



MAM4
Polytech'Nice-Sophia
January 2010

JAVA Interactive Applet 2D Oceanographic Simulation

Jonathan Bauer - Meryem Chibo - Yasmine Lahlou - Vivien Massart



Sophia Antipolis

Project directed by Mr Dreyfuss

Acknowledgements

In the first instance, we would like to deeply special thank to the Profesor Pierre DREYFUSS for having provided assistance, good advice and for being patient throughout these three weeks. We would like also to thank Doctor Antoine ROUSSEAU for his thesis which helped us having interesting information , and all those who were interested in our first term project.

Summary

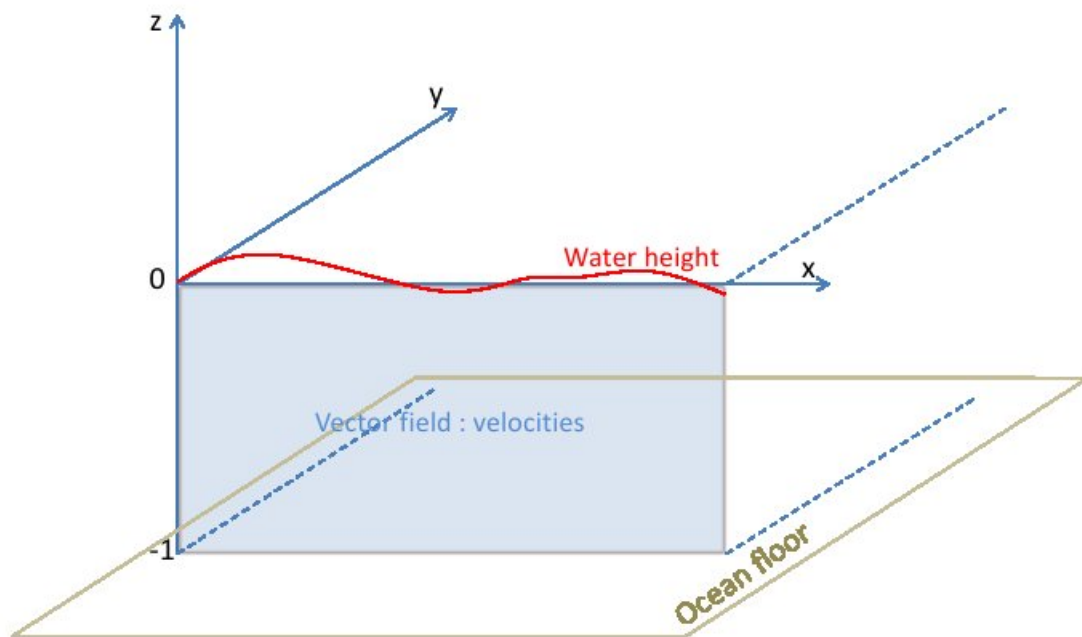
1	Introduction	4
2	Position of the problem	5
2.1	The physical model	5
2.1.1	The equations	5
2.1.2	Boundary and initial conditions	6
2.1.3	An exact solution	6
2.2	The mathematical model	7
2.2.1	The space discretization of the domain	7
2.2.2	Crank-Nicolson scheme for the velocity	8
2.2.3	The upwind scheme for the level of water	9
3	Programming	10
3.1	Tools of programming	10
3.1.1	Matlab	10
3.1.2	IDE Netbeans	10
3.1.3	The packages loaded	10
3.1.4	Wamp	11
3.1.5	Winscp	11
3.1.6	Launch4j and Inno Setup	11
3.1.7	The GIMP and Dia	12
3.2	Evolution of the project	12
3.2.1	The team and the website	12
3.2.2	Learning of programming tools	13
3.2.3	Theory and researches	13
3.2.4	SGT specifications	14
3.2.5	Development of the GUI with Netbeans	14
3.2.5.1	Java code basis	14
3.2.5.2	Creation of the GUI	15

3.2.6	Deployment of the application	19
3.2.6.1	Java Applet	19
3.2.6.2	Desktop Application	20
3.2.7	Correction of the problems	20
3.2.7.1	SGT display	21
3.2.7.2	Edit the vector attributes of the vector field	23
3.2.7.3	Initialization of parameters	24
4	How to use the application?	25
4.1	Use the application online	25
4.2	Install the application on the desktop	27
5	Future	29
6	Conclusion	29
7	References	30
7.1	Books and Articles	30
7.2	Internet Websites	30

1 Introduction

In general, the evolution of the ocean is modeled by the use of the viscous primitive equations, which have been studied for more than twenty years. Fortunately the important theoretical results are now available. This project is dedicated to those equations (PE), both from the theoretical and the numerical sides.

Our team's goal, on the one hand, is to implement a graphic interface modeling the evolution of some parameters (especially the velocity and height of water) which represents the Ocean, and, on the other hand, create a web application (Applet) to make it accessible for the public. We will start by studying the physical model (The equations, boundary and initial conditions and the exact solution), then, we will focus on the mathematical model by using a Crank-Nicholson Scheme for the velocity in time, an Upwind Scheme for the height of the water in time and finite volume in space for both of them. After that we will be able to implement it so we can compare our results and the theoretical solution.



2 Position of the problem

2.1 The physical model

2.1.1 The equations

Before focusing on the mathematical model and the coding part, we should first study the theory, in other words, the physical model. We consider the primitive equations of the ocean on the domain $(x, y, z, t) \in \mathbb{R} \times \mathbb{R} \times [-H(x, y), \zeta(x, y, t)]$ where $-H(x, y)$ denotes the topography of the ocean and $\zeta(x, y, t)$ denotes the altitude of the free surface of the ocean.

Here are the primitive equations :

$$\left\{ \begin{array}{l} \partial_t U_h + U_h \cdot \nabla U_h - \nu \Delta U_h + \frac{2}{\rho_0} \vec{\Omega} \wedge U_h + \frac{1}{\rho_0} \nabla_h p = 0 \\ \Delta_h U_h + \partial_z w = 0 \\ \partial_z p = -\rho g \\ \rho = \rho(z, T, S) \\ \partial_t T + U_0 \cdot \nabla T - \nu_T \Delta T = Q_T \\ \partial_t S + U_0 \cdot \nabla S - \nu_S \Delta S = Q_S \end{array} \right.$$

where the gravity g , the viscosity ν , the earth rotation vector $\vec{\Omega}$ and the diffusion coefficients ν_T and ν_S are **the parameters** and the 3d-velocity $(U_h, \omega) = (u, v, \omega)$ the pressure p , the density ρ , the temperature T and the salinity S are **the unknowns**.

Some hypothesis :

We have made some hypothesis to simplify the equations in order to make the java-programming part less difficult. On the one hand, we have considered that the diffusion coefficients influence is neglected and the density is constant $\rho = 1$. On the other hand, we have considered that the vertical velocity is a function of the horizontal one, and the pressure p is a function of the water height ζ . We have also considered that the problem is dimensionless, so we have introduced those dimensionless quantities :

$$\begin{aligned} (x, y) &= L(\tilde{x}, \tilde{y}), & t &= (L/U)\tilde{t}, \\ \zeta &= H\tilde{\zeta}, & z &= H\tilde{z}, \\ U_h &= U\tilde{U}_h, & U_0 &= U\tilde{U}_0 \end{aligned}$$

where L is the characteristic horizontal length and H is the vertical one, U is the velocity of the problem.

2.1.2 Boundary and initial conditions

The solution must verify the initial and the boundary conditions.

