Université Grenoble Alpes
MAP350 – L3 MI

TP7 : Splines naturelles cubiques C2 d'interpolation & Cas paramétrique

---

Etant donnée une fonction f définie sur un intervalle $[a, b]$, le but de ce TP est d'interpoler f par une fonction spline cubique C^2 en n points distincts de l'intervalle $[a, b]$. Dans une première partie nous considérons le cas (dit uniforme) de n points équirépartis dans l'intervalle $[a, b]$. Ensuite nous considérerons le cas général (cas non uniforme) de n points distincts quelconques dans $[a, b]$. Finalement, nous considérerons le cas de l'interpolation spline paramétrique avec paramétrisation uniforme et cordale. Dans tous les cas la suite des points d'interpolation est strictement croissante. Dans ce document nous considérons uniquement les *splines naturelles* et nous employons parfois le terme *spline* pour *spline cubique C2 naturelle.*

Exercise 1

Lire attentivement l'ensemble de ce document et réaliser un schéma organisationnel des différentes fonctions Python à écrire afin d'obtenir un squelette du programme. Pour chacune de ces fonctions, préciser
- les données en Input (les paramètres de la fonction en précisant leur type),
- les tâches réalisées par cette fonction,
- les données en Output (les sorties et affichage éventuels).

1. Cas uniforme

Soit n points x_i équirépartis dans l'intervalle $[a, b]$

$$a = x_1 < x_2 < \cdots < x_{n-1} < x_n = b,$$

$$x_i = a + (i-1)\,h, \quad i = 1, 2, \ldots, n, \qquad h = \frac{b-a}{n-1}.$$

Les données d'interpolation sont les points $(x_i, y_i = f(x_i))$, $i = 1, \ldots, n$. La spline d'interpolation est définie sur chaque intervalle $[x_i, x_{i+1}]$ par un polynôme cubique exprimé dans la base de Hermite associée à cet intervalle. Les dérivées y'_i en chaque point x_i sont déterminées de sorte à assurer la continuité C^2 entre les différents polynômes cubiques aux noeuds intérieurs x_i et la condition des splines naturelles en a et b.
Ces dérivées y'_i sont obtenues par résolution du système linéaire suivant.

$$\begin{pmatrix} 2 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 1 & 4 & 1 & & & & 0 \\ 0 & 1 & 4 & 1 & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & 1 & 4 & 1 & 0 \\ 0 & & & & 1 & 4 & 1 \\ 0 & 0 & \cdots & \cdots & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} y'_1 \\ y'_2 \\ y'_3 \\ \vdots \\ y'_{n-2} \\ y'_{n-1} \\ y'_n \end{pmatrix} = \frac{3}{h} \begin{pmatrix} y_2 - y_1 \\ y_3 - y_1 \\ y_4 - y_2 \\ \vdots \\ y_{n-1} - y_{n-3} \\ y_n - y_{n-2} \\ y_n - y_{n-1} \end{pmatrix}.$$

Exercise 2

1. Ecrire un script Python permettant de réaliser l'interpolation spline uniforme.
On rappelle que la construction de la matrice du système linéaire ci-dessus a déjà été réalisée lors du TP d'introduction à Python. On s'appuiera également sur le TP concernant l'interpolation de Hermite.
2. Tester cette interpolation spline pour la fonction $f(x) = \sin(x^2 - 2\,x + 1) + \big[\cos(x^3 + x)\big]^2$ sur l'intervalle $[a, b] = [-1, 2]$.

2. Cas non uniforme

On considère désormais n points distincts quelconques x_i dans l'intervalle $[a, b]$

$$a = x_1 < x_2 < \cdots < x_{n-1} < x_n = b \quad \text{et} \quad h_i = x_{i+1} - x_i,\ i = 1\ldots, n-1.$$

La méthode est similaire au cas uniforme. La spline d'interpolation des données $(x_i, y_i = f(x_i))$ est définie sur chaque intervalle $[x_i, x_{i+1}]$ par un polynôme cubique exprimé dans la base de Hermite associée à cet intervalle. Les dérivées y_i' réalisant la continuité C^2 et la condition des splines naturelles sont obtenues par résolution du système linéaire suivant.

