Étude numérique d'équations aux dérivées partielles avec conditions aux bords

Dans les programmes, on pourra utiliser à volonté les fonctions de la bibliothèque numpy du langage Python.

I. Intégrales

Soit f une fonction continue sur un intervalle $[a,b]$.

On veut résoudre numériquement le problème de Cauchy : $\begin{cases}\forall t\in[a,b],\ y'(t)=f(t)\\ y(a)=0\end{cases}$

1. Rappeler la solution exacte.
2. Analyser le rôle du programme suivant :

```
def integrale(f,a,t,n):
    h=(t-a)/n
    x=np.arange(a,t,h)
    return h*sum(f(x))
```

   Le tester pour pour approcher $\displaystyle\int_0^1 x\ \mathrm{d}x$.
3. Retrouver la fonction prédéfinie permettant d'améliorer l'approximation.

Pour la suite, munissez-vous des rappels de cours sur les matrices Numpy.

II. Équation de la chaleur

On s'intéresse ici à l'étude de l'évolution spatiale et temporelle de la température dans une barre (assimilée à un fil de longueur 1).
Cette barre a une température initiale T_0. Elle est soumise, à ses extrémités, à deux températures T_1 et T_2 .
On travaille dans un repère adapté $(A,\overrightarrow{AB})$, où A est l'extrémité gauche de la barre (soumise à la température T_1), et B est l'extrémité droite de la barre (soumise à la température T_2).
On travaille l'intervalle de temps $[t_0,t_f]$.
En supposant qu'il n'y a pas de diffusion de la chaleur avec l'air qui entoure la barre, et que celle-ci se propage donc uniquement dans la direction $\overrightarrow{u}_x$, on peut ramener cette barre à un modèle à une seule dimension.
On admet alors que la température $T(x,t)$ au temps t, et au point de la barre d'abscisse x, vérifie l'équation de la chaleur [1] :

1. *Cette relation requiert que $x\mapsto T(x,t)$ soit deux fois dérivable sur $[0,1]$ et $t\mapsto T(x,t)$ soit dérivable sur $[t_0,t_f]$.*

$$(E)\begin{cases}\boxed{\dfrac{\partial T(x,t)}{\partial t}=\lambda\dfrac{\partial^2 T(x,t)}{\partial x^2}}\\ \forall t\in[t_0,t_f],\ T(0,t)=T_1\\ \forall t\in[t_0,t_f],\ T(1,t)=T_2\\ \forall x\in]0,1[,\ T(x,t_0)=T_0\end{cases}$$

avec λ une constante réelle indépendante de t et de x.

Dans ce problème, on prendra $\boxed{\lambda=10^{-4}}$, $\boxed{t_0=0 \text{ et } t_f=50}$.
Soit alors $n\in\mathbb{N}^*$ fixé.
On discrétise l'intervalle $[0,1]$ en n intervalles $[x_i,x_{i+1}]$ de longueur $h_x=\frac{1}{n}$ et $x_i=ih_x$.
On discrétise l'intervalle $[0,t_f]$ en n intervalles $[t_j,t_{j+1}]$ de longueur $h_t=\frac{t_f}{n}$ et $t_j=jh_t$.
À chaque nœud x_i, on associe cherche à approcher $T(x_i,t_j)$, la température au temps t_j, par une méthode aux différences finies.

1. On fixe $t\in[t_0,t_f]$ et $x\in[0,1]$.

Rappeler comment approcher numériquement les dérivées $\dfrac{\partial T(x,t)}{\partial t}$ et $\dfrac{\partial^2 T(x,t)}{\partial x^2}$.

