Magistère de mathématiques

TP2 – Résolution de systèmes linéaires

Exercice 1. *Conditionnement*

Étant donnée une matrice $M \in \mathsf{GL}_n(\mathbb{R})$ et une donnée $b \in \mathbb{R}^n$, la résolution du système linéaire $Mx = b$ peut s'avérer particulièrement sensible aux perturbations sur les données, ceci même pour des calculs qui seraient menés en arithmétique exacte. L'erreur relative commise sur la solution x peut être contrôlée (cf. cours) par le réel $\mathsf{cond}\, M := \|M\|\,\|M^{-1}\|$, qui se calcule en python avec `np.linalg.cond(`M`)`.

On s'intéresse à la matrice M_α suivante, dépendant du réel α :

$$M_\alpha = \begin{pmatrix} 1+\alpha & 1 & 2 \\ 3 & 1 & \alpha \\ 1 & 2\alpha & 1 \end{pmatrix}.$$

L'espace vectoriel $\mathbb{R}^3$ est muni de la norme euclidienne usuelle notée $\|\cdot\|$ et codée en python par `np.linalg.norm(`$\cdot$`)`.

a. Soit $\alpha = 2.01$. On pose $x = (1,1,1)^T$ et $b = M_\alpha x$.

(i) Calculer $y \in \mathbb{R}^3$ la solution de $M_\alpha y = b + \delta b$ où le vecteur $\delta b \in \mathbb{R}^3$ est une perturbation vectorielle obtenue via `0.01*np.random.rand(3,1)`.

(ii) Déterminer pour cette perturbation vectorielle l'erreur relative sur la solution :

$$\frac{\|y-x\|}{\|x\|},$$

et vérifier que l'inégalité suivante est bien réalisée

$$\frac{\|y-x\|}{\|x\|} \leq \mathsf{cond}\, M_\alpha \frac{\|\delta b\|}{\|b\|}.$$

b. Pour différentes valeurs de $\alpha \in [-5,5]$ et de multiples choix de données b et δb, reprendre la question précédente de sorte à contrôler par un tracé graphique l'inégalité précédente. On pourra tracer, en fonction de α, le rapport

$$\left(\frac{\|y-x\|}{\|x\|}\right)\left(\mathsf{cond}\, M_\alpha \frac{\|\delta b\|}{\|b\|}\right)^{-1}.$$

Exercice 2. *Mise en place d'un problème aux différences finies*

On souhaite déterminer une approximation de la solution $u \in \mathcal{C}^2(]0,1[) \cap \mathcal{C}^0([0,1])$ du problème suivant :

$$\begin{cases} \dfrac{d}{dx}\left(a(x)\dfrac{du}{dx}\right) = f(x), \quad 0 < x < 1, \\ u(0) = u(1) = 0. \end{cases}$$

Ce problème modélise le déplacement vertical d'une corde élastique pesante dont la position en l'absence de gravité serait horizontale (i.e. lorsque $f \equiv 0$). Son élasticité en tout point d'abscisse $x \in [0,1]$ est donnée par le coefficient $a(x) > 0$. Elle est fixée aux deux points de coordonnées $(0,0)$ et $(1,0)$ et est soumise à la charge linéique f qui en modifie la géométrie par l'effet de la gravité.

Une approximation de dimension finie du problème s'obtient de la manière suivante. On se donne $n \in \mathbb{N}$ un entier naturel non nul dont est déduit le pas de discrétisation d'espace $h = (n+1)^{-1}$. Pour tout entier i, on pose alors $x_i = ih$ et $x_{i+1/2} = (i+1/2)h$. L'inconnue est alors le vecteur $U = (u_1, \ldots, u_n)^T \in \mathbb{R}^n$ solution des équations aux différences finies :

$$\frac{1}{h}\left(a(x_{i+1/2})\frac{u_{i+1}-u_i}{h} - a(x_{i-1/2})\frac{u_i - u_{i-1}}{h}\right) = f(x_i), \quad 1 \leq i \leq n,$$
$$u_0 = u_{n+1} = 0.$$

On admettra que la solution de ce problème discret approche bien celle du problème continu au sens par exemple où la quantité $\max_{1\leq i\leq n} |u_i - u(x_i)|$ tend vers 0 lorsque la dimension n tend vers l'infini.