Initial conditions :

- the first condition : $U_h(., 0) = U_{h,i}$ in Ω
- the second condition : $\zeta(., 0) = \zeta_i$ in ω

Boundary conditions :

- the first condition : $\frac{\partial u}{\partial n} = \frac{\partial v}{\partial n} = 0$ in Γ_1
- the second condition (Newmann) : $\partial_z U_h(x, y, 0, t) = \partial_z U_h(x, y, -1, t) = 0$ in Γ_2

where $\Omega = R_x \times R_y \times (-1, 0)_z$, $\omega = R_x \times R_y$, Γ_1 is the floor ($z = -1$) and the surface ($z = 0$) of the ocean, and Γ_2 corresponds to the sides of the studied domain.

We have also introduced the following quantities :

- $\epsilon = \frac{U}{fL}$ the Rossby number.
- $Re = \frac{UL}{\nu}$ the horizontal Reynolds number.
- $Re' = \frac{H^2}{L^2 \times Re}$ the vertical Reynolds number.
- $Fr = \frac{U}{\sqrt{gH}}$ the Froude number.

We are interested in long-time oceanographic circulation for which the Rossby number is typically a small parameter. The values of Reynolds and Froude numbers vary with respect to the turbulent processes and to the height of the area that is considered respectively.

2.1.3 An exact solution

The solution below checks the boundary and the initial conditions :

$$\begin{cases} u(t, x, z) = t \cdot \sin\left(\frac{\pi x}{3}\right) \cdot \cos(\pi z) \\ v(t, x, z) = t \cdot \sin\left(\frac{2\pi x}{3}\right) \cdot \cos(\pi z) \\ h(t, x) = t \cdot \sin\left(\frac{2\pi x}{3}\right) \end{cases}$$

We don't obtain the expected 0-result if we inject these functions in the primitives equations, but a second member appears. We consider f_1 solution of the u -primitive equation, f_2 solution of the v -primitive equation and f_3 solution of the h -primitive one.

$$\begin{aligned} f_1(t, x, z) &= t \left(\left(\frac{10\pi^2}{9} \right) \cos(\pi z) \cdot \sin\left(\frac{\pi x}{3}\right) + \frac{\pi}{3} \cos(\pi z) \cdot \cos\left(\frac{\pi x}{3}\right) - \frac{\cos(\pi z)}{ee} \cdot \sin\left(\frac{2\pi x}{3}\right) \right) + \cos(\pi z) \cdot \sin\left(\frac{\pi x}{3}\right) \\ f_2(t, x, z) &= t \left(\left(\frac{13\pi^2}{9} \right) \cos(\pi z) \cdot \sin\left(\frac{2\pi x}{3}\right) + \frac{2\pi}{3} \cos(\pi z) \cdot \cos\left(\frac{2\pi x}{3}\right) + \frac{\cos(\pi z)}{ee} \cdot \sin\left(\frac{\pi x}{3}\right) \right) + \cos(\pi z) \cdot \sin\left(2\frac{2\pi x}{3}\right) \\ f_3(t, x) &= \frac{\pi t}{3} \cos\left(\frac{\pi x}{3}\right) \end{aligned}$$

We finally obtain a new system of equations for which u, v and h are solutions.

$$\left\{ \begin{array}{l} \left\{ \partial_t + U_0 \cdot \nabla_h - \frac{\Delta_h}{Re} - \frac{\partial_z^2}{Re'} + \frac{C}{\epsilon} \right\} U_h + \frac{\nabla_h \zeta}{Fr^2} = (f_1, f_2) \\ \partial_z U(x, y, -1, t) = \partial_z U(x, y, 0, t) = 0 \\ \partial_t + U_0 \cdot \nabla_h \zeta + \nabla_h \cdot \overline{U_h} = f_3 \end{array} \right.$$

where $C = \overline{U_h} = \begin{pmatrix} \overline{u} \\ \overline{v} \end{pmatrix} := \frac{1}{H} \int_{-H}^0 U_h dz$

Now we can focus on the mathematical aspect.

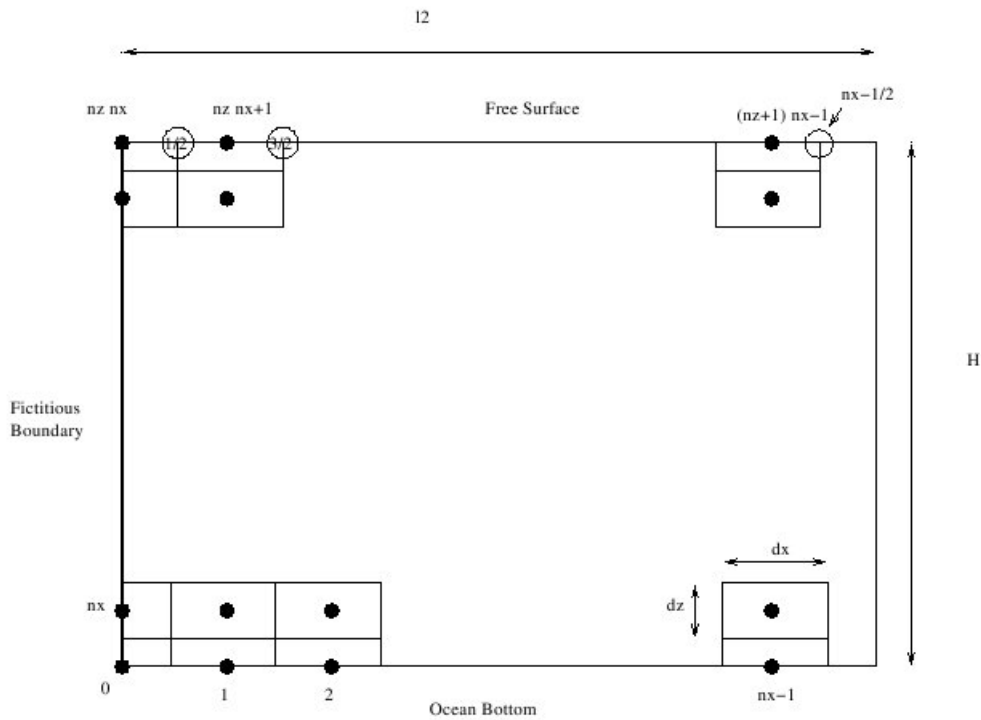
2.2 The mathematical model

Now that we had established the discretization of the domain, we can integrate the two equations of (u, v) on the time-space cell $[t_k, t_{k+1} \times C_1]$. In fact, we consider a system of equations related to the Primitive Equations of the ocean, and we supplement these equations with transparent boundary equations indicated in the previous part.

2.2.1 The space discretization of the domain

We first describe the space discretization of the domain. We consider a regular cartesian grid of $n_x \times n_z$ points and we apply the finite volume method.

We introduce the horizontal space step Δx and the vertical space step Δz . Here we are going to deal with the horizontal velocity and we are not going to compute a 3D pressure but a 2D water height. Note the all velocities can be computed on the same cells since we consider a $2d(x, z)$ problem for the horizontal velocity (u, v) in the (x, y) plane. We thus have to introduce two types of finite volume meshes. The first one is a 2D finite volume mesh and is related to the computation of the velocities. For $i = 0 \dots n_x - 1$ and $j = 0 \dots n_z$, we denote $I = i + jn_x$. The cells of this first mesh will be denoted $C_I = X_I + (-\Delta x, \Delta x) \times (-\Delta z, \Delta z)$, where the points X_I stand for $X_I = (0, -H) + (i\Delta x, j\Delta z)$ (they are represented by a black circle inside the figure). The second grid is a 1D finite volume mesh devoted to the computation of the water height. The cells of this second mesh will be denoted $c_i + 1 = x_i + 1 + (-\Delta x, \Delta x)$, where the points $x_i + 1$ stand for $x_i + 1 = (i + 1)\Delta x$ (they are represented by a circle with a number inside in the figure).



