

OCEA0036-1

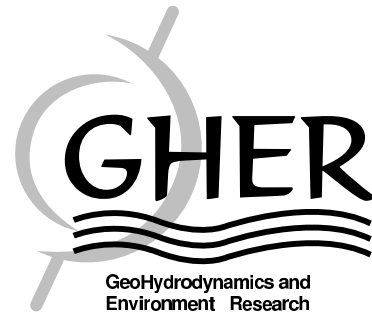
Structure and application of numerical ocean models

Alexander Barth

a.barth@ulg.ac.be

Revision 1.6

Université
de Liège



September 18, 2014

Contents

1	Introduction	5
2	Equations for hydrodynamic flow	8
2.1	Navier-Stokes equations	8
2.2	Non-hydrostatic primitive equations	12
2.3	Primitive equations	12
2.4	Shallow water equations	14
2.5	Quasi-Geostrophic dynamics	16
3	Boundary conditions	19
3.1	Surface boundary conditions	20
3.1.1	The momentum flux	20
3.1.2	Heat flux	20
3.1.2.1	Net long-wave radiation	21
3.1.2.2	Latent heat flux	21
3.1.2.3	Sensible heat flux	22
3.1.2.4	The solar heat flux	23

3.2	Bottom boundary conditions	24
3.3	Lateral boundary condition	25
3.3.1	Coast line	25
3.3.2	Open-ocean boundary conditions	25
3.3.2.1	Dirichlet boundary conditions	26
3.3.2.2	Radiation boundary conditions	26
3.3.2.3	Flow relaxation	27
3.3.2.4	Flather boundary condition	27
4	Model grids	31
4.1	Vertical coordinate	32
4.1.1	General coordinate transformation	34
4.1.2	z -coordinate	43
4.1.3	σ -coordinate	45
4.1.4	Isopycnals	51
4.2	Horizontal grid	52
4.2.1	Structured mesh	52
4.2.1.1	Cartesian mesh	52
4.2.1.2	Spherical mesh	52
4.2.1.3	Generalized orthogonal mesh	55
4.2.2	Grid staggering	59
4.2.3	Unstructured mesh	62
4.3	Time stepping	66
5	Solving model equations on a grid	71
5.1	Finite difference	72
5.2	Finite volume	75
5.3	Finite elements	76
5.4	Spectral methods	79

6	Sub-grid scale processes	81
6.1	Surface mixed layer	81
6.2	Bottom boundary layer	82
6.3	Horizontal sub-grid scale process	82
7	Programming aspects	84
7.1	Programming languages	85
7.2	Elements of a programming language	86
7.2.1	Elementary types	86
7.2.2	Arrays and structures	87
7.2.3	Statements and commands	88
7.2.4	Subroutines and functions	89
7.3	General structure of an ocean model	93
8	Data assimilation	94
A	Transformation of coordinates	95
A.1	Example	99
B	Measures of humidity	101
B.1	Definitions	101
B.2	Mixing ratio and specific humidity	102
B.3	The ideal gas law	103
B.4	Water vapour saturation pressure	103
B.5	Relative humidity	104
B.6	From water vapour pressure to specific humidity	104
C	NetCDF	106
C.1	Fortran 90	106

C.1.1	Reading NetCDF files	106
C.1.2	Writing NetCDF files	109
C.2	Matlab and Octave	113
C.2.1	Reading NetCDF files	113
C.2.2	Writing NetCDF files	114

References		121
-------------------	--	------------

Chapter 1

Introduction

Purpose of ocean models

- ▶ developed to understand and to predict the 3-D ocean circulation, as well as the distribution of temperature, salinity and biogeochemical variables.
- ▶ Knowing the ocean circulation allows to compute transports, which are important for e.g. assessing/predicting biological activity, climate interactions and transport of pollutants.

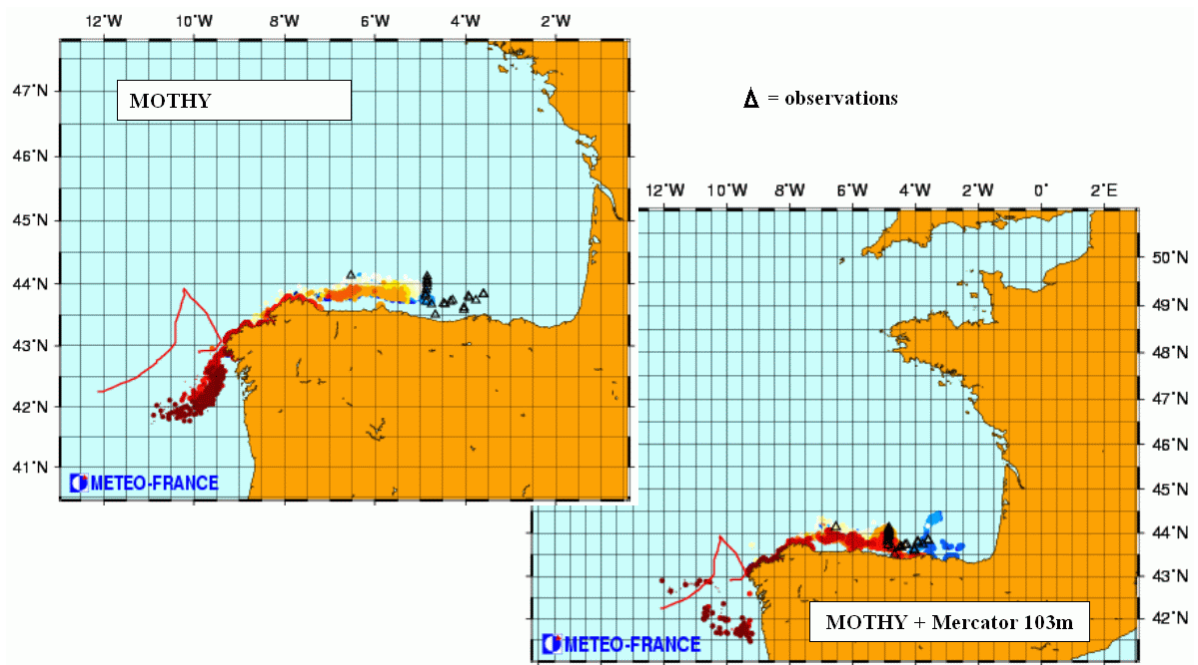


Figure 1.1: Oil spill forecast by METEO-FRANCE using currents from Mercator (adapted from [Daniel, 2004](#))

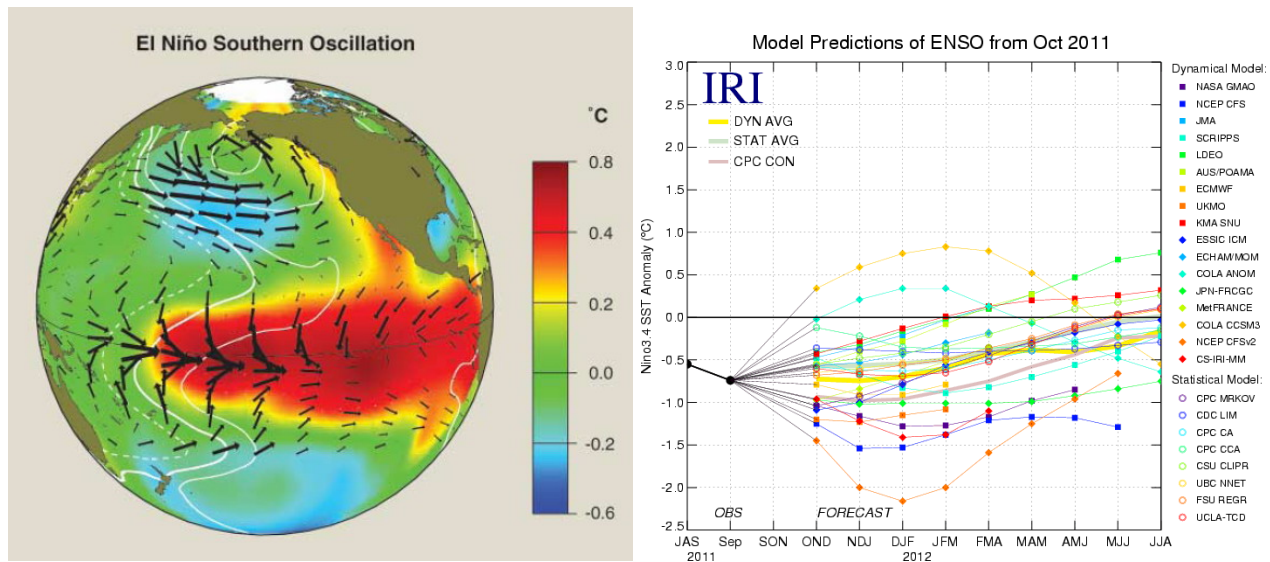


Figure 1.2: Left: SST (sea surface temperature) anomaly during El Niño (McPhaden *et al.*, 2006). Right: SST prediction in the Niño 3.4 region (http://iri.columbia.edu/climate/ENSO/currentinfo/SST_table.html)

Chapter 2

Equations for hydrodynamic flow

Contents

2.1	Navier-Stokes equations	8
2.2	Non-hydrostatic primitive equations	12
2.3	Primitive equations	12
2.4	Shallow water equations	14
2.5	Quasi-Geostrophic dynamics	16

2.1. Navier-Stokes equations

The Navier-Stokes equations provide the basis for the simplified and approximated set of equations used in numerical ocean model. The terms in the Navier-Stokes equations can be interpreted as different processes. The

approximations are justified by introducing scales of variations which allow to estimate the magnitude of these processes and neglect some terms under the given conditions.

$$\frac{d\rho}{dt} + \rho(\nabla \cdot \mathbf{v}) = 0 \quad (2.1)$$

$$\rho \frac{d\mathbf{v}}{dt} + 2\rho\mathbf{\Omega} \wedge \mathbf{v} = -\nabla p + \rho g \mathbf{e}_z + \nabla \cdot \mathbf{F}^v \quad (2.2)$$

where $\mathbf{\Omega}$ is the angular velocity vector of the Earth and \mathbf{F}^v viscosity tensor of the flow. The operator ∇ and the material derivative are defined as:

$$\nabla = \mathbf{e}_x \frac{\partial}{\partial x} + \mathbf{e}_y \frac{\partial}{\partial y} + \mathbf{e}_z \frac{\partial}{\partial z} \quad (2.3)$$

$$\frac{d}{dt} = \frac{\partial}{\partial t} + u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} + w \frac{\partial}{\partial z} \quad (2.4)$$

The density ρ is computed using the state equations and the internal energy, salinity and pressure. Instead using internal energy, potential temperature is used which is related directly to internal energy.

$$\rho = \rho(T, S, p) \quad (2.5)$$

Temperature and salinity are governed by advection-diffusion equations:

$$\rho \frac{dT}{dt} = \nabla \cdot \mathbf{F}^T \quad (2.6)$$

$$\rho \frac{dS}{dt} = \nabla \cdot \mathbf{F}^S \quad (2.7)$$

Exercise 1:

Give an interpretation of each term in the Navier-Stokes equations for a rotating fluid

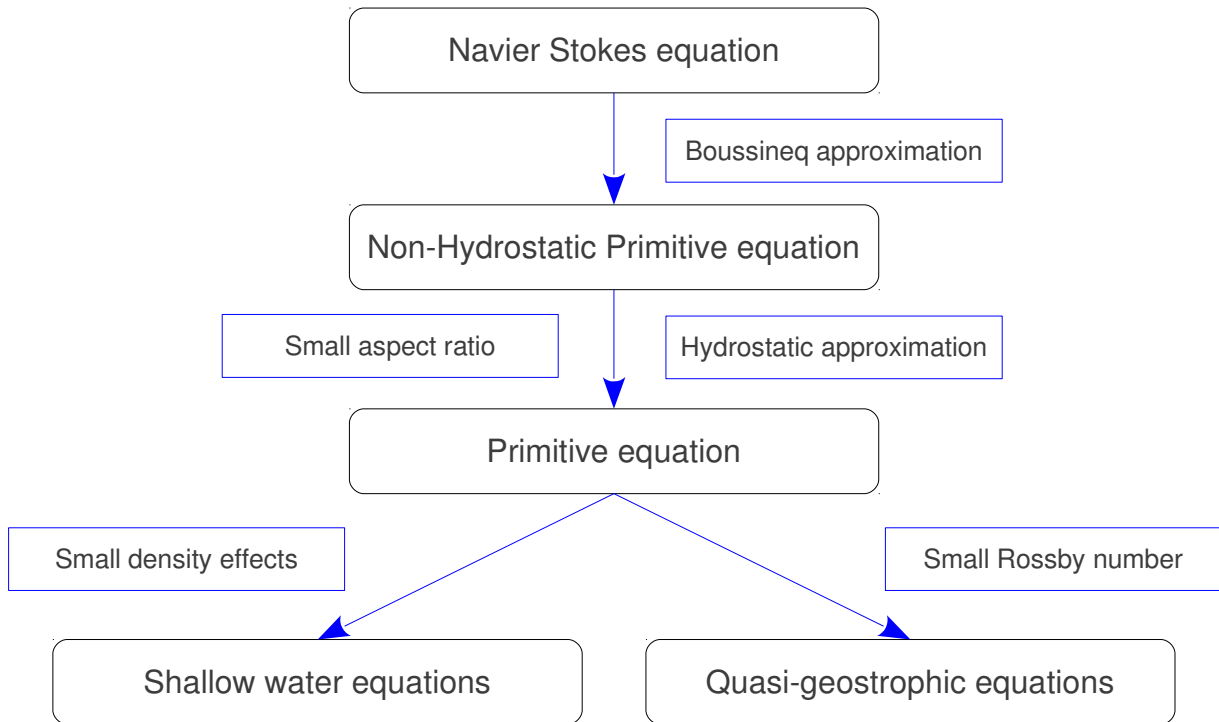


Figure 2.1: Different level of approximation of geophysical fluids

2.2. Non-hydrostatic primitive equations

The non-hydrostatic primitive equations are obtained by applying the Boussinesq approximation to the Navier-Stokes equations. In the Boussinesq approximation, density variations are neglected except for gravity. **Boussinesq approximation removes sound waves** in the ocean which would otherwise require a very small time step. Under the Boussinesq approximation, the total mass of the fluid is no longer conserved but the total volume is. This can introduce some difficulties in modeling effects such as sea level rise due to thermal expansion.

$$\nabla \cdot \mathbf{v} = 0 \quad (2.8)$$

$$\frac{d\mathbf{v}}{dt} + 2\boldsymbol{\Omega} \wedge \mathbf{v} = -\frac{1}{\rho_0} \nabla p + \frac{\rho g}{\rho_0} \mathbf{e}_z + \frac{1}{\rho_0} \nabla \cdot \mathbf{F}^v \quad (2.9)$$

where ρ_0 the reference density.

Exercise 2:

How are the maximum allowable time step and wave speed linked?

2.3. Primitive equations

In most circumstances, the vertical momentum equation is dominated by the pressure gradient and gravity. In the hydrostatic approximation, the pressure gradient is assumed to balance perfectly gravity and all other terms are neglected. The vertical velocity is no longer computed prognostically, but it is diagnosed based on the continuity equation.

$$\nabla \cdot \mathbf{v} = 0 \quad (2.10)$$

$$\frac{d\mathbf{u}}{dt} + 2\boldsymbol{\Omega} \wedge \mathbf{u} = -\frac{1}{\rho_0} \nabla_h p + \frac{1}{\rho_0} \nabla \cdot \mathbf{F}^u \quad (2.11)$$

$$\frac{\partial p}{\partial z} = -\rho g \quad (2.12)$$

The differential operator ∇_h is defined as:

$$\nabla_h = \mathbf{e}_x \frac{\partial}{\partial x} + \mathbf{e}_y \frac{\partial}{\partial y} \quad (2.13)$$

The velocity \mathbf{v} is decomposed into its horizontal \mathbf{u} and vertical w component:

$$\mathbf{v} = \mathbf{u} + w\mathbf{e}_z \quad (2.14)$$

Note:

- ▶ If one subtracts from the pressure p the hydrostatic pressure due to a constant density ρ_0 , one obtains the generalized pressure (apart from a constant factor ρ_0):

$$q = \frac{p}{\rho_0} + gz \quad (2.15)$$

where g is gravitational acceleration. The buoyancy b is given by the state equation $\rho(T, S)$:

$$b = -\frac{\rho(T, S) - \rho_0}{\rho_0} g \quad (2.16)$$

Under the hydrostatic approximation, the generalized pressure and the buoyancy are related by:

$$\frac{\partial q}{\partial z} = b \quad (2.17)$$

Instead of working with pressure p and density ρ , some ocean models work with generalized pressure q and buoyancy b . The effect of rounding errors due to the finite precision of floating number is smaller with the later (*Can you explain why the rounding error is smaller?*).

- ▶ Since the vertical velocity is much smaller than the horizontal velocity, the Coriolis force is generally simplified as:

$$2\boldsymbol{\Omega} \wedge \mathbf{v} = f\mathbf{e}_z \wedge \mathbf{u} \quad (2.18)$$

where f is called the Coriolis frequency.

- ▶ If a model is hydrostatic, **non-hydrostatic effects such as deep water formation have to be parametrized.**

2.4. Shallow water equations

If the aspect ratio is small,

$$\delta = \frac{H}{L} \ll 1 \quad (2.19)$$

and if the fluid is homogeneous, then the depth-integrated current U and V are governed by:

$$\frac{\partial \eta}{\partial t} + \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} = 0 \quad (2.20)$$

$$\frac{dU}{dt} - fV = -g \frac{\partial \eta}{\partial x} + \tau_x \quad (2.21)$$

$$\frac{dV}{dt} + fU = -g \frac{\partial \eta}{\partial y} + \tau_y \quad (2.22)$$

where η is surface height, (τ_x, τ_y) is the total friction (surface and bottom friction), and material derivative is now:

$$\frac{d}{dt} = \frac{\partial}{\partial t} + u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} \quad (2.23)$$

If the fluid is not homogeneous and its density variations are known, then an additional term called the baroclinic pressure gradient is included in equations (2.21) and (2.22). The shallow water equations are also solved in three-dimensional numerical ocean models to simulate the evolution of the free surface.

The rigid lid approximation neglects the sea surface height variations in the continuity equation:

$$\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} = 0 \quad (2.24)$$

The surface elevation is no longer computed prognostically, but is chosen such that the previous equations are satisfied. This approximation removes surface gravity waves. Surface gravity waves are fast waves (wave speed of \sqrt{gH}) and introduce thus a severe CFL stability criterion.