a. Déterminer *sur le papier* une matrice $A \in \mathsf{M}_n(\mathbb{R})$ et un vecteur $F \in \mathbb{R}^n$ permettant de réécrire le problème d'inconnue $U \in \mathbb{R}^n$ sous la forme d'un système linéaire $AU = F$.

b. Programmer la construction de la matrice A et du second membre F, à travers une fonction dont la syntaxe sera la suivante, à compléter :

```
def construction(n,a,f):
    # n entier, a et f le nom des fonctions concernees
    h = 1/(n+1)
    vec = a((0.5+np.arange(n+1))*h)
    ... # Astuce: utiliser np.diag
```

c. Vérifier que pour les fonctions a et f constantes égales à 1, le vecteur suivant

$$U = \left(\frac{x_i(x_i-1)}{2}\right)_{1\leq i\leq n}$$

est bien solution du problème, à l'erreur machine près.

Exercice 3. ***Chargement d'une corde d'élasticité inhomogène***

Attention : cet exercice suit et réutilise les fonctions développées dans l'exercice précédent.

On considère le problème précédemment présenté. Les fonctions a et f sont désormais choisies ainsi :

$$a(x) = 1 + 0.8\sin(10x)\cos(5x),$$
$$f(x) = 0.1 + \mathrm{e}^{-1000(x-\xi_1)^2} + \mathrm{e}^{-1000(x-\xi_2)^2}.$$

La contribution constante dans la fonction f peut s'interpréter comme la contribution de la masse propre de la corde, tandis que les deux termes exponentiels figurent deux masses suspendues, localisées au voisinage des points d'abscisses respectives $\xi_1 = 0.2$ et $\xi_2 = 0.7$.

a. Calculer la matrice A qui intervient lorsque la discrétisation correspond à $n = 10$ points et vérifier numériquement que A est bien symétrique, définie et négative.

b. Résoudre le problème pour les données précédentes et représenter la forme de la corde successivement pour des valeurs de n de plus en plus grandes. En parallèle, étudier le temps de calcul nécessaire pour construire A et déterminer U, en fonction de n.

On pourra utiliser le module `time` : la fonction `time.time()` donne l'heure en secondes.

c. On suppose la position de la seconde masse fixée à la valeur $\xi_2 = 0.7$. Déterminer par un procédé numérique de votre choix la position ξ_1 de la première masse dans $[0, 1]$ qui semble minimiser, puis celle qui semble maximiser la quantité $\min u(x)$.

d. Reprendre la question précédente, avec cette fois-ci les deux positions ξ_1 et ξ_2 libres et à déterminer de manière optimale.

e. (Bonus) Tracer une solution approchée de

$$\begin{cases} -\partial_t^2 u(t,x) + \partial_x(a(x)\partial_x u(t,x)) = f(x), & t \geq 0, x \in]0,1[\\ u(0,x) = \sin(x) & x \in]0,1[\\ u(t,0) = u(t,1) = 0 & t \geq 0 \end{cases}$$

Pour cela, on utilisera le schéma aux différences finies suivant

$$\begin{cases} -\dfrac{U_j^{n+1} - 2U_j^n + U_j^{n-1}}{\Delta t} + \dfrac{1}{\Delta x}\left(a(x_{j+1/2})\dfrac{U_{j+1}^n - U_j^n}{\Delta x} - a(x_{j-1/2})\dfrac{U_j^n - U_{j-1}^n}{\Delta x}\right) = f(x_j) \\ U_0^n = U_{J+1}^n = 0 \\ U_j^{-1} = 0, U_j^0 = \sin(x_j) \end{cases}$$

où U_j^n approche la valeur de $u(t^n, x_j)$ avec $t^n = n\Delta t$, $x_j = j\Delta x$, $\Delta x = \frac{1}{J+1}$ et $\Delta t > 0$.

On tracera la solution en fonction du temps en adaptant le code suivant :

```
from math import *
import numpy as np
import matplotlib.pyplot as plt

T = np.linspace(0,10,300)
X = np.linspace(0,10,400)

def f(t,x):
   return cos(x-t)

for t in T:
   Y = np.vectorize(f)(t,X)
   if t == 0:
      line, = plt.plot(X, Y)
   else:
      line.set_ydata(Y)
   plt.pause(0.01) # pause avec duree en secondes
plt.close()

plt.show()
```