2.2.2 Crank-Nicolson scheme for the velocity

The famous scheme of Crank Nicolson is known to be second order in both, time and space, and unconditionally in quadratic norm, and conditionally stable in the L norm under a CFL condition on the time step. We apply this scheme to all the velocities U which has two components u (the abscissa of the velocity) and v (the ordinate of the velocity); then we obtain the discrete relations below :

For u :

$$\begin{aligned} u_{I,k+1} + \frac{\Delta t}{2} \left\{ u_0 D_x u_{I,k+1} - \frac{1}{Re} D_x^2 u_{I,k+1} - \frac{1}{Re'} D_z^2 u_{I,k+1} - \frac{1}{\epsilon} v_{I,k+1} + \frac{1}{Fr^2 \Delta x} \overline{D_{x1} \zeta_{j,k+1}} \right\} \\ = u_{I,k} - \frac{\Delta t}{2} \left\{ u_0 D_x u_{I,k} - \frac{1}{Re} D_x^2 u_{I,k} - \frac{1}{Re'} D_z^2 u_{I,k} - \frac{1}{\epsilon} v_{I,k} + \frac{1}{Fr^2 \Delta x} \overline{D_{x1} \zeta_{j,k}} \right\} \end{aligned}$$

For v :

$$\begin{aligned} v_{I,k+1} + \frac{\Delta t}{2} \left\{ u_0 D_x v_{I,k+1} - \frac{1}{Re} D_x^2 v_{I,k+1} - \frac{1}{Re'} D_z^2 v_{I,k+1} - \frac{1}{\epsilon} u_{I,k+1} \right\} \\ = v_{I,k} - \frac{\Delta t}{2} \left\{ u_0 D_x v_{I,k} - \frac{1}{Re} D_x^2 v_{I,k} - \frac{1}{Re'} D_z^2 v_{I,k} - \frac{1}{\epsilon} u_{I,k} \right\} \end{aligned}$$

where $D_x u_{I,k} = (u_{I+1,k} - u_{I-1,k})/\Delta x$ denotes a classical approximation of the first derivative in space in horizontal direction, $D_x^2 u_{I,k} = (u_{I+1,k} - 2u_{I,k} + u_{I-1,k})/(2\Delta x)$ and $D_z^2 u_{I,k} = (u_{I+n_x,k} - 2u_{I,k} + u_{I-n_x,k})/(2\Delta x)$ denotes a classical approximation of the second derivatives in space in horizontal and vertical directions, respectively.

2.2.3 The upwind scheme for the level of water

Now, let's move on the height of water's equations. This time, we integrate the equations on the time-space cell $[t_k, t_{k+1}] \times C_{i+\frac{1}{2}}$ except for $i = 0$, in this case we have to use the transmission conditions. This scheme helps us to compute the interface fluxes because it's a classic explicit scheme and known to be a first order in time and conditionally stable under a CFL condition. After discretization, the equation of the height of water is :

$$\epsilon_{i+\frac{1}{2},k+1} = \left\{ 1 - \frac{\Delta t}{\Delta x} u_0 \right\} \zeta_{i+\frac{1}{2},k} + \frac{\Delta t}{\Delta x} u_0 \zeta_{i-\frac{1}{2},k} - \frac{\Delta t \Delta z}{\Delta x} (\bar{u}_{i,k}^{+,n+1} - \bar{u}_{i-1,k}^{+,n+1})$$

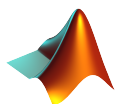
where $\bar{u}_{i,k}^{+,n+1} = \frac{u_{j,k}}{2} + \sum_{j=1}^{n_z-1} u_{j(n_x+1),k} + \frac{u_{n_z(n_x+1),k}}{2}$

3 Programming

3.1 Tools of programming

In this project we chose to program in the powerful oriented object language Java because Unlike most other languages, it provides a rich API to the developer allowing him to do many things and offers almost everything needed directly into the JDK. In addition Java is known with its excellent portability These are huge advantages, which increases greatly the development productivity. Our project implies to program a JAVA GUI by using a Matlab code. Therefore, we used some appropriate JAVA tools for the realization of our code. Effectives tools were needed to complete our project in the short time of 3 weeks, so we chose them with meticulousness.

3.1.1 Matlab



MATLAB is a language for scientific computing and high-level interactive environment for algorithm development, visualization and data analysis, or numerical calculation. We used **MATLAB 7.0** to run the complet version of Matlab code and compare it with our JAVA code.

3.1.2 IDE Netbeans



The **NetBeans** platform is a powerful tool for achieving Swing application, it offers time savings in productivity. So we decided to download the Java IDE NetBeans, the 6.5 version. The Netbeans GUI Builder greatly reduces the learning curve and development time needed to produce professional quality Java GUIs, and in particular to create Graphical User Interface with the GUI builder and its palette swing. Thanks to Netbeans, we can drag and drop components on a Swing interface very simply. It's very useful to develop a small application or an applet such as ours.

3.1.3 The packages loaded

To create matrices, we have used the **JAMA package**. It's a basic linear algebra package for Java. It provides user-level classes for constructing and manipulating real and dense matrices (but

no sparse ones). We have used the *class Matrix* of this package which provides various constructors to create matrices and fundamental operations of numerical linear algebra.

To create interactive graphics we have used the *JFreeChart package* and the *SGT package*. Firstly, JFreeChart makes it easy for developers to display professional quality charts in their applications. We have used it to represent graphically the level of water by drawing a XYChart. Secondly, SGT provides an easy development of independent platform Java applications to produce highly interactive, flexible, publication quality, object oriented graphics of scientific data. Features include user setttable or automatically scaled axes, sophisticated, automatically self-scaling time axes, labels as movable, customizable objects, automatic generation of legends to explain the data being displayed, and many more. We have used SGT to represent graphically the velocity of ocean by drawing a vector field. It's the only JAVA package which enable everyone to create a vector field.

3.1.4 Wamp



WAMP5 (meaning WAMP Windows Apache Mysql PHP) is a platform for Web development on Windows. Installation is easy and doesn't requires special knowledge : the Apache2 web server (installed as a service named wampapache), the mysql server (installed as a service named wampmysql) and the platform are operational at the end of the installation. The user must only select the installation folder where he put his web pages. That's why we decided to use it to develop our website and test it with the applet locally.

3.1.5 Winscp

WinSCP is an open source free SFTP client and FTP client for Windows. Legacy SCP protocol is also supported. Its main function is safe copying of files between a local and a remote computer. We needed to use the software winscp to transfer our html pages, php, our images and our application on the server *morag.polytech.unice.fr* and thus be able to access internet.

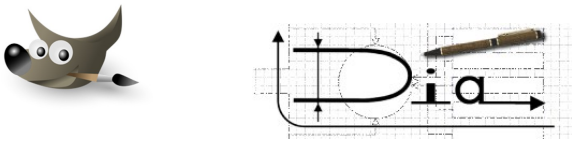
3.1.6 Launch4j and Inno Setup

launch4j



To deploy our program as a desktop application, it was necessary to use the software **Launch4j** and **InnoSetup**. Firstly, Launch4j enabled us to create launcher for our Java program. Secondly, Inno setup is an installer completely free. It creates executable to install the software developed. The installation of software packaged with Inno Setup is very simple, and uninstall is also supported. We used these two softwares to enable the installation of our application automatically on the desktop.

