Université Jospeh Fourier
Maître de stage : Achim WIRTH
Responsable universitaire : Magdelaine NORMANDIN
RAPPORT DE STAGE

Modélisation numérique des interactions entre vagues de shallow water et structures solides

Martin PUY

Grenoble, le 10 juin 2010

Table des matières

Résumé

La modélisation numérique est devenue l'outil incontournable dans tous les métiers de l'environnement et de la mécanique des fluides. En effet pour tous ce qui touche à l'aménagement hydraulique, l'érosion cotère, les énergies renouvelables. . .etc, le numérique est indispensable. Elle permet de résoudre des systèmes physiques qui étaient alors très difficiles à résoudre.

La simulation numérique repose sur un modèle physique décrit par des équations mathématiques. En mécaniques des fluides, les équations de Navier-Stokes sont la base de la modélisation mathématique de tout systêmes. Durant ce stage j'ai modélisé des écoulements à surface libre ayant une faible profondeur (écoulements en couches minces). Les équations de Shallow water (ou de Saint Venant) sont des équations aux dérivées partielles découlant des équations de Navier-stokes auquelles on applique certaines hypothèses. Ces équations définissent le champs de vitesse horizontal et les mouvements de la surface libre de l'écoulement. On utilise alors une méthode de résolution numérique du premier ordre en temps : Euler explicite, pour programmer ces équations qui sont la base de notre programme.

On utilise le logiciel SCILAB pour traiter nos données et les visualiser. Ainsi on peut créer des animations permettant une bonne visualisation des phénomens importants.

Mes tâches seront la conception, la modélisation et l'observation

Abstract

Numerical modeling has become an essential tool in all profession of the environment and fluid mechanics. In fact for all matters relating to hydraulic development, coastal erosion, renewable energy, digital is essential.

The numerical simulation is based on a physical model described by mathematical equations. In fluid mechanics, the Navier-Stokes equations are the basis of mathematical modeling of any system. During this placement I modeled free surface flows with a low depth (flow in thin layers). Shallow water equations (or Saint-Venant) are hyperbolic partial differential equations that describe the flow below a pressure surface in a fluid (sometimes, but not necessarily, a free surface) arising from the Navier-Stokes equations which we apply hypothesis.These equations define the horizontal velocity and movement of free surface flow. We uses a numerical method of the first order (Euler method) to program these equations

We use the software SCILAB to process and visualize our data. Thus we can create animations for a good visualization of importants phenomenom.

My tasks will be manufacturing, modeling and observation.

REMERCIEMENT

Tout d'abord je tiens tout particulièrement à remercier Christophe BAUDET pour m'avoir permis d'effectuer mon stage au sein du LEGI.

Je remercie surtout Achim WIRTH qui a énormément contribué à ma formation et mon bien être dans le laboratoire grâce à ses explications, sa disponibilité et sa bonne humeur.

Je remercie également Emmanuel et Carolina pour leur accueil et pour leur contribution à maintenir une bonne ambiance au sein du bureau.

Pour finir je tiens à remercier Bénoit Coronel qui, grâce à notre association nous a permis de réaliser les taches qui nous incombaient avec succès.

INTRODUCTION

Voulant orienter mes études vers le domaine de la mécanique des fluides et étant intéressé par le numérique, j'ai donc choisi d'effectuer mon stage de fin d'étude au laboratoire des écoulements géophysiques et industriels.

Il m'a été proposé de modéliser des écoulements à surface libre et à faible profondeur grâce aux équations de Shallow Water (ou de Saint Venant) dérivant des équations de Navier-stokes. On utilisera la méthode numérique d'Euler explicite, un méthode du premier ordre en temps. On construira notre programme pas à pas en testant les differentes parties des équations de shallow water. On pourra ainsi contôler les termes de diffusion grâce à leur similitude avec léquation de la chaleur, contrôler les termes d'advection et mettre en evidence l'importance des conditions de stabilités. Ainsi on pourra le calibrer et vérifier qu'il n'y a pas d'erreurs.

De plus on modélisera des îles pour observer le comportement de la houle sur celles-ci et appliquer un forçage sur l'océan pour ce rapprocher au maximum des conditions réelles. Ainsi on pourra observer et définir le comportement des vagues de shallow water sur une structure solide.

On pourra aussi déterminer les limites de la méthode d'Euler éplicite sur ce types de problèmes.

Chapitre 1

PRESENTATION DU L.E.G.I

1.1 Généralités

Le Laboratoire des Ecoulements Géophysiques et Industriels (LEGI) est un laboratoire de recherche publique de l'Université de Grenoble . C'est une Unité Mixte de Recherche commune au Centre National de la Recherche Scientifique (CNRS), à l'Université Joseph Fourier (UJF) et à l'Institut National Polytechnique de Grenoble (Grenoble-INP), qui rassemble plus de 180 personnes dont 70 permanents et autant de doctorants et post-doctorants.

Les activités de recherche en Mécanique des Fluides et Transferts menées au LEGI s'appuient sur une combinaison d'approches méthodologiques alliant modélisation, expérimentation (plus de 40 bancs expérimentaux dont de grands instruments), simulation numérique à hautes performances (machines de calcul paralléle, calculateurs nationaux...) et développement d'instruments de mesure innovants. Ces activités sont liées à de trés nombreux domaines d'application relevant de problématiques environnementales aussi bien qu'industrielles.

1.2 Domaine de recherche

Les axes scientifiques principaux du LEGI sont au nombre de trois :

Dynamique des écoulements turbulents.
Intégrant compréhension et simulations avancées de l'hydrodynamique,
du mélange et des transferts

Dynamique des écoulements à très forts couplages hydrodynamiques.
Portant sur la physique et la modélisation d'écoulements à fort contraste de
densité, d'écoulements multiphasiques discrets (bulles, gouttes...) et continus (phases dispersées, séparées...)

Dynamique des fluides géophysiques.
Déclinée en analyse de processus et en simulations de systèmes naturels
(océans, atmosphère, côtier, fluvial ...), et participant à la compréhension
des évolutions climatiques et à l'élaboration d'outils de prévision

Ce large domaine de recherche permet de nombreuses applications dans de très variés domaines. En effet les recherches du LEGI peuvent êtres appliquées dans les milieux naturels (Océanographie, Côtier et fluvial, Atmosphère) ou l'ingénierie (aéronautique, santé, transport)...

1.3 Contrat, collaboration, formation

Le LEGI développe une large part de ses activités au travers de contrats de recherche avec des partenaires privés aussi bien que publics (programmes régionaux, nationaux, européens et internationaux), de collaborations universitaires, de thèses et de post-doctorats, d'expertises.

Le LEGI est très actif dans la formation par la recherche : ils accueillent un grand nombre de doctorants relevant principalement des Ecoles Doctorales Mécanique et Energétique et Terre Univers Environnement . Plusieurs dizaines d'étudiants de tous horizons (masters, écoles d'ingénieurs, licence, IUT...) effectuent chaque année au LEGI leur stage.

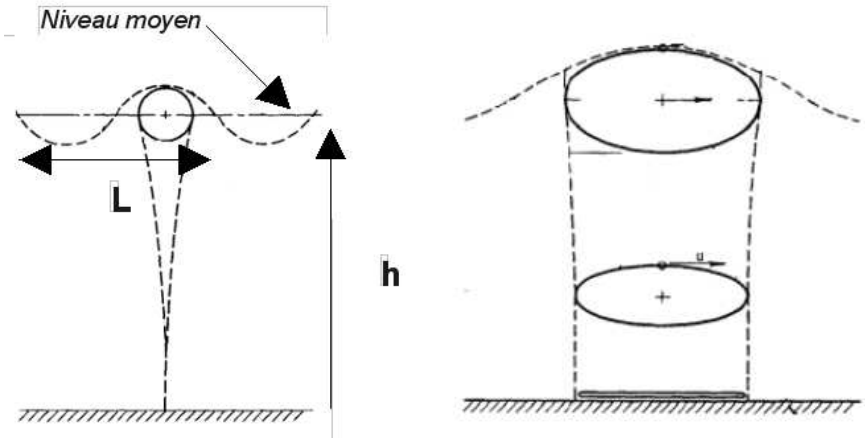
Chapitre 2

MODELISATION PHYSIQUE

Nous allons donc modéliser un partie de l'océan et des obstacles tels que des iles pour observer le comportement des vagues sur celles-ci. Il est donc très important de déterminer notre domaine et se qui le compose. On utilise un modèle physique réduit.

2.1 La houle

La houle est un mouvement ondulatoire de la surface de la mer qui est formé par un champ de vent éloigné de la zone d'observation C'est la partie de l'état de la mer qui se caractérise par son absence de relation avec le vent local. Dans cette acception le mot houle est équivalent à l'anglais « swell » par opposition avec la mer du vent (wind sea). Une fois générées dans les tempêtes, les houles de grandes périodes peuvent se propager sur des dizaines de milliers de kilomètres. Les vagues et la houle à la surface de la mer sont les exemples les plus communs d'ondes de gravité de surface. Plus la distance sur laquelle s'exerce le vent est longue (ce qu'on appelle le fetch), plus les vagues seront hautes avec une certaine valeur de vent. L'eau au sommet de la vague ayant une énergie potentielle plus grande que celle dans les creux, la gravité tend à ramener les deux vers le point médian et induit la propagation de l'onde.



(a) eau profonde (b) eau peu profonde

FIG. 2.1 – trajectoire particule d'eau

Lorsque la hauteur d'eau est élevée, les particule d'eau on une trajectoire circulaire et lorsque

elle tend à diminuer, leur trajectoire s'applatie jusqu'à devenir elliptique. Si $\frac{h}{L} \ll 1$ alors on se situe en eau peu profonde, au contraire si $\frac{h}{L} \gg 1$ alors on se situe en eau profonde. Avec L la longueur d'onde et h la profondeur d'eau jusqu'au fond.

Si on diminue encore la hauteur d'eau, la cambrure augmente (la cambrure est le rapport entre l'amplitude crête à crête de la vague et sa longueur d'onde.) et les particules d'eau effectuant un mouvement elliptique, tende finalement vers un mouvement horizontal. Arrivé à la cambrure critique, la vitesse des particules d'eau dépasse la célérité de la vague, il y a alors déferlement. Les particules d'eau de la surface libre de la crête sortent du système de vagues et chutent dans le creux de la vague du fait de la gravité.

Notre modèl est basé sur les équations de shallow water, de ce fait on fixera toujours $\frac{h}{L} \ll 1$. On appelera alors nos vagues, « vagues de shallow water » différentes de la houle.

2.2 Dimensionnement

Le fait de travailler sur des modèles réduit entraine certaines contraintes. Il faut trouver un compromis entre une grande échelle proche de la réalité et suffisamment petite pour pouvoir reproduire la réalité dans un domaine aux dimensions fixes et pouvoir ainsi observer des phénomènes susceptible de se produire en réalité.

Pour obtenir une modélisation représentative de la réalité, il faut respecter les différentes caractéristiques des éléments constituants le modèle,les longueurs, volumes...doivent être mises à l'échelle mais également tout ce qui concerne la houle, l'amplitude et la durée du test. Plus généralement, tous les éléments intervenant dans l'éxperience doivent être convertis à l'échelle.

Si la profondeur d'eau de notre domaine réel est H, la vitesse V et que l'éxperience dure T alors notre modèl réduit a une profondeur h, une vitesse v et une durée d'éxperience t. De plus on note par « Echelle » la réduction on l'augmentation de l'echelle, dans notre cas $Echelle = \frac{1}{40}$ et c'est une réduction d'echelle.