On en déduit (...) que la température $T(x_i,t_j)$ peut être approchée par le scalaire T_i^j vérifiant la relation :

$$\forall j\in[\![0,n-1]\!],\quad \begin{pmatrix}T_0^{j+1}\\T_1^{j+1}\\\vdots\\T_n^{j+1}\end{pmatrix}=\begin{pmatrix}T_0^{j}\\T_1^{j}\\\vdots\\T_n^{j}\end{pmatrix}+\underbrace{\frac{\lambda h_t}{(h_x)^2}\begin{pmatrix}0&0&0&0&\dots&0&0&0\\1&-2&1&0&\dots&0&0&0\\0&1&-2&1&&&&\\\vdots&\ddots&\ddots&\ddots&\ddots&\ddots&&\\&&&\ddots&\ddots&\ddots&1&0\\\vdots&&&&\ddots&1&-2&1\\0&\dots&&&\dots&0&0&0\end{pmatrix}}_{K_n}\begin{pmatrix}T_0^{j}\\T_1^{j}\\\vdots\\T_n^{j}\end{pmatrix}$$

Dans la suite, on posera : $Y_j=\begin{pmatrix}T_0^{j}\\T_1^{j}\\\vdots\\T_n^{j}\end{pmatrix}$. D'où $\boxed{Y_{j+1}=Y_j+K_nY_j}$

2. Déterminer le vecteur Y_0, en fonction de T_0, T_1 et T_2, puis écrire une fonction `Y0(`$n,T0,T1,T2$`)` qui retourne le vecteur colonne Y_0, sous forme d'un tableau `numpy.array`.

3. Écrire une fonction `K(`n`)` qui retourne la matrice thermique K_n, de taille $(n+1)$, sous forme d'un tableau `numpy.array`.
On pourra initialiser $K(n)$ grâce à la matrice nulle préprogrammée dans `numpy`, puis remplacer ses coefficients par double balayage.

4. En déduire une fonction `chaleur(`$j,n,T0,T1,T2$`)` qui retourne le vecteur colonne Y_j.

5. Quel est l'ordre de complexité de cette fonction, en fonction de n ?

REMARQUE : condition de stabilité du schéma numérique

On admet que l'on doit avoir la condition $\dfrac{\lambda h_t}{(h_x)^2} < \dfrac{1}{2}$ pour avoir la stabilité du schéma numérique explicite [2].

6. Écrire une fonction `coupleXY(`$j, n, T0, T1, T2$`)` qui retourne le couple (X, Z_j), où
- X est le $(n+1)$-uplet discrétisant à pas $\frac{1}{n}$ l'intervalle $[0, 1]$.
- Z_j est le $(n+1)$-uplet **en ligne** des coefficients de Y_j (donné par la fonction `chaleur`).

7. On prend $n = 100$.

On peut maintenant représenter graphiquement la fonction de température $x \mapsto T(x, t)$, pour chaque temps t_j appartenant à la discrétisation de l'intervalle $[t_0, t_f]$.

a) Écrire une suite d'instructions permettant de faire afficher, sur un même graphe, la courbe représentative de $x \mapsto T(x, t)$, pour t évoluant dans $[t_0, t_f]$.
On pourra s'amuser à modifier les valeurs de T_0, T_1 et T_2, par exemple :

$$T_0 = 50,\ T_1 = 40,\ T_2 = 60, \quad \text{puis} \quad T_0 = 100,\ T_1 = 50,\ T_2 = 50...$$

b) *Animation temporelle* : copier-coller les instructions du script `anim-chaleur.py`, permettant de faire afficher une animation de la courbe représentative de $x \mapsto T(x, t)$, pour t évoluant dans $[t_0, t_f]$.

2. Le scalaire $\dfrac{\lambda h_t}{(h_x)^2}$ définit le nombre CFL (Courant-Friedrich-Levy).

III. Équation d'onde de D'Alembert

On s'intéresse ici à l'étude d'une des équations les plus connues de la physique mathématique, que nous devons à Jean le Rond D'Alembert en 1747.
Elle décrit la propagation d'une onde dans le vide ou un milieu matériel quelconque.
On considère une corde inextensible, de masse linéique μ, tendue horizontalement avec une force constante F, fixée à ses deux bouts.
À l'équilibre, la corde est horizontale. On supposera dans la suite que la pesanteur n'intervient pas (sinon, la forme de la corde serait une chaînette).
On assimile la corde à un segment de longueur 1.
Pour tout $x \in [0,1]$, on note $(x,0)$ les coordonnées d'un point de la corde à l'équilibre et $(x, y(x,t))$ ses coordonnées au temps t.
On admet que la fonction y satisfait au problème de Cauchy (dit "de Dirichlet homogène") :

