Thème info 4 – Exemples de résolution numérique d'une équation aux dérivées partielles

Il s'agit ici d'étudier quelques exemples — issus des sciences physiques — de résolution approchée d'une *équation aux dérivées partielles*, assortie de conditions initiales et de conditions aux limites. L'existence et l'unicité d'une solution dans un tel cadre est en général un problème mathématique délicat... C'est pourquoi on l'admettra, en s'appuyant sur l'interprétation physique.
Mais on verra que la question de la stabilité du schéma numérique employé est sensible...

I - Équation de la chaleur en dimension 1

1) Position du problème

Pour cet exemple de diffusion en dimension 1, on considère deux barres d'un même métal (considérées comme filiformes) de longueurs L_1 et L_2, initialement portées à des températures T_1 et T_2 et mises bout à bout, formant ainsi une unique barre de longueur $L = L_1 + L_2$.
On s'intéresse à l'évolution de la température du point d'abscisse $x \in [0, L]$ à l'instant t, notée $T(x,t)$, cela en situation de *régime forcé*, où les extrémités de ladite barre restent maintenues aux températures T_1 (pour $x = 0$) et T_2 (pour $x = L$).

La fonction de deux variables T vérifie l'*équation de la chaleur*

$$\frac{\partial T}{\partial t} = D\frac{\partial^2 T}{\partial x^2} \quad \text{où la constante } D \text{ est le } \textit{coefficient de diffusion} \text{ du métal.}$$

Et dans la situation qui nous intéresse, nous avons

- les *conditions initiales* :

$$\forall x \in [0, L_1[\quad T(x,0) = T_1 \quad \text{et} \quad \forall x \in]L_1, L] \quad T(x,0) = T_2 \ ;$$

- les *conditions aux limites* :

$$\forall t \geq 0 \quad T(0,t) = T_1 \quad \text{et} \quad T(L,t) = T_2.$$

2) Discrétisation

Dans le même esprit que celui de la méthode d'Euler pour les équations différentielles ordinaires, on considère une subdivision de $[0, L]$ en N intervalles d'amplitude $h = L/N$ (*pas de discrétisation*).
Pour le temps, puisqu'on ne pourra faire qu'une quantité finie de calculs (!), on fixe un pas de discrétisation k, un entier P et l'on se limite à $t \in [0, Pk]$.

On est alors amené à remplir un tableau $\widetilde{T}$ de type $(N+1, P+1)$, tel que $\widetilde{T}[n,i]$ contienne une valeur approchée de $T(nh, ik)$, pour $(n,i) \in [\![0, N]\!] \times [\![0, P]\!]$.

3) Schéma numérique

S'appuyant sur les formules de Taylor,

- on approche $\dfrac{\partial^2 T}{\partial x^2}(x,t)$ par $\dfrac{1}{h^2}\left[T(x+h,t) - 2T(x,h) + T(x-h,t)\right]$;
- on approche $\dfrac{\partial T}{\partial t}(x,t)$ par $\dfrac{1}{k}\left[T(x,t+k) - T(x,t)\right]$.

L'équation de la chaleur se traduit alors par le *schéma numérique*

$$\frac{1}{k}\left(\widetilde{T}[n,i+1] - \widetilde{T}[n,i]\right) = \frac{D}{h^2}\left(\widetilde{T}[n+1,i] - 2\widetilde{T}[n,i] + \widetilde{T}[n-1,i]\right)$$

soit

$$\widetilde{T}[n,i+1] = \widetilde{T}[n,i] + C\left(\widetilde{T}[n+1,i] - 2\widetilde{T}[n,i] + \widetilde{T}[n-1,i]\right) \quad \text{où} \quad C = \frac{Dk}{h^2},$$

cela pour $(n,i) \in [\![1, N-1]\!] \times [\![0, P-1]\!]$.

On peut montrer qu'une condition nécessaire et suffisante de stabilité de ce schéma est $C \leq \dfrac{1}{2}$.

