Le calcul numérique en Python

Le module `numpy` fournit, en Python, un ensemble de fonctions et d'objets utiles au calcul numérique. Nous allons nous y intéresser aujourd'hui à travers une application illustrative : la résolution d'un système d'équations linéaire inhomogène.

1 Rappels mathématiques

Dans la suite de ce document, nous considèrerons le système suivant :

$$\begin{aligned} a_{1,1}x_1 + \cdots + a_{1,n}x_n &= b_1 \\ \vdots & \\ a_{n,1}x_1 + \cdots + a_{n,n}x_n &= b_n \end{aligned}$$

que nous considèrerons sous sa forme matricielle :

$$A \cdot x = b$$

où

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix}$$

$$x = (x_1, \dots, x_n)$$

$$b = (b_1, \dots, b_n)$$

Rappelons que cette équation admet une solution x_{sol} unique si et seulement si A est inversible et qu'alors cette solution est

$$x = A^{-1} \cdot b$$

Rappelons également que si A est inversible, son inverse peut s'écrire :

$$A^{-1} = \frac{1}{det(A)} \cdot com(A)^t$$

2 Résolution d'un tel système d'équations

D'après les formules précédentes, pour résoudre ce type de système, il nous faudra inverser la matrice A et donc calculer son déterminant et la transposée de sa comatrice. Pour ce qui est de la transposition, `numpy` fournit une fonction à cet effet, nous l'utiliserons :

```
transposee_a = numpy.transpose(a)
```

Pour le reste, ce sera à vous de coder les fonctions `det` et `comatrice`. De plus, le calcul du déterminant d'une matrice (n, n) faisant intervenir des sous-matrices de taille $(n-1, n-1)$, vous devrez coder une fonction `enlever_ligne_colonne` qui retourne une copie de la matrice fournie en entrée dans laquelle on a supprimé la ligne i et la colonne j. Pour cela, vous utiliserez la fonction `delete` de `numpy` (documentation de la fonction `delete`). Une fois ces fonctions codées, vous pouvez réaliser la fonction `resoudre_systeme_lineaire_inhomogene` qui prend en entrée une matrice A, un vecteur b et retourne x_{sol}. Vous pourrez tester cette fonction à l'aide des données suivantes :

```
a = numpy.array([[2, 3, 3, 1],
                 [-4, -6, 3, 2],
                 [-1, 1, 1, 1],
                 [-2, -1, 1, 1]])
b = numpy.array([15, 3, 5, 1])
```

Ce sujet de TD avait pour but de vous faire manipuler `numpy`. La fonction de résolution de systèmes que vous avez implémentée existe en fait déjà dans `numpy`. Testez le résultat obtenu avec votre implémentation à celui fourni par :

```
x = numpy.linalg.solve(a, b)
```

3 Exercice de synthèse

Il existe, dans le module `numpy`, une fonction qui permet de charger des matrices écrites dans des fichiers texte : la fonction `loadtxt` (documentation de la fonction `loadtxt`). Utilisez cette fonction pour résoudre le système d'équations représenté par la matrice A du fichier `A.txt` et le vecteur b contenu dans `b.txt`.