Introduction to Numerical Ocean Modelling

Marcello Vichi

Department of Oceanography
University of Cape Town
marcello.vichi@uct.ac.za

May 18, 2025

Course plan

This course is structured in sections. It is designed with an intensive full-time schedule (in-person lectures in bold)

Day	Morning	Afternoon
1	**Sec. 1; Sec. 2 (ex. 1-3)**	Study and exercises
2	**Sec. 3 (ex. 4-5); Sec. 4 (ex. 6-7)**	Study and exercises
3	**Sec. 5 (ex. 8-9);**	Study and exercises: Sec. 6.1 (ex. 10)
4	**Sec. 6.1-6.2 (ex. 11-12)**	Study and exercises
5	**Sec. 6.3 (ex. 13)**	Study and exercises
6	**Team project, Sec. 7**	Start of team work
TBD	Team project; **Project presentations**	

This PDF and all the material are found on GitHub `https://github.com/mvichi/HonsUCT_IntroOceMod.git` Please use your own repository for uploading the exercises. **If you use AI for the exercises, please also upload the chat conversations!**

Outline

Describing and predicting changes in the ocean

Fact

The areas of mathematics which are critical to the description of changing processes are ***calculus*** *and* ***differential equations***[a]

[a] For an accurate definition of mathematical terms see http://mathworld.wolfram.com/

- Calculus (also analysis) is the branch of mathematics that studies variables, functions and their rate of change, as well as length, area and volume of objects
- Differential equations (ordinary differential equations, ODE) and partial differential equations (PDE) are used to describe quantities that change continuously in time and space
- All areas of oceanography rely heavily on these subjects to describe the physical, chemical and biological processes. But obtaining practical solutions is another thing...

Mathematical modelling of ocean systems

Definition

Mathematical modelling is the process of defining **relevant system state variables** and constructing appropriate **relationships** between them to represent **determined features** of a system

- The state variables can be described by means of **continuous** or **discrete** quantities. Ocean temperature and the concentration of dissolved carbon can be treated as continuous variables (owing to the nature of the fluid). Higher trophic levels are better described with discrete quantities. All living organisms are discrete, although continuous approximation are often justifiable
- Models can be **dynamical** or **empirical** (statistical). This course introduces dynamical models based on continuous state variables
- Dynamical modelling includes the various methods to build the equations, to obtain exact and approximate solutions, and to perform simulations

Definition

A **dynamical mathematical model** is a set of differential equations that describe the evolution of a system in time and/or space. It can be solved analytically and/or numerically

A mathematical model for the ocean

The three-dimensional Navier-Stokes equation for an incompressible fluid of constant density in hydrostatic equilibrium on the rotating earth

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} - fv = -\frac{1}{\rho_0}\frac{\partial p}{\partial x} + A_H\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + A_V\frac{\partial^2 u}{\partial z^2}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + fu = -\frac{1}{\rho_0}\frac{\partial p}{\partial y} + A_H\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + A_V\frac{\partial^2 v}{\partial z^2}$$

$$\frac{\partial p}{\partial z} = -\rho_0 g$$

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

$$\frac{\partial \theta}{\partial t} + u\frac{\partial \theta}{\partial x} + v\frac{\partial \theta}{\partial y} + w\frac{\partial \theta}{\partial z} = \kappa_H\left(\frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \theta}{\partial y^2}\right) + \kappa_V\frac{\partial^2 \theta}{\partial z^2}$$

$$\frac{\partial S}{\partial t} + u\frac{\partial S}{\partial x} + v\frac{\partial S}{\partial y} + w\frac{\partial S}{\partial z} = \kappa_H\left(\frac{\partial^2 S}{\partial x^2} + \frac{\partial^2 S}{\partial y^2}\right) + \kappa_V\frac{\partial^2 S}{\partial z^2}$$

Numerical modelling in ocean sciences

- All processes in oceanography are dynamical, that is they change both in time and space
- Finding the time evolution and space distribution of a property requires solving (*integrating*) systems of differential equations

  - This process is done by creating numerical **discretized** forms of the continuous equations that can be solved with a computer code written with a suitable programming language
  - The model is integrated in different ways:
    - hindcast mode
    - forecast mode
    - scenario mode

Ordinary Differential Equations (ODE)

Definition

ODEs are mathematical objects useful to describe changes in time or in space because they are based on one single independent variable

$$\frac{dC}{dt} = f\left(C, t, \text{constants}\right); \quad \frac{dM}{dx} = f\left(M, x, \text{constants}\right)$$

t and x are the independent variables. The functions $C(t)$ and $M(x)$ are the solutions of the differential equations. Type of ODEs:

- **Initial value problem, 0-dimensional (time)**: this model describes the time evolution of a variable (sea surface height, phytoplankton population, etc.). The solution is the (continuous) time series of the variable values, starting from an *initial condition* $C(t = 0)$
- **Boundary value problem, 1-dimensional (space)**: the Lambert-Beer equation is an example of 1-D spatial ODE because it describes the change with depth of the light distribution. The solution is a spatial function (in this case a profile). Spatial ODEs need a *boundary condition*

Malthusian growth model

- The derivation of the Malthusian model is based on the concept that the growth rate is proportional to the population. The independent variable is time and there is an increase of population with time that is expressed as

$$\Delta N = \mu N \Delta t$$

where the growth rate μ is in units of $1/\text{time}$.

- We put this equation in differential form and obtain what is called the dynamics of the exponential growth model

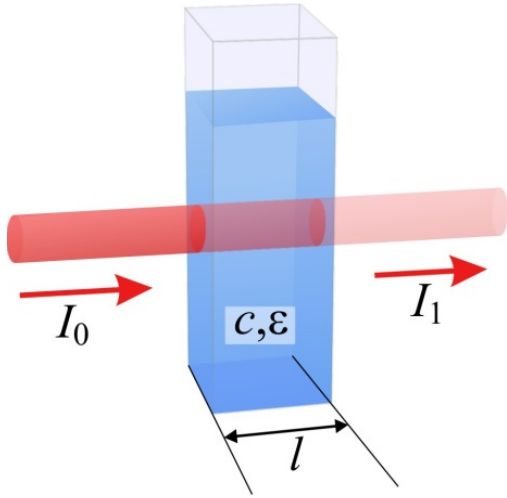
$$\frac{dN}{dt} = \mu N \tag{1}$$

which can be solved analytically by separating the variables, integrating the two sides and applying an initial condition $N(t = 0) = N_0$

$$\int \frac{1}{N} dN = \mu \int dt \Rightarrow N = N_0 e^{\mu t}$$

Light attenuation: the Lambert-beer equation

- Light is attenuated of a certain fraction along its pathway through any medium, which is characterized by the constant attenuation coefficient ε (units are 1/length)



- $$I_1 = I_0 - \varepsilon I_0 \ell \Rightarrow \Delta I = -\varepsilon I \Delta x$$

- By putting it in differential form, one obtains the Lambert-Beer equation

$$\frac{dI}{dx} = -\varepsilon I \tag{2}$$

which can be solved analytically by separating the variables, integrating the two sides and applying the boundary condition $I(x = 0) = I_0$

$$\int \frac{1}{I} dI = -\varepsilon \int dx \Rightarrow I = I_0 e^{-\varepsilon x}$$

Useful ODE features to remember

- the **order** is the value of the highest-order derivative

$$\frac{dC}{dt} = \lambda C^3 - C^2 + a_0 \ \text{(1st)}; \quad \frac{d^2C}{dx^2} = \mu \frac{dC}{dx} - C \ \text{(2nd)}$$

- a **homogeneous** (or autonomous) ODE has only terms containing the dependent variable or its derivative $\frac{dC}{dt} = -\lambda C$
- a **non-homogeneous** (or non-autonomous) ODE contains additional terms, such as a constant or the independent variable itself

$$a_2 \frac{d^2C}{dx^2} + a_1 \frac{dC}{dx} - a_0 C = x^2$$

- a **linear** ODE only contains linear terms, like the decay equation or population growth. If one term is added to represent higher-order terms such as for instance quadratic mortality as in the example below,

$$\frac{dP}{dt} = \mu P - mP^2$$

  then the ODE becomes **non-linear**. Most of the non-linear ODEs are solved numerically
- **initial value problems** have a starting point $C(t = t_0)$ and are *integrated* forward in time to obtain $C(t)$ (for all) $\forall t > t_0$
- **boundary value problems** seek the solution $C(x)$ over a certain domain given specific values at the beginning and end of the domain or also within the domain itself

Exercise 1

- Phytoplankton growth follows an exponential phase, at least for a certain period of time until resources may become limiting or other factors control the Malthusian growth
- The file `flynn94.csv` contains data from a batch experiment conducted by Flynn et al. (1994). The full table and the related units are available in file `flynn94.pdf`. *Flynn, K.J., Davidson, K., Leftley, J.W., 1994. Carbon-nitrogen relations at whole-cell and free-amino-acid levels during batch growth of Isochrysis galbana (Prymnesiophyceae) under conditions of alternating light and dark. Marine Biology 118, 229–237. https://doi.org/10.1007/BF00349789*
- Plot the data using a software of your choice and answer the following questions
  1. Is the growth exponential?
  2. Can you separate the growth curve into some phases?
  3. How different is the growth expressed in cellular carbon from the one in cell number and chlorophyll?
  4. Is the ammonium consumption approximated by an exponential curve?
  5. How would you estimate the parameters of the growth curve?

Ocean systems that can be described with multiple ODEs

Definition

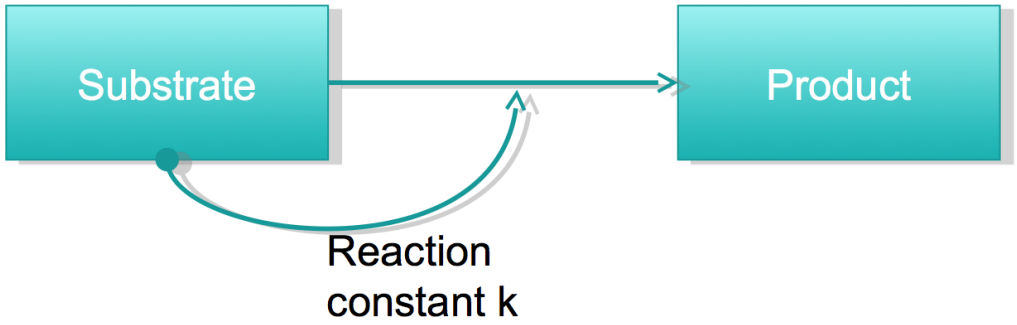
A system of (coupled) ODEs is a combination of ODE equations where the state variables interact with each other.

This is an initial value problem. We know the initial state of the system and want to determine the time evolution of the state variables

- A (biogeo)chemical system with multiple reactions ($A + B \Rightarrow C + D$)
- A system with prey-predator interactions (a trophic web, like primary producers and grazers)
- A physiological model describing biochemical transformations (the carbon biomass formation through photosynthesis driven by light availability)

These models are usually called **box models**: they are characterized by the dynamical co-existence of the state variables in the same enclosure.
Very important: exchanges with the region(s) outside the system are known or negligible

Initial value problems in biology/biogeochemistry



A simple chemical reaction model where the substrate is transformed into a product:

$$\text{substrate} \overset{\text{k}}{\Longrightarrow} \text{product}$$

It is based on the concept that the **transformation rate is proportional to the chemical equilibrium constant and to the concentration of the substrate**. The equations are a direct consequence of the mass conservation principle: the rate of change of the substrate is equal and opposite to the rate of change of the product:

$$\frac{dS}{dt} + \frac{dP}{dt} = \frac{d\left(S+P\right)}{dt} = 0 \tag{3}$$

It is described by a system of 2 variables $S(t)$, $P(t)$ and 2 ODEs

$$\frac{dS}{dt} = -kS$$

$$\frac{dP}{dt} = kS$$

Analytical solutions

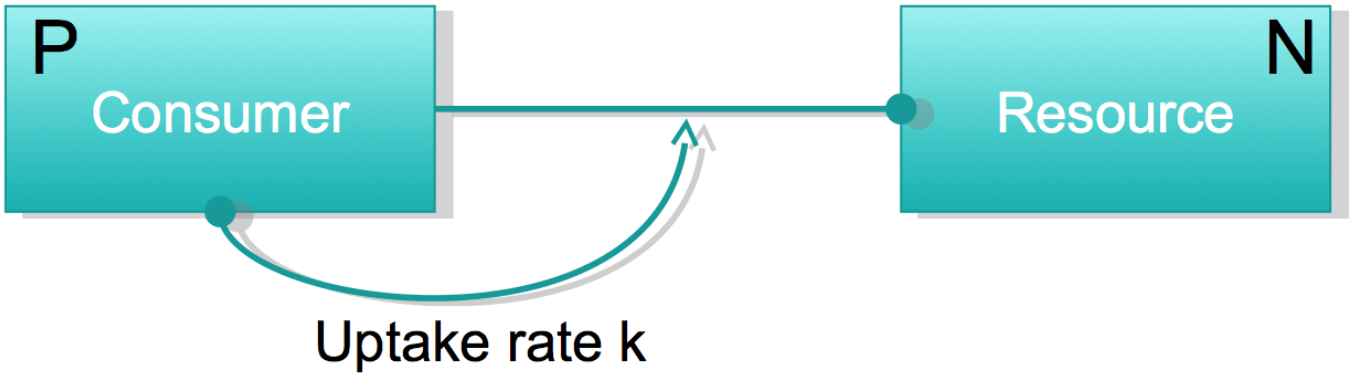
Some ODEs like the decay or Malthusian growth can be integrated by separation of variables (see the Malthusian equation 1); other ODEs can be solved by substitution of variables, or trigonometric equations. All the ODEs that can be analytically solved are tabulated (best reference: CRC Standard Mathematical Tables, Zwillinger, 2002).

The first equation is solved and then the solution goes into the second equation which is again solved by separation of variables with the application of the initial conditions. In this specific case, it is easier to use eq. (3) $P + S = P_0 + S_0$, which is valid for every t ($\forall t$) to derive the solution

$$\begin{aligned} S(t) &= S_0 e^{-kt} \\ P(t) &= P_0 + S_0 - S_0 e^{-kt} \end{aligned}$$

Note that the substrate will never be exactly 0.

Resource and consumer



Let's analyse a system with one resource and one consumer (e.g. phytoplankton and one dissolved nutrient like ammonium). If we assume that the rate of change of the resource is **only controlled by the consumer** (as in the previous case) according to the uptake rate and the abundance we write:

$$\frac{dN}{dt} = -kP$$

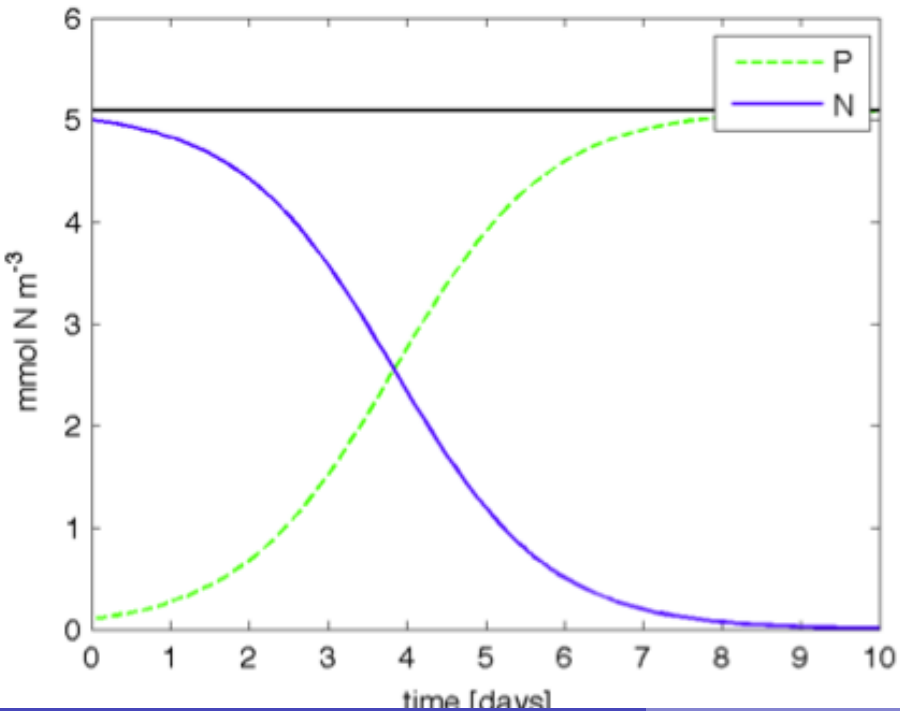
But then the application of mass conservation gives

$$\frac{d\left(P+N\right)}{dt} = 0 \longrightarrow \frac{dP}{dt} = kP$$

which is the infinite Malthusian growth that by definition is not resource-limited. **This set of ODE does not work to model this system** because the **resource should also control the consumption rate** (see the different shape of the arrow in the graph, which means dependence, as opposed to a normal arrow that indicates direct relationship)

Limits to growth

- Growth with limited resources is by definition a non-linear model and we need to adjust the functional form of the equation and include a new parameters that has not only units "per time" but also "per concentration"



$$
\begin{aligned}
k &= k'N \\
\frac{dN}{dt} &= -k'NP \qquad (4) \\
\frac{dP}{dt} &= k'NP
\end{aligned}
$$

- In this case the consumer growth is limited by the resource. This system has the analytical solution

$$P + N = P_0 + N_0$$

$$P(t) = P_0 \frac{P_0 + N_0}{P_0 + N_0 e^{-k'(P_0+N_0)t}}$$

Exercise 2

- Write a python notebook to plot the analytical solution of the model presented in eq. (4) to show the time evolution of phytoplankton and nutrient concentrations. Use the following parameters:
  - Time axis in days, from 0 to 20 days
  - $k' = 0.1$ d^{-1}; $P_0 = 0.1$ mmol N m^{-3} and $N_0 = 5$ mmol N m^{-3}
- Answer the following questions (adding plots where needed)
  1. Why are the curves identical in shape?
  2. What happens when you change the value of the uptake parameter? Think about it as the percentage of food consumed per day.
  3. What happens if the initial biomass of phytoplankton is close to the amount of resource available? How does the shape of the curve change?

More complex models

- Model building is a delicate process, which takes different nuances depending on the group of people doing it and the specific discipline
- There is no *one-size-fits-all* or universal recipe for building a model that describes a certain aspect of the marine environment
- A useful introduction is given by Glover et al. (2011) in their Chapter 9 (available in the document folder). We will use their example 9.3 to explore the prey-predator interactions with the Lotka-Volterra model

The Lotka-Volterra model

- Volterra (1926) wanted to study prey-predator interactions in fish population; Lotka (1920;1925) studied chemical oscillations between reagents and products
- The model describes the interaction between two populations. The prey X_1 grows following the Malthusian equation and is consumed by the predator X_2 similarly to the resource-consumer model. The latter dies according to a constant decay term:

$$\frac{dX_1}{dt} = X_1 (p_1 - p_2 X_2)$$
$$\frac{dX_2}{dt} = X_2 (p_2 p_3 X_1 - p_4)$$

- The presence of the parameter p_3 implies that there is no mass conservation, because the predator is not efficient in translating the ingestion rate into growth. Some mass from the prey is lost to the environment
- **Please note that this not a realistic model: it has major assumptions** (read more about them in the textbook chapter): 1) The prey population is not resource limited. 2) The food supply of the predator population depends entirely on the prey populations. 3) The predator mortality is proportional to the population size. 4) The environment does not change as well as the type of preys and predators.

Model description (from Glover et al 2011)

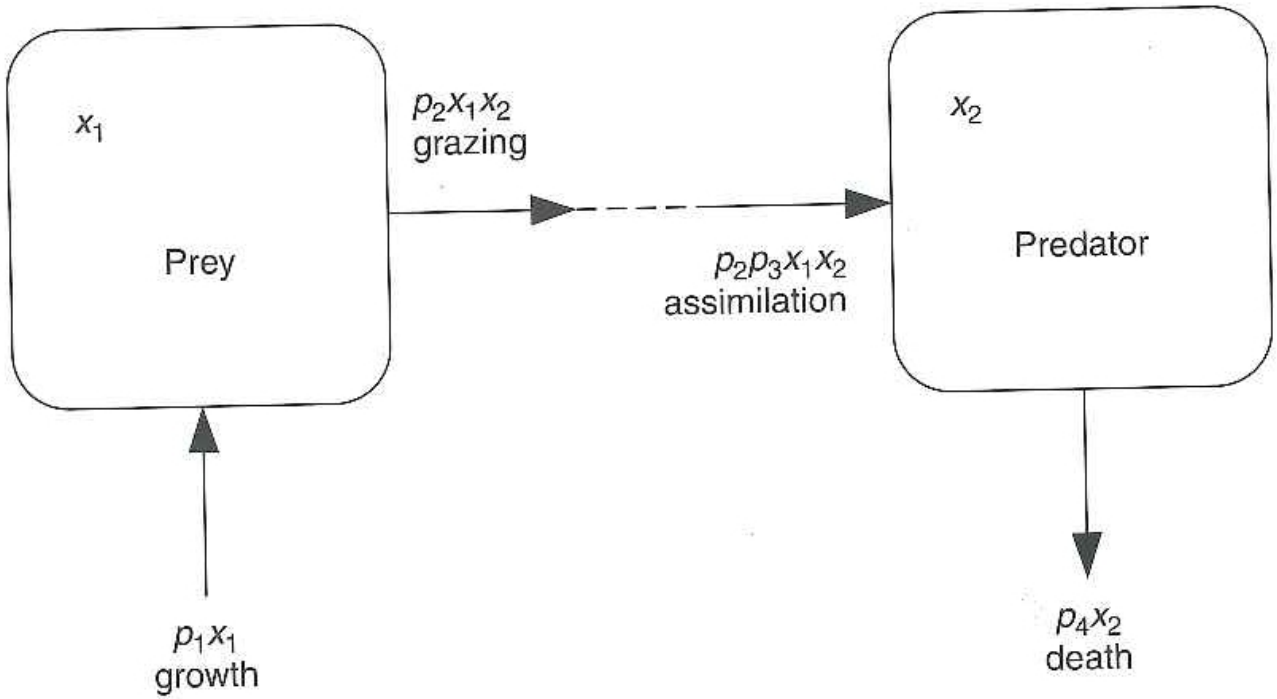
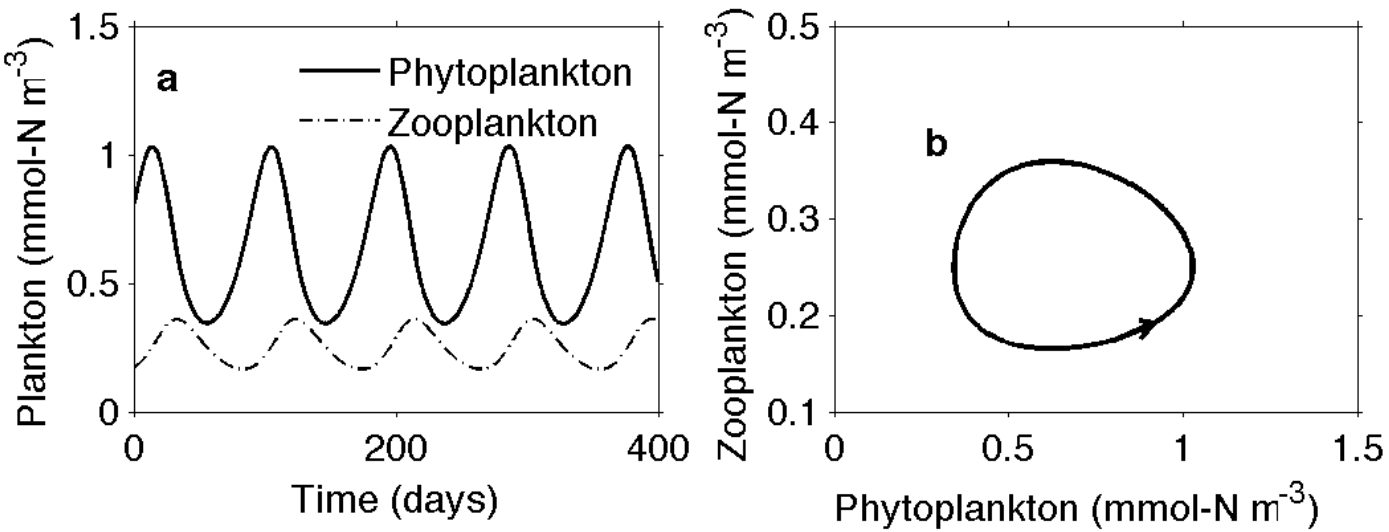


Figure 9.2 A schematic of our simple Lotka–Volterra predator–prey (PZ) model. In this model the state variables are prey (x_1) and predator (x_2). Note that the arrow from box x_1 to box x_2 has a dashed segment to represent the fact that $p_2x_1x_2$ does not necessarily need to equal $p_2p_3x_1x_2$.

A numerical solution from Glover et al (2011)



The LV model is nowadays used mostly for educational purposes, because it allows a rigorous analysis to illustrate the major aspects of mathematical biology and it is a tractable example of numerical solution of ODE systems. Moreover, it does contain elements that are common to any marine model.

Exercise 3

- The L-V model cannot be solved analytically. Section 9.3.6 in the textbook above gives an example of a MATLAB computer code to resolve numerically the system of equations. This code is available in the python notebook `LV_pz.ipynb`. It uses an ODE solver that we will understand more in detail on page 43. Run the code to reproduce the panel a from the previous figure
- Modify the notebook to answer the following questions and upload it on your GitHub repository
  1. Write the code to plot panel b in the appropriate cell block
  2. Set the parameter $p_3 = 1$ and compare how different the trajectory is in the phase space
  3. Write the code to compute the solution of the L-V model using the initial conditions given in the caption of Fig. 9.4 in the Glover at al textbook and plot the results
  4. Why are they different? Think about a few hypotheses and then see the explanation in the document `Glover_etal_2011_ERRATA.pdf`

Numerical methods

Approximation of functions

- We need to specify a method by which a continuous function of one or more space dimensions and/or time can be represented in discrete form (finite number of unknown terms).

Approximation of equations

- Having chosen the discrete representation of functions, we need to adopt a way of discretizing and solving the particular equations that the functions must satisfy.

