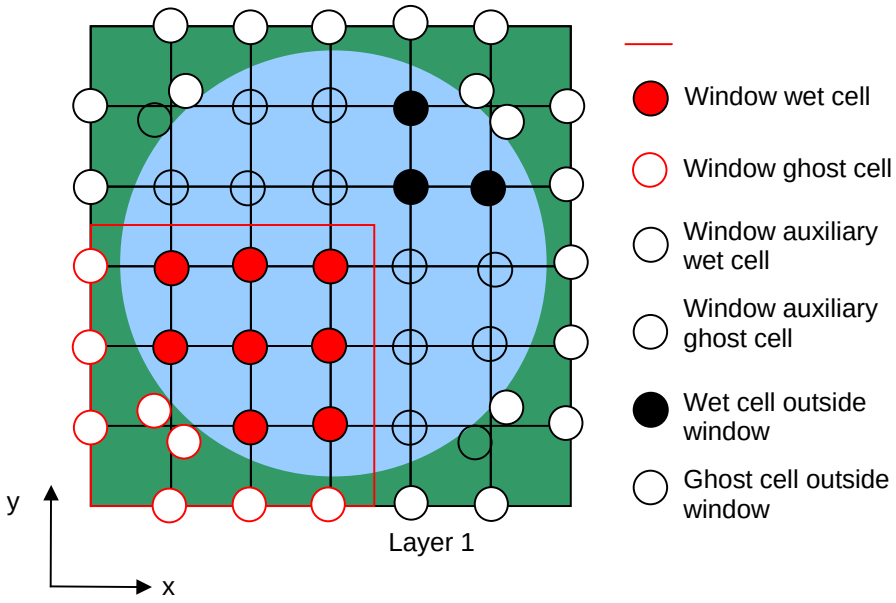
13. Generating Sparse Windows

In order to facilitate distributed / parallel processing, tune vectorisation and process a domain that requires more memory than is available, an approach is adopted whereby the full domain is divided into a number of subsets, or 'windows'. Each window is sequentially processed (i.e. the window 'slides' over the full domain). The updated state variables resulted from processing each window are then written to the array containing the state variable at the forward time-step.
Suppose the sparse array of all potentially wet points (global sparse array) is stored in gsa[c]. This vector is subdivided into nw windows, or local arrays, wsa[n,c], each of size naux[n], which each contain a unique subset of gsa, such that;

$$gsa[] = \sum_{n=1}^{nw} wsa[n][] \qquad 13.1$$

All operations performed on a window are done so with reference to the local sparse coordinates for that window, i.e. cells numbered from 1 to naux[n]. This means that spatial maps must be generated which map cells in all directions on the local sparse coordinate system in each window. These can be easily generated from the global spatial maps once the extent of the window is known. Near the edges of a window it is possible that some spatial mappings will map to cells in another window. These extra, or auxiliary cells, must be included in the window. The number of these auxiliary cells in each window is generally dependent on the order of the advection and diffusion schemes used. For a second order scheme using the ULTIMATE filter, an extra two cells beyond the window boundary are required. This is illustrated in Figure 13.1.
Note that semi-Lagrangian schemes are not suitable to use with windows since it is possible for these schemes to run with Courant numbers of 10 – 20, making it possible for 10 to 20 auxiliary cells being required. Also, the number of auxiliary cells required is dependent on the flow, making it necessary for dynamic allocation of auxiliary cells or initially including the maximum number anticipated. Clearly these requirements lead large operating inefficiencies from a memory perspective. Inside corner and diagonal ghost cells may be omitted from the window if a semi-Lagrangian scheme is not used.

Figure 13.1 : Cells required to define a window



The partitioning into cells into windows may be completely arbitrary, however there are several considerations that make it advantageous to include all cells in a water column into a window. This is primarily due to solving the vertical diffusion terms, which are solved implicitly and require all cells in a water column. The vertical integration of the density gradient for the pressure terms in the momentum equations also requires all cells to the surface above a given cell. While these constraints could be accommodated by including all cells in the water column not in the window as auxiliary cells, to avoid memory related inefficiencies arising from excessive auxiliary cell specification a restriction is imposed whereby all cells in the entire water column must be included in a window. Therefore, to specify a window a group of sparse locations in the surface layer is specified from which the window is generated including all cells from those surface cells to the bottom cells. Extra auxiliary cells are required for 'sediment' cells to specify bottom boundary conditions.

13.1 Local sparse system and local maps

As with the global sparse system, it is desirable to place all the surface cells in contiguous locations at the beginning of the local sparse array so that the 2D maps and vectors can be used as a subset of the 3D. The local sparse system therefore contains all surface wet cells followed by all surface auxiliary (wet and ghost) and ghost cells, followed by all sub-surface wet cells, followed by all sub-surface auxiliary (wet and ghost) and ghost cells (see Figure 13.2).
In order to generate the window arrays, a map must first be established which provides the window number and local window sparse coordinate from any given global sparse coordinate. An array of data structures is created where for each global coordinate, c;

n = fm[c].wn = window number of sparse coordinate c (range = 1 to wn)
cc = fm[c].sc = local sparse coordinate (range = 1 to naux[n])

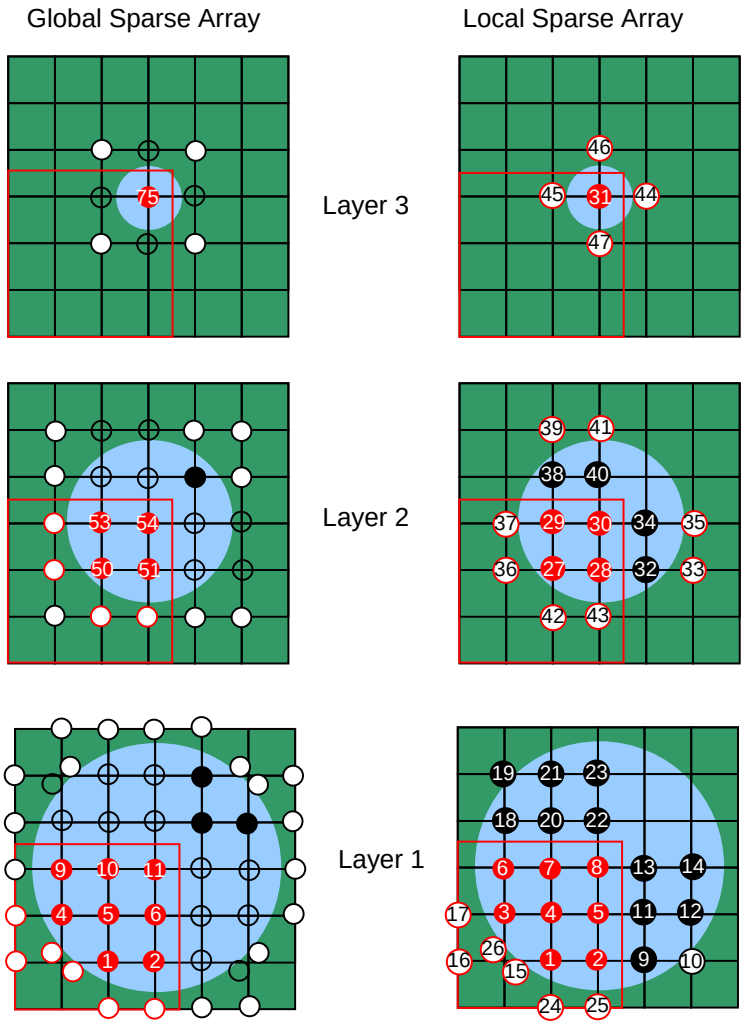
With reference to Figure 10, an example of this map may be:

fm[10].wn = 1
fm[10].sc = 7
fm[8].sc = 12

Once the global to local map is established, the local sparse system is established in two stages. First the surface wet and auxiliary cells are identified within the window. Suppose c is the global sparse coordinate, map(c) is the global sparse coordinate spatial map in any horizontal direction and nwet2D is the number of surface layer wet cells in the window number n. Wet cells are initially placed in the first nwet2D locations of the window, followed by auxiliary cells, e.g;

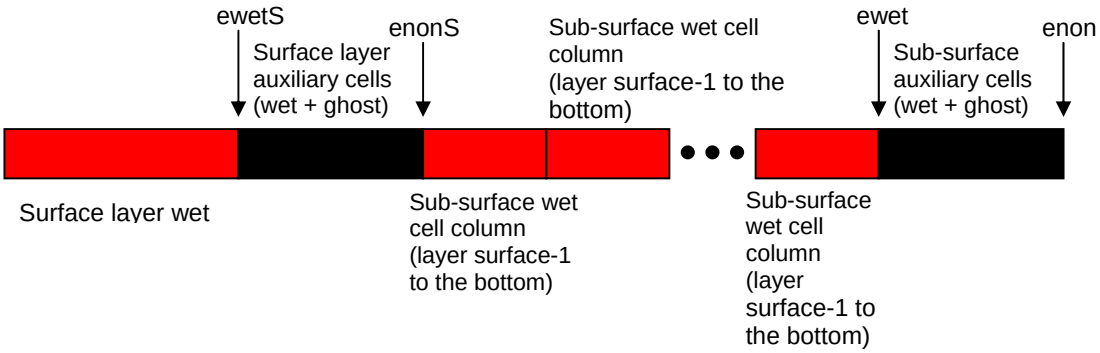
```
Loop 1
naux2D[n] = nwet2D + 1;
for(cc=1; cc<=naux[n]; cc++) {
  c = wsa[n][cc];
  c1 = map(c);
  if(c == surface layer cell) {
      if(fm(c1).wn == n)
          local_map(cc) = fm(c1).sc          /* Local map of cc is in window n */
      else {
          local_map[cc] = naux2D[n] ;        /* Append the auxiliary cell to the end of the 2D wet cells
          naux2D[n]++;
      }
  }
}
```

Figure 13.2 : Global to local window maps. The left panel labels sparse coordinates in the global system, and the right panel has sparse coordinates labeled in the local system. Note the first 26 locations contain all information relating to the surface layer.



Where local_map() is the spatial map in the local sparse system and naux2D[n] is the total number of surface cells in the window. This procedure effectively defines all the required surface auxiliary cells via the local spatial maps. Note that map(c) may map further than the nearest neighbour to a cell. The second stage is to set the local maps for all sub-surface cells and re-order the global to local map and the local sparse system so that the sequence of cells corresponds to the numbering depicted in Figure 13.2. The complete re-ordered local sparse system corresponds to the sequence illustrated in Figure 13.3.

Figure 13.3 : Re-ordered local sparse system. Note : snonS=ewetS+1 and snon=enonS+1.



The re-ordering of the global to local map can be accomplished via;

Loop 2

```
/* First re-order the surface layer cells */
c1 = 1;
for(cc=1; cc<=nauxS; cc++) {
   c = wsa2D[n][c];
   if(fm[c].wn==n) {
       fm[c].sc=c1;
       c1++;
   }
}

/* Re-order the sub-surface cells */
c2 = 1;
c1 = naux2D + 1;
for(cc=1; cc<=naux[n]; cc++) {
   c = wsa[n][cc];
   if(fm[c].wn == n && c != wsa2D[c2]) {
       fm[c].sc = c1;
       c1++;
       c2++;
   }
}
```

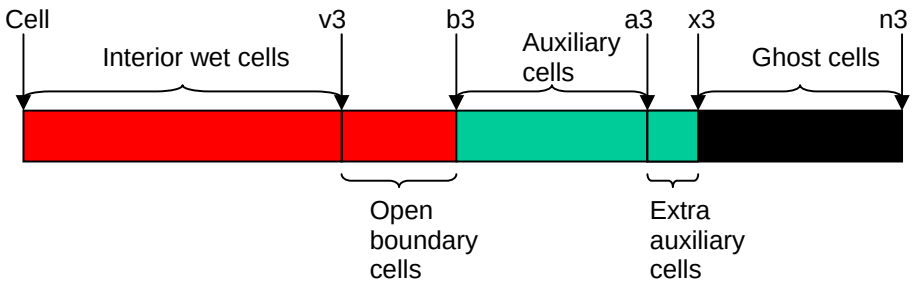
where n is the window number, n and wsa2D[cc][n] are the cells in the surface layer of window n (with a size of naux2D). It is assumed that wsa[][] is filled in the (k,i,j) direction and wsa2D[][] in the (i,j) direction. The local_map() for the sub-surface cells is established via a loop analogous to Loop 1 where auxiliary cells are appended to the entire existing local array.

13.2 Cells to process vectors

The cells that are required to be updated at every time-step are stored in vectors for each window. Again these vectors differ for u1, u2 velocity and tracer since solid boundaries differ depending on whether the cell face or center is considered. Once these vectors are established the domain may be updated without any cell status checking within the code. These vectors are required for u1, u2 velocity, tracers, u1av, u2av velocity and elevation. The solution of the advection and diffusion equations in flux form requires that fluxes be calculated at half cell distances (e.g. on the cell faces for cell centered variables). The differences of these fluxes then

yields the updated solution. This means that there must exist a vector of cells to process for calculating the fluxes and a vector for updating the variable. Generally the cells to process for fluxes include one extra cell beyond the boundary of the domain. Rather than create two vectors containing cells to process for fluxes and updates, one vector exists where a given number of the first consecutive cells corresponds to the wet cells to update and the entire vector corresponds to the wet cells for fluxes. This amalgamation of the processing vectors into one vector means that the 2D vectors cannot exist as the first cells in the 3D vectors and consequently separate processing vectors exist for the 2D variables. The cells to process vectors are organized as illustrated in Figure 13.4.

Figure 13.4 : Cells to process vectors



The extra auxiliary cells comprise of southern edge auxiliary cells for u1 vectors and western edge auxiliary cells for u2 velocity required to solve the advection equation for momentum. These vectors are refined at every time-step to produce vectors of wet cells which exclude those regions of the domain which may have dried.