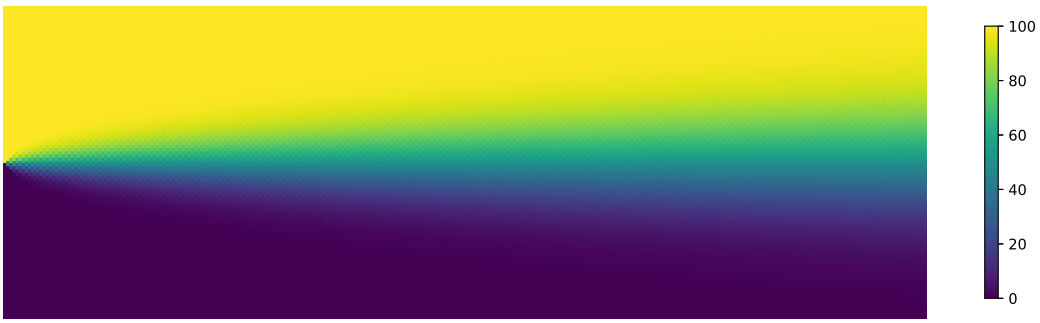
4) Implémentation avec Python

Le schéma numérique ci-dessus est *explicite*, il permet de remplir progressivement un tableau `numpy T`.

Coder cela en écrivant une fonction `tableau(L1, L2, T1, T2, N, P, C)` renvoyant ledit tableau `T`.

On choisit de transmettre comme paramètre la constante C pour tester plus facilement la stabilité : avec des valeurs de C inférieures à 0.5, on constate une diffusion d'autant plus rapide que C est grand (C est proportionnelle à $D\dots$). Mais avec $C > 0.5$, on ne voit plus rien de probant.

Pour visualiser l'évolution de la température, un simple `plt.matshow` suffit : il suffit d'imaginer la barre métallique placée verticalement, à gauche dans son état initial, la distribution de température aux instants successifs se voyant à la répartition des couleurs dans chaque "colonne" du rectangle qui s'affiche. Voici un exemple de résultat, pour $L_1 = L_2 = 1$, $T_1 = 100$, $T_2 = 0$:



L'affichage (ci-dessus à droite) de l'échelle des couleurs correspondant aux valeurs contenues dans la matrice s'obtient par `plt.colorbar(shrink=0.7)` (l'option `shrink` permet d'ajuster la longueur de la barre).

La suppression des graduations des axes autour de la matrice s'effectue par `plt.axis('off')`.

II - Équation des cordes vibrantes

1) Position du problème

Pour cet exemple, on considère une corde sans raideur fixée horizontalement à ses deux extrémités, d'abscisses 0 et L.
Le déplacement vertical du point d'abscisse x à l'instant t est noté $u(x,t)$.
On suppose que la corde est déformée à l'instant 0 selon une fonction f définie sur $[0, L]$ et lâchée avec une vitesse initiale nulle.

La fonction de deux variables u vérifie l'*équation de d'Alembert*

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \quad \text{où la constante } c \text{ est la } \textit{célérité} \text{ de l'onde sur la corde.}$$

Et dans la situation décrite ci-dessus, nous avons

- les *conditions initiales* : $\forall x \in [0, L] \quad u(x,0) = f(x) \quad \text{et} \quad \dfrac{\partial u}{\partial t}(x,0) = 0$;
- les *conditions aux limites* : $\forall t \geq 0 \quad u(0,t) = 0 \quad \text{et} \quad u(L,t) = 0.$

2) Discrétisation

Comme au **I**, on considère une subdivision de $[0, L]$ en N intervalles d'amplitude $h = L/N$.
Pour le temps, on fixe un pas de discrétisation k, un entier P et l'on se limite à $t \in [0, Pk]$.
On est alors amené à remplir un tableau U de type $(N+1, P+1)$, tel que $U[n,i]$ contienne une valeur approchée de $u(nh, ik)$, pour $(n,i) \in [\![0,N]\!] \times [\![0,P]\!]$.