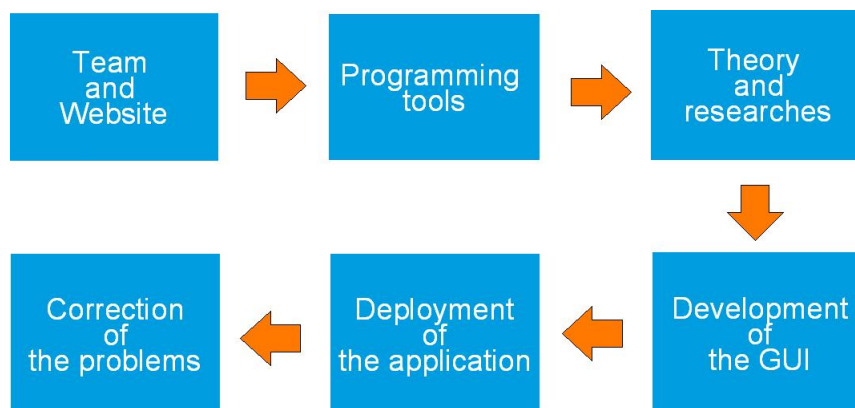
3.1.7 The GIMP and Dia



There is firstly *the Gimp* (Gnu Image Manipulation Program) which is a free software image processing bitmap (drawing, editing, animation, etc..). We used it to make the logos and some illustrative pictures : the OceanSim penguin, the OceanSim Polytech logo.

Secondly, *Dia* which is a free software for Charting. We used Dia to make the diagrams.

3.2 Evolution of the project

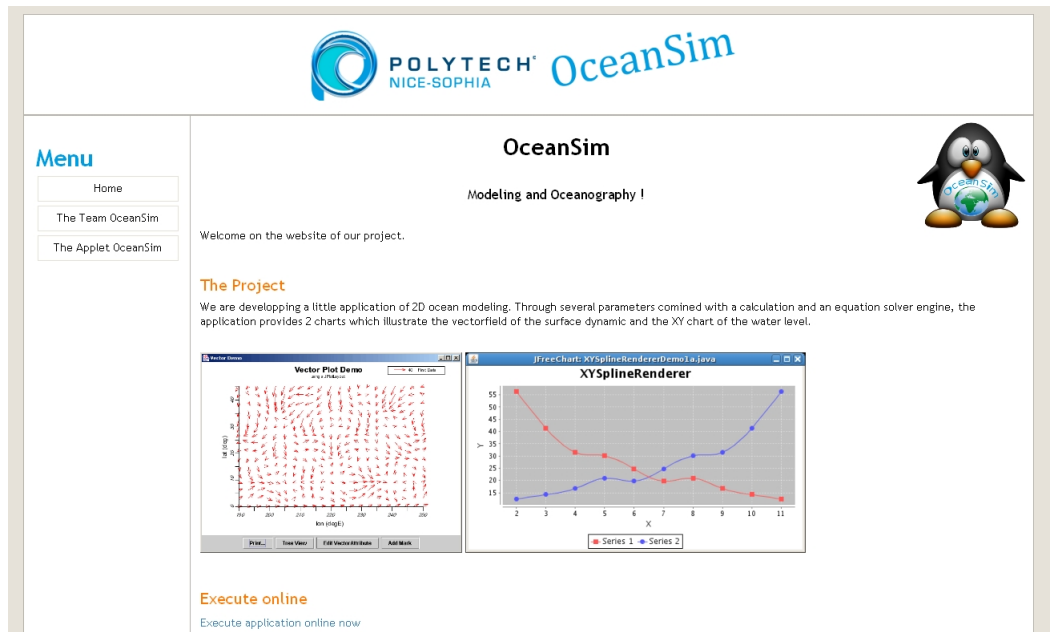


3.2.1 The team and the website

First of all, we decided to create a name for the team of research *OceanSim Polytech* and some logos which exemplify the identity of our team. We used *the Gimp* to create the logos, the first one created from the *Linux penguin* symbol of Open-Source and the other one from the *logo*

Polytech'Nice-Sophia symbol of our school. We think it is important within a team of project to create cohesion among members of the team.

Then we have created a website to deploy our applet and share our creations (source code, application...). So we have programmed in php, html and css to make the web site. Now everyone can visit our website and test our applet OceanSim : <http://users.polytech.unice.fr/jbauer/index.php>.



3.2.2 Learning of programming tools

We have downloaded all the tools which enabled us to program a GUI in JAVA optimally. We have also discovered the platform NetBeans and learn to use the different libraries downloaded (JAMA, SGT and JFreeChart).

3.2.3 Theory and researches

Two days were devoted to research and learning of the theory about the mathematical modeling and the primitive equations of the ocean. We used three documents to conduct our research and to better understand the model and the equations.

3.2.4 SGT specifications

The Scientific Graphics Toolkit (SGT) is designed to allow a graphics client developer a great deal of flexibility and freedom. SGT is a package that greatly aids a developer in creating graphics applets. sgt is not a general purpose graphics package, but provides the tools to enable scientific graphics to be easily incorporated into applications or Applets. SGT has three main components, the JPanel, on which all graphics are drawn. The Layer, of which several can be associated with a single JPanel, that insulates the developer from device coordinates. And the Graph, of which a single instance can be associated with a Layer, that transforms from user coordinates (e.g. cm/sec, time, etc) to the layer coordinate system (physical coordinates). Here is some useful SGT's packages :

- gov.noaa.pmel.sgt : Core classes for the Scientific Graphics Toolkit.
- gov.noaa.pmel.sgt.dm : Classes and interfaces that define the sgt datamodel.
- gov.noaa.pmel.sgt.swing : Components that use the package javax.swing.

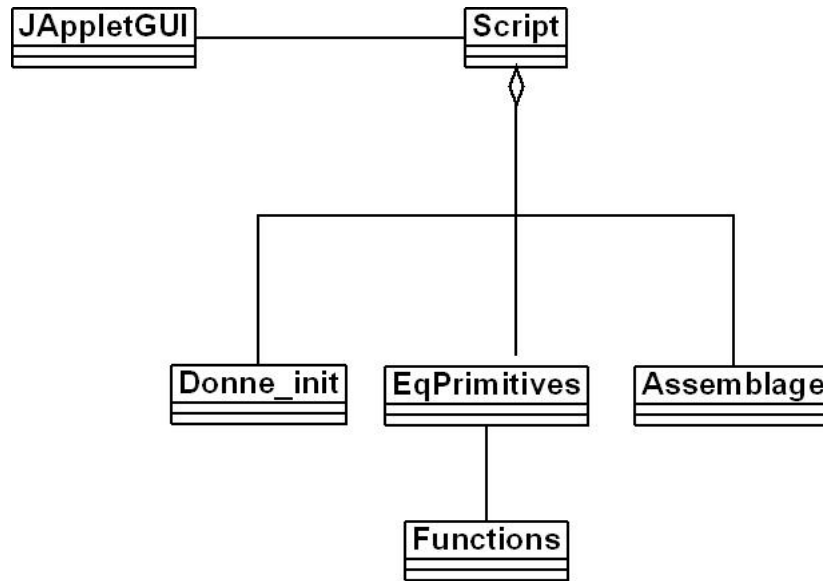
3.2.5 Development of the GUI with Netbeans

3.2.5.1 Java code basis

Mr Dreyfuss gave us a *complet version of Matlab code* of the simulation program and a *Java code basis* of the classes used to execute calculations for the simulation. The Java code basis which is the translation of the Matlab code, wasn't really efficient. For example, contrary to Matlab which uses *sparse matrices* and calculate quickly matrices our Java machine calculate slowly matrices with an important dimension. A solution proposed by Mr. Dreyfuss was to create sparse matrix. This kind of matrix allows saving only the values different from zero, and the corresponding index. In this case the Random Access Memory (RAM) is less used. Unfortunately, we have not enough time to implement and to develop this idea.

Our frist aim was *to create a GUI* by using the Java code basis, *to deploy it as an applet* and *to correct the Java code basis* to make correct and precise the calculations and display them on the GUI.

To understand our code, we represent the architecture of it, thanks to the classe diagram which only the name of the classes have been written on it.



