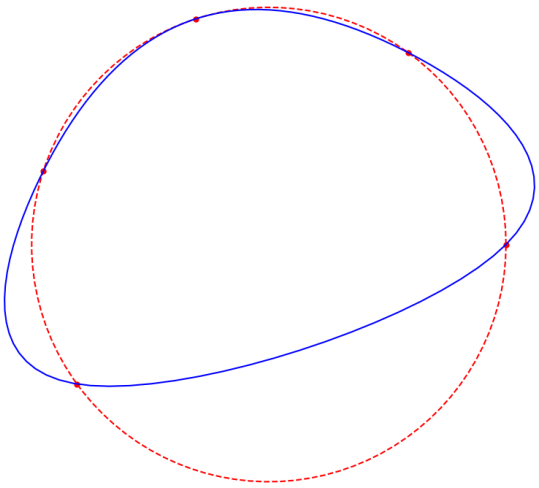
Notebook : Splines cubiques périodiques

Jérome EERTMANS
Gilles PONCELET

Ce document a pour but d'aider les étudiants du cours de Méthodes Numériques, donné par M. Vincent LEGAT à l'Ecole Polytechnique de Louvain, en leur offrant une solution détaillée aux devoirs.



1 Le problème à résoudre

L'objectif de ce devoir est de réaliser une interpolation périodique à l'aide de Splines cubiques. Pour ce faire, il faut implémenter la fonction `spline(x, h, U)`.

1.1 Les arguments de la fonction

— `x` : un vecteur numpy unidimensionnel de longueur quelconque reprenant les abscisses auxquelles on souhaite évaluer l'interpolation obtenue
— `h` et `U` : des vecteurs numpy unidimensionels de longueur n

1.2 Le système à résoudre

Comme vous ne disposez pas des dérivées secondes U'' et que vous en avez besoin pour effectuer l'interpolation, il va falloir les déterminer en trouver la solution aux équations :

$$\frac{h^2}{6}\underbrace{(U''_{i-1} + 4U''_i + U''_{i+1})}_{Ax} = \underbrace{U_{i-1} - 2U_i + U_{i+1}}_{b} \tag{1}$$

Tout ceci peut être représenté [1] sous forme de système matriciel de la forme $\frac{h^2}{6}Ax = b$. On trouve donc que la matrice A est de la forme :

$$A = \begin{pmatrix} 4 & 1 & 0 & 0 & 1 \\ 1 & 4 & 1 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 1 & 4 & 1 \\ 1 & 0 & 0 & 1 & 4 \end{pmatrix} \tag{2}$$

Pour le vecteur b, un raisonnement similaire peut-être appliqué en utilisant la périodicité de U :

$$..., U_{-2} = U_3, U_{-1} = U_4, \underbrace{U_0, U_1, U_2, U_3, U_4}_{U_i}, U_5 = U_0, U_6 = U_1, ...$$

Dans Python, les indices négatifs sont acceptés et bouclent bien sur le vecteur comme on le souhaiterait ici. Ce n'est pas le cas des indices dépassant l'indice maximum : il faut donc gérer ce cas spécifiquement ! Appliquons la formule sur chacun des intervalles :

$$b_i = U_{i-1} - 2U_i + U_{i+1} \quad \forall \quad i = 0, .., 4 \tag{3}$$

Après construction du système, on peut le résoudre avec la fonction `solve` bien connue.

1.3 Interpolation à réaliser

Il s'agit d'implémenter l'interpolation donnée par la formule suivante :

$$\begin{aligned} u^h(x) = &\underbrace{\frac{U''_{i-1}}{6h}}_{A}(X_i - x)^3 \\ &+ \underbrace{\frac{U''_i}{6h}}_{B}(x - X_{i-1})^3 \\ &+ \underbrace{(\frac{U_{i-1}}{h} - \frac{U''_{i-1}h}{6})}_{C}\underbrace{(X_i - x)}_{r} \\ &+ \underbrace{(\frac{U_i}{h} - \frac{U''_i h}{6})}_{D}\underbrace{(x - X_{i-1})}_{s} \end{aligned}$$

1. Ici on prend l'exemple de $n = 5$

Rien de plus ! Les éventuelles améliorations à réaliser sont déjà détaillées dans la solution du devoir précédent :).

2 Résolution du problème

2.1 Le code

Pour ce devoir, la construction du système était la partie la plus compliquée et on vous conseille de le faire sur papier et/ou avec des boucles Python pour commencer plus facilement. L'interpolation est quand à elle très similaire à celle du devoir 2.

```
def spline(x, h, U):
    n = np.size(U)
    X = np.arange(0, n+1)*h
    i = searchsorted(X[1:],x) # Ceci est la version plus rapide
                              # de détection des intervalles :)

    # Construction du système
    d = np.full(n, 4) # vecteur rempli de 4
    A = np.diag(d) + np.diag(no.ones(n-1), -1) + np.diag(ones(n-1), 1)
    A[0, n-1] = 1; A[n-1, 0] = 1
    b = U[range(-1, n-1)] - 2*U[0:n] + U[np.append(np.arange(1, n), [0])]

    ddU = solve(A, b) / ((h**2)/6) # Vaut mieux multiplier n éléments
    # par 6/h**2, plutôt que n*n éléments par h**2/6

    n = len(X) # n += 1 revient au même
    # On calcul sur les petits  vecteurs
    A = ddU[0:n-1]/(6*h)
    B = ddU[1:n]/(6*h)
    C = (U[0:n-1]/h - ddU[0:n-1]*h/6)
    D = (U[1:n]/h - ddU[1:n]*h/6)

    # Puis on évalue sur beaucoup de points
    r = X[i+1] - x
    s = x - X[i]

    return r*(A[i]*r*r + C[i]) + s*(B[i]*s*s + D[i])
```

2.2 Les explications

Ce devoir étant très similaire au précédent, on vous invite en lire la résolution si des explications sur la vectorisation effectuée vous intéressent.