14. Sparse Array Procedures

14.1 Lateral boundary conditions

The sparse array spatial maps and boundary arrays may be used to prescribe lateral boundary conditions in the domain and perform finite difference operations. The lateral boundary conditions at the ghost cells must always be set before the relevant parts of the equations of motion are solved so that the correct boundary condition is prescribed at the land/water interface.

Tracers.
A no-flux lateral boundary condition is achieved by prescribing a no-gradient condition at the ghost cells, i.e.

```
for(c=1; c<=num_gc; c++) {
   c1 = bca[c];
   c2 = bin[c];
   tracer[c1] = tracer[c2];
}
```

Velocity.
No-flux conditions normal to the lateral boundaries is achieved by directly setting the velocity on the boundary to zero. Note that it is assumed the boundary array here corresponds to the locations of velocity in the grid, which are be offset from the cell center (e.g. on the cell face for an Arakawa C grid – the vector bpt_e1 must be used). For the velocity in the x direction, u1, this may be written.

```
 for(c=1; c<=be1n; c++) {
   c1 = bca_e1[c];
   u1[c1] = 0.0;
}
```

The tangential velocity is prescribed so that a free-slip condition exists; i.e. a no-gradient condition is set on the tangential ghost cells:

```
for(c=be1n+1; c<=bpte1; c++) {
   c1 = bpt_e1(c);
   c2 = bin_e1(c);
   u1(c1) = u1(c2);
}
```

14.2 Finite difference operations : example – QUICKEST in sparse coordinates

The QUICKEST scheme (Quadratic Upstream Interpolation for Convective Kinematics with Estimated Streaming Term) is a 3rd order accurate upwind scheme described by Leonard (1979). This scheme has very little numerical diffusion and numerical dispersion is limited to one small ripple in the vicinity of fronts. QUICKEST is stable for $q \le 1$. The flux representation of the QUICKEST scheme is given by:

$$T_i^{t+1} = T_i^t - (q_{i+\frac{1}{2}} F_{i+\frac{1}{2}} - q_{i-\frac{1}{2}} F_{i-\frac{1}{2}}) \qquad 14.1.1$$

where T is a prognostic tracer, $q_{i-\frac{1}{2}} = u_i \Delta t / \Delta x$ and $q_{i+\frac{1}{2}} = u_{i+1} \Delta t / \Delta x$ are the Courant numbers on the left and right cell faces respectively. The values $F_{i-1/2}$ and $F_{i+1/2}$ are the tracer concentrations at the left and right cell faces where;

$$F_{i-\frac{1}{2}} = 0.5(T_{i+1} + T_{i-1}) - 0.5q_{i-\frac{1}{2}}(T_i - T_{i-1}) - \left[\tfrac{1}{6} - \tfrac{1}{6}q^2_{i-\frac{1}{2}}\right]CURV \qquad 14.1.2$$

with:

$$\begin{aligned} CURV &= T_i - 2T_{i-1} + T_{i-2} \quad for\ \ q_{i-\frac{1}{2}} > 0 \\ CURV &= T_{i+1} - 2T_i + T_{i-1} \quad for\ \ q_{i-\frac{1}{2}} < 0 \end{aligned} \qquad 14.1.3$$

The finite difference representation of the QUICKEST algorithm in sparse coordinates is presented in one dimension (x dimension) for brevity. This is easily expanded to full three dimensions. Using the cells to process vectors described in Section 11.13 (i.e. w3_t for tracers, w3_e1 for u1 velocity and using the vector fragmentation illustrated in Figure 11.13), and assuming the lateral boundary conditions have been set, first the Courant numbers are defined over the whole grid.

```
In c:
for(cc=1; cc<=n3_t; cc++) {
   c=w3_t[cc];
   qx[c]=u1[c]*Δt/Δx[c];
   Fx[c]=0.0;
}
```

where u1 is the velocity in the x direction, Fx[c] is the flux through the face corresponding to cell c (remembering that the u1 location in the Arakawa C grid is on the face to the west of the cell center) and and Dx, Dy and Dz are the grid spacings in the x, y and z directions respectively. Next the tracer concentrations at the cell face is computed.

```
for(cc=1; cc<=v3_e1; cc++) {
   c=w3_e1[cc];
   xp1=xp1[c];
   xm1=xm1[c];
   xm2=xm1[xm1];

   face = 0.5*(T[c]+T[xm1]);
   grad = T[c]-T[xm1];
   curv = 0.0;
   if(qx[c] > 0.0) {
      curv = T[c] – 2*T[xm1] + T[xm2];
   else if(qx[c]  <  0.0)
      curv = T[xp1] – 2*T[c] + T[xm1];

   Fx[c] = face + 0.5*qx[c]*grad – (1 – qx[c]*qx[c])*curv/6.0;

   /* This is then multiplied by the mass flux through the face to provide the tracer flux */
   Fx[c] = u1[c]*Δy[c]*Δz[c];

}
```

Note that the u1 velocity cells to process vector, w3_e1, is used to get the fluxes over wet cells only. This improves efficiency by only calculating non-zero fluxes at those cells that will enter into the finite difference computations. The fluxes at ghost cells (normal velocities at solid boundaries)

are set to zero due to the zero flux lateral boundary condition. Next the tracer concentration is updated.

```
for(cc=1; cc<=v3_t; cc++) {
   c=w3_t[cc];
   xp1=xp1[c];
   T[c] -= (Fx[xp1]-Fx[c])*Δt/(Δx*Δy*Δz);
}
```

This finite difference representation only performs computations where non-zeroed values of variables exist with no conditional statements required to examine the land/wet status of a cell, making for a very efficient algorithm. In F9X this algorithm becomes even more compact, e.g.

```
! Initialize
Fx=0.0
qx(w3_t)=u1(w3_t) ]*Δt/Δx[w3_t]

! Get the fluxes
xm2=xm1[xm1]
face = 0.5*(T(w3_e1)+T(xm1(w3_e1)))
grad = T(w3_e1)-T(w3_e1(xm1))
curv = 0.0
where(qx(w3_e1) > 0.0)
   curv = T(w3_e1) – 2*T(xm1(w3_e1)) + T(xm2(w3_e1))
elsewhere(qx(w3_e1)  <  0.0)
    curv = T(xp1(w3_e1)) – 2*T(w3_e1) + T(xm1(w3_e1))
endwhere

 Fx[w3_e1] = face + 0.5*qx(w3_e1)*grad – (1 – qx(w3_e1)*qx(w3_e1))*curv/6.0

 ! This is then multiplied by the mass flux through the face to provide the tracer flux */
 Fx(w3_e1) = u1(w3_e1)*Δy(w3_e1)*Δz(w3_e1)

! Update the tracer
T(w3_t) -= (Fx(xp1(w3_t))-Fx(w3_t))*Δt/(Δx*Δy*Δz)
```

14.3 Vertical Integration.

The order in which cells are processed in the solutions to the finite difference representations of the hydrodynamic equations is generally not important. The exception to this is the solution to the implicit vertical diffusion. These algorithms require sequential access to cells from the surface to the bottom (and *vice versa*). Other processes, e.g. calculation of vertical velocity or vertical integral of pressure, may also benefit from a computational efficiency perspective from processing in a vertical sequential order. The vertical velocity is retrieved from the continuity equation.

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \qquad 14.2.1$$

where (u,v,w) are velocities in the (x,y,z) directions respectively. This is typically discretized in flux form:

$$w_{z+1}=\left(w_z\Delta x\Delta y-\left[(u\Delta y\Delta z)_{x+1}-(u\Delta y\Delta z)_x+(v\Delta x\Delta z)_{y+1}-(v\Delta x\Delta z)_y\right]\right)/\Delta x\Delta y \qquad 14.2.2$$

where (Δx,Δy,Δz) are the grid spacings in the (x,y,z) directions respectively. The subscript x refers to variables evaluated at cell x and x+1 refers to variables evaluated at the cell xp1[]. Equation 14.2.2 is solved in the sparse system thus:

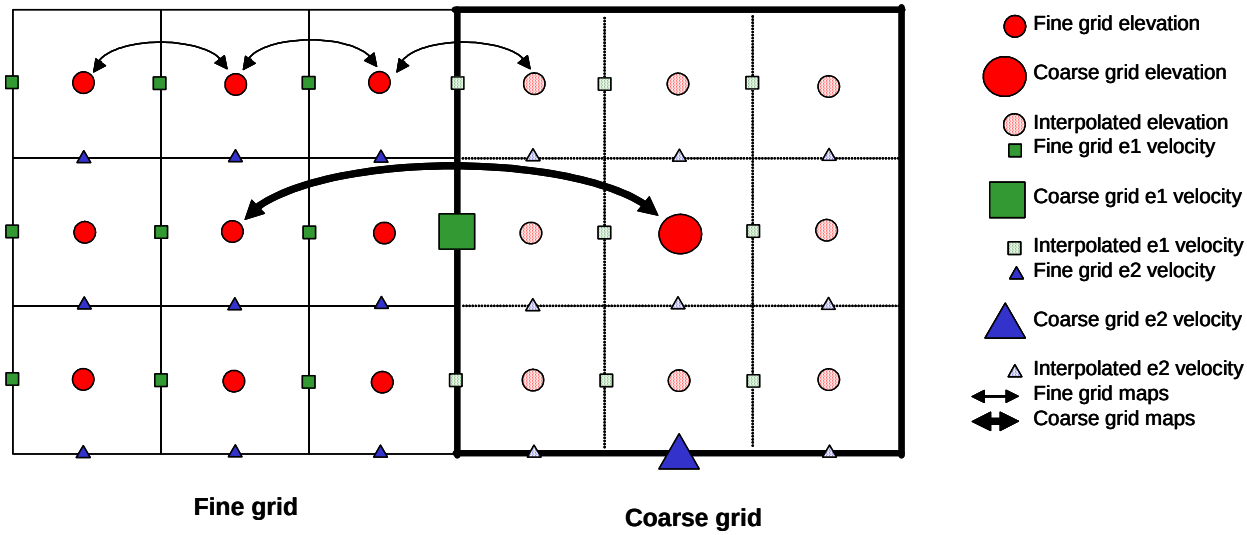
```
for(cc=1; cc <= v2_e1; cc++) {
   c = bot_e1[cc];
   fbot = 0.0;
   w[c] = 0.0;
   while(c != w2_e1[cc] {
      xp1 = xp1[c];
      yp1 = yp1[c];
      zp1 = zp1[c];

      ftop = fbot + u1[c]*Δy*Δz - u1[xp1]*Δy*Δz + u2[c]*Δx*Δz - u1[yp1]*Δx*Δz;
      w[zp1] = ftop/(Δx *Δy);
      fbot = ftop;
      c = zp1;
   }
}
```

here w2_e1 is the surface cells to process vector for u1 velocity (Section 11.13) and bot_e1 holds the bottom sparse coordinate for u1 velocity (Section 11.12). The loop is carried out vertically from the bottom to the surface. Generally surface and bottom boundary conditions for velocity need also be applied.

15. Grid Refinement

Grid refinement, or two-way nesting, allows a fine resolution grid (FRG) to be embedded within a coarse resolution grid (CRG) so that increased resolution is achievable in a sub-region of the whole domain. Although the time-step for the simulation is determined by the smallest grid, savings in computer time is generally achieved by not highly resolving the whole domain. This practice has been common among atmospheric models for some time (e.g. Zhang et al, 1986, Clarke and Farley, 1984), and has also been applied to several ocean models (Fox and Maskell, 1995, Kowalik and Murty, 1993, p 147, 201). Grid refinement has been implemented in shoc using the methodology outlined in Kowalik and Murty (1993) p149 – 154. This relies on constructing a grid where the number of fine grid cells that comprise one coarse grid cell (the zoom factor, zf) is an odd number so that cell faces and centers in the coarse grid are coincident with fine grid locations at the coarse-fine boundary. As with the distributed processing method, an overlap zone between coarse and fine grids (auxiliary cells) must be present so that information can be accessed for finite difference derivatives and averages. The grid arrangement to facilitate two-way nesting is illustrated in Figure 15.1:

Figure 15.1 : Grid structure for grid refinement



The grid refinement mapping process is made easier by use of the sparse maps. Separate windows are established for the FRG and CRG. In the CRG window the maps are reset so that zf successive mappings occur (e.g. every 3 cells in the example of Figure 15.1). Over coarse-fine grid boundaries, the CRG will map directly onto a cell in the fine grid, provided the zoom factor is an odd number (e.g. the coarse grid map arrow in Figure 15.1). The FRG maps remain unaltered. At coarse-fine grid boundaries the FRG maps will generally map into cells that are not included in the CRG, and hence are not updated via the equations of motion every time-step (e.g. the speckled shapes in Figure 15.1.). These cells are prescribed values using bi-linear interpolation between the closest calculated fine and coarse grid cells. Note that the closest fine and coarse grid calculated values used in the interpolation depend on whether a cell or face centered variable is being interpolated. Figure 15.2 illustrates the velocities at cell faces and cell center locations used in the fine and coarse grids in the bi-linear interpolation. This interpolation takes place on the master and the interpolated values are transferred to auxiliary cells on the fine grid window during the master to slave transfers.

Although the CRG face centered velocities are shifted the appropriate distance from the cell center (i.e. (int)zf / 2 cells), they are actually stored in memory in the CRG window at the same cell location as the cell center. This simplifies mapping between cells in the coarse grid window,

but means that during master to salve (and slave to master) transfers, values of velocity must be taken from the correct wet cell location in the fine/coarse grid and placed in the corresponding correct auxiliary location in the coarse/fine grid. This is illustrated in Figure 15.3 (a) for fine to coarse transfers and 15.3 (b) for coarse to fine transfers.