There are 6 codes. The *JAppletGUI.java* is our interface and we present the GUI result on the next paragraph. The file *Script.java* can be considered like the main code. In this code we recover the two matrices characterizing height and velocity during time. *Script.java* calls three other objects : *Donne_init*, *EqPrimitives* and *Assemblage*.

⇒ Donne_init : initialization of several parameters and matrices.

⇒ EqPrimitives : calculations of the primitive equations to find U (matrix of velocity) and a (matrix of height).

⇒ Assemblage : calculations of different matrices to obtain the two matrixes A_p and A_m created in blocks.

⇒ Functions : generation of different functions.

3.2.5.2 Creation of the GUI

Matrices : our goal is to visualize in a Graphic User Interface the velocity and the height of the ocean (by using SGT and JFreeChart). *How to use the application ?* will be addressed in the next section. We are going to present the structure of the velocity and of the height obtained with the codes. In reality the velocity and the height are characterized by matrices which are calculated with the **Java code basis**

⇒ **The velocity** is represented by the matrix 'U' below, whom size is $2N \times (n_t + 1)$:

$$U = \begin{pmatrix} u_{0,0} & u_{0,1} & u_{0,2} & \dots & u_{0,n_t} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & u_{i,j} & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ u_{N-1,0} & u_{N-1,1} & u_{N-1,2} & \dots & u_{N-1,n_t} \\ v_{0,0} & v_{0,1} & v_{0,2} & \dots & v_{0,n_t} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & v_{i,j} & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ v_{N-1,0} & v_{N-1,1} & v_{N-1,2} & \dots & v_{N-1,n_t} \end{pmatrix}$$

Comment : the velocity \mathbf{U} is determined by a group of vectors for each time ; each column represents one time. For one time, there is one vector on one knot of the grid ; each vector is represented by its u-velocity, whom value is given by the term $u_{i,j,t}$, and its v-velocity, whom value is given by the term $v_{i,j,t}$, with the trinomial (i, j, t) given.

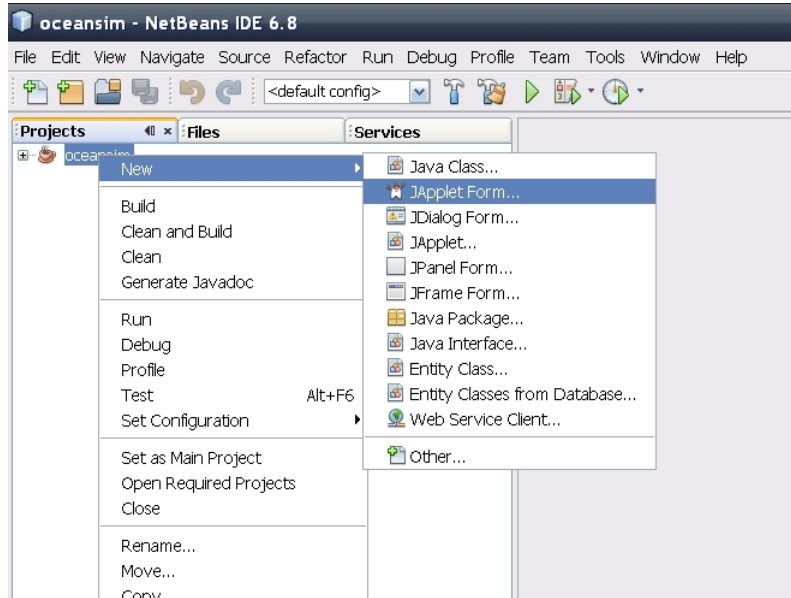
⇒ **The height** is represented by the matrix \mathbf{a} below, whom size is $n_x \times (n_t + 1)$:

$$a = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \dots & a_{0,n_t} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & a_{i,j} & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ a_{n_x-1,0} & a_{n_x-1,1} & a_{n_x-1,2} & \dots & a_{n_x-1,n_t} \end{pmatrix}$$

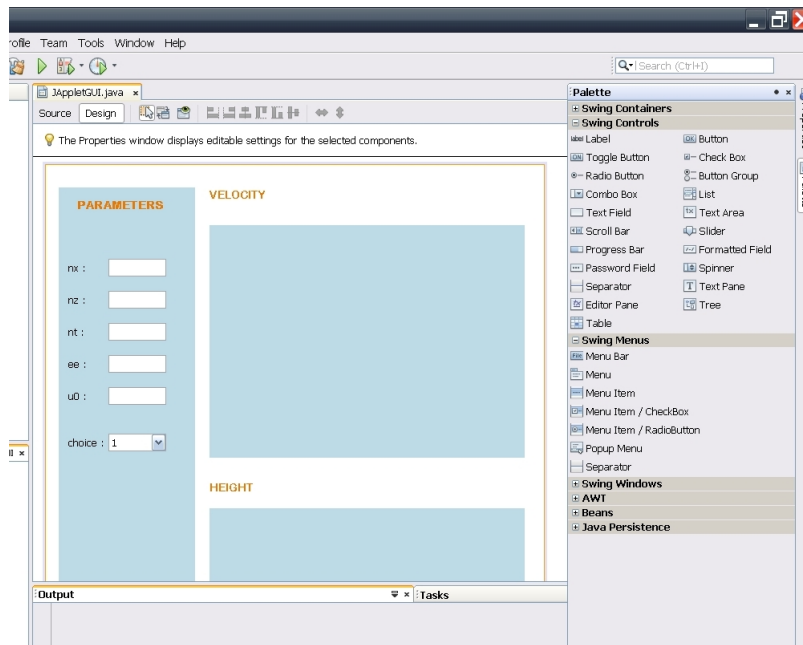
Comment : to create an animation we draw all the points of columns functions of time. In a first time we draw on a graphic the point of the first column. We obtain the height of the ocean, namely a curve at time 0 (this value zero corresponds to the index of the column). Each value of a column represents the height at time t, at the abscissa determine by the second index of values. Then this first curve is drawn, and we can draw the value of the second column, etc.

JAppletGUI class : It is the class built to create the GUI. By using Netbeans and its GUI editor (with its Swing palette), we created the class **JAppletGUI** which extends of

JApplet. Like we wanted to use the Swing palette, we selected **JAppletForm** in the textual menu :

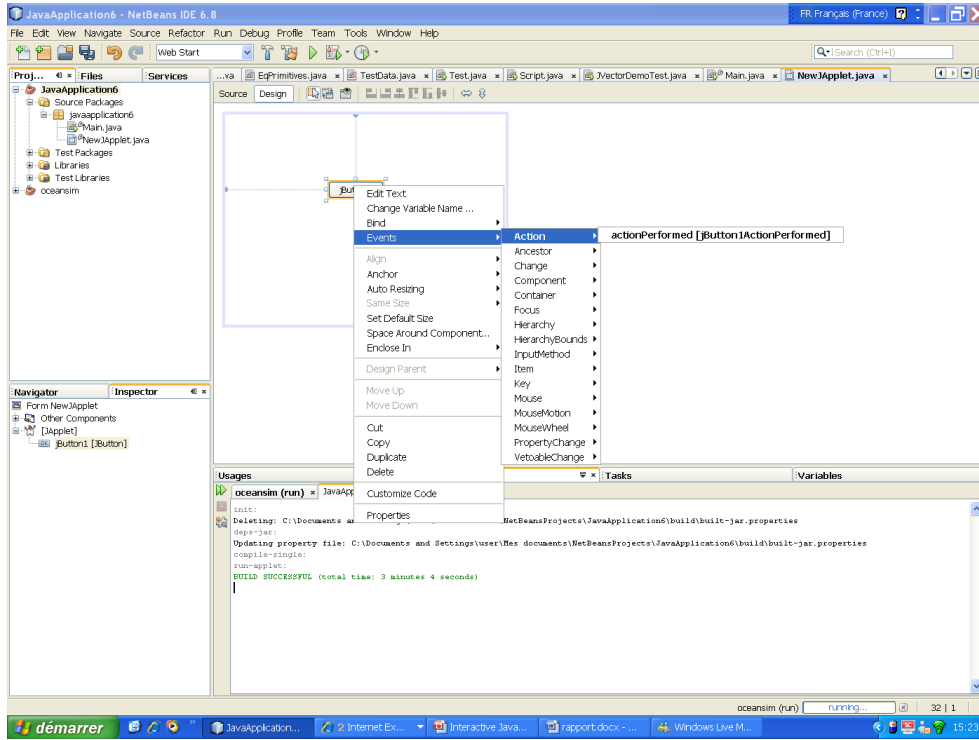


It enabled us to create easily, intuitively and quickly our Graphic User Interface :



It is not over, because Netbeans never ceases to surprise us. Building a button is good with

the palette, but the best thing is to add and attribute actions to the button.