Taylor series

Definition

Any function can be described by a series expansion about a fixed point x_0

$$f(x) = f(x_0) + f\prime(x_0)(x - x_0) + \frac{f''(x_0)(x - x_0)^2}{2!} + \frac{f^{(3)}(x_0)(x - x_0)^3}{3!} + \cdots$$
$$= f(x_0) + \sum_{n=1}^{\infty} \frac{f^{(n)}(x_0)(x - x_0)^n}{n!} \quad (5)$$

$(2! = 2 \times 1\ ;\ 3! = 3 \times 2 \times 1)$The Taylor series expansion is used in ocean and atmosphere sciences to find linear solutions to non-linear dynamical equations. It is also used to derive numerical integration techniques as we will see soon .

The higher order derivatives of $f(x)$ are equivalent to writing

$$f' = \frac{df}{dx};\ f'' = \frac{d^2 f}{dx^2}; \cdots$$

Taylor series as the foundation of finite differences

- The Taylor expansion is useful to know the error we make if we neglect the terms with higher order derivatives

$$f(x) \approx f(x_0) + f\prime(x_0)(x - x_0) + \mathcal{O}\left((x - x_0)^2\right)$$

$$f(x) \approx f(x_0) + f\prime(x_0)(x - x_0) + \frac{f''(x_0)(x - x_0)^2}{2!} + \\ + \frac{f^{(3)}(x_0)(x - x_0)^3}{3!} + \mathcal{O}\left((x - x_0)^4\right)$$

- The symbol $\mathcal{O}\left((x - x_0)^k\right)$ indicates the order of magnitude k of the error. Since we are considering to operate in the close vicinity of the fixed point, we usually have $(x - x_0) \ll 1$ and the error decays geometrically

Exercise 4

- Calculate the Taylor series expansion for the following univariate functions $f(x)$ about the point $x = x_0$. Only show the series terms until the truncation error is $\mathcal{O}\left((x - x_0)^3\right)$ and upload a picture or a document on your GitHub:
  1. $f(x) = \frac{x}{x+1}$
  2. $f(x) = \frac{2x^2}{x^4}$
  3. $f(x) = \ln x^2$

Finite differences

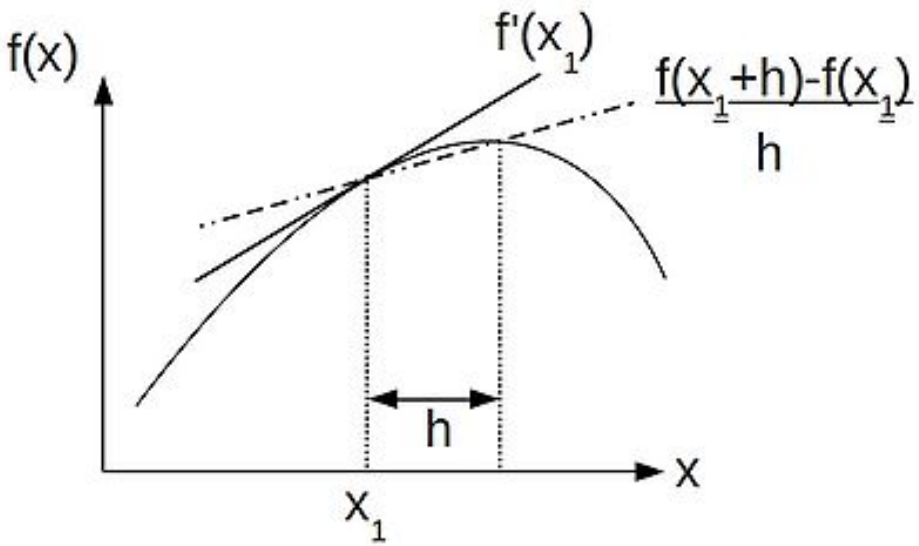
- We can use the Taylor expansion to express any function and its derivatives in a *discrete* way, which means looking at the way this function changes in steps.
- Every step is a finite number, a small interval h. By truncating the series expansion at the first order derivative, the function at point x_1 can be approximated by

$$f(x_1 + h) = f(x_1) + f\prime(x_1)\, h + \mathcal{O}\left(h^2\right)$$

- The first derivative can be approximated to the first order using the truncated series above, which becomes the first-order, **forward-difference** approximation

$$f\prime(x_1) \approx \frac{f(x_1 + h) - f(x_1)}{h} \qquad (6)$$

The truncation error is $\mathcal{O}(h) = \mathcal{O}\left(h^2\right)/h$. This is a graphical representation of the exact and approximated derivatives



The approximated time derivative: first order

Eq. (6) is valid for all the continuous functions of space and time used in the ODE.

To approximate the time derivative of function $f(t)$ to the first order using a time step Δt, we write

$$f' = \frac{df}{dt} = \frac{f(t + \Delta t) - f(t)}{\Delta t} + \mathcal{O}(\Delta t) \tag{7}$$

A simple example: The Decay Problem

Problem adapted from "Ocean Modelling for Beginners".

Chlorophyll-a concentration at the sea- surface in the Benguela upwelling system measured on June 1, 2014 was 0.5 mg/m^3. Zooplankton activity was claimed responsible for a daily consumption rate of 5% (here we take this uptake as the only mechanism responsible for such decay).
How does the chlorophyll concentration evolve with time? What is the value after 20 days?

- Think about a solution for this problem. Notice that what you have to do is just to subtract 5% off on a daily basis from the chlorophyll pool.

Recipes for initial value problems

How to model the problem? And how to resolve it using numerical methods?

1. Express the equation mathematically using an ODE
2. Search an analytical solution if available
3. Discretize the variables (the independent variable time and the dependent variables)
4. Discretize the derivatives and all the terms in the equation using their finite difference approximations based on the Taylor series
5. Separate the known terms on the right hand side (rhs) and unknown terms on the left (lhs) and compute the forward-in-time solution

Solving the decay problem

- 1. Build the differential equation: Based on the system, we formulate the driving ODE

$$\frac{dC}{dt} = -kC \tag{8}$$

C - concentration; t - time; k - constant zooplankton grazing rate $k = 0.05$ d^{-1}

- 2. Is there an analytical solution?: This ODE has a mathematical solution but we pretend we don't know it:

$$C(t) = C_0 e^{-kt} \tag{9}$$

- 3. Discretization: We define *discrete* variables to substitute the *continuous* independent variable (time) and the dependent variable (concentration)

$$t \longrightarrow t^n = t_0 + n\Delta t;\ \ n = 0, 1, 2, \ldots, N$$

$$C(t) \longrightarrow C(t^n) = C^n;\ \ n = 0, 1, 2, \ldots, N$$

Note that the superscript n **is not an exponent**. It is the index of the series of values (an array).

The simplest discretization step: Euler time-forward

- 4. Taylor approximation: We now write the derivative using the discretized form of the equation from (7). This is the simplest first-order approximated form but other forms can be used:

$$\frac{dC}{dt} \approx \frac{C(t+\Delta t) - C(t)}{\Delta t} \longrightarrow \frac{C^{n+1} - C^n}{\Delta t} \tag{10}$$

  All the terms in the equation are now turned into their discrete forms:

$$\frac{C^{n+1} - C^n}{\Delta t} = -kC^n \tag{11}$$

- 5. Separation of the knowns/unknowns: The rhs now contains only known variables

$$C^{n+1} = C^n - k\Delta t C^n = (1 - k\Delta t)C^n \tag{12}$$

This method is called **explicit** because the *new* values of C i.e. C^{n+1} are explicitly given in terms of the *old* values C^n

The Euler-backward or implicit solution

Step 4 can be done in different ways. One other simple method proposed by Euler is to discretize the explicit term on the right hand side using the future (unknown) value of the dependent variable C

$$\frac{C^{n+1} - C^n}{\Delta t} = -kC^{n+1} \tag{13}$$

Step 5 now becomes

$$C^{n+1} = \frac{C^n}{(1 + k\Delta t)} \tag{14}$$

This method is called implicit because the new values of C i.e. C^{n+1} are determined in terms of the future values C^{n+1}. It cannot be done for all the ODEs, because it depends on the degree of linearity of the terms. Not always we can solve the equation for C^{n+1}.

Properties of numerical schemes: convergence

- A numerical scheme is said to be **convergent** if its numerical solution tends towards the solution of the governing differential equation as the interval tend to zero
- If a scheme is unstable (see next slide), it is very unlikely to converge

Properties of numerical schemes: stability

A numerical scheme is stable if it does not *diverge* with time. In simple terms, it means that the forward-in-time solution is found in a **plausible** surrounding of the value from the previous time step

- Inspecting the explicit *Euler forward* scheme

$$C^{n+1} = (1 - k\Delta t)C^n \tag{15}$$

we see that to have $(1 - k\Delta t) \geq 0$

- $k\Delta t$ MUST be less than or equal to 1. If $k\Delta t > 1$, the resulting numerical concentration would be negative which is mathematically plausible but not physically acceptable for a concentration
- $k\Delta t < 1$ is the conditional stability for the explicit numerical scheme: $\Delta t < 1/k$

- The stability of the solution depends on the choice of Δt

Exercise 5 unconditional stability

- Consider the implicit *Euler backward* scheme

$$C^{n+1} = \frac{C^n}{(1 + k\Delta t)}$$

- This scheme is unconditionally stable
- Demonstrate that this is true by using the same method used in the previous slide. Upload a picture or a document

Hybrid methods

Implicit and explicit schemes can be combined in order to alter some properties of the numerical schemes and obtain more stable solutions

$$C^{n+1} = C^n + (-\alpha k C^{n+1} - (1-\alpha) k C^n)\Delta t \tag{16}$$

- $\alpha = 0$ – Explicit case
- $\alpha = 1$ – Implicit case
- $\alpha = 0.5$ – Semi-Implicit case

Accuracy

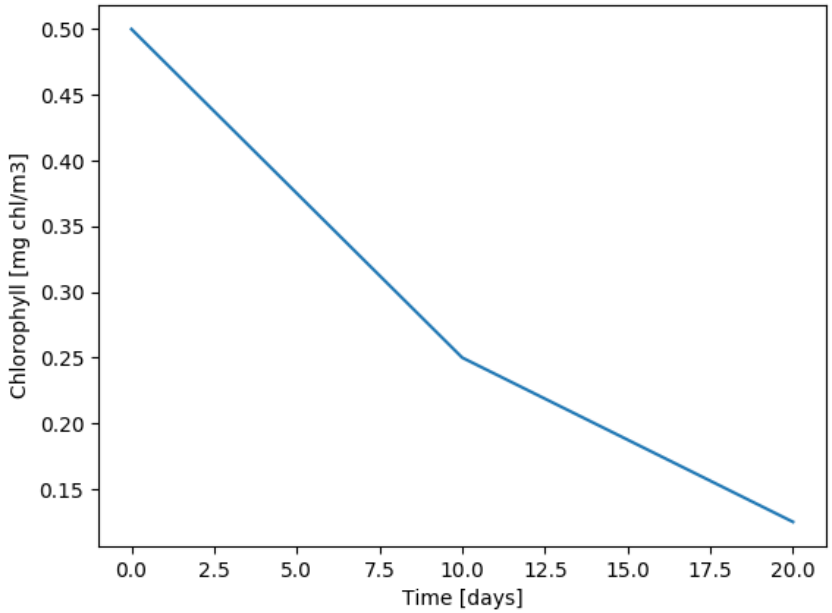
- By using finite difference approximation a **TRUNCATION** error is made.
- The use of higher order numerics in the discretization implies smaller truncation errors.
- Since computers can represent numbers only with finite number of digits, there is always a **ROUND OFF** error. This is what caused the inconsistency in the numerical solution of the L-V model using MATLAB in Ex. 3
- For the numerical solution to be accurate it must have a reasonably smaller errors over the duration of the simulation.

Efficiency

- Model codes have to be written in an efficient manner such that the task is completed within a reasonable time-span, and without stuffing up the computer.

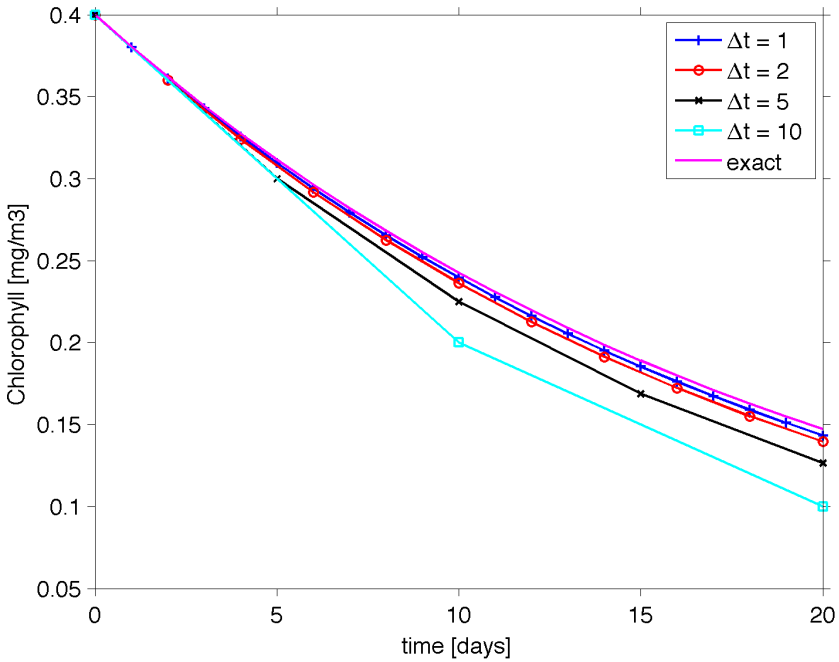
The decay ODE with the Euler-forward method in python

```
"""
A numerical solution of the decay equation.
"""
import pylab as pl

t0 = 0.      # time is in days
tn = 20.
dt = 10.
kappa = 0.05     # day-1
C0 = 0.5         # mg chl/m3

def F(k,C): # define the function for the right
      hand side of the equation
    return -k*C

# compute the total number of steps
Ntot = (pl.floor((tn-t0)/dt) + 1).astype(int)

# initialization
C = pl.zeros(Ntot)
t = C.copy() # this array stores the time and
      has the same size as C
t[0] = t0
C[0] = C0

# start the loop
for n in range(Ntot-1):
    C[n+1] = C[n] + F(kappa,C[n])*dt
    t[n+1] = t[n] + dt
```



The decay ODE with the Euler-forward method in MATLAB

```
% A numerical solution of the decay equation
% time is in days
t0 = 0;
tn = 20;
dt = 10;
kappa = 0.05; % day-1
C0 = 0.4; % mg chl m-3

% compute the total number of steps
Ntot = floor((tn - t0)/dt) + 1;

% define the function using a MATLAB anonymous
      function handle
F = @(k,C) -k*C;

% initializations
C = zeros(Ntot,1); % this array stores the
      output
t = C; % this array stores the time
t(1) = t0;
C(1) = C0;

% start the loop
for n=1:Ntot-1
    C(n+1) = C(n) + F(kappa,C(n))*dt;
    t(n+1) = t(n) + dt;
end
```



Numerical time integration: ODE solvers

- Most natural systems can only be described with mathematical models that do not have an analytical solution
- But we don't need to do the full derivation and write the computer code for each case
- There exist several numerical methods based on **finite differences** and they are grouped under the name of **ODE solvers**. We have already used the python ODE solver called `odeint` from the `scipy` module
- They basically require you to input the right hand side of the differential equations and then use it to compute the solution forward in time at discrete time intervals

Initial value methods

There are 3 broad categories of numerical solvers, and scientific computing software (python, matlab, R, julia) implement various versions.

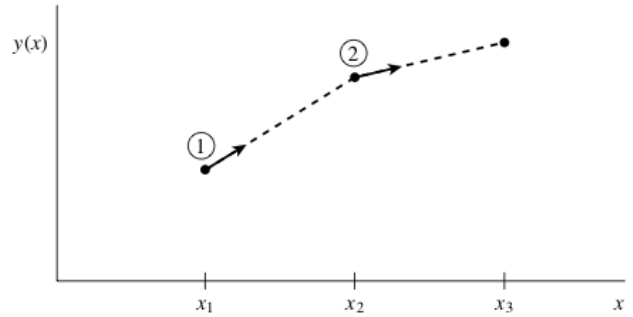
1. Modified Euler-syle methods (Runge-Kutta)
2. Extrapolation type integrators (Bulirsch-Stoer)
3. Predictor-corrector methods (Adams-Bashforth-Moulton)

We will give a few examples of the first group because it's easier to understand. It does not mean though that it is the best to use. There is plenty of algorithms that have been collected in the *Numerical recipes* book by Press et al (http://numerical.recipes/)

Limitations of the Euler method

We have already seen that the Euler method is unstable but there's more to that as explained below. In this page and in all the following examples the ODE is

$$\frac{dy}{dx} = f(x, y)$$



The formula for the Euler method is

$$y_{n+1} = y_n + hf(x_n, y_n) \qquad (16.1.1)$$

which advances a solution from x_n to $x_{n+1} \equiv x_n + h$. The formula is unsymmetrical: It advances the solution through an interval h, but uses derivative information only at the beginning of that interval (see Figure 16.1.1). That means (and you can verify by expansion in power series) that the step's error is only one power of h smaller than the correction, i.e $O(h^2)$ added to (16.1.1).

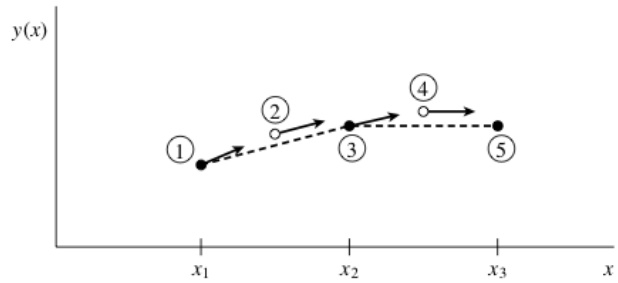
There are several reasons that Euler's method is not recommended for practical use, among them, (i) the method is not very accurate when compared to other, fancier, methods run at the equivalent stepsize, and (ii) neither is it very stable (see §16.6 below).

Exercise 6

- Modify the Euler-forward method implemented in the codes at the beginning of this section (either the python or matlab version) and change it into the Euler-backward method.
- Compare the results for different values of Δt and upload your code and results on the GitHub repository.

Runge-Kutta methods

A more refined method can be obtained by evaluating the function at intermediate steps, as explained in the Numerical recipes textbook



Consider, however, the use of a step like (16.1.1) to take a "trial" step to the midpoint of the interval. Then use the value of both x and y at that midpoint to compute the "real" step across the whole interval. Figure 16.1.2 illustrates the idea. In equations,

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1\right) \\ y_{n+1} &= y_n + k_2 + O(h^3) \end{aligned} \tag{16.1.2}$$

As indicated in the error term, this symmetrization cancels out the first-order error term, making the method *second order*. [A method is conventionally called nth order if its error term is $O(h^{n+1})$.] In fact, (16.1.2) is called the *second-order Runge-Kutta* or *midpoint* method.

4th-order Runge-Kutta (4.5)

The most popular R-K method is the classical fourth-order Runge-Kutta formula. It uses four evaluation points to approximate the numerical solution at each step:

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\ k_3 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) \\ k_4 &= hf(x_n + h, y_n + k_3) \\ y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + O(h^5) \end{aligned}$$

This is a great workhorse that will take you quite far with limited effort.

ODE solvers

Table 8.3 *Summary of MATLAB ODE Solvers*

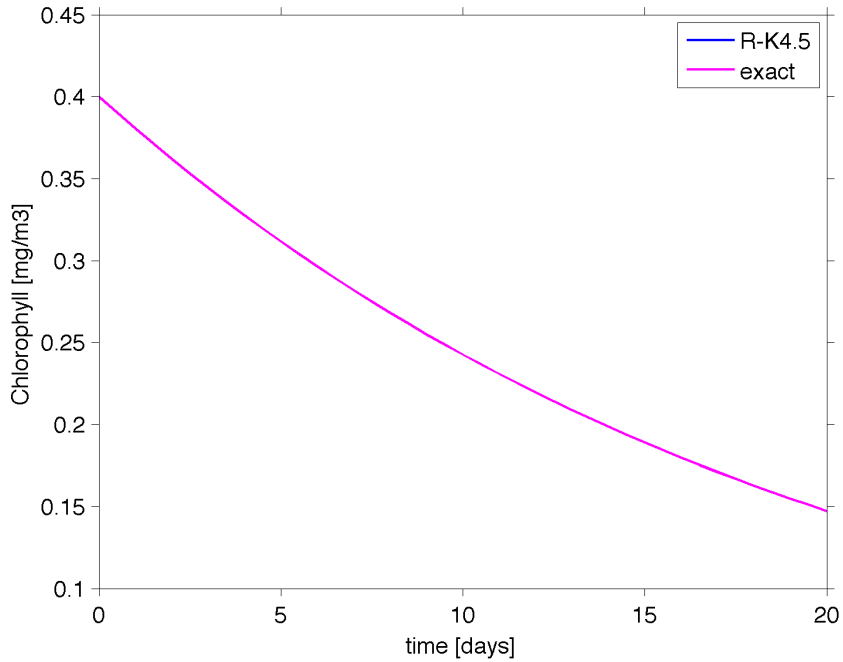
Solver	Problem type	Accuracy	Mathematical method	Recomended uses
`ode45`	non-stiff	medium	Explicit Runge–Kutta 4,5 method.	Try this method first. Often this is as far as you'll need to go.
`ode23`	non-stiff	low	Explicit Runge–Kutta 2,3 method.	Although it uses crude error tolerances, this method may be quicker than `ode45`. Also good for "moderately" stiff systems.
`ode113`	non-stiff	low to high	Variable-order Adams–Bashforth–Moulton predictor–corrector (PECE) method.	For strict error tolerances and/or computationally difficult problems. May give better results than `ode45` at strict tolerances.
`ode15s`	stiff	low to medium	Based on numerical differentiation formulas (NDF), variable order.	If `ode45` is slow (stiff systems). Try this one if `ode45` fails. First-order accurate.
`ode15i`	stiff	low to medium	Fully implicit backward differentiation (BDF), variable order.	If your problem must be formulated in a fully implicit fashion. Initial conditions must be consistent. This one is a little different than the others.
`ode23s`	stiff	low	Modified Rosenbrock method of order 2.	For stiff systems when `ode15s` fails. Crude error tolerances, but higher-order accuracy.
`ode23t`	moderately stiff	low	The trapezoidal rule using a "free" interpolant.	If the problem is only moderately stiff and you need a solution without numerical damping.
`ode23tb`	stiff	low	An implicit Runge–Kutta method with first stage trapezoidal rule and second stage backward	If using crude error tolerances to solve stiff systems. May work when `ode15s` fails at crude tolerances.

Here is a table from Glover et al (2011) explaining the various solvers available in MATLAB

- For MATLAB, ode45 and ode15s are the most used
- python provides package `scipy.integration` that we used in the Lotka-Volterra model integration `https://apmonitor.com/pdc/index.php/Main/SolveDifferentialEquations`

The decay ODE with the R-K4.5 method in MATLAB

```
%% Use MATLAB ODE solver
% time is in days. dt is not
    needed as this solver is
    adaptive
t0 = 0;
tn = 20;
kappa = 0.05; % day-1
C0 = 0.4; % mg chl m-3

% note the different use of the
    function handle. ODE solvers
    require a
% function that takes time and
    the variable(s) as inputs
F = @(t,C) -kappa*C;
[t,C]=ode45(F,[t0 tn],C0);
```



As you can see from the code, there is no need to define the time step. Only the integration limits, and the time step is automatically adjusted to ensure stability and convergence in an efficient way

Exercise 7

- Compare the analytical solution of the model presented in eq. (4) from Exercise 2 with the numerical solution obtained using `odeint` in file `NP_to_NPD.ipynb`. Check the code carefully to understand how the right-end-side of the dynamical equation is implemented.
- Modify the notebook to answer the following questions and upload it on your repository
  1. What are the inputs and outputs in the function `model`? What kind of objects are they?
  2. Write the equtions for a new model to simulate the nutrient-phytoplankton-detritus system (NPD). You will Include a detritus variable D that is produced from the mortality of phytoplankton and it is remineralized as a nutrient. You will need 2 additional parameters: the mortality rate (phytoplankton lysis) and the remineralization rate
  3. Write the code to solve the new model using `odeint` and propose some reasonable values for the parameters that lead to a realistic solution

Approximating functions of multiple variables

The value of a function $f(x)$ in the vicinity of a given location x_0 can be approximated with the Taylor series (5)

- If function f is a multivariate function like the zonal velocity field $U = U(x, y, z, t)$, we have multiple discrete steps to consider around a given point x_0, y_0, z_0 at time t_0: $\Delta X = x - x_0$; $\Delta Y = y - y_0$; $\Delta Z = z - x_0$; $\Delta T = t - t_0$;
- The ordinary derivatives are substituted by partial derivatives for every independent variable
  $\frac{\partial U}{\partial x}|_{x=x_0,y,z,t}$; $\frac{\partial U}{\partial y}$; $\frac{\partial^2 U}{\partial x^2}$; $\frac{\partial U}{\partial t}$
- The formal notation can become pretty heavy, because time and space have to be considered. We will therefore drop some variables to simplify the writing, or we will use the vector notation
  $x_i = (x_1, x_2, x_3) \equiv (x, y, z)$

Taylor series for PDE

With this notations in mind, we can modify eq. (5) to obtain the Taylor Series approximation for the multivariate functions in the surrounding of any time value t (the choice of the truncation error is arbitrary):

$$U(t+\Delta t,x,y,z)=U(t,x,y,z)+\frac{\partial U}{\partial t}\Delta t+\frac{1}{2!}\frac{\partial^2 U}{\partial t^2}\Delta t^2+O(\Delta t^3) \tag{17}$$

and space point x_i:

$$U(t,x_i+\Delta x_i)=U(t,x_i)+\frac{\partial U}{\partial x_i}\Delta x_i+\frac{1}{2!}\frac{\partial^2 U}{\partial x_i^2}\Delta x_i^2+\frac{1}{3!}\frac{\partial^3 U}{\partial x_i^3}\Delta x_i^3+O(\Delta x_i^4) \tag{18}$$

where Δx_i is the increment along one of the 3 spatial directions. To simplify the notation we will only work with the x-axis, thus dropping the index i

$$U(t,x+\Delta x,y,z)=U(t,x,y,z)+\frac{\partial U}{\partial x}\Delta x+\frac{1}{2!}\frac{\partial^2 U}{\partial x^2}\Delta x^2+\frac{1}{3!}\frac{\partial^3 U}{\partial x^3}\Delta x^3+O(\Delta x^4) \tag{19}$$

The same form is used for y and z. [we will also drop the $(t,...,y,z)$ from function U since we are only working with functions of more than one variable]

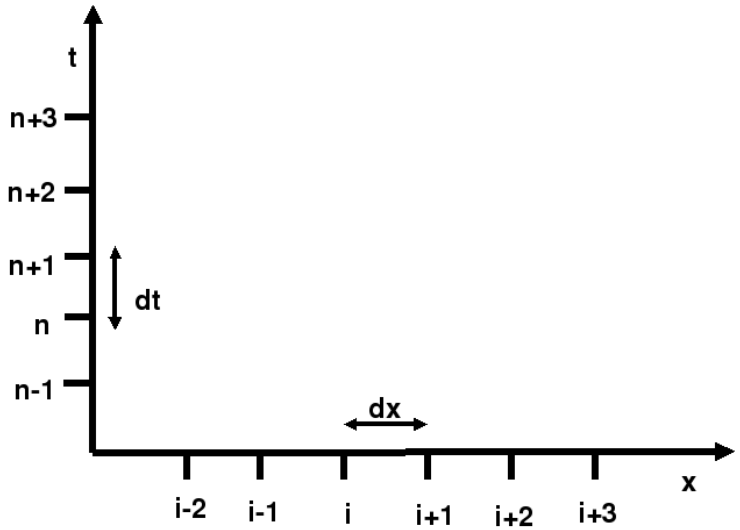
Important difference: space has no preferential direction

The spatial gradients exist in all directions. Think about the slope of a mountain: you can measure it by walking uphill or downhill. The step Δx can thus be either positive or negative, and we should still find exactly the same function

$$U(x-\Delta x)=U(x)-\frac{\partial U}{\partial x}\Delta x+\frac{1}{2!}\frac{\partial^2 U}{\partial x^2}\Delta x^2-\frac{1}{3!}\frac{\partial^3 U}{\partial x^3}\Delta x^3+O(\Delta x^4) \tag{20}$$

This is equivalent to the previous eq. (19), with the only difference that we now look backward instead of forward. We are evaluating the same function, just in a different way.