2.5. Quasi-Geostrophic dynamics

Quasi-Geostrophic equations approximate the flow of the ocean if the temporal Rossby number, the Rossby number and the Ekman number are much smaller than one,

$$Ro_T = \frac{\text{Acceleration}}{\text{Coriolis}} = \frac{1}{fT} \ll 1 \quad (2.25)$$

$$Ro = \frac{\text{Inertia}}{\text{Coriolis}} = \frac{U}{fL} \ll 1 \quad (2.26)$$

$$Ek = \frac{\text{Vert. friction}}{\text{Coriolis}} = \frac{\nu}{fH^2} \ll 1 \quad (2.27)$$

In this case, the pressure gradient is mostly balanced by the Coriolis force (geostrophic equilibrium). It is also assumed that the density variations $\rho'(x, y, z, t)$ around a average density profile $\bar{\rho}(z)$ are small:

$$\rho = \bar{\rho}(z) + \rho'(x, y, z, t) \quad |\bar{\rho}| \gg |\rho'| \quad (2.28)$$

Due to the hydrostatic equilibrium, the pressure can be decomposed in a similar way:

$$p = \bar{p}(z) + p'(x, y, z, t) \quad |\bar{p}| \gg |p'| \quad (2.29)$$

The quasi-geostrophic equations are derived by substituting horizontal velocity components by the corresponding pressure gradient in the momentum equations, which gives a evolution equations for the potential vorticity q and an equations for the stream function ϕ :

$$\frac{\partial q}{\partial t} + J(\psi, q) = \nu \frac{\partial^2}{\partial z^2} \nabla^2 \psi \quad (2.30)$$

$$q = \nabla^2 \psi + \frac{\partial}{\partial z} \left(\frac{f_0^2}{N^2} \frac{\partial \psi}{\partial z} \right) + \beta_0 y \quad (2.31)$$

where the Jacobian J is defined by:

$$J(\psi, q) = \frac{\partial \psi}{\partial x} \frac{\partial q}{\partial y} - \frac{\partial \psi}{\partial y} \frac{\partial q}{\partial x} \quad (2.32)$$

The velocity components, pressure and density are obtained by:

$$u = -\frac{\partial \psi}{\partial y} \quad (2.33)$$

$$v = \frac{\partial \psi}{\partial x} \quad (2.34)$$

$$w = -\frac{f_0}{N^2} \left(\frac{\partial^2 \psi}{\partial t \partial z} + J \left(\psi, \frac{\partial \psi}{\partial z} \right) \right) \quad (2.35)$$

$$p' = \rho_0 f_0 \psi \quad (2.36)$$

$$\rho' = -\frac{\rho_0 f_0}{g} \frac{\partial \psi}{\partial z} \quad (2.37)$$

Since only one evolution equations has to be solved, the models based on the quasi-geostrophic equations are numerically more efficient than model based on the primitive equations. However, the approximation need to derive the quasi-geostrophic equations limit their applicability.

Exercise 3:

We consider the 2d-quasi-geostrophic system with viscosity on an f -plane governed by:

$$\frac{\partial q}{\partial t} + J(\psi, q) = A_H \nabla^2 q + B_H \nabla^4 q \quad (2.38)$$

$$q = \nabla^2 \psi \quad (2.39)$$

where $A_H = 0.5 \text{ m}^2/\text{s}$ and $B_H = 1.6015 \cdot 10^{10} \text{ m}^4/\text{s}$. The initial condition is given by:

$$\tilde{r} = r(1 + \epsilon \cos(2\theta))/L \quad (2.40)$$

$$\psi_0 = \exp(-\tilde{r}^2) L^2 \omega_0 \quad (2.41)$$

where r and θ are the polar coordinates. The parameter $\epsilon = 0.03$ introduces a perturbation of the eddy's structure ($L = 100 \text{ km}$, $\omega_0 = 10^{-5} \text{ s}^{-1}$). The domain is a square $[-10L, 10L] \times [-10L, 10L]$. As time step 1000 s is suggested. For simplicity, q and ψ are prescribed to zero at the boundary.

Exercise from "Introduction to Geophysical Fluid Dynamics" by B. Cushman-Roisin and J.-M. Beckers.

Chapter 3

Boundary conditions

Contents

3.1	Surface boundary conditions	20
3.1.1	The momentum flux	20
3.1.2	Heat flux	20
3.2	Bottom boundary conditions	24
3.3	Lateral boundary condition	25
3.3.1	Coast line	25
3.3.2	Open-ocean boundary conditions	25

The equation of the previous chapter could not be solved for a fluid with finite extent without prescribing what happen at the boundary of the fluid.

3.1. Surface boundary conditions

At the ocean surface for example, the ocean and atmosphere exchange heat, water and momentum. These exchanges are prescribed at surface boundary conditions.

3.1.1. The momentum flux

The winds at the air-sea interface drag the surface water along its direction. This wind stress τ gives the momentum flux between ocean and atmosphere and it is parameterized by:

$$\tau = C_D \rho_a \|\mathbf{u}_a\| \mathbf{u}_a \quad (3.1)$$

ρ_a is the air density and \mathbf{u}_a the wind vector at the reference level. The drag coefficient C_D is parameterized (e.g. [Kondo, 1975](#)). The momentum flux is a vector with the same direction of the wind vector.

Exercise 4:

Equation (3.1) actually assumes that the ocean currents are much smaller than the winds (which is in general a realistic assumption). Propose a modification of this equation to take current speed into account.

3.1.2. Heat flux

The exchange of heat modifies the temperature of the ocean since the temperature is directly related to the internal energy E_i :

$$E_i = c_p \rho T \quad (3.2)$$

where c_p is the heat capacity at constant pressure and ρ is the density of sea-water. The turbulent temperature fluxes at the ocean surface are prescribed through the ocean-atmosphere exchange:

$$\nu_E \frac{\partial T}{\partial z} = \frac{Q^t}{c_p \rho} \quad (3.3)$$

where Q^t is the net thermal energy reaching the ocean surface per unit of length squared. Q^t is the sum of:

3.1.2.1. Net long-wave radiation

- ▶ corresponds to the infrared radiation that the ocean surface emits similar to the back-body radiation at a given temperature.
- ▶ can be reflected back to the ocean by the presence of clouds.
- ▶ the atmosphere emits also long-wave radiation that it partially absorbed by the ocean surface.
- ▶ the net long-wave radiation is the total flux due to these effects and it depends thus mainly on **sea-surface temperature**, **air temperature** and **cloud fraction**.

3.1.2.2. Latent heat flux

- ▶ due to a difference in the water vapor content of the air at the ocean surface and at the reference level.
- ▶ this gradient induces evaporation or condensation.
- ▶ to this mass transfer corresponds a heat exchange, which is equal to the rate of vaporisation times the latent heat of evaporation L .

3.1.2.3. Sensible heat flux

- ▶ due to the temperature difference between the air at the ocean surface and the air at the reference level.
- ▶ heat exchanged by conduction and is proportional to this temperature gradient, the heat conductivity of the ocean surface and the specific heat of air at constant pressure
- ▶ for the air temperature at the ocean surface, the sea surface is taken assuming a local equilibrium.

The latent heat flux and the sensible heat flux are parameterized by classical bulk turbulent transfer formulas (Rosati and Miyakoda, 1988; Castellari *et al.*, 1998):

$$Q_L = C_L L \rho_a \|\mathbf{u}_a\| (q_s - q_a) \quad (3.4)$$

$$Q_H = C_H c_{p_a} \rho_a \|\mathbf{u}_a\| (T_s - T_a) \quad (3.5)$$

where ρ_a is the air density, \mathbf{u}_a is the wind vector, q_s is the specific humidity of saturated air at T_s , q_a is specific humidity of air and c_{p_a} the heat capacity of air at constant pressure. For an air pressure p_a expressed in hPa, q_a is obtained by the air temperature T_a and the relative humidity r and q_s is obtained from the sea surface temperature T_s by:

$$q_a = r e_{\text{sat}}(T_a) \frac{\epsilon}{p_a} \quad (3.6)$$

$$q_s = e_{\text{sat}}(T_s) \frac{\epsilon}{p_a} \quad (3.7)$$

where $\epsilon = 0.622$ is the ratio of the gas constants of dry air and water vapor (see appendix B). Expressions (3.4) and (3.5) are well established bulk parameterizations for the latent and sensible heat flux. Matter of discussions are however the exchange coefficient C_H (Stanton number) and C_E (Dalton Number). Numerous parameterizations are proposed in the literature (e.g. Castellari *et al.*, 1998; Kondo, 1975).

3.1.2.4. The solar heat flux

- ▶ The solar (or short-wave) heat flux is sometimes included in the net heat flux at the ocean surface.
- ▶ However, the solar energy penetrates into the water column and heat the water not only at the surface.
- ▶ The solar heat flux is thus more realistically described as a energy source in the temperature equations:

$$\frac{\partial T}{\partial t} = \dots + \frac{1}{c_p \rho_0} \frac{\partial I}{\partial z} \quad (3.8)$$

By considering only two visible frequencies, the radiation flux I as a function of depth can be described by the following equation ($z = 0$ at the surface and negative in water):

$$I(z) = |Q_s| (A e^{g_1 z} + (1 - A) e^{g_2 z}) \quad (3.9)$$

where Q_s is the light intensity at the surface, $A = 0.58$ is the fraction long-wave solar energy and $g_1 = 0.35 \text{ m}^{-1}$ and $g_2 = 23.0 \text{ m}^{-1}$ are the absorption coefficients of the short-wave (blue) and long-wave (red) solar energy respectively of the visible spectrum. This distribution of the light intensity corresponds to the water of type I according to the classification of [Jerlov \(1968\)](#).

Exercise 5:

Explain on the basis of equation (3.9) why objects immersed in the ocean appear blue.

3.2. Bottom boundary conditions

- ▶ The ocean floor is generally treated as impermeable boundary.
- ▶ The velocity normal to the ocean floor is set to zero.
- ▶ Similar to the air-sea boundary, the ocean floor also exerts a friction on the flow parallel to ocean flow. This friction is often parameterized a quadratic or logarithmic friction laws.
- ▶ Bottom friction is crucial for tidal simulation.
- ▶ Prescribing the horizontal velocity components to zero is only a possibility if the bottom boundary layer is well resolved.
- ▶ Analytical ocean models use in general a linear bottom drag not because it is more realistic, but because it is much easier to obtain a analytical solution.

Exercise 6:

Explain why bottom friction is more important for simulating tides than for the general ocean circulation.

When a numerical ocean model is coupled to a sediment transport models, the bottom floor itself can vary in time over sufficiently long time scales.

3.3. Lateral boundary condition

3.3.1. Coast line

Formally, the lateral boundary at the coastline is similar to the bottom boundary condition. The coastline is generally treated as a wall. The velocity perpendicular to the coast-line is zero. Different options for the boundary conditions for the flow parallel to the coast-line are possible:

- ▶ no-slip: The velocity tangent to the coastline is set to zero (if lateral boundary layer is resolved).
- ▶ lateral drag: Turbulent viscosity is prescribed at the coastline, for example proportional to the square of the velocity (if lateral boundary layer is not resolved).
- ▶ free slip: The flow moves freely parallel to the coast (applicable if lateral boundary layer is much smaller than the grid size such that its effect can be ignored).

Rivers represent a fresh-water flux into the model domain. They can be represented as a boundary condition with prescribed salinity (and possibly temperature) and velocity. Rivers can also be modeled as a point source for salinity (and temperature) in the evolution equation of the tracers.

For applications such as storm surge modeling, the coastline can move due to inundation and the retreat of the water. Grid-cells can thus be either wet or dry. Special **wetting and drying scheme** have been developed for these applications. The challenge of these methods is to provide a numerical stable and volume conserving scheme.

3.3.2. Open-ocean boundary conditions

For high-resolution application, only a small portion of the global ocean can be covered. In these cases, it is necessary to introduce boundary conditions at the open-sea boundary.

3.3.2.1. Dirichlet boundary conditions

The simplest open-ocean boundary condition is to prescribe the values of the model variables at the open boundary (Dirichlet or clamped boundary conditions).

$$\phi = \phi^{\text{ext}} \quad \text{at the open-boundary} \quad (3.10)$$

where ϕ is any model variable. This approach however is rarely used since it suffers from several drawbacks:

- ▶ only in rare cases there are sufficient observations to provide ϕ^{ext} (a larger-scale model is thus often used)
- ▶ waves approaching the open boundary are in general reflected at the open boundary
- ▶ if the external data is not compatible with the model results at the boundary (due to problems in the model or in the external data) a spurious boundary layer is created with strong gradients. A strong spurious density gradient generates a strong spurious geostrophic flow which exacerbates the problem and can lead to numerical instabilities.

3.3.2.2. Radiation boundary conditions

To address the problem of wave reflection, the radiation boundary conditions are constructed to let a wave propagate freely out of the model domain:

$$\frac{\partial \phi}{\partial t} + c \frac{\partial \phi}{\partial n} = 0 \quad (3.11)$$

where n is the dimension perpendicular to the open-boundary and c is the wave propagation speed. For the Sommerfeld condition, c is constant and it must be determined *a priori*. [Orlanski \(1976\)](#) proposed a scheme where the propagation speed is determined by the flow one grid point from the open boundary and at the previous time step by:

$$c = -\frac{\partial \phi}{\partial t} / \frac{\partial \phi}{\partial n} \quad (3.12)$$

The method returns the correct propagation speed for a single wave (propagating at a constant speed) reaching the boundary at normal incidence. But the scheme can be problematic if the solution contains several waves at different propagation speed.

External data can be included in the radiation boundary condition by introducing an relaxation term.

$$\frac{\partial \phi}{\partial t} + c \frac{\partial \phi}{\partial n} = \frac{\phi^{\text{ext}} - \phi}{\tau} \quad (3.13)$$

where τ is the relaxation time-scale. The smaller the relaxation time-scale, the stronger the model is forced by the external data. The relaxation time-scale is in general adjusted to reflect the accuracy of the external data.

3.3.2.3. Flow relaxation

To allow a smoother transition between the external data and the model results, [Davis \(1976\)](#) introduced the flow relaxation method: the relaxation term is not only active at the open-boundary but also in some zone near the boundary. The relaxation is also added to the prognostic equations:

$$\frac{\partial \phi}{\partial t} + \dots = \frac{\phi^{\text{ext}} - \phi}{\tau(x, y)} \quad (3.14)$$

The coefficient $1/\tau(x, y)$ defines the flow relaxation zone and is only non-zero near the boundary.

3.3.2.4. Flather boundary condition

These previous boundary conditions do not take the dynamical relationship between the variables into account. [\(Flather, 1976\)](#) proposed a boundary condition for the shallow water equations. The propagation of a surface gravity wave approaching a boundary is described as:

$$\frac{\partial \eta}{\partial t} + \sqrt{gh} \frac{\partial \eta}{\partial n} = 0 \quad (3.15)$$

where h is the water depth. The 1-D approximation of the continuity equations can be written as:

$$\frac{\partial \eta}{\partial t} + h \frac{\partial \bar{v}_n}{\partial n} = 0 \quad (3.16)$$

By subtracting the previous equations, one obtains

$$\frac{\partial}{\partial n} \left(\bar{v}_n - \sqrt{\frac{g}{h}} \eta \right) = 0 \quad (3.17)$$

By integrating this equations across the open boundary, one obtain the Flather-boundary condition:

$$\bar{v}_n - \sqrt{\frac{g}{h}} \eta = \bar{v}_n^{\text{ext}} - \sqrt{\frac{g}{h}} \eta^{\text{ext}} \quad (3.18)$$

The Flather boundary condition provides only one constrain for two variables (elevation and normal velocity). The Flather boundary condition is often augmented by one of the boundary condition proposed by [Chapman \(1985\)](#), such as:

$$\eta_b^{n+1} = \frac{\eta_b^n + \mu_e \eta_{b+1}^{n+1}}{1 + \mu_e} \quad (3.19)$$

where $\mu_e = \sqrt{gh} \frac{\Delta t}{\Delta x_n}$, b is the grid index of the model boundary, $b+1$ is the index of the first grid point inside the model domain. This boundary condition can be obtained discretizing equations (3.15) using finite differences.

Concluding remarks for open boundaries:

- A more rigorous framework for deriving open boundary condition is the method of characteristics. The linearized system of equations are transformed into a system of independent equations for the characteristics. Each of these characteristics has its own propagation speed. The sign of the propagation speed at the boundary determines if it is an incoming or out-coming characteristic. An interesting discussion can be found in ([Blayo and Debreu, 2005](#)).

- ▶ An open-ocean boundary conditions for the primitive equations is a delicate task since they admit a broad spectrum of waves. Barotropic waves are in general faster than the ocean currents (sub-critical regime) while high order internal waves are slower than ocean currents (super-critical regime).
- ▶ The boundary condition play also a crucial role in model nesting: a coarse-grid model provides boundary condition of a fine-grid model. In one-way nesting, the coarse-grid model is independent of the fine-grid model. If in turn, the fine-grid model results are incorporated into the coarse-grid model one speaks of two-way nesting.

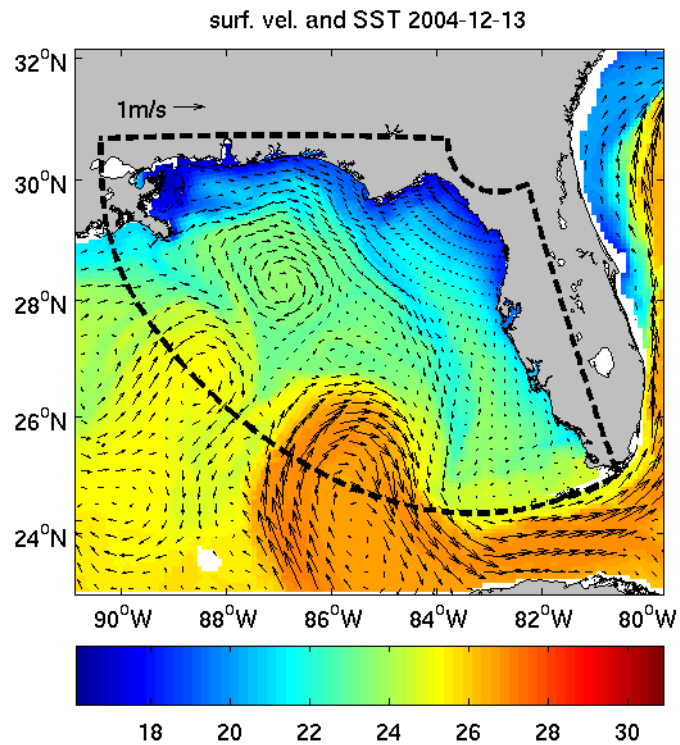


Figure 3.1: Example of 1-way nesting of ROMS in HYCOM ([Barth et al., 2008](#)). [Animation](#)

Chapter 4

Model grids

Contents

4.1	Vertical coordinate	32
4.1.1	General coordinate transformation	34
4.1.2	z -coordinate	43
4.1.3	σ -coordinate	45
4.1.4	Isopycnals	51
4.2	Horizontal grid	52
4.2.1	Structured mesh	52
4.2.2	Grid staggering	59
4.2.3	Unstructured mesh	62
4.3	Time stepping	66

- ▶ For a stratified fluid such as the ocean, the representation of gravity is crucial.
- ▶ In the vast majority in ocean models, the model grid lines are vertically aligned because of the the dominance of later versus vertical transport and hydrostatic balance.
- ▶ The horizontal and vertical grid are therefore be considered separately as two successive steps to generate the tri-dimensional model grid.

4.1. Vertical coordinate

The choice of vertical coordinate system is the single most important aspect of an ocean model's design ([Chassignet and Malanotte-Rizzoli, 2000](#); [Chassignet *et al.*, 2000](#)).