On aura alors :

$$h = H \times Echelle \tag{2.1}$$

$$v = V \times \sqrt{Echelle} \tag{2.2}$$

$$t = T \times \sqrt{Echelle} \tag{2.3}$$

Le nombre de Froude, est un nombre adimensionnel qui caractérise dans un fluide l'importance relative des forces liées à la vitesse et à la force de pesanteur. Ce nombre apparaît essentiellement dans les phénomènes à surface libre.

$$Fr = \frac{V}{\sqrt{gH}}$$

Grâce à la conservation du nombre de froude on peut trouver une relation entre la réalité et notre modèl réduit :

$$\frac{V}{\sqrt{gH}} = \frac{v}{\sqrt{gh}}$$

On sait par définition qu'une longeur mise à l'échelle est tout simplement divisée par celle-ci (équation (2.1)), on démontre alors facilement les trois équations ci-dessus.

Chapitre 3

MODELISATIONS MATHEMATIQUES

3.1 De Navier-Stokes aux équation de shallow water

Les fluides dynamiques et incompressibles sont décrits par les équations de Navier-Stokes

$$\partial_t u + u\partial_x u + v\partial_y u + w\partial_z u + \frac{1}{\rho}\partial_x P = \nu\Delta u \tag{3.1}$$

$$\partial_t v + u\partial_x v + v\partial_y v + w\partial_z v + \frac{1}{\rho}\partial_y P = \nu\Delta v \tag{3.2}$$

$$\partial_t w + u\partial_x w + v\partial_y w + w\partial_z w + \frac{1}{\rho}\partial_z P = -g + \nu\Delta w \tag{3.3}$$

$$+ \text{ conditions limites}$$

où u est la composante zonale, v méridienne et w la composante verticale de la vitesse. P représente la pression, ρ la densité et ν la viscosité dynamique.

On peut simplifier ces équations grâce à deux très importantes observations :

- Avec un profondeur moyenne de 4km et une longueur moyenne de 10000km les océans sont très plat.
- L'eau de mer a de très faibles variations de densité ($\Delta\rho/\rho \approx 3.10^{-3}$).

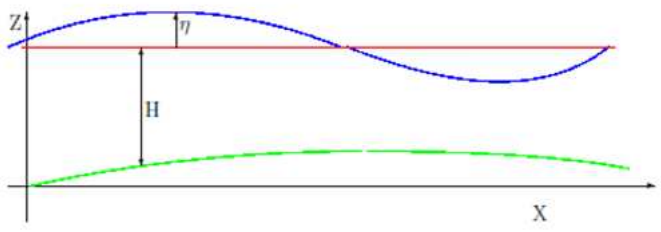


FIG. 3.1 – Configuration de shallow water

On utilise l'équation de conservation du volume :

$$\partial_x u + \partial_y v + \partial_z w = 0 \tag{3.4}$$

En effet elle suggére que $\frac{w}{H}$ est du même ordre que $\frac{u_k}{L}$, où $u_k = \sqrt{u^2 + v^2}$ la vitesse horizontale. On a donc $w \approx \frac{H u_k}{L}$, du fait que $\frac{H}{L} \ll 1$, l'équation (2.3) est alors réduite à $\partial_z P = -\rho g$. Cette approximation hydrosatatique rend le gradient de pression vertical indépendant de la vitesse du fluide.

De ce fait on peut dire que la vitesse horizontale est indépendante de la profondeur et on a alors :

$$\partial_t u + u\partial_x u + v\partial_y u + \frac{1}{\rho}\partial_x P = \nu \Delta u \tag{3.5}$$

$$\partial_t v + u\partial_x v + v\partial_y v + \frac{1}{\rho}\partial_y P = \nu \Delta v \tag{3.6}$$

$$+ \ conditions \ limites \tag{3.7}$$

Les conditions limites sont obtenues grâce à la surface libre de l'océan. La variations de cette surface est notée η et le mouvement des particules fluides sont définis par :

$$\frac{D(H+\eta)}{Dt} = w(\eta) \tag{3.8}$$

Si on dérive par z l'équation de conservation du volume et que l'on fait l'hypothèse que les champs de vitesse horizontaux sont constant selon la profondeur, soit $\partial_z u = \partial_z v = 0$, on peut écrire : $\partial_z w = Cste$

Le gradiant vertical de la vitesse verticale étant constant, il vient naturellement

$$w(\eta) = (H+\eta)\partial_z w$$

Avec $\frac{D}{Dt}$ la dérivée Lagrangienne horizontale et $H = cste$, on obtient donc :

$$\partial_t \eta + u\partial_x \eta + v\partial_y \eta - (H+\eta)\partial_z w = 0 \tag{3.9}$$

Et avec l'équation (2.4) :

$$\partial_t \eta + u\partial_x (H+\eta) + v\partial_y (H+\eta) + (H+\eta)(\partial_x u + \partial_y v) = 0 \tag{3.10}$$

Pour finir on utilise une approximation hydrostatique, en effet la pression à la profondeur d est équivalente à $P = \rho g(\eta + d)$ et le gradient de pression horizontale n'est définit que par la surface libre.

D'ou :

$$\partial_t u + u\partial_x u + v\partial_y u + g\partial_x \eta = \nu \Delta u \tag{3.11}$$

$$\partial_t v + u\partial_x v + v\partial_y v + g\partial_y \eta = \nu \Delta v \tag{3.12}$$

$$\partial_t \eta + \partial_x[(H+\eta)]u + \partial_y[(H+\eta)v] = 0 \tag{3.13}$$

Les équations (2.10), (2.11), (2.12) sont totalement indépendantes de z et sont appelées équations de shallow water (ou de Saint Venant). Elles permettent de calculer en chaque point du maillage les champs de vitesse u et v et les variations de la surface libre η.

On néglige la force de coriolis qui a très peu d'impact sur notre étude. En effet à notre échelle la force de coriolis a un impact sur de longue éxperience alors que nous n'effectuerons que des éxperiences de moins de un jour.

3.2 La houle

Une houle se caractérise en première approximation par une hauteur, double de l'amplitude, (de quelques décimètres à quelques mètres) et une longueur d'onde ou une période (généralement de l'ordre d'une dizaine de secondes).

En réalité, il s'agit d'un phénomène qui n'est pas périodique et qui peut s'interpréter comme une somme d'une infinité de composantes sinusoidales infiniment petites .
Pour définir la vitesse de notre onde on réalise certaines approximations pour simplifier les équations de shallow water. En effet, si $\frac{1}{2}u^2 \ll g\eta$, si la viscosité $\nu \ll g\eta L/u$ et $\eta \ll H$ alors les équations de shallow water deviennent :

$$\partial_t u = -g\partial_x \eta \tag{3.14}$$
$$\partial_t v = -g\partial_y \eta \tag{3.15}$$
$$\partial_t \eta = -H(\partial_x u + \partial_y v) \tag{3.16}$$

D'ou on tire :

$$\partial_{tt}\eta = gH \triangle \eta \tag{3.17}$$

C'est une équation de propagation des ondes de la forme : $\partial_{tt}\eta = c^2 \triangle \eta$
De plus si on cherche une solution de la forme $\eta = Af(x - ct)$, on en déduit rapidement $Ac^2 f"(x - ct) = AgHf"(x - ct)$.

On arrive alors à

$$c^2 = gH \tag{3.18}$$

Chapitre 4

MODELISATION NUMERIQUE

4.1 Introduction

4.1.1 La base du programme

Lors de la programmation j'ai taché de réaliser mon programme pas à pas, en intégrant petit à petit les differents éléments . J'ai utilisé une méthode des differences finies (la méthode d'Euler Explicite) pour modéliser les vagues. On se situe dans un environnement bi-dimentionnel, donc les diffentes vitesses dépendent des variables x,y et du temps. Tout d'abords, on discrétise en espace les équations de shallow water :

Sachant que : $\partial_x u = \frac{uo(i_x+1,i_y)-uo(i_x-1,i_y)}{2\Delta x}$

et $\partial_{xx} u = \frac{\partial_x u(i_x+\frac{1}{2},i_y)-\partial_x u(i_x-\frac{1}{2},i_y)}{\Delta x}$ avec $\partial_x u(i_x+\frac{1}{2},iy) = \frac{uo(i_x+1,i_y)-uo(i_x,i_y)}{\Delta x}$

D'ou $\partial_{xx} u = \frac{1}{\Delta x^2}(uo(i_x+1,i_y) - 2uo(i_x,i_y) + uo(i_x-1,i_y)$

De plus pour facilité la programmation on note $uo = u_{old}$ la valeur de la composante u au temps t et $un = u_{new}$ au temps $t+1$.

$$\partial_t u = \frac{un(i_x,i_y) - uo(i_x,i_y)}{\Delta t}$$

On finit par discrétisé en temps et les équation de shallow water sont alors sous la forme discrétisées :

$$\begin{aligned} un(i_x,i_y) &= uo(i_x,i_y) + \Delta t\Big[\\ &+ \frac{\nu}{\Delta x^2}\Big(uo(i_x+1,i_y) - 2uo(i_x,i_y) + uo(i_x-1,i_y)\Big) \\ &+ \frac{\nu}{\Delta y^2}\Big(uo(i_x,i_y+1) - 2uo(i_x,i_y) + uo(i_x,i_y-1)\Big) \\ &- uo(i_x,i_y)\frac{uo(i_x+1,i_y) - uo(i_x-1,i_y)}{2\Delta x} \\ &- vo(i_x,i_y)\frac{uo(i_x,i_y+1) - uo(i_x,i_y-1)}{2\Delta y} \\ &- g\frac{ho(i_x+1,i_y) - ho(i_x-1,i_y)}{2\Delta x}\Big] \end{aligned} \tag{4.1}$$

$$
\begin{aligned}
vn(i_x, i_y) &= vo(i_x, i_y) + \Delta t\Big[\\
&+ \frac{\nu}{\Delta x^2}\Big(vo(i_x+1, i_y) - 2vo(i_x, i_y) + vo(i_x-1, i_y)\Big) \\
&+ \frac{\nu}{\Delta y^2}\Big(vo(i_x, i_y+1) - 2vo(i_x, i_y) + vo(i_x, i_y-1)\Big) \\
&- uo(i_x, i_y)\frac{vo(i_x+1, i_y) - vo(i_x-1, i_y)}{2\Delta x} \\
&- vo(i_x, i_y)\frac{vo(i_x, i_y+1) - vo(i_x, i_y-1)}{2\Delta y} \\
&- g\frac{ho(i_x, i_y+1) - ho(i_x, i_y-1)}{2\Delta y}\Big]
\end{aligned} \tag{4.2}
$$

Pour les composantes u et v, on reconnait deux termes de diffusion, deux termes d'advection et un terme lié au gradient de pression.

$$
\begin{aligned}
hn(i_x, i_y) &= ho(i_x, i_y) + \Delta t\Big[\\
&- \frac{uo(i_x+1, i_y)ho(i_x+1, i_y) - uo(i_x-1, i_y)ho(i_x-1, i_y)}{2\Delta x} \\
&- \frac{vo(i_x, i_y+1)ho(i_x, i_y+1) - vo(i_x, i_y-1)ho(i_x, i_y-1)}{2\Delta y}
\end{aligned} \tag{4.3}
$$

le calcul de ces trois valeurs est la base du programme. En effet, on calcul ces valeurs pour chaque point du domaine, pour ce faire on réalise deux boucles « do » allant de $i_x = 1$ jusqu'a n_x puis de $i_y = 1$ jusqu'a n_y. On définit n_x par le nombre de points sur l'axe des x et n_y le nombre de points sur l'axe des y.

De plus, on nomme lx et ly les longueurs réspectives du domaine suivant x et y, de ce fait on peut calculer la sensibilité $\Delta x = \frac{lx}{nx}$ et $\Delta y = \frac{ly}{ny}$, les pas suivant x et y.