Discrete grids for PDE



The function $U(x,t)$ needs a two dimensional discrete grid, one axis for space and another axis for time. The value of the function exists at every node of the grid. If $U(x,y,z,t)$ then we will need 4 dimensions.

- For educational purposes, we choose equally spaced points along the x and t dimensions and discretize the independent variables as done on page 32

$$x_i = x_0 + i\Delta x;\ \ t_n = t_0 + n\Delta t$$

to describe the discrete function

$$U_i^n = U\left(x_i, t_n\right).$$

- Hopping on the grid of one or more time and space steps can be written in a convenient way

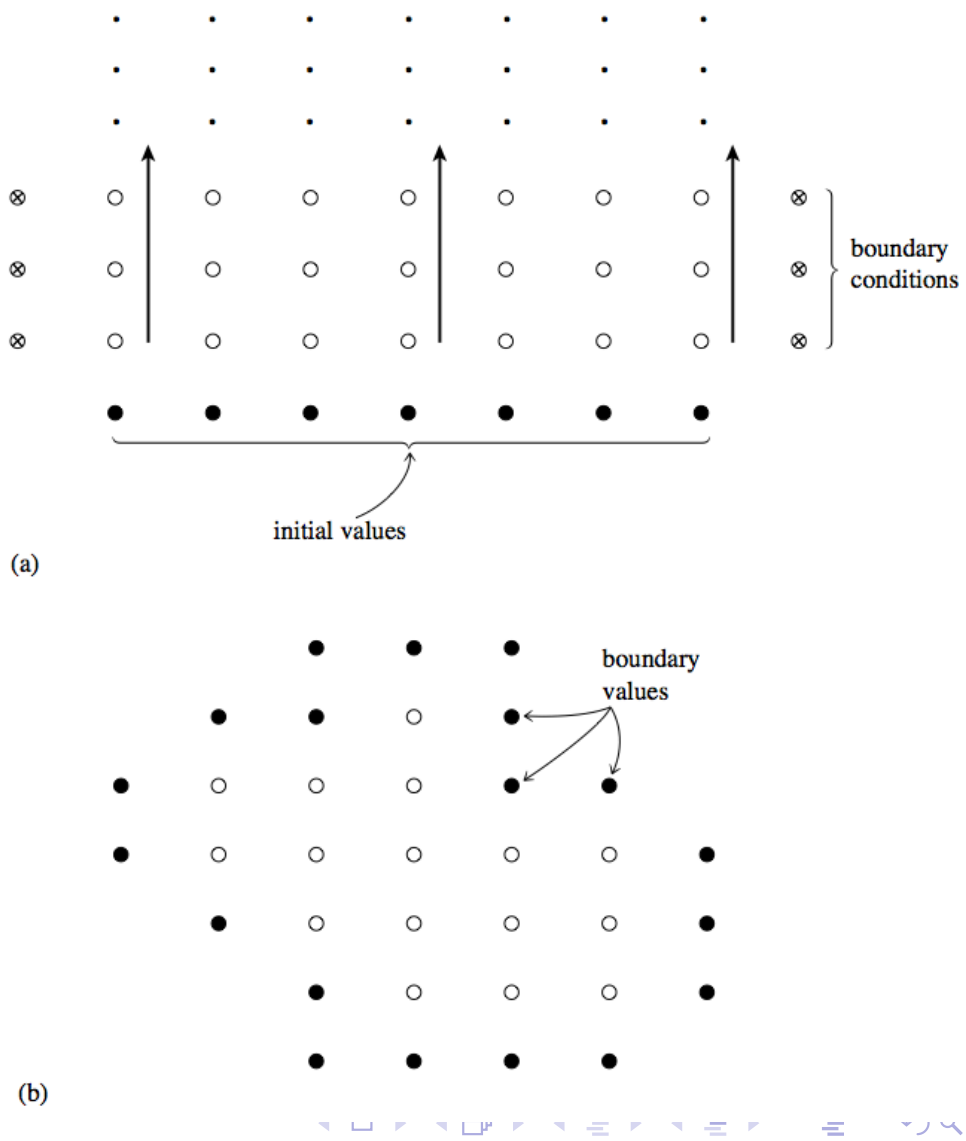
$$U(x+\Delta x, t) = U_{i+1}^n;\ \ U(x, t+\Delta t) = U_i^{n+1}$$

A full 3-D variable would be written as

PDE problems

Comparison of initial values and boundary value problems in PDE (Press et al. 1997)

- In (a) initial values are given on one "time slice," and it is desired to advance the solution in time, computing successive rows of open dots in the direction shown by the arrows. Boundary conditions at the left and right edges of each row (⊗) must also be supplied, but only one row at a time. Only one, or a few, previous rows need be maintained in memory.
- In (b), boundary values are specified around the edge of a grid, and an iterative process is employed to find the values of all the internal points (open circles). All grid points must be maintained in memory



Finite difference approximation: first order in time

The time derivative can be approximated from (17) in the simplest possible manner using a first order forward time step as done for the ODE functions in (7):

$$\frac{\partial U}{\partial t} = \frac{U(t_i + \Delta t) - U(t_i)}{\Delta t} + O(\Delta t) \tag{21}$$

On the discrete grid introduced a few slides earlier, this becomes

$$\frac{\partial U}{\partial t} \approx \frac{U_i^{n+1} - U_i^n}{\Delta t}$$

First order approximation: forward in space

By approximating the function $U(x, y, z, t)$ to the first order along the positive x, we obtain a truncation error that is proportional to Δx^2

$$U(x + \Delta x) = U(x) + \frac{\partial U}{\partial x}\Delta x + O(\Delta x^2) \tag{22}$$

If we now solve for the derivative, we obtain its first-order approximation and the grid-based equivalent form for the **forward in space approximation**, which is sometimes called downstream:

$$\frac{\partial U}{\partial x} = \frac{U(x + \Delta x) - U(x)}{\Delta x} + O(\Delta x) \rightarrow \frac{\partial U}{\partial x} \approx \frac{U_{i+1}^n - U_i^n}{\Delta x} \tag{23}$$

Note that the truncation error for the derivatives is now $O(\Delta x)$ because we divide by Δx and that the spatial differences are taken at the same time index n

Exercise 8 backward in space

Following the method of the previous slide and considering the backward Taylor expansion shown in eq. (20), write the first order approximation and truncation error of

$$U(x - \Delta x) =$$

Rearrange the equation to obtain the velocity gradient

$$\frac{\partial U}{\partial x} =$$

and show the **backward in space numerical approximation** (sometimes called upstream) using the discrete indices (define them accordingly)

$$\frac{\partial U}{\partial x} \approx$$

Upload either a picture of your workings or a document.

Other finite difference approximations: centred in space

Both the forward and backward approximations are valid and equal, as well as it is any linear combination of both, which incidentally cancels the error if we start from the Taylor series and take the difference term by term

$$U(x+\Delta x) = U(x) + \frac{\partial U}{\partial x}\Delta x + \frac{1}{2!}\frac{\partial^2 U}{\partial x^2}\Delta x^2 + O(\Delta x^3) \tag{24}$$

$$U(x-\Delta x) = U(x) - \frac{\partial U}{\partial x}\Delta x + \frac{1}{2!}\frac{\partial^2 U}{\partial x^2}\Delta x^2 + O(\Delta x^3) \tag{25}$$

This leads to the centred difference approximation

$$\frac{\partial U}{\partial x} = \frac{U(x+\Delta x) - U(x-\Delta x)}{2\Delta x} + O(\Delta x^2) \rightarrow \frac{\partial U}{\partial x} \approx \frac{U^n_{j+1} - U^n_{j-1}}{2\Delta x} \tag{26}$$

Note that the centered difference form of the first derivative is accurate to the second order (i.e. has a relatively smaller error than the forms shown in the previous slide)

Higher order derivatives

Using a similar method, one can obtain the other higher order derivatives

$$U(x+\Delta x)=U(x)+\frac{\partial U}{\partial x}\Delta x+\frac{1}{2!}\frac{\partial^2 U}{\partial x^2}\Delta x^2+O(\Delta x^3) \tag{27}$$

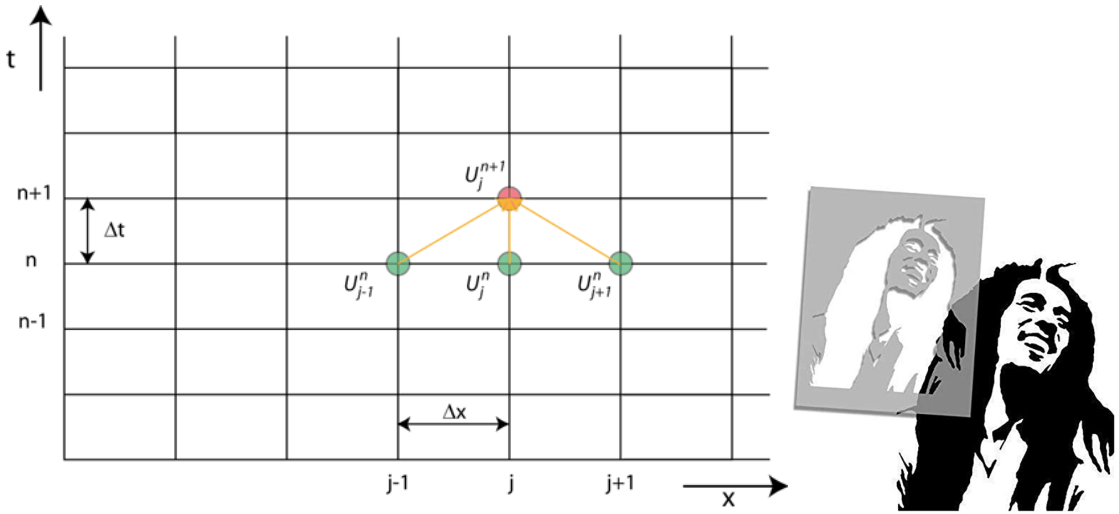
$$\frac{\partial^2 U}{\partial x^2}\Delta x^2=2\left[U(x+\Delta x)-U(x)-\frac{\partial U}{\partial x}\Delta x\right]+O(\Delta x^3) \tag{28}$$

We now replace the full expression for the first order derivative, but to have the same order of error we need to use the centred difference (26) and obtain

$$\frac{\partial^2 U}{\partial x^2}=\frac{U(x+\Delta x)-2U(x)+U(x_i-\Delta x)}{\Delta x^2}+O(\Delta x) \tag{29}$$

We need more points to compute higher order derivatives. Remember that the 2nd derivative measures the curvature and it is understandable that more than 2 points are required. This unfortunately has the consequence to increase the truncation error, so this is less accurate.

The stencil concept



To represent the functioning of a numerical scheme we use the concept of the **stencil**. The solution in time will depend on a fixed set of determined points in space.This example shows the forward-in-time and centred-in-space scheme(FCTS) that is the simplest scheme for PDE illustrated mathematically in the next slide. This scheme can be used to solve the advection and diffusion problems as it will become clearer with the examples in the next section.

Cheat sheet: approximated derivatives for PDEs

Forward in time, **centered** in space (FCTS)

$$Forward - Time : \quad \frac{\partial U}{\partial t} = \frac{U(t_i + \Delta t) - U(t_i)}{\Delta t} + O(\Delta x) \tag{30}$$

$$Centered - Space : \quad \frac{\partial U}{\partial x} = \frac{U(x_i + \Delta x) - U(x_i - \Delta x)}{2\Delta x} + O(\Delta x^2) \tag{31}$$

$$Centered - Space : \quad \frac{\partial U}{\partial y} = \frac{U(y_i + \Delta y) - U(y_i - \Delta y)}{2\Delta y} + O(\Delta y^2) \tag{32}$$

$$Centered - Space : \quad \frac{\partial U}{\partial z} = \frac{U(z_i + \Delta z) - U(z_i - \Delta z)}{2\Delta z} + O(\Delta z^2) \tag{33}$$

Exercise 9

- Write the grid-based, discretized form of eq. (29), first defining the discrete time and space axes to be used as indices

$$t^n = \qquad ; x_i =$$

$$\frac{\partial^2 U}{\partial x^2} \approx$$

- Given the discrete 3-D horizontal velocity field U^n_{ijk}, write the discretized form of eq. (33)
- As usual, upload a picture of your paper or a document

The primitive equations

Ocean models determine the state of the ocean using basic physical (non-linear) relationships (Primitive Equations).

- Navier-Stokes equation (Momentum balance).
  $\sum$Forces = Gravity + Pressure gradient force + Coriolis +Friction.
- Continuity equations (Water, Salt, Heat conservation).
  What goes in, eventually comes out. There is no gain/loss in the system (Changes are only due to imbalances).
- Equation of State.
  Density varies in a non-linear and complex way with Temperature, Salinity and Pressure.

The full set of equations

$$\text{Momentum} \quad \underbrace{\frac{D\mathbf{u}}{Dt}}_{\text{Total acceleration}} = \underbrace{-\frac{\nabla p}{\rho}}_{\text{Pressure gradient}} \underbrace{-f\hat{\mathbf{k}} \times \mathbf{u}}_{\text{Coriolis}} \underbrace{-\nabla\phi}_{\text{Geopotential}} \underbrace{+\mathcal{F}}_{\text{Friction}}$$

$$\text{Water} \quad \underbrace{\nabla \cdot \mathbf{u}}_{\text{Incompressible fluid}} = 0$$

$$\text{Potential temperature} \quad \underbrace{\frac{D\theta}{Dt}}_{\text{Total rate of change}} = \underbrace{\Theta}_{\text{Turbulent fluxes}}$$

$$\text{Salt} \quad \underbrace{\frac{DS}{Dt}}_{\text{Total rate of change}} = \underbrace{\Sigma}_{\text{Turbulent fluxes}}$$

$$\text{Equation of state} \quad \rho = \rho(\theta,\ S,\ P)$$

Component form (Boussinesq+hydrostatic+constant friction)

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} - fv = -\frac{1}{\rho_0}\frac{\partial p}{\partial x} + A_H\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + A_V\frac{\partial^2 u}{\partial z^2}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + fu = -\frac{1}{\rho_0}\frac{\partial p}{\partial y} + A_H\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + A_V\frac{\partial^2 v}{\partial z^2}$$

$$\frac{\partial p}{\partial z} = -\rho_0 g$$

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

$$\frac{\partial \theta}{\partial t} + u\frac{\partial \theta}{\partial x} + v\frac{\partial \theta}{\partial y} + w\frac{\partial \theta}{\partial z} = \kappa_H\left(\frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \theta}{\partial y^2}\right) + \kappa_V\frac{\partial^2 \theta}{\partial z^2}$$

$$\frac{\partial S}{\partial t} + u\frac{\partial S}{\partial x} + v\frac{\partial S}{\partial y} + w\frac{\partial S}{\partial z} = \kappa_H\left(\frac{\partial^2 S}{\partial x^2} + \frac{\partial^2 S}{\partial y^2}\right) + \kappa_V\frac{\partial^2 S}{\partial z^2}$$

Numerical solution of hydrodynamical equations

- the 3D primitive equations of the oceans are complex. They are non linear and involve many partial derivatives
- It is useful to analyse portions of the full equations that are nevertheless meaningful to understand how numerical methods works. The methods described in the next slides are not used in modern numerical models because they are not much efficient, but the underlying concepts are the same. We will consider the
  - Advection terms: transport of a property due to the current speed
  - Turbulent diffusion terms: the parameterization of mixing due to the presence of spatial gradients in a property
- We will work on the advection and diffusion of a scalar variable in one dimension, along the x direction, assuming a constant velocity field and a constant turbulent diffusivity

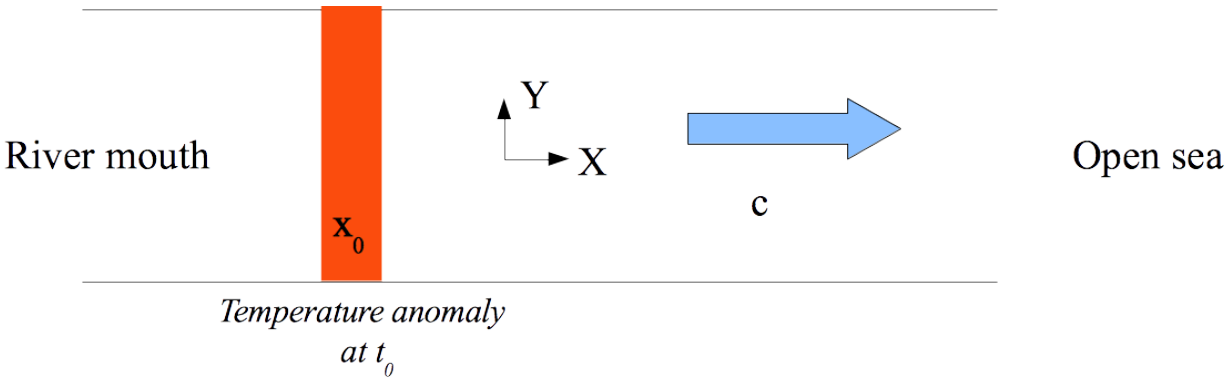
$$\frac{\partial \theta}{\partial t} + c\frac{\partial \theta}{\partial x} = \kappa \frac{\partial^2 \theta}{\partial x^2} \tag{34}$$

- This is a combination of an initial value and a boundary value problem, complex enough to understand the numerical methods

An advection example in one dimension

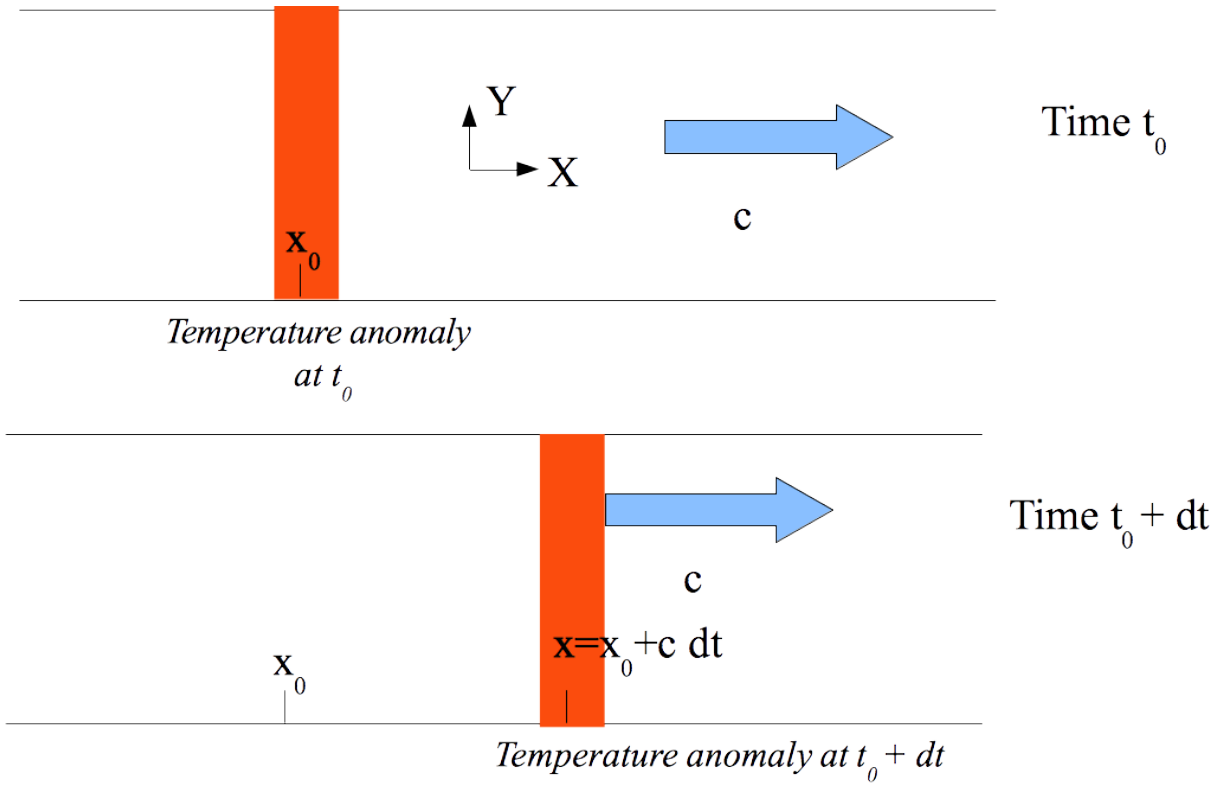
- We model a longitudinal estuary, where the current velocity is along the x direction and constant; $u = c$ and $v = 0$
- It's a wide estuary, with homogeneous processes in the meridional and vertical directions: $\partial_y = 0$ and $\partial_z = 0$ for all properties
- The vertical depth is too small to generate effective vertical velocities: $w = 0$
- An initial perturbation of temperature (e.g. hot water leak from a power plant) occurs at one point in time and over a range of space. We want to study the evolution of this perturbation, which is described by the 1D advection equation

$$\frac{\partial \theta}{\partial t} + c\frac{\partial \theta}{\partial x} = 0$$



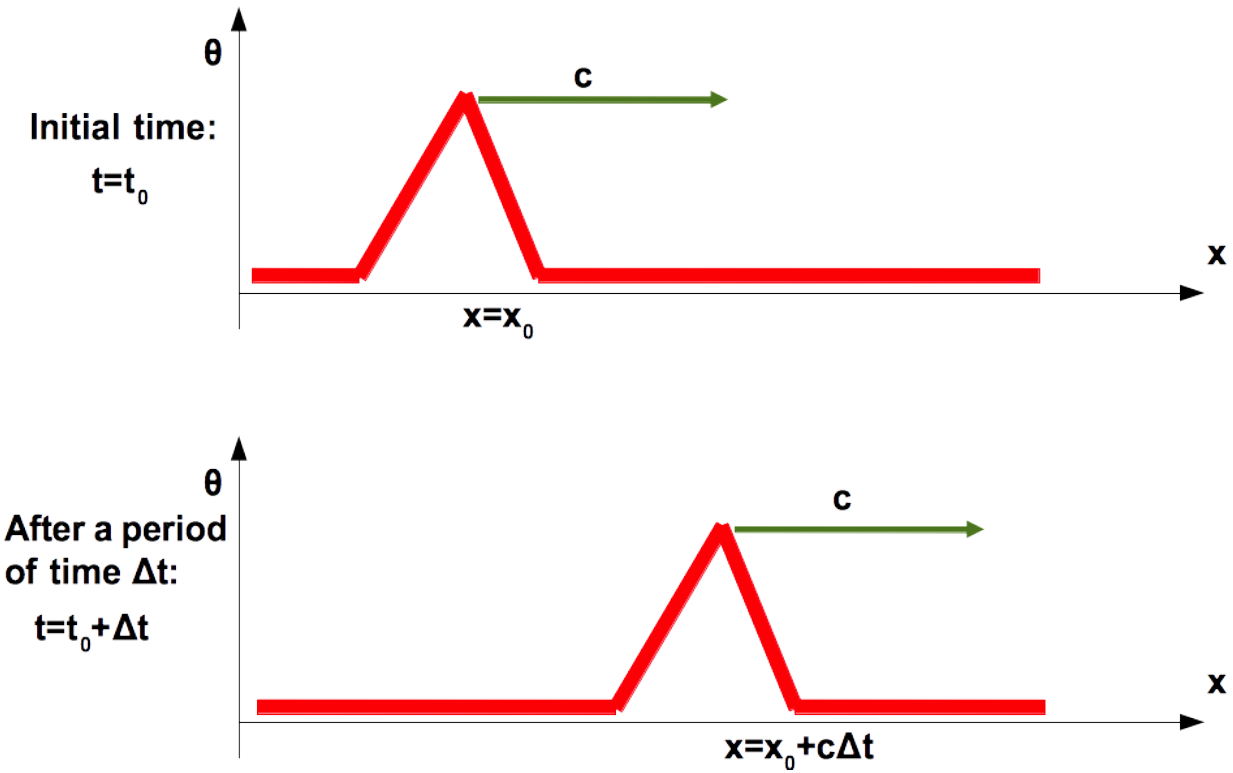
What do we expect?