Different regimes are found in the vertical that a numerical ocean model has to simulated and a vertical coordinate has to resolve:

- ▶ Surface mixed layer: higher resolution near the surface is necessary to represent air-sea heat, fresh-water and momentum flux. Intense turbulent mixing and non-hydrostatic convection takes place in this weakly stratified layer. Those processes are in general parameterized. The currents in this layer are strong affected by the wind stress (surface Ekman layer). Below this layer, large temperature and salinity variation are in general observed (thermocline and halocline)
- ▶ Ocean interior: this part of the water column is in general well stratified. This stratification constrain the movement of tracers along direction of constant density. Water mass properties are thus maintained over very long time scales.
- ▶ Ocean bottom: bottom boundary layer exert friction on the overlying fluid. This is especially important for shallow areas. In some places, dense water masses flows down along the ocean floor. These overflows are

crucial in the formation of deep water. The bottom depth (*i.e.* the geometry of the basin) itself is also very important since the flow tends to follow lines of constant f/H (under unstratified conditions).

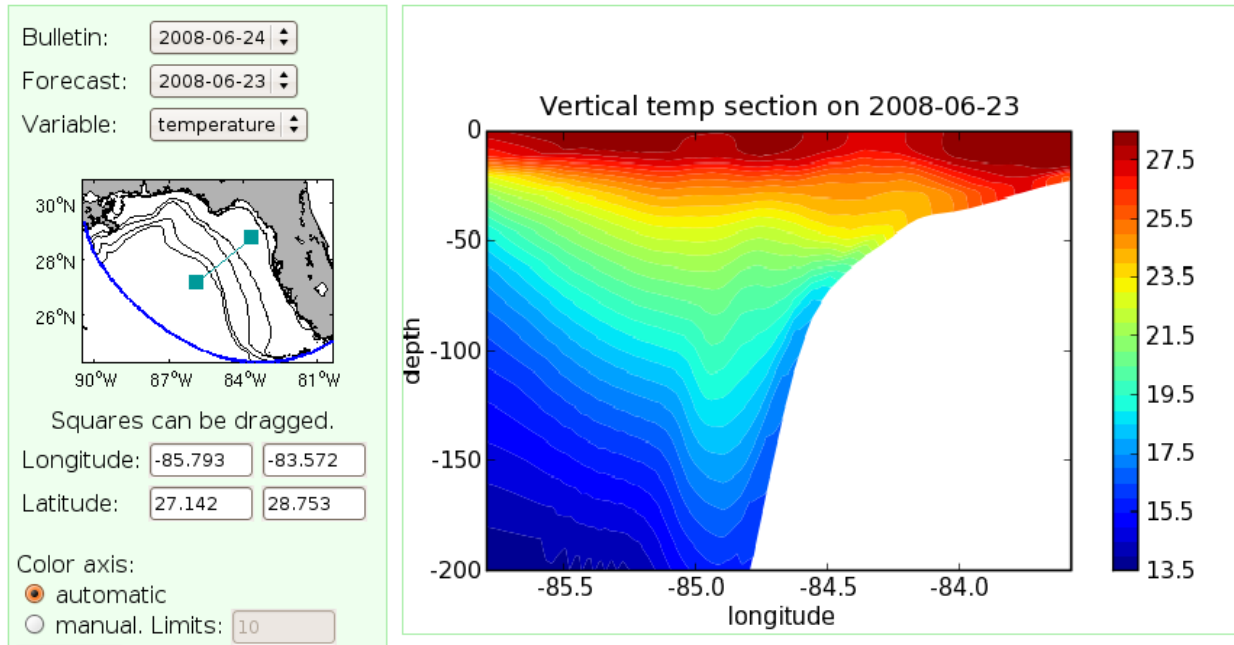


Figure 4.1: Vertical section of the WFS ROMS model (<http://ocgmod1.marine.usf.edu/WFS>)

4.1.1. General coordinate transformation

The easiest way to discretize the water column is to use the depth. But this is only one possibility. First, we examine the general coordinate system transformation $(x, y, z, t) \rightarrow (x', y', z', t')$:

$$x' = x \quad (4.1)$$

$$y' = y \quad (4.2)$$

$$z' = z'(x, y, z, t) \quad (4.3)$$

$$t' = t \quad (4.4)$$

The transformed variable z' may vary not only in space but also with time. This transformation is only invertible if z' is a uniformly increasing or decreasing function of z . We need now to express the primitive equations in the transformed coordinate system. Following equation (A.12) of appendix A, the derivatives are transformed as:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial x'} + \frac{\partial f}{\partial z'} \frac{\partial z'}{\partial x} \quad (4.5)$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial y'} + \frac{\partial f}{\partial z'} \frac{\partial z'}{\partial y} \quad (4.6)$$

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial z'} \frac{\partial z'}{\partial z} \quad (4.7)$$

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial t'} + \frac{\partial f}{\partial z'} \frac{\partial z'}{\partial t} \quad (4.8)$$

The derivative in x is not simply equal to the derivative in x' . Indeed, the derivative in x is taken for constant z while the derivative in x' is taken along constant z' . Since z' may depend on x , both are not necessarily equal. Note also the similarity in the transforming of the derivatives in x , y and t .

A central quantity in coordinate transformation is the **Jacobian**. The Jacobian of this transformation is

$$J = \frac{\partial z}{\partial z'} \quad (4.9)$$

The Jacobian corresponds to the local stretching of the new coordinate system relative to the old coordinate system.

The **material derivative** is often used to express the primitive equations. In cartesian coordinate, it is defined by:

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + u \frac{\partial f}{\partial x} + v \frac{\partial f}{\partial y} + w \frac{\partial f}{\partial z} \quad (4.10)$$

In the transformed coordinate, the material derivative becomes:

$$\frac{df}{dt} = \frac{\partial f}{\partial t'} + u \frac{\partial f}{\partial x'} + v \frac{\partial f}{\partial y'} + \omega \frac{\partial f}{\partial z'} \quad (4.11)$$

where ω is defined by:

$$\omega = \frac{\partial z'}{\partial t} + u \frac{\partial z'}{\partial x} + v \frac{\partial z'}{\partial y} + w \frac{\partial z'}{\partial z} \quad (4.12)$$

The first three terms of the rhs (right hand side) of the previous equations correspond to the material derivative of the surfaces of constant z' . $J\omega$ is thus the movement of the fluid relative to the surfaces of constant z'

The volume conservations

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \quad (4.13)$$

becomes

$$\frac{\partial J}{\partial t'} + \frac{\partial}{\partial x'}(Ju) + \frac{\partial}{\partial y'}(Jv) + \frac{\partial}{\partial z'}(J\omega) = 0 \quad (4.14)$$

with the volume conservation, the material derivative for a scalar f multiplied by the Jacobian J can be expressed in the following flux conservative form:

$$J \frac{df}{dt} = J \left(\frac{\partial f}{\partial t'} + u \frac{\partial f}{\partial x'} + v \frac{\partial f}{\partial y'} + \omega \frac{\partial f}{\partial z'} \right) = \frac{\partial}{\partial t'}(Jf) + \frac{\partial}{\partial x'}(Jfu) + \frac{\partial}{\partial y'}(Jfv) + \frac{\partial}{\partial z'}(Jf\omega) \quad (4.15)$$

Numerical models are generally based on the flux form of the evolution equations since they lead more easily to conservative schemes. The advection terms are formally similar to their expression in Cartesian coordinates where tracer f is replaced by Jf . The Jacobian takes into account that the real volume of a model grid cell varies in space and time.

The appearance of the Jacobian J to express the material derivative in conservative form is not surprising, since it is also necessary to perform integration in a different coordinate system:

$$\int_{\Omega} f(x, y, z) \, dx \, dy \, dz = \int_{\Omega'} f(x', y', z') \, J \, dx' \, dy' \, dz' \quad (4.16)$$

The equations governing the evolution of a tracers includes beside advection also the diffusion. The vertical diffusion in the transformed space would give:

$$\frac{\partial}{\partial z} \left(\nu \frac{\partial f}{\partial z} \right) = \frac{1}{J} \frac{\partial}{\partial z'} \left(\frac{\nu}{J} \frac{\partial f}{\partial z'} \right) \quad (4.17)$$

The horizontal velocity components, the advection and diffusion terms are similar to those of tracer. The Coriolis and buoyancy terms do not contain a spatial derivative. Thus they are not changed by the coordinate transformation. The remain term to complete the momentum equation is in the pressure gradient. The x -component of the pressure gradient becomes:

$$\frac{\partial p}{\partial x} = \frac{\partial p}{\partial x'} + \frac{\partial p}{\partial z'} \frac{\partial z'}{\partial x} \quad (4.18)$$

The pressure gradient is thus the sum of two components. If the pressure is only a function of z (as it is approximately the case in the ocean), the horizontal pressure gradient is zero and both terms should cancel out each other. This is in general not the case for the discretized pressure gradient. The residual pressure gradient drives a spurious current. This is the so called pressure gradient problem (e.g. [Haney, 1991](#); [Deleersnijder and Beckers, 1992](#)).

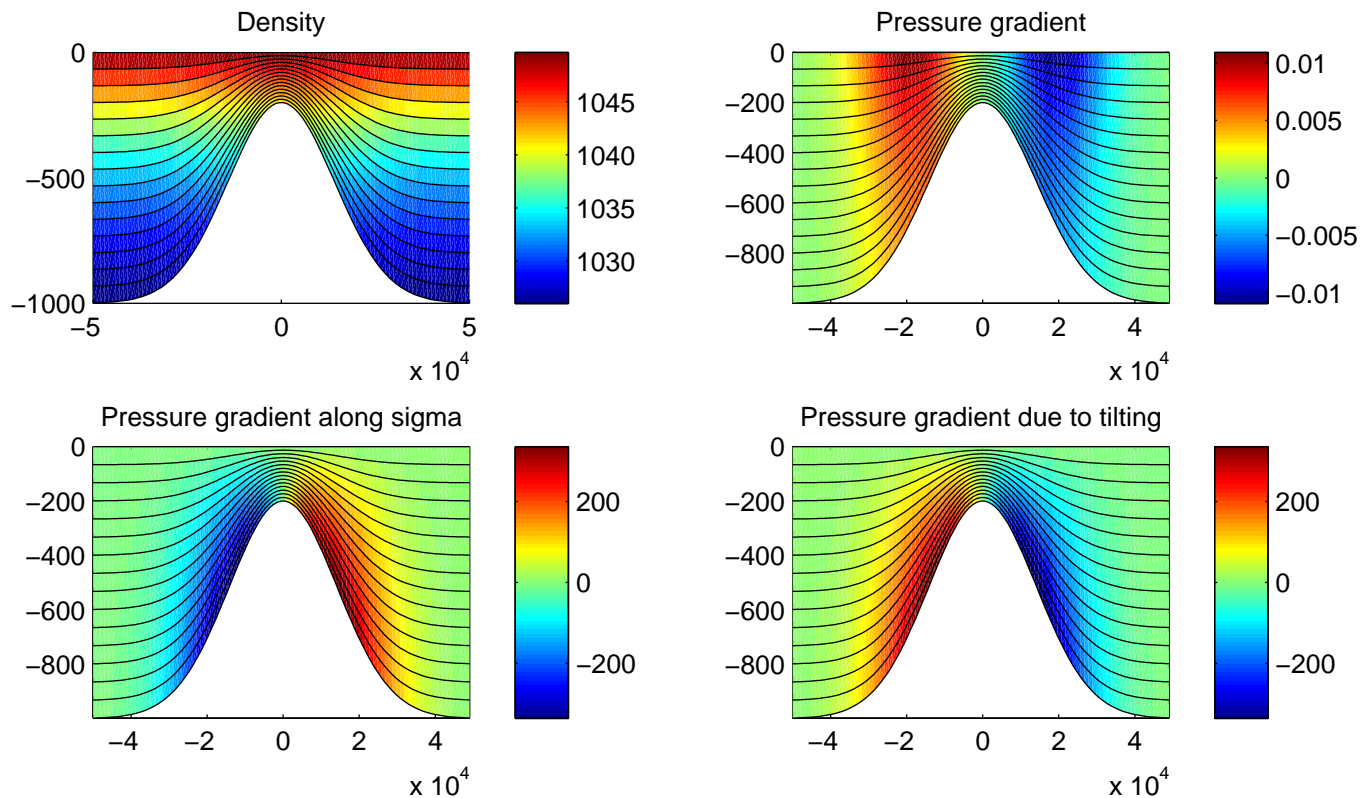


Figure 4.2: "Naive" pressure gradient discretization for the sea mount problem

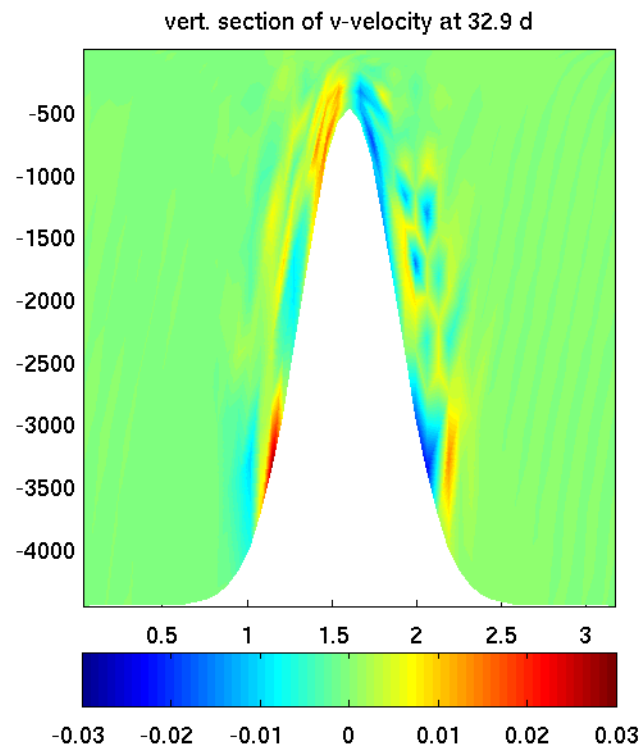
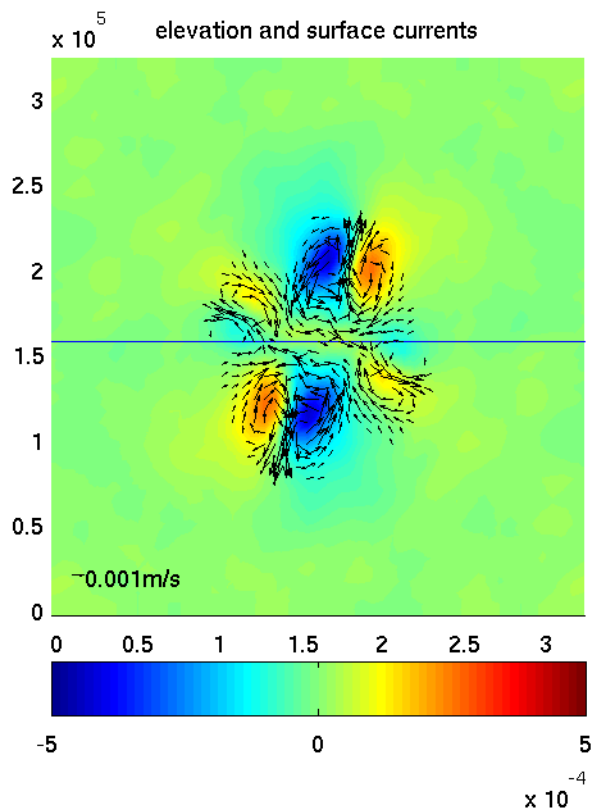


Figure 4.3: Sea mount test-case with ROMS using the spline density Jacobian formulation by [Shchepetkin and McWilliams \(2003\)](#). [Animation of the sea mount simulation](#).

The vertical variation of pressure are known because of the hydrostatic equilibrium:

$$\frac{\partial p}{\partial z} = \frac{1}{J} \frac{\partial p}{\partial z'} = -\rho g \quad (4.19)$$

The pressure gradient in transformed coordinate can thus be written as:

$$\frac{\partial p}{\partial x} = \frac{\partial p}{\partial x'} - \rho g J \frac{\partial z'}{\partial x} \quad (4.20)$$

$$= \frac{\partial p}{\partial x'} + \rho g \frac{\partial z}{\partial x'} \quad (4.21)$$

since

$$\frac{\partial z'}{\partial x} = -\frac{\partial z'}{\partial z} \frac{\partial z}{\partial x'} \quad (4.22)$$

$$= -\frac{1}{J} \frac{\partial z}{\partial x'} \quad (4.23)$$

The pressure gradient (equation 4.21) can further be transformed into:

$$\frac{\partial p}{\partial x} = \frac{\partial p}{\partial x'} + \rho \frac{\partial gz}{\partial x'} + gz \frac{\partial \rho}{\partial x'} - gz \frac{\partial \rho}{\partial x'} \quad (4.24)$$

$$= \frac{\partial P}{\partial x'} - gz \frac{\partial \rho}{\partial x'} \quad (4.25)$$

where $P = p + \rho gz$ is the Montgomery potential.

The pressure gradient can be interpreted as the gradient of the pressure obtained by vertical interpolated of the pressure of the neighboring cells (figure 4.1.1). However, if the slope of the grid-lines increases, the interpolated

can become an extrapolation. The depth at which the vertical pressure gradient is evaluated is in this case no longer consistent with the depth of the horizontal pressure gradient. This problem is called *hydrostatic consistency*.

$$\left| \frac{\frac{\partial x'}{\partial z}}{\frac{\partial z'}{\partial z}} \right| \leq \frac{\Delta z'}{\Delta x} \quad (4.26)$$

To reduce the pressure gradient problem, it is thus not sufficient to increase the vertical resolution alone. Hydrostatic consistency requires that vertical and horizontal resolution are refined.

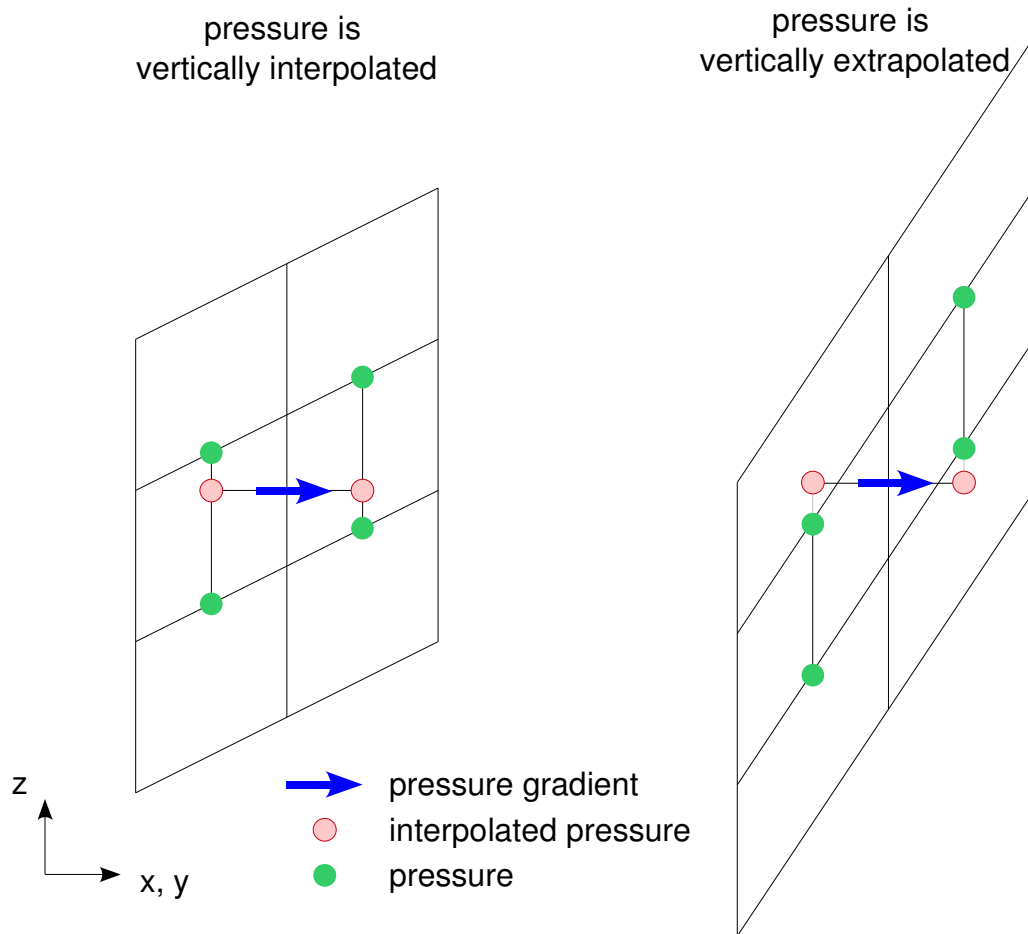


Figure 4.4: Horizontal pressure gradient and hydrostatic consistency

4.1.2. z -coordinate

For model using the z -coordinate, the depth of each model levels depends only on z .

