LINMA1170 **Devoir 1 (v2)**
Analyse Numérique **Décomposition QR, moindres carrés et B-splines**
Prof. J-F Remacle **Echéance : vendredi 23 février 2024 à 18:15**

On considère le problème des moindres carrés. Soit $A \in \mathbb{R}^{m\times n}$ avec $m \geq n$ et $B \in \mathbb{R}^{m\times p}$. On cherche $X \in \mathbb{R}^{n\times p}$ qui minimise

$$J(X) = \|AX - B\|_F^2,$$

où $\|\cdot\|_F$ est la norme de Frobenius.

1. Montrez que X est solution du problème des moindres carrés ssi l'équation normale

$$A^T AX = A^T B \tag{1}$$

est vérifiée. Montrer ensuite que ce système admet toujours au moins une solution et qu'elle est unique si $\text{rang}(A) = n$.

2. Montrez comment utiliser judicieusement la décomposition `QR` de la matrice A pour résoudre le problème des moindre carrés dans le cas où $\text{rang}(A) = n$.

3. On considère m points successifs du plan $\{P_i\}_{i=1}^m$ de coordonnées (x_i, y_i) et de paramètres t_i. Montrez comment trouver n *points de contrôle* $\{P_j^*\}_{j=1}^n$ de paramètres préalablement définis T_j tel que la courbe paramétrée définie par

$$U(t) = \sum_{j=1}^{n} B_j(t) P_j^*, \quad 0 \leq t \leq 1,$$

approxime au mieux ces m points au sens des moindres carrés. Les fonctions $B_i(t)$ sont des B-splines d'ordre 3.

Rappel sur les B-splines

Étant donné n nœuds de paramètres $T_1 \leq T_2 \leq ... \leq T_n$, les B-splines de degré p sont les $n - p - 1$ fonctions $B_i^p(t)$ définies par la relation de récurrence

$$B_i^p(t) = \frac{t - T_i}{T_{i+p} - T_i} B_i^{p-1}(t) + \frac{T_{i+1+p} - t}{T_{i+1+p} - T_{i+1}} B_{i+1}^{p-1}(t),$$

avec

$$B_i^0(t) = \begin{cases} 1 & \text{si } t \in [T_i, T_{i+1}[, \\ 0 & \text{sinon.} \end{cases}$$

On peut engendrer des courbes paramétrées dans le plan en se servant de ces fonctions. En utilisant des B-splines de degrée p avec n nœuds de paramètre T_i, il est possible de combiner les $n - p - 1$ B-splines avec $n - p - 1$ points de contrôle (X_i, Y_i) pour créer une courbe paramétrique $U(t) = (x(t), y(t))$ de la manière suivante :

$$U(t) = \begin{cases} x(t) = \sum_{i=1}^{n-p-1} B_i(t) X_i \\ y(t) = \sum_{i=1}^{n-p-1} B_i(t) Y_i \end{cases}$$

Pour le devoir, vous allez générer vos propres suites de m points $P_i = (x_i, y_i)$. Il existe ensuite plusieurs manières d'assigner des paramètres t_i pour chacun de ces points. Nous vous proposons d'utiliser la paramétrisation cordale, qui se définit ainsi :

$$\begin{aligned} t_1 &= 0 \\ t_i &= t_{i-1} + \frac{\|P_i - P_{i-1}\|}{D} \ , \ i = 2, \dots, m-1 \\ t_m &= 1 \end{aligned}$$

où $D = \sum_{i=2}^{m} \|P_i - P_{i-1}\|$ est la somme totale de la corde.

Vous devez ensuite trouver n points de contrôle en utilisant des splines cubiques. Il faut alors définir le paramètre des $n+4$ noeuds de vos fonctions B-splines. Nous vous proposons le choix suivant :

$$\begin{aligned} T_1 = T_2 = T_3 = T_4 &= 0 \\ T_{j+3} &= (1-\alpha)t_{i-1} + \alpha t_i \ , \ j = 2, \dots, n-3 \\ \text{avec } i = \lfloor jd \rfloor, \quad \alpha &= jd - i, \quad d = \frac{m}{n-3} \\ T_{n+1} = T_{n+2} = T_{n+3} = T_{n+4} &= 1 \end{aligned}$$

Ce choix permet de bien repartir les points de contrôle en fonction des paramètres de vos points générés. Imposer que les quatre premières valeurs ainsi que quatre dernières valeurs soient identiques force la courbe à passer par le premier et dernier point de contrôle.
Il est possible d'obtenir de très bonnes approximations avec beaucoup moins de points de contrôle que de points donnés. Un exemple est montré à la Figure 1, où 532 points ont été approximés par une courbe en utilisant 40 points de contrôle.