The solution we expect is the transport of the perturbation along the estuary. We are not considering heat exchange or turbulent dissipation, so the signal will remain undeformed throughout the pathway. This assumption is not fully realistic but simple to implement codewise

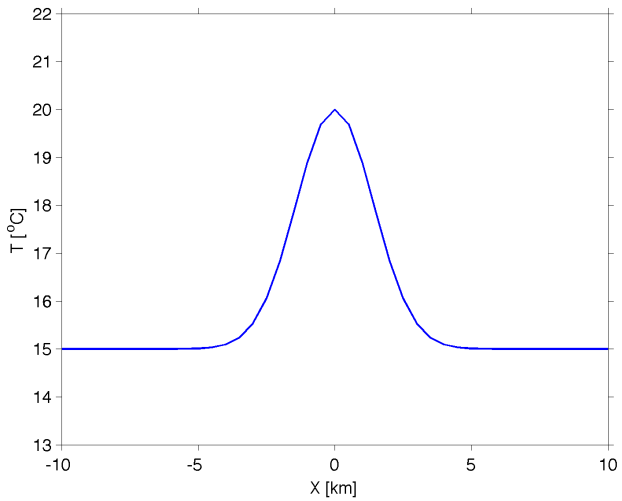


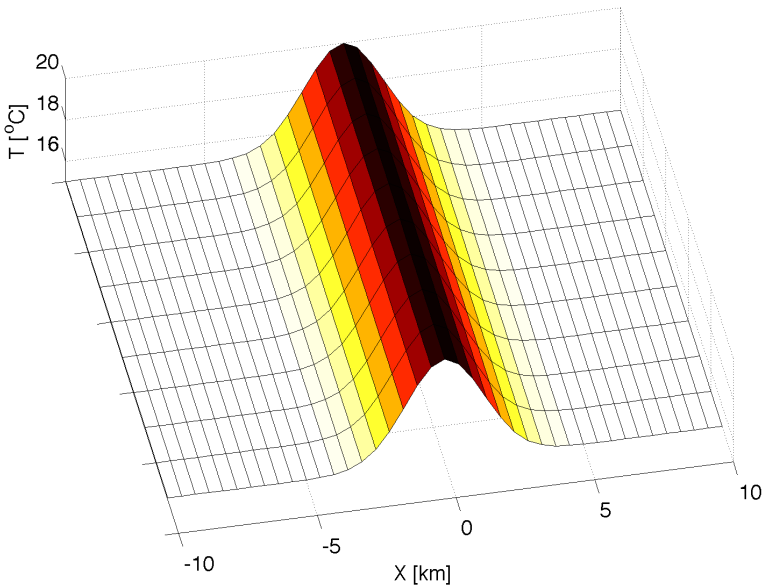
The expected result shown as a mathematical function

The solution we expect is the transport of the perturbation along the estuary. We are not considering heat exchange or dissipation, so the signal stays undeformed throughout the pathway



Building the mathematical model





- For the spatial representation, we choose a channel of length 20 km and an initial grid step of 500 m. All features are homogeneous along y, so we just need to describe the x-axis
- Our choice for the initial condition function is a smoothly peaked function, the Gaussian function, that is controlled by A the amplitude and σ the width parameter:

$$\theta(t_0, x) = \theta_0 + Ae^{-\left(\frac{x}{\sigma}\right)^2}$$

By centering the x-axis around $x_0 = 0$, we get a nicely shaped perturbation. x is in m, but we are showing km in the plot

Analytical solution

- The solution is a function that should **behave like a moving wave**. The perturbation is moving following the speed of the current, which means that if we stay at the same location in space and time (say our x_0 and t_0), after a certain time step δt we should see the passage of the signal that was "behind us" at a distance $\delta x = x_0 - c\delta t$ at time $t_0 - \delta t$.
- This is exactly equivalent as to say that the signal has moved of a distance $\delta x = x_0 + c\delta t$ after a time interval δt
- Wave solutions mix time and space, that is we are seeking for a function of 2 independent variables that are linked by the phase velocity

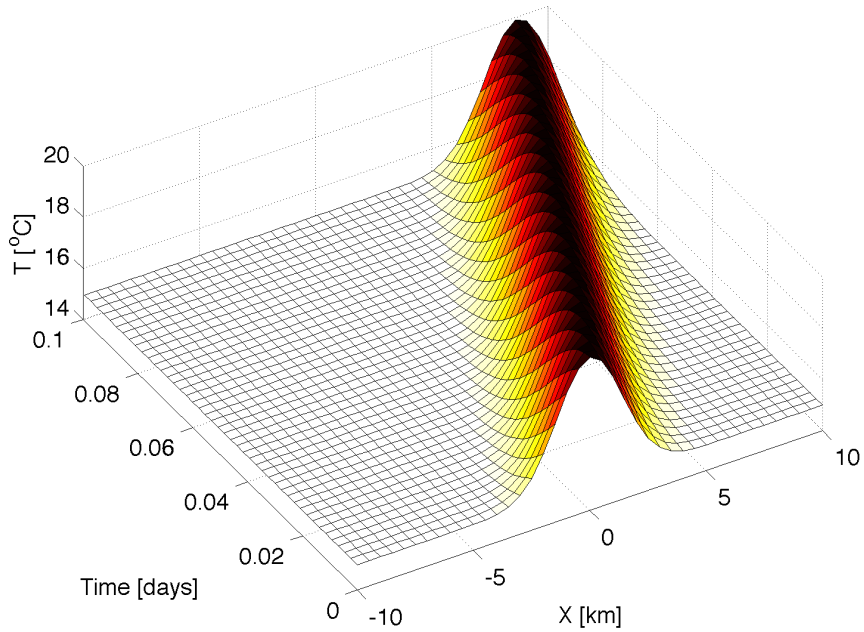
$$\theta = \theta(x, t) = \theta(x - ct)$$

- Mathematically, we are using a variable substitution: we say that the temperature function θ is regulated by a new independent variable $\chi = x - ct$, $\theta = \theta(\chi)$, which is a linear combination of the basic independent variables
- By using the chain rule to derive the function of a function, we can verify that this function is a solution of the PDE

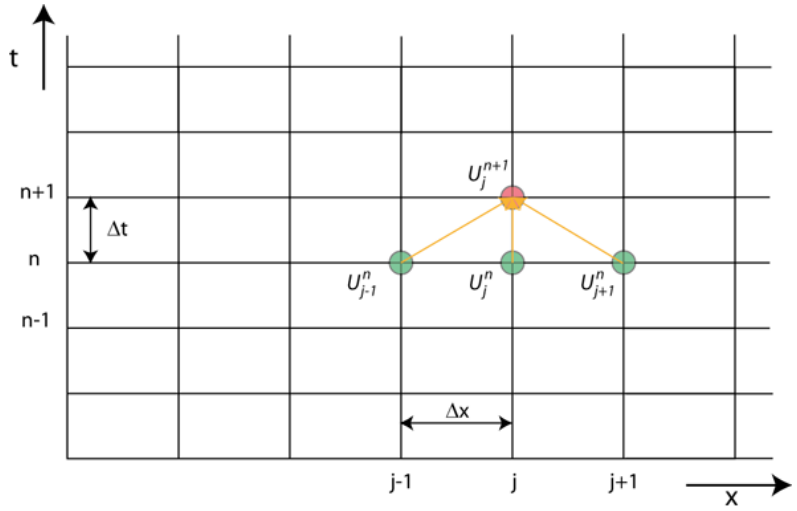
$$\frac{\partial \theta}{\partial t} = \frac{\partial \theta}{\partial \chi}\frac{\partial \chi}{\partial t} = -c\frac{\partial \theta}{\partial \chi}$$
$$\frac{\partial \theta}{\partial x} = \frac{\partial \theta}{\partial \chi}\frac{\partial \chi}{\partial x} = \frac{\partial \theta}{\partial \chi}$$

Exercise 10 : display the analytical solution

- Run the following script using python: `advection_analytical.py` (note: this is not a jupyter notebook)
- Change it into a notebook and show what happens when
  1. you change the values of dx and c
  2. you increase or decrease the parameter σ
- To learn more about animations with matplotlib:



The grid for the advection equation



This is the stencil that shows how the solution in time depends on the space variable. A quite natural choice for the numerical schemes is to use a forward scheme in time and a centered scheme in space (FTCS). In this example, you should substitute U with our temperature variable θ and j with i (i is more natural for indicating the index along the x-axis. We choose equally spaced points along the x and t dimensions

$$x_i = x_0 + i\Delta x; \;\; t_n = t_0 + n\Delta t$$

and we seek for the solution $\theta_i^n = \theta\left(x_i, t_n\right)$

Derivation of the discrete advection equation

$$\frac{\partial \theta}{\partial t} + c\frac{\partial \theta}{\partial x} = 0 \tag{35}$$

We have several choices for the derivative term, but the most obvious way is the explicit Euler forward for time integration

$$\frac{\partial \theta}{\partial t} = \frac{\theta_i^{n+1} - \theta_i^n}{\Delta t} + \mathcal{O}(\Delta t) \tag{36}$$

The advantage of this scheme is that it is explicit: one only needs the known quantities at time step n to compute the solution at $n+1$. For the space derivative we use the explicit centered-difference scheme

$$\frac{\partial \theta}{\partial x} = \frac{\theta_{i+1}^n - \theta_{i-1}^n}{2\Delta x} + \mathcal{O}\left(\Delta x^2\right) \tag{37}$$

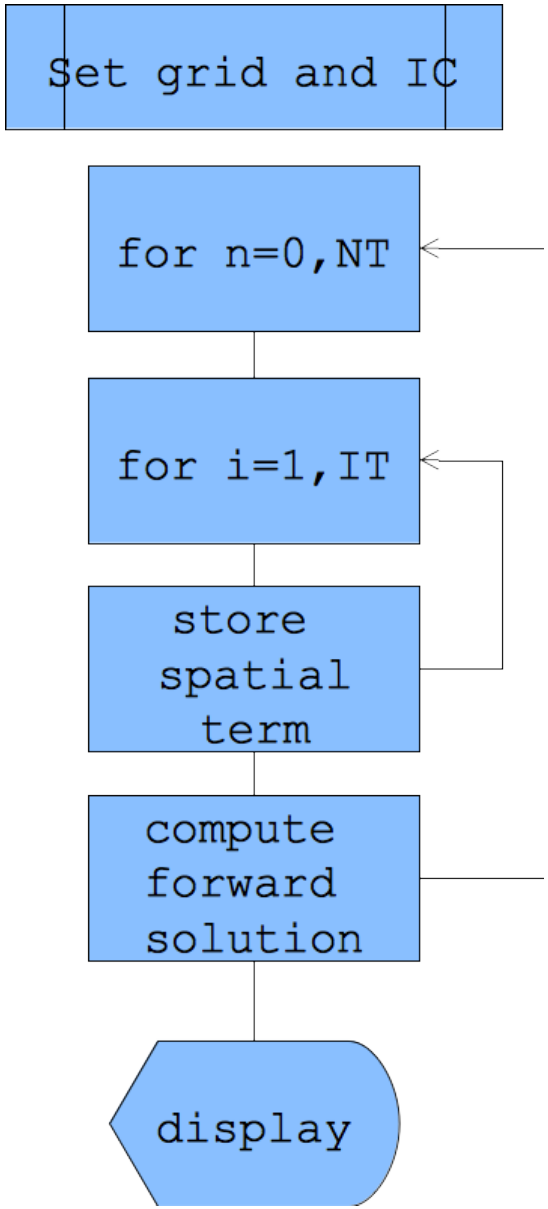
By combining eqs. (36) and (37) we obtain the complete discrete form

$$\frac{\theta_i^{n+1} - \theta_i^n}{\Delta t} = -c\frac{\theta_{i+1}^n - \theta_{i-1}^n}{2\Delta x}$$

which can be rearranged to get the forward-in-time value θ_i^{n+1}

$$\theta_i^{n+1} = \theta_i^n - c\frac{\theta_{i+1}^n - \theta_{i-1}^n}{2\Delta x}\Delta t$$

A basic algorithm for the numerical scheme



Is that all?

- One may think that by having the algorithm, we now have all the ingredients to go to the computer room, code it and get a "good solution". And, obviously, the computer will give some results if we implement the previous algorithm...

1. Is the code efficient or does just work?
2. Will our numerical solution be meaningful? $\Longrightarrow$ Convergence
3. Can we improve it by choosing other finite difference schemes for our partial derivatives? $\Longrightarrow$ Consistency - Order and accuracy
4. How can we choose values for Δx and Δt? $\Longrightarrow$ Stability analysis

Stability: the Courant number

By rearranging the numerical solution of the FTCS scheme

$$\theta_i^{n+1} = \theta_i^n - c\frac{\theta_{i+1}^n - \theta_{i-1}^n}{2\Delta x}\Delta t$$

we can observe that the value of the function at θ_i^n is modified by the gradient through the action of a non-dimensional number

$$\theta_i^{n+1} = \theta_i^n - c\frac{\Delta t}{\Delta x}\frac{\theta_{i+1}^n - \theta_{i-1}^n}{2} \tag{38}$$

$$\mathcal{C} = c\frac{\Delta t}{\Delta x} = \frac{c}{\Delta x/\Delta t}$$

which is called the **Courant number** (or Courant-Friedrichs-Lewy CFL condition). This is the ratio between the current speed and what we may call the "speed" that can be captured by the discrete numerical scheme.

In the case of the upstream or downstream spatial schemes, we also find the Courant number:

$$\theta_i^{n+1} = \theta_i^n - \mathcal{C}\left(\theta_i^n - \theta_{i-1}^n\right);\ \theta_i^{n+1} = \theta_i^n - \mathcal{C}\left(\theta_{i+1}^n - \theta_i^n\right)$$

Stability: significance of the Courant number

A scheme is stable if the solution is meaningful. This means that the advection should not take away from the grid box an amount of the quantity that is larger than what is available. Numerical errors may create *overshooting* and *undershooting*, which means that more temperature is advected away from one side of the cell than it is replaced by the same process on the other side. We are thus seeking for the positiveness of the solution (see the slide on page 36), and write this mathematically as $\theta_i^{n+1} \geq 0$. Using the FTCS solution we obtain:

$$\theta_i^n - \mathcal{C}\frac{\theta_{i+1}^n - \theta_{i-1}^n}{2} \geq 0$$

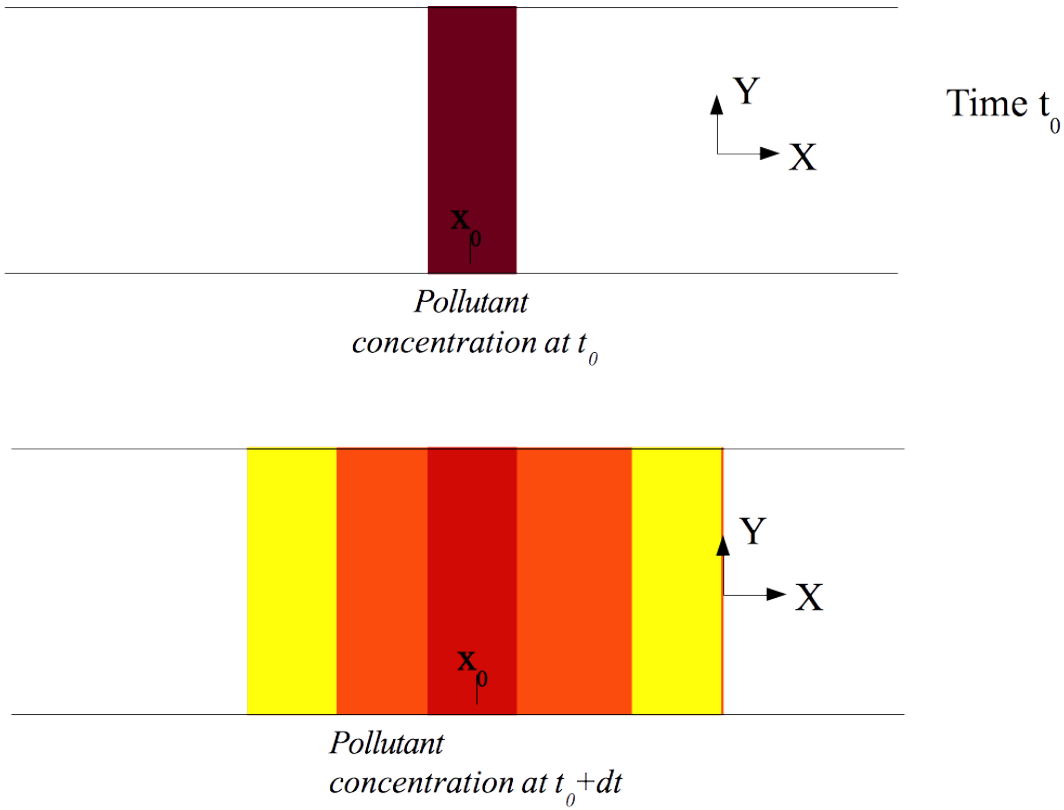
which has to be true for every value of the gradient. Since we cannot control the gradient value, the **Courant number must be close to 0**, to ensure positiveness of the solution (note that this does not mean that the temperature cannot become negative if it would physically do so).

Exercise 11

The numerical solution of the advection problem is implemented in the notebook `advection.ipynb`. Familiarize yourself with the code and answer the following questions in a modified version of the notebook (make sure that you use headers to distinguish the various parts):

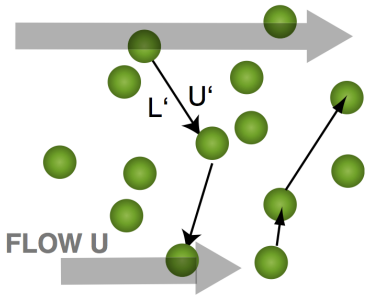
1. Look for the comment `#Question 1` in the code. What do you think the next two lines do? [hint: see slide on page 56]
2. Look for `#Question 2` in the code. Which component of the numerical scheme is implemented in line `#L1`? What does line `#L2` do?
3. Look for `#Question 3` in the code. Why the code does not perform the computation on all the elements of the array `T`, i.e. `T[:,n]`?
4. Analysis of the numerical solution. How different is the numerical solution from the expected moving perturbation? What is the possible reason?
5. The grid parameters determine the value of the Courant number and hence the stability. The spatial and temporal steps are linked. More spatial resolution (smaller dx) means that dt must also decrease to maintain the same Courant number. However, things are not so simple. The higher spatial resolution (which should theoretically increase the Courant number) also enhance the capability to resolve the real gradients (see the previous slide to understand how $\mathcal{C}$ interacts with the difference between adjacent points). If you increase the Courant number to 1 by halving dx, what does it happen? Now set $dx = 500$ m and $dt = 100$ s to get $\mathcal{C} = 0.1$; what do you see and how would you interpret it?
6. The choice of the grid parameters also depend on the features that you want to resolve. Set the parameter $\sigma = 1000$ m (a steeper wave). What do you have to do to recover the same degree of accuracy of the previous solution? (note that it takes longer to generate the movie when you decrease the time step because NT increases and more frames are generated). Make another copy of the original notebook and double the current speed. How would you explain the result?

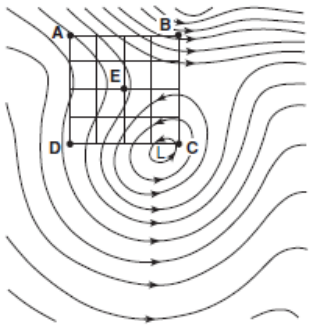
A diffusion problem in the estuary

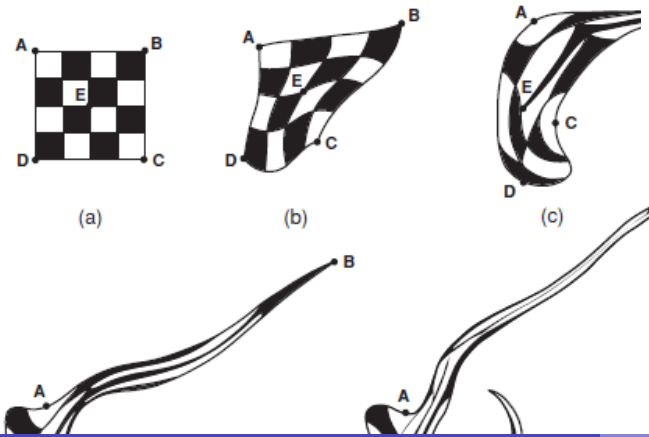


This case is similar to the heat anomaly advection. In this case we assume that the spill is coming from a sewage treatment pipe with a very high dissolved inorganic nitrogen concentration (DIN, measured in mmol m^{-3})

Reminder: molecular and eddy viscosity/diffusivity







- **Friction** in a moving fluid results from the transfer of **momentum** between different parts of the fluid. The physical process that "transmit" momentum is called **viscosity**, which can be seen as the resistance of the fluid to motion. In a laminar flow, momentum exchange occurs as a result of the transfer of molecules with different velocities between adjacent layers: **molecular viscosity**. The propension of the fluid to transfer heat and dissolved chemicals is the **molecular diffusivity**, which depends on the characteristic of both the fluid and the substance. Molecular viscosity and diffusivity are found to be proportional to the mean velocity of the molecules and the mean pathway. Kinematic viscosity and molecular diffusivity have units of $m^2\ s^{-1}$.
- **Turbulence occurs at scales above the molecular level**. Patches of fluids are stirred and diffused following internal eddies. The stirring generates new gradients that are dispersed due to molecular diffusion. **Eddy diffusivity** (of heat/substances) and **eddy viscosity** (of momentum) are parameterized similarly to the molecular equivalents but are much larger

	Molecular, at salinity = 35	Eddy: horizontal (along-isopycnal)	Eddy: vertical (diapycnal)
Viscosity	1.83×10^{-6} m^2/sec at 0 °C 1.05×10^{-6} m^2/sec at 20 °C	10^2 to 10^4 m^2/sec	10^{-4} m^2/sec
Thermal diffusivity	1.37×10^{-7} m^2/sec at 0 °C 1.46×10^{-7} m^2/sec at 20 °C	10^2 to 10^4 m^2/sec	10^{-5} m^2/sec
Haline diffusivity	1.3×10^{-9} m^2/sec	10^2 to 10^4 m^2/sec	10^{-5} m^2/sec

The discrete diffusion equation

$$\frac{\partial N}{\partial t} = \kappa \frac{\partial^2 N}{\partial x^2} \tag{39}$$

We use the same approach seen when discretizing the advection on page 75. We seek for a discrete solution N_i^n. The main difference is that we now have to write the approximation of the second order derivative seen in eq. (29)

$$\frac{\partial^2 N}{\partial x^2} = \frac{N(x_0 + \Delta x) - 2N(x_0) + N(x_0 - \Delta x)}{\Delta x^2} + O(\Delta x) \tag{40}$$

$$\left.\frac{\partial^2 N}{\partial x^2}\right|_{n,i} = \frac{N_{i+1}^n - 2N_i^n + N_{i-1}^n}{\Delta x^2} + \mathcal{O}(\Delta x) \tag{41}$$

We combine the Euler-forward time scheme with the second derivative

$$\frac{\partial N}{\partial t} = \frac{N_i^{n+1} - N_i^n}{\Delta t} + \mathcal{O}(\Delta t)$$

to obtain the discrete form

$$\frac{N_i^{n+1} - N_i^n}{\Delta t} = \kappa \frac{N_{i+1}^n - 2N_i^n + N_{i-1}^n}{\Delta x^2}$$

Stability of the discrete diffusion equation

By rearranging the numerical solution of the diffusion equation we get another non-dimensional number

$$N_i^{n+1} = N_i^n + \kappa \frac{\Delta t}{\Delta x^2} \left(N_{i+1}^n - 2N_i^n + N_{i-1}^n \right)$$

$$D = \kappa \frac{\Delta t}{\Delta x^2} = \frac{\kappa}{\Delta x^2 / \Delta t}$$

which is the ratio between the diffusivity of the real problem and what we may call the "diffusivity" of the discrete numerical scheme. This condition is actually less restrictive than the Courant number derived in the advection equation that must be close to 0 for stability. It can be demonstrated using the *von Neumann stability analysis* that $D \leq 1/2$ for conditional stability. But now that Δx is to the power of 2, it is much more tricky to maintain stability if we aim at high spatial resolution.

Exercise 12

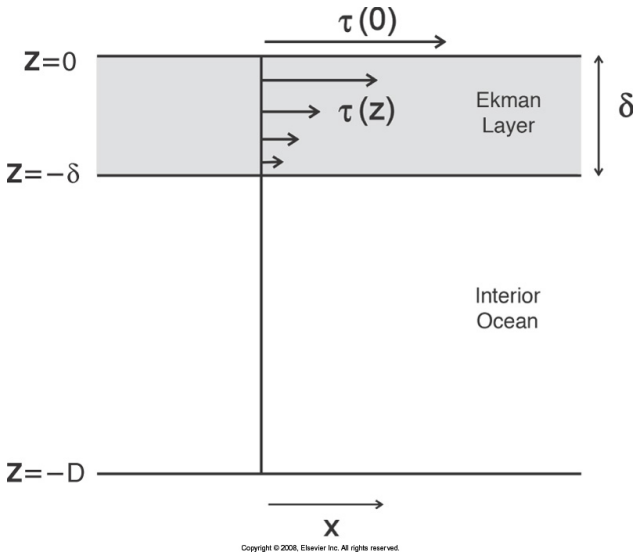
The numerical solution is similar to Ex. 11 implemented in the notebook `diffusion.ipynb`. Familiarize yourself with the code and answer the following questions in a modified version of the notebook:

1. Study the numerical solution. Why does the flattening of the peak slow down with time?
2. Look for the comment `#Question 2` in the code. This is a different type of boundary conditions called *Neumann conditions* (not to be mistaken with *von Neumann*). How would you describe them? Change the boundary conditions to the ones used in the advection equation (*Dirichlet boundary conditions*, where the value at the boundary is prescribed). Run this code and compare it with the original: why do you think the Neumann boundary conditions are more realistic?
3. The grid parameters determine the stability of the numerical solution and its evolution, based on the physical parameters. In this case, the key parameter is the eddy diffusivity coefficient, which indicates the turbulence of the fluid and how quickly it dissipates the signal. What happens if you double the diffusivity coefficient? [you can adjust the y-axis limits to see the behaviour better]. What would you do to fix it and why?