3) Schéma numérique

S'appuyant sur les formules de Taylor,

- on approche $\dfrac{\partial^2 u}{\partial x^2}(x,t)$ par $\dfrac{1}{h^2}\left[u(x+h,t)-2u(x,h)+u(x-h,t)\right]$;
- on approche $\dfrac{\partial^2 u}{\partial t^2}(x,t)$ par $\dfrac{1}{k^2}\left[u(x,t+k)-2u(x,t)+u(x,t-k)\right]$.

L'équation de d'Alembert se traduit alors par le schéma numérique

$$\frac{1}{k^2}\left(U[n,i+1]-2U[n,i]+U[n,i-1]\right)=\frac{c^2}{h^2}\left(U[n+1,i]-2U[n,i]+U[n-1,i]\right)$$

soit

$$U[n,i+1]=2U[n,i]-U[n,i-1]+C\left(U[n+1,i]-2U[n,i]+U[n-1,i]\right) \quad \text{où} \quad C=\frac{c^2k^2}{h^2},$$

cela pour $(n,i)\in[\![1,N-1]\!]\times[\![1,P-1]\!]$.

On peut montrer qu'une condition nécessaire de stabilité de ce schéma est $C\leq 1$.

La condition initiale $\dfrac{\partial u}{\partial t}(x,0)=0$ peut se traduire en première approximation par $U[n,1]=U[n,0]$.

On peut affiner grâce à la formule de Taylor :

$$u(nh,k)=u(nh,0)+k\frac{\partial u}{\partial t}(nh,0)+\frac{k^2}{2}\frac{\partial^2 u}{\partial t^2}(nh,0)+o\left(k^2\right),$$

or, pour tout x,

$$\frac{\partial u}{\partial t}(x,0)=0 \quad \text{et} \quad \frac{\partial^2 u}{\partial t^2}(x,0)=c^2\frac{\partial^2 u}{\partial x^2}(x,0).$$

On utilisera donc l'approximation, pour $n\in[\![1,N-1]\!]$,

$$U[n,1]=U[n,0]+\frac{C}{2}\left(U[n+1,0]-2U[n,0]+U[n-1,0]\right).$$

4) Implémentation avec Python

Les relations ci-dessus permet de remplir progressivement un tableau `numpy` `U`.

Coder cela en écrivant une fonction `tableau(L, f, N, P, C)` renvoyant ledit tableau `U`, en supposant déjà définie la fonction Python `f` correspondant à la position initiale donnée par la fonction f.

Comme au **I**, on choisit de transmettre comme paramètre la constante C pour tester plus facilement la stabilité : avec des valeurs de C supérieures à 1, on obtient des résultats aberrants.

Pour visualiser les vibrations de la corde, on peut créer un "dessin animé" grâce au module `animation` de `matplotlib`. Voici un exemple de code supposant le tableau `U` déjà rempli :

```
import matplotlib.pyplot as plt
from matplotlib import animation

x = np.linspace(0, L, N+1)

fig = plt.figure()

def init():
    plt.clf()
    return plt.plot()

def animate(i):
    plt.clf()
    plt.axis([0, L, uMin, uMax])
    plt.plot(x,U[:, i])
    return plt.plot()

anim = animation.FuncAnimation(fig, animate, init_func=init, repeat=False,
                               frames=P-1, interval=10)
plt.show()
```

Les fonctions `init` et `animate` renvoient une "diapositive" et la méthode `FuncAnimation` crée le dessin animé résultant de l'enchaînement des diapositives ainsi créées.
Les constantes uMin et uMax (à ajuster) correspondent aux amplitudes extrêmes des vibrations.