Then we could implement entirely the even associated to the button.

```
Start Page x JAppletGUI.java x
Source Design
80
81 */
82
83 /** This method is called from within the init() method to
84  * initialize the form.
85  * WARNING: Do NOT modify this code. The content of this method is
86  * always regenerated by the Form Editor.
87  */
88 @SuppressWarnings("unchecked")
89 Generated Code
464
465 private void jMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
466 // TODO add your handling code here:
467 java.awt.EventQueue.invokeLater(new Runnable() {
468     public void run() {
469         final About dialog = new About(new javax.swing.JFrame(), true);
470         dialog.addWindowListener(new java.awt.event.WindowAdapter() {
471             @Override
472             public void windowClosing(java.awt.event.WindowEvent e) {
473                 dialog.closeAboutBox();
474             }
475         });
476         dialog.setVisible(true);
477     }
478 });
479 }
480
481 private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
482 // TODO add your handling code here:
483 System.exit(0);
484 }
```

Finally, NetBeans offered us so interesting and important applications which make it easily to use.

Parameters : In our GUI, a user must fill several parameters to initiate the animations. He must put only numbers. It is possible to write number with a point like '1.23' for instance. If this condition is not respected an information message appeared, and the field is automatically filled by default values. It is important to manage the data given by a user. Indeed, if a user writes a letter instead of a number, an error will be generated, and the applet will have certainly problems while the execution. We present the parameters in the part ***How to use the application ?***

Graphics : Thanks to the matrix U and the SGT library, we have implemented the display of the graphic which represents the vectorfield of the velocity.

We have made the same thing with the matrix a and the JFreechart library to display the graphic which represents the curve of the height of water.

3.2.6 Deployment of the application

It was an important part of our project. We chose 2 deployment modes, because we think It's an important thing to provide users different ways to use our program. The modes chosen are : **Java Applet**, and **Desktop Application**. We found that despite the apparent ease to deploy an application, it was actually difficult.

3.2.6.1 Java Applet

Why did we choose to build an applet ? Firstly, because it's an application downloaded directly from the web page that contains it and reflecting on this page a window that runs the application. The advantage is that you do not install any additional application on your machine. The applet is downloaded every time you open a new page containing it. The second advantage is that the applet will work on most browsers, regardless of the operating system of your machine. The disadvantage is that this gives the page which contains the applet a much larger size ; but thanks to the rapid spread of broadband, it's not a major obstacle !

Sign the applet Signing an applet is to sign the JAR file containing the applet and the JAR files of the librairies. Signing an applet enables to conduct transactions only allowed to a specific identity by the security policy in force or the deployment positions. Signing the JAR file containing the applet is done via jarsigner tool that supports combinations of algorithms supplied with Java 2.

It must first generate a certificate :

```
keytool -genkey -alias signature -keystore myStore
```

It must then answer the various questions (name, company, address ,....). This generates a certificate ***signature*** that is stored in a repository ***myStore***.

Then to sign the jar :

```
jarsigner -keystore myStore -signedjar sMyJar.jar myJar.jar signature
```

This generates the jar *sMyJar.jar* which is the version signed with the certificate *signature* of the JAR file *myJar.jar*.

Safety and BufferedImage As we used *BufferedImage* to create our graphics, we must specify in the applet that we disable secure access control in the constructor of the applet :

```
System.setSecurityManager(null);
```

Applet tag in html page

```
<applet  
code="oceansim.JAppletGUI"  
codebase="./secure/"  
archive="oceansim.jar"  
width="600"  
height="897">  
</applet>
```

It's the code that we have written in the html page and which enables the display of the applet.

3.2.6.2 Desktop Application

We also decided to deploy our application as a desktop application. So we have firstly created an executable from the JAR file with Launch4j. Then, we have created an installer that installs the program directly and simply. We made it to distribute our program and launch launch it as most existing programs by double-clicking the file.

3.2.7 Correction of the problems

At the end of these three weeks project, we have been confronted to some problems and we have decided to provide corrections and improvements on different points that we consider important :

3.2.7.1 SGT display

The main problem was concerning SGT, in fact we had difficulties with the display of the vectors fields, so we decided to read SGT's source files and edit the code reflecting observations made.

In our PDE model, we scan the rectangle of discretization on the bottom line horizontally from left to right, then at the end of the line we continue with the points of the line above.

$$\begin{bmatrix} nz \times (nx - 1) & \dots & \dots & \dots & \dots & N - 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ nx & nx + 1 & nx + 2 & 3 & \dots & 2nx - 1 \\ 0 & 1 & 2 & 3 & \dots & nx - 1 \end{bmatrix}$$

With this method we obtain the matrix of velocity U .

$$U = \begin{pmatrix} u_{0,0} & u_{0,1} & u_{0,2} & \dots & u_{0,n_t} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & u_{i,j} & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ u_{N-1,0} & u_{N-1,1} & u_{N-1,2} & \dots & u_{N-1,n_t} \\ v_{0,0} & v_{0,1} & v_{0,2} & \dots & v_{0,n_t} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & v_{i,j} & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ v_{N-1,0} & v_{N-1,1} & v_{N-1,2} & \dots & v_{N-1,n_t} \end{pmatrix}$$

As concerned SGT, we need to provide 1D vectors for the u-component (horizontal) and the v-component (vertical component) and we must take into account that SGT scan the space of discretization on the left column vertically upwards, then at the end of the column, it continues with the points of the right column.

$$\begin{bmatrix} nz - 1 & \dots & \dots & \dots & \dots & N - 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & \dots & \dots & \dots & \dots & \dots \\ 0 & nz & \dots & \dots & \dots & nz \times (nx - 1) \end{bmatrix}$$

For the u-component As concerned the u-component, if we consider $u_{comp}[0...N-1]$ (where N is the number of points) as the 1D vector of SGT we affect the values at time k in $u_{comp}[0...N-1]$ with this formula by corresponding the space discretization and the space ordonated by SGT :

```

for i = 0...nx - 1
    for j = 0...nz - 1
         $u_{comp}[j + i \times nz] = U[i - 1 + j \times nx, k]$ 
    end
end
end