Reminder: the Ekman layer

$$\frac{D\vec{u}_H}{Dt} = -\frac{\nabla p}{\rho_0} - f\left(\hat{k} \times \vec{u}_H\right) + A_v \nabla^2 \vec{u}_H$$

- Ekman studied the balance of forces in the upper layer by making some major assumptions on the horizontal Navier-Stokes (Boussinesq) equations
  - homogeneous density
  - the surface ocean is in steady state
  - the velocity field only changes in the vertical and the vertical eddy viscosity A_v is constant
  - there is no horizontal pressure gradient
  - the momentum flux at the surface is equal to the wind stress $\left(A_v \frac{\partial u}{\partial z}\right)_{z=0} = \frac{\tau_x}{\rho_0}$; $\left(A_v \frac{\partial v}{\partial z}\right)_{z=0} = \frac{\tau_y}{\rho_0}$
  - the momentum flux at a certain depth $-\delta$ is zero, $\left(A_v \frac{\partial u}{\partial z}\right)_{z=-\delta} = \left(A_v \frac{\partial v}{\partial z}\right)_{z=-\delta} = 0$



The resulting equation describes a special velocity field $\vec{u}_E$ that balances the Coriolis acceleration with the frictional acceleration

$$f\hat{k} \times \vec{u_E} = A_v \frac{\partial^2 \vec{u}_E}{\partial z^2}$$

The Ekman velocity profile

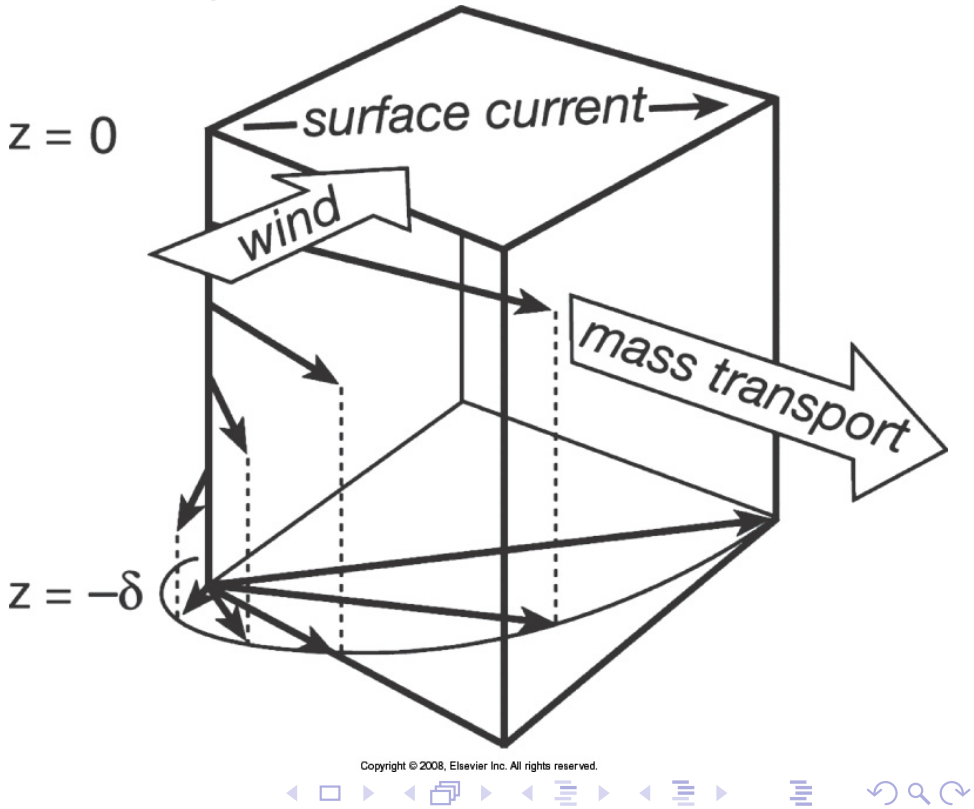
- The resulting equation describes a velocity field that balance the Coriolis acceleration with the (turbulent) frictional acceleration

$$f\hat{k} \times \vec{u_E} = A_V \frac{\partial^2 \vec{u}_E}{\partial z^2}$$

- It shows how the velocity changes with depth in the Ekman layer and it can be written as an ODE system, since there is only one independent variable z

$$\begin{aligned} -fv_E &= A_V u_E'' \\ fu_E &= A_V v_E'' \end{aligned}$$

This system has an analytical solution that was proposed by Ekman (a combination of exponential and trig functions) that describe the shape of the Ekman spiral



Reminder: Ekman transport

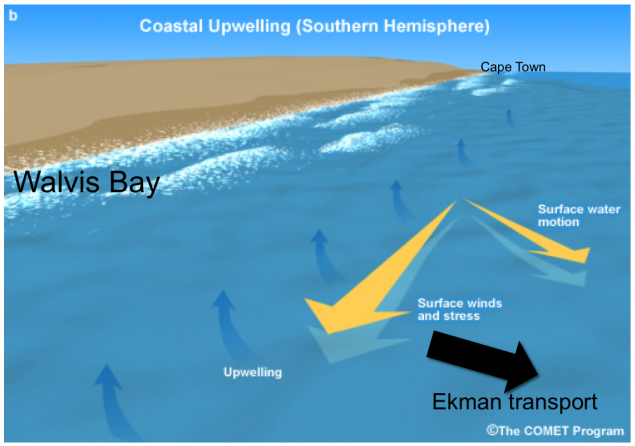
- The Ekman transport is the vertical integral of the velocity in the Ekman layer (*remind yourself of the units*)

$$M_E^{(y)} = \int_{D_E} v_E(z)dz; \; M_E^{(x)} = \int_{D_E} u_E(z)dz$$

- Using the boundary conditions above, we can demonstrate that is proportional to the wind stress and exactly perpendicular to the right (left) of the wind in the Northern (Southern) hemisphere because the sign is given by the Coriolis parameter

$$f\hat{k} \times \int_0^{-\delta} \vec{u_E} dz = A_v \int_0^{-\delta} \frac{\partial^2 \vec{u_E}}{\partial z^2} dz$$

$$f\hat{k} \times \vec{M}_E = \frac{\vec{\tau_W}}{\rho_0}$$



$$M_E^{(x)} = \frac{\tau_W^{(y)}}{\rho_0 f}; \; M_E^{(y)} = -\frac{\tau_W^{(x)}}{\rho_0 f}$$

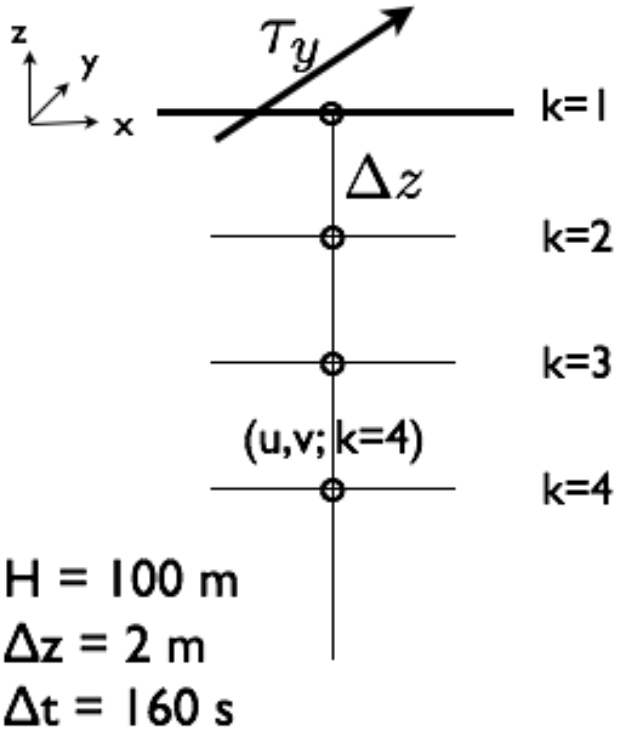
The numerics of the dynamical Ekman layer

- The equations for the Ekman velocity represent a system of diffusion PDEs, one for $u_E(z,t)$ and one for $v_E(z,t)$, with the **addition of a source term** due to the Coriolis acceleration. Numerically, we can solve the full set of dynamical equations, not just the steady state

$$\frac{\partial u_E}{\partial t} - f v_E = A_v \frac{\partial^2 u_E}{\partial z^2} \qquad (42)$$

$$\frac{\partial v_E}{\partial t} + f u_E = A_v \frac{\partial^2 v_E}{\partial z^2}$$

The vertical grid starts at the surface with $z_0 = 0$ and $z_k = z_0 - k\Delta z$; time is discretized with $t_i = t_0 + n\Delta t$. We seek for the solutions u_k^n and v_k^n.



Exercise 13 : the discretized Ekman equations

1. Write the discretized form of the equations (42) for the zonal and meridional components using the grid shown in the previous slide.
2. Derive the non-dimensional Courant number that depends on the vertical viscosity
3. Using the vertical eddy viscosity shown on page 83, what would be your choice of Δz and Δt to ensure stability?

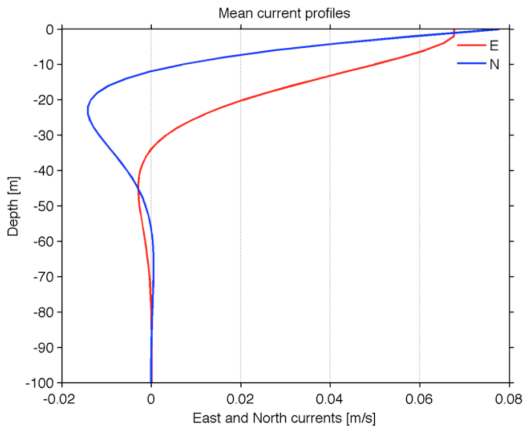
Collaborative project: code porting and writing

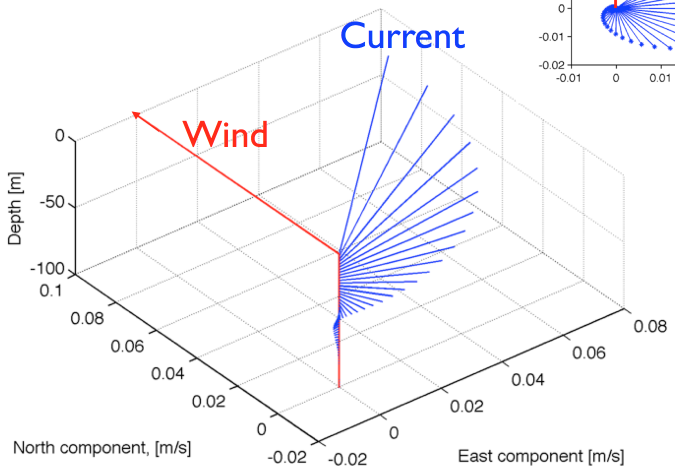
- In ocean modelling, as in any other discipline that relies on scientific computing, it makes sense to reuse code produced by others and make it your own. However, people use different types of computer languages and you will often have to *port* (translate) the code in another language
- Working in a team is the best way to learn coding, since you will have to confront each other on the best way to write the various parts and whether it is understandable and/or efficient (there is often more than one way). You do not need to create a very efficient code, but the whole team needs to understand how it works.
- Your collaborative project will be the porting of a MATLAB code that solves the Ekman equation in python. The code is available in the file ekman_spiral.m. A good reference for comparing numpy and MATLAB commands is http://mathesaurus.sourceforge.net/matlab-numpy.html.
- You will decide how to
  - manage your team (nominate a team leader(s), create groups of software designers and programmers, etc...)
  - design and produce the software (which IDE to use, create a notebook or not, how to handle different versions, etc...)
  - present the results (choose the colors and the attributes of the figures, create an animation or a series of static figures, decide the main example for the user, etc.)
- **The final deliverable will be the python code and a brief presentation showing how to use the code and what it produces**

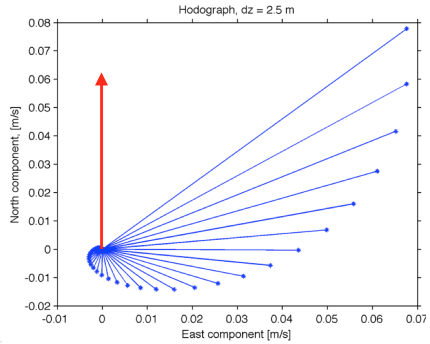
The MATLAB code output in the Northern Hemisphere

- The zero-crossing of the north component is at about 15 m
- The depth of the Ekman layer can be derived as the e-folding depth scale of the spiral

$$\delta_E = \sqrt{2A/f}$$







What is an Ocean Model?

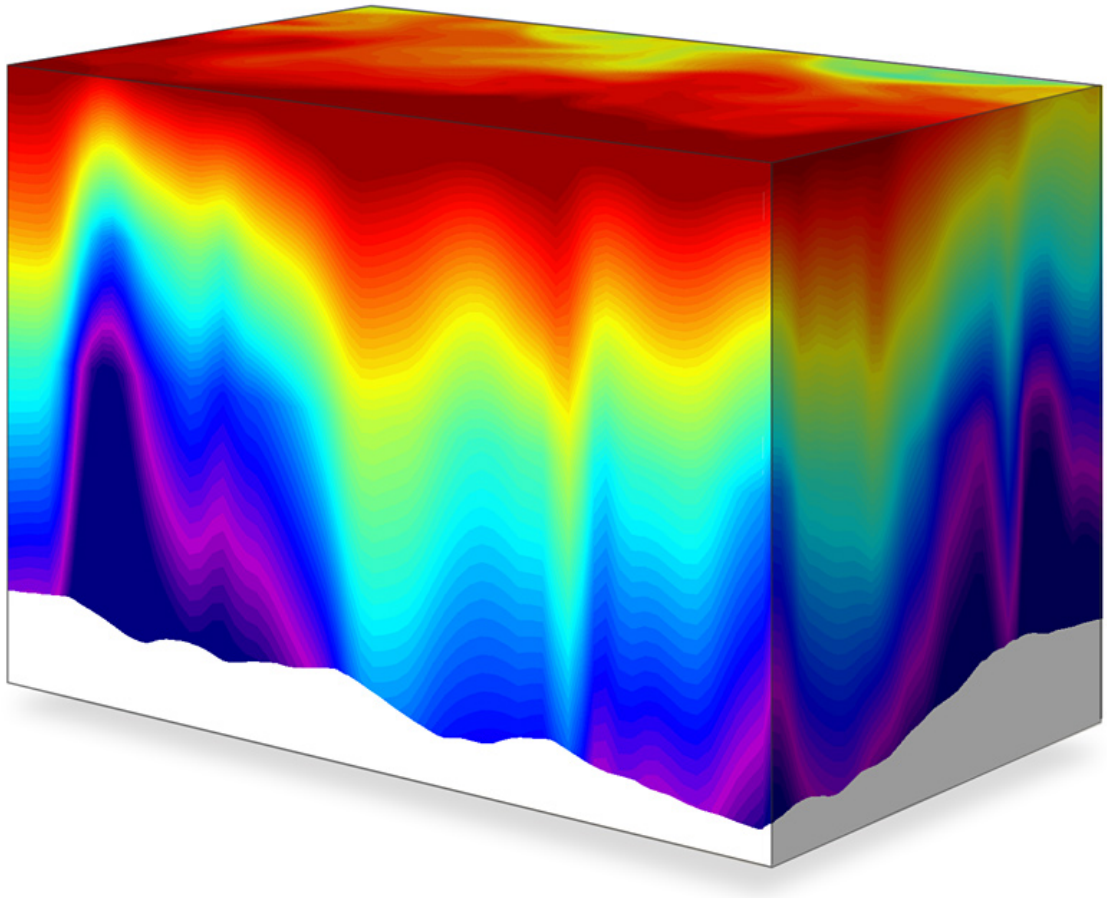


Figure: Source: *http : //www.mercator − ocean.fr/fre/mercator − ocean/*

Some oceanic physical processes:

- Ocean circulation (Thermohaline, gyres, currents, jets, filaments,...)
- Turbulence, geostrophic eddies
- Waves (Tsunamis, Kelvin waves, Rossby waves, Gravity waves, internal waves, ..etc)
- Tides
- Upwelling and Downwelling
- Transport (Mass, Volume and Energy)
- Mixing (Lateral and vertical), stirring, diffusion

Some oceanic physical processes:

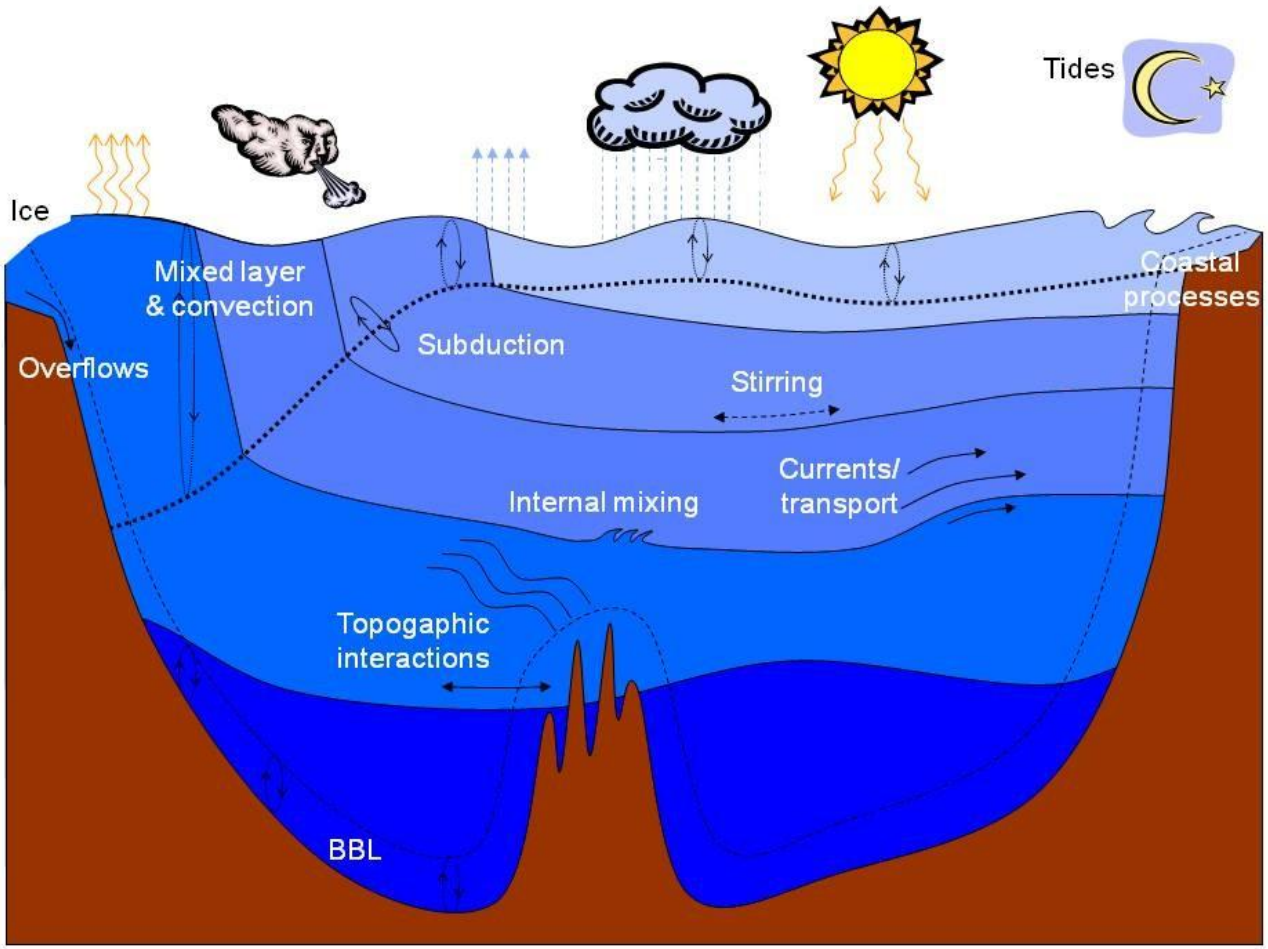


Figure: http://www.gfdl.noaa.gov/oceanmodelsatgfdl.

Some oceanic biogeochemical processes

- Phytoplankton blooms
- Harmful algal blooms
- Hypoxia and anoxia
- Nutrient upwelling
- (Green House) Gas exchange at the air-sea interface
- Deep water ventilation
- Organic material transfer through the food web and export
- Ocean carbon cycle

Some oceanic biogeochemical processes

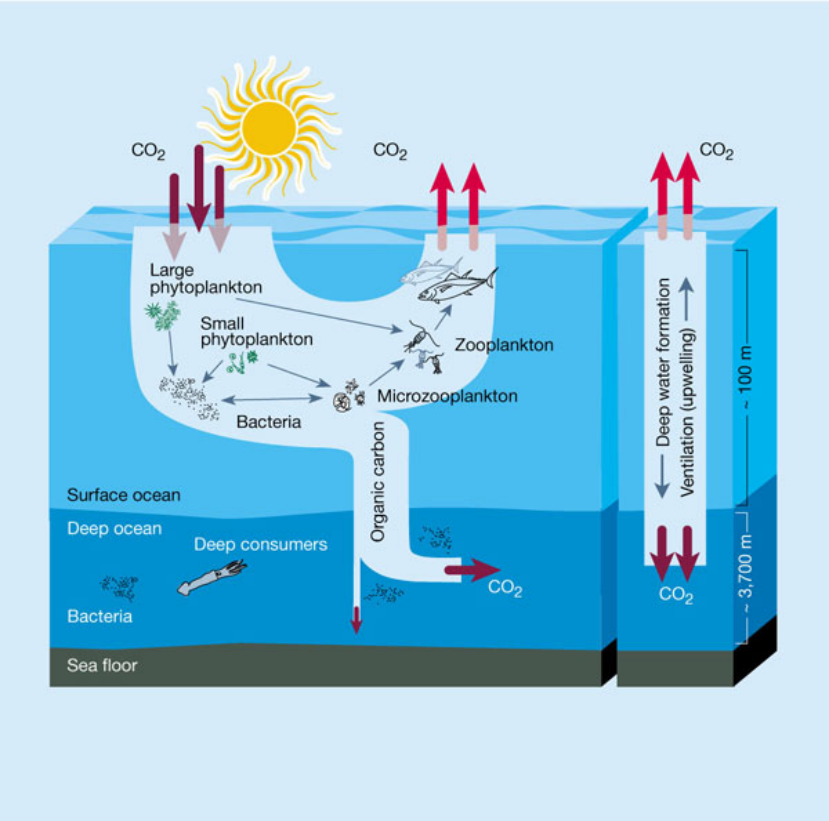


Figure: Chisolm, Nature 407, 685-687 (2000) doi:10.1038/35037696

Scales and processes

- Models cannot simulate all possible scales and choices must be made to ensure sufficient realism while preserving feasibility
- Everything smaller than your grid scale must be parameterized (sub-grid scale)
- The parameterizations assumptions and the modeled spatial and temporal scales are the first things to be decided

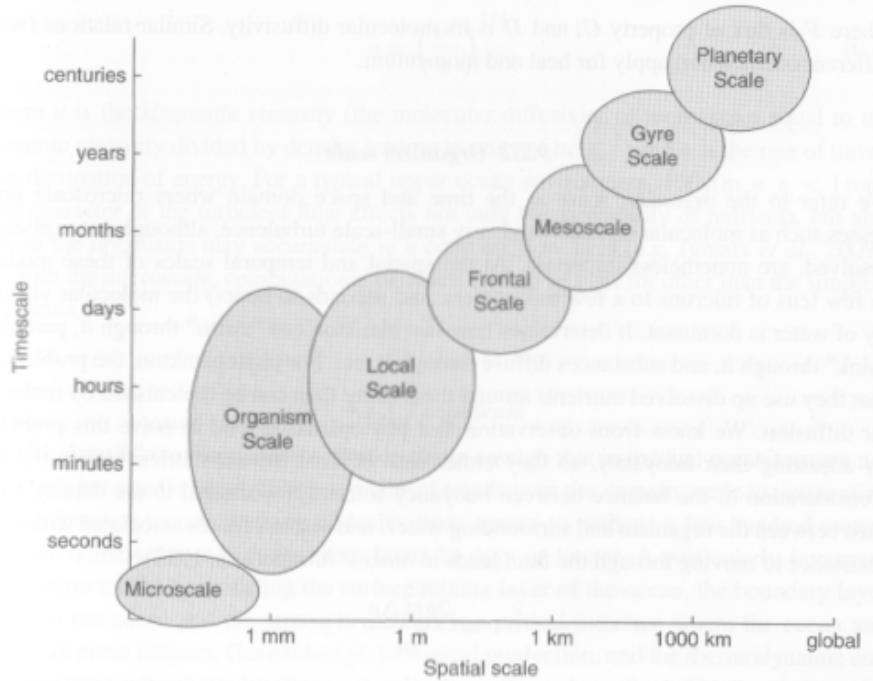


Figure 9.1 A schematic relationship between space- and timescales for various ocean processes.

