Matmeca 1ère année - TP de programmation en Fortran
année 2022-2023

TP 5: Propagation d'ondes acoustiques et élastiques

Le but de ce TP est de calculer la propagation d'ondes élastiques dans des solides. Un problème similaire est le cas de la propagation d'ondes acoustiques dans des fluides. La perturbation de pression u satisfait une équation d'ondes (dans l'hypothèse de petites perturbations) :

$$\begin{cases} \dfrac{1}{c^2}\dfrac{\partial^2 u}{\partial t^2} - \Delta u = f(x,t) \ \text{ dans } \Omega \\ u(x,t) = 0 \text{ sur } \partial\Omega \\ u(x,0) = \dfrac{\partial u}{\partial t}(x,0) = 0 \end{cases} \tag{1}$$

où c est la vitesse du son, f la source sonore, Ω le domaine de calcul.

Points de discrétisation

On choisit de résoudre l'équation des ondes sur un rectangle $[x_0, x_N] \times [y_0, y_N]$ avec des points de discrétisation réguliers :

$$x_i = x_0 + (i-1)\Delta x,\ i = 1..N_x, \quad y_j = y_0 + (j-1)\Delta y,\ j = 1..N_y$$

On note N_x et N_y le nombre de points selon chaque direction, les pas d'espaces valent alors :

$$\Delta x = \frac{x_N - x_0}{N_x - 1}, \quad \Delta y = \frac{y_N - y_0}{N_y - 1}$$

Stocker les points de discrétisation x_i et y_j dans deux vecteurs. On prendra les valeurs suivantes :

$$x_0 = y_0 = -10,\ x_N = y_N = 10,\ N_x = 3,\ N_y = 2$$

Afficher ces deux vecteurs de taille Nx et Ny.

Ecriture de fichiers binaires

L'inconnue u est calculée sur les points de discrétisation :

$$u_{i,j}(t) \approx u(x_i, y_j, t)$$

On stockera U sous la forme d'une matrice de taille $N_x \times N_y$. Allouer cette matrice et la remplir aléatoirement (avec la fonction rand()). On vous fournit une sous-routine **WriteBinary** pour écrire la matrice dans un fichier binaire. Tapez dans un terminal (dans le répertoire de travail) :

```
wget http://www.math.u-bordeaux1.fr/~durufle/output_mat.f90
```

On pourra inclure cette sous-routine dans le module de son choix. Si on fait des calculs en simple précision, il faut changer la précision comme suit :

```
integer, parameter :: wp = kind(1.0)
```

La sous-routine prend en argument les points x, y, la matrice U et le nom du fichier. Vous appelerez cette fonction dans le programme principal :

```
call WriteBinary(x, y, u, "Un.dat")
```

Pour visualiser la matrice, on vous propose de télécharger visu2D.py à partir du terminal :

```
wget http://www.math.u-bordeaux1.fr/~durufle/visu2D.py
```

Vous pouvez ensuite charger la matrice avec ipython (à taper dans un terminal) :

```
ipython --pylab
```

Cette commande ouvre une console ipython. Vous importez le package visu2D.py (cet import doit être réalisé une seule fois) :

```
from visu2D import *
```

Pour charger une matrice écrite en binaire avec le format demandé :

```
X, Y, U = load2D("Un.dat")
U
```

La deuxième commande doit afficher la matrice. Vérifier que les valeurs correspondent à ce que vous avez en Fortran.

Calcul d'une source gaussienne

Pour la source, on choisit une gaussienne en espace qu'on stocke aussi sous forme d'une matrice :

$$F_{i,j} = \exp(-7(x_i^2 + y_j^2))$$

Stocker la matrice F et l'écrire dans un fichier binaire. Pour cette question on choisira

$$N_x = 201,\ N_y = 201$$

On ne calculera F que pour les points intérieurs :

$$i = 2..N_x - 1,\ j = 2..N_y - 1$$

de telle sorte que F soit nulle sur les bords du carré. Sous ipython, charger la matrice, et l'afficher avec plot2dinst :

```
X, Y, F = load2D("source.dat")
plot2dinst(X, Y, F)
```

Calcul de la source temporelle

On prend une dérivée seconde de gaussienne (qu'on appelle Ricker) comme source temporelle :

$$h(t) = f_0\,(2\pi^2(f_0\,t - 1)^2 - 1)e^{-\pi^2(f_0 t - 1)^2}$$

Implémenter une fonction qui renvoie $h(t)$ et qui prend en argument f_0 la fréquence centrale du Ricker, et t le temps. On choisit les paramètres suivants:

$$f_0 = 440\text{Hz}, \quad t_f = 0.1s, \quad \Delta t = 10^{-4}s$$

Δt est le pas de temps, t_f le temps final. Écrire une boucle en temps, où vous évaluez h à chaque pas de temps. Vous pouvez écrire les valeurs dans un fichier texte (form='formatted'), on n'utilisera pas de fichier binaire ici. Sous ipython, vous pouvez afficher le résultat :

```
H = loadtxt('H.dat')
plot(H)
```

Vous devez voir une dérivée seconde de gaussienne.