Advantages:

- ▶ simple numerical discretization and visualization and interpretation of model results
- ▶ surface mixed layer can be naturally represented and resolved pressure gradient

Disadvantages:

- ▶ ignore small bottom slope. This leads to problem in representing potential vorticity variations
- ▶ unrealistic mixing for bottom flow (bottom boundary layer)
- ▶ later mixing and advection along constant-density surfaces is cumbersome and need a high number of vertical levels to adequately resolve those processes. These model have in general a unrealistic large cross-isopycnal mixing.

Improvements of the z -coordinate to include a better representation of the bottom topography ([Adcroft et al., 1997](#)):

- ▶ Partial cell approach. The lowest grid cell can have a thickness that is a function of latitude and longitude. The depth of a water column is no longer restricted to finite set of representable water depth.
- ▶ Shaved cells: The bottom grid cell is no longer a cuboid. The depth of all bottom vertexes are allowed to follow the bottom topography. Shaved cells are thus more realistic than partial cells, however they shaved are numerically less efficient than partial cells.

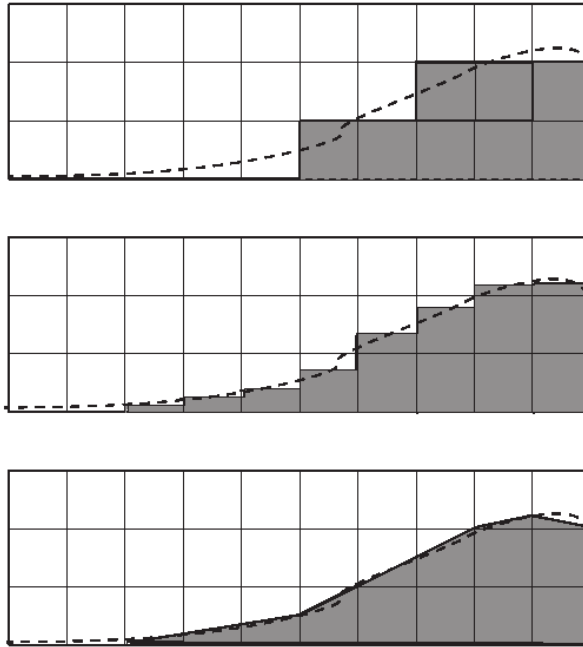


Figure 4.5: Different representation of the ocean floor in z -coordinate ocean models: the traditional full-cell approach (top), partial cells (middle) and shaved cells (bottom). This figure is based on figure 3 from ([Griffies *et al.*, 2000](#)).

4.1.3. σ -coordinate

The σ -coordinate is defined by:

$$\sigma = \frac{z - \eta}{H + \eta} \quad (4.27)$$

$$\begin{array}{llll} \text{for surface} & z = \eta & \longrightarrow & \sigma = 0 \\ \text{for bottom} & z = -H & \longrightarrow & \sigma = -1 \end{array}$$

Advantages:

- ▶ Realistic representation of the ocean bottom
- ▶ Well suited for shallow water
- ▶ all vertical levels are actually used (z-coordinate level run into the bottom floor and the depth of isopycnal become zero is a density level is not present).

Disadvantages:

- ▶ The resolution of the surface mixed layer varies according to the water depth. To resolve the mixing layer in deep, a very fine discretization of σ is necessary. To distribute the surface layer more uniformly, a so-called s -coordinate is introduced ([Song and Haidvogel, 1994](#); [Shchepetkin and McWilliams, 2005](#)) which is defined using the σ -coordinate by:

$$z(x, y, \sigma) = \sigma h_{\min} + C(\sigma) (h(x, y) - h_{\min}) \quad (4.28)$$

where $H(x, y)$ is the depth $C(\sigma)$, a function that defines the vertical grid spacing.

- ▶ It is difficult to align advection and diffusion along inclined density surfaces in the ocean interior

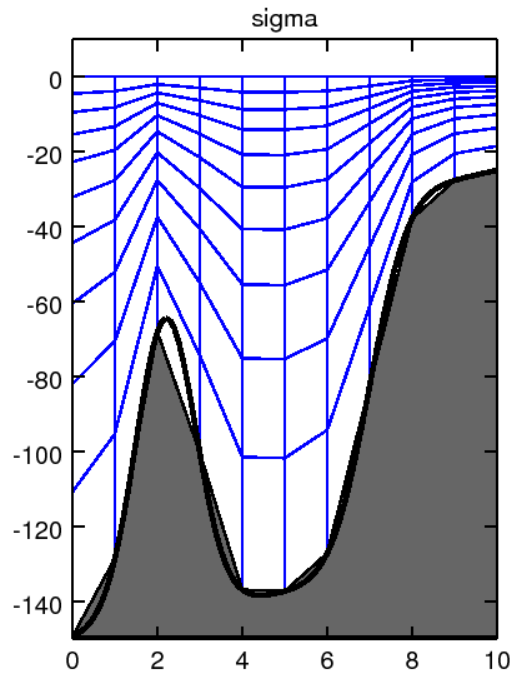
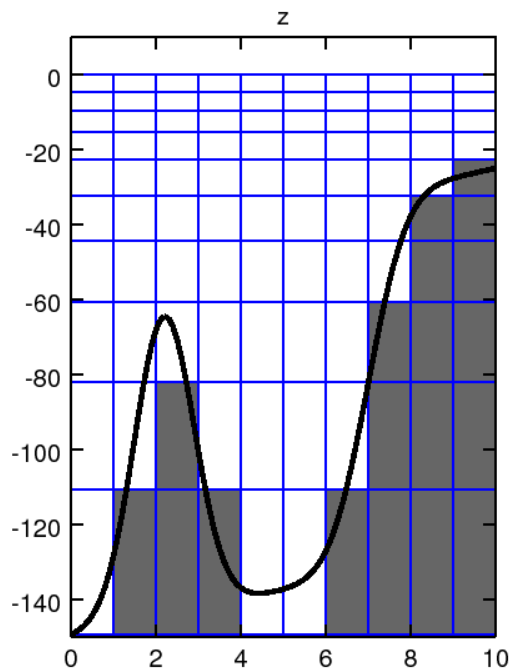


Figure 4.6: z and σ coordinate

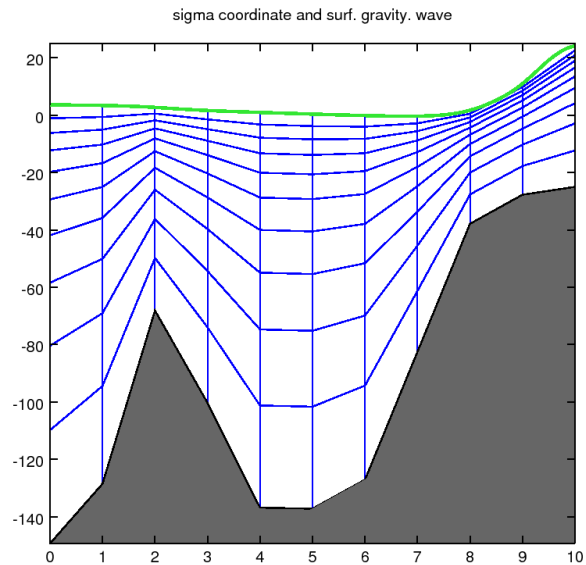


Figure 4.7: Since the σ -levels depend on elevation, the depth vary in time. [Animation of a surface gravity wave and the movement of the coordinate system.](#)

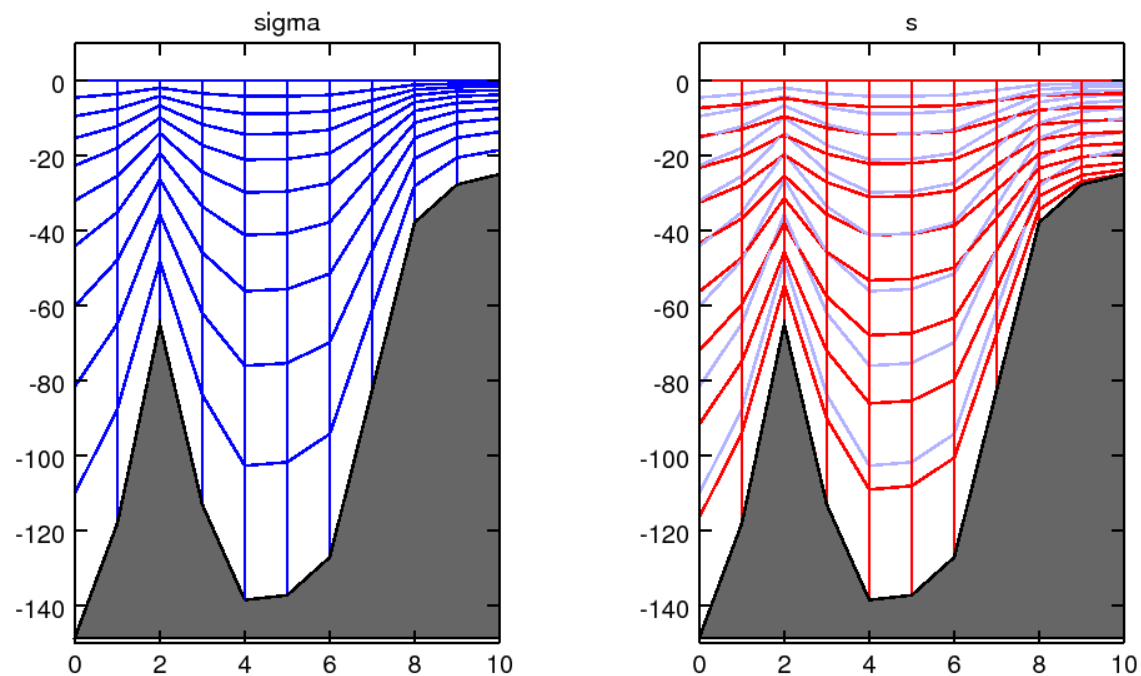


Figure 4.8: σ and s coordinate

- ▶ pressure gradient error: The pressure gradient error is a problem near steep topography, in particular at the shelf break. Smoothing of the bathymetry is often required. The problem can be address with the double-sigma coordinate [Beckers \(1991\)](#): the domain is divided in a upper and lower region at the approximate mean depth of the shelf-break and a sigma coordinate transformation is realized in both regions.
- ▶ Advection and diffusion along constant density surfaces is difficult.

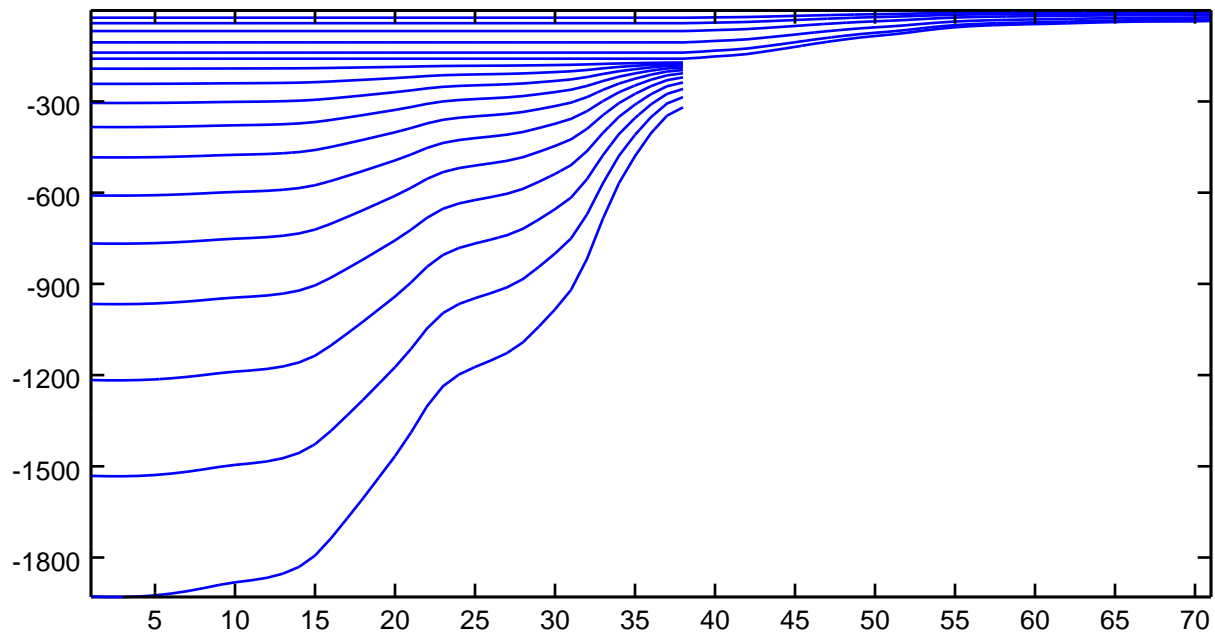


Figure 4.9: The double-sigma coordinate: a first sigma coordinate covers the first 170 m and a second sigma coordinate goes from 170 m down to the ocean floor.

4.1.4. Isopycnals

This coordinate is a close analog to the atmosphere's entropy or potential temperature. The levels are chosen such that the density of each level is a constant.

Advantage:

- ▶ in the ocean interior, tracers have the tendency to move along isopycnal surface. The isopycnal coordinate is thus well suited for this transport.
- ▶ these models follow the bottom topography
- ▶ overflow can be represented more realistically than in z -models
- ▶ horizontal pressure gradient can be easily represented using the Montgomery potential (4.25).

Disadvantages:

- ▶ In unstratified conditions, such as surface and bottom boundary and during deep water formation, density is inappropriate to provide sufficient vertical resolution.
- ▶ The range of densities can vary from sub-basin to another. It might be necessary to add density layer for a small sub-basin which are not used at other places.

Exercise 7:

Compute and plot z , σ and isopycnal levels of a meridional section at 24 deg W in the Atlantic. Try to choose an appropriate resolution which resolves sufficiently the mixed layer. You may use the annual temperature and salinity mean of World Ocean Atlas 2005 to compute the density ([available here](#)).

4.2. Horizontal grid

4.2.1. Structured mesh

4.2.1.1. Cartesian mesh

If only a limited portion of the earth is considered, then the curvature of the earth can be neglected. The differential operator have the simplest possible form in Cartesian coordinates.

Exercise 8:

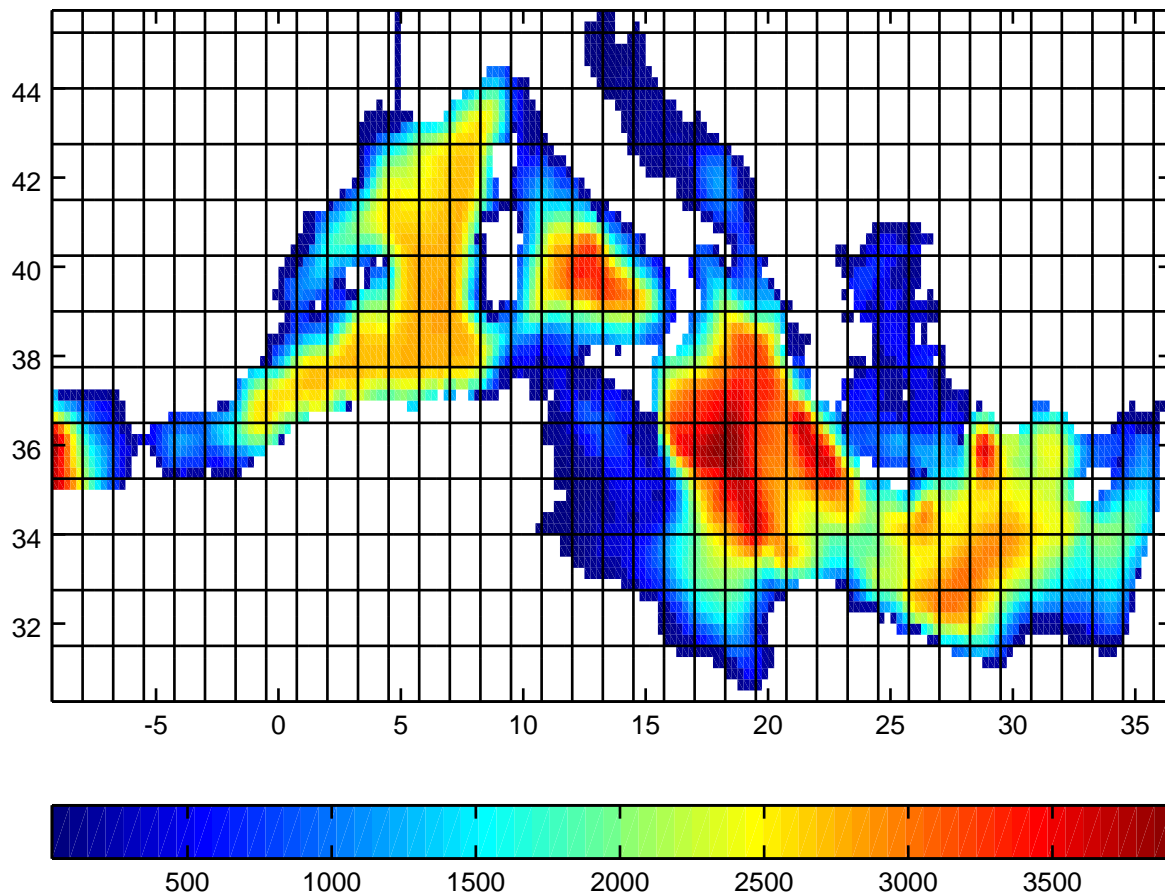
Create a model grid and the model bathymetry of the western part of the Mediterranean at 1/4 degree based on the ETOPO5 bathymetry. Choose an appropriate position of the open boundary.