5) Exemples

a) Solutions en ondes stationnaires

Tester le programme précédent avec des fonctions f de la forme $x \mapsto \sin\left(k\pi\dfrac{x}{L}\right)$, k fixé dans $\mathbb{N}^*$.

b) Propagation d'une gaussienne

Tester le programme précédent avec des fonctions f de la forme $x \mapsto \exp\left(-\lambda\,(x-a)^2\right)$, $\lambda > 0$ et $a \in]0, L[$ fixés.

III - Équation de la chaleur en dimension 2 : solution stationnaire

1) Position du problème

En dimension 2, l'équation de la chaleur devient

$$\frac{\partial T}{\partial t} = D\Delta T \ \left(= D\left[\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right]\right)$$

La description générale des solutions est hors de portée...

On se contente ici de rechercher la solution correspondant à l'état stationnaire vers lequel tend une plaque métallique carrée $[0, L]^2$ dont les quatre bords sont maintenus à des températures fixes T_g, T_h, T_d, T_b.
On cherche donc une fonction de deux variables $(x, y) \mapsto T(x, y)$ telle que $\Delta T = 0$ (*fonction harmonique*) et vérifiant les conditions aux limites

$$\forall x \in [0, L] \quad T(x, 0) = T_g \ , \ T(x, L) = T_d \quad \text{et} \quad \forall y \in [0, L] \quad T(0, y) = T_h \ , \ T(L, y) = T_b.$$

2) Discrétisation

À nouveau, on considère une subdivision de $[0, L]$ en N intervalles d'amplitude $h = L/N$.
Mais ici le temps n'intervient pas, on est donc amené à remplir un tableau $\widetilde{T}$ de type $(N+1, N+1)$, tel que $\widetilde{T}[n, p]$ contienne une valeur approchée de $T(nh, ph)$, pour $(n, p) \in [\![0, N]\!]^2$.

3) Schéma numérique

Selon l'approximation habituelle des dérivées partielles secondes, la condition $\Delta T = 0$ s'écrit

$$\frac{1}{h^2}\left(\widetilde{T}[n+1, p] - 2\widetilde{T}[n, p] + \widetilde{T}[n-1, p]\right) + \frac{1}{h^2}\left(\widetilde{T}[n, p+1] - 2\widetilde{T}[n, p] + \widetilde{T}[n, p-1]\right) = 0$$

soit, pour tout $(n, p) \in [\![1, N-1]\!]^2$

$$\widetilde{T}[n+1, p] + \widetilde{T}[n-1, p] - 4\widetilde{T}[n, p] + \widetilde{T}[n, p+1] + \widetilde{T}[n, p-1] = 0.$$

À ces $(N-1)^2$ équations s'ajoutent les $4N$ relations imposant les températures des quatre bords (ce sont bien $2(N+1) + 2(N-1)$ relations, puisqu'on doit choisir une seule valeur pour chaque coin !).
On est donc confronté à un système linéaire de N^2 équations à N^2 inconnues. La résolution "exacte" par une méthode du style pivot de Gauss aurait une complexité de l'ordre de N^6, ce qui est rédhibitoire pour de grandes valeurs de N.

4) Méthode itérative

On s'oriente pour ce type de résolution vers une *méthode itérative*, qui consiste à construire *une suite de tableaux $T^{(k)}$ convergeant vers la solution $\widetilde{T}$*, en s'arrêtant dès que l'on considère que l'on est "suffisamment proche" de la limite.
Faute de mieux, on utilise une condition d'arrêt de la forme $\nu\left(T^{(k)} - T^{(k-1)}\right) \leq \varepsilon$ où ν est une norme sur $\mathcal{M}_{N^2}(\mathbb{R})$ (le lecteur avisé remarquera que c'est dangereux...).
On choisit en pratique ε "petit mais pas trop", typiquement 0.01.