$$\begin{pmatrix}
2 & 1 & 0 & \cdots & \cdots & \cdots & 0 & 0 \\
h_2 & 2(h_1+h_2) & h_1 & 0 & & & & 0 \\
0 & h_3 & 2(h_2+h_3) & h_2 & 0 & & & \vdots \\
\vdots & & \ddots & \ddots & \ddots & & & \vdots \\
\vdots & & 0 & h_i & 2(h_{i-1}+h_i) & h_{i-1} & 0 & \vdots \\
\vdots & & & & \ddots & \ddots & \ddots & 0 \\
0 & & & & 0 & h_{n-1} & 2(h_{n-2}+h_{n-1}) & h_{n-2} \\
0 & 0 & \cdots & \cdots & \cdots & 0 & 1 & 2
\end{pmatrix}
\begin{pmatrix}
y_1' \\ y_2' \\ y_3' \\ \vdots \\ y_{n-2}' \\ y_{n-1}' \\ y_n'
\end{pmatrix}$$

$$= 3 \begin{pmatrix}
\frac{1}{h_1}(y_2 - y_1) \\
\frac{h_1}{h_2}(y_3 - y_2) + \frac{h_2}{h_1}(y_2 - y_1) \\
\vdots \\
\frac{h_{i-1}}{h_i}(y_{i+1} - y_i) + \frac{h_i}{h_{i-1}}(y_i - y_{i-1}) \\
\vdots \\
\frac{h_{n-2}}{h_{n-1}}(y_n - y_{n-1}) + \frac{h_{n-1}}{h_{n-2}}(y_{n-1} - y_{n-2}) \\
\frac{1}{h_{n-1}}(y_n - y_{n-1})
\end{pmatrix}.$$

Exercise 3

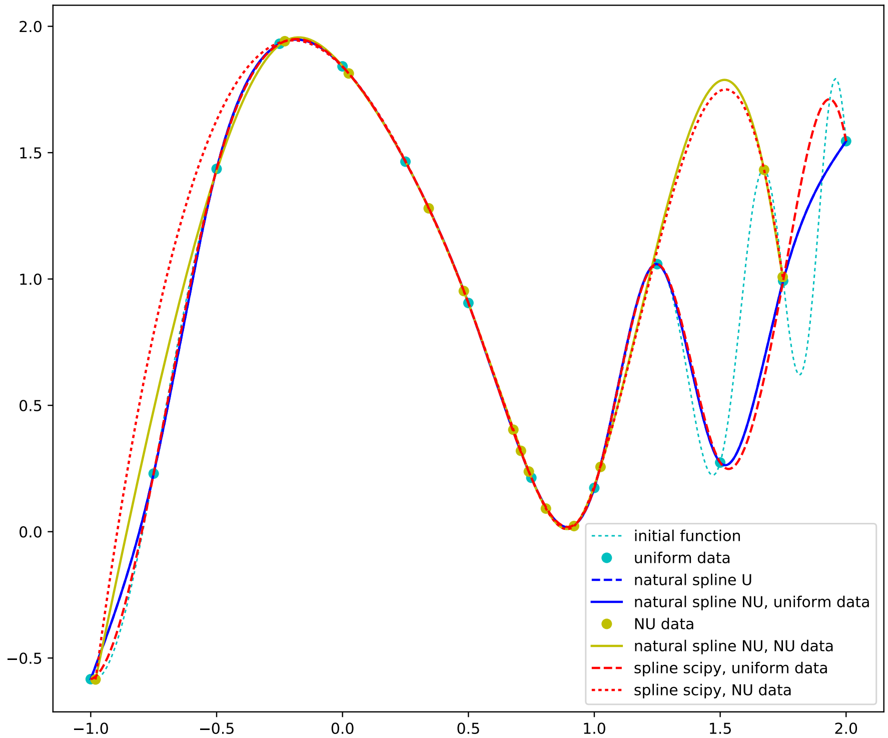
1. *Ecrire un script Python permettant de réaliser l'interpolation spline non uniforme.*
2. *Tester cette interpolation spline pour la fonction* $f(x) = \sin(x^2 - 2x + 1) + \left[\cos(x^3 + x)\right]^2$ *sur l'intervalle* $[a, b] = [-1, 2]$*. Les* n *points d'interpolation* x_i *pourront être déterminés selon une loi de distribution uniforme, par exemple :* `xi = a + (b-a) * np.random.rand(n)`.