Figure 15.2 : Cell locations used for interpolation. The shapes and colours correspond to those in the legend of Figure 15.1.

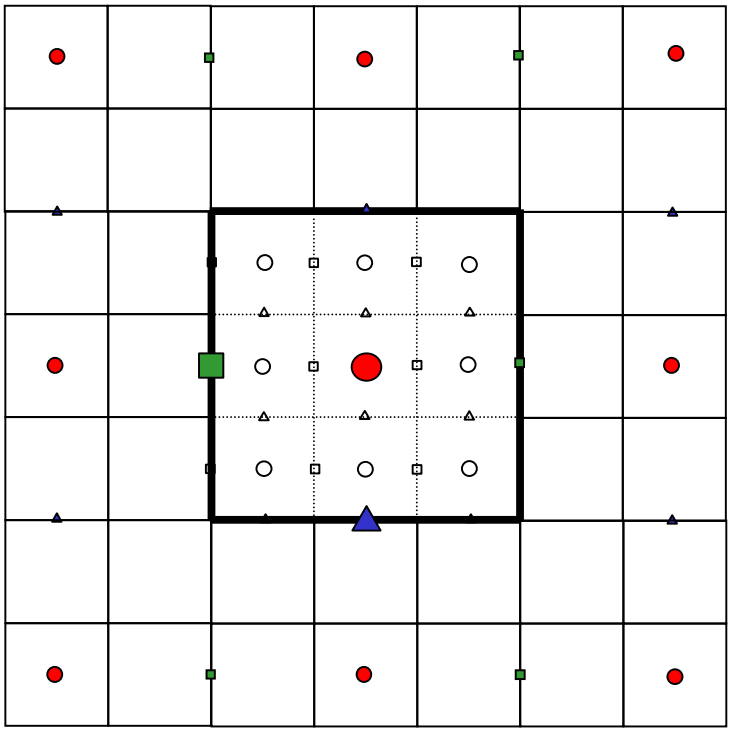
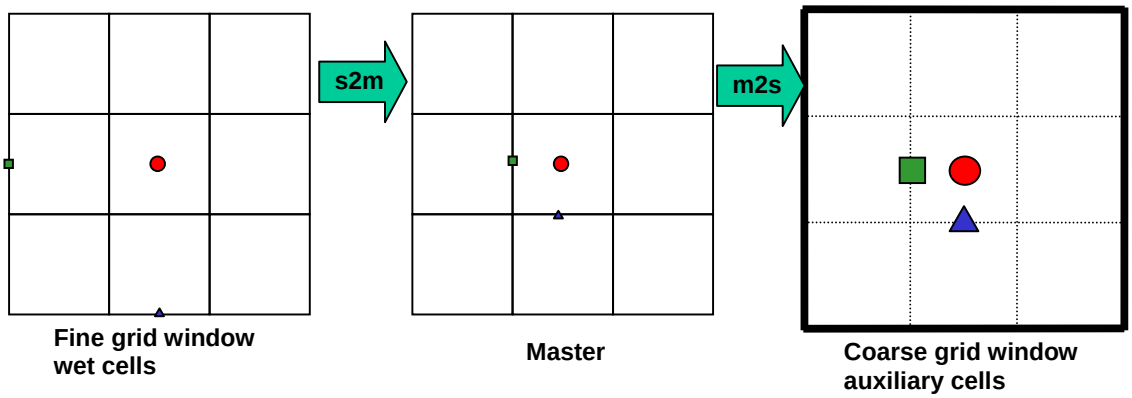
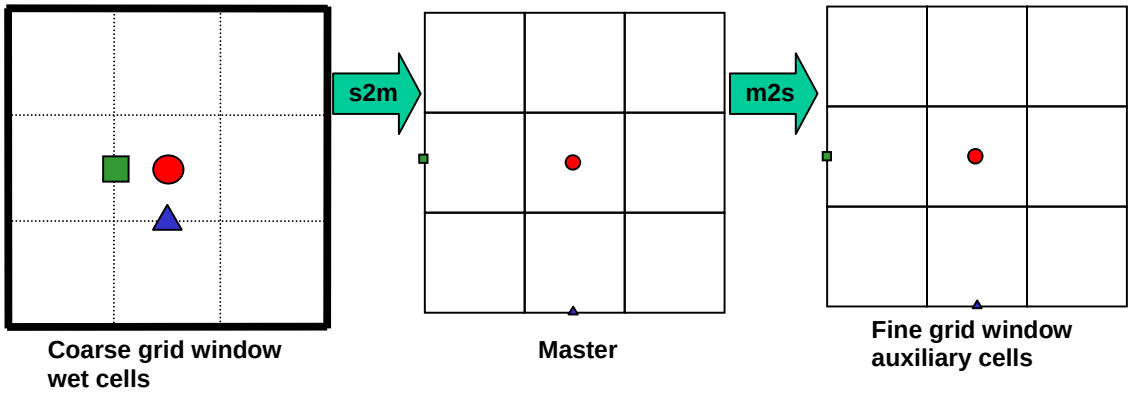


Figure 15.3 : Velocity transfers between coarse / fine grids
(a) Transfers from fine grid wet cells to coarse grid auxiliary cells



(b) Transfers from coarse grid wet cells to fine grid auxiliary cells



Finally the grid spacing, horizontal diffusion and pre-calculated constants must be adjusted in the coarse grid to account for the larger grid size. Once this is accomplished, the coarse grid mappings have been re-set and the mater/slave transfer vectors adjusted at the preprocessing stage, the code can largely proceed with the primitive equation computations with no additional overhead. However, in order for the code to remain conservative, care must be exercised when using fluxes at the coarse-fine grid boundary. When a coarse grid cell uses fluxes from auxiliary cells in the fine grid, the fluxes must be summed over the relevant fine grid faces (Figure 15.4). When a fine grid cell uses fluxes at interpolated coarse grid auxiliary locations the fluxes must be consistent in the sense that the vertical integral of the 3D velocity must equal the 2D velocity (i.e. 3D velocities must be adjusted as is done in the main body of the code; velocity adjustment Figure 3.1). Also surface cell thicknesses must not be interpolated at these cells but rather computed using the vertical layer heights and the height of the free surface. In order to suppress short waves in the fine grid unresolved in the coarse grid, which may lead to flow distortion at the fine-coarse boundary, a horizontal diffusion sponge zone is implemented over the fine-coarse boundary and fine grid velocities are smoothed with the 9 point Shuman spatial filter (e.g. Kowalik and Murty (1993), eqn 3.135) before transfer to the coarse grid auxiliary cells.

Figure 15.4 (a) : Fluxes used to calculate cell centered variables (η, T, S) in the coarse grid at coarse-fine grid boundaries. Large arrows are coarse grid fluxes and small arrows are fine grid fluxes. The fine grid fluxes in e1 and e2 directions are summed.

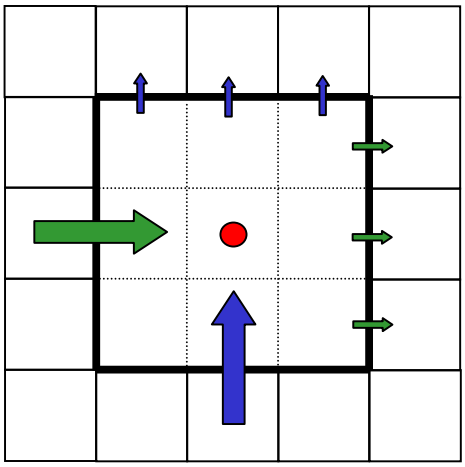
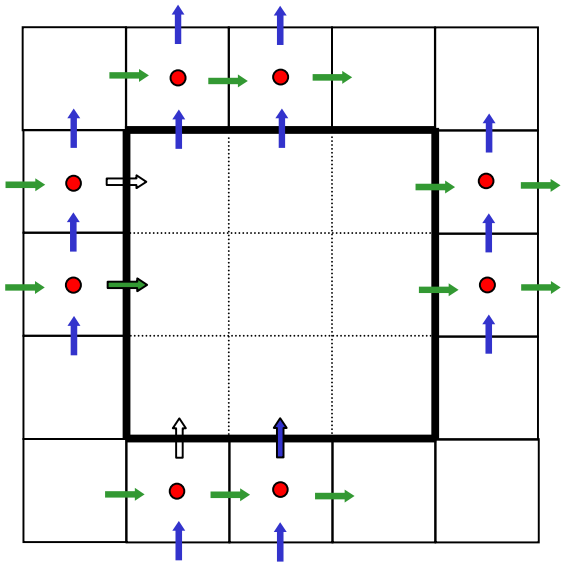


Figure 15.4 (b) : Fluxes used to calculate cell centered variables (η, T, S) in the fine grid at coarse-fine grid boundaries. Solid arrows are fluxes derived from computed velocities in the fine or coarse grids and speckled arrows are fluxes derived from interpolated velocities in the coarse grids. Note a *scaled* computed flux is used in the coarse grid where fine and coarse grid velocities are coincident (solid-lined solid arrows). Also, on front and right coarse-fine boundary edges the fine grid does not use any values from the coarse grid.



The cells involved in computing the e1 velocity at the e2 face and e2 velocity at the e1 face are illustrated in Figure 15.5. A flow diagram of how grid-refinement proceeds is presented in Figure 15.5.

Figure 15.5 : Cells involved in velocity averaging (e.g. for Coriolis terms) in the fine grid. Note the right and back coarse-fine boundary edges the fine grid does not use any e1 velocity values from the coarse grid, and the front and left coarse-fine boundary edges the fine grid does not use any e2 velocity values from the coarse grid.

(a) u1@u2 (b) u2@u1

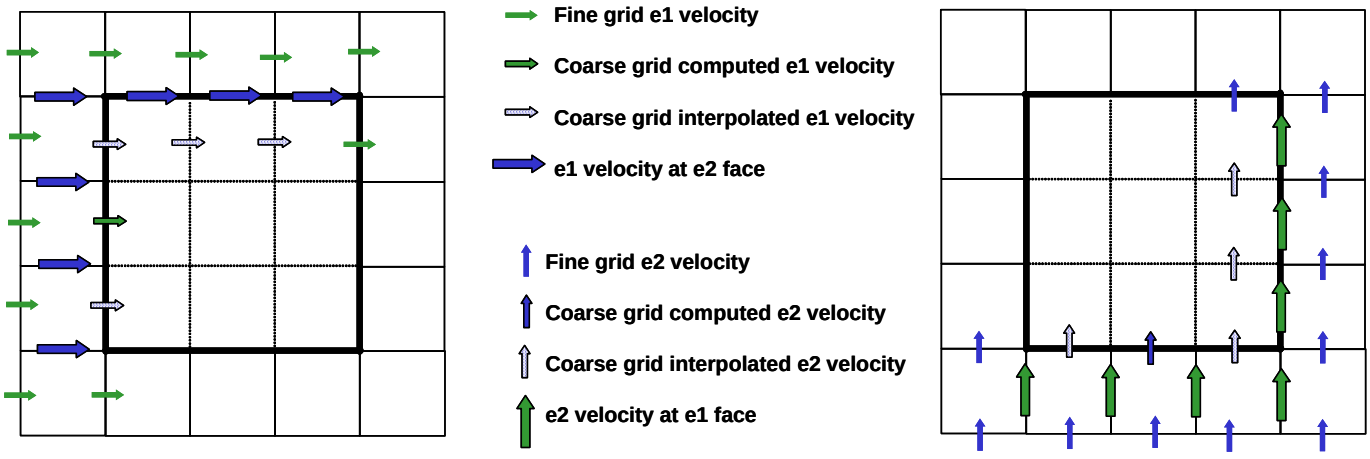
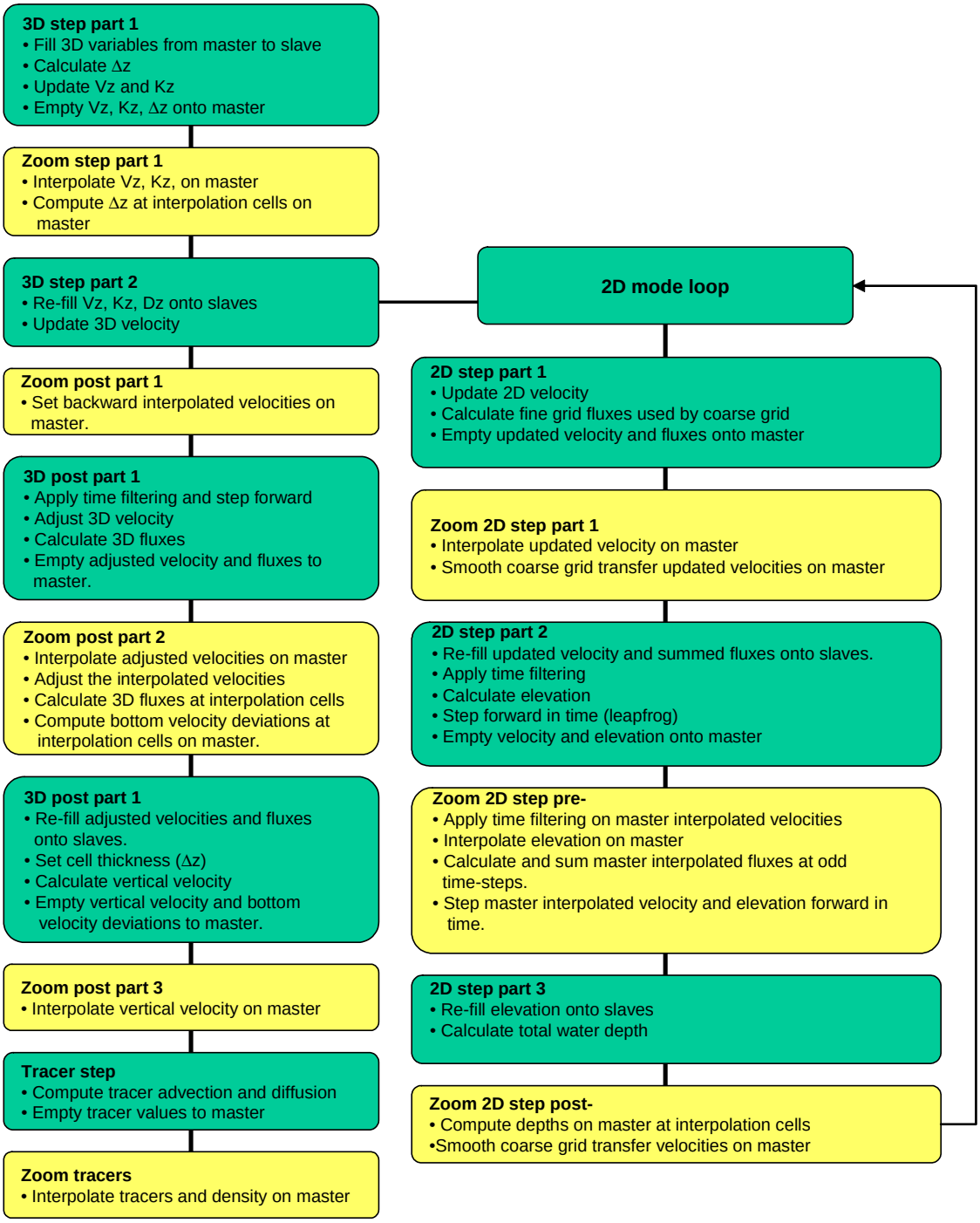
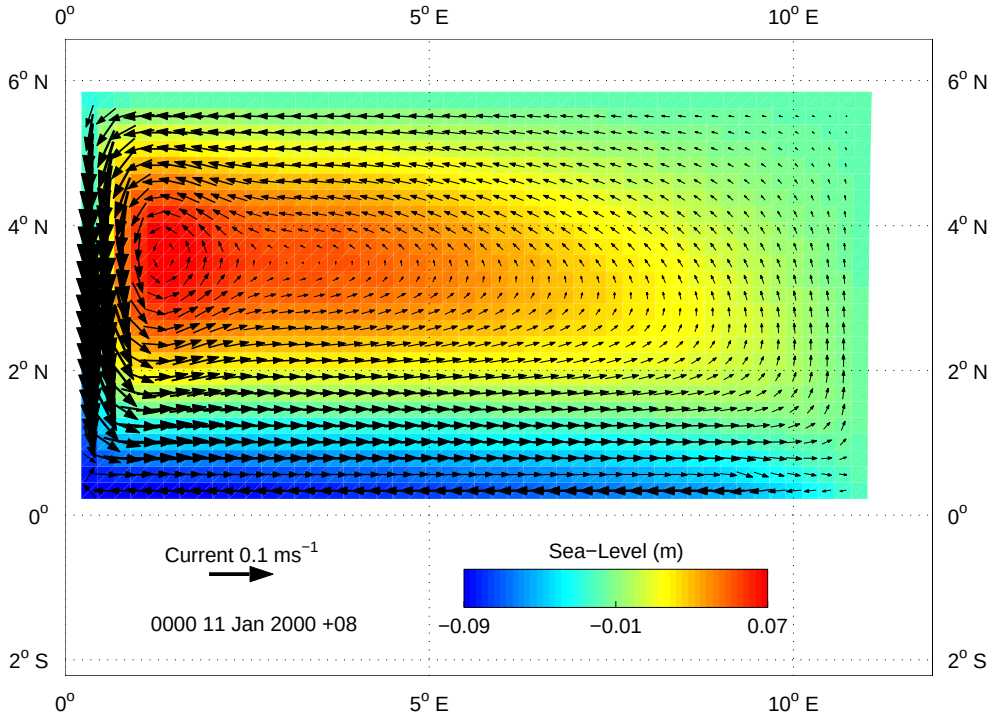