Calcul du laplacien

On utilise un schéma aux différences finies centré d'ordre 2 pour évaluer le laplacien :

$$(\Delta u)_{i,j} = G(U)_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}$$

Implémenter une sous-routine pour calculer ce laplacien (qui est aussi une matrice de même taille que U). Du fait de la condition de Dirichlet, on considère que u est nulle sur le bord, on ne calcule donc le laplacien que pour les points intérieurs :

$$i = 2..N_x - 1,\ j = 2..N_y - 1$$

Schéma en temps

On a noté G l'opérateur qui associe à la matrice U son laplacien $G(U)$. En utilisant la méthode des lignes, on doit alors résoudre le système semi-discret suivant :

$$\frac{1}{c^2}\frac{d^2U}{dt^2} = G(U) + h(t)\,F$$

qu'on discrétise ensuite avec un schéma centré d'ordre deux

$$\frac{U^{n+1} - 2U^n + U^{n-1}}{c^2\Delta t^2} = G(U^n) + h(t^n)F$$

Une analyse de stabilité au sens de Von Neumann montre que ce schéma temporel est stable sous la condition CFL suivante :

$$\Delta t \leq \frac{\min(\Delta x, \Delta y)}{c\sqrt{2}}$$

Implémenter ce schéma. On pourra modifier U pour tout i et j (bords compris). Comme F et G(U) sont des matrices nulles sur les bords, la solution devrait rester nulle sur les bords, ce qui correspond à la condition de Dirichlet. On prendra c=340 m/s. On écrira la solution à intervalles réguliers comme suit :

```
integer :: Ndisplay
character(len=6) entier

Ndisplay = 10
do nt=1, nb_iterations
  if (mod(nt, Ndisplay) == 0) then
    write(entier, '(i6)') nt / Ndisplay
    call WriteBinary(x, y, U, "Un" // trim(adjustl(entier)) // ".dat")
  end if
end do
```

Vous devriez avoir des fichiers Un0.dat, Un1.dat, etc. Pour visualiser la propagation de l'onde sous ipython, vous pouvez écrire :

```
film2D("Un", ".dat", 1, 101, -1, 1)
```

Cette commande affiche le champ de pression à chaque instant sauvegardé dans les fichiers de Un1.dat jusqu'à Un100.dat. Les deux derniers arguments sont l'échelle de couleur, que vous pouvez modifier si vous voulez saturer les couleurs. film2D a deux arguments optionnels, le premier est le ratio de raffinement, le second est le temps d'attente entre l'affichage de deux frames. Par défaut, ces arguments valent 1 et 0. Si vous écrivez :

```
film2D("Un", ".dat", 1, 101, -1, 1, 2, 0.05)
```

la solution sera interpolée sur une grille deux fois plus fine (pour avoir un rendu plus joli) et la console attendra 0.05s entre chaque fichier.

Convergence spatio-temporelle

Écrire la solution au temps final. Vous pouvez comparer deux solutions avec un pas de temps et d'espace divisé par deux en tapant sous python :

```
[X, Y, V] = load2D("UnFinal.dat", 2);
[X, Y, Vraff] = load2D("UnFinalRaff.dat")
norm(V-Vraff, inf)/norm(V, inf)
```

Le schéma étant d'orde deux en temps et en espace, vous devez observer que

$$||u_{\Delta t,\Delta x} - u_{\Delta t/2,\Delta x/2}|| \approx 4||u_{\Delta t/2,\Delta x/2} - u_{\Delta t/4,\Delta x/4}||$$

voir l'analyse correspondante donnée à la fin du TP2.

Elastodynamique

On veut maintenant résoudre les équations de l'élastodynamique :

$$\begin{cases} \rho\dfrac{\partial^2 u_x}{\partial t^2} - \dfrac{\partial}{\partial x}\left[(\lambda+2\mu)\dfrac{\partial u_x}{\partial x} + \lambda\dfrac{\partial u_y}{\partial y}\right] - \dfrac{\partial}{\partial y}\left[\mu\dfrac{\partial u_x}{\partial y} + \mu\dfrac{\partial u_y}{\partial x}\right] = f(x,t) \\ \rho\dfrac{\partial^2 u_y}{\partial t^2} - \dfrac{\partial}{\partial x}\left[\mu\dfrac{\partial u_x}{\partial y} + \mu\dfrac{\partial u_y}{\partial x}\right] - \dfrac{\partial}{\partial y}\left[(\lambda+2\mu)\dfrac{\partial u_y}{\partial y} + \lambda\dfrac{\partial u_x}{\partial x}\right] = 0 \\ u_x(x,t) = u_y(x,t) = 0 \text{ sur } \partial\Omega \quad \text{(CL d'encastrement)} \\ u_x(x,0) = u_y(x,0) = \dfrac{\partial u_x}{\partial t}(x,0) = \dfrac{\partial u_y}{\partial t}(x,0) = 0 \quad \text{(CI nulles)} \end{cases}$$

où u_x, u_y sont les déplacements du solide, ρ la masse volumique, λ, μ les coefficients de Lamé. Pour la source f, on prendra une gaussienne comme précédemment. Physiquement, si on s'intéresse par exemple à des ondes sismiques, le centre de la gaussienne correspond à l'hypocentre du séisme. Proposer un schéma aux différences finies sur le même modèle que pour l'équation des ondes. Pour les dérivées croisées, on pourra utiliser l'approximation :

$$\frac{\partial^2 u}{\partial x \partial y} \approx \frac{u_{i+1,j+1} - u_{i-1,j+1} - u_{i+1,j-1} + u_{i-1,j-1}}{4\Delta x \Delta y}$$

On prendra comme paramètres ceux de l'aluminium

$$\rho = 2700,\ E = 69e9,\ \nu = 0.32,\ \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)},\ \mu = \frac{E}{2(1+\nu)},\ f_0 = 4400,\ t_f = 0.01,\ \Delta t = 10^{-5}$$

Observer les ondes P qui se propagent à une vitesse de $\sqrt{\frac{\lambda+2\mu}{\rho}}$ et les ondes S qui se propagent à une vitesse de $\sqrt{\frac{\mu}{\rho}}$.