```

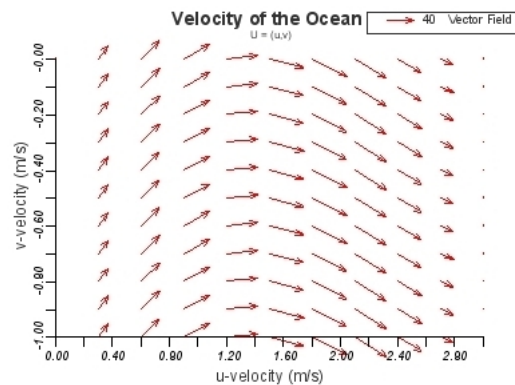
For the v-component As concerned the v-component, if we consider $v_{comp}[0...N-1]$ (where N is the number of points) as the 1D vector of SGT we affect the values at time k in $v_{comp}[0...N-1]$ with this formula by corresponding the space discretization and the space ordonated by SGT :

```

for i = 0...nx - 1
    for j = 0...nz - 1
         $v_{comp}[j + i \times nz] = U[N + i - 1 + j \times nx, k]$ 
    end
end
end

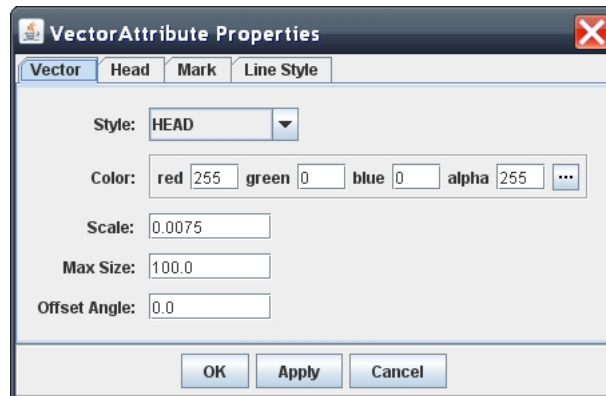
```

This correction has provided us a great result for the vectorField generated thanks to SGT. The graphic obtained is similar to the one obtained with Matlab.

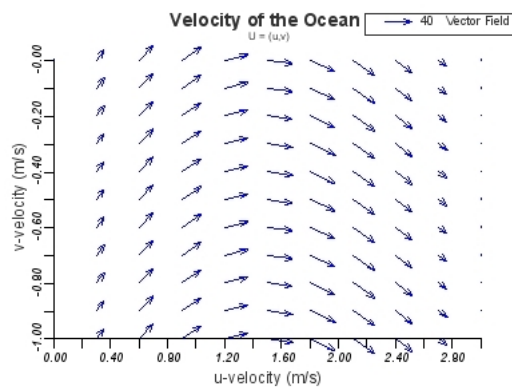


3.2.7.2 Edit the vector attributes of the vector field

Once a first version of the application finished, we thought we could improve the ergonomy and the fonctions available for the user. So, we have decided to add a button which enables the edition of the vector attributes in the vectorfield (graphic of velocity - SGT). A click on the button open a frame :

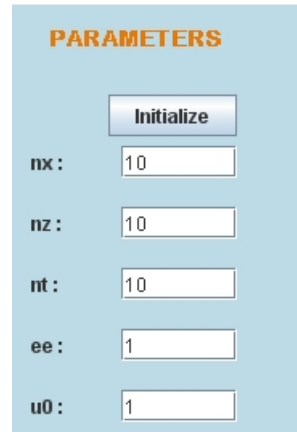


Thanks to it, it is possible to choose different attributes for the vector the scale, the color...



3.2.7.3 Initialization of parameters

The last problem was concerning the initialization of the parameters by the user of our GUI. So we have decided to edit the code in order to provide a button with a that initialize the parameters automatically without fill the fields :



The image shows a GUI window titled "PARAMETERS" with a light blue background. At the top center, there is a button labeled "Initialize". Below the button, there are five input fields, each preceded by a parameter label: "nx:", "nz:", "nt:", "ee:", and "u0:". The values in the input fields are 10, 10, 10, 1, and 1, respectively.

Parameter	Value
nx :	10
nz :	10
nt :	10
ee :	1
u0 :	1

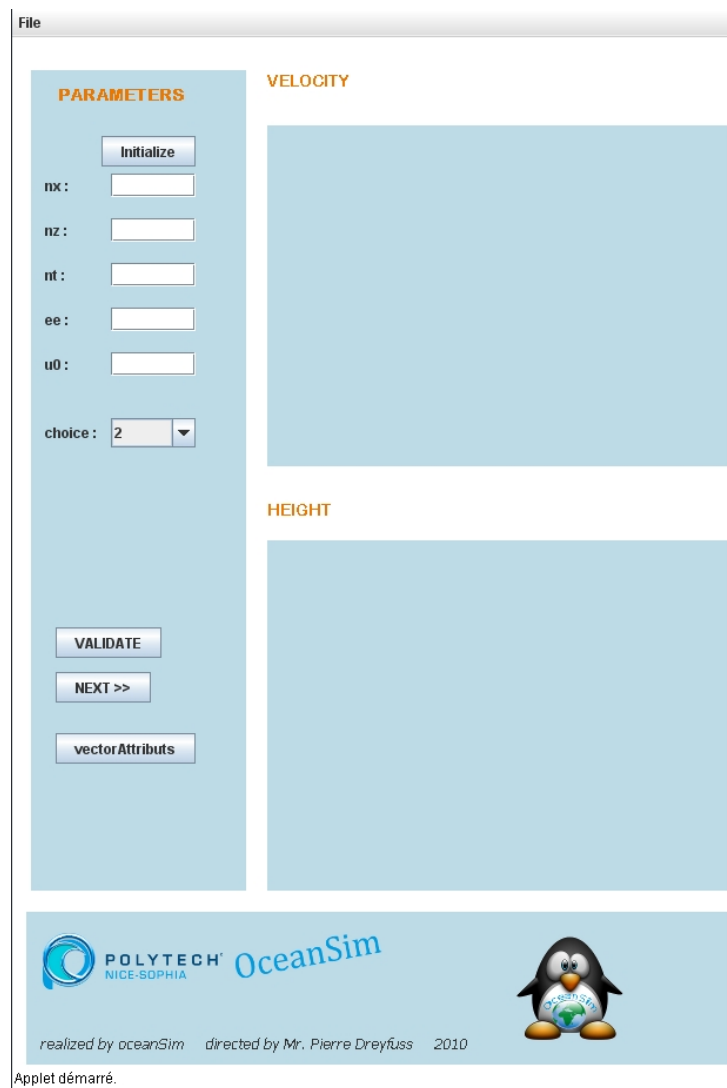
4 How to use the application ?

The first step to use our application is to have internet access and a functional browser (mozilla firefox for example). As we said previously there are two solutions to use our application : run it directly online in the browser via the applet or download the application, install and execute It on your desktop.

begin, we start to explain how to use the application online.

4.1 Use the application online

Firstly, you must go at the web address : <http://users.polytech.unice.fr/jbauer/index.php?p=applet> And you must see the applet like that :

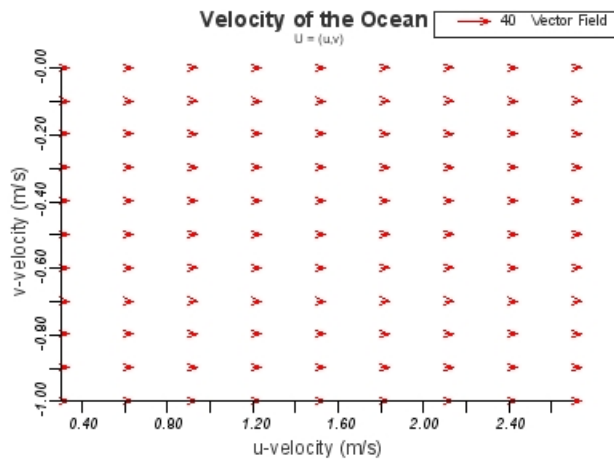


On the top left you should fill the parameters :

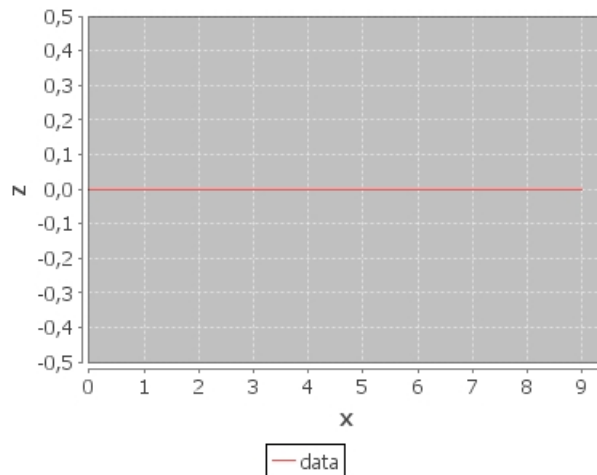
- n_x : this parameter represents the number of points along the X-axis,
- n_z : this parameter represents the number of points along the Z-axis,
- n_t : this parameter represents the number of step-time,
- ee : this parameter is the Rossby number,
- u_0 : this parameter is the initial velocity.