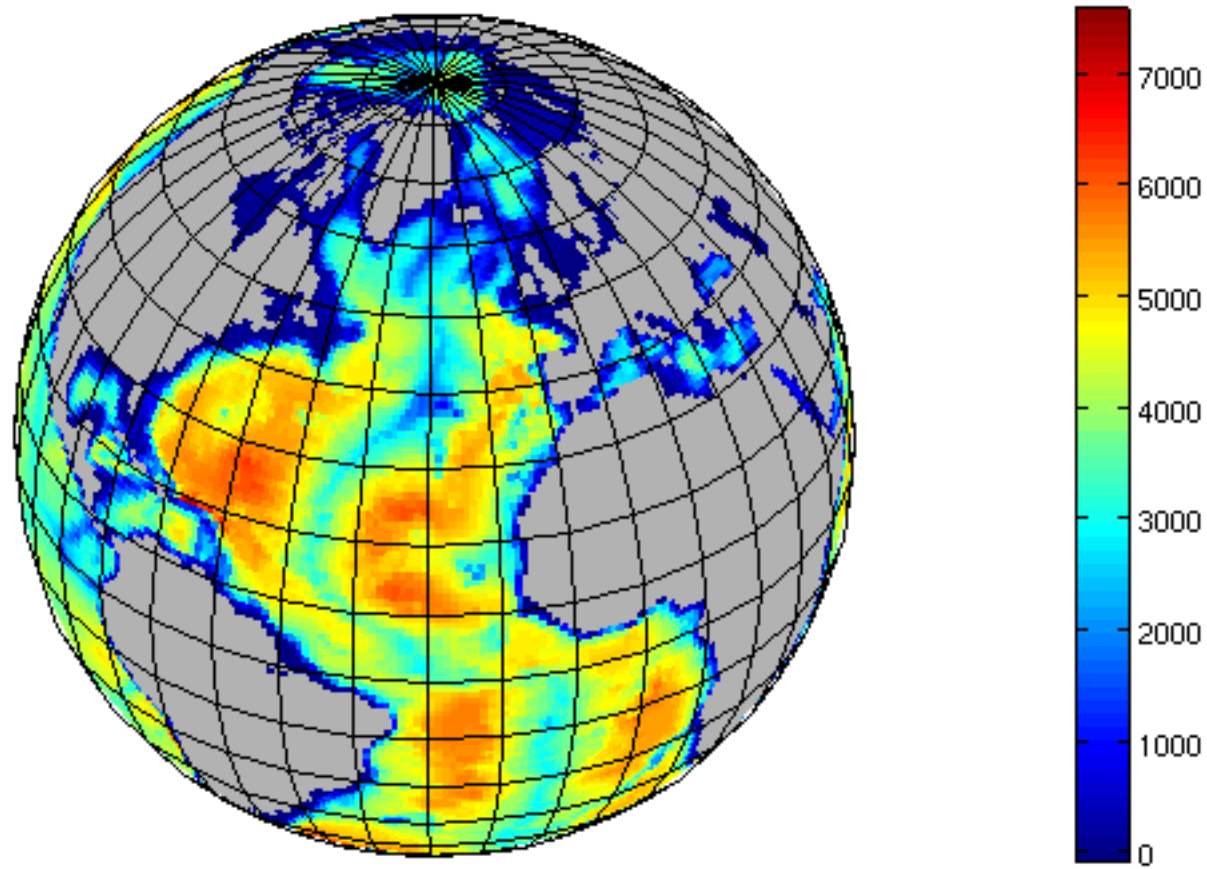
4.2.1.2. Spherical mesh

For a spherical mesh, the domain is discretized along longitude and latitude lines. The longitude increments are normally chosen constant. The latitude increments sometimes also chosen to be constant. In this case, the grid cells corresponds to squares at the equator and become more and more elongated rectangles ones approaches the poles. To obtaine grid cells which corresponds locally to squares everywhere, the latitude increment $\Delta\lambda$ is equal to:

$$\Delta\lambda = \Delta\phi \cos(\lambda) \quad (4.29)$$

where λ is the latitude and $\Delta\phi$ is the longitude increment. Due to the similarity to the Mercator projection, this grid is also called a Mercator grid. In any case, the convergence of the meridians at the poles require a very small time step. This problem can be circumverted by rotating pole to land.





4.2.1.3. Generalized orthogonal mesh

The curvilinear grid is defined as a change of coordinate system:

$$\xi = \xi(x, y) \quad (4.30)$$

$$\eta = \eta(x, y) \quad (4.31)$$

This change of coordinate system is assumed to be invertible:

$$x = x(\xi, \eta) \quad (4.32)$$

$$y = y(\xi, \eta) \quad (4.33)$$

Essential quantities to describe the local characteristics of the curvilinear grids are the scale factors m and n .

$$(ds)_\xi = \frac{1}{m} d\xi \quad (4.34)$$

$$(ds)_\eta = \frac{1}{n} d\eta \quad (4.35)$$

where ds is the distance between two points at constant ξ or at constant η . If (x, y) are the Cartesian coordinates on a plane, the scale factors m and n are:

$$\frac{1}{m^2} = \left(\frac{\partial x}{\partial \xi} \right)^2 + \left(\frac{\partial y}{\partial \xi} \right)^2 \quad (4.36)$$

$$\frac{1}{n^2} = \left(\frac{\partial x}{\partial \eta} \right)^2 + \left(\frac{\partial y}{\partial \eta} \right)^2 \quad (4.37)$$

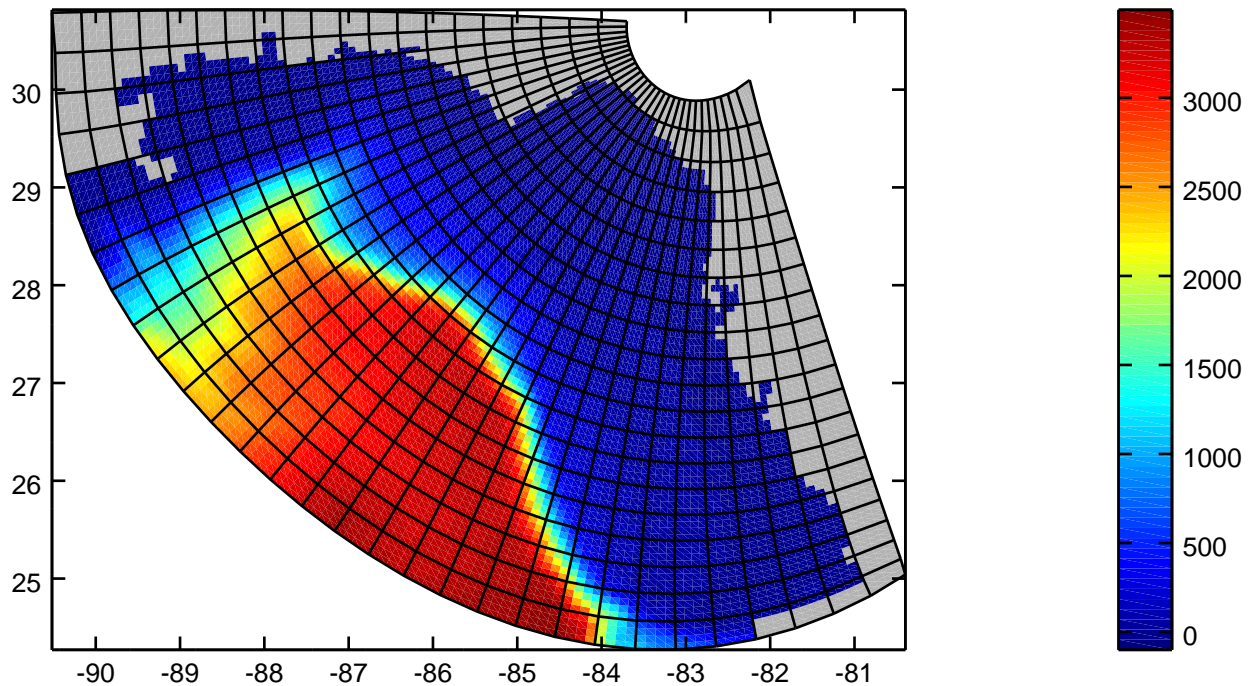


Figure 4.12: Example of a curvilinear mesh of the West Florida Shelf. Only one grid line of 4 is shown.

The derivative in the transformed coordinate system are related to the original derivative by:

$$\frac{\partial}{\partial \xi} = \frac{\partial x}{\partial \xi} \frac{\partial}{\partial x} + \frac{\partial y}{\partial \xi} \frac{\partial}{\partial y} \quad (4.38)$$

$$\frac{\partial}{\partial \eta} = \frac{\partial x}{\partial \eta} \frac{\partial}{\partial x} + \frac{\partial y}{\partial \eta} \frac{\partial}{\partial y} \quad (4.39)$$

Vectors are locally rotated according to:

$$\mathbf{e}_\xi = \frac{\partial x}{\partial \xi} \mathbf{e}_x + \frac{\partial y}{\partial \xi} \mathbf{e}_y \quad (4.40)$$

$$\mathbf{e}_\eta = \frac{\partial x}{\partial \eta} \mathbf{e}_x + \frac{\partial y}{\partial \eta} \mathbf{e}_y \quad (4.41)$$

Those vectors are assumed to be orthogonal:

$$\mathbf{e}_\xi \cdot \mathbf{e}_\eta = 0 = \frac{\partial x}{\partial \xi} \frac{\partial x}{\partial \eta} + \frac{\partial y}{\partial \xi} \frac{\partial y}{\partial \eta} \quad (4.42)$$

Jacobian of the change of coordinate system can be written as:

$$J = \frac{\partial(x, y)}{\partial(\eta, \xi)} = \frac{\partial x}{\partial \eta} \frac{\partial y}{\partial \xi} - \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} = \frac{1}{mn} \quad (4.43)$$

indeed after some calculations one obtains,

$$J^2 = \frac{\partial x^2}{\partial \eta} \frac{\partial y^2}{\partial \xi} - 2 \frac{\partial x}{\partial \eta} \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} \frac{\partial y}{\partial \xi} + \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} \quad (4.44)$$

$$= \frac{\partial x^2}{\partial \eta} \frac{\partial y^2}{\partial \xi} + \frac{\partial x^2}{\partial \eta} \frac{\partial x^2}{\partial \xi} + \frac{\partial y^2}{\partial \eta} \frac{\partial y^2}{\partial \xi} + \frac{\partial x^2}{\partial \xi} \frac{\partial y^2}{\partial \eta} \quad (4.45)$$

$$= \left(\frac{\partial x^2}{\partial \eta} + \frac{\partial y^2}{\partial \eta} \right) \left(\frac{\partial x^2}{\partial \xi} + \frac{\partial y^2}{\partial \xi} \right) \quad (4.46)$$

$$= \frac{1}{m^2 n^2} \quad (4.47)$$

The velocity in the curvilinear coordinate system (u, v) is obtained from the velocity in the Cartesian system by:

$$u = \frac{\partial x}{\partial \xi} v_x + \frac{\partial y}{\partial \xi} v_y \quad (4.48)$$

$$v = \frac{\partial x}{\partial \eta} v_x + \frac{\partial y}{\partial \eta} v_y \quad (4.49)$$

To express the dynamical equations using the variables for the curvilinear system, one need to substitute the variables of the old coordinate system by the transformed one. For the derivative, one obtains,

$$\frac{\partial}{\partial x} = mn \frac{\partial y}{\partial \eta} \frac{\partial}{\partial \xi} - mn \frac{\partial y}{\partial \xi} \frac{\partial}{\partial \eta} \quad (4.50)$$

$$\frac{\partial}{\partial y} = -mn \frac{\partial x}{\partial \eta} \frac{\partial}{\partial \xi} + mn \frac{\partial x}{\partial \xi} \frac{\partial}{\partial \eta} \quad (4.51)$$

and the velocity

$$v_x = mn \frac{\partial y}{\partial \eta} u - mn \frac{\partial y}{\partial \xi} v \quad (4.52)$$

$$v_y = -mn \frac{\partial x}{\partial \eta} u + mn \frac{\partial x}{\partial \xi} v \quad (4.53)$$

For example the advection of a tracer is written as:

$$v_x \frac{\partial}{\partial x} T + v_y \frac{\partial}{\partial y} T = m^2 n^2 \left(\frac{\partial y}{\partial \eta} u - \frac{\partial y}{\partial \xi} v \right) \left(\frac{\partial y}{\partial \eta} \frac{\partial}{\partial \xi} - \frac{\partial y}{\partial \xi} \frac{\partial}{\partial \eta} \right) T \quad (4.54)$$

$$+ m^2 n^2 \left(-\frac{\partial x}{\partial \eta} u + \frac{\partial x}{\partial \xi} v \right) \left(-\frac{\partial x}{\partial \eta} \frac{\partial}{\partial \xi} T + \frac{\partial x}{\partial \xi} \frac{\partial}{\partial \eta} T \right) \quad (4.55)$$

$$= m^2 u \frac{\partial}{\partial \eta} T + n^2 v \frac{\partial}{\partial \xi} T \quad (4.56)$$

Because the transformed coordinate system is locally orthogonal, the differential operator can be written in a compact form similar to the Cartesian system. The essential difference is the appearance of the factor m and n . For example the Laplacian can be written as:

$$\nabla^2 \phi = mn \frac{\partial}{\partial \xi} \left(\frac{m}{n} \frac{\partial}{\partial \xi} \phi \right) + mn \frac{\partial}{\partial \eta} \left(\frac{n}{m} \frac{\partial}{\partial \eta} \phi \right) \quad (4.57)$$

4.2.2. Grid staggering

By placing variables at different location, the accuracy of the discretization scheme can be improved ([Arakawa, 1966](#); [Arakawa and Lamb, 1981](#)).

If variables are necessary on other location, spatial average (which amount to interpolation) is necessary. Since spatial averaging smoothes the solution, it introduces numerical diffusion. Some scheme with spatial have also

numerical modes with a structure of a check-board. In the averaged field, the high frequency structure disappears and there is thus no dynamical feedback to dissipate the check-board pattern. For example, the 1 dimensional shallow water equations on a unstaggered grid:

$$\frac{\eta^{n+1} - \eta}{\Delta t} = -H \frac{u_{i+1} - u_{i-1}}{2\Delta x} \quad (4.58)$$

$$\frac{u^{n+1} - u}{\Delta t} = -g \frac{\eta_{i+1}^{n+1} - \eta_{i-1}^{n+1}}{2\Delta x} \quad (4.59)$$

The rhs are finite differences over $2\Delta x$. These terms can also be viewed as differences over Δx of averaged values. These admit the following as a stationary solution:

$$\eta = Ae^{i\pi i} \quad (4.60)$$

$$u = Be^{i\pi i} \quad (4.61)$$

The sign of the elevation and velocity changes every grid point. In general a numerical scheme and a grid is sought which minimizes the need of spatial averaging.

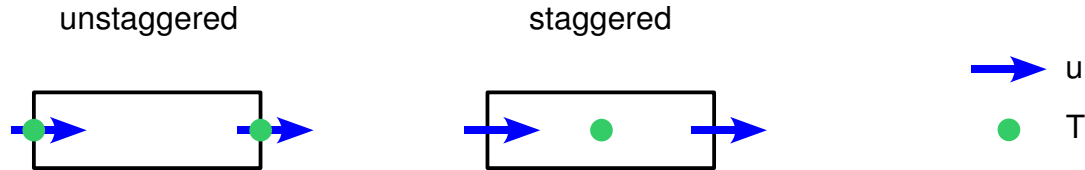


Figure 4.13: Staggering of variables in 1 dimensions

Exercise 9:

Discretize the 1D linear shallow water equations on a staggered grid with a wall at $x = 0$ and $x = L$. Determine and stability criterion and solve the discretized equations numerically.

$$\frac{\partial h}{\partial t} = -\frac{\partial \bar{h} u}{\partial x} \quad (4.62)$$

$$\frac{\partial u}{\partial t} = -g \frac{\partial h}{\partial x} \quad (4.63)$$

where \bar{h} is 30 m. The domain is 100 km (L) long and discretized with 100 grid points. The average depth (\bar{h}) is 30 m and the initial h given by:

$$h(x) = \bar{h} + a \exp(-(x/b)^2) \quad (4.64)$$

where $a = 2$ m and $b = 5$ km. The fluid is initially at rest.

Arakawa (1966) introduced several ways to place the variables of the primitive equations on a two dimensional grids (figure 4.2.2. The variables u and v corresponds to the horizontal velocity and tracer flux components. T are the tracers (temperature, salinity, turbulent kinetic energy, concentration of biological and chemical tracers) and sea surface height. variable ϕ represent the location of barotropic stream function.

Most common grids are B and C. Numerous authors have compared the merit of the different grid under different conditions:

- ▶ in B grid, the Coriolis force can be easily represented while the C grid requires spatial averaging for this term. Geostrophy is thus well represented on a B grid.
- ▶ at coarse resolution inertia-gravity waves are better represented on a B than a C grid, at fine resolution the C grid is better than the B grid (Arakawa and Lamb, 1977; Hsieh *et al.*, 1983; Beckers and Deleersnijder, 1993).
- ▶ B grid is better for Rossby-waves (resolved and under-resolved) because of their superior representation of the Coriolis force (Dukowicz, 1995).
- ▶ C grid has a better representation of the Energy cascade with baroclinic eddies (Janjić, 1984).

4.2.3. Unstructured mesh

Most ocean models use currently structured grids. However, recently numerical ocean model using unstructured grids are developed. With unstructured meshes the smallest resolved scale varies in general of the model domain.

Advantages:

- ▶ very flexible to represent complex coastline and other isobaths
- ▶ increased resolution in zones of interest
- ▶ finite volume or finite elements

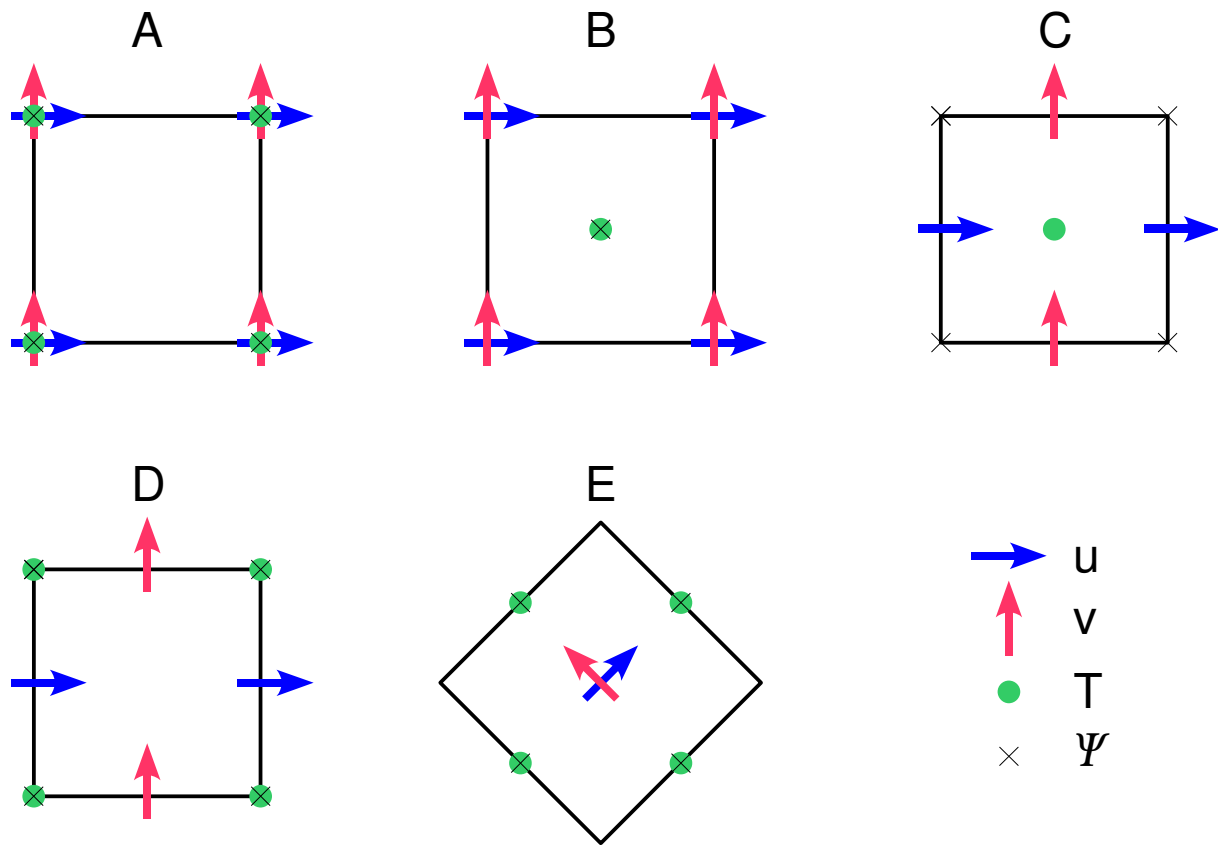


Figure 4.14: Location of variables in staggered Arakawa A, B, C, D and E grid.