Some oceanic processes, and their Characteristic Scales

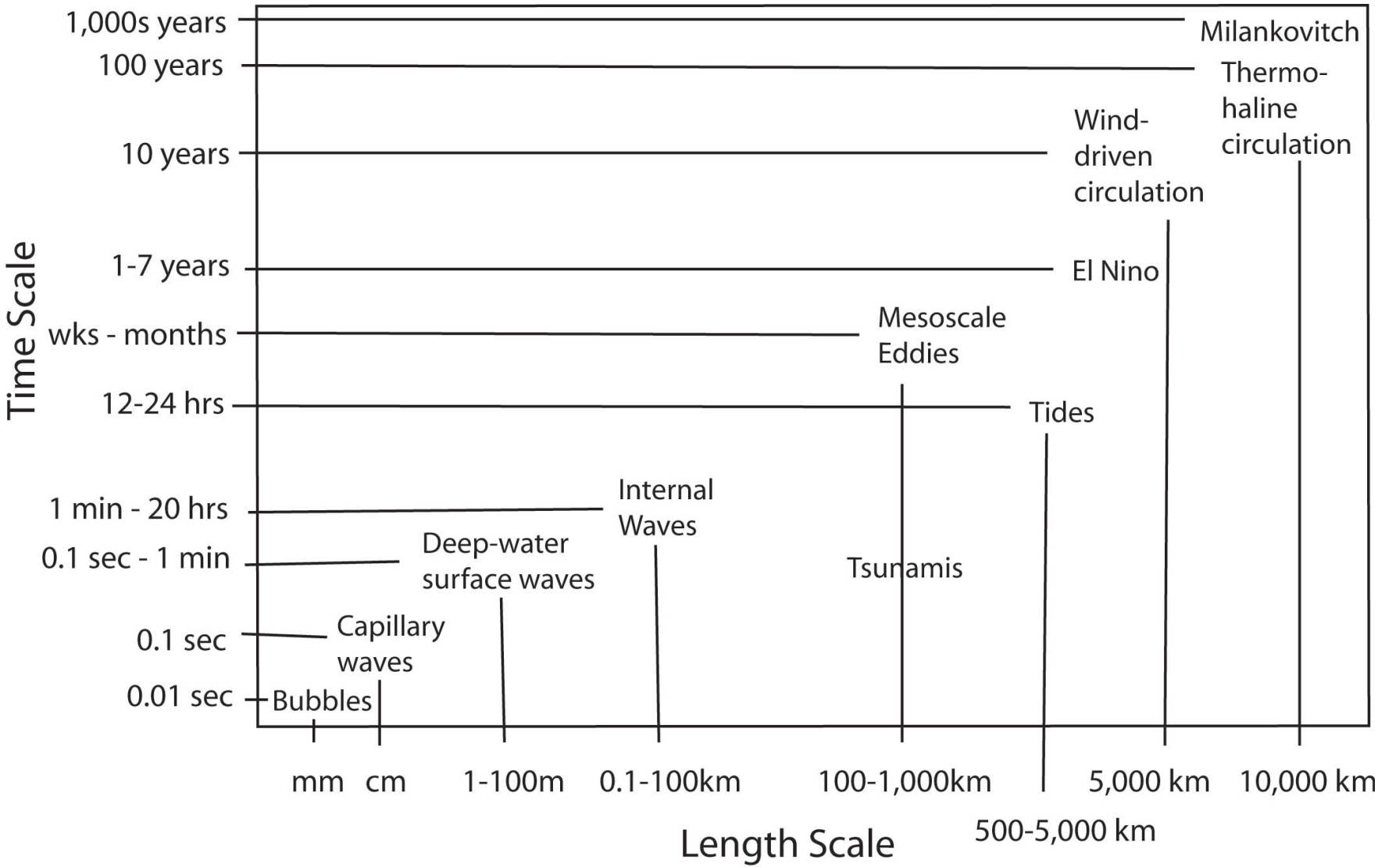
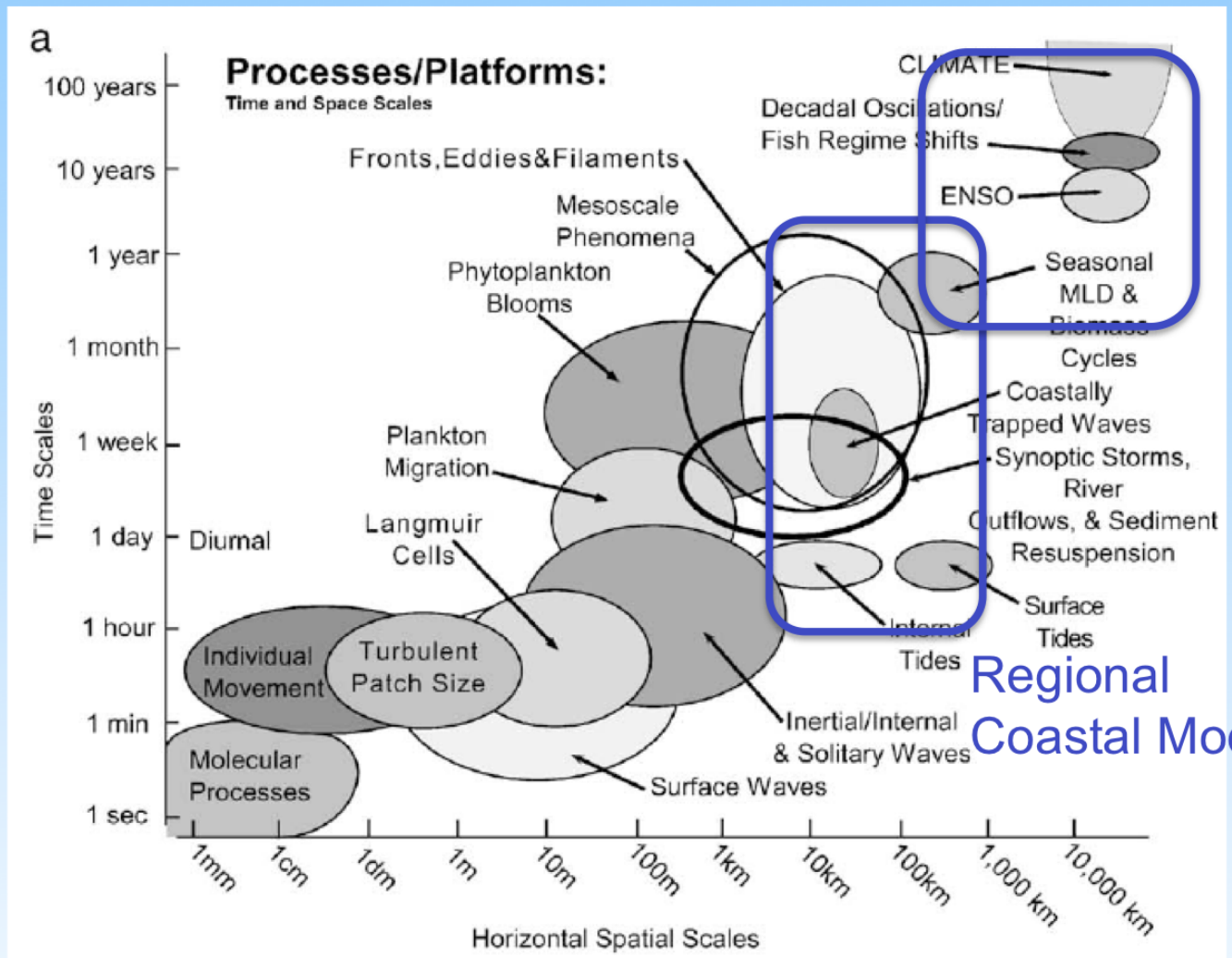


Figure: Source: nerd.ucsd.edu

Scales and models



Use of numerical models

Some Advantages:

- Desirable spatial and temporal coverage.
- Regular sampling of the variables.
- Synoptic information of a wider-range of the oceanic parameters throughout the ocean column (it allows better understanding).

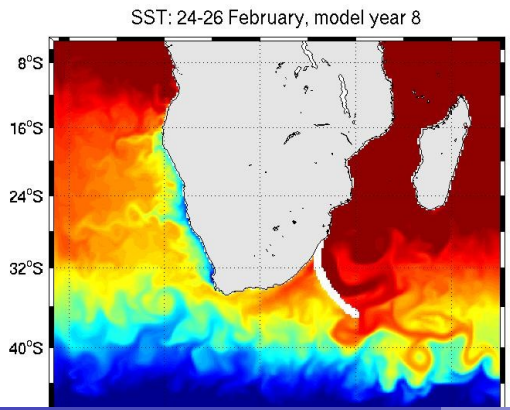
Why use numerical models?

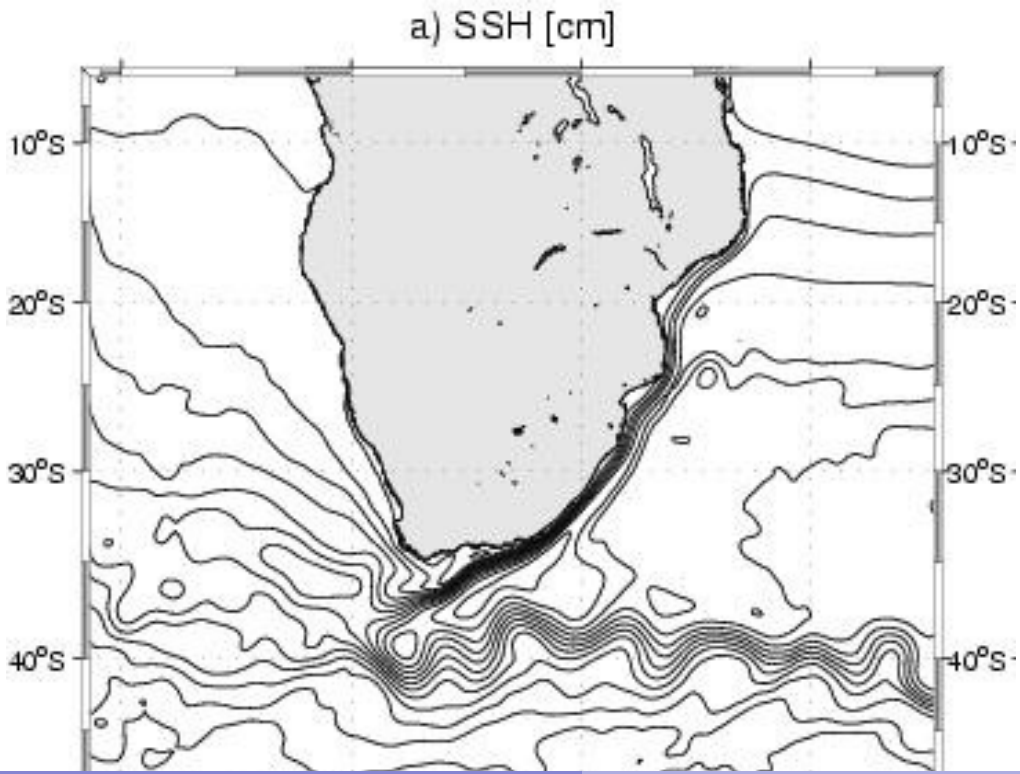
Some Advantages:

- Allows to diagnose specific process (e.g: Particular form of instability, or role of friction under particular conditions, species dominance under upwelling conditions, etc)
- Allows to test hypothesis (e.g: what if, removal of components or processes, etc).
- Allows to reconstruct the past (hindcast), fill in gaps in ebservations (nowcast), make short term predictions (forecast) or long term predictions (scenarios).

Applications of Ocean Models around Southern Africa

- Allows to diagnose specific processes (e.g: Warming of the Agulhas Current since 1980, due to the augmentation of the transport in response to the increased wind stress curl)., Rouault et al. 2009
- Allows to test hypothesis, perform idealized experiments (e.g: No Agulhas Current (Chang et al., 2008; Veitch et al. 2009)., No Madagascar (Penven et al. 2006).





Numerical models

Some Challenges:

- The numerical equations driving the model are approximations to the continuous analytic equations that describe fluid flow.
- Models contain no information about flow between grid points (descrete information).
- Furthermore, Not every process in the ocean is understood and represented by a mathematical formulation.
- Parametrizations of Subgrid-scale processes (Unresolved processes).
- Numerical errors (e.g: Truncation and rounding numbers).
- Model results are NOT REALITY! (Just good approx of reality)

HISTORY OF OCEAN MODELS

- First Oceanic General Circulation Model (OGCM) is credited to Kirk Bryan at GFDL (late 1960's).
- Special efforts on the evolution of this model type by Mike Cox & Bert Semtner (mid 1970's).
- Global Z - Vertical coordinate model (isopycnal models emerged during the 1970's).
- Low-order numerics with finite difference schemes.

Ocean Models: Evolution of an Ocean Model

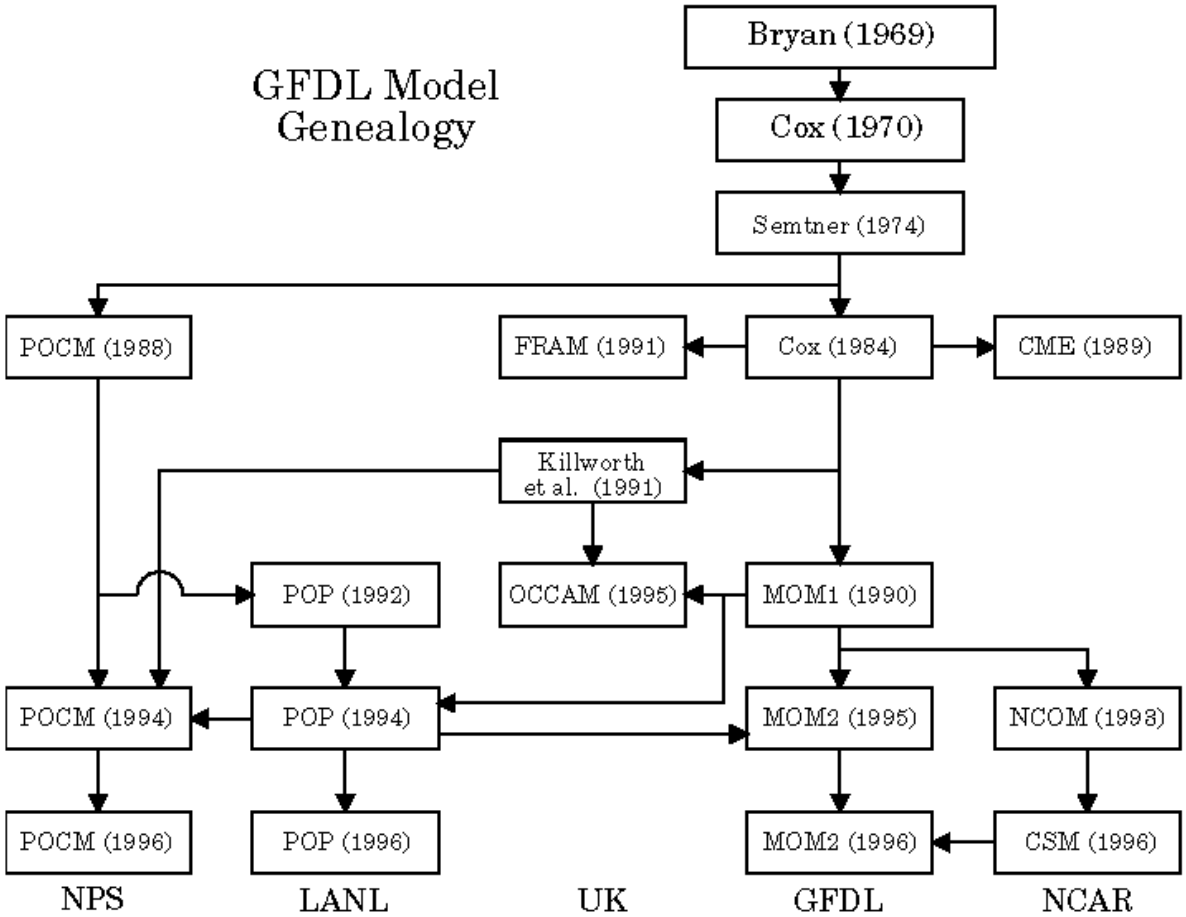


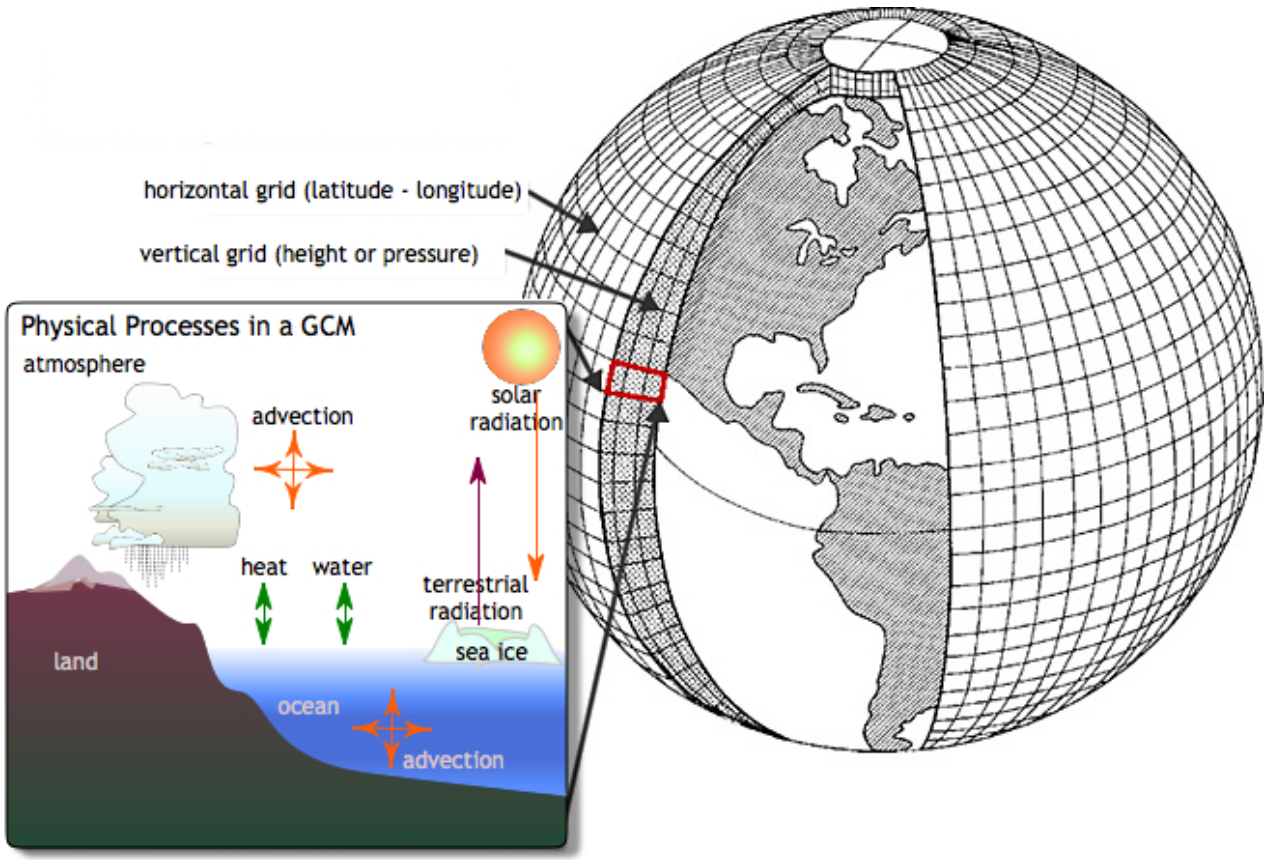
Figure: http://climate.lanl.gov/Models/POP/download

Why so many different models?

Acronym	Description
ADCIRC	Advanced CIRCulation model
BOM	Bergen Ocean Model
COHERENS	Coupled Hydrodynamical Ecological model for RegioNal Shelf seas
DCMGMB	Dartmouth Circulation Models for the Gulf of Maine and Georges Bank
DieCAST	Dietrich Center for Air Sea Technology
ECOM-si	Estuarine Coastal and Ocean Model (semi-implicit)
ELCIRC	Eulerian-Lagrangian Circulation
FEOM	Finite Element Ocean Modelling
FRAM	Fine Resolution Antarctic Model
FMS	Flexible Modelling System
FVCOM	Finite Volume Coastal Ocean Model
GOTM	General Ocean Turbulence Model
HIM	Hallberg Isopycnal Model
HOPE	Hamburg Ocean Primitive Equation General Cieculation Model
HOPS	Havard Ocean Prediction System
HYCOM	Hybrid Coordinate Ocean Model
LOAM	Lamont Ocean-AML Model
LSG	Hamburg Large Scale Geostrophic Ocean General Circulation Model
MICOM	Miami Isopycnic Coordinate Ocean Model
MITgcm	M.I.T. General Circulation Model
MOHID	Modelo Hidrodinamico
MOM	GFDL Modular Ocean Model
NEMO	Nucleus for European Modelling of the Ocean
NCOM	NCAR CSM Ocean Model
NLOM	Navy Layered Ocean Model
OCCAM	Ocean Circulation Climate Atmospheric Model
OFES	OGCM for Earth Simulator
OPA	Ocean Parallellis
OPNML	Ocean Processes Numerical Modelling Laboratory
OPYC	Ocean IsoPycnal General Circulation Model
PEQMOD	Primitive Equation Model
POCM	Parallel Ocean Circulation Model
POGCM	Poseidon Ocean General Circulation Model
POM	Princeton Ocean Model
POP	Parallel Ocean Program
POSUM	Parallel Oregon State University Model
PSTSWM	Parallel Spectral Transform Shallow Water Model
QTCM	Quasi-equilibrium Tropical Circulation Model
ROMS	Regional Ocean Modelling System
SCRUM	S-Coordinate Rutgers University
SEA	Southampton East Anglia Parallel Ocean Ciirculation Model
SPEM	S-Coordinate Primitive Equation Model
STSWM	NCAR Spectral Transform Shallow Water Model

Classification of Ocean Models

Models are classified according to the different approaches used to represent the Horizontal grid and the Vertical coordinate system.



Main Criteria of Classification

- Geography (domains)
- Physical process
- Surface approximation
- Density variation
- Discretization of the vertical coordinate
  (Representation of the vertical structure of oceanic water column)

Geographic classification: Domain size

- Global-scale
- Basin-scale
- Regional or local-scale

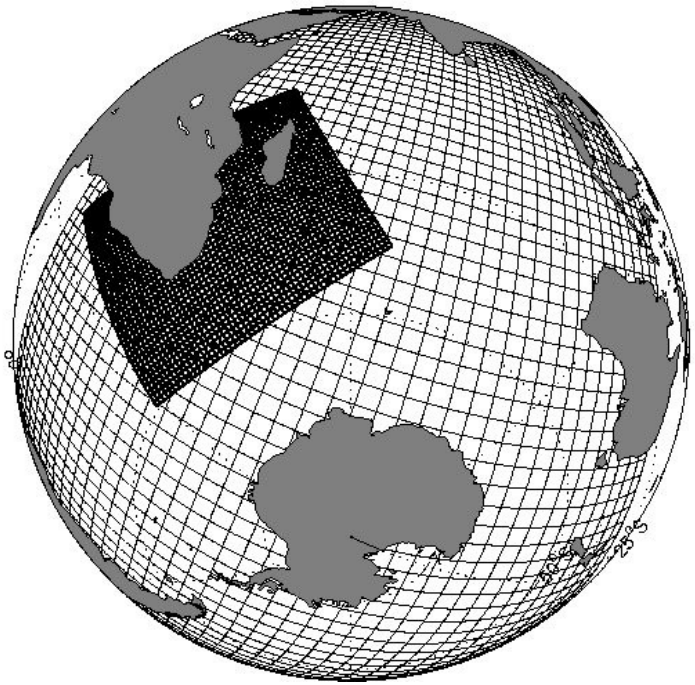
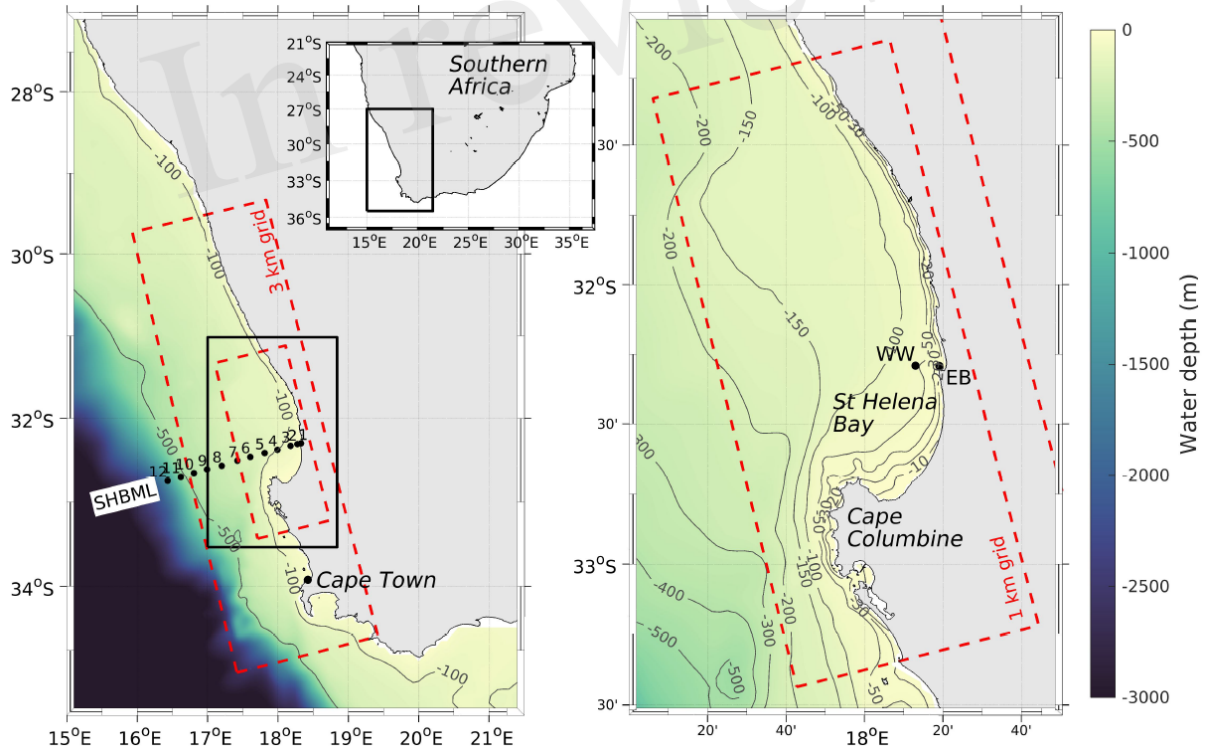


Figure: Backeberg et al., 2009

Nested grids

Regional models are often built with a combination of nested grids, from a parent grid at coarser resolution to children at higher resolutions. All regional models need boundary conditions. The boundary can be passive (one-way coupling) or active (two-way coupling).



(Fearon et al., 2023)

Surface approximation (representation of the sea surface)

All finite difference numerical models are constrained by the timestep. Advective and wave-like processes must not move more than one grid point per timestep (CFL criterion). Surface processes are the fastest

- **Rigid-Lid models** (Ocean-atmosphere interface doesn't evolve) To avoid severe limitations on the time-steps due to fast gravity waves (surface gravity waves are neglected from the system, and retaining only the horizontal pressure gradient due to the large-scale circulation.
- **Implicit surface models:** A smoothed solution of the surface that retains the influence of the faster waves but not their resolution
- **Free-Surface models:** The height of the free-surface (SSH) is driven up or down by convergence/divergence of the fluid throughout the underlying water column. Slow modes and fast modes are treated separately. Gravity waves are still not resolved explicitly but parameterized, or obtained from external wave models.

Rigid-Lid approximations

Advantages:

- With less constraint on the permissible timesteps, it was possible to use steps of order several hours and a very coarse resolution.

Disadvantages:

- It increases the surface wave speed to infinity, and modifies certain other long-wave dynamics (e.g: Barotropic Rossby or Planetary waves).
- Barotropic flow had not been removed, rather it had been modified! So to calculate it, a two-dimensional stream function had to be calculated. In case of large domains, finer grids, complicated bottom topography and varying coastlines the computation was too expensive.