Figure 15.6 : Flow chart of grid refinement processes



A domain consisting of a closed basin with sloping bottom was constructed and forced with a constant windstress of 0.1 Nm^{-2}. The surface elevation and depth averaged velocity is displayed in Figure 15.6.

Figure 15.6 : Closed basin solutions with no grid refinement



The center of the domain was then coarsened using a zoom factor of 3, resulting in the grid illustrated in Figure 15.7 (a). Solutions using this grid refinement configuration are presented in Figure 15.7 (b). The zoom factor was further increased to 5 (Figure 15.8 (a)) with corresponding solutions presented in Figure 15.8 (b). The conservation characteristics were analysed by computing the total heat in the domain for the 10 day simulation. A time series of total heat using no grid refinement and zoom factors of 3 and 5 is displayed in Figure 15.9 (a) – (c). The series represents the change in heat content expressed as a percentage of the initial total heat content. It is observed that conservation characteristics are good with grid refinement invoked, with the change in heat content less than 0.008% after 10 days. The change in heat content had not reached a steady state however, as is the case without grid refinement. The change in total volume after 10 days with a zoom factor of 5 had stabilized at less than 0.0015%.

A similar domain was set up consisting of an open channel with a sloping bottom. The open boundaries existed at the western and eastern ends of the channel, and all variables were approximated using cyclic OBC's. A constant alongshore wind of 0.1 Nm^{-2} was applied. This test case is identical to the test in Section 9.4 of the Users Manual. The results with no grid refinement and refinement using a zoom factor of 3 are displayed in Figure 15.9.

Figure 15.7(a) : Closed basin grid with zoom factor = 3

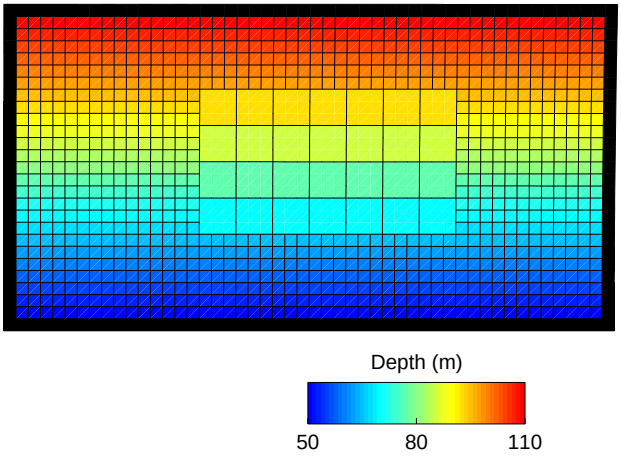


Figure 15.7 (b) : Closed basin solutions with grid refinement; zoom factor = 3

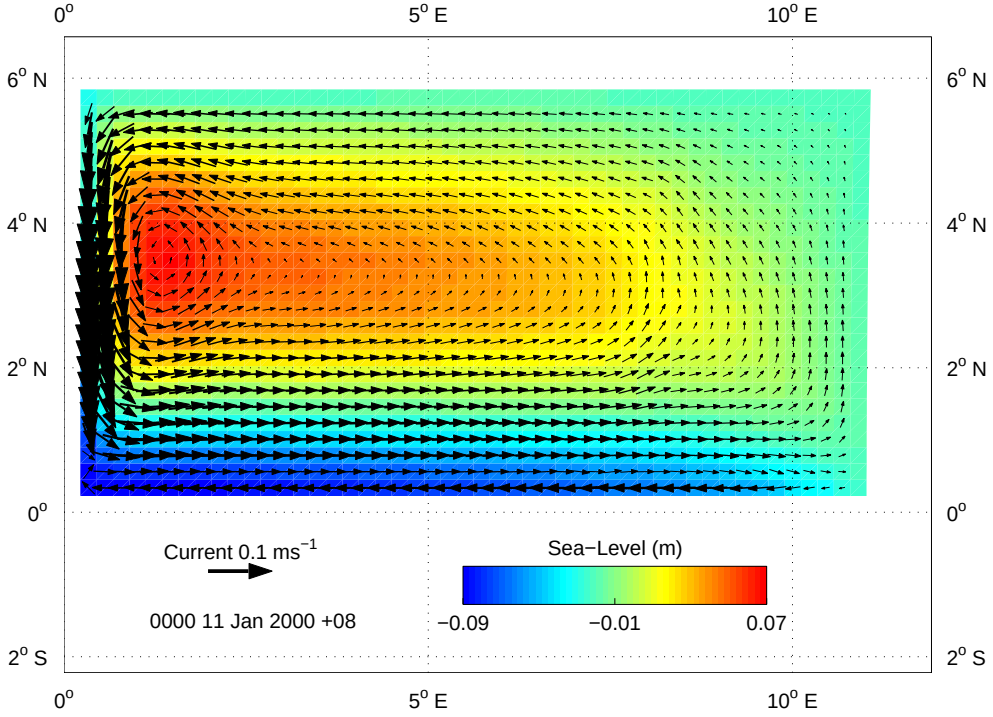
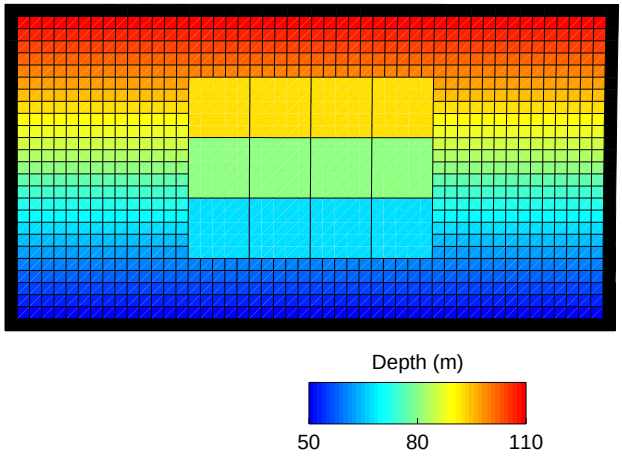


Figure 15.8(a) : Closed basin grid with zoom factor = 5



15.8 (b) : Closed basin solutions with grid refinement; zoom factor = 5

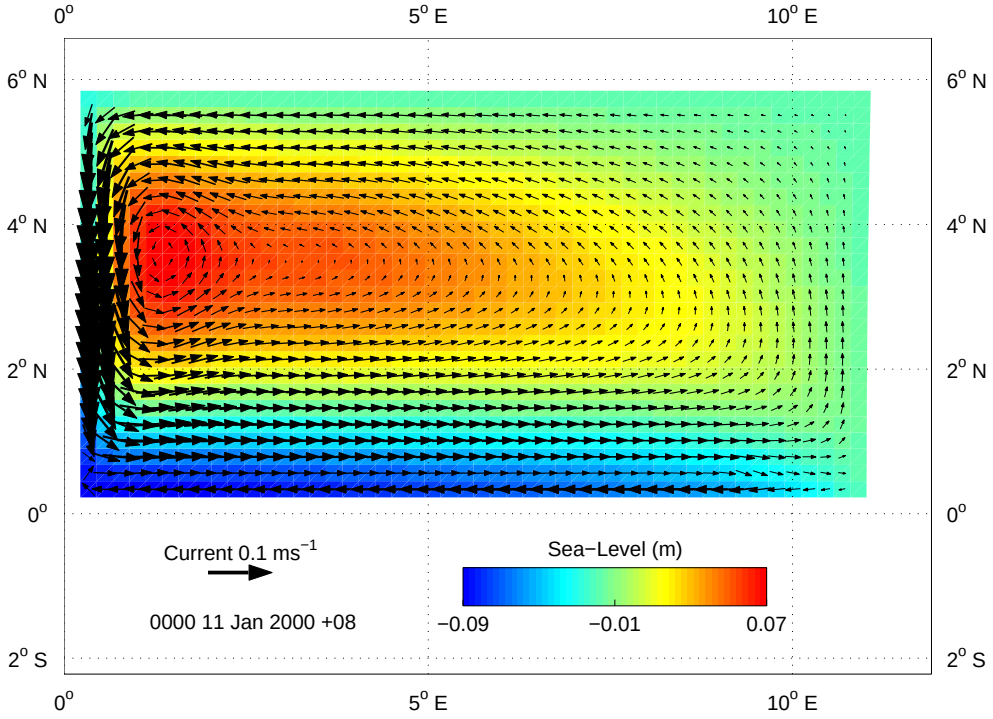


Figure 15.9 (a) : Total heat in the closed basin domain. Units are the change in total heat content expressed as %.