Then, you have to validate your parameters by clicking the validate button at the bottom left. To start the simulation it remains only to click the start button below the validate button. At this stage you should see two graphics that appear in the GUI, 2D Velocity of the ocean and 2D height of the ocean at the initial step ($t=0$) of the simulation :

VELOCITY at t:= 0



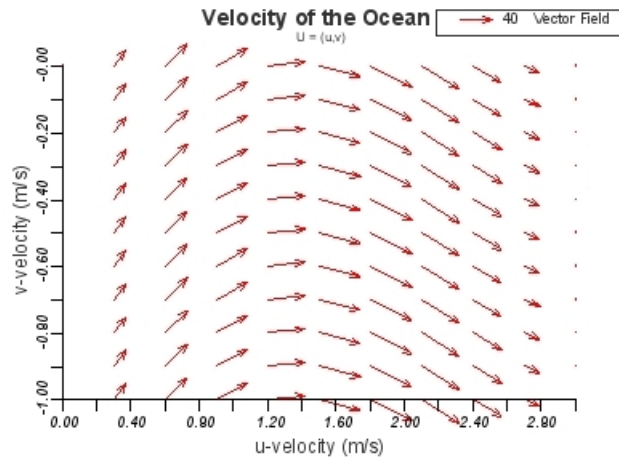
HEIGHT at t:= 0



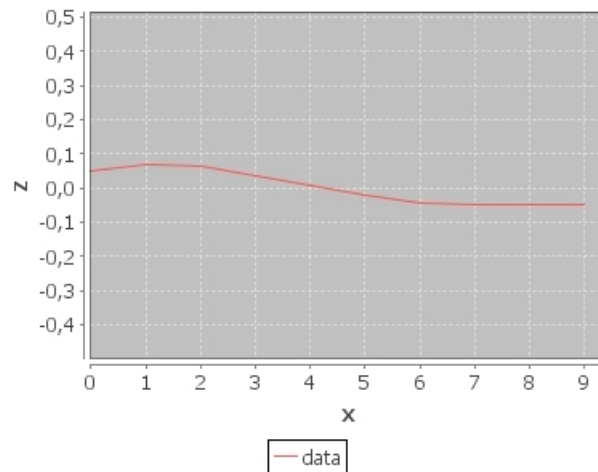
To continue the simulation and to see what happens some time later ($T=8$ for exemple)

you have to click on the button *next* :

VELOCITY at t:= 8



HEIGHT at t:= 8

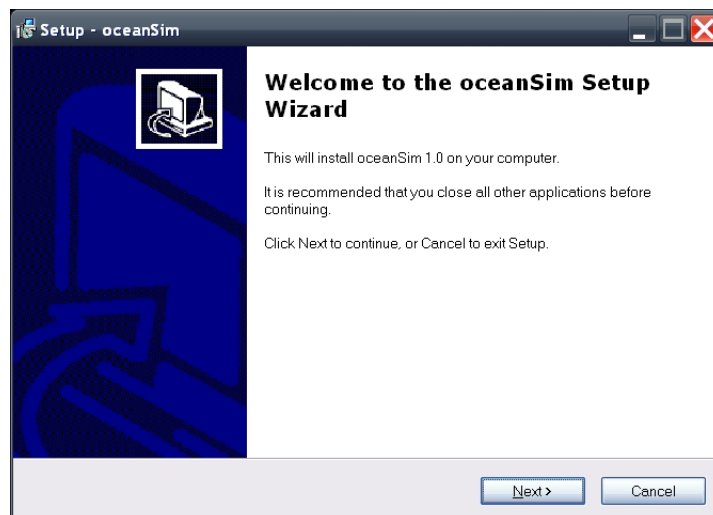
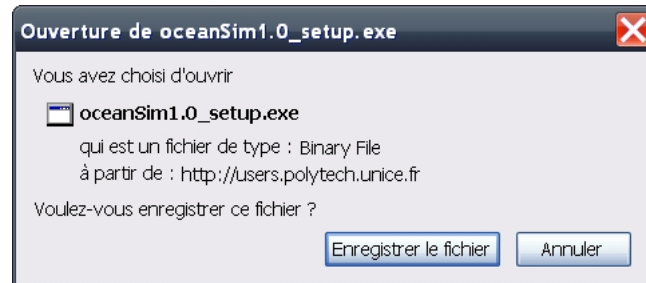


4.2 Install the application on the desktop

To execute the application directly on your computer, it's very easy. Firstly, you must download the application in ".exe" at this link http://users.polytech.unice.fr/jbauer/secure/files/oceanSim1.0_setup.exe

Then follow the instruction given by the installer of windows and launch OceanSim... It's magic! You can use the application like on the web but on your computer.

This can be useful if you want to make big computing greedy in memory.



5 Future

The monodomain model, that we have attempted to imitate, doesn't allow to have right previsions, because of a bad knowledge of boundary conditions. In a large domain, like the Ocean, the researchers hope for improving these results. To do that, the best manner is to divide the domain in several subdomain, and then find the solution in each subdomain. Thus, they call on efficient domain decomposition methods : the Schwarz waveform relaxation type algorithms.

The heart of the classical Schwarz method is to solve the problem on the whole domain thanks to an iterative procedure where a problem is solved on each subdomain by the use of boundary conditions that contain the information coming from the neighbouring subdomains. The correlation between all the solutions optimize the general and final solution.

6 Conclusion

Despite the lake of time, we have succeeded in creating a team and use welded properly and the default quality of each. We have created a real team of research with its identity and its spirit. It took a lot of patience and also reading (especially a thesis on numerical and theoretical study of the primitive equations of the Ocean) to understand the mathematical equations, to understand the algorithms and to create and deploy the JAVA GUI. Despite difficulties in the programming part of the applet such as the deployment on the website or the utilization of SGT, we have finally reached our goal by creating a functional and usable application by everyone.

7 References

7.1 Books and Articles

E. Audusse, P. Dreyfuss, B. Merlet. Optimized Schwarz waveform relaxation for Primitive Equations of the Ocean. January 16, 2009.

Antoine Rousseau. Etudes Theoriques et Numeriques des equations Primitives de l'Ocean sans Viscosite. June 15, 2005.

Roger Lewandowski. Etudes d'un système stationnaire linéarisé d'équations primitives avec des termes de pression additionnels. October 7, 1996.

Mhamed Ben Jmaa. Introduction à l'API Palette de Netbeans. June 16, 2008.

7.2 Internet Websites

- P. Dreyfuss : Personal page of Mr Dreyfuss (math.unice.fr/~dreyfuss)
- Matlab : Help online (www.mathworks.com)
- Java : Editor, API and Virtual Machine (java.sun.com)
- NetBeans : Using of NetBeans 6.5 (www.netbeans.org)
- NetBeans : Help to program in JAVA (java.developpez.com)
- JFreeChart : The official home page for JFree (www.jfree.org)
- SGT : Help for the Scientific Graphics Toolkit (www.epic.noaa.gov/java/sgt/)
- JAMA : Java Matrix Package (math.nist.gov/javanumerics/jama/)