Université catholique de Louvain
LEPL1110 - Introduction aux éléments finis

Rapport sur le Tsunami d'Okada

Amaury Desmette 32361900
Nicolas Giourgas 38961900

Ordre de précision

On sait que pour une méthode d'ordre p, on a la relation suivante :

$$\frac{e_h}{e_{h/2}} = 2^p \tag{1}$$

Puisque nous utilisons une méthode d'ordre 2, si notre programme est correct on devrait avoir $p = \log_2(\frac{e_h}{e_{h/2}}) = 2$. Malheureusement, dans notre cas nous ne possédons pas de solution de référence, nous avons donc fait le choix de réaliser une extrapolation de Richardson pour estimer la solution exacte (voir Figure 1).

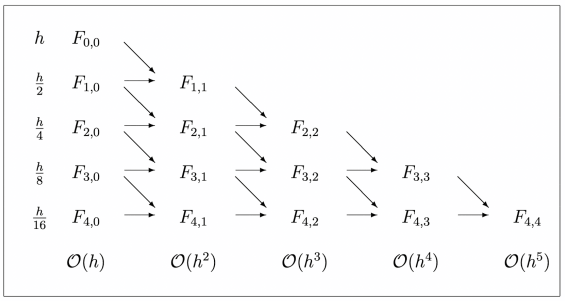

Figure 1 – Schéma extrapolation de Richardson

Nous avons pris comme solution de référence $F_{3,3}$. Nous avons testé 4 pas de temps différents allant de 8 à 1. Pour que nos tests ne soient pas trop long, nous les avons fait sur le maillage "*Pacific Medium*". Le programme nous donnait les valeurs de $F_{0,0}$, $F_{1,0}$, $F_{2,0}$ et $F_{3,0}$. Ensuite, l'équation 2 nous a permis d'extrapoler toutes les autres valeurs jusqu'à $F_{3,3}$.

$$F_{i,j} = \frac{\left(2^j F_{i,j-1} - F_{i-1,j-1}\right)}{(2^j - 1)} \tag{2}$$

Pour calculer l'ordre expérimental de notre programme nous avons utilisé la formule 3. Nous obtenons ainsi le tableau 1 où E, U et V correspondent respectivement à l'élévation et aux deux composantes de la vitesse horizontale.

$$p_{F,i} = \log_2\left(\frac{F_{i,0} - F_{33}}{F_{i+1,0} - F_{33}}\right) \quad \forall i = 0, 1, 2 \tag{3}$$

i	p_E	p_U	p_V
0	2.0001	2.0003	2.0005
1	2.0002	2.0007	2.0013
2	1.9998	2.0014	2.0021

TABLE 1 – Ordre expérimental de E, U, V pour différents dt

Au vu des résultats obtenus, nous pouvons conclure que notre programme est aussi précis que ce qu'il devrait être.

Analyse de la solution

Afin de vérifier si la solution de notre programme est réaliste, nous avons cherché des données sur le vrai tsunami d'Okada. Nous avons ainsi trouvé une vidéo [1] produite par le *"National Oceanic and Atmospheric Administration"* des Etats-Unis.

Le temps auquel la simulation affiche le tsunami est donné par le nom du fichier en cours de lecture. Il suffit de multiplier le pas de temps utilisé pour la simulation par l'itération à laquelle le fichier est enregistré : $t = dt \times n[s]$.

Nous allons comparer les temps réels et simulés auxquels le front atteint différents lieux géographiques.

Dans un premier temps comparons l'arrivée du front du tsunami aux côtes de l'île japonaise d'Hokkaïdo.

Le temps réel est de 36 minutes et le temps calculé pour la simulation est de 38 minutes. Bien que cet écart soit de presque 6%, il est possible que les heures de départ des deux animations ne soit pas la même, ou que l'arrêt sur image ne représente pas exactement le même instant.

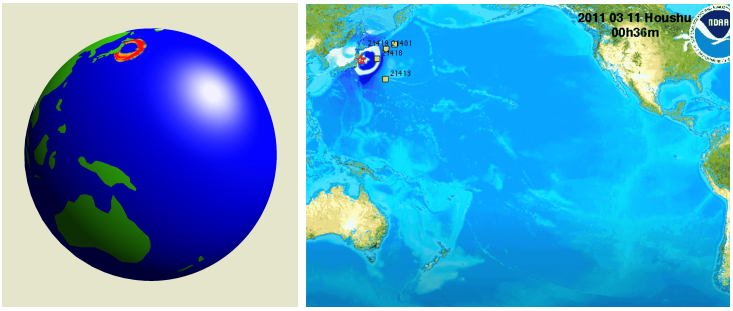

FIGURE 2 – Arrivée du Tsunami à Hokkaïdo

Ensuite comparons l'arrivée du tsunami à Taïwan.

Dans ce cas-ci, le temps réel est de 3h50 et le temps simulé est de 3h53.