(a) No grid refinement

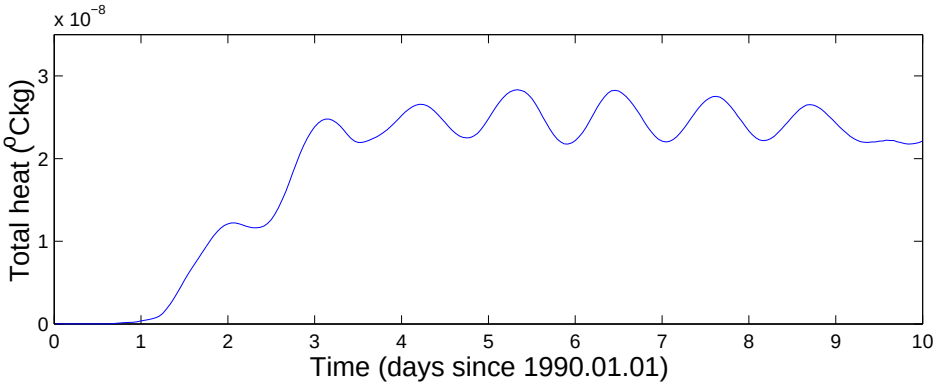


Figure 15.9 (b) : Grid refinement zoom factor = 3

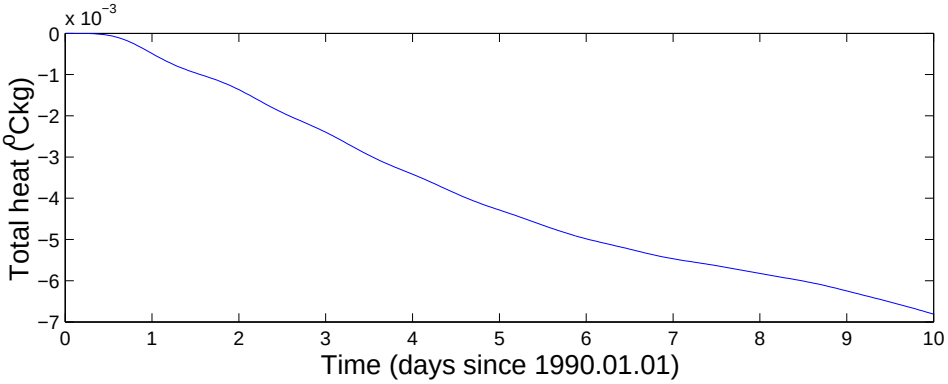


Figure 15.9 (b) : Grid refinement zoom factor = 5

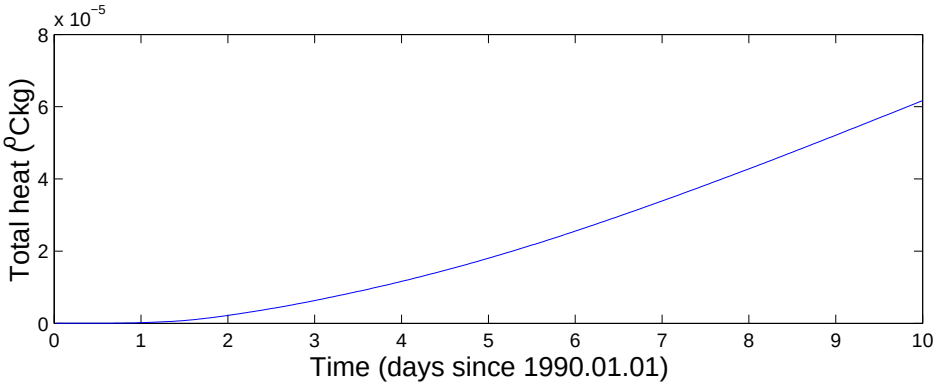


Figure 15.9 (a) : Open channel domain with no grid refinement.

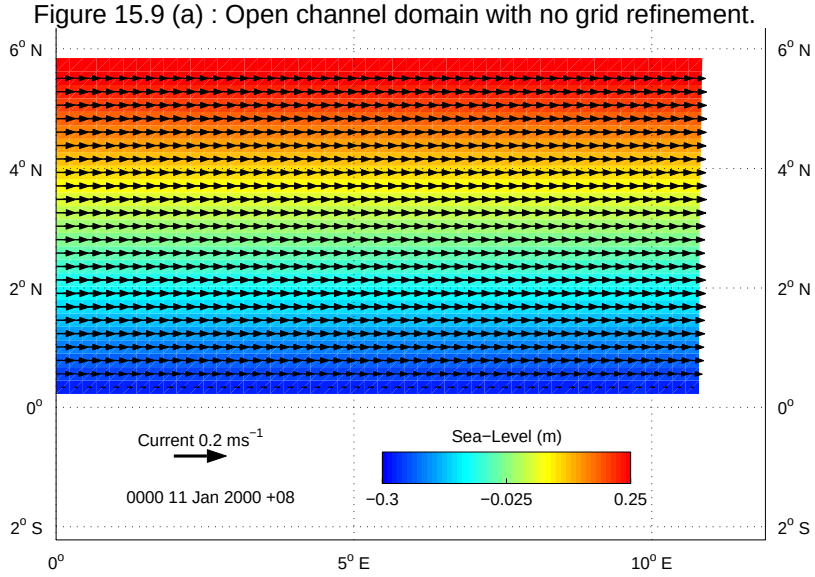
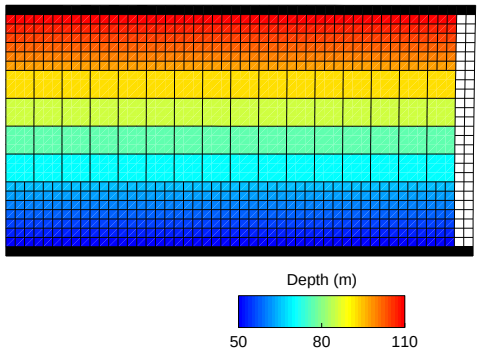
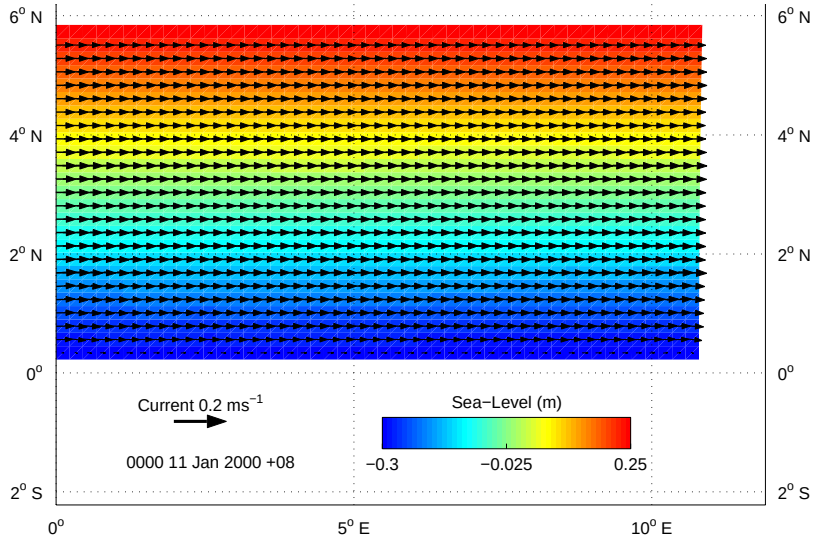


Figure 15.9(b) : Open basin grid with zoom factor = 3



15.9 (c) : Open basin solutions with grid refinement; zoom factor = 3



16. Applications.

SHOC has been applied to a variety of aquatic environments ranging from high resolution riverine/estuarine environments to large scale regional coastal domains. A selection of output demonstrating SHOC's capabilities are included below.

16.1 Upper Derwent Estuary

The Derwent Estuary bisects the city of Hobart, capital to the state of Tasmania in southern Australia. This estuary is micro-tidal and extends approximately 60km from the seward end to the head of the estuary. The object of applying SHOC to the upper regions of the estuary was to assess the impact of discharge from a pulp mill into the estuary. SHOC generated the circulation regime on a grid with a resolution from 10m in the cross-river direction to 80m in the along-river direction. There existed 20 vertical levels in the 'z' system. A series of shallow wetlands exists adjacent to the down-river open boundary which were subject to wetting and drying. Forcing of the model included tidal and baroclinic forcing at the seaward open boundary, river discharge at the landward end and time dependent winds. The pulp mill discharge was treated as a point source.

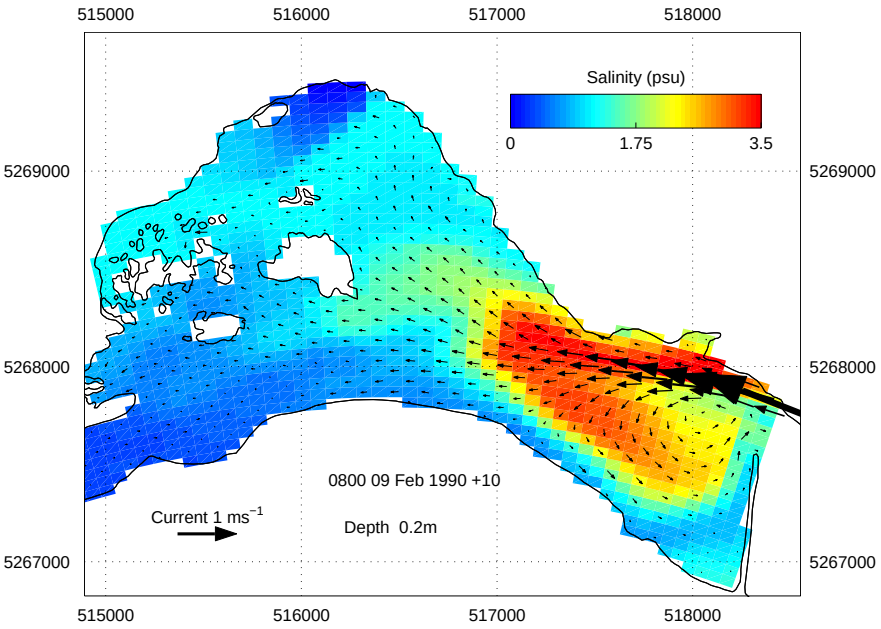


Figure 16.1.1 : Salinity and velocity at the down-river boundary. The tide is at the peak flood phase and generates large currents through the narrow opening at the down-river boundary. The large flows induce topographic upwelling that leads to higher salinity bottom water breaking the surface, resulting in the observed large plume of salty water. The inflow also generates an anti-clockwise eddy in which the high salinity plume is advected, gradually relocating to the southern side of the estuary as the flood tide weakens.

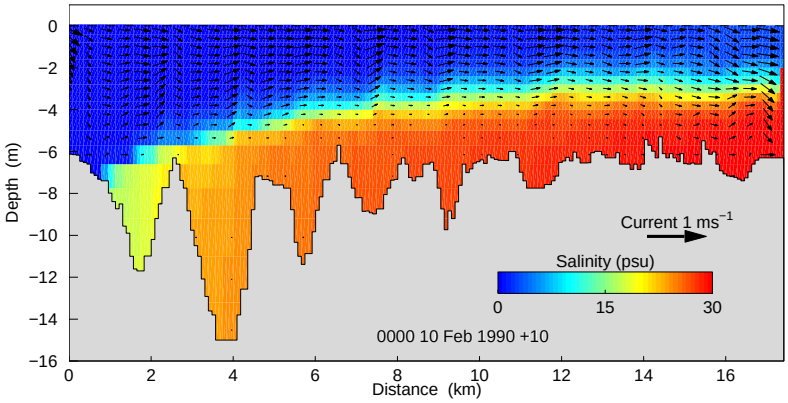


Figure 16.1.2 : Cross section of salinity along the deepest part of the river channel. The salt wedge has attained its equilibrium position, determined by the surface and bottom density difference at the down-river boundary, the strength of river discharge, the water and pycnocline depth at the boundary and the strength of the tidal mixing. Internal waves are visible at the fresh / salt interface with corresponding velocity perturbations.

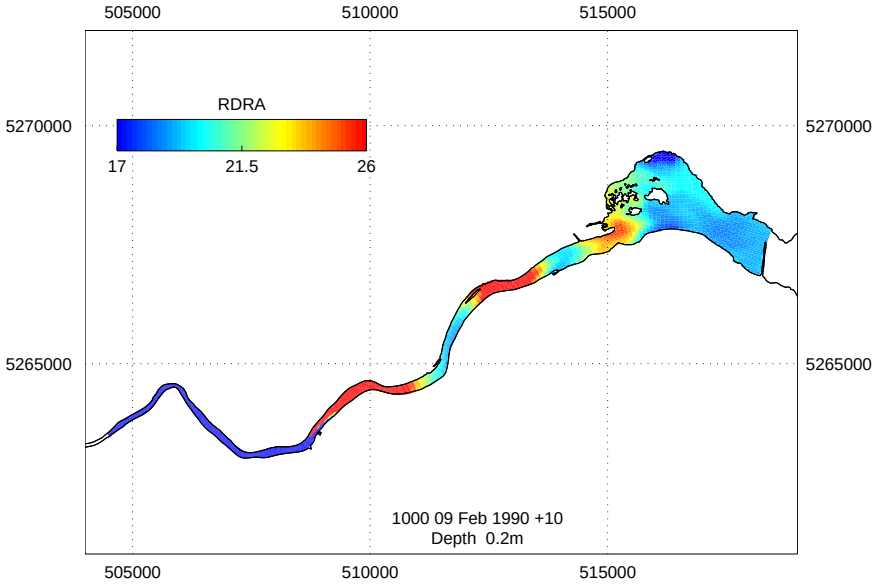


Figure 16.1.3 : Concentration of Dissolved Refractory Resin Acid (RDRA) from the pulp mill. The entire estuary is depicted and the location of the pulp mill is easily identified. RDRA moves down-river in distinct pulses, dispersing throughout the wetland area at the down-river boundary. This pulsing effect is the result of pooling of RDRA when the tide opposes the river flow (low velocities in the vicinity of the outfall lead to higher concentrations) then advection downstream when the tide ebbs and tidal and river flows combine to create larger velocities.

16.2 North West Shelf

The continental shelf off north-western Australia (NWS) was modelled in order to derive the circulation for use in an ecological model. The shelf is wide over this region and is subject to large semidiurnal tides which induce flow in a predominantly cross-shelf direction. Large internal tides are also present. The NWS circulation may be influenced by tropical cyclones, the occurrence of which has the highest frequency along Australia's coastline (3 - 4 / year). In winter the Leeuwin Current flows in a poleward direction and is opposed in summer by south-westerly wind driven circulation. The three open boundaries were forced with surface elevation derived from global circulation and tide models and temperature / salinity distributions from a global circulation model. The model is forced with wind data taken from the NCEP-NCAR 40-year Reanalysis data set (Kalnay et al. 1996). Resolution in the horizontal is 5km with 34 'z' layers having 3m resolution at the surface. Vertical mixing was parameterised with the k-ε scheme.