4.1.2 La méthode d'Euler explicite

La méthode d'Euler est une méthode numérique élémentaire de résolution d'équations différentielles, pour nous, du premier ordre en temps. Elle est dite explicite lorsque pour calculé un point au temps $t+1$, on utilise les points au temps t. En effet si on effectue un développement de Taylor sur une fonction f(x), on aura :

$$
f(t+\Delta t) = f(t) + \Delta t f^{'}(t) + \frac{\Delta t^2}{2} f^{"}(t) + \frac{\Delta t^3}{6} f^{"'}(t) + \dots
$$

Alors :

$$
\frac{f(t+\Delta t) - f(t)}{\Delta t} = f^{'}(t) + \Delta t\Big(\frac{\Delta t}{2} f^{"}(t) + \frac{\Delta t^2}{6} f^{"'}(t) + \dots\Big)
$$

L'erreur commise sur le calcul est alors du même ordre de grandeur que Δt le pas en temps du programme, c'est une méthode du premier ordre en temps.

On effectue la même approche pour l'espace,les développements de Taylor seront :

$$f(t+\Delta x) = f(x) + \Delta x f^{'}(x) + \frac{\Delta x^2}{2} f^{"}(x) + \frac{\Delta x^3}{6} f^{"'}(t) + \dots$$

et

$$f(t-\Delta t) = f(x) - \Delta x f^{'}(x) + \frac{\Delta x^2}{2} x^{"}(x) - \frac{\Delta x^3}{6} f^{"'}(x) + \dots$$

d'ou

$$\frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} = f^{'}(x) + \Delta x^2 \Big(\frac{\Delta x^2}{6} f^{"'}(x) + \dots \Big)$$

Pour les dérivées secondes la marche à suivre reste la même et :

$$\frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{2\Delta x^2} = f^{"}(x) + \Delta x^2 \Big(\frac{\Delta x^2}{12} f^{"'}(x) + \dots \Big)$$

L'erreur commise sur le calcul est alors du même ordre de grandeur que Δx^2 le pas en espace du programme, c'est une méthode du deuxieme ordre en espace.

La methode d'Euler est connue pour être particulierement efficace pour propagée les erreurs de calcul, c'est pourquoi il faut être très vigilent dans le choix des pas de temps et d'espace pour garder une bonne présision.

4.1.3 Enregistrement de données

Le but du programme est donc de calculer les champs de vitesses horizontaux et la hauteur d'eau, il faut donc pouvoir stocker ces données. Pour chacune de ces composantes et pour chaque séries de pas de temps, on devra stocker la valeur de chaque noeud du maillage.

Prendre toutes ces données à chaque pas de temps n'est pas très intéressant, en effet celui-ci étant lors de nos expérience très inférieur à l'ordre de grandeur temporelle du phénomen observé. C'est pourquoi on crée deux boucles de temps, un allant de 0 à $nt1$ et l'autre de 0 à $nt2$. Ainsi on peut appeler la subroutine nous permettant de stocker les données entre ces deux boucles et sauter un certain nombres de pas de temps pour les enregistrer.

On appelle ces subroutines « OUTSUBu »pour stocker le champ de vitesse u, « OUTSUBv »pour v et « OUTSUBh »pour la hauteur d'eau h. Elles enregistrerons donc tous les $nt1$ pas de temps, après une éxperience on possède $nt2$ fichiers de u, v et h. Les fichiers commence par l'abréviation de shallow water « sw »suivi de la composante étudié, du numéro de l'éxperience puis du numéro correspondant au pas de temps $nt2$, soit par exemple :

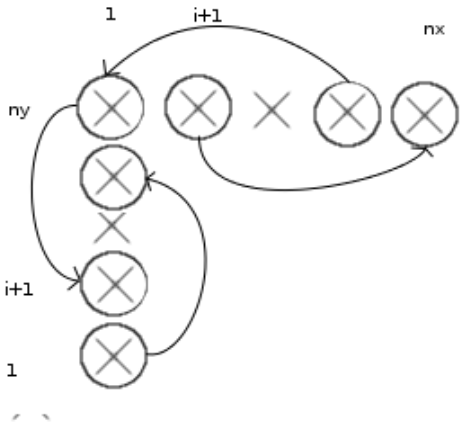
$$sw\_h100001\_1027.data$$

correspond à la mesure de h au 27^{eme} tour de la boucle n_o : 1, c'est à dire $27 \times nt1$ pas de temps, de la première éxperience.
(voir« Subroutine de stockage »en annexe page a)

4.2 Conditions limites et initiales

Il se pose tout de suite la question des conditions initales et des conditions limites, en effet on travail dans un domaine bidimentionnel carré, de plus on utilise, dans les équations de shallow water, des points avec des indices tels que i_{x-1} ou i_{x+1}, de ce fait on a un problème au bords de notre domaine. c'est pourquoi on utilise des conditions de bords périodiques, de ce fait chaque point sortant du domaine par une limite réapparaît à son opposée.(Voir « Conditions de bors » en annexe page b)

On aura alors :



$$\begin{aligned} uo(1, i_y) &= un(n_x - 1, i_y) \\ uo(n_x, i_y) &= un(2, i_y) \end{aligned}$$

$$\begin{aligned} uo(i_x, 1) &= un(i_x, n_y - 1) \\ uo(i_x, n_y) &= un(i_x, 2) \end{aligned}$$

Pour tous x allant de 1 à nx et pour tous y allant de 1 à ny.

Pour commencer on prendra comme condition initiale une gaussienne normalisée définie par :

$$\begin{aligned} x &= \frac{(ix - (nx/2.)}{nx} \\ y &= \frac{(iy - (ny/2.)}{ny} \end{aligned}$$

$$uo(i_x, i_y) = Aexp((x^2 + y^2)/L^2) \tag{4.4}$$

Il faut donc appliquer ces valeurs initiales à tous les points du domaines en effectuant deux boucles, de plus on stocke ces valeurs dans un fichier, on pourra ainsi vérifier si il n'y a pas d'erreurs sur les conditions initiales. (Voir « Conditions initiales » en annexe page b)

4.3 Diffusion

4.3.1 La viscosité cinématique

Pour commencer on programme seulement les termes de diffusion, en effet grâce à la viscosité cinématique, on peut se ramener à quelque chose que l'on connait. Si on ne programme que les termes des équations de shallow water ne dependant que de la viscosité, on se ramene à une simple équation de diffusion.

On aura :

$$\partial_t u = \nu \partial_{xx} u \tag{4.5}$$

4.3.2 Similitude avec l'équation de la chaleur

On rapelle l'équation de la diffusion de la chaleur : $\partial_t Q = \kappa \partial_{xx} Q$ avec κ la diffusité thermique.
On connait la solution de cette équation, on pourra donc plus facilement contrôler notre travail. On discretise alors l'équation (4.5) :

$$un(ix, iy) = uo(ix, iy) + \Delta_t \Big[\frac{\nu}{{\Delta_x}^2} (u(ix+1, iy) - 2uo(ix, iy) + uo(ix-1, iy)) \Big] \tag{4.6}$$

Si on pose $\lambda = \frac{\nu \Delta_t}{{\Delta_x}^2}$

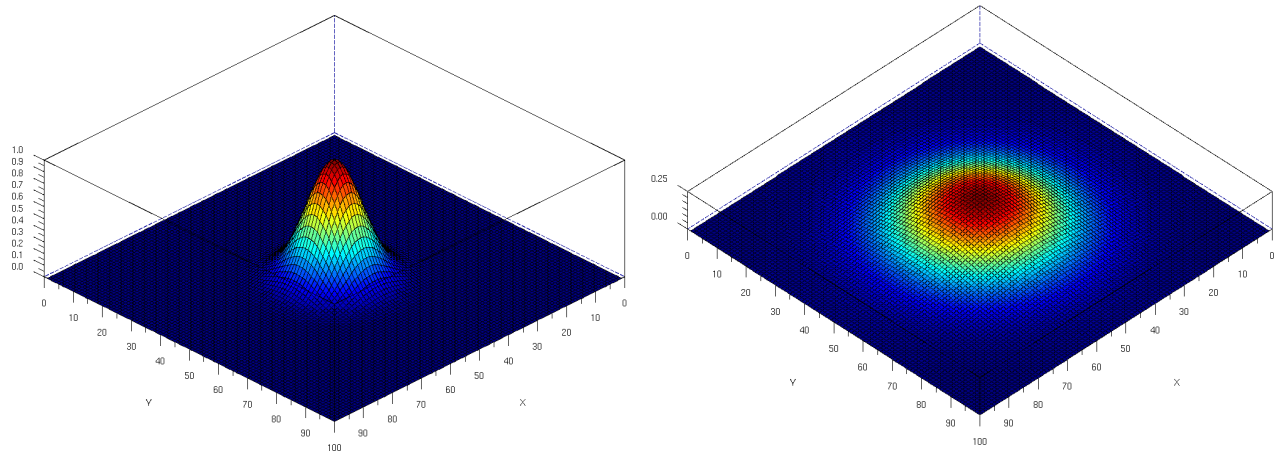
On obtient : $un(i_x, i_y) = uo(i_x, i_y)(1 - 2\lambda) + \lambda \Big(uo(i_{x+1}, i_y) + uo(i_{x-1}, i_y) \Big)$

$un(i_x, i_y)$ est une combinaison convexe si $\lambda \leq \frac{1}{2}$, c'est à dire que tous ces coéficient sont positifs et leur somme est 1. Une combinaison convexe admet toujours une solution comprise dans l'enveloppe convexe, c'est à dire le plus petit espace contenant les points constituants la combinaison. Il est donc impératif de garder $\lambda \leq \frac{1}{2}$ pour garder une bonne stabilité.

On programme alors cette équation on prennant soin de satisfaire la condition de stabilité. (Voir « Termes Diffusifs » en annexe page c)

4.3.3 Quelques résultats

Grâce à nos conditions initiales, on peut observer la diffusion du champ de vitesse u



(a) t initial (b) t final

FIG. 4.1 – diffusion du champ de vitesse u

On a pris comme condition initiale la gaussienne d'équation (4.4) sur u, une viscosité $\nu = 10^{-2} m^2.s^{-1}$ et on effectue mille pas de temps $\Delta t = 10^{-3} s$, soit une durée totale d'experience de 1 *seconde*. Le domaine est de dimension 1×1 mètres et le maillage est de 100×100

4.4 Advection

Les équations de shallow water contiennent des termes advectifs tel que :

$$vo(i_x, i_y)\frac{vo(i_x, i_{y+1}) - vo(i_x, i_{y-1})}{2\Delta y}$$

Pour commencer on transformera le terme multiplicatif non linéaire en constante pour simplifier l'analyse des résultats.

4.4.1 Condition de stabilité CFL

Il est indispensable de respecter la condition de Courant-Friedrichs-Lewy pour un calcul stable : la propagation des ondes durant un pas de temps Δt doit être inférieure à la plus petite dimension de la maille. Cette condition est nécessaire pour la convergence de certaines résolutions numérique d'équations diff érentielles partielles. Elle apparait lorsque l'on utilise des schemas éxplicite. On aura donc à respecter :

$$\frac{v\Delta t}{2\Delta x} \leq 1$$

En effet, l'information ne doit pas être propagée trop vite. Pour calculer un point, on a besoin du point placé avant et après celui-ci, donc si on se deplace de plus de $2\Delta x$ alors l'information est perdu et on obtient des resultat faux. En effet $v\Delta t$ correspond au déplacement du à l'advection à chaque pas de temps, si ce déplacement est trop important et supérieur à deux fois le pas d'espace, le nouveau point sera calculé avec les mauvaises valeurs.