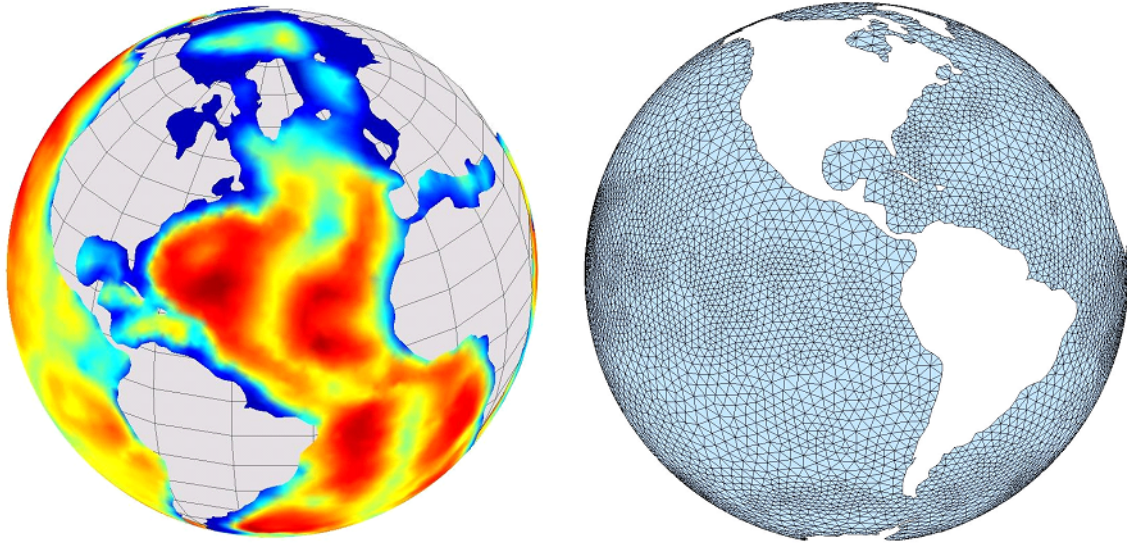


Figure 4.15: Example of an unstructured mesh. Image from Applied Mechanics Division at UCL.

Disadvantages:

- ▶ Difficulty to represent the geostrophic balance correctly
- ▶ Unphysical wave scattering when the resolution changes abruptly

Exercise 10:

Consider the linear shallow water equation in a 1d-domain bounded by two coastal walls.

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial U}{\partial x} \quad (4.65)$$

$$\frac{\partial U}{\partial t} = -gh \frac{\partial \zeta}{\partial x} \quad (4.66)$$

Between 0 and L_1 the domain is discretized with a resolution $\Delta x = 1$ km and between L_1 and L_2 with a coarser resolution of $r\Delta x$. The initial surface elevation is given by:

$$\zeta(x) = A \exp(-x^2/L^2) \quad (4.67)$$

where $h = 500$ m, $L_1 = 200$ km, $L_2 = 400$ km, $\Delta t = 10$ s and $A = 1$ m. The velocity is initially zero. Integrate the equation forward until the (main) perturbation reached $x = 300$ km. At this moment integrate the wave energy over the first half of the domain:

$$E = \int_0^{L_1} g\zeta^2 + \frac{U^2}{h} dx \quad (4.68)$$

- ▶ Carry out the experiment for different value for $r = 2, 3$ and 5 and $L = 4$ km, 10 km and 20 km.
- ▶ Describe what happen at $x = L_1$ and why.
- ▶ Explain the dependence of E on r and L .

4.3. Time stepping

Time stepping is the temporal equivalent of the spatial coordinates and grid staggering. The primitive equations admit a range of wave-like solution with a broad spectrum of possible propagation speed. For example surface gravity waves have a propagation speed of \sqrt{gH} (about 100 m/s for 1000 m deep ocean) and Rossby waves ($c = -\beta/k^2$) These waves are produced in barotropic, baroclinic, and thermodynamic adjustment processes and the time-scale of these processes is related to the corresponding wave propagation speed. Each of those wave like solution introduce a stability criterion which is increasingly severe for faster waves. To reduce computational cost, a different time steps for various equations is often used ([Bryan, 1969b,a](#)). The barotropic variables (surface elevation and depth averaged current), the baroclinic velocity and the tracer can thus be integrated with different time steps.

Exercise 11:

Discretize the linear two-layer model on a staggered grid with a wall at $x = 0$ and $x = L$.

$$\frac{\partial h_k}{\partial t} = -\frac{\partial \overline{h_k} u_k}{\partial x} \quad (4.69)$$

$$\frac{\partial u_k}{\partial t} = -\frac{1}{\rho_k} \frac{\partial p_k}{\partial x} \quad (4.70)$$

for $k = 1$ and $k = 2$ and where the pressure for each level is given by:

$$p_1 = \rho_1 g (h_1 + h_2 - H) \quad (4.71)$$

$$p_2 = \rho_1 (g h_1 + (g + g') (h_2 - H)) \quad (4.72)$$

where $H = \overline{h_1} + \overline{h_2}$ and $g' = \frac{\rho_2 - \rho_1}{\rho_1} g$. Initially the velocity is zero and h_1 and h_2 are given by:

$$h_1 = A_1 \exp(-x^2/L'^2) + \overline{h_1} \quad (4.73)$$

$$h_2 = A_2 \exp(-x^2/L'^2) + \overline{h_2} \quad (4.74)$$

where $A_1 = 40 \text{ m}$, $A_2 = -30 \text{ m}$, $L = 100 \text{ km}$, $L' = 20 \text{ km}$, $\rho_1 = 1020 \text{ kg m}^{-3}$, $\rho_2 = 1035 \text{ kg m}^{-3}$, and $\overline{h_1} = \overline{h_2} = 50 \text{ m}$.

Hint: Use the pressure at time step $n + 1$ to compute the velocity at this time step.

- ▶ Draw h_2 as a function of time and space and describe it
- ▶ Determine graphically the propagation speeds and compare it to the theoretical values
- ▶ Repeat the simulation with $\rho_1 = 1000 \text{ kg m}^{-3}$, and $\rho_1 = 1035 \text{ kg m}^{-3}$ and explain the changes relative to the first simulation in physical terms.

The dynamics of a stratified fluid can be decomposed vertically in orthogonal eigenmodes (Gill, 1982). The gravest mode is called the barotropic mode and all higher modes are called baroclinic modes. In practice the barotropic mode is obtained by depth averaging.

The barotropic mode contains the fast moving surface gravity waves and the slow moving planetary and topographic Rossby waves (or the geostrophic equilibrium for a flow ocean with constant f). The fast moving surface gravity waves can be handled in different ways:

- ▶ Explicit free surface models: Barotropic shallow water equations are solved with a small time step (according to the CFL condition for the surface gravity waves)
- ▶ The surface gravity waves are removed altogether with the rigid lid approximation. However, this means that:
 - need to solve an elliptic problem for the stream function ϕ or surface pressure. Direct solution of the elliptic problem is only feasible for smaller problems. For an iterative elliptic solver, it is difficult to achieve convergence in a reasonable number of iterations.
 - Not possible to add/remove water (due to river or precipitation/evaporation). Freshwater flux is treated as a virtual salinity flux.
 - rigid lid approximation modifies also the dispersion relation of Rossby waves
 - no tides

The rigid lid method is becoming obsolete even for ocean climate modeling (Griffies *et al.*, 2000).

- ▶ Implicit free surface models: implicit methods admit large time-step, but not resolving the barotropic dynamics. Still need to solve an elliptic problem.

Only local boundary conditions are needed for explicit free surface models while implicit and rigid lid approaches require non-local boundary conditions. The free surface is more easier to implement for σ and isopycnal models than for z -models, since the model cells in a z -grid might be partially empty. Also free surface models are more efficient on a parallel computer.

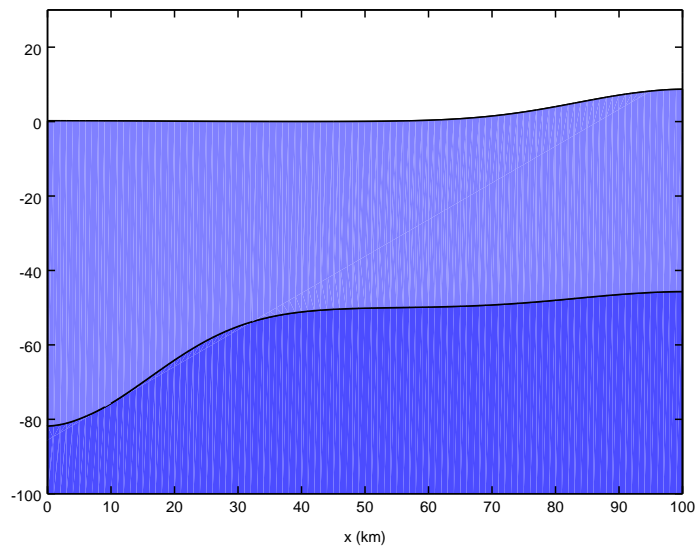


Figure 4.16: Snapshot of the simulation

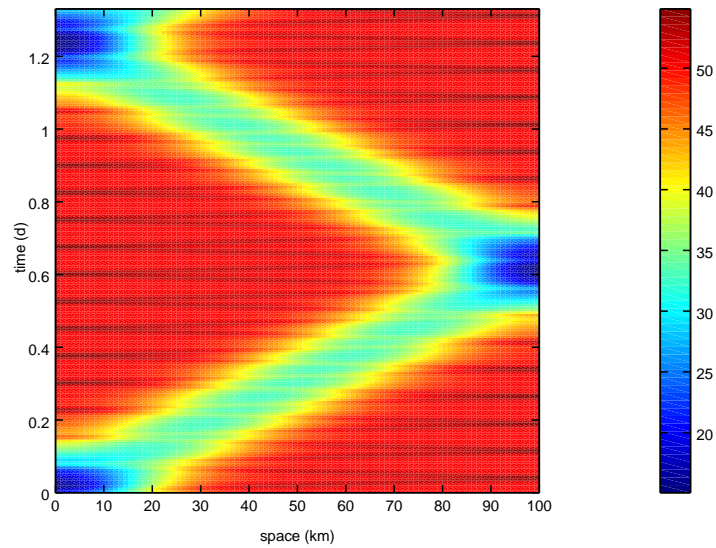


Figure 4.17: Hovmöller Diagram of h_2 showing the propagation of the external and internal waves

Chapter 5

Solving model equations on a grid

Contents

5.1	Finite difference	72
5.2	Finite volume	75
5.3	Finite elements	76
5.4	Spectral methods	79

The purpose of this chapter is to review main methods to discretize a partial differential equations to resolve it numerically. The methods are illustrated with the 1D advection equations:

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0 \tag{5.1}$$

All discretization method take only into account a certain range of scales. Larger scale should be taken into account as boundary conditions and smaller scales have to be parameterized.

5.1. Finite difference

The continuous function is function u is sampled at discrete locations $(x_j, t_n) = (\Delta x j, \Delta t n)$ where j and n are integers. The derivative of the partial differential equations are approximated by finite differences:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = 0 \quad (5.2)$$

The dispersion relation of this numerical scheme is determined by assuming a wave-like solution:

$$u_j^n = A \exp(i(k\Delta x j - \omega \Delta t n)) \quad (5.3)$$

which leads to:

$$\exp(-i\omega \Delta t) + i \frac{c\Delta t}{\Delta x} \sin(k\Delta x) = 0 \quad (5.4)$$

The dispersion relation is thus,

$$\omega = \tan^{-1}(C \sin(k\Delta x)) + \frac{i}{2} \ln(1 + C^2 \sin^2(k\Delta x)) \quad (5.5)$$

where $C = \frac{c\Delta t}{\Delta x}$. The angular frequency has an imaginary part which is always larger than 1. The scheme is thus unconditionally unstable. In general, there is no guarantee that a partial differential equation solved by the finite difference approach is stable. The only way to find out is by applying a stability analysis.

In equations (5.2), the spatial derivative was treated differently than the temporal derivative. The leap-frog scheme discretizes both derivatives in a symmetric way.

$$\frac{u_j^{n+1} - u_j^{n-1}}{2\Delta t} + c \frac{u_{j-1}^n - u_{j+1}^n}{2\Delta x} = 0 \quad (5.6)$$

It can be shown that this scheme is conditionally stable if $C < 1$. However, this scheme suffers from other issues:

- ▶ it requires two initial conditions at two successive time steps
- ▶ it requires an unphysical downstream boundary condition
- ▶ it admits a spurious numerical mode as solution

Exercise 12:

An additional relaxation term is often included in numerical models to avoid unrealistic drifts due to e.g. systematic error in the heat flux. This term “nudges” the model towards a reference state such as a climatology or observations. Study the evolution equations of temperature where only this relaxation term is present:

$$\frac{\partial T}{\partial t} = \frac{1}{\tau} (T^e - T) \quad (5.7)$$

where $\tau = 1$ month and $T(0) = 10^\circ \text{C}$.

- ▶ For $T^e = 20^\circ \text{C}$. Solve this equation analytically and numerically using an Euler-forward scheme. Integrate this equation for 3 years with a suitably chosen time step.
- ▶ For $T^e = A + B \cos(\omega t)$ where $A = 20^\circ \text{C}$, $B = 5^\circ \text{C}$ and the period of the cosine is one year. Solve this equation numerically and discuss the phase difference between T and T^e .
- ▶ Can you think of other processes (possibly in other fields) which are similar to the nudging terms?

Exercise 13:

Using the sea-surface height in NetCDF file `ssh_20071127.nc` compute the corresponding surface geostrophic current by finite difference.

5.2. Finite volume

The partial differential equation is written in flux form:

$$\frac{\partial u}{\partial t} + \frac{\partial q}{\partial x} = 0 \quad (5.8)$$

$$q = cu \quad (5.9)$$

The partial differential equations are integrated over a finite volume:

$$\int_{(j-1/2)\Delta x}^{(j+1/2)\Delta x} \frac{\partial u}{\partial t} + \frac{\partial q}{\partial x} dx = 0 \quad (5.10)$$

$$\Delta x \frac{\partial u_j}{\partial t} + q_{j+1/2} - q_{j-1/2} = 0 \quad (5.11)$$

where u_j represents the average over the grid cell and $q_{j+1/2}$ is the flux at the interface:

$$u_j = \frac{1}{\Delta x} \int_{(j-1/2)\Delta x}^{(j+1/2)\Delta x} u dx \quad (5.12)$$

$$q_{j+1/2} = q((j+1/2)\Delta x) \quad (5.13)$$

For the upwind-scheme, the flux is given by:

$$q_{j+1/2} = cu_j \quad \text{if } c \geq 0 \quad (5.14)$$

$$= cu_{j-1} \quad \text{if } c < 0 \quad (5.15)$$

The time derivative can be discretized by using an Euler forward step:

$$u_j^{n+1} = u_j^n + \frac{\Delta t}{\Delta x} (q_{j+1/2} - q_{j-1/2}) \quad (5.16)$$

This numerical scheme could also be obtained by the finite difference approach. For simple partial differential equations, it is common that the finite difference approach and the finite volume approach yield the same numerical scheme, but for more complex partial differential equations this is in general not the case.

This method does not require that the finite volumes are rectangular. Indeed, some numerical ocean models apply the finite volume approach to unstructured triangular meshes.

5.3. Finite elements

The solution is projected into a series of (non-orthogonal) functions which are only non-zero over a given element

$$u^* = \sum_{i=0}^N u_i(t) \phi_i(x) \quad (5.17)$$

The basis function ϕ_i are defined by:

$$\phi_i = \frac{1}{h} (x - (i-1)h) \quad (i-1)h \leq x \leq ih \quad (5.18)$$

$$= \frac{1}{h} ((i+1)h - x) \quad ih \leq x \leq (i+1)h \quad (5.19)$$

$$= 0 \quad \text{otherwise} \quad (5.20)$$

where $i = 1, \dots, N$.

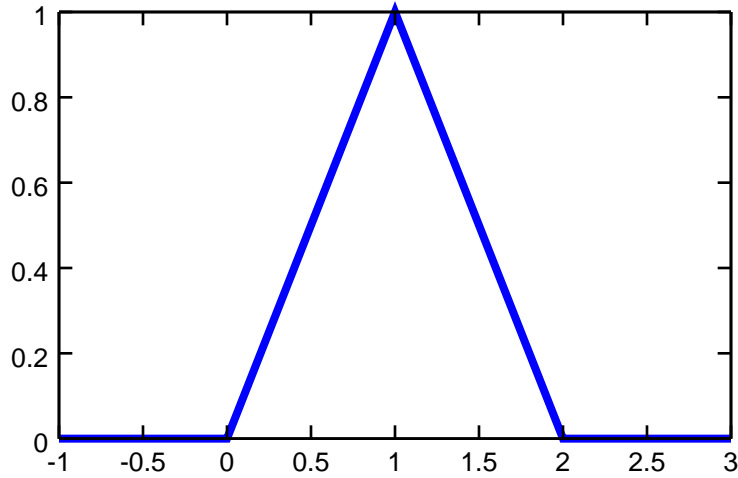


Figure 5.1: The shape of function ϕ_i for $i = 1$ and $h = 1$

In general, there are no coefficients $u_i(t)$ such that the function u^* satisfies equations (5.1) exactly. There will be a residual, noted r .

$$\frac{\partial u^*}{\partial t} + c \frac{\partial u^*}{\partial x} = r \quad (5.21)$$

However, we want that the residual should be “as small as possible”. The coefficients $u_i(t)$ are determined

such that the residual is orthogonal to a set of test functions:

$$\int r w_i dx = 0 \quad \text{for } i = 1, \dots, N \quad (5.22)$$

For the Galerkin method, the basis function themselves are chosen as test functions: $\phi_i = w_i$. For the 1d-advection case, it follows that:

$$\sum_{j=1}^N \frac{du_j}{dt} \int \phi_i \phi_j dx + u_j \int \phi_i \frac{d\phi_j}{dx} dx = 0 \quad (5.23)$$

After evaluating the integrals, one obtains:

$$\frac{1}{6} \frac{du_{i-1}}{dt} + \frac{2}{3} \frac{du_i}{dt} + \frac{1}{6} \frac{du_{i+1}}{dt} = -\frac{c}{2h} (u_{i+1} - u_{i-1}) \quad (5.24)$$

Semi-discrete equation since the time derivative is not yet discretized. This equation is implicit. For the finite element method in general a large but sparse system must be solved.