Surface approximation

In reality the ocean surface is free to deform, under the influence of winds, heating, and tidal forces. It has fast moving and short-lived waves. These needs to be retained!

- A prognostic equation for free surface height (SSH) is obtained by integrating the non-divergent continuity equation in the vertical from the bottom $z{=}{-}H(x,y)$, to the top at $z{=}\eta$(x,y,t):

$$\eta = \int_{-H}^{\eta} \frac{\partial w}{\partial z} dz = -\int_{-H}^{\eta} \nabla_h . \vec{v} dz \tag{43}$$

Density variation

In general ocean models describe the response of a variable density ocean to atmospheric momentum and heat forcing (such response can be represented by Barotropic and Baroclinic modes.)

- Barotropic models: The ocean's water column is vertically integrated to obtain one value for vertically different horizontal currents.
- Baroclinic models: The ocean's column has variable horizontal flows with depth.

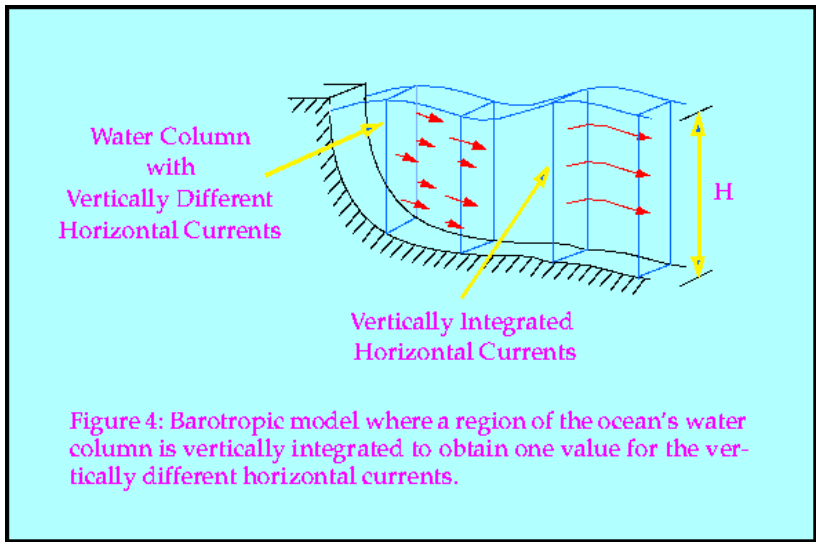


Figure 4: Barotropic model where a region of the ocean's water column is vertically integrated to obtain one value for the vertically different horizontal currents.

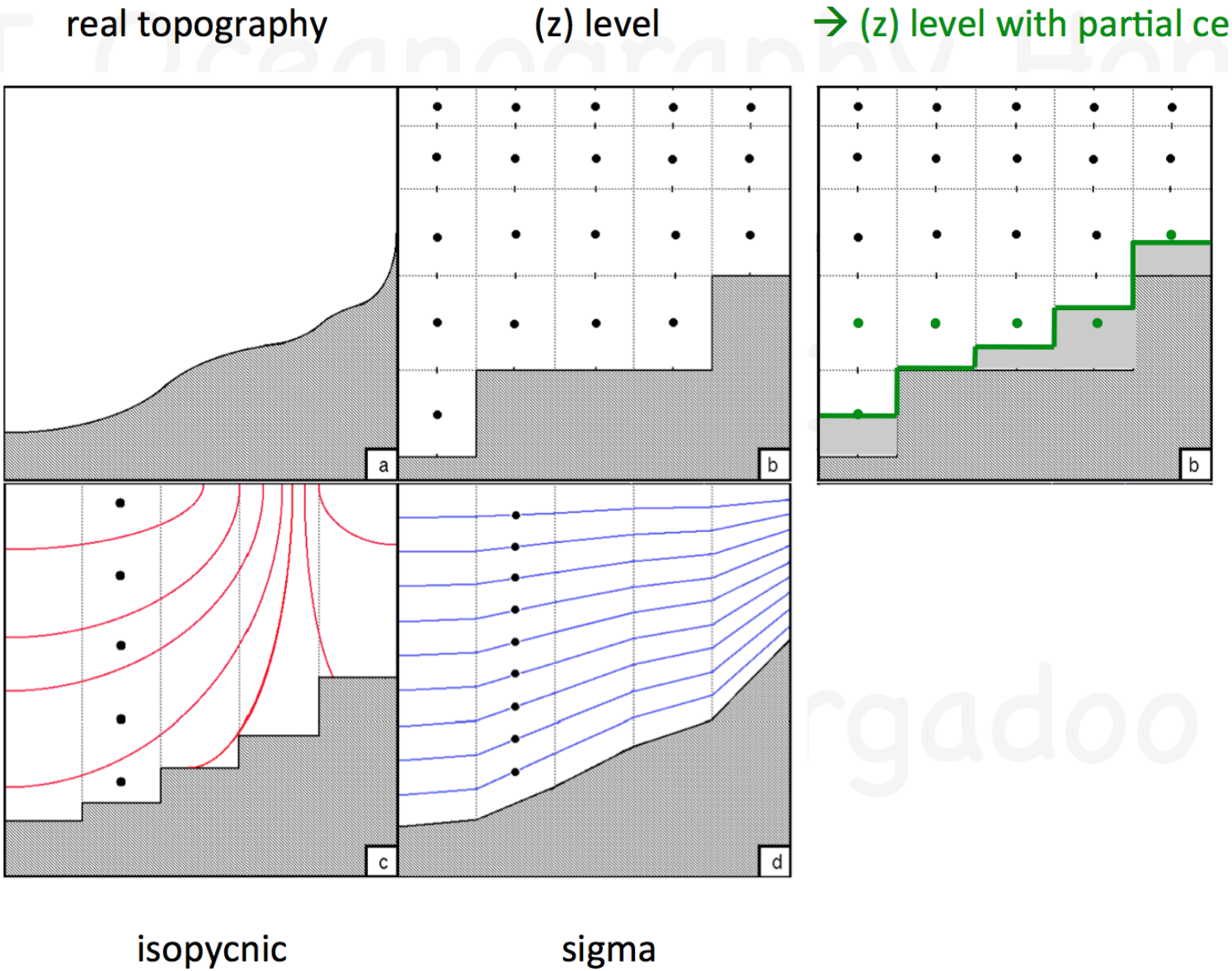
Vertical discretization

There are several choices for a vertical coordinate system used to represent the vertical structure of the ocean's water column.
All choices allows to increase detail near the surface where most of the action occurs. However, all have their strong and weak points!

The 4-most used coordinates are:

- Fixed-Level, Geopotential or Z-Coordinate model
- Isopycnal-coordinate model
- Sigma or Stretched-coordinate model
- Hybrid-coordinate model

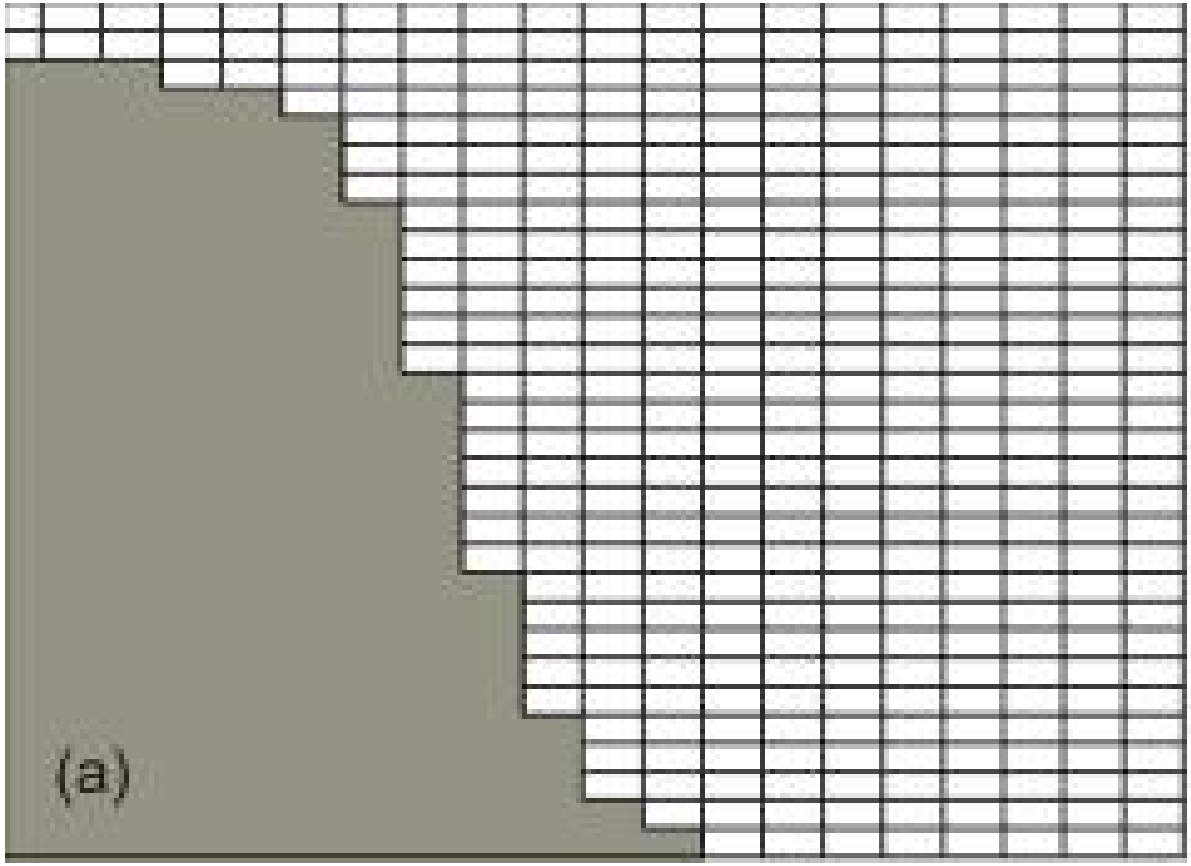
Vertical coordinates (J. Durgadoo lecture)



Z-Coordinate

- Consist on a series of regular surface depths.
  It is easy to set-up and is computationally efficient.
- The resolution is increased by decreasing the spacing between the levels.
- There is no easy-way to add grid cells without adding on the entire model domain.
- It has a poor representation of the continental slope.
  (Problems with flow over topography).

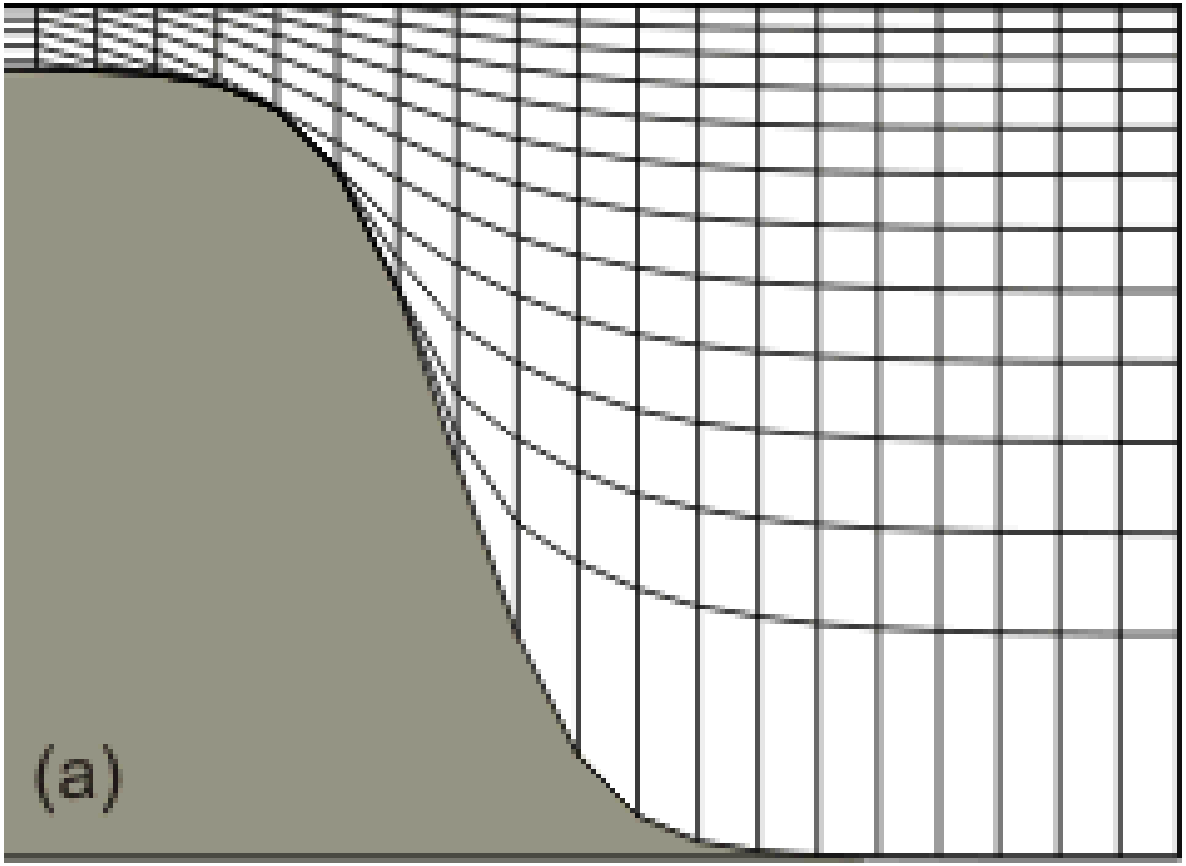
Z-Coordinate



Isopycnal-Coordinate

- Envisions the ocean as being made-up of a set of non-mixing layers whose interface locations adjust in time as part of the dynamics.
- The equations of motion are vertically integreted between isopycnals (surface of equal density). The layer-averaged velocities and layer ticknesses are dependent variables.
- The main advantage is that Mixing across the layers can be neglected (simplifying some computations and complexities).
- It has a very good representation of a stratified ocean (e.g: thermocline).
- The main disadvantage is that these models perform poorly in shallow waters where the ocean is less stratified. (It is not good for shelf dynamics).

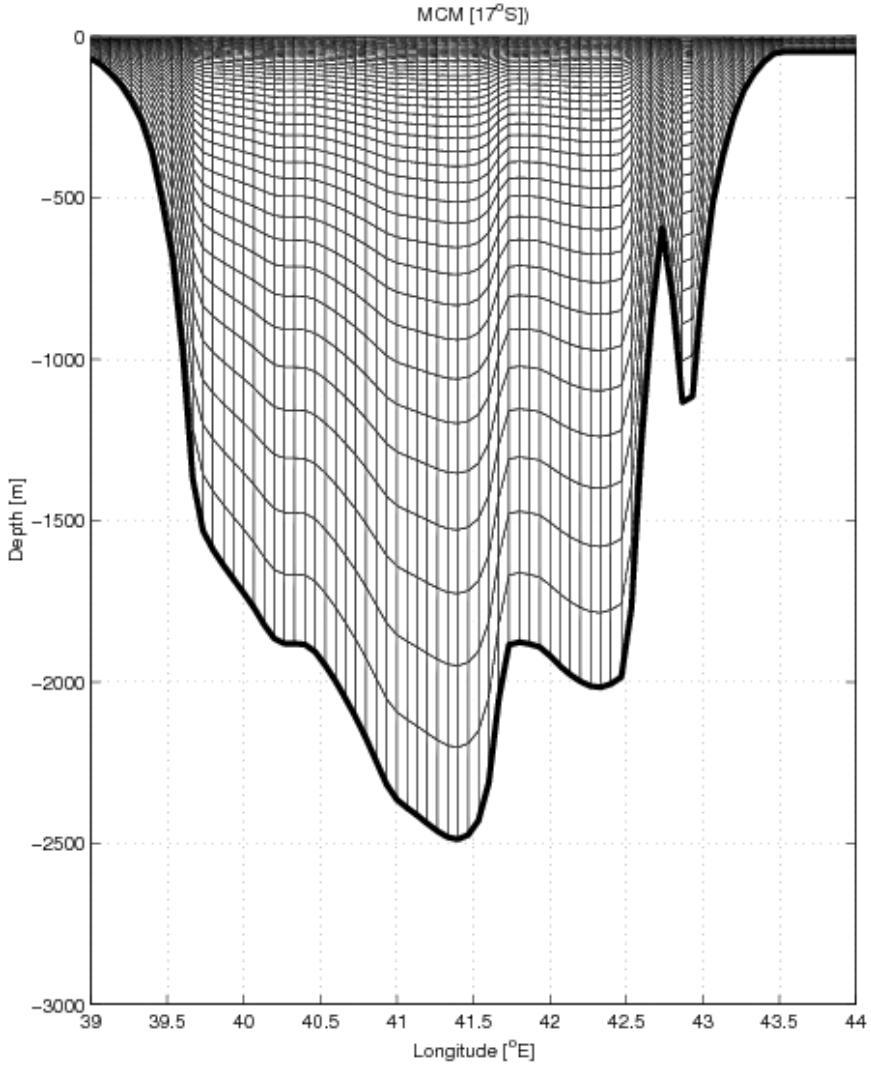
Isopycnal-Coordinate



Sigma-Coordinate

- It assumes coordinate surfaces which are fixed in time, but follow the underlying topography (are therefore not geopotential surfaces for non-flat bethymetry).
- Handles well the lateral boundaries and complex bathymetry. Are fairly horizontal near the ocean surface, and mimic the bathymetry near the seafloor (e.g: ROMS).
- It allows a better representation of the flow over the bathymetry.
- Can cause instabilities where the topography is very steep. Pressure gradient errors.

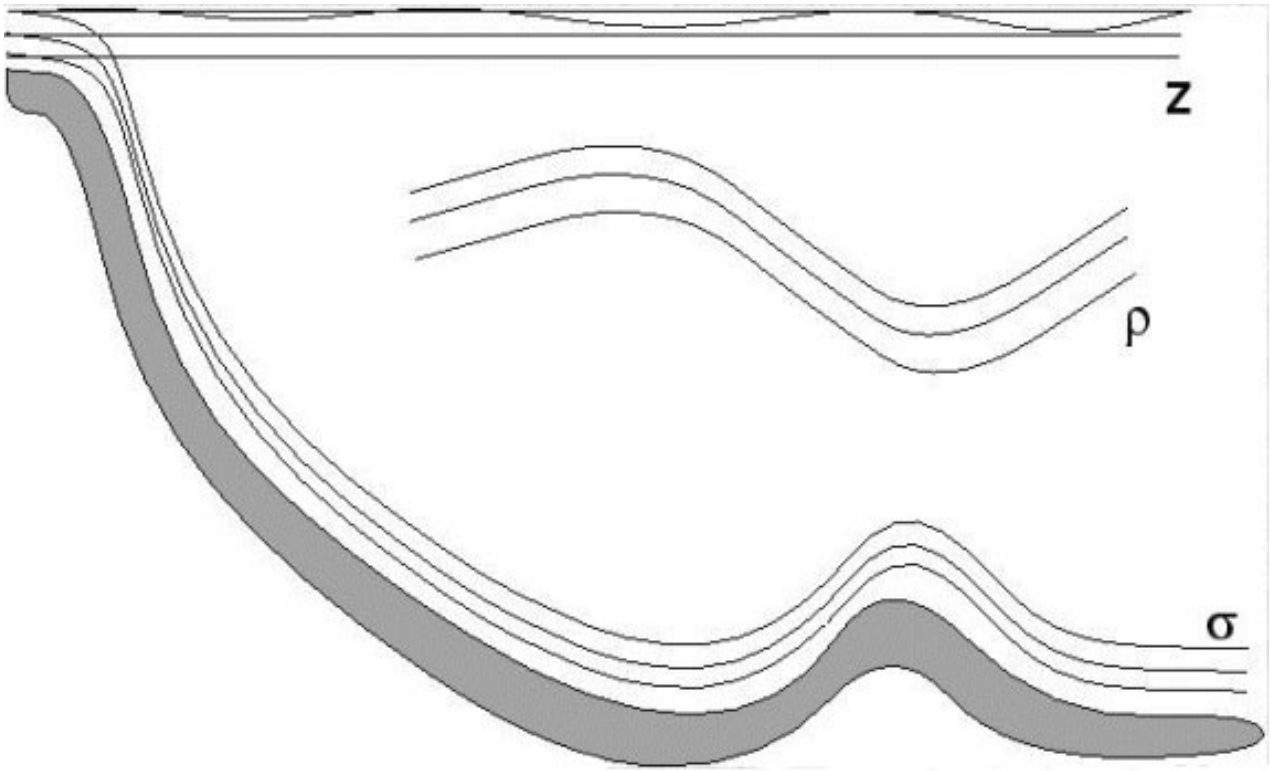
Sigma-Coordinate



Hybrid-Coordinate

- Seeks to optimize model performance by combining the best coordinate systems in different regions, based on the dominant process at work. (e.g: HYCOM uses Z, Iso, sigma- coordinates).
- It has a good representation of the vertical structure of the ocean throughout the water column.
- It allows the vertical coordinates to evolve in time and space as the depth of the mixed layer changes.
- It has a high computational cost.

Hybrid-Coordinate

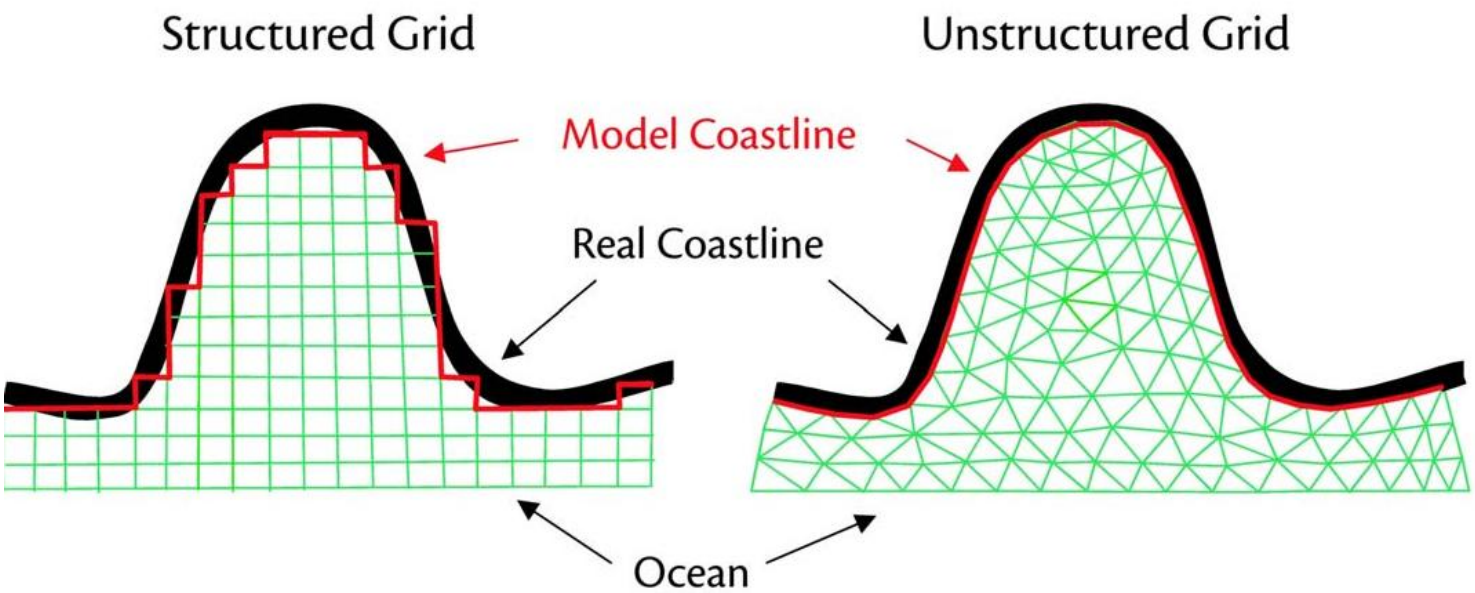


1 Courtesy – Prof. Eric Chassignet (U. Miami)

Horizontal grid discretization

To apply the mathematical formulation to a model ocean it is required to convert the equations to a series of algorithms that can be numerically applied to a digitized gridded ocean.

- Structured grids (finite volumes; regular or irregular/curvilinear)
- Unstructured grids (finite elements, best way to approximate a coastline)



Horizontal structured grids

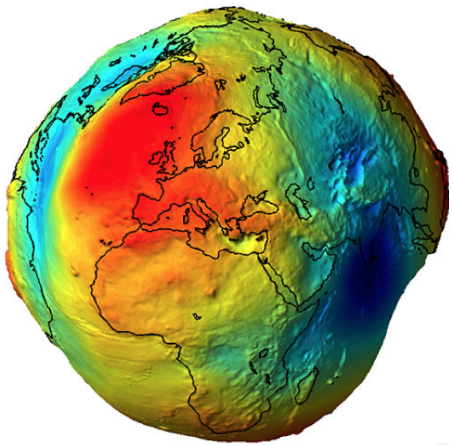
Structured Grids (Rectangular, Curvilinear and Spherical)

Consist of a series of spaced orthogonal lines.

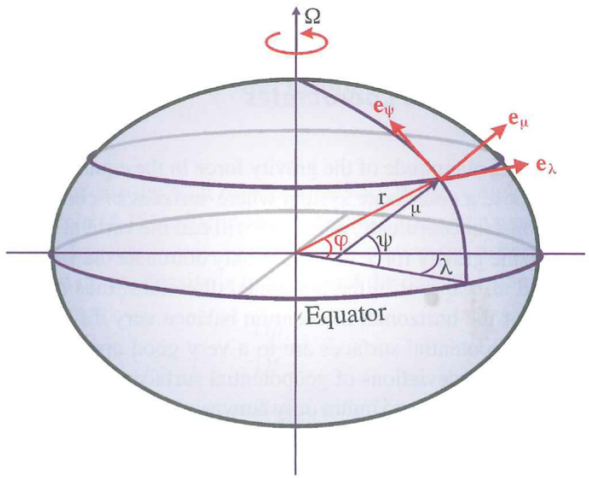
- Neighbour cells can be identified in a structured way
- Are computationally efficient.
- Uses straightforward algorithms derived from the discrete equations.
- Because the surface of the earth is a sphere, we cannot truly apply a uniform spacing across the grid and keep the lines straight (problems at the poles).