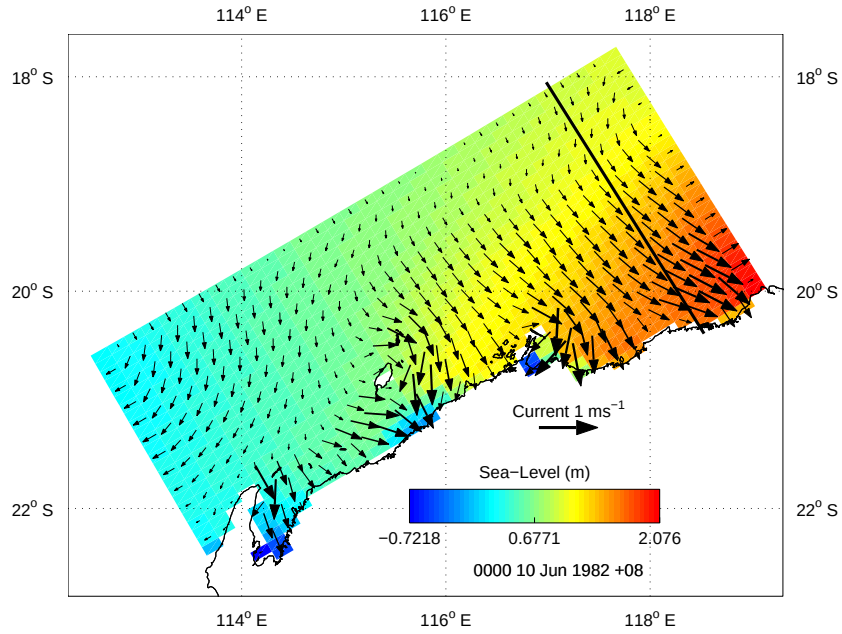


Figure 16.2.1 : Surface elevation and currents on the NWS. Note the large gradients of surface elevation leading to cross-shelf flow.

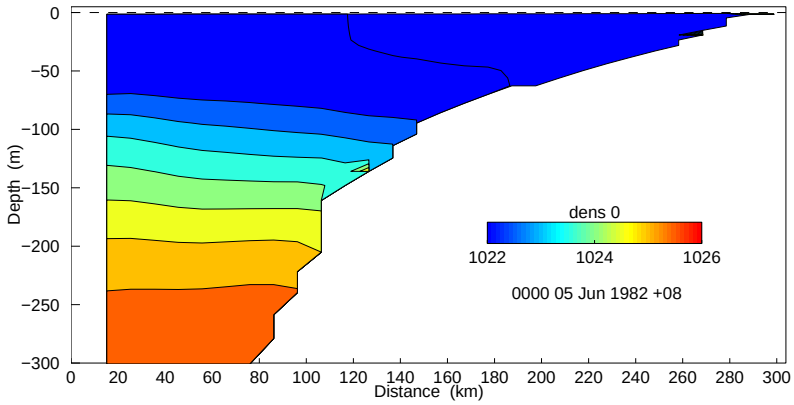


Figure 16.2.2 : Density section (cross-shelf along the transect shown in Figure 9.4). Isopycnals slope downward towards the coast to support the winter Leeuwin Current.

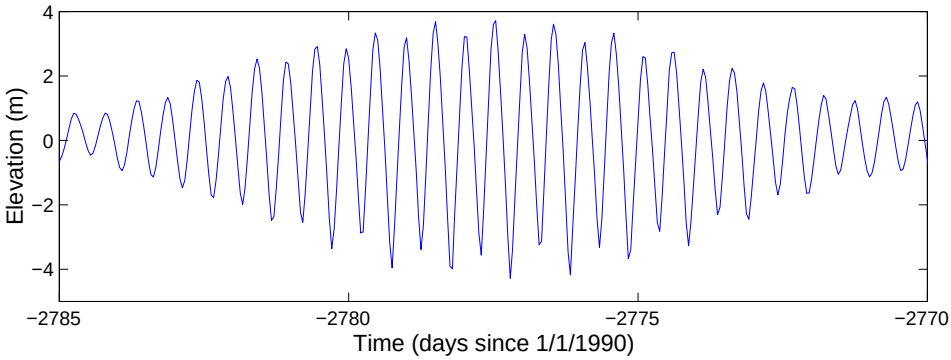


Figure 16.2.3 : Surface elevation at Port Hedland. Maximum tidal range is almost 8m.

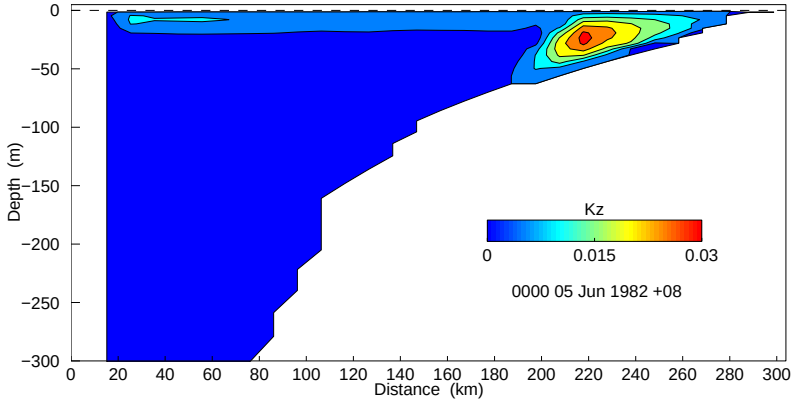


Figure 16.2.4 : Vertical diffusion coefficient along a cross-shelf section. The large tide generates vigorous mixing. Inshore there exists overlap of the surface and bottom boundary layers resulting in a region of strong vertical mixing throughout the whole water column. This is characterised by very large diffusivities which attain maximum values at mid-depth. The critical value for the formation of u^3 fronts is placed at a depth of ~70m (using spring tidal velocities of 0.6 ms^{-1}) which is the approximate limit of large diffusivities above. In reality these fronts never form owing to the very high stably stratifying influence of incident solar radiation overcoming the vertical mixing (Tranter and Leech, 1987).

16.3 D'entrecasteaux Channel

The D'entrecasteaux Channel lies between the southern Tasmanian mainland and Bruny Island in southern Australia. The Huon Estuary joins the D'entrecasteaux Channel near the southern limit of the channel and is a significant source of fresh water. The estuary/channel is highly stratified at times, and also characterised by complex geography, making modelling of the region challenging. Long period simulations were required (>1 year) to assess the impact of aquaculture on the aquatic environment, and these simulations required acceptable run time rations (>100:1). The model was forced with river flow from various sources (the largest being the head of the Huon Estuary) wind stress and surface elevations, temperature and salinity on the northern and southern limits of the channel. These northern and southern boundary conditions were derived from a larger scale model of the region. The period simulated (Feb 1996) included a massive flood event (maximum flow of 1940 m^3s^{-1}) which had significant impact on the salinity distribution throughout the channel. A segment of this flow event is displayed as a time series in Figure 16.3.2 (a) – (j), exhibiting a freshwater plume propagating up the channel to collide with a solid boundary and subsequently bifurcate.

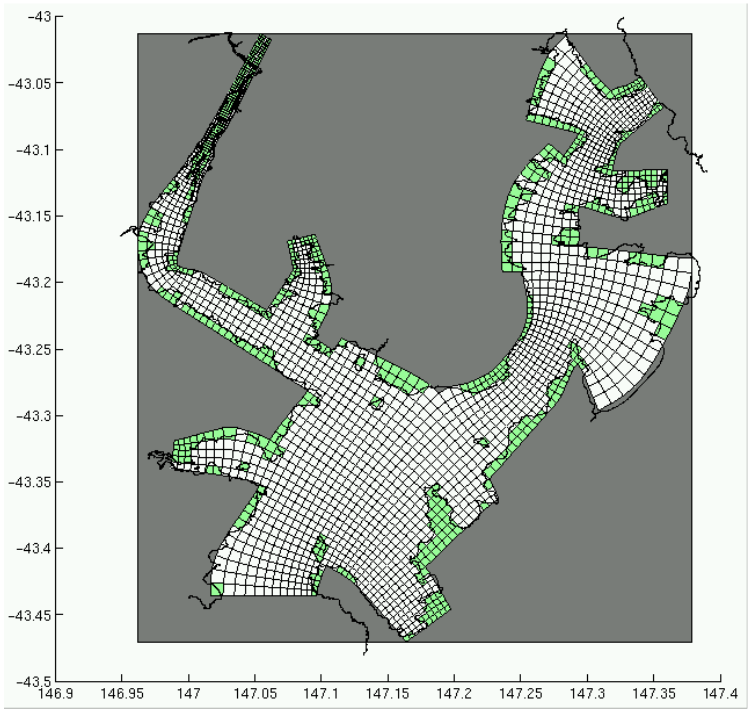


Figure 16.3.1 : The curvilinear grid used for the simulations exhibiting a sufficient number of cells so as not compromise resolution whilst maintaining acceptable time-steps and hence run-time ratios.

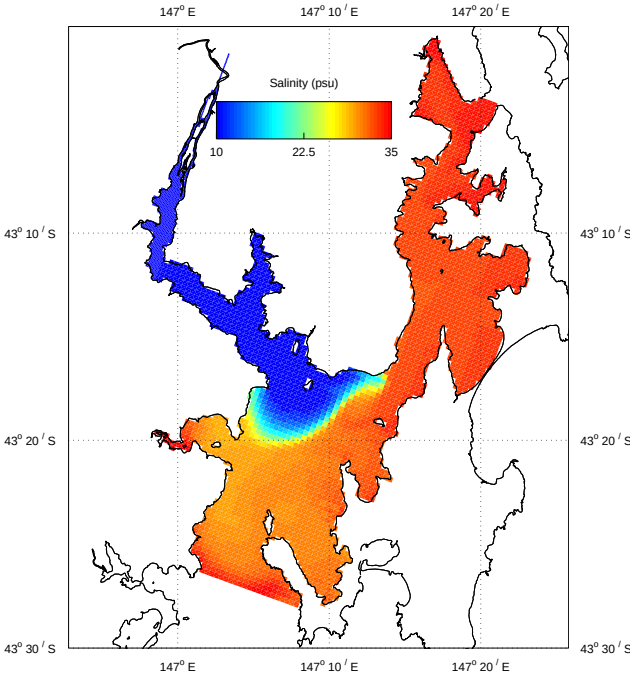


Figure 16.3.2 (a)

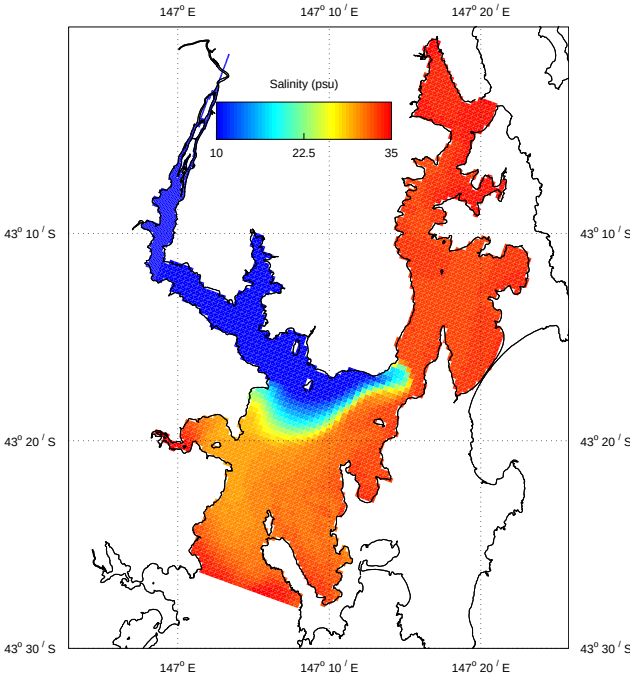


Figure 16.3.2 (b)

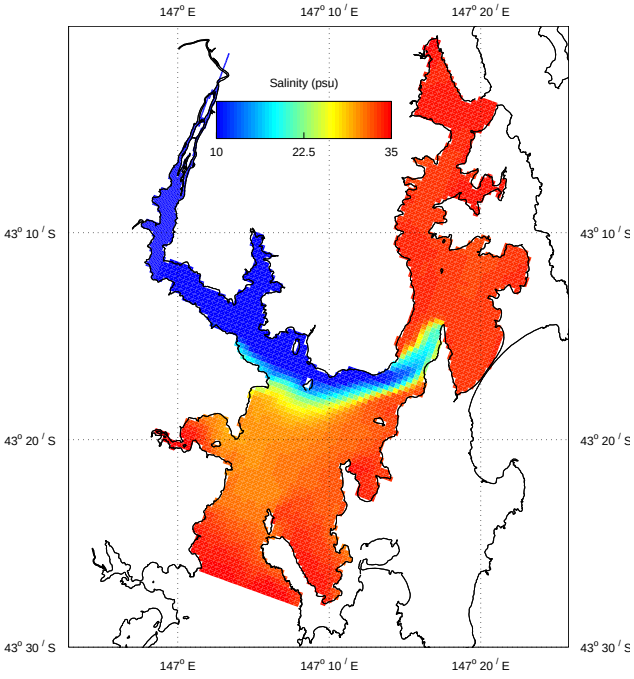


Figure 16.3.2 (c)