4.4.2 Quelques résultats

On obtient de bon resultat, avec une très bonne visualisation de l'advection grâce aux conditions de bords, ici on peut voir une advection suivant l'axe des x. On prend comme condition initiale la gaussienne d'équation (4.4) sur u, une viscosité $\nu = 10^{-2} m^2.s^{-1}$ et on effectue mille pas de temps $\Delta t = 10^{-3} s$, soit une durée totale d'experience de 1 *seconde*. Le domaine est de dimension 1×1 mètres et le maillage est de 100×100. On fixe comme vitesse initiale, $uo = 0.5 m.s^{-1}$

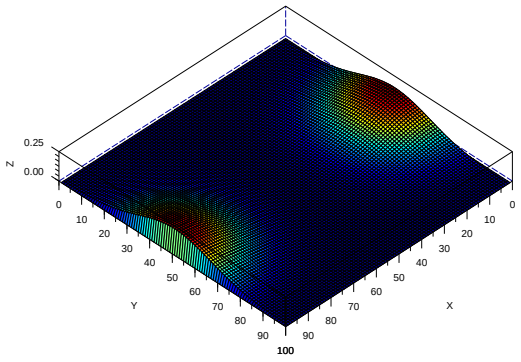


Fig. 4.2 – Phenomen d'advection

On peut donc rajouter les termes réstants et pour finir remplacer les termes non linéaires u et v à la place des constantes.

4.5 Les équation de shallow water completes

On a modélisé entierement les équations de shallow water, on peut maintenent visualiser la surface libre (Voir « Equation shallow water complètes »en annexe page c). On effectue les calculs sur un maillage 100×100 et le domaine est de dimension $1m \times 1m$. On fixe une condition initiale « gaussienne »sur η, on fixe une viscosité $\nu = 10^{-4} m^2.s^{-1}$ et $\Delta t = 10^{-5} s$. La hauteur d'eau est $H = 1m$.
On a alors :

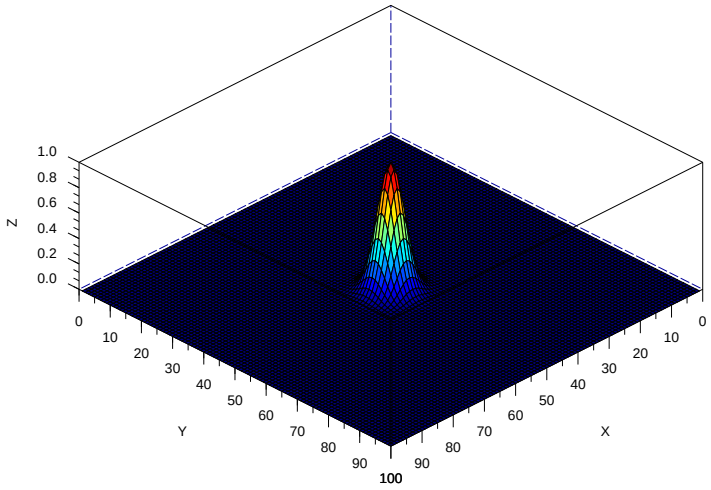


FIG. 4.3 – t initial

On obtient après 2500 pas de temps de 10^{-6}s, soit après 2.5ms :

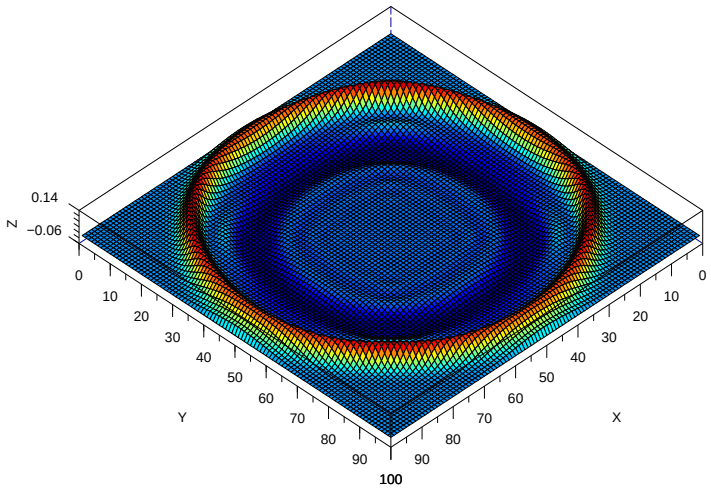


FIG. 4.4 – t final

Qualitativement le resultat semble correct, cependant il faut vérifier que la vitesse de l'onde

correspond à la réalitée. Théoriquement on a :

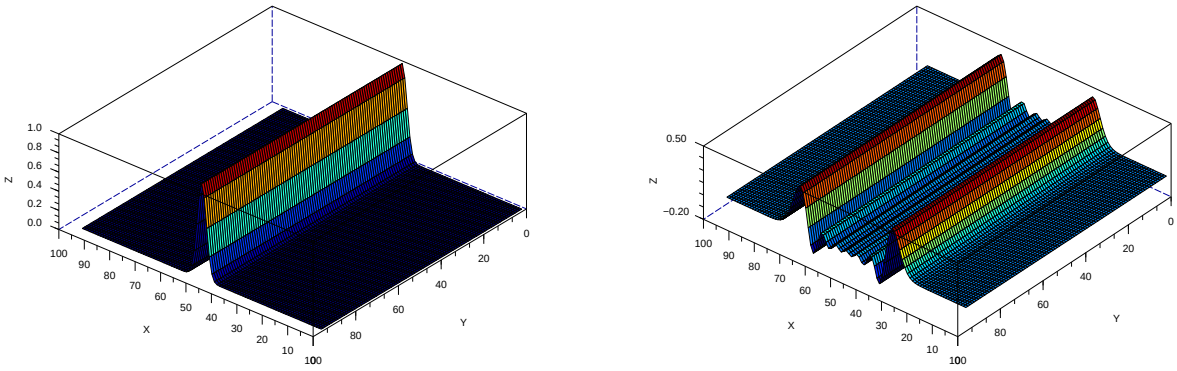
$$\sqrt{gH} = c$$

donc

$$\sqrt{9.81 \times 1} = 3.13 m.s^{-1}$$

ici l'onde parcourt 0.5m en 0.16 s, soit $3.125 m.s^{-1}$. Pour une onde circulaire, la vitesse possède une dépendance à la distance parcourue, cependant à notre echelle les variation son minime.

On teste le model avec des conditions initiales differentes. On prend une gaussienne sur η ne dependant que de x, on garde inchangées toutes les autres conditions.



(a) *temps* initial ($t = 0$)

(b) *temps* final ($t = 10ms$)

FIG. 4.5 – gaussienne sur x

Les calculs sont exactement les mêmes, la vitesse de phase reste la même à $3.13 m.s^{-1}$.

On se propose de calculer l'expression de la vitesse des deux vagues crées. En effet on a définit l'expression (3.14) lors du paragraphe sur la houle , elle lie le champ de vitesse u et la perturbation à la surface η, soit :

$$\partial_t u = g \partial_x \eta$$

On cherche des solution de la forme $\eta = {\eta_0}^+(x - ct)$ et $u = {u_0}^+(x - ct)$

On a donc $c{u_0}^{+'}(x - ct) = g{\eta_0}^{+'}(x - ct)$

et ${u_0}^{+'}(x - ct) = \frac{g}{\sqrt{gH}} {\eta_0}^{+'}(x - ct)$

On a calculé plus tôt que $c = Cste = \sqrt{gH}$, de plus on considère qu'il n'y a pas de constantes. On a donc a $t = 0$:

$${u_0}^+(x) = \sqrt{\frac{g}{H}} {\eta_0}^+(x) \tag{4.7}$$

On avait fixé comme condition initial ${\eta_0}^+ = Aexp - (x^2/L)$, avec $A = 1$ et $L = \frac{1}{1500}$.
On pourra donc directement crée des vagues progressives en programment cette vitesse comme condition initiale.

4.6 Modélisation des iles

Pour modéliser les iles, on fixe les champs de vitesse horizontaux u et v à zero, cela aura l'effet d'un mur. Cependant la hauteur h dans ces intervales ne doit pas être prise en compte. On constate que lorsque l'on a besoin de $ho(i_x + 1, i_y)$ au bord d'une île par exemple, il est toujours multiplié par un terme de vitesse au même point, soit $uo(i_x + 1, i_y)$ qui est fixé à zero. Les iles n'entrainent donc pas de problème de programmation.

4.6.1 La diffraction

Pour comment on se propose de modéliser une vague passant entre deux digues pour observer la diffraction de la vague. On effectue les calculs sur un maillage 200×200 et le domaine est de dimension $10m \times 10m$. On garde notre condition initiale « gaussienne »sur η et on fixe un condition initiale sur u de la forme de l'équation (4.7), On fixe une viscosité $\nu = 10^{-3} m^2.s^{-1}$ et $\Delta t = 10^{-5} s$. La hauteur d'eau est $H = 40cm$.

La condition initiale sur u sera alors :${u_0}^+(x) = \sqrt{\frac{9.81}{0.4}} exp - ((x - x_0)^2/1500)$, avec x_0 le décalage que l'on veut appliquer a la gaussienne.

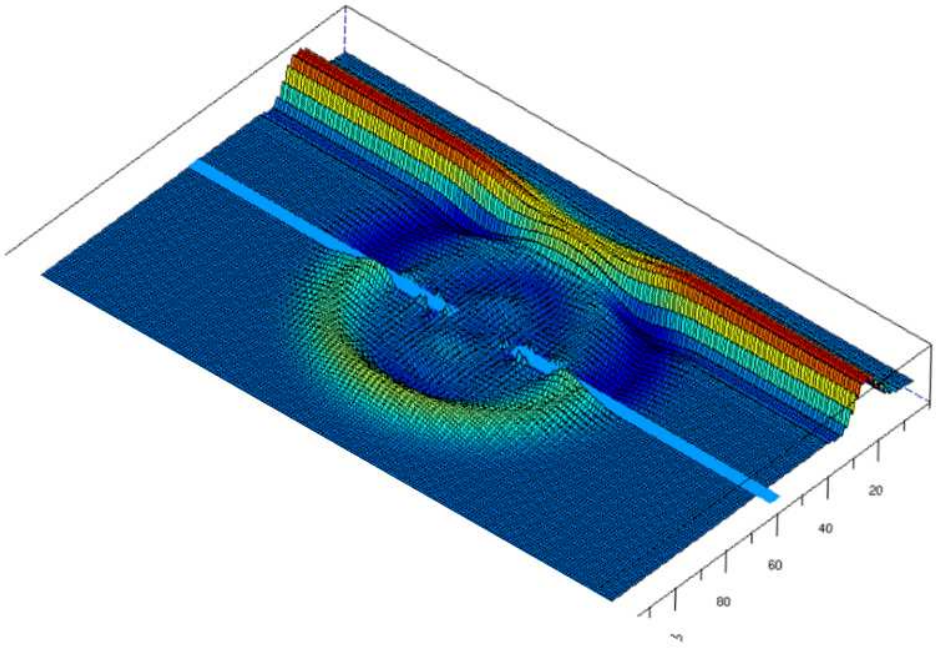


FIG. 4.6 – Diffraction d'un vague entre deux digues

On observe parfaitement la diffraction de l'eau sur l'ouvrage, de plus on calcul la vitesse de propagation de l'onde :

$$\begin{aligned} c &= \sqrt{gH} \\ c &= \sqrt{9.81 \times 0.4} \\ c &= 1.98 m.s^{-1} \end{aligned}$$