1. `https://youtu.be/Lo5uH1UJF4A`

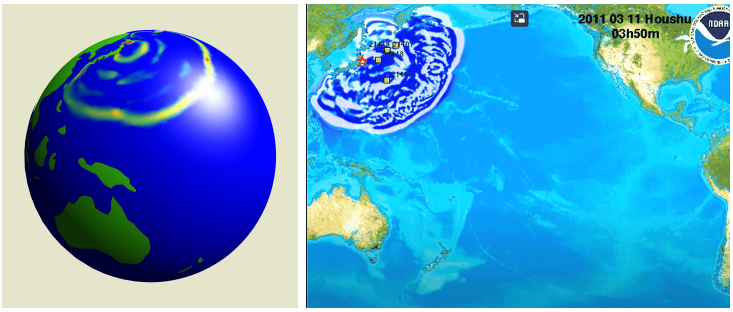

FIGURE 3 – Arrivée du Tsunami à Taiwan

pour finir comparons l'arrivée de la vague en Indonésie.

Le temps réel est de 5h58 et le temps de la simulation est de 6h03.

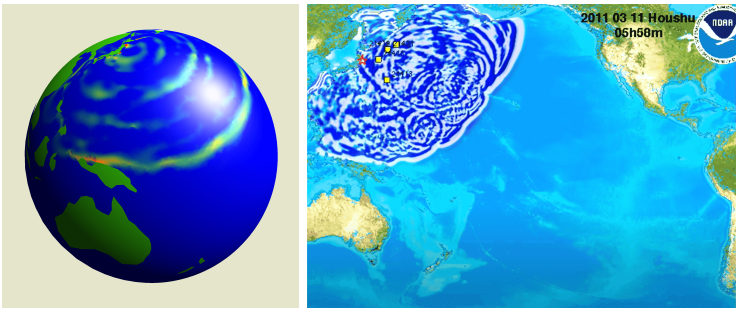

FIGURE 4 – Arrivée du Tsunami en Indonésie

Nous pouvons donc conclure que notre solution est fidèle à la réalité. L'écart entre la simulation et le tsunami historique ne dépasse pas les 5 minutes. Ces différences peuvent aussi être expliquées par une différence des instants représentés dans les deux captures de chaque cas.

Optimisation

Méthode d'intégration

Il nous était imposé d'utiliser comme méthode d'intégration temporelle Runge-Kutta d'ordre 2. Utiliser la méthode d'Euler explicite à la place demandait moins de calcul et moins de mémoire mais était moins précise. Le tableau ci-dessous illustre la différence de temps de calcul pour les deux méthodes pour différents paramètres avec le maillage "*Pacific Fine*".

Méthode Runge-Kutta d'ordre 2 :

$$\begin{aligned} U_{i+1} &= U_i + \frac{dt}{2} \times (K_1 + K_2) \\ K_1 &= f(U_i) \\ K_2 &= f(\underbrace{U_i + hK_1}_{P_{i+1}}) \end{aligned} \tag{4}$$

Methode Euler explicite :

$$U_{i+1} = U_i + dt \times f(U_i) \tag{5}$$

	Temps [s]	dt[s]	Nombre d'itérations	Nombre d'itérations écrites
Runge-Kutta d'ordre 2	191,8	4	10000	200
Euler explicite	95,8	4	10000	200

TABLE 2 – Rapidité du code

Nous nous rendons bien compte que Runge-Kutta d'ordre 2 est deux fois moins rapide que Euler explicite. Cela semble logique puisque cette méthode nécessite deux fois plus de calcul lors de l'intégration temporelle (voir Eq. 4). Cependant Euler explicite est moins précis car c'est une méthode d'ordre 1. Dans un soucis d'efficacité et de mémoire, on peut se dire que avec un pas de temps assez petit, pour un grand nombre d'itération et avec avec un grand maillage, utiliser Euler explicite pourrait donner une solution tout à fait acceptable.

Code

Pour Optimiser notre programme nous avons eu recours à quelques optimisations :

- Utilisation des variables globales au lieu de structures.
- Utilisation de la fonction *calloc* au lieu de *malloc*. Pour éviter une boucle supplémentaire lors de l'initialisation des valeurs à zéro.
- Utilisation de macros au lieu d'une fonction pour l'interpolation de la vitesse et de l'élévation.
- Nous avons sortis certaines opérations des boucles pour éviter de les calculer plusieurs fois.

Idées

Puisque notre vague se propage à la vitesse $\sqrt{gh}$, nous avions pensé qu'il serait intéressant de ne calculer que les éléments situés à une distance $\sqrt{gh} \times t$ de la position initiale. Cette solution nous permettrais d'éviter de faire des calculs inutiles. Malheureusement, nous avons manqué de temps pour la réaliser. De plus, notre programme aurait perdu en compacité.