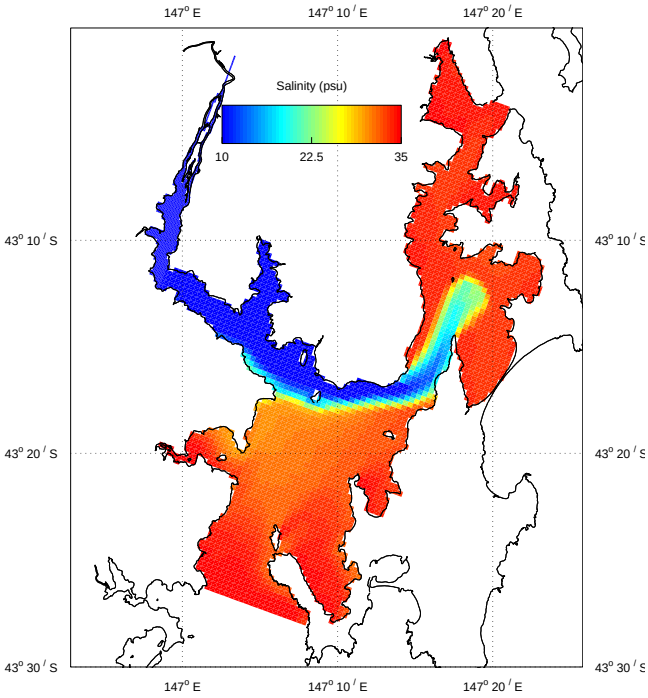


Figure 16.3.2 (d)

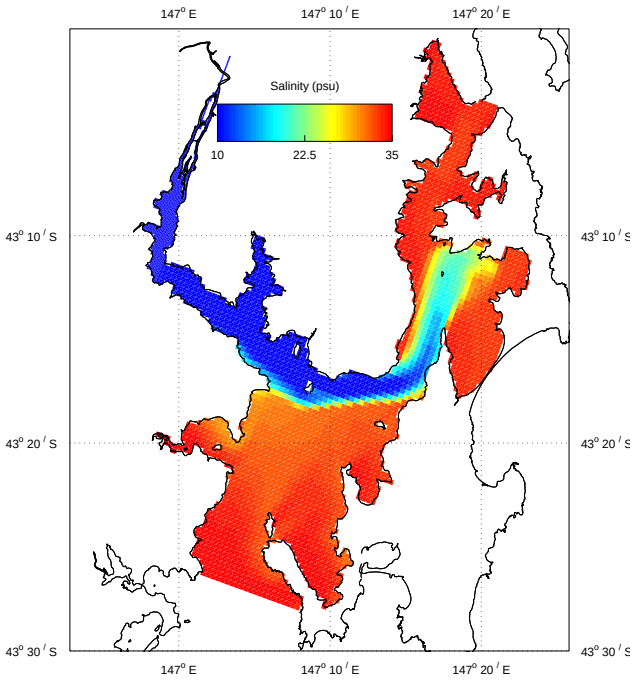


Figure 16.3.2 (e)

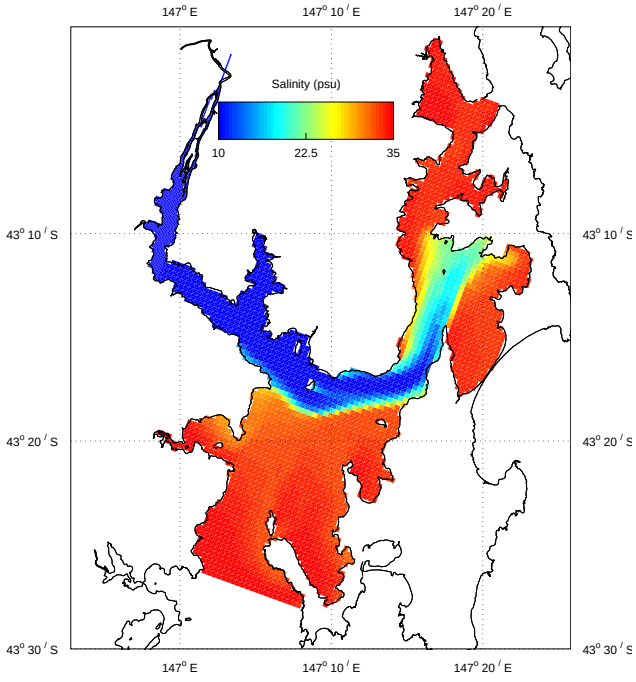


Figure 16.3.2 (f)

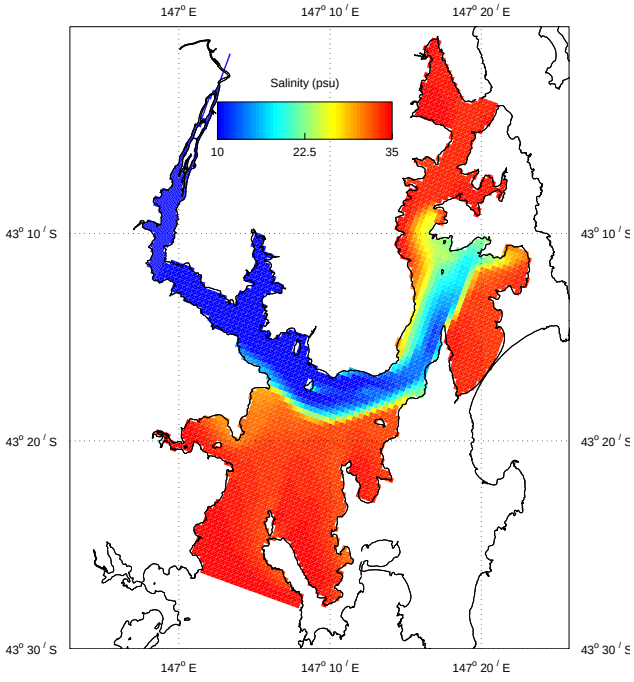


Figure 16.3.2 (g)

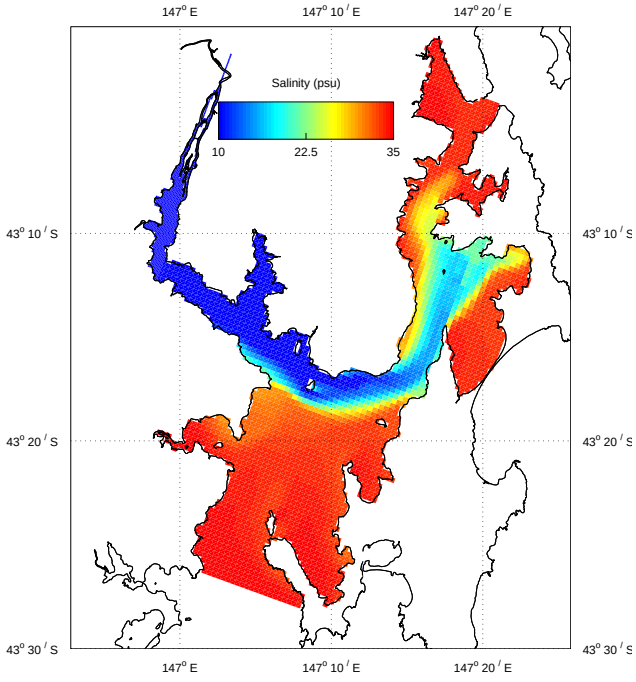


Figure 16.3.2 (h)

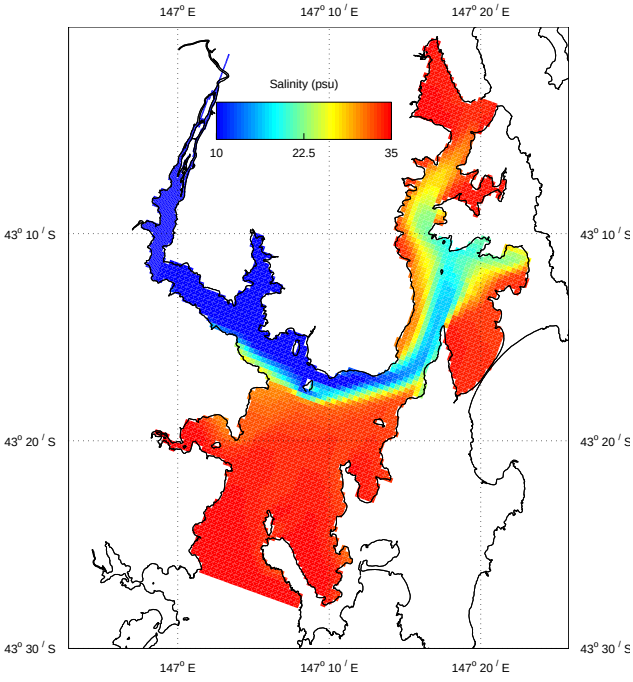


Figure 16.3.2 (i)

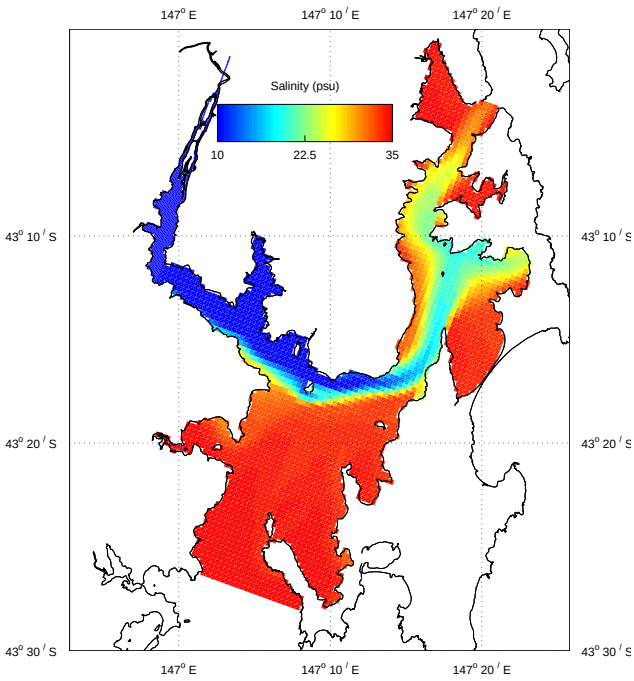


Figure 16.3.2 (j)

Figure 16.3.2 : Surface salinity distribution as a result of a 1940 m^3s^{-1} flow input at the head of the Huon Estuary. The output interval is 2 hours. The fresh water plume propagates up the channel favouring the left bank, due to the influence of Coriolis on the baroclinic mode. The plume collides with the coast at Missionary Bay where it bifurcates, sending one arm westward back into the channel and the other eastward further around the bay. The currents due to the tide are directed up the channel during this phase; when the tide turns the plume recedes back down the channel.

17. References.

Allan, D.J., A.R. Douglass, R.B. Rood and P.D. Guthrie (1991) Application of a monotonic upstream-biased transport scheme to three-dimensional constituent transport calculations. *Mon. Wea. Rev.*, **119**, 2456 – 2464.

Apel, J.R. (1987) Principles of ocean physics. International Geophysics Series, Vol 38. Academic Press, London, 631pp.

Arakawa, A. and Lamb, V.R. (1977) Computational design of the basic dynamical process of the UCLA general circulation model. *Methods in Computational Physics*, 17. Academic Press, pp. 173-265.

Asselin, R. (1972) Frequency filters for time integrations. *Mon. Weather Rev.*, **100**, 487 – 490.

Binliang, L. and A. Falconer (1997) Tidal flow and transport modelling using ULTIMATE QUICKEST scheme. *Journal of Hydraulic Engineering*, 123, 303 – 314.

Blackadar, A.K. (1962) The vertical distribution of wind and turbulent exchange in neutral atmosphere. *J. Geophys. Res.*, 67, 3095-3102.

Blanc, T.V. (1985) Variation of bulk-derived surface flux, stability, and roughness results due to the use of different transfer coefficient schemes. *J. Phys. Oceanogr.*, **15**, 650-669.

Blumberg, A.F. and J. Herring (1987) Circulation modelling using orthogonal curvilinear coordinates, in *Three-Dimensional Models of marine and Estuarine Dynamics*, Ed. J.C.J. Nihoul and B.M. Jamart, Elsevier.

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**, 281-301.

Blumberg, A.F. and L.H. Kantha (1985) Open boundary condition for circulation models, *Journal of Hydraulic Engineering*, 111, pp237-255.

Budyko, M.I. (1963) Atlas Teplovogo Balansa, *Gidrometeorologicheskos Izdatel'skos*, Leningrad.

Burchard, H., K., Bolding and M.R. Villarreal (1999) GOTM, a general ocean turbulence model. Theory implementation and test cases. Technical Report EUR 18745EN, European Commission, 103pp.

Burchard, H., K., Bolding (2001) Comparative analysis of four second-moment turbulence closure models for the oceanic mixed layer. *J. Phys. Oceanogr.*, **31**, 1943 – 1968.

Burchard, H., K., O. Peterson and T.P. Rippeth (1998) Comparing the performance of the Mellor Yamada and the k-ε turbulence models. *J. Geophys. Res.*, **103**, 10,543 – 10,554.

Bye, J.A.T. (1988) The flow series of Thallasso – models. Selected topics in atmospheric and marine sciences, Flinders University of South Australia, 59pp.

Bye, J.A.T. (1977a) The flow series of Thallasso – models. The FLOWM model, supplement to the FLOWC model. Selected topics in atmospheric and marine sciences, No 6, Flinders University of South Australia.

Camerlengo. A.L. and J.J. O'Brien (1980) Open boundary condition in rotating fluids. *J. Compt. Physics*, 35, 12 – 35.

Cartwright, D.E. and R.D. Ray (1990) Oceanic tides from Geosat altimetry. *J. Geophys. Res.*, 95 C3, 3069-3090.

Canuto, V.M., A. Howard, Y. Cheng and M.S. Dubovikov (2001) Ocean Turbulence. Part I: One-point closure model-momentum and heat vertical diffusivities. *J. Phys. Oceanogr.*, **31**, 1413 – 1426.