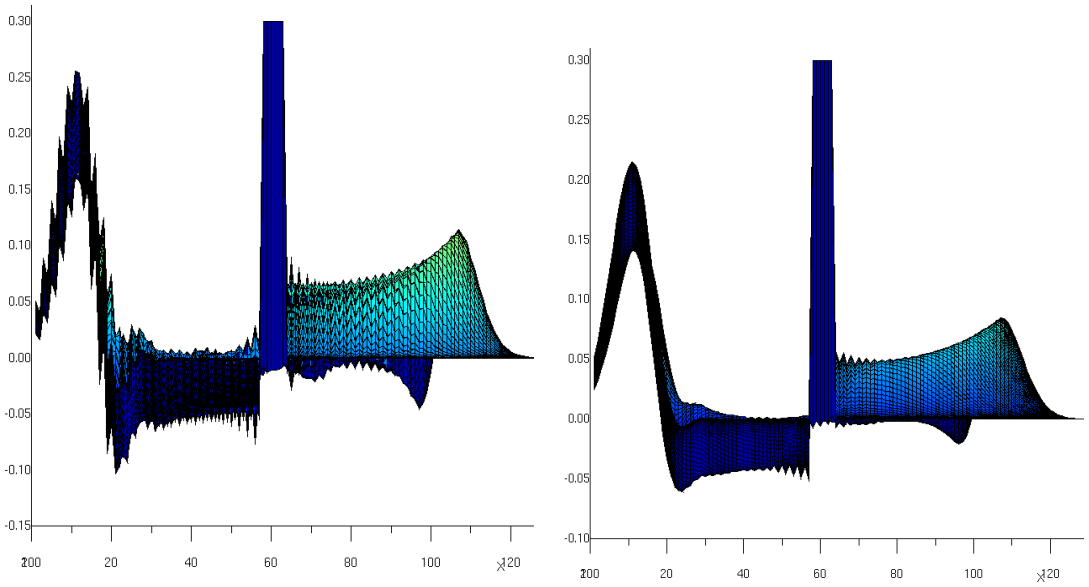
Sur l'image, la vague vient de se refléchir contre la digue et se propage donc suivant les x négatifs, quant à l'onde diffractée, elle se dirige suivant les x positifs.

les digues sont placées à 4m de la vague, elle met donc $2.02 seconde$ à parcourir cette distance. Lors de la simulation la vague atteint la digue après 2000×100 pas de temps , soit 2seconde. De plus on observe que l'onde diffractée et réflechie garde la même vitesse.

Cependant on observe un problème sur l'onde réfléchie et difractée, en effet elles semblent « cisaillées ». Ce problème est du à la résolution paire et impaire. En effet si on se place seulement sur l'axe des abscisses et que l'on regarde les équation de shallow water, on remarque que $u(2i)$ dépend de $h(2i+1)$, de même $h(2i+1)$ dépend de $u(2i)$. De la même manière, $u(2i+1)$ dépend de $h(2i)$ et $h(2i)$ de $u(2i+1)$.

On constate alors l'existance de deux systêmes de points ne se recoupant jamais. Les erreurs d'un systême ne se propage que dans celui-ci et vice et versa. Ainsi ce « cisaillement » correspond aux differentes erreurs propagées dans les deux systêmes. Cependant grâce à la viscosité, on peut faire le lien entre les deux .

On peut observer l'effet de la viscosité sur ces erreurs sur les figures ci-dessous représentant la figure (4.6) ci-dessus vu de coté avec différentes valeurs de viscosité.



(a) $\nu = 10^{-2} m^2.s^{-1}$ (b) $\nu = 1 m^2.s^{-1}$

FIG. 4.7 – Erreur lié à la parité

On constate qu'en multipliant la viscosité par 100 on lisse presque toutes les erreurs, cependant il ne faut pas en abuser, si on augmente trop la viscosité, on rique de ne plus respecter les conditions de stabilité du programme ou de trop tassé les résultats et de ne plus rien observer.

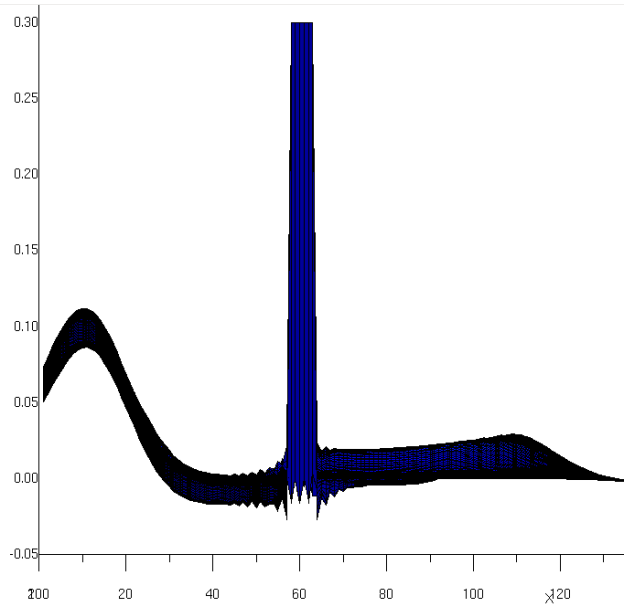


FIG. 4.8 – ($\nu = 10m^2.s^{-1}$)

Avec cette grande viscosité, on constate que la diffusion est beaucoup plus rapide que dans les cas precedent, les phénomen sont trop rapides, on ne peut plus rien observer.

4.6.2 Palm Deira

Le projet de Palm Deira se situe dans les Emirats Arabes au sud-est de la Péninsule Arabique, à proximité de la ville de Dubai. Palm DEIRA est le troisième palmier et le quatrième projet gigantesque construits sur ces cotes.

Lors d'un stage en 2008 à la sogreah, j'ai pu participer à l'étude d'un modèle réduit de l'entrée d'un chenal de cette île.(Voir « Plan Palm Deira »en annexe page e). Le but était de tester la solidité de l'ouvrage et d'observer les perturbations à l'intérieur du chennal.

Aujourd'hui, j'essaie de recréer numériquement cette situation. Cette étude est réalisée au $40^{\text{ème}}$ dans un bassin mesurant $30m \times 30m$ (Voir « Plan Palm Deira »en annexe page d) .

Grâce au plan du bassin j'ai pu modéliser les deux digues appelées « Crown » et « Crescent ». Cela fut laborieux car j'ai du définir chaque noeuds du maillage. En effet ces deux îles sont fabriquées à l'aide de 70 boucles dans le programme (Voir « Exemple programmation île »en annexe page f). On pourra regler la hauteur d'eau dans le programme en changeant ho pour la hauteur des digue dh.

Du fait de la courte durée du stage, on ne pourra pas modéliser le fond marin et la pente des digues, cependant il est intéressant d'observer le comportement de la houle sur ces îles qui peuvent alors représenter des falaises. De plus il existe des modèls numérique pour les enrochements. En effet ils peuvent êtres asimilés à des éponges absorbant les vagues et ainsi représenté la « réalité ». Cependant ces modèls sont compliqués, difficiles à mettre en place et peu fiables. C'est pourquoi on se contentera de la modélisation actuelle.

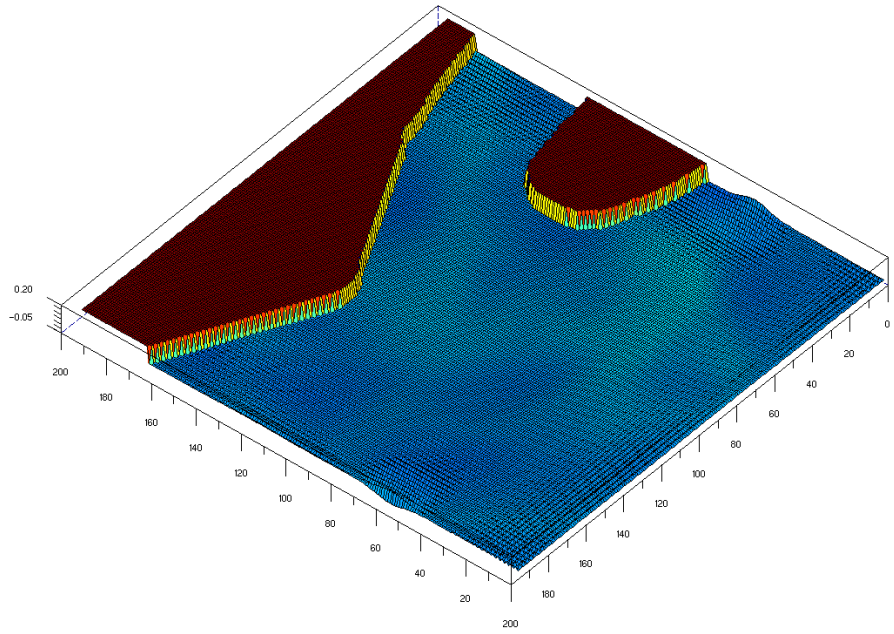


FIG. 4.9 – modelisation des digues de Palm deira

4.7 Le forcage de l'océan

La quasi totalité du comportement dynamique et thermique de l'océan est dictée par l'intermédiaire de sa surface, plus précisément l'interface océan/atmosphère. Partout ailleurs les frontières de l'océan sont solides et sont composées de roches voire parfois de glace. L'océan reçoit donc pratiquement toute son énergie par sa surface, sous forme d'énergie cinétique par l'action du vent.

On modélisera se forçage par un oscillateur harmonique amortie d'équation :

$$Fexp(-\frac{(x-xo)^2}{L^2})cos(\frac{2\pi t}{T})$$

Cette modélisation est très simpliste mais nous permet d'obtenir une bonne approximation du forçage océanique. Cette éxpression s'additionne au reste des termes composant le champ de vitesse u. On ne pourra donc pas directement régler l'amplitude des vague en faisant varier F.

F représente l'amplitude du forcage, dans notre cas $0 \leq F \leq \frac{1}{2}$ dépassé cette limite le programme explose. xo correspond au décalage que l'on veut appliqué à notre zone d'excitation et L agit sur la largeur de cette zone. Pour finir T est la période de l'excitation.

On réalise plusieurs éxperience avec differentes conditions initiales. On constate que pour différentes période, le bassin réagit differemment. En effet pour certaine période le bassin rentre en raisonnance du fait de sa géometrie. A une période de 10 secondes, l'amplitude des vagues est de 0.05 m sur le modèl, soit 2m en réalité

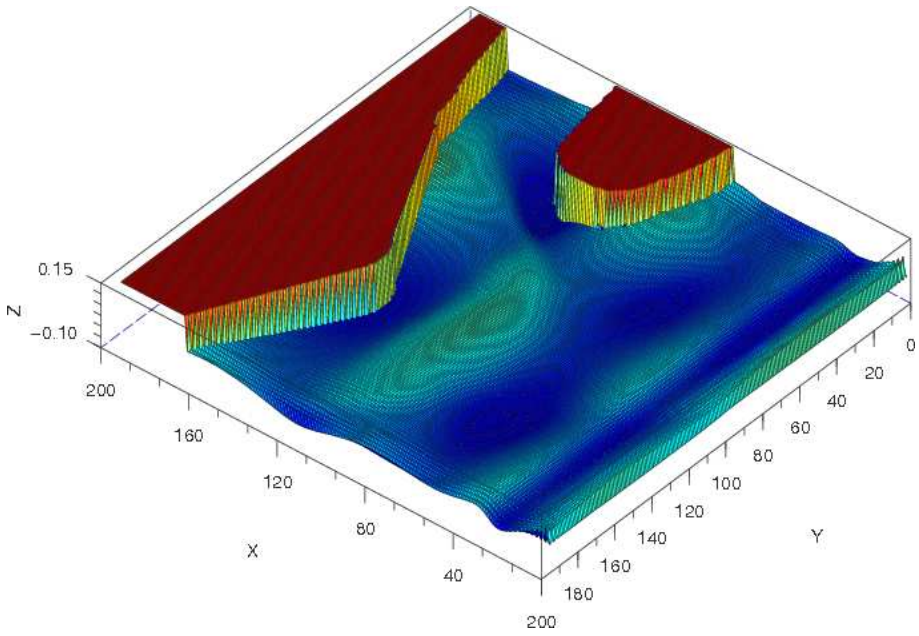


FIG. 4.10 – Forcage de période $T = 10s$

On remarque qu'à une période de forcage de 100 secondes, l'amplitude des vagues est deux à trois fois plus élevées. En effet elles oscillent entre 0.5 a 0.15 m sur le modèl, soit 2m à 6m en réalité.