For the present 1d-advection case, a tri-diagonal system for which efficient solver exists such as the Thomas algorithm.

$$u_j = A \exp(i(jkh - \omega t)) \quad (5.25)$$

Dispersion relation:

$$\omega = \frac{3}{2 + \cos(hk)} \frac{\sin(kh)}{h} \quad (5.26)$$

is a approximation of the true dispersion relation to the fourth order in h.

5.4. Spectral methods

As previously, the solution is projected into a series of function.

$$u^* = \sum_{j=0}^N u_j(t) \phi_j(x) \quad (5.27)$$

But now, the basis function are chosen orthogonal. The choice of the orthogonal function is often determined by the geometry of domain. For the 1d-advection problem, we choose Fourier modes:

$$\phi_j = \exp(ik_j x) \quad (5.28)$$

By substitution, u^* in the 1d-advection equation, one obtains:

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = \quad (5.29)$$

$$\sum_{j=0}^N \frac{du_j}{dt} \phi_j + i c k_j u_j \phi_j = 0 \quad (5.30)$$

Since the basis function ϕ_j are orthogonal,

$$\frac{du_j}{dt} + i c k_j u_j = 0 \quad (5.31)$$

Spectral method has thus transformed the partial differential equations (5.1) into a set of trivial and decoupled ordinary differential equations. The dispersion relation for the semi-discrete equation obtained by the spectral method is thus identical to the dispersion relation of the continuous equations.

The spectral method is often used for global atmospheric circulation model where spherical harmonics are used as basis functions. However, it is difficult to apply the spectral method to the ocean because of the complex geometry of the domain.

Chapter 6

Sub-grid scale processes

Contents

6.1	Surface mixed layer	81
6.2	Bottom boundary layer	82
6.3	Horizontal sub-grid scale process	82

6.1. Surface mixed layer

Bulk mixed layer, assumes a perfectly mixed layer. All variables are perfectly uniform over this layer. No vertical structure is a problem where this layer extends to over hundred meters (e.g. subpolar regions).

Continuously formulated surface mixed layer:

- ▶ K-Profile Parameterization (KPP) ([Large et al., 1994](#)): diffusibility is based on the Richardson number, includes non-local mixing processes
- ▶ Mellor-Yamada ([Mellor and Yamada, 1982](#))
- ▶ k - ϵ . Additional equations for turbulent kinetic energy and turbulence dissipation or length-scale

z and s -coordinate allow a good representation of the surface mixed layer. For hydrostatic models, the turbulence scheme must handle also if hydrostatically unstable, then large diffusibility to parameterize convection (since hydrostatic approximation)

6.2. Bottom boundary layer

σ coordinate allow a good representation of the bottom boundary layer since the flow is constrained by the bottom topography.

Width of several tens of meters depending on the roughness for the sea floor and the strength of the currents

Overflow are currents following the bottom topography. Their water is in general much denser than the surrounding waters. Overflows are problematic in z -coordinates since topography is approximated by steps.

6.3. Horizontal sub-grid scale process

While several parametrization exists for the vertical sub-grid scale processes, only a few and simple parametrizations are used for the horizontal sub-grid scale processes. The simplest form is the horizontal diffusion:

$$D(c) = \frac{\partial}{\partial x} \left(A \frac{\partial c}{\partial x} \right) + \frac{\partial}{\partial y} \left(A \frac{\partial c}{\partial y} \right) \quad (6.1)$$

where A is either constant or depends on the characteristics of the flow (e.g [Smagorinsky, 1963](#)). Most numerical ocean model require a horizontal diffusion to dissipate energy at small grid scale.

Exercise 14: Refraction of surface gravity waves

As in optics, the refraction of ocean surface gravity waves obeys the Snell-Descartes law. The phase velocity of the surface gravity waves is a function of depth. The refraction of surface gravity waves is applied to the propagation of a tsunami in the Pacific Ocean. The bottom topography is approximated by a series of contours: each contours represents a discontinuity of the bottom depth and between two successive contours, the depth is assumed constant. The starting point of the tsunami wave is 38.297° N, 142.372° E and 16 initial propagation directions (or more) spanning the full circle are used. For each initial direction, the path and the time when the tsunami reaches the shoreline of the Pacific Ocean are modeled. Results will be discussed and compared to the NOAA tsunami simulation.

Exercise 15: Lagrangian surface drift

A 100 surface drifters are released in the Pacific Ocean. The position of each of those drifters can be modeled by using the surface velocity \mathbf{v} which is a function of the position \mathbf{x} and time t .

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{v}(\mathbf{x}, t) \quad (6.2)$$

Initially the drifters are located on a regular 10×10 grid with a grid spacing of 1 km. The student can choose the precise positioning of this grid and the starting time. It is advised to use the 4th order Runge–Kutta discretization scheme. For the surface velocity fields, the student can either use the results of the global HYCOM model (<http://hycom.org/dataserver/>, <http://hycom.org/dataserver/glb-analysis/expt-90pt9>) or other model results. The simulation is stopped after 60 days. Results will be discussed in connection with the general circulation.

Chapter 7

Programming aspects

Contents

7.1	Programming languages	85
7.2	Elements of a programming language	86
7.2.1	Elementary types	86
7.2.2	Arrays and structures	87
7.2.3	Statements and commands	88
7.2.4	Subroutines and functions	89
7.3	General structure of an ocean model	93

7.1. Programming languages

There are two general approaches to implement programming languages:

- ▶ Interpretation: An interpreter takes the program in some language, and performs the actions written in that language on some machine.
- ▶ Compilation: A compiler translates the a program into some other language, which is in general machine code that a computer can execute directly.

Fortran, C and C++ are examples of languages which are compiled to machine code. Interpreted languages are for example Octave/Matlab, Python and Shell scripts. Compiled languages require the declaration of types of a variables. It is thus easier to develop programs written in Interpreted languages. However, programs written in compiled languages are generally faster than programs in interpreted languages. Most ocean models are written in Fortran. But interpreted languages are often used for preparing the models fields and post-processing the results.

7.2. Elements of a programming language

7.2.1. Elementary types

	Fortran	Matlab/Octave
boolean (true or false)	logical	logical
integer (whole number)	integer(1), integer(2), integer(4), integer(8)	int8, int16, int32, int64
unsigned integer (positive whole number)	not available	uint8, uint16, uint32, uint64
real (real number)	real(4), real(8)	single, double
characters and string	character(length)	char
variable declaration	real (4) :: variable	not needed

7.2.2. Arrays and structures

	Fortran	Matlab/Octave
arrays (collection of variables of the same type accessed by an index)	<code>real(4) :: array(10,20)</code>	<code>array = zeros(10,20)</code>
structure (collection of variables of the different type accessed by their name)	<pre>! definition of type type type_name ! fields, e.g.: real(4) :: fieldname end type [type_name] ! declaration of ! variable type(type_name) :: s ! access s%fieldname</pre>	<pre>% definiton s.fieldname = value; % access s.fieldname</pre>
cell arrays (collection of variables of the different types accessed by an index)	not available	<pre>array{1} = value1; array{2} = value2;</pre>

7.2.3. Statements and commands

	Fortran	Matlab/Octave
conditions	<pre>if (condition) then ! do this if condition ! is true else ! otherwise this end if</pre>	<pre>if condition % do this if condition % is true else % otherwise this end</pre>
loops using an iterator	<pre>do i=imin ,imax ,step ! do something end do</pre>	<pre>for i=imin:step:imax % do something end</pre>
loops with stop conditions	<pre>while (condition) do ! do something while ! condition is true end do</pre>	<pre>while condition % do something while % condition is true end</pre>
terminate loop prematurely	<pre>exit</pre>	<pre>break</pre>

7.2.4. Subroutines and functions

	Fortran	Matlab/Octave
main program	<pre>program main ! do some thing end program main</pre>	Commands can be regrouped in a file (script). The script can be called using the file name (without the extension ".m")
subroutine (block of code)	<pre>subroutine sub(param1, param2) ! do some thing end subroutine sub ! call a subroutine call fun(p1,p2)</pre>	see functions

<p>function (block of code with return value)</p>	<p>A function can have only a single return value</p> <pre> real function fun(param) ! do some thing fun = ... end function fun ! call a function r = fun(p) </pre>	<p>A function can have multiple return values and it must be saved in a file with the same name (plus extension ".m")</p> <pre> function [r] = fun(p) % do some thing r = ... end % call a function r = fun(p) </pre>
<p>parameters of function and sub-routines</p>	<p>Type of parameters have to be declared after the subroutine or function statement. Parameters are passed by reference (i.e. modifications will also affect the corresponding parameter from the calling level).</p> <pre> real function fun(param) integer :: param ! do some thing end function fun </pre>	<p>Type of parameters are not declared. Parameters are passed by value (i.e. modifications will not affect the corresponding parameter from the calling level).</p>

Exercise 16:

Programming exercise:

- ▶ 1D-diffusion equation (for $i = 1, \dots, N$)

$$c_i^{(n+1)} = c_i^{(n)} + \frac{\Delta t}{\Delta x} (F_{i+1} - F_i) \quad (7.1)$$

$$F_i = \frac{\kappa}{\Delta x} (c_i^{(n)} - c_{i-1}^{(n)}) \quad (7.2)$$

with closed and periodic boundary conditions. Initially all values of c are zero except one (at the center or near the boundary).

- ▶ 2D-diffusion equation. Generalize previous equations to 2D and implement it.

7.3. General structure of an ocean model

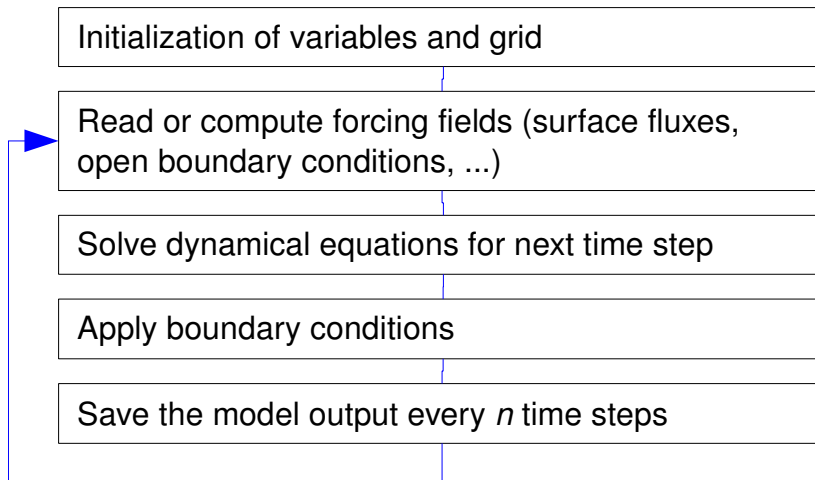


Figure 7.1: Different parts of a numerical ocean model

Chapter 8

Data assimilation

Go to http://modb.oce.ulg.ac.be/mediawiki/index.php/Introduction_to_data_assimilation

Appendix A

Transformation of coordinates

In order to solve a partial differential equations analytically, we are free to chose the coordinate system which suites best the geometry of the problem. Also for numerical problems, such transformations are interesting since it may help the discretizations of the model domains (for example a discretization which follows the bottom topography) or it may reduce discretization error when the coordinate system is chosen such to follow variations of a given property (for example density in isopycnal coordinates).

Change of the coordinates (x_1, x_2, \dots, x_n) to the new coordinate systems $(x'_1, x'_2, \dots, x'_n)$

$$x_1 = x_1(x'_1, x'_2, \dots, x'_n) \tag{A.1}$$

$$x_2 = x_2(x'_1, x'_2, \dots, x'_n) \tag{A.2}$$

$$\vdots$$

$$x_n = x_n(x'_1, x'_2, \dots, x'_n) \tag{A.3}$$

This transformations is assumed to be invertible. The coordinates $(x'_1, x'_2, \dots, x'_n)$ can also be expressed in terms of (x_1, x_2, \dots, x_n) :

$$x'_1 = x'_1(x_1, x_2, \dots, x_n) \quad (\text{A.4})$$

$$x'_2 = x'_2(x_1, x_2, \dots, x_n) \quad (\text{A.5})$$

$$\vdots$$

$$x'_n = x'_n(x_1, x_2, \dots, x_n) \quad (\text{A.6})$$

Any function f of the old coordinate (x_1, x_2, \dots, x_n) can be transformed into the new coordinate systems by substituting (x_1, x_2, \dots, x_n) :

$$f(x_1, x_2, \dots, x_n) = f(x_1(x'_1, x'_2, \dots, x'_n), \dots, x_n(x'_1, x'_2, \dots, x'_n)) \quad (\text{A.7})$$

$$\frac{\partial}{\partial x'_i} f(x_1(x'_1, x'_2, \dots, x'_n), \dots, x_n(x'_1, x'_2, \dots, x'_n)) = \frac{\partial x_j}{\partial x'_i} \frac{\partial}{\partial x_j} f(x_1, x_2, \dots, x_n) \quad (\text{A.8})$$

or in matrix form

$$\begin{pmatrix} \frac{\partial}{\partial x'_1} \\ \frac{\partial}{\partial x'_2} \\ \vdots \\ \frac{\partial}{\partial x'_n} \end{pmatrix} = \begin{pmatrix} \frac{\partial x_1}{\partial x'_1} & \frac{\partial x_2}{\partial x'_1} & \cdots & \frac{\partial x_n}{\partial x'_1} \\ \frac{\partial x_1}{\partial x'_2} & \frac{\partial x_2}{\partial x'_2} & \cdots & \frac{\partial x_n}{\partial x'_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_1}{\partial x'_n} & \frac{\partial x_2}{\partial x'_n} & \cdots & \frac{\partial x_n}{\partial x'_n} \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{pmatrix} \quad (\text{A.9})$$

The matrix in the previous equations is also written as:

$$\mathbf{M} = \begin{pmatrix} \frac{\partial x_1}{\partial x'_1} & \frac{\partial x_2}{\partial x'_1} & \cdots & \frac{\partial x_n}{\partial x'_1} \\ \frac{\partial x_1}{\partial x'_2} & \frac{\partial x_2}{\partial x'_2} & \cdots & \frac{\partial x_n}{\partial x'_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_1}{\partial x'_n} & \frac{\partial x_2}{\partial x'_n} & \cdots & \frac{\partial x_n}{\partial x'_n} \end{pmatrix} \quad (\text{A.10})$$

To transform a partial differential equations, we need to expressed derivatives in (x_1, x_2, \dots, x_n) in derivatives in $(x'_1, x'_2, \dots, x'_n)$:

$$\begin{pmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{pmatrix} = \mathbf{M}^{-1} \begin{pmatrix} \frac{\partial}{\partial x'_1} \\ \frac{\partial}{\partial x'_2} \\ \vdots \\ \frac{\partial}{\partial x'_n} \end{pmatrix} \quad (\text{A.11})$$

The determinant of the matrix \mathbf{M} is called the Jacobian J . At some locations, the Jacobian may be zero and the inverse does not exists. For example, the Jacobian is zero at the origin of polar coordinate system.

If the transformations is given in the form of equations (A.4) - (A.6), the derivative in the new coordinate system can be obtained directly by:

$$\begin{pmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \frac{\partial x'_1}{\partial x_1} & \frac{\partial x'_2}{\partial x_1} & \cdots & \frac{\partial x'_n}{\partial x_1} \\ \frac{\partial x'_1}{\partial x_2} & \frac{\partial x'_2}{\partial x_2} & \cdots & \frac{\partial x'_n}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x'_1}{\partial x_n} & \frac{\partial x'_2}{\partial x_n} & \cdots & \frac{\partial x'_n}{\partial x_n} \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial x'_1} \\ \frac{\partial}{\partial x'_2} \\ \vdots \\ \frac{\partial}{\partial x'_n} \end{pmatrix} \quad (\text{A.12})$$

An infinitesimal increment dx_j is transformed according to the following rule:

$$dx_j = \frac{\partial x_j}{\partial x'_i} dx'_i \quad (\text{A.13})$$

This can be expressed in matrix form as:

$$\begin{pmatrix} dx_1 \\ dx_2 \\ \vdots \\ dx_n \end{pmatrix} = \mathbf{M}^T \begin{pmatrix} dx'_1 \\ dx'_2 \\ \vdots \\ dx'_n \end{pmatrix} \quad (\text{A.14})$$

Note that here the infinitesimal increments are transformed by the multiplication of the matrix \mathbf{M}^T whereas the derivative are transformed by \mathbf{M}^{-1} .

For vector fields, a new set of basis vector $(\mathbf{e}'_1, \dots, \mathbf{e}'_n)$ need to be introduced. The vector \mathbf{e}'_i is proportional to the direction the variable x'_i and all other variables are constant.

$$h_i \mathbf{e}'_i = \frac{\partial}{\partial x'_i} (x_j \mathbf{e}_j) \quad (\text{A.15})$$

$$= \frac{\partial x_j}{\partial x'_i} \mathbf{e}_j \quad (\text{A.16})$$

The proportionality constant h_i is determined by requiring that the norm of \mathbf{e}'_i is 1.

$$h_i = \sqrt{\sum_{j=1}^n \left(\frac{\partial x_j}{\partial x'_i} \right)^2} \quad (\text{A.17})$$

The components of a vector field \mathbf{v} are obtained by projecting this vector on the basis vectors:

$$v_j \mathbf{e}_j = v'_i \mathbf{e}'_i \quad (\text{A.18})$$

$$v_j = \mathbf{e}'_i \cdot \mathbf{e}_j v'_i \quad (\text{A.19})$$

$$= \frac{\partial x_j}{\partial x'_i} \frac{v'_i}{h_i} \quad (\text{A.20})$$

$$\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \mathbf{M}^T \begin{pmatrix} v'_1/h_1 \\ v'_2/h_2 \\ \vdots \\ v'_n/h_n \end{pmatrix} \quad (\text{A.21})$$

A.1. Example

Polar coordinates:

$$x = r \cos(\theta) \quad (\text{A.22})$$

$$y = r \sin(\theta) \quad (\text{A.23})$$

$$\mathbf{M} = \begin{pmatrix} \frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} \\ \frac{\partial x}{\partial \theta} & \frac{\partial y}{\partial \theta} \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -r \sin(\theta) & r \cos(\theta) \end{pmatrix} \quad (\text{A.24})$$

Jacobian

$$J = r \cos^2(\theta) + r \sin^2(\theta) = r \quad (\text{A.25})$$

The inverse matrix

$$\mathbf{M}^{-1} = \begin{pmatrix} \cos(\theta) & -\frac{1}{r} \sin(\theta) \\ \sin(\theta) & \frac{1}{r} \cos(\theta) \end{pmatrix} \quad (\text{A.26})$$

$$\frac{\partial}{\partial x} = \cos(\theta) \frac{\partial}{\partial r} - \frac{1}{r} \sin(\theta) \frac{\partial}{\partial \theta} \quad (\text{A.27})$$

$$\frac{\partial}{\partial y} = \sin(\theta) \frac{\partial}{\partial r} + \frac{1}{r} \cos(\theta) \frac{\partial}{\partial \theta} \quad (\text{A.28})$$

Appendix B

Measures of humidity

The water vapor content of the air is important to compute the latent heat flux between the air and the ocean. It also intervenes in the long-wave radiation due to the greenhouse effect. Unfortunately, several ways exist to express the humidity (absolute humidity, relative humidity, specific humidity, partial pressure of water vapor, mixing ratio, dew point temperature, ...). Due to this proliferation of humidity measures, it might be necessary to convert the humidity measure provided by the atmospheric model to the humidity measure need by the ocean model.

