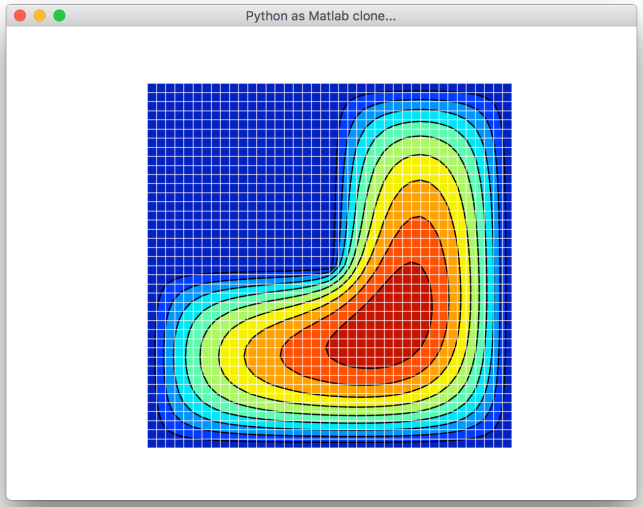
Python 21-22 for dummies : problème 10
Python comme clône de MATLAB



Un des exemples les plus célèbres de MATLAB est la résolution d'une équation de Poisson sur un domaine en forme de L : c'est même encore et toujours le logo de Mathworks qui commercialise ce logiciel. Il s'agit maintenant de faire aussi bien et même mieux avec `python`.

On considère un carré de côté deux, centré à l'origine et discrétisé spatialement par un maillage de $(2n+1) \times (2n+1)$ noeuds, ainsi que l'équation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + 1 = 0$$

dont le domaine est un sous-ensemble du carré et forme la lettre L inversée. Les opérateurs différentiels sont discrétisés avec des différence finies. Sur l'ensemble de la frontière du domaine, on impose une unique condition essentielle homogène : $u(x, y) = 0$.

Quelques indications sur les différences finies...

Pur obtenir une solution approchée par la méthode des différences finies, on définit une grille ou un maillage uniforme de n^2 points distants d'un pas $h = 2/(n-1)$. Les coordonnées des noeuds du maillage sont

$$\mathbf{X}_{ij} \;=\; (X_i, Y_j) \;=\; (-1+ih, -1+jh) \qquad\qquad i,j = 0, \ldots, n-1$$

Les valeurs approchées de la fonction inconnue en ces points du maillage seront notées :

$$U_{ij} \;=\; u^h(\mathbf{X}_{ij}) \;\approx\; u(\mathbf{X}_{ij})$$

Ensuite, on approche les dérivées partielles du laplacien par des différences finies du second ordre afin d'obtenir une approximation discrète du laplacien avec 5 noeuds.

$$\left(\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h^2} + \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{h^2}\right) + 1 \;=\; 0$$

$$\downarrow$$

$$\frac{U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4U_{i,j}}{h^2} + 1 \;=\; 0$$

Plus précisément, on vous demande de :

1. Ecrire une fonction `U = poissonSolve(n)` qui fournit un tableau de taille $(2n+1) \times (2n+1)$ contenant le solution du problème telle qu'illustrée sur la figure.
2. Comme d'habitude, un programme `poissonTest.py` vous est fourni pour tester votre fonction.

```
from numpy import *
from timeit import default_timer as timer
from poisson import poissonSolve

n = 20; tic = timer()
U = poissonSolve(n)
print("      Elapsed time : %f seconds" % (timer() - tic))
print(" ==== Maximum value of U : %10.8f " % amax(U))
print(" ==== Minimum value of U : %10.8f " % amin(U))

import matplotlib.pyplot as plt
import matplotlib
myColorMap = matplotlib.cm.jet

X = linspace(-1,1,2*n+1); U = abs(U)
plt.figure("Python as Matlab clone...")
plt.contourf(X,X,U,10,cmap=myColorMap)
plt.contour(X,X,U,10,colors='k',linewidths=1)
plt.hlines(X,X.min(),X.max(),color='white',linewidths=0.5)
plt.vlines(X,X.min(),X.max(),color='white',linewidths=0.5)
plt.axis("off"); plt.axis("equal")
plt.show()
```

3. Comme il faut fêter la fin du quadrimestre et peut-être bientôt la fin du confinement, on vous a fourni un canevas de départ qui résout l'équation sur l'ensemble du carré : il suffit de faire une toute petite (vraiment toute petite) modification pour obtenir le résultat souhaité :-)
4. Comme vous êtes astucieux et intelligents, il faut en outre effectuer encore quelques toutes petites modifications dans le code pour tirer profit du caractère creux de la matrice.
5. Votre fonction (avec les éventuelles sous-fonctions que vous auriez créées) sera incluse dans un unique fichier `poisson.py`, sans y adjoindre le programme de test fourni !

Ce n'est pas si monstrueux que cela à faire et c'est vraiment un bon exercice à réaliser pour vérifier votre compréhension de `python`. Par contre, cela n'est pas utile de consacrer un temps irréaliste à ce petit problème : une solution détaillée sera postée sur le site web à la fin du quadrimestre : elle fera réellement partie de la matière de l'examen et il est donc judicieux de la comprendre et de la confronter avec votre propre solution.

Bon courage à tous, bon blocus et bonne chance pour vos examens.