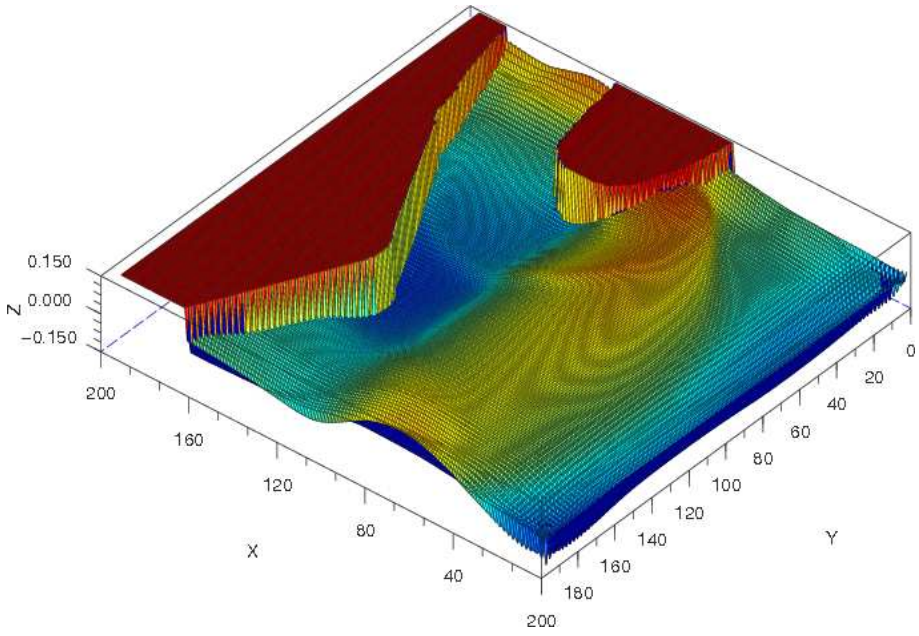


FIG. 4.11 – Forcage de période $T = 100s$

Il aurait été très intéressant de pouvoir modéliser le fond marin, la pente de la digue et l'absorption engendrée par les enrochemment contituant celle-ci. On aurait pu ainsi obtenir des résultats proches de la réalité et ainsi pouvoir les exploiter.

CONCLUSION

Ces neuf semaines ont été très enrichissantes, le laboratoire m'a laissé une grande liberté de choix et d'actions, ce qui m'a permis de m'épanouir pleinement. Ce stage au LEGI m'a permis d'apprendre beaucoup d'éléments intéressants sur le numérique et sur le fonctionnement d'un laboratoire publique. De plus j'ai pu appliquer les connaissances acquises tout au long de ma licence. Il aura été aussi pour moi l'occasion d'apprendre des méthodes de travail scientifique qui pourront m'être très utile pour mon avenir professionnel.

Entre autre, j'ai pu me familiariser avec divers outils informatiques, tels que le language latex, le fortran, l'utilisation du logiciel Scilab et le système d'exploitation Unix, qui sont très utilisées et me sont etrémement utiles.

J'ai trouvé la mise en oeuvre d'un programme tel que celui-ci très intéressante tant dans la conception que dans l'utilisation. J'ai apprécié la rigueur nécéssaire et l'avancement pas à pas de la conception. J'ai trouvé d'autant plus intéressant de créer et d'utiliser mon propre programme.

La durée de stage étant un peu courte et les derniers élements du problème un peu compliqué, ne pas pouvoir terminer entièrement ce programme reste pour moi un grand regret. Cependant, les résultats que j'ai pu obtenir son pour moi très satisfaisant et encouragent.

ANNEXE

Subroutine de stockage

```
*****************************************************************
**                                                             **
**   subroutine    OUTSUB H                                    **
**                                                             **
*****************************************************************
SUBROUTINE OUTSUBH(step,expnum)
IMPLICIT NONE
INTEGER ::   nx,ny
PARAMETER   (nx=100,ny=100)
INTEGER ::  ix,iy
REAL(4) ::  aa(nx,ny)
INTEGER ::  step,expnum
REAL(8) ::  un(nx,ny),vn(nx,ny), hn(nx,ny)
COMMON / newcom  /  un,vn,hn
CHARACTER(4)  :: filename
CHARACTER(6)  :: expnumber
CHARACTER(4)  :: stepnumber
CHARACTER(5)  :: fileext
CHARACTER(10) :: filenamel
CHARACTER(11) :: filenamell
CHARACTER(15) :: filenamelll
CHARACTER(20) :: filenamef
*****************************************************************
**                                                             **
**  On nomme le fichier                                        **
**                                                             **
*****************************************************************
filename="sw_u"
fileext=".data"
WRITE(expnumber,FMT='(I6)') 100000+ expnum
WRITE(stepnumber,FMT='(I4)') 1000+ step
filenamel = filename // expnumber
filenamell = filenamel // "_"
filenamelll = filenamell // stepnumber
filenamef = filenamelll // fileext
OPEN(UNIT=13,STATUS='NEW',FORM='FORMATTED',
FILE=filenamef)
DO  ix=1,nx
DO  iy=1,ny
aa(ix,iy)=REAL(hn(ix,iy),4)
ENDDO
ENDDO

WRITE(13,*)  ((aa(ix,iy),iy=1,ny),ix=1,nx)

RETURN
END SUBROUTINE OUTSUBH
```

Conditions de bords

```
************************************************************************
*                  CONDITIONS DE BORDS                                 *
************************************************************************


                do ix = 1 , nx
                  uo( ix , 1 ) = un (ix, ny-1)
                  uo( ix , ny) = un (ix, 2 )
                  vo( ix , 1 ) = vn (ix, ny-1)
                  vo( ix , ny) = vn (ix, 2 )
                  ho( ix , 1 ) = hn (ix, ny-1)
                  ho( ix , ny) = hn (ix, 2 )
                enddo

                do iy = 1 , ny
                  uo( 1  , iy) = un ( nx-1 , iy)
                  uo( nx , iy) = un ( 2 , iy )
                  vo( 1  , iy) = vn ( nx-1 , iy)
                  vo( nx , iy) = vn ( 2 , iy)
                  ho( 1  , iy) = hn ( nx-1 , iy)
                  ho( nx , iy) = hn ( 2 , iy)
                enddo
```

Conditions initiales

```
************************************************************************
*                  CONDITIONS INITIALES                                *
************************************************************************



      do ix = 1, nx
      do iy = 1, ny

      x = (ix - (nx/2.))/nx
      y = (iy - (ny/2.))/ny

      uo( ix,iy) = exp (-(x**2 + y**2)*100.)
      vo( ix,iy) = 0
      ho( ix,iy) = 0

      un(ix,iy)=uo(ix,iy)
      vn(ix,iy)=vo(ix,iy)
      hn(ix,iy)=ho(ix,iy)
      enddo
      enddo

     CALL OUTSUBU(0,expnum)
     CALL OUTSUBV(0,expnum)
     CALL OUTSUBH(0,expnum)
```

Termes diffusifs

```
 ************************************************************************
 ************************************************************************
 *                              U                                       *
 ************************************************************************
 ************************************************************************

           un( ix , iy )= uo( ix , iy ) + dt*(

 ************************************************************************
 *                    TERMES DIFFUSSIFS                                 *
 ************************************************************************

&       +nu/(dx**2)*(uo( ix+1,iy)
&       -2*uo(ix , iy)+ uo( ix-1 , iy))+ nu/(dy**2)*(uo( ix , iy+1)
&       -2*uo( ix , iy)+ uo(ix , iy-1))
```

Equation de shallow water complètes

La hauteur d'eau H

```
 ************************************************************************
 ************************************************************************
 *                              H                                       *
 ************************************************************************
 ************************************************************************

        hn( ix , iy )= ho( ix , iy ) + dt*(

 ************************************************************************
 *                    TERMES ADVECTIFS                                  *
 ************************************************************************

&   -((uo(ix+1,iy)*ho(ix+1,iy)) -(uo(ix-1,iy)*ho(ix-1,iy))) /( 2*dx)
&   -((vo(ix,iy+1)*ho(ix,iy+1)) -(vo(ix,iy-1)*ho(ix,iy-1))) /( 2*dy)

&       )
```

Les champs de vitesses horizontaux u et v

```
      ********************************************************************
      ********************************************************************
      *                              U                                   *
      ********************************************************************
      ********************************************************************

                un( ix , iy )= uo( ix , iy ) + dt*(

      ********************************************************************
      *                    TERMES ADVECTIFS                              *
      ********************************************************************

     &       -uo(ix,iy)*( uo(ix+1,iy) - uo(ix-1,iy)) /( 2*dx)
     &       -vo(ix,iy)*( uo(ix,iy+1) - uo(ix,iy-1)) /( 2*dy)

      ********************************************************************
      *                    TERMES DIFFUSSIFS                             *
      ********************************************************************

     &       +nu/(dx**2)*(uo( ix+1,iy)
     &       -2*uo(ix , iy)+ uo( ix-1 , iy))+ nu/(dy**2)*(uo( ix , iy+1)
     &       -2*uo( ix , iy)+ uo(ix , iy-1))

      ********************************************************************
      *                    TERMES LIES GRADIENT DE PRESSION              *
      ********************************************************************

     &       -g*(ho(ix+1,iy)-ho(ix-1,iy))/(2*dx ))




      ********************************************************************
      ********************************************************************
      *                              V                                   *
      ********************************************************************
      ********************************************************************

             vn( ix , iy )= vo( ix , iy ) + dt*(

      ********************************************************************
      *                    TERMES ADVECTIFS                              *
      ********************************************************************

     &       -uo(ix,iy)*( vo(ix+1,iy) - vo(ix-1,iy)) /( 2*dx)
     &       -vo(ix,iy)*( vo(ix,iy+1) - vo(ix,iy-1)) /( 2*dy)

      ********************************************************************
      *                    TERMES DIFFUSSIFS                             *
      ********************************************************************

     &       + nu/(dx**2)*(vo( ix+1,iy)
     &       -2*vo(ix , iy)+ vo( ix-1 , iy))+ nu/(dy**2)*(vo( ix , iy+1)
     &       -2*vo( ix , iy)+ vo(ix , iy-1))

      ********************************************************************
      *                    TERMES LIES GRADIENT DE PRESSION              *
      ********************************************************************

     &        -g*(ho(ix,iy+1)-ho(ix,iy-1))/(2*dy ))
```