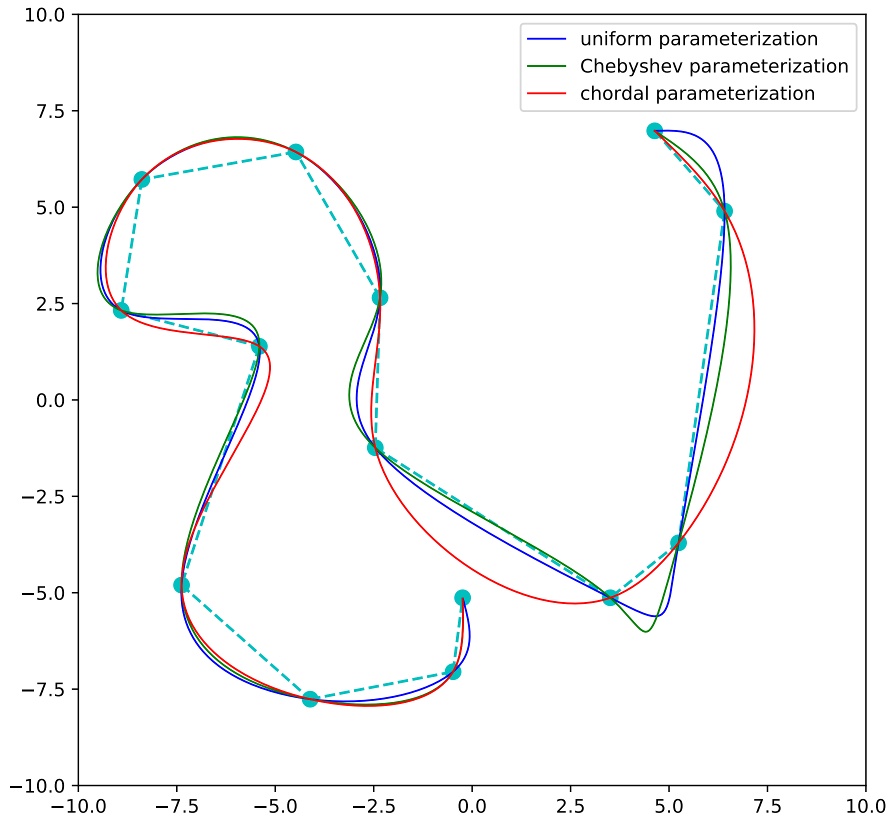
3. Cas paramétrique

Le contexte est similaire à celui de l'*interpolation paramétrique polynomiale*. Etant donné un polygone de n points $M_i = (x_i, y_i)$, $i = 1\ldots, n$, on cherche ici une courbe spline paramétrée $t \in [0, 1] \mapsto m(t) = (m_x(t), m_y(t))$ interpolant les points M_i, c'est à dire telle que $m(t_i) = M_i$ pour une suite donnée de paramètres t_i dans l'intervalle $[0, 1]$.

Exercise 4

Ecrire un script Python réalisant cette interpolation spline paramétrique. On considèrera la paramétrisation uniforme et la paramétrisation cordale. On fera attention qu'ici la numérotation des points commence à 1 (et non pas à 0). L'acquisition des points sera à nouveau réalisée à l'aide de la fonction `PolygonAcquisition(color1,color2)`.





Le compte rendu de ce TP consistera en un seul fichier Python dont le nom sera `TP7_NOM1_NOM2.py`
Ce script Python contiendra en entête (et donc en commentaire) les noms NOM1 et NOM2 des éléments du binôme. Ce script sera envoyé dans un mail dont le sujet sera `TP7_NOM1_NOM2`. L'exécution de ce script devra permettre d'éxécuter <u>uniquement</u> l'interpolation paramétrique (uniforme et cordale) selon les instructions de l'exercice 4.