Casulli, V. and R.T. Cheng (1992) Semi-implicit finite difference methods for three-dimensional shallow water flow. *Int. J. Numer. Methods in Fluids*, 15, 629 – 648.

Chapman, D.C. (1985) Numerical treatment of cross-shelf boundaries in a barotropic coastal ocean model. *Journal of Physical Oceanography*, 15, 1060 – 1075.

Clark, T.L. and R.D. Farley (1984) Severe downslope windstorm calculations in two and three spatial dimensions using anelastic interactive grid nesting: A possible mechanism for gustiness. *J. Atmos. Sci.*, **41**, 329 – 350.

Craig, P.D., and Banner, M.L. (1994) Modelling wave-enhanced turbulence in the ocean surface-layer. *J. Phys. Oceanogr.*, **24**, 2546-2559.

Csanady, G.T. (1982) Circulation in the coastal ocean, D. Reidel Publishing company.

Demirov, E., E. Eifler, M. Ouberdous, and N. Hibma (1998) ISPRAMIX – a three-dimensional free surface model for coastal ocean simulations and satellite data assimilation on parallel computers. Technical report EUR 18129EN, European Commission, 76pp.
Dippner, J.W. (1998) Vorticity analysis of transient shallow water eddy fields at the river plume front of the River Elbe in the German Bight. *Journal of Marine Systems*, **14**, 117-133.
Eifler, W. and W. Schrimpf (1992) ISPRAMIX, a hydrodynamic program for computing regional sea circulation patterns and transfer processes. CEC Report EUR 14856 EN.
Eringen, A.C. (1962) Nonlinear theory of continuous media. McGraw Hill, New York.
Flather, R.A. (1976) A tidal model of the northwest European continental shelf. *Mem. Soc. R. Sci. Liege, Ser 6,* **10**, 141-164.
Flather, R.A. (1988) A numerical investigation of tides and diurnal-period continental shelf waves along Vancouver Island. *J. Phys. Oceanogr.*, **18**, 115-139.
Fox, A.D. and S.J. Maskell (1995) Two-way interactive nesting of primitive equation ocean models with topography, *Journal of Physical Oceanography*, Volume: 25, (1995), pp. 2977—2996.
Galperin, B., L.H. Kantha, S. Hassid and A. Rosati (1988) A quasi-equilibrium turbulent energy model for geophysical flows. *J. Atoms. Sci.*, 45, 55 – 62.
Gill, A. E. (1982) Atmosphere-Ocean Dynamics. *Academic Press Inc*.
GOTM Manual. www.gotm.net/pages/documentation/manual/pdf/a4.pdf
Haney, R.L. (1971) Surface thermal boundary condition for ocean circulation models. *J. Phys. Oceanog.*, 1, 241 – 248.
Hastings, C. (1955) Approximations for digital computers. *Princeton Univ. Press.* Princeton, N.J.
Herzfeld, M and M. Tomczak (1997) Numerical modelling of sea surface temperature and circulation in the Great Australian Bight. *Prog. Oceanog.*, 39, pp 29-78.
Hipsey, M.R., J.R. Romero, J.P. Antenucci and D. Hamilton (2006) Computational aquatic ecosystem dynamics model: CAEDYM v2. v2.3 science manual. Centre for Water Research, University of Western Australia. 90pp.
Israeli, M. and S.A. Orszag (1981) Approzimation of radiation boundary conditions. *J. Compt. Physics*, 41, 115 – 135.
Jensen, T.G. (1998) Open boundary conditions in stratified ocean models. *Journal of Marine Systems*, **16**, 297 – 322.
Jones, N.L., Monosmith, S.G. (2008) Modelling the influence of wave-enhanced turbulence in a shallow tide- and wind-driven water column. *J. Geophys. Res.*, 113, C03009, doi:10.1029/2007JC004246.
Kalnay, E., M. Kanamitsu, R. Kistler, W. Collins, D. Deaven, L. Gandin, M. Iredell, S. Saha, G. White, J. Woollen, Y. Zhu, M. Chelliah, W. Ebisuzaki, W. Higgins, J. Janowiak, K.C. Mo, C. Ropelewski, J. Wang, A. Leetmaa, R. Reynolds, R. Jenne and D. Joseph (1996) The NCEP/NCAR 40-year reanalysis project. *Bull. Amer. Meteor. Soc.*, **77**, 437-471.
Kantha, L.H., Clayson, C.A. (1994) An improved mixed layer model for geophysical applications. *J. Geophys. Res.*, **99**, 25235-25266.
Kitaigorodskii, S.A., O.A. Kuznetsov and G.N. Panin (1973) Coefficients of drag, sensible heat and evaporation in the atmosphere over the surface of the sea. *Izv. Acad. Sci. USSR*
Kondo, J. (1975) Air-sea bulk transfer coefficients in diabatic conditions. *Boundary-Layer Meteorology*, **9**, 91-112.
Kowalik, Z. and T.S. Murty (1993) Numerical modelling of ocean dynamics. Advanced series on ocean engineering, Volume 5. World Scientific, Singapore. 481pp.
Large, W.G., and S. Pond (1981) Open ocean momentum flux measurements in moderate to strong winds, *J. Phys. Oceanogr.*, **11**, 324-336.
Leonard, B.P. (1979) A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer methods in applied Mech. and Eng.*, 19, 59 – 98.
Leonard, B.P. (1991) The ULTIMATE conservative difference scheme applied to unsteady one-dimensional advection. *Comp. Methods in Appl. Mech. and Eng.*, **19**, 17 – 74.
Leonard, B.P. (1994) Note on the von Neumann stability of explicit one-dimensional advection schemes. *Comput. Methods Appl. Mech. Engrg.*, 118, 29-46.
Longuet-Higgins, M.S. (1970) Longshore currents generated by obliquely incident sea waves. *J. Geophys. Res.*, 101(C10), 22705 - 22714.

Marchesiello, P., J.C. McWilliams and A. Shchepetkin (2001) Open boundary conditions for long term integration of regional oceanic models. *Ocean Modelling*, **3**, 1-20.
Martinsen, E.A. and H. Engedahl (1987) Implementation and testing of a lateral boundary scheme as an open boundary condition in a barotropic ocean model. *Coastal Eng.*, 11, 603-627.
Masagutov, T.F. (1981) Calculation of the vertical turbulent fluxes in the near-water atmospheric layer over the ocean in tropical latitudes. *Meteor. Gidrol.*, **12**, 61-68.
Mellor, G.L. (1992) User's guide for a three-dimensional, primitive equation, numerical ocean model. Princeton University, Princeton, NJ, 34pp.
Mellor, G.L. (1996) Introduction to physical oceanography. *American Institute of Physics Press,* Woodbury, N.Y, 260pp.
Mellor, G.L. and A.F. Blumberg (1985) Modelling vertical and horizontal diffusivities with the sigma coordinate system. *Mon. Wea. Rev.*, **113**, 1379 – 1383.
Mellor, G.L., T. Ezer and L.Y. Oey (1994) The pressure gradient condrum of sigma ocean models. *J. Atmos. Oceanic Technol.*, 11, 1126 -1134.
Mellor, G.L., T. Ezer and L.Y. Oey (1998) Sigma coordinate pressure gradient errors and the seamount problem. *J. Atmos. Oceanic Technol.*, 15, 1122-1131.
Mellor, G.L. and T. Yamada (1982) development of a turbulence closure model for geophysical fluid problems. *Rev. Geophys.*, **20**, 851 – 875.
Mertz, M. and D.G. Wright (1992) Interpretations of the JEBAR term. *Journal of Physical Oceanography*, **22**, 301 - 305.
Miller, M.J. and A.J. Thorpe (1981) Radiation conditions for the lateral boundaries of limited-area numerical models. *Q. J. R. Meteorol. Soc.*, 107, 615 - 628.
Moore, W.J. (1963) Physical chemistry, Longman, London.
Müller, P. (1995) Ertel's potential vorticity theorem in physical oceanography**.** *Reviews of Geophysics,* **33**, 67-97.
Munk, W.A., Anderson, E.R. (1948) Notes on the theory of the thermocline. *J. Marine Res.*, 3, 276-295.
Oberhuber, J.M. (1988) An atlas based on the 'COADS' data set: the budgets of heat, buoyancy and turbulent kinetic energy at the surface of the global ocea. *Max-Planck-Institut fur Meteorologie*, Report no. 15, Hamburg.
Orlanski, I (1976) A simple boundary condition for unbounded hyperbolic flows. *J. Compt. Physics*, 21, 251 – 269.
Palma, E.D. and R.P. Matano (1998) On the implementation of passive open boundary conditions for a general circulation model: The barotropic mode. *J. Geophys. Res.*, 103(C1), 1319 - 1341.
Palma, E.D. and R.P. Matano (2001) Dynamical impacts associated with radiation boundary conditions. *J. Sea Res.*, **46**, 117-132.
Paulson, C. A. and Simpson, J. J. (1977) Irradiance measurements in the upper ocean. *J. Phys. Oceanogr.*, **7**, 952-956.
Phillips, N.A. (1957) A coordinate system having some special advantages for numerical forecasting. *J. of Meteorology*, 14, 184 -185.
Pond, S. and G.L. Pickard (1983) Introductory dynamical oceanography, *Pergamon Press*, Oxford, 329pp.
Press, W.H., B.P. Flannery, S.A. Teukolsky and W.T. Vetterling (1992) Numerical recipes in Fortran 77 : the art of scientific computing. *Cambridge University Press*, Cambridge, UK.
Rattray, M. (1982) A simple exact treatment of the baroclinicity-bathymetry interaction in a frictional, iterative, diagnostic ocean model. *Journal of Physical Oceanography*, **12**, 997 - 1003.
Raymond, W.H. and H.L. Kuo (1984) A radiation boundary condition for multidimensional flows. *Quarterly Journal of the Royal Meteorological Society*, **110**, 535-551.
Reed, R.K. (1977) On estimating insolation over the ocean. *J. Phys. Oceanogr.*, **6**, 781-800.
Roed, L.P and C. Cooper (1987) A study of various open boundary conditions for wind-forced barotropic numerical ocean models, in *Three-dimensional Models of marine and Estuarine Dynamics*, edited by J.C.J. Nihoul and B.N. Jamart, pp 305 – 335, Elsevier, New York.
Roed, L.P. and O.M. Smedstad (1984) Open boundary conditions for forced waves in a rotating fluid. *SIAM J. Sci. Stat. Comput.*, **5**, 414-426.

Rood, R.B. (1987) Numerical advection algorithms and their role in atmospheric transport and chemistry models. *Reviews of Geophysics*, **25**, 71 – 100.
Schumann, U., Gerz, T. (1995) Turbulent mixing in stably stratified shear flows, *J. Appl. Meteorol.*, **34**, 33-48.
Simons, T.J. (1974) Verification of numerical models of lake Ontario. Part I, circulation in spring and early summer. *Journal of Physical Oceanography*, **4**, 507 - 523.
Simpson, J.J. and T.D. Dickey (1981) The relationship between downward irradiance and upper ocean structure. *Journal of Physical Oceanography*, **11**, 309 - 323.
Smagorinsky, J. (1963) General circulation experiments with the primitive equations, I. The basic experiment. *Mon. Wea. Rev.*, **91**, 99 – 164.
Slordal, L.H. and J.E. Weber (1995) Adjustment to JEBAR forcing in a rotating ocean. *Journal of Physical Oceanography*, **26**, 657 - 670.
Sommerfeld, A. (1949) Partial differential equations, *Lect. Theoret. Phys.*, vol 6, Academic, San Diego.
Stevens, D.P. (1990) On open boundary conditions for three dimensional primitive equation ocean circulation models. *Geophysic. Astrophys. Fluid Dynamics*, **51**, 103 – 133.
Sutton, O.G. (1953) Micrometeorology. *M^{C}Graw-Hill*, New York.
Tang, Y.M., and R. Grimshaw (1996) *J. Geophys. Res.*, 101(C10), 22705 - 22714.
Toba, Y. (1978) Stochastic form of the growth of wind waves in a single-parameter representation with physical implications. *J. Phys. Oceanogr.*, **8**, 494-507.
Tranter, D.J. and G.S. Leech (1987) Factors influencing the standing crop of phytoplankton on the Australian Northwest Shelf seaward of the 40m isobath. *Continental Shelf Res.*, 7, 115 – 133.
Umlauf, L.H., and Burchard, H, (2003) A generic length-scale equation for geophysical turbulence models. *J. Mar. Res.*, **61**, 235-265.
Umlauf, L., H. Burchard and K. Hutter (2003) Extending the k-ω turbulence model towards oceanic applications. Ocean Modelling, **5**, 195 – 218.
Van Leer, B. (1979) Towards the ultimate conservative difference scheme. V: a second order sequel to Godanov's method. *J. Comput. Phys.*, 32, 101-136.
Warner, J.C., C.R. Sherwood, H.G. Arango and R.P. Signell (2005) Performance of four turbulence closure models implemented using a generic length scale method. *Ocean Modelling*, **8**, 81-113.
Wilcox, D.C. (1988) Reassessment of the scale-determining equation for advanced turbulence models. *AIAA Journal*, **26(11)**, 1299-1310.
Yanenko, N.N. (1971) The method of fractional steps. *Springer-Verlag, New York*, 160pp.
Zillman, J.W. (1972) A study of some aspects of the radiation and heat budgets of the southern hemisphere oceans. *Bureua of meteorology, Meteorological study no. 26*, Australian Govt. Pub. Service, Canberra.
Zhang, D., H. Chang, L.S. Nelson, T.T. Warner and J.M. Fritsch (1986) A two-way interactive nesting procedure with variable terrain resolution. Mon. Wea. Rev, 14, 1330 – 1339.