Plan Palm Deira

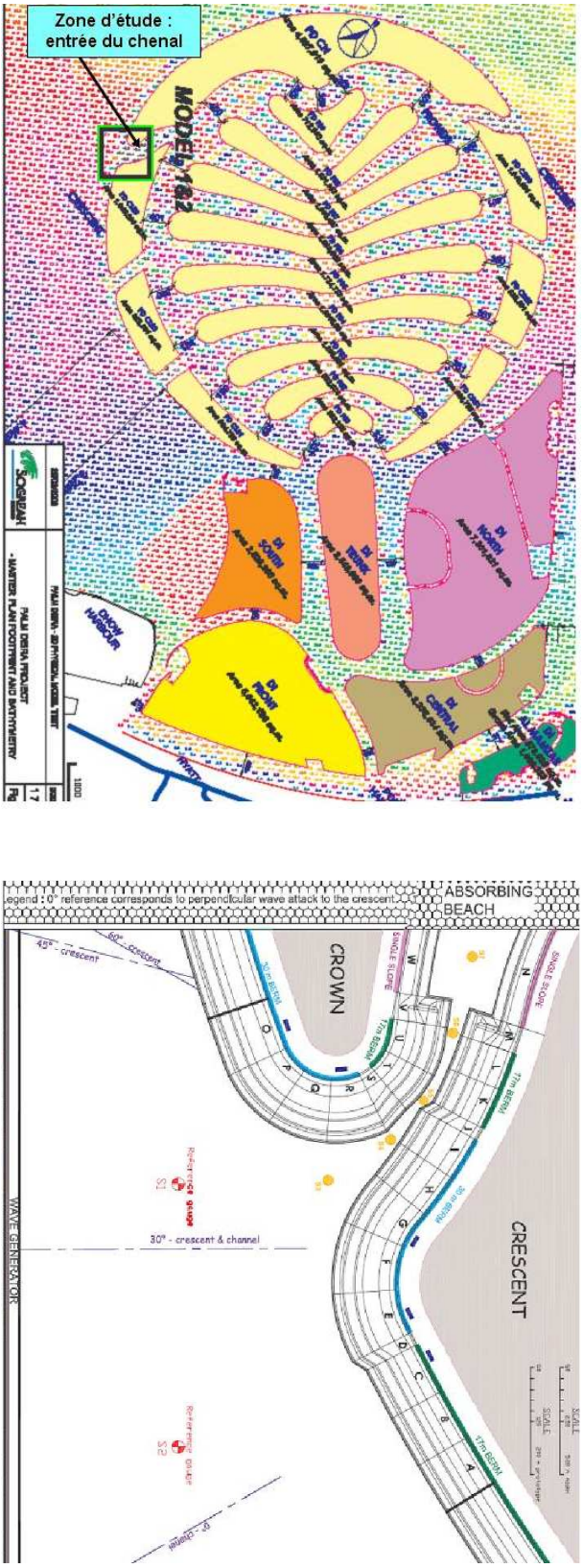


FIG. 12 – plan du bassin

Exemple programmation île

```
*********************************************
                CROWN
*********************************************
              do ix = 82,135
              do iy = 1,4
              un(ix,iy)=0
              vn(ix,iy)=0
              enddo
              enddo


              do ix = 83,134
              do iy = 5,11
              un(ix,iy)=0
              vn(ix,iy)=0
              enddo
              enddo

              do ix = 84,133
              do iy = 12,17
              un(ix,iy)=0
              vn(ix,iy)=0
              enddo
              enddo

               do ix = 85,133
              do iy = 18,21
              un(ix,iy)=0
              vn(ix,iy)=0
              enddo
              enddo

               do ix = 86,132
              do iy = 22,25
              un(ix,iy)=0
              vn(ix,iy)=0
              enddo
              enddo

               do ix = 87,131
              do iy = 26,28
              un(ix,iy)=0
              vn(ix,iy)=0
              enddo
              enddo

               do ix = 88,131
              do iy = 29,31
              un(ix,iy)=0
              vn(ix,iy)=0
              enddo
              enddo
```