$$(E)\begin{cases} \boxed{\dfrac{\partial^2 y(x,t)}{\partial x^2} = \dfrac{1}{c^2}\dfrac{\partial^2 y(x,t)}{\partial t^2}} \\ \forall t \in [t_0, t_f],\ y(0,t) = y(1,t) = 0 \\ \forall x \in [0,1],\ y(x,t_0) = g(x) \quad \text{(position initiale)} \\ \forall x \in]0,1[,\ \dfrac{\partial y(x,t_0)}{\partial x} = g'(x) \quad \text{(vitesse initiale)} \end{cases}$$

avec $c = \dfrac{F}{\mu}$ la célérité de l'onde et g une fonction dérivable sur $[0,1]$ telle que $g(0) = g(1) = 0$.

Pour le schéma numérique aux différences finies, on reprend des notations similaires à celle de I.
On pose y_i^j une approximation de $y(x_i, t_j)$, puis Y_j la matrice colonne des y_i^j.
On démontre aisément que, pour tout j, Y_j vérifie la relation de récurrence :

$$\boxed{\forall j \in [\![1, n-1]\!], \quad Y_{j+1} = \frac{ch_t}{h_x} A Y_j + 2Y_j - Y_{j-1} \quad \text{avec } A \text{ la matrice tridiagonale de } 1, -2, 1}$$

Pour simplifier, on prendra $c = 1$, $t_0 = 0$ et $t_f = 50$.

1. Écrire des fonctions `Y0(n,g)` et `Y1(n,dg)` qui retournent respectivement le vecteur **colonne** Y_0 et Y_1, sous forme d'un tableau `numpy.array`.

2. Programmer une fonction `onde(j,n,g,dg)`, de paramètres un entier j, un entier n, une fonction g et sa fonction dérivée dg, et qui retourne le vecteur colonne Y_j.

REMARQUE : condition de stabilité du schéma numérique :

On admet que l'on doit avoir la condition $c\dfrac{h_t}{h_x} < 1$ pour avoir la stabilité de notre schéma numérique [3].

On prend maintenant $n = 100$.
On peut encore représenter graphiquement la position de la corde en fonction du temps.

a) Écrire une suite d'instructions permettant de faire afficher, sur un même graphe, la courbe représentative de $x \mapsto y(x,t)$, pour t évoluant dans $[t_0, t_f]$, avec par exemple :

$$g : x \mapsto \sin(\pi x)$$

3. Le scalaire $c\dfrac{h_t}{h_x}$ définit le nombre CFL de l'équation étudiée.

b) *Animation temporelle* : ...

ANNEXE : quelques rappels sur les tableaux numpy

On rappelle qu'une matrice $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ s'implémente en : `A=np.array([[1,2],[3,4]])`.
• Elle est du type 'numpy.ndarray'.
• On peut créer des tableaux lignes par le raccourci : `np.arange(start,stop,step)`.
Ne pas confondre avec `range(start,stop,step)` qui renvoie une liste (type *list*).

• Pour obtenir un coefficient, on appelle : `A[i][j]` ou `A[i,j]`. Attention, les indices commencent à 0...
• Les coefficients d'un tableau sont modifiables.

• **Matrices usuelles** :
`np.eye(taille)` ;
`np.zeros((largeur,hauteur))` ;
`np.ones((largeur,hauteur))` ;
`np.diag([liste des éléments diagonaux])` ;
`np.diag([liste des éléments surdiagonaux],k=1)`...

• Notez que A*B est la multiplication terme à terme, A**k est la puissance terme à terme.

`np.dot(A,B)` est la multiplication matricielle habituelle
`np.linalg.matrix_power(A,7)` la puissance matricielle.

• On comprendra aisément les commandes :
`A+B` ; `np.trace(A)` ; `np.transpose(A)` ; `np.linalg.det(A)` ; `np.linalg.inv(A)`