Grids on the Earth surface

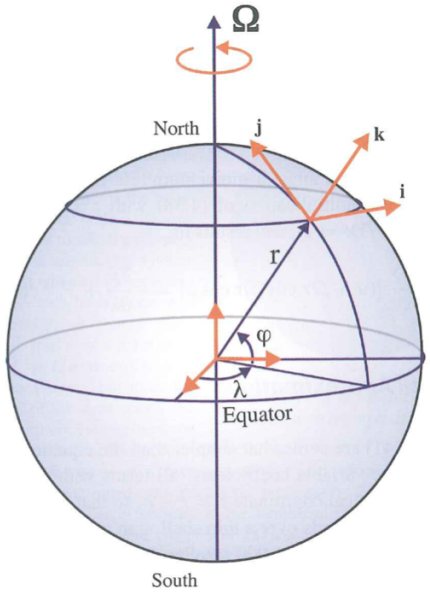
a.k.a. Shallow water approximation



Geoid

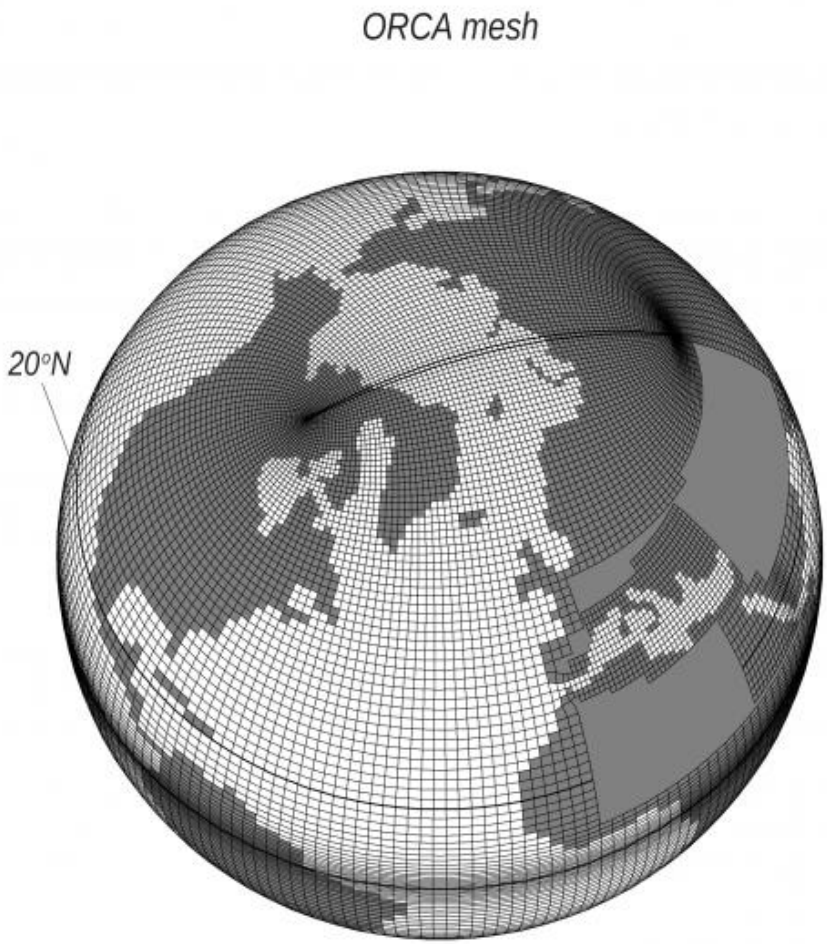


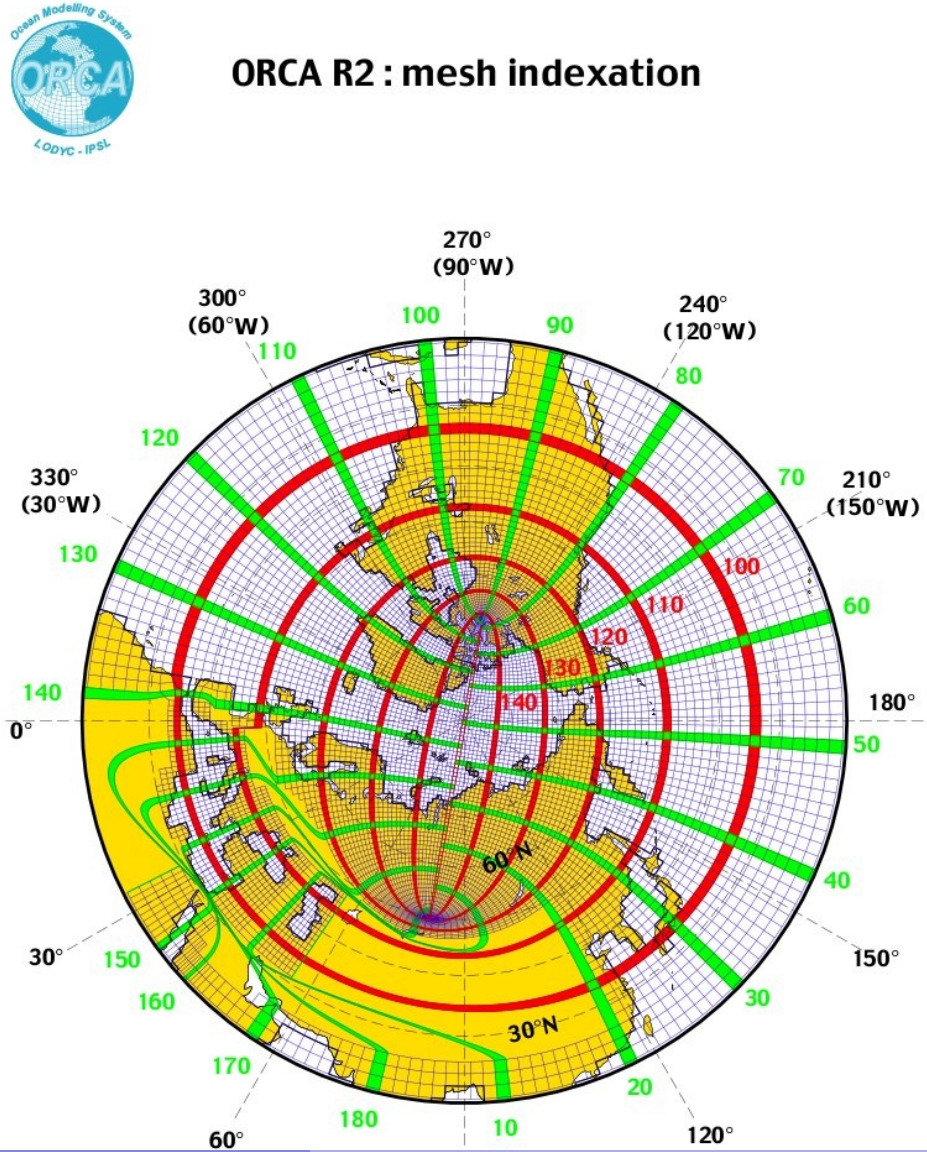
Ellipsoid



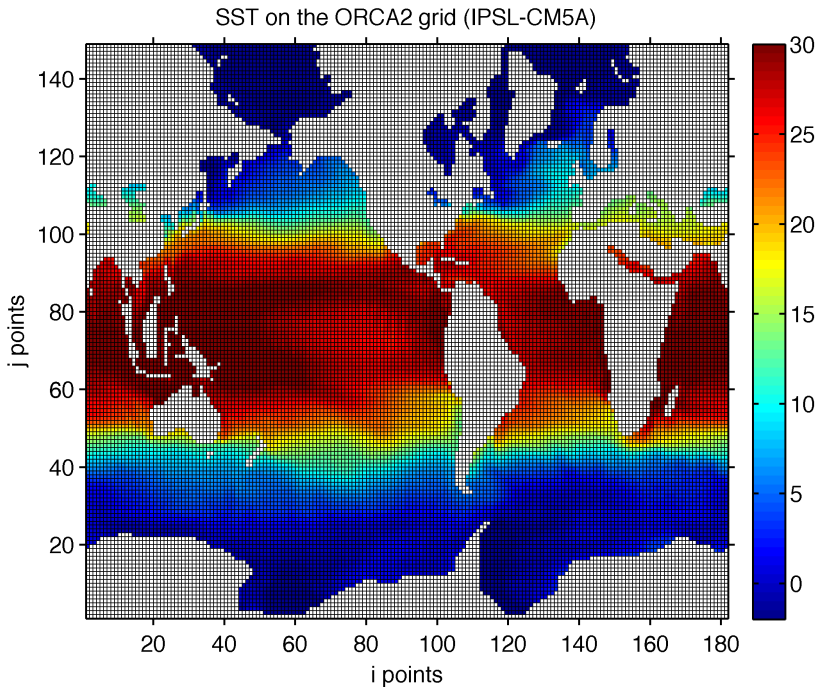
Spherical Earth

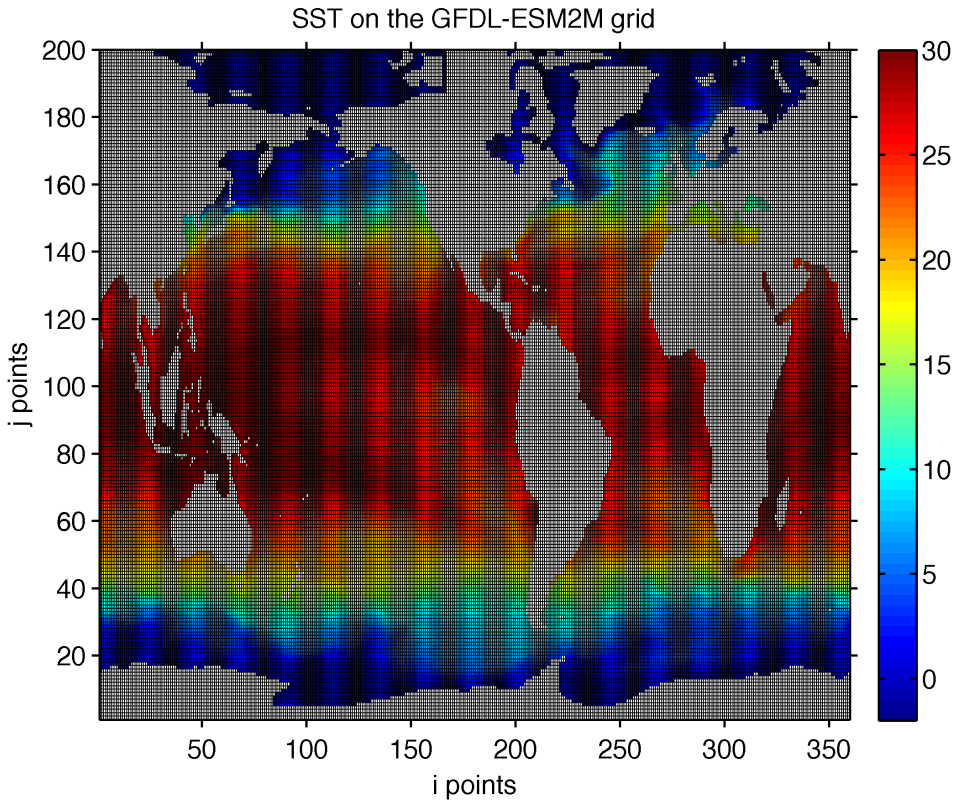
Spherical Earth grid and singularities





Data on the i,j (matrix) grid





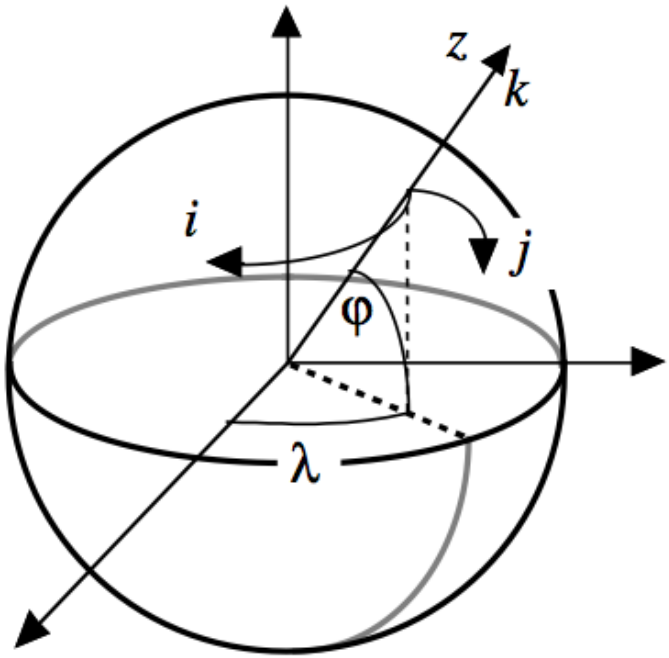
Curvilinear coordinates (Madec, NEMO book)

Let (i, j, k) be a set of orthogonal curvilinear coordinates on the sphere associated with the positively oriented orthogonal set of unit vectors $(\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}})$ linked to the earth such that $\hat{\mathbf{k}}$ is the local upward vector and $(\hat{\mathbf{i}}, \hat{\mathbf{j}})$ are two vectors orthogonal to $\hat{\mathbf{k}}$, i.e. along geopotential surfaces. Let (λ, φ, z) be the geographical coordinate system in which a position is defined by the latitude $\varphi(i, j)$, the longitude $\lambda(i, j)$ and the distance from the centre of the earth a + z(k) where a is the earth's radius and z the altitude above a reference sea level (in some cases we apply the thin shell approximation: $a + z \approx a$). The local deformation of the curvilinear coordinate system is given by $(e1, e2, e3)$, the three scale factors:

$$e_1 = (a + z) \left[\left(\frac{\partial \lambda}{\partial i} \cos \varphi \right)^2 + \left(\frac{\partial \varphi}{\partial i} \right)^2 \right]^{1/2}$$

$$e_2 = (a + z) \left[\left(\frac{\partial \lambda}{\partial j} \cos \varphi \right)^2 + \left(\frac{\partial \varphi}{\partial j} \right)^2 \right]^{1/2}$$

$$e_3 = \left(\frac{\partial z}{\partial k} \right)$$



Vector-invariant operators

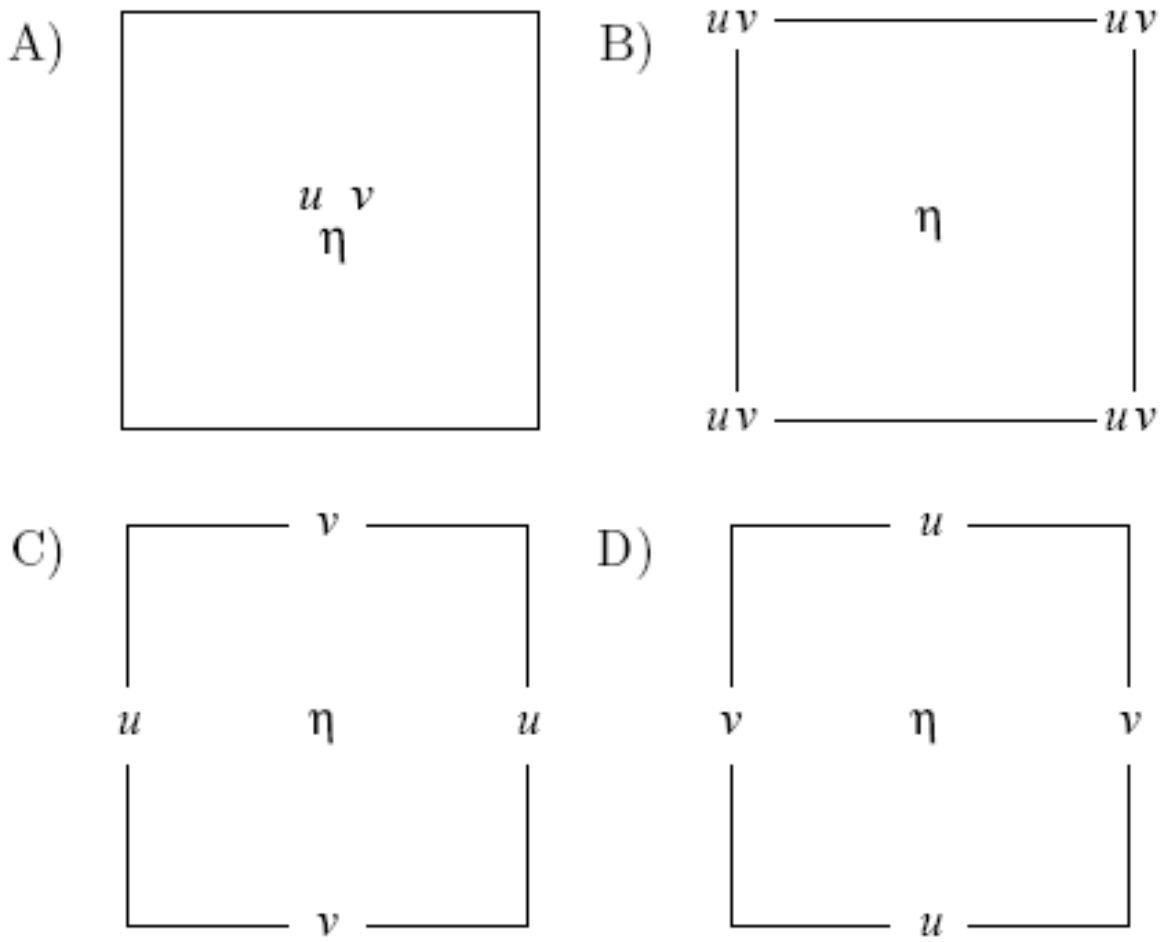
All the vector operators can thus be written in terms of the new coordinate system. Given a scalar field $q(i,j,k)$ and vector $\mathbf{A} \equiv (a_1, a_2, a_3)$, these are invariant for any scale deformation

$$\nabla q = \frac{1}{e_1}\frac{\partial q}{\partial i}\,\mathbf{i} + \frac{1}{e_2}\frac{\partial q}{\partial j}\,\mathbf{j} + \frac{1}{e_3}\frac{\partial q}{\partial k}\,\mathbf{k}$$

$$\nabla \cdot \mathbf{A} = \frac{1}{e_1\ e_2}\left[\frac{\partial\left(e_2\ a_1\right)}{\partial i} + \frac{\partial\left(e_1\ a_2\right)}{\partial j}\right] + \frac{1}{e_3}\left[\frac{\partial a_3}{\partial k}\right]$$

$$\nabla \times \mathbf{A} = \left[\frac{1}{e_2}\frac{\partial a_3}{\partial j} - \frac{1}{e_3}\frac{\partial a_2}{\partial k}\right]\,\mathbf{i} + \left[\frac{1}{e_3}\frac{\partial a_1}{\partial k} - \frac{1}{e_1}\frac{\partial a_3}{\partial i}\right]\,\mathbf{j} + \frac{1}{e_1 e_2}\left[\frac{\partial\left(e_2 a_2\right)}{\partial i} - \frac{\partial\left(e_1 a_1\right)}{\partial j}\right]\,\mathbf{k}$$

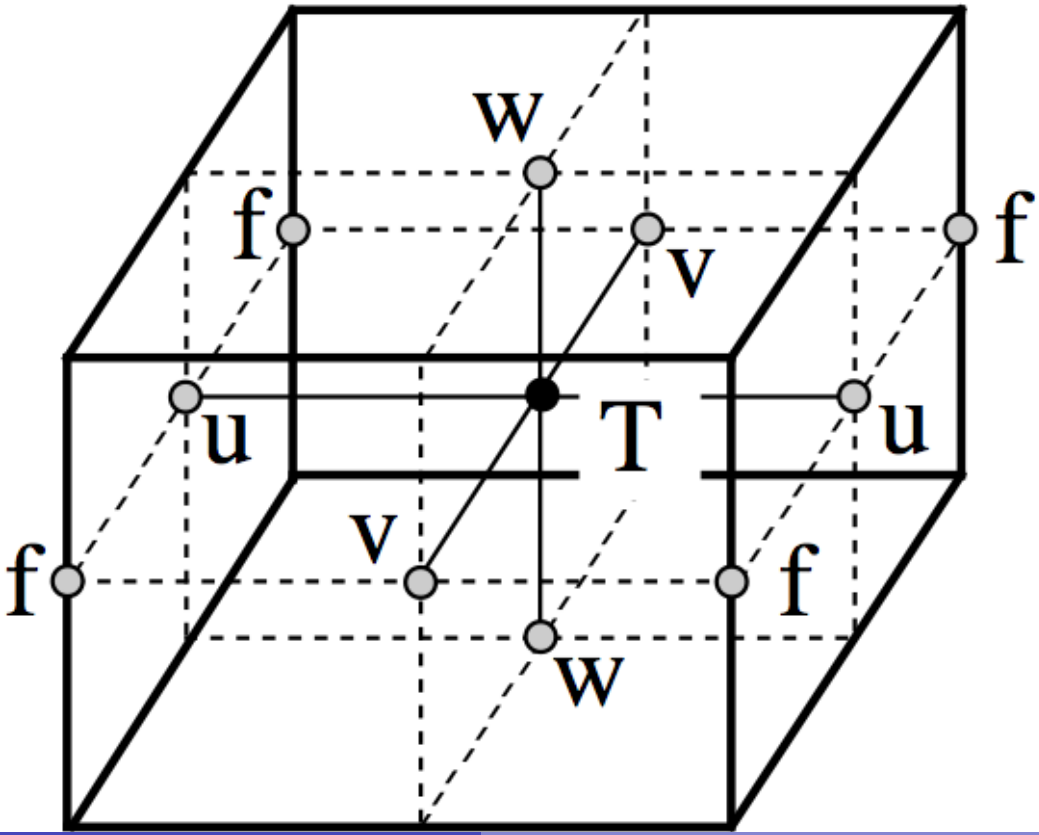
Structured grid types



Scalar and vector variables are "staggered" in different ways. Different names associated with the placement of the variables in the grid cells

Arrangement of variables on the NEMO C grid

T indicates the scalar variable points where temperature, salinity, density, pressure and horizontal divergence are defined. (u, v, w) indicates the vector variable points, and f indicates the vorticity points where both relative and planetary vorticities are defined

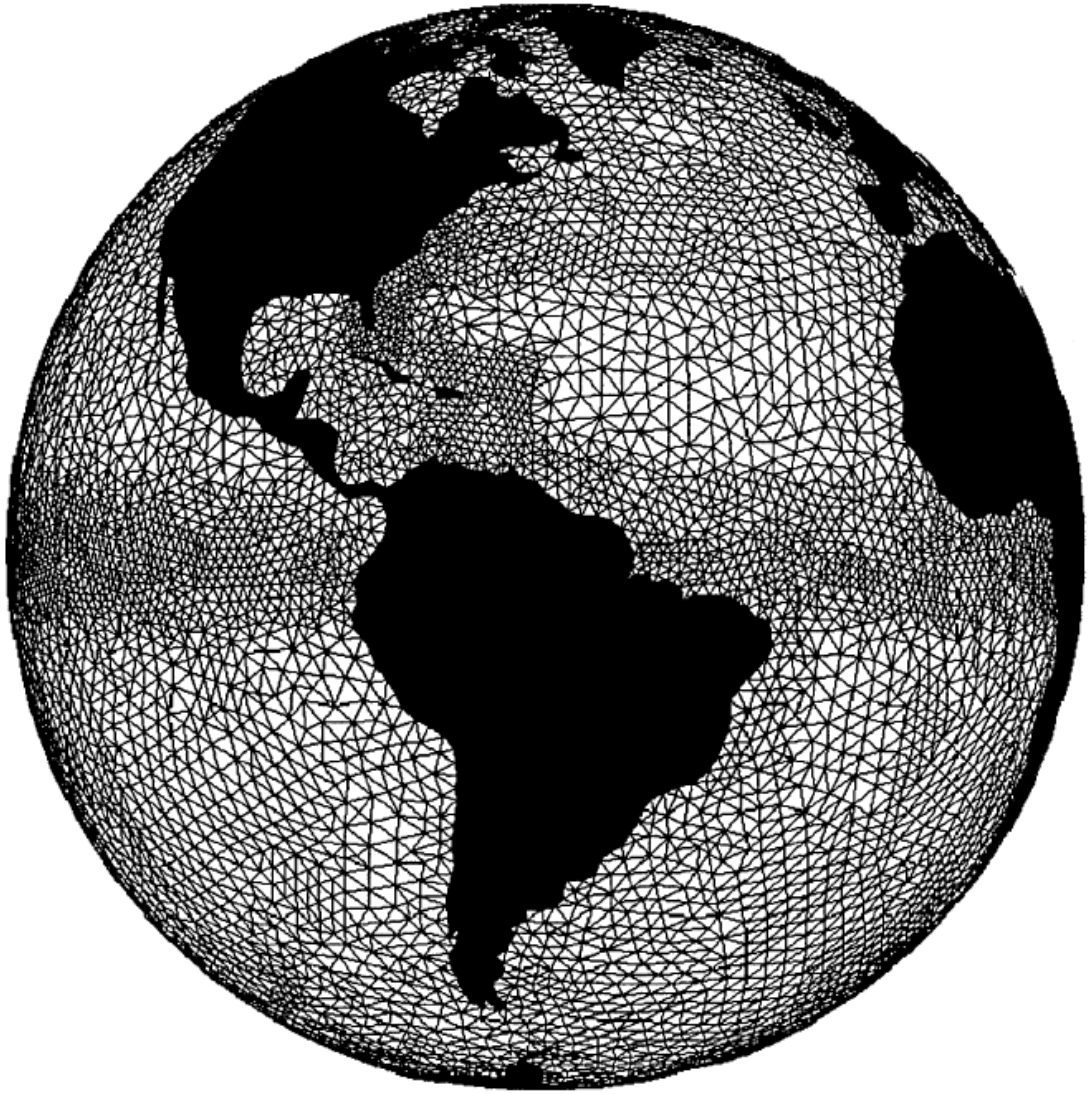


Unstructured grids (finite elements)

Also called triangular grids, they consist of a series of triangles

- Developed specially for the regional coastal ocean and estuaries.
- Allows to increase the resolution near the coast where small-scale processes are important, while keeping coarser resolutions where not required
- Can be more difficult to set-up and need specialized algorithms
- Computationally expensive

Unstructured grid models (global ocean example)



Domain decomposition

Calculations are not done sequentially in modern numerical modelling. Parallel computing is a fundamental aspect of solving discrete equations, which is implemented using different types of libraries (MPI and OpenMP). Domain decomposition is the process by which the model domain is subdivided in smaller portions, each one assigned to a computing core. The data exchange and gathering of the full domain is done with MPI (example of the ORCA1 tripolar grid with 8x16=128 domains).