Programme complet

```
c     ****************************************************************
c     *                                                              *
                        PROGRAMME PRINCIPAL
c     *                                                              *
c     ****************************************************************

      implicit none
c     ****************************************************************
c     *                    DECLARATION                               *
c     ****************************************************************
      integer ::   it1, it2 , ix , iy

      integer,parameter :: nx=200 ,ny=200 ,nt1=150,nt2=400,expnum=1
      real,parameter ::    lx = 30., ly = 30.
      real,parameter ::    dt =0.001 ,g=9.81, nu=5.E-2, h0=0.5,dh=0.2
      real,parameter ::    L=5.E-3 , F=0.5, x0=-0.35 , T=5.

      real(8) ::  uo(nx , ny) , un(nx , ny), vo(nx , ny) , vn(nx , ny )
      real(8) ::  ho(nx , ny) , hn(nx , ny)
      real    ::  x , y, dx , dy

      COMMON / newcom  /  un,vn,hn

c     ****************************************************************
c     *                    SENSIBILITE                               *
c     ****************************************************************


           dx = lx /nx
           dy = ly / ny


c     ****************************************************************
c     *                   CONDITIONS INITIALES                       *
c     ****************************************************************




           do ix = 1, nx
           do iy = 1, ny

           x = (ix - (nx/2.))/nx
           y = (iy - (ny/2.))/ny

           uo( ix,iy) = 0.
           vo( ix,iy) = 0.
```

```
            ho( ix,iy) = h0

            un(ix,iy)=uo(ix,iy)
            vn(ix,iy)=vo(ix,iy)
            hn(ix,iy)=ho(ix,iy)

            enddo
            enddo



c       ----On enregistre les conditions initiales dans un fichier------


           CALL OUTSUBU(0,expnum)
           CALL OUTSUBV(0,expnum)
           CALL OUTSUBH(0,expnum,h0)







c      ***********************************************************************
c      *                      RESOLUTION EQUATION SW  2D                     *
c      ***********************************************************************



            do it2 = 1 , nt2



c      ------------ On calcul le champ et la hauteur d'eau dans la boucle1-------

           do it1 = 1 , nt1


            do ix = 2, nx-1
            do iy = 2, ny-1

            x = (ix - (nx/2.))/nx

c      ***********************************************************************
c      ***********************************************************************
c      *                             U                                       *
c      ***********************************************************************
c      ***********************************************************************
```

```
                un( ix , iy )= uo( ix , iy ) + dt*(

c     ****************************************************************
c     *                   TERMES ADVECTIFS                           *
c     ****************************************************************

     &       -uo(ix,iy)*( uo(ix+1,iy) - uo(ix-1,iy)) /( 2*dx)
     &       -vo(ix,iy)*( uo(ix,iy+1) - uo(ix,iy-1)) /( 2*dy)


c     ****************************************************************
c     *                   TERMES DIFFUSSIFS                          *
c     ****************************************************************

     &       +nu/(dx**2)*(uo( ix+1,iy)
     &       -2*uo(ix , iy)+ uo( ix-1 , iy))+ nu/(dy**2)*(uo( ix , iy+1)
     &       -2*uo( ix , iy)+ uo(ix , iy-1))


c     ****************************************************************
c     *                   TERMES LIES GRADIENT DE PRESSION           *
c     ****************************************************************

     &       -g*(ho(ix+1,iy)-ho(ix-1,iy))/(2*dx )


c     ****************************************************************
c     *                   FORCAGE                                    *
c     ****************************************************************

     & +F*exp(-(x-x0)**2/L)*cos(3.14159265*2*((it1+(it2-1)*nt1)*dt)/T)


     &   )



c     ****************************************************************
c     ****************************************************************
c     *                           V                                  *
c     ****************************************************************
c     ****************************************************************

                vn( ix , iy )= vo( ix , iy ) + dt*(

c     ****************************************************************
```

```
c     *                    TERMES ADVECTIFS                                 *
c     *****************************************************************

     &       -uo(ix,iy)*( vo(ix+1,iy) - vo(ix-1,iy)) /( 2*dx)
     &       -vo(ix,iy)*( vo(ix,iy+1) - vo(ix,iy-1)) /( 2*dy)


c     *****************************************************************
c     *                    TERMES DIFFUSSIFS                              *
c     *****************************************************************

     &       + nu/(dx**2)*(vo( ix+1,iy)
     &       -2*vo(ix , iy)+ vo( ix-1 , iy))+ nu/(dy**2)*(vo( ix , iy+1)
     &       -2*vo( ix , iy)+ vo(ix , iy-1))


c     *****************************************************************
c     *                    TERMES LIES GRADIENT DE PRESSION               *
c     *****************************************************************

     &        -g*(ho(ix,iy+1)-ho(ix,iy-1))/(2*dy ))




c     *****************************************************************
c     *****************************************************************
c     *                           H                                       *
c     *****************************************************************
c     *****************************************************************

            hn( ix , iy )= ho( ix , iy ) + dt*(


c     *****************************************************************
c     *                    TERMES ADVECTIFS                               *
c     *****************************************************************

     &  -((uo(ix+1,iy)*ho(ix+1,iy)) -(uo(ix-1,iy)*ho(ix-1,iy))) /( 2*dx)
     &  -((vo(ix,iy+1)*ho(ix,iy+1)) -(vo(ix,iy-1)*ho(ix,iy-1))) /( 2*dy)

     &      )



           enddo
           enddo
```

```
c     ******************************************************************
c     *                    ILES                                        *
c     ******************************************************************

c     ******************************************************************
c     *        CROWN       On programme les iles en mettant a 0 u et v *
c     ******************************************************************
                    do ix = 82,135
                    do iy = 1,4
                    un(ix,iy)=0
                    vn(ix,iy)=0
                    enddo
                    enddo


                    do ix = 83,134
                    do iy = 5,11
                    un(ix,iy)=0
                    vn(ix,iy)=0
                    enddo
                    enddo

                    do ix = 84,133
                    do iy = 12,17
                    un(ix,iy)=0
                    vn(ix,iy)=0
                    enddo
                    enddo

                     do ix = 85,133
                    do iy = 18,21
                    un(ix,iy)=0
                    vn(ix,iy)=0
                    enddo
                    enddo

                     do ix = 86,132
                    do iy = 22,25
                    un(ix,iy)=0
                    vn(ix,iy)=0
                    enddo
                    enddo

                     do ix = 87,131
                    do iy = 26,28
                    un(ix,iy)=0
                    vn(ix,iy)=0
```

```
enddo
enddo

 do ix = 88,131
do iy = 29,31
un(ix,iy)=0
vn(ix,iy)=0
enddo
enddo

 do ix = 89,130
do iy = 31,32
un(ix,iy)=0
vn(ix,iy)=0
enddo
enddo

 do ix = 90,130
do iy = 33,36
un(ix,iy)=0
vn(ix,iy)=0
enddo
enddo

 do ix = 91,129
do iy = 37,39
un(ix,iy)=0
vn(ix,iy)=0
enddo
enddo

 do ix = 92,129
do iy = 40,41
un(ix,iy)=0
vn(ix,iy)=0
enddo
enddo



 do ix = 93,128
do iy = 42,43
un(ix,iy)=0
vn(ix,iy)=0
enddo
enddo

 do ix = 94,127
do iy = 44,45
```

```
un(ix,iy)=0
vn(ix,iy)=0
enddo
enddo

 do ix = 95,126

un(ix,45)=0
vn(ix,45)=0

enddo

 do ix = 96,125

un(ix,46)=0
vn(ix,46)=0

enddo

  do ix = 97,124

un(ix,47)=0
vn(ix,47)=0

enddo

  do ix = 98,123

un(ix,48)=0
vn(ix,48)=0

enddo

  do ix = 99,122

un(ix,49)=0
vn(ix,49)=0

enddo

  do ix = 100,120

un(ix,50)=0
vn(ix,50)=0

enddo

  do ix = 104,118
```

```
                    un(ix,51)=0
                    vn(ix,51)=0

                    enddo
c     ****************************************************************
c     *                      CRESCENT                                *
c     ****************************************************************

                    do ix = 185,200
                    do iy = 0,200
                    un(ix,iy)=0
                    vn(ix,iy)=0
                    enddo
                    enddo

                    do iy = 7,200
                    un(184,iy)=0
                    vn(184,iy)=0
                    enddo


                    do iy = 11,200
                    un(183,iy)=0
                    vn(183,iy)=0
                    enddo


                    do iy = 16,200
                    un(182,iy)=0
                    vn(182,iy)=0
                    enddo


                    do iy = 20,200
                    un(181,iy)=0
                    vn(181,iy)=0
                    enddo

                    do iy = 25,200
                    un(180,iy)=0
                    vn(180,iy)=0
                    enddo


                    do iy = 29,200
                    un(179,iy)=0
                    vn(179,iy)=0
                    enddo
```

```
do iy = 33,200
un(178,iy)=0
vn(178,iy)=0
enddo

do iy = 36,200
un(177,iy)=0
vn(177,iy)=0
enddo

do iy = 40,200
un(176,iy)=0
vn(176,iy)=0
enddo


do iy = 41,200
un(175,iy)=0
vn(175,iy)=0
enddo

do iy = 45,200
un(174,iy)=0
vn(174,iy)=0
enddo

do iy = 50,200
un(173,iy)=0
vn(173,iy)=0
enddo

do iy = 56,200
un(172,iy)=0
vn(172,iy)=0
enddo

do iy = 58,200
un(171,iy)=0
vn(171,iy)=0
enddo

do iy = 60,200
un(170,iy)=0
vn(170,iy)=0
enddo

do iy = 60,200
un(169,iy)=0
```

```
 vn(169,iy)=0
 enddo

do iy = 61,198
 un(168,iy)=0
 vn(168,iy)=0
 enddo

 do iy = 64,196
 un(167,iy)=0
 vn(167,iy)=0
 enddo

 do iy = 66,194
 un(166,iy)=0
 vn(166,iy)=0
 enddo

 do iy = 68,192
 un(165,iy)=0
 vn(165,iy)=0
 enddo

 do iy = 70,190
 un(164,iy)=0
 vn(164,iy)=0
 enddo

 do iy = 72,188
 un(163,iy)=0
 vn(163,iy)=0
 enddo

 do iy = 74,186
 un(162,iy)=0
 vn(162,iy)=0
 enddo

 do iy = 76,184
 un(161,iy)=0
 vn(161,iy)=0
 enddo

 do iy = 78,182
 un(160,iy)=0
 vn(160,iy)=0
 enddo

 do iy = 80,180
```

```
un(159,iy)=0
vn(159,iy)=0
enddo

do iy = 82,178
un(158,iy)=0
vn(158,iy)=0
enddo

do iy = 84,176
un(157,iy)=0
vn(157,iy)=0
enddo

do iy = 86,174
un(156,iy)=0
vn(156,iy)=0
enddo

do iy = 88,172
un(155,iy)=0
vn(155,iy)=0
enddo

do iy = 90,170
un(154,iy)=0
vn(154,iy)=0
enddo

do iy = 92,168
un(153,iy)=0
vn(153,iy)=0
enddo


do iy = 94,166
un(152,iy)=0
vn(152,iy)=0
enddo

do iy = 96,164
un(151,iy)=0
vn(151,iy)=0
enddo

do iy = 98,162
un(150,iy)=0
vn(150,iy)=0
enddo
```

```
do iy = 100,160
un(149,iy)=0
vn(149,iy)=0
enddo

do iy = 102,158
un(148,iy)=0
vn(148,iy)=0
enddo

do iy = 104,156
un(147,iy)=0
vn(147,iy)=0
enddo

do iy = 106,154
un(146,iy)=0
vn(146,iy)=0
enddo

do iy = 108,152
un(145,iy)=0
vn(145,iy)=0
enddo

do iy = 110,150
un(144,iy)=0
vn(144,iy)=0
enddo

do iy = 112,148
un(143,iy)=0
vn(143,iy)=0
enddo

do iy = 114,146
un(142,iy)=0
vn(142,iy)=0
enddo

do iy = 116,144
un(141,iy)=0
vn(141,iy)=0
enddo

do iy = 118,142
un(140,iy)=0
vn(140,iy)=0
```

```
                      enddo

                      do iy = 119,140
                      un(139,iy)=0
                      vn(139,iy)=0
                      enddo

                      do iy = 120,138
                      un(138,iy)=0
                      vn(138,iy)=0
                      enddo

                     do iy = 122,136
                      un(137,iy)=0
                      vn(137,iy)=0
                      enddo


                     do iy = 125,133
                      un(136,iy)=0
                      vn(136,iy)=0
                      enddo

c     ******************************************************************

          do ix = 2, nx-1
          do iy = 2, ny-1

             uo( ix , iy) = un (ix , iy )
             vo( ix , iy) = vn (ix , iy )
             ho( ix , iy) = hn (ix , iy )

          enddo
          enddo


c     ******************************************************************
c     *                  CONDITIONS DE BORDS                           *
c     ******************************************************************


c                      do ix = 1 , nx
c                        uo( ix , 1 ) = un (ix, ny-1)
c                        uo( ix , ny) = un (ix, 2 )
c                        vo( ix , 1 ) = vn (ix, ny-1)
c                        vo( ix , ny) = vn (ix, 2 )
c                        ho( ix , 1 ) = hn (ix, ny-1)
c                        ho( ix , ny) = hn (ix, 2 )
c                      enddo
```

```
c
c                     do iy = 1 , ny
c                       uo( 1  , iy) = un ( nx-1 , iy)
c                       uo( nx , iy) = un ( 2 , iy )
c                       vo( 1  , iy) = vn ( nx-1 , iy)
c                       vo( nx , iy) = vn ( 2 , iy)
c                       ho( 1  , iy) = hn ( nx-1 , iy)
c                       ho( nx , iy) = hn ( 2 , iy)
c                     enddo



       enddo ! fin de la boucle intermdiaire it1



c     ****************************************************************
c     *                 ECRITURE DONNES DANS FICHIER                 *
c     ****************************************************************
        write(6,*)it2
        CALL OUTSUBU(it2,expnum)
        CALL OUTSUBV(it2,expnum)
        CALL OUTSUBH(it2,expnum,h0)

      enddo ! fin de la boucle sur it2



      end

c     ****************************************************************
c     ****************************************************************
c     **              FIN DU PROGRAMME PRINCIPAL                    **
c     ****************************************************************
c     ****************************************************************



c     ****************************************************************
c     **                                                            **
c     **   subroutine    OUTSUB U                                   **
```

```
c     **                                                              **
c     ****************************************************************
      SUBROUTINE OUTSUBU(step,expnum)
      IMPLICIT NONE
      INTEGER :: nx,ny
      PARAMETER   (nx=200,ny=200)
      INTEGER ::  ix,iy
      REAL(4) ::  aa(nx,ny)
      INTEGER ::  step,expnum
      REAL(8) ::  un(nx,ny),vn(nx,ny), hn(nx,ny)
      COMMON / newcom  /  un,vn,hn
      CHARACTER(4) :: filename
      CHARACTER(6) :: expnumber
      CHARACTER(4) :: stepnumber
      CHARACTER(5) :: fileext
      CHARACTER(10) :: filenamel
      CHARACTER(11) :: filenamell
      CHARACTER(15) :: filenamelll
      CHARACTER(20) :: filenamef
c     ****************************************************************
c     **                                                              **
c     **  nameing of output file                                      **
c     **                                                              **
c     ****************************************************************
      filename="sw_u"
      fileext=".data"
      WRITE(expnumber,FMT='(I6)') 100000+ expnum
      WRITE(stepnumber,FMT='(I4)') 1000+ step
      filenamel = filename // expnumber
      filenamell = filenamel // "_"
      filenamelll = filenamell // stepnumber
      filenamef = filenamelll // fileext
      OPEN(UNIT=13,STATUS='NEW',FORM='FORMATTED',
     &FILE=filenamef)
      DO  ix=1,nx
      DO  iy=1,ny
      aa(ix,iy)=REAL(un(ix,iy),4)
      ENDDO
      ENDDO

      WRITE(13,*)  ((aa(ix,iy),iy=1,ny),ix=1,nx)


      CLOSE(13)


      RETURN
      END SUBROUTINE OUTSUBU
c     ****************************************************************
```

```
c     **                                                              **
c     **  end                                                         **
c     **                                                              **
c     ******************************************************************
c     ******************************************************************


c     ******************************************************************
c     **                                                              **
c     **   subroutine    OUTSUB V                                     **
c     **                                                              **
c     ******************************************************************
      SUBROUTINE OUTSUBV(step,expnum)
      IMPLICIT NONE
      INTEGER :: nx,ny
      PARAMETER   (nx=200,ny=200)
      INTEGER ::  ix,iy
      REAL(4) ::  aa(nx,ny)
      INTEGER ::  step,expnum
      REAL(8) ::  un(nx,ny),vn(nx,ny), hn(nx,ny)
      COMMON / newcom  /  un,vn,hn
      CHARACTER(4) :: filename
      CHARACTER(6) :: expnumber
      CHARACTER(4) :: stepnumber
      CHARACTER(5) :: fileext
      CHARACTER(10) :: filenamel
      CHARACTER(11) :: filenamell
      CHARACTER(15) :: filenamelll
      CHARACTER(20) :: filenamef
c     ******************************************************************
c     **                                                              **
c     **  nameing of output file                                      **
c     **                                                              **
c     ******************************************************************
      filename="sw_v"
      fileext=".data"
      WRITE(expnumber,FMT='(I6)') 100000+ expnum
      WRITE(stepnumber,FMT='(I4)') 1000+ step
      filenamel = filename // expnumber
      filenamell = filenamel // "_"
      filenamelll = filenamell // stepnumber
      filenamef = filenamelll // fileext
      OPEN(UNIT=13,STATUS='NEW',FORM='FORMATTED',
     &FILE=filenamef)
      DO  ix=1,nx
      DO  iy=1,ny
      aa(ix,iy)=REAL(vn(ix,iy),4)
      ENDDO
      ENDDO
```

```
      WRITE(13,*) ((aa(ix,iy),iy=1,ny),ix=1,nx)

      CLOSE(13)

      RETURN
      END SUBROUTINE OUTSUBV
c     ****************************************************************
c     **                                                            **
c     **  end                                                       **
c     **                                                            **
c     ****************************************************************
c     ****************************************************************



c     ****************************************************************
c     **                                                            **
c     **   subroutine    OUTSUB H                                   **
c     **                                                            **
c     ****************************************************************
      SUBROUTINE OUTSUBH(step,expnum,h0)
      IMPLICIT NONE
      INTEGER :: nx,ny
      PARAMETER   (nx=200,ny=200)
      INTEGER ::  ix,iy
      REAL(4) ::  aa(nx,ny),h0
      INTEGER ::  step,expnum
      REAL(8) ::  un(nx,ny),vn(nx,ny), hn(nx,ny)
      COMMON / newcom  /  un,vn,hn
      CHARACTER(4) :: filename
      CHARACTER(6) :: expnumber
      CHARACTER(4) :: stepnumber
      CHARACTER(5) :: fileext
      CHARACTER(10) :: filenamel
      CHARACTER(11) :: filenamell
      CHARACTER(15) :: filenamelll
      CHARACTER(20) :: filenamef
c     ****************************************************************
c     **                                                            **
c     **  nameing of output file                                    **
c     **                                                            **
c     ****************************************************************
      filename="sw_h"
      fileext=".data"
      WRITE(expnumber,FMT='(I6)') 100000+ expnum
      WRITE(stepnumber,FMT='(I4)') 1000+ step
      filenamel = filename // expnumber
```

```
      filenamell = filenamel // "_"
      filenamelll = filenamell // stepnumber
      filenamef = filenamelll // fileext
      OPEN(UNIT=13,STATUS='NEW',FORM='FORMATTED',
     &FILE=filenamef)
      DO  ix=1,nx
      DO  iy=1,ny
      aa(ix,iy)=(REAL(hn(ix,iy),4)-h0)
      ENDDO
      ENDDO


      WRITE(13,*)  ((aa(ix,iy),iy=3,ny-3),ix=3,nx-3)


      CLOSE(13)
      RETURN
      END SUBROUTINE OUTSUBH
c     ***************************************************************
c     **                                                           **
c     **  end                                                      **
c     **                                                           **
c     ***************************************************************
c     ***************************************************************
```