Python for dummies : problème 3
Splines cubiques périodiques

L'interpolation par les splines cubiques usuelles sur l'intervalle $[X_{i-1}, X_i]$ est définie par l'expression :

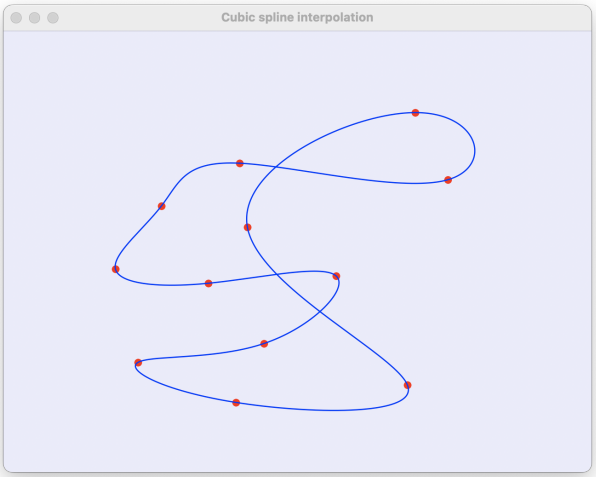
$$\begin{aligned} u^{h_i}(x) &= \frac{U''_{i-1}}{6h_i}(X_i - x)^3 + \frac{U''_i}{6h_i}(x - X_{i-1})^3 \\ &+ \left(\frac{U_{i-1}}{h_i} - \frac{U''_{i-1}\, h_i}{6}\right)(X_i - x) \\ &+ \left(\frac{U_i}{h_i} - \frac{U''_i\, h_i}{6}\right)(x - X_{i-1}) \end{aligned}$$

où U_i et U''_i sont les valeurs de la fonction et de la dérivée seconde en $x = X_i$. Finalement, on définit les écarts $h_i = X_i - X_{i-1}$. Les dérivées secondes U''_i sont obtenues en résolvant une système linéaire permettant d'obtenir une interpolation cubique par morceaux de classe C_2 d'une fonction $u(x)$ dont on connaît les ordonnées U_i aux abscisses X_i. Le système à résoudre s'écrit sous la forme

$$\frac{h_i}{6}\, U''_{i-1} + \frac{2(h_i + h_{i+1})}{6}\, U''_i + \frac{h_{i+1}}{6}\, U''_{i+1} = \frac{(U_{i+1} - U_i)}{h_{i+1}} - \frac{(U_i - U_{i-1})}{h_i}$$

$$i = 1, \ldots, n-1$$

Dans cette mission, nous allons calculer l'interpolation cubique par morceaux d'une courbe fermée $(x(t), y(t))$ qui sera définie à partir de n points (X_k, Y_k) pour des instants équidistants $T_k = kh$ avec $k = 0, \ldots, n$.



Attention, on ne dédouble pas le point initial de la courbe que l'on souhaite construire !

Plus concrètement, il s'agit d'écrire une fonction `uh = splines(x,h,U)` qui détermine les valeurs en x de l'interpolation périodique cubique par morceaux pour les valeurs U_i aux points $X_i = ih$. Le point initial et le point final étant identiques pour une courbe fermée, il faut évidemment légèrement modifier le système linéaire pour obtenir une solution périodique, dont la dérivée seconde, au point de départ, sera identique à celle au point d'arrivée. Il est utile d'observer, par contre, qu'on peut tirer profit du fait que les abscisses temporelles sont toujours équidistantes. En d'autres mots, tous les intervalles ont une longueur identique $h_i = h$.

1. Comme `python` est très compliqué, nous vous donnons, exceptionnellement, un canevas contenant déjà une grosse partie de la solution car elle calcule l'interpolation linéaire par morceaux....

```
def spline(x,h,U):
  n = size(U); X = arange(0,n+1)*h

  i = zeros(len(x),dtype=int)
  for j in range(1,n):
      i[X[j]<=x] = j

  U = append(U,U[0])
  return U[i]*(X[i+1]-x)/h + U[i+1]*(x-X[i])/h
```

   Bien comprendre les instructions qui permettent d'identifier pour chaque élément `x[index]` du tableau `x` l'intervalle `i[index]` à considérer !

   Cela permet ensuite de bêtement calculer l'interpolation linéaire en prenant les deux points qui encadrent cet intervalle. De manière vectorisée, tout cela se fait avec une unique ligne de code :-)

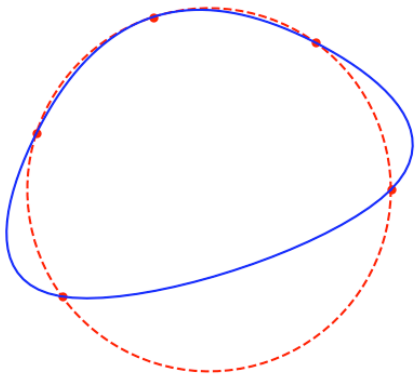
   Et après, vous allez me dire que les petits programmes `python` sont trop compliqués à réaliser.

2. Ensuite pour tester, votre programme, on vous a vous fourni un tout petit programme simple `splineTest.py` pour interpoler un cercle.

```
n = 4;
h = 3*pi/(2*(n+1));
T = arange(0,3*pi/2,h)
X = cos(T); Y = sin(T)

fig = plt.figure()
plt.plot(X,Y,'.r',markersize=10)
t = linspace(0,2*pi,100)
plt.plot(cos(t),sin(t),'--r')

t = linspace(0,3*pi/2,100)
plt.plot(spline(t,h,X),spline(t,h,Y),'-b')
plt.axis("equal"); plt.axis("off")
plt.show()
```



3. Finalement, nous avons toujours le même programme un peu plus rigolo `splineTestFun.py` qui permet de définir plein de points en cliquant sur la figure. Un double click permet d'obtenir la courbe fermée obtenue par l'interpolation périodique cubique par morceaux.

4. Votre fonction (avec les éventuelles sous-fonctions que vous auriez créées) sera soumise via le site web du cours.

Un petit programme interactif avec `matplotlib`...

Pour les petits curieux, voici l'implémentation de `splineTestFun.py` pour définir de manière magique une courbe à interpoler...

```
import matplotlib
from matplotlib import pyplot as plt
from numpy import *
from splineTest import spline

# ====================== callback pour les événements avec la souris ======
#
#  Observer la gestion distincte du clic simple et double :-)
#  Apres un evenement, on redessine la figure avec draw()
#

def mouse(event):
  global X,Y,n
  if (event.dblclick):
    t  = arange(0,n+0.001,0.001)
    x  = spline(t,1.0,X)
    y  = spline(t,1.0,Y)
    plt.plot(x,y,'-b')
    X,Y = [],[]; n = 0
  else :
    x = event.xdata
    y = event.ydata
    if (x != None and y != None) :
      n = n + 1
      X = append(X,[x])
      Y = append(Y,[y])
      print("New data : " + str(x) + "," + str(y))
      plt.plot([x],[y],'.r',markersize=10)
  fig.canvas.draw()


# ============================ mainProgram ================================


matplotlib.rcParams['toolbar'] = 'None'
matplotlib.rcParams['lines.linewidth'] = 1
plt.rcParams['figure.facecolor'] = 'lavender'

X,Y = [],[]; n = 0
fig = plt.figure("Cubic spline interpolation")
fig.canvas.mpl_connect('button_press_event',mouse)
plt.ylim((0,1)); plt.xlim((0,1.3)); plt.axis("off")

plt.show()
```