Une méthode itérative classique est celle de *Gauss-Seidel*, qui conduit ici à poser, pour $k \in \mathbb{N}^*$ et $(n,p) \in [\![1, N-1]\!]^2$,

$$T^{(k)}[n,p] = \frac{1}{4}\left(T^{(k)}[n-1,p] + T^{(k)}[n,p-1] + T^{(k-1)}[n+1,p] + T^{(k-1)}[n,p+1]\right). \qquad (1)$$

Le tableau initial $T^{(0)}$ peut être choisi arbitrairement, l'état stationnaire n'en dépend pas. Cela dit, pour limiter le nombre d'itérations, il semble raisonnable d'initialiser à une température "moyenne" au regard des températures imposées sur les bords...

5) Implémentation en Python

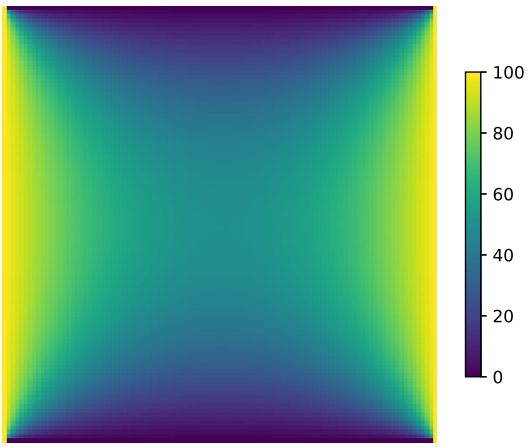
Dans ce cas particulier, la méthode de Gauss-Seidel converge, mais lentement... Il est donc sage de prévoir un *garde-fou*, à savoir un nombre d'itérations maximal au delà duquel on sort de la boucle `while` même si la condition $\nu\left(T^{(k)} - T^{(k-1)}\right) \leq \varepsilon$ n'est pas encore satisfaite.

Programmer ladite boucle `while`.
On peut utiliser pour ν la norme N_∞ qui à un tableau `numpy` `M` associe `np.max(np.abs(M))`.

Pour visualiser le résultat, un simple `plt.matshow` suffit, permettant de visualiser la distribution des températures sur la plaque. Comme au **I**, on peut afficher l'échelle des couleurs grâce à `plt.colorbar()`.

Avant la matrice, on peut afficher le nombre d'itérations effectuées.
Comparer les résultats obtenus avec divers tableaux initiaux $T^{(0)}$.

Voici un exemple de résultat, pour $T_g = T_d = 100$, $T_h = T_b = 0$:



Accélération par sur-relaxation : on peut obtenir une convergence plus rapide en modifiant la relation (1) de cette façon :

$$T^{(k)}[n,p] = (1-\omega)\,T^{(k-1)}[n,p] + \frac{\omega}{4}\left(T^{(k)}[n-1,p] + T^{(k)}[n,p-1] + T^{(k-1)}[n+1,p] + T^{(k-1)}[n,p+1]\right)$$

avec un paramètre ω bien choisi.

Pour l'équation étudiée ici, on peut montrer que le paramètre optimal est $\omega = \dfrac{2}{1+\sin(\pi/N)}$.

Comparer le nombre d'itérations effectuées pour une même précision avec et sans cette sur-relaxation.

Animation : on peut ici aussi créer une animation pour visualiser l'évolution de la distribution des températures au fil des itérations.
Pour cela, il suffit d'adapter le code proposé au **II-4)**, en remplaçant le `plt.plot` par un `plt.matshow`.
Attention toutefois, avec les options par défaut `matshow` ouvre une nouvelle fenêtre graphique à chaque appel, ce qui empêche l'affichage attendu de l'animation dans une seule fenêtre.
Pour remédier à cela, il suffit d'ajouter l'option `fignum` avec un numéro fixe, qui force `matshow` à toujours placer le dessin dans la même fenêtre, celle où l'animation apparaîtra.
Utiliser par exemple, dans la fonction `animate` : `plt.matshow(diapos[i], fignum=1)`, après avoir stocké à chaque itération (ou lors de certaines itérations s'il y en a beaucoup !) le nouveau tableau dans la liste `diapos`.