B.1. Definitions

Absolute humidity : The absolute humidity ρ_v is the mass of water vapour per volume of wet air.

Density of dry air : The density of dry air ρ_d is mass of air per volume of wet air.

Density of wet air : The density of wet air ρ is mass of air and water per volume of wet air.

All are related by:

$$\rho = \rho_v + \rho_d \quad (\text{B.1})$$

Specific humidity : q_s

$$q_s = \frac{\rho_v}{\rho} = \frac{\rho_v}{\rho_v + \rho_d} \quad (\text{B.2})$$

Mixing ratio q_v : the mass of water vapor divided by the mass of dry air

$$q_v = \frac{\rho_v}{\rho_d} = \frac{\rho_v}{\rho - \rho_v} \quad (\text{B.3})$$

Dew point temperature T_d : The dew point is the temperature at which a given parcel of humid air must be cooled, at constant barometric pressure, for water vapor to condense into water.

B.2. Mixing ratio and specific humidity

Mixing ratio and specific humidity are directly related by:

$$q_s = \frac{\rho_v}{\rho_v + \rho_d} = \frac{q_v}{1 + q_v} \quad (\text{B.4})$$

and

$$q_v = \frac{\rho_v}{\rho - \rho_v} = \frac{q_s}{1 - q_s} \quad (\text{B.5})$$

B.3. The ideal gas law

For a mixture of ideal gases, the partial pressure of any component can be found from the ideal gas law applied to that component only. The ideal gas law is applied to water vapor and dry air”

$$e = \rho_v R_v T \quad (\text{B.6})$$

$$p_d = \rho_d R_d T \quad (\text{B.7})$$

where T is temperature, e is water vapour pressure and p_d is pressure of dry air. R_v and R_d are the specific gas constant for water vapor (462 J/(kg K)) and dry air respectively (287 J/(kg K)).

The total air pressure p is the sum of these partial pressures:

$$p = e + p_d \quad (\text{B.8})$$

For most application, only the ratio of these constants are important:

$$\epsilon = \frac{R_d}{R_v} = 0.62198 \quad (\text{B.9})$$

B.4. Water vapour saturation pressure

Water vapour saturation pressure e_s is the maximum partial pressure that water vapor molecules would exert if the air were saturated with vapor at a given temperature and pressure.

Several empirical formulas exist for the water vapor saturation pressure. Over liquid water, the Tetens formula approximates $e_s(T, p)$:

$$e_s(T, p) = 611.21 (1.0007 + 3.46 \cdot 10^{-8} P) \exp \left(\frac{17.502T}{240.97 + T} \right) \quad (\text{B.10})$$

where P and e_s are Pascal and T in degree Celsius.

B.5. Relative humidity

The relative humidity is defined by:

$$r_h = \frac{e}{e_s} \quad (\text{B.11})$$

It is often expressed in %. From the ideal gas law, it follows that:

$$r_h = \frac{\rho_v}{\rho_s} \quad (\text{B.12})$$

where ρ_s is the density of water vapour in saturated air.

$$\rho_s = \frac{e_s}{R_v T} \quad (\text{B.13})$$

B.6. From water vapour pressure to specific humidity

To convert the from one humidity measure to another, it is convenient to use the water vapor pressure. As an example, we derive the equation linking specific humidity and vapour pressure.

$$q_s = \frac{\rho_v}{\rho_v + \rho_d} \quad (\text{B.14})$$

$$= \frac{\frac{e}{R_v T}}{\frac{e}{R_v T} + \frac{p_d}{R_d T}} \quad (\text{B.15})$$

$$= \frac{\epsilon e}{\epsilon e + p_d} \quad (\text{B.16})$$

$$= \frac{\epsilon e}{\epsilon e + p - e} \quad (\text{B.17})$$

$$= \frac{\epsilon e}{p + (\epsilon - 1)e} \quad (\text{B.18})$$

Since in general $p \gg e$, the following approximation is often used:

$$q_s \sim \frac{\epsilon e}{p} \quad (\text{B.19})$$

Appendix C

NetCDF

NetCDF is a machine-independent file format for scientific data sets. Most numerical models save their output as NetCDF files either directly or as a post-processing step. The file format allows to describe the saved data, for example it allows to specify units and the meaning of the dimensions of the variables. The NetCDF library is available at www.unidata.ucar.edu/software/netcdf/.

C.1. Fortran 90

C.1.1. Reading NetCDF files

```
!  
! Read data to a netcdf file  
!  
!
```

```
! Compile with something like:
!  
! gfortran -o read_netcdf read_netcdf.f90 -I.../netcdf/include -L.../netcdf/lib -lnetcdff -lnetcdf  
!  
! or  
! gfortran -o read_netcdf read_netcdf.f90 $(nc-config --fflags --flibs)  
!  
! Execute:  
!  
! ./read_netcdf  
!
```

```
program read_netcdf  
  use netcdf  
  implicit none  
  
  integer :: ncid, status, dimids(2), varid  
  integer :: i,j  
  real :: temp(6,4), valid_range(2)  
  character(64) :: units  
  
  ! open netcdf file example.nc in read-only  
  
  status = nf90_open('example.nc',nf90_nowrite,ncid)  
  call check_error(status)  
  
  ! find the identifier for the variable 'temp'
```

```

status = nf90_inq_varid(ncid, 'temp', varid)
call check_error(status)

! retrieve the netcdf variable temp
! the variable temp must have the same size than in the NetCDF file

status = nf90_get_var(ncid, varid, temp)
call check_error(status)

! retrieve the attribute units of variable temp

status = nf90_get_att(ncid, varid, 'units', units)
call check_error(status)

! retrieve the attribute valid_range of variable temp

status = nf90_get_att(ncid, varid, 'valid_range', valid_range)
call check_error(status)

! close file

status = nf90_close(ncid)
call check_error(status)

write(6,*) 'Units: ',units
write(6,*) 'Valid_range: ',valid_range
write(6,*) 'Temp: '

```

```

do j=1,4
  write(6,*) (temp(i,j),i=1,6)
end do

contains

subroutine check_error(status)
  integer, intent ( in) :: status

  if(status /= nf90_noerr) then
    write(6,*) 'NetCDF error: ',trim(nf90_strerror(status))
    stop "Stopped"
  end if
end subroutine check_error

end program read_netcdf

```

C.1.2. Writing NetCDF files

```

!
! Write data to a netcdf file
!
!
! Compile with something like:
!
! gfortran -o write_netcdf write_netcdf.f90 -I.../netcdf/include -L.../netcdf/lib -lnetcdff -lnetcdf
!

```

```
! or
!  
! gfortran -o write_netcdf write_netcdf.f90 $(nc-config --fflags --flibs)  
!  
! Execute:  
!  
! ./write_netcdf  
!
```

```
program write_netcdf  
  use netcdf  
  implicit none  
  
  integer :: ncid, status, dimids(2), varid  
  integer :: i,j  
  real :: temp(6,4)  
  
  ! create some data  
  
  do j=1,4  
    do i=1,6  
      temp(i,j) = i+j  
    end do  
  end do  
  
  ! create netcdf file called example.nc  
  ! nf90_clobber: overwrite if exists
```

```
status = nf90_create('example.nc',nf90_clobber,ncid)
call check_error(status)

! define the dimension longitude and latitude of size
! approximate size

status = nf90_def_dim(ncid, 'longitude', 6, dimids(1))
call check_error(status)

status = nf90_def_dim(ncid, 'latitude', 4, dimids(2))
call check_error(status)

! create a variable temp of type float of the size 6x4
! (dimension longitude and latitude).

status = nf90_def_var(ncid, 'temp', nf90_float, dimids, varid)
call check_error(status)

! define a string as attribute of the variable

status = nf90_put_att(ncid, varid, 'units', 'degree Celsius')
call check_error(status)

! define a vector of floats as attribute of the variable

status = nf90_put_att(ncid, varid, 'valid_range', (/ -10., 40. /))
call check_error(status)
```



```

! end definitions: leave define mode

status = nf90_enddef(ncid)
call check_error(status)

! store the variable temp in the netcdf file

status = nf90_put_var(ncid,varid,temp)
call check_error(status)

! close netcdf file and all changes are written to disk

status = nf90_close(ncid)
call check_error(status)

write(6,*) 'example.nc file created. You might now inspect this file'
write(6,*) 'with the shell command "ncdump example.nc"'

contains

subroutine check_error(status)
  integer, intent ( in) :: status

  if(status /= nf90_noerr) then
    write(6,*) 'NetCDF error: ',trim(nf90_strerror(status))
    stop "Stopped"
  end if

```

```
end subroutine check_error

end program write_netcdf
```

C.2. Matlab and Octave

Matlab R2012 or newer has support for NetCDF. For Octave you need to install the package “netcdf” from <http://octave.sourceforge.net/netcdf/>.

C.2.1. Reading NetCDF files

```
% Example for reading a netcdf file
% in Matlab and Octave

% the name of the netcdf file
filename = 'example.nc';

% retrieve the netcdf variable temp
temp = ncread(filename,'temp');

% retrieve the attribute units of variable temp
temp_units = ncreadatt(filename,'temp','units');

% retrieve the attribute valid_range of variable temp
temp_valid_range = ncreadatt(filename,'temp','valid_range');

% retrieve the global attribute history
global_history = ncreadatt(filename,'/','history');
```

C.2.2. Writing NetCDF files

```
% Example for creating a netcdf file
% in Matlab and Octave

% create some variables to store them in a netcdf file

latitude = -90:1:90;
longitude = -179:1:180;
[y,x] = meshgrid(pi/180 * latitude,pi/180 * longitude);
temp = cos(2*x) .* cos(y);

%-----%
%                               %
% write data to a netcdf file   %
%                               %
%-----%

filename = 'example.nc';
delete(filename);

% coordinate variable longitude

% create a variable longitude of type double with
% 360 elements (dimension longitude).

nccreate(filename,'longitude','Dimensions',{'longitude',size(temp,1)},'Format','classic');
ncwriteatt(filename,'longitude','standard_name','longitude');
```

```

% define a string attribute of the variable longitude
ncwriteatt(filename,'longitude','units','degree_east');

% coordinate variable latitude

nccreate(filename,'latitude','Dimensions',{'latitude',size(temp,2)});
ncwriteatt(filename,'latitude','standard_name','latitude');
ncwriteatt(filename,'latitude','units','degree_north');

% define variable temp

% create a variable temp of type double of the size 360x181
% (dimension longitude and latitude).

nccreate(filename,'temp','Dimensions',{'longitude','latitude'});
ncwriteatt(filename,'temp','standard_name','northward_sea_water_velocity');
ncwriteatt(filename,'temp','units','m s-1');
ncwriteatt(filename,'temp','valid_range',[-10 40]);

ncwriteatt(filename,'/', 'history','netcdf file created by write_netcdf.m');

% store the octave variables longitude, latitude
% and temp in the netcdf file

ncwrite(filename,'longitude',longitude);
ncwrite(filename,'latitude',latitude);
ncwrite(filename,'temp',temp);

```

```
disp(['example.nc file created. You might now inspect this file']);  
disp(['with the shell command "ncdump -h example.nc"']);
```

Bibliography

- Adcroft, A., C. Hill, and J. Marshall, 1997: Representation of topography by shaved cells in a height coordinate ocean model. *Monthly Weather Review*, **125**, 2293–2315.
- Arakawa, A., 1966: Computational design for long-term numerical integration of the equation of fluid motion: two-dimensional incompressible flow. Part I. *Journal of Computational Physics*, **1**, 119–143.
- Arakawa, A. and V. Lamb, 1977: *Computational design of the basic dynamical process of the UCLA general circulation model*. Methods in Computational Physics, Academic Press, New York, 173–265.
- 1981: A potential enstrophy and energy conserving scheme for the shallow water equations. *Monthly Weather Review*, **109**, 18–36.
- Barth, A., A. Alvera-Azcárate, and R. H. Weisberg, 2008: A nested model study of the Loop Current generated variability and its impact on the West Florida Shelf. *Journal of Geophysical Research*, **113**, C05009, doi:10.1029/2007JC004492.
URL <http://hdl.handle.net/2268/26199>
- Beckers, J.-M., 1991: Application of a 3D model to the Western Mediterranean. *Journal of Marine Systems*, **1**, 315–332.

- Beckers, J.-M. and E. Deleersnijder, 1993: Stability of a FBTCS scheme applied to the propagation of shallow-water inertia-gravity waves on various space grids. *Journal of Computational Physics*, **108**, 95–104.
- Blayo, E. and L. Debreu, 2005: Revisiting open boundary conditions from the point of view of characteristic variables. *Ocean Modelling*, **9**, 231–252.
- Bryan, K., 1969a: Climate and the ocean circulation. III. The ocean model. *Monthly Weather Review*, **97**, 806–827.
- 1969b: A numerical model for the study of the circulation of the world oceans. *Journal of Computational Physics*, **4**, 347–376.
- Castellari, S., N. Pinardi, and K. Leaman, 1998: A model study of air-sea interactions in the Mediterranean Sea. *Journal of Marine Systems*, **18**, 89–114.
- Chapman, D., 1985: Numerical treatment of cross-shelf open boundaries in a barotropic coastal ocean model. *Journal of Physical Oceanography*, **15**, 1060–1075.
- Chassignet, E., H. Arango, D. Dietrich, T. Ezer, M. Ghil, D. Haidvogel, C.-C. Ma, A. Mehra, A. Paiva, and Z. Sirkes, 2000: DAMEE-NAB: The base experiments. *Dynamics of Atmospheres and Oceans*, **32**, 155–184.
- Chassignet, E. and P. Malanotte-Rizzoli, eds., 2000: *Dynamics of Atmospheres and Oceans (special issue)*, Elsevier Science, Amsterdam, volume 32, chapter Ocean Circulation Model Evaluation Experiments for the North Atlantic Basin. 155–432.
- Daniel, P., 2004: Oil slick drift prediction and operational oceanography systems. *OCEAN OPS 04 workshop*, Toulouse, France.
- Davis, R. E., 1976: Predictability of sea surface temperature and sea level pressure anomalies over the North Pacific Ocean. *Journal of Physical Oceanography*, **6**, 249–266.
- Deleersnijder, E. and J.-M. Beckers, 1992: On the use of the σ -coordinate system in regions of large bathymetric variations. *Journal of Marine Systems*, **3**, 381–390.

- Dukowicz, J., 1995: Mesh effects for Rossby waves. *Journal of Computational Physics*, **119**, 188–194.
- Evensen, G., 1994: Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research*, **99**, 10143–10162.
- Flather, R., 1976: A tidal model of the northwest European continental shelf. *Mémoires de la Société Royale des Sciences de Liège*, **6**, 141–164.
- Fleming, R. J., 1971a: On stochastic dynamic prediction. Part I: The energetics of uncertainty and the question of closure. *Monthly Weather Review*, **99**, 851–872.
- 1971b: On stochastic dynamic prediction. Part II: Predictability and utility. *Monthly Weather Review*, **99**, 927–938.
- Gandin, L. S., 1965: *Objective analysis of meteorological fields*. Israel Program for Scientific Translation, Jerusalem, 242 pp.
- Gill, A. E., 1982: *Atmosphere-Ocean Dynamics*, volume 30 of *International Geophysics Series*. Academic Press.
- Griffies, S., C. Boning, F. Bryan, E. Chassignet, R. Gerdes, H. Hasumi, A. Hirst, A.-M. Treguier, and D. Webb, 2000: Developments in ocean climate modelling. *Ocean Modelling*, **2**, 123–192.
- Haney, R. L., 1991: On the pressure gradient force over steep topography in sigma coordinate ocean models. *Journal of Physical Oceanography*, **21**, 610–619.
- Hsieh, W. W., M. Davey, and R. Wajswicz, 1983: The free Kelvin wave in finite-difference numerical models. *Journal of Physical Oceanography*, **13**, 1383–1397.
- Janjić, Z., 1984: Non-linear advection schemes and energy cascade on semi-staggered grids. *Monthly Weather Review*, **112**, 1234–1245.
- Jerlov, N. G., 1968: *Optical oceanography*. Elsevier Publishing Co., New York, 194 pp.

- Kondo, J., 1975: Air-sea bulk transfer coefficients in diabatic conditions. *Boundary-Layer Meteorology*, **91**–112.
- Large, W., J. McWilliams, and S. Doney, 1994: Oceanic vertical mixing: a review and a model with a nonlocal boundary layer parameterization. *Reviews of Geophysics*, **32**, 363–403.
- Leith, C. E., 1971: Atmospheric predictability and two-dimensional turbulence. *Journal of the Atmospheric Sciences*, **28**, 145–161.
- 1974: Theoretical skill of Monte Carlo forecasts. *Monthly Weather Review*, **102**, 409–418.
- Leith, C. E. and R. H. Kraichnan, 1972: Predictability of turbulent flows. *Journal of the Atmospheric Sciences*, **29**, 1041–1058.
- McPhaden, M. J., S. E. Zebiak, and M. H. Glantz, 2006: ENSO and an integrating concept in Earth Science. *Science*, **314**, 1710–1715.
- Mellor, G. and T. Yamada, 1982: Development of a turbulence closure model for geophysical fluid problems. *Reviews of Geophysics and Space Physics*, **20**, 851–875.
- Orlanski, I., 1976: A simple boundary condition for unbounded hyperbolic flows. *J. Comput. Phys.*, **21**, 251–269.
- Rosati, A. and K. Miyakoda, 1988: A general circulation model for upper ocean simulation. *Journal of Physical Oceanography*, **18**, 1601–1626.
- Shchepetkin, A. and J. McWilliams, 2003: A Method for Computing Horizontal Pressure-Gradient Force in an Oceanic Model with a Non-Aligned Vertical Coordinate. *Journal of Geophysical Research*, **108**, 1–34.
- 2005: The Regional Oceanic Modeling System: A split-explicit, free-surface, topography-following-coordinate ocean model. *Ocean Modelling*, **9**, 347–404.
- Smagorinsky, J., 1963: General circulation experiments with the primitive equations: I. The basic experiment. *Mon. Wea. Rev.*, **91**, 99–164.

Song, Y. T. and D. Haidvogel, 1994: A Semi-implicit ocean circulation model using a generalized topography following coordinate system. *Journal of Computational Physics*, **115**, 228–244.