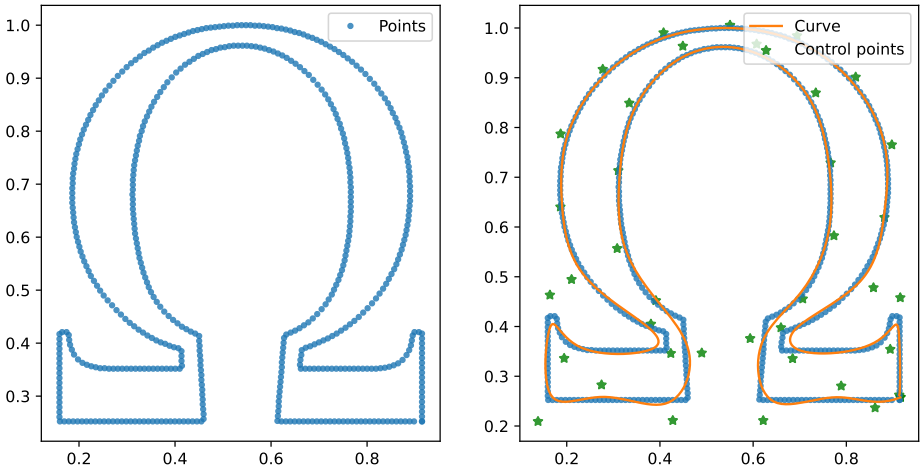

FIGURE 1 – Example.

Le devoir

Echéance

Le travail demandé est un travail individuel. Il consiste en (i) la remise d'un rapport et (ii) la remise de codes écrits en Python.

Les codes et le rapport sont à soumettre sur le `Gradescope` du cours d'Analyse Numérique pour le vendredi 23 février 2024 à 18:15. Nous vous enverrons la semaine prochaine les instructions pour soumettre votre devoir sur Gradescope.

Le code

La partie code contiendra les fichiers suivants :

- Un module Python `devoir1.py` contenant, *a minima*, les fonctions suivantes :
  — `qr(A)`, qui renvoie deux matrices `Q` et `R` correspondant à la décomposition QR réduite de `A`,
  — `lstsq(A, B)`, qui renvoie une matrice `X` correspondant à la solution du problème au moindres carrés (1).
  L'exactitude et la performance de ces fonctions seront testées par un *autograder*, il est donc très important que vous respectiez scrupuleusement ces spécifications. Vous aurez l'occasion de tester ce module via Gradescope sur un ensemble réduit de tests.
- Un ensemble de scripts Python permettant de reproduire les figures de votre rapport. Chaque figure doit pouvoir être reproduite par un script.

Les seules librairies externe admises sont les librairies `numpy` et `numba`. Il est interdit d'utiliser les fonctions du module `numpy.linalg`.

Toutes les implémentations seront soumises à un logiciel anti-plagiat.

Le rapport

Le rapport doit être réalisé avec LaTeX, avec la `documentclass article [11pt]` en `pagestyle plain`. Le .PDF et les sources .TEX de ce rapport (compressées dans un archive `.zip`) sont également à remettre sur `Gradescope`.

La longueur maximale du rapport est de 3 pages $A4$.

Le rapport devra contenir les éléments suivants

- La réponse aux 3 questions théoriques posées.
- Au moyen d'une expérience numérique que vous décrivez dans ce rapport, évaluez la complexité temporelle de factorisation `QR` appliquée à des matrices aléatoires de relativement grande taille ($m, n < 1000$).

- Illustrez graphiquement quelques résultats de l'approximation appliqués à des suites de points que vous génerez et ce dans différents cas de figure ($m = n$, $m > n$, $m >> n$).
- Bonus : Expliquez quelles conditions il faut appliquer aux points de contrôle afin que la courbe soit périodique. Illustrez ce cas avec des exemples.

Les sources des rapports seront soumises à un logiciel anti-plagiat.