Intro. à l'Analyse Numérique

TP n° 2

Nom :
Prénom :
Section :

- Vous pouvez faire ce travail en groupe. Veillez cependant à indiquer clairement dans votre projet le nom des participants. Si vous utilisez le langage `Python`, indiquez ces informations dons le fichier `setup.py`. Si vous utilisez `Ocaml`, indiquez dans chaque fichier, dans un commentaire, le noms des personnes ayant participé au projet.
- N'employez pas des spécificités propres à un système d'exploitation ou à un compilateur, n'envoyez pas de résidus de compilation, ni l'énoncé du TP,... Votre code doit être compilable et exécutable sur MacOS, Unix, et Win32 avec des outils libres (voir le site du cours). Plus spécifiquement, les programmes OCaml doivent être générés par l'exécution de `make` à la racine du projet. Les programmes peuvent porter des extensions selon le langage utilisé. Par exemple, l'exécutable pour le programme `p` peut être `p.py` ou `p.native`.
- Vous pouvez rédiger un rapport contenant les solutions aux questions de ce TP. Ce rapport doit être remis sur le site Moodle du cours. Veillez à ne pas remettre plus de dix pages de rapport.

Nous avons vu au cours comment résoudre les systèmes linéaires $Ax = b$ lorsque la matrice A est inversible. Comme nous l'avons mentionné, en pratique il est préférable d'utiliser la routine `DGESV`[1] de la librairie LAPACK qui a été développée avec soin par des professionnels.

Dans cette question, nous allons nous intéresser à la résolution de $Ax = b$ lorsque $A \in \mathbb{R}^{M\times N}$ avec $\operatorname{rang} A = N < M$. Dans ce cas, $Ax = b$ n'a pas forcément une solution pour tout b. Dans ce cas, nous allons rechercher la meilleure solution au sens des moindres carrés. Ceci consiste à chercher la valeur de x qui minimise la fonction

$$x \mapsto |Ax - b|_2^2 \tag{1}$$

où $|\cdot|_2$ désigne la norme Euclidienne.[2]

Question 1. Montrez que x est un point qui réalise le minimum de (1) si et seulement si x vérifie l'équation

$$A^\top Ax = A^\top b. \tag{2}$$

Question 2. Montrez que, sous les hypothèses sous lesquelles nous travaillons, (2) possède toujours une et une seule solution.

Question 3. Écrivez un programme `q1` qui prend sur la ligne de commande M et N suivi des nombres (séparés par des espaces) $a_{1,1}\ \cdots\ a_{1,N}\ a_{2,1}\ \cdots\ a_{2,N}\ \cdots\ a_{M,1}\ \cdots\ a_{M,N}\ b_1\ \cdots\ b_M$ où

[1] La façon dont cette routine est disponible dépend du langage utilisé. Veuillez vous référer à la documentation de `numpy.linalg` ou `Lacaml`.

[2] Pour rappel, si $x = (x_1, \dots, x_d) \in \mathbb{R}^d$, on définit $|x|_2 = \left(x_1^2 + \cdots + x_d^2\right)^{1/2}$.

Nom :
Prénom :
Section :

$A = (a_{i,j})$ et $b = (b_i)$, et retourne sur sa sortie standard la solution x de (2) sous la forme

$$\begin{matrix} x_1 \\ \vdots \\ x_N \end{matrix}$$

Le système linéaire soit être résolu grâce à la routine `DGESV`.

Supposons maintenant que les objets A et b varient dans le temps. Plus précisément, supposons qu'ils soient des fonctions $A : \mathbb{R} \to \mathbb{R}^{M\times N} : t \mapsto A(t)$ et $b : \mathbb{R} \to \mathbb{R}^M : t \mapsto b(t)$. Dans ce cas, en chaque t fixé, on peut résoudre (2) et obtenir la valeur de x au temps t. On a donc une fonction $x : \mathbb{R} \to \mathbb{R}^N : t \mapsto x(t)$.

Question 4. On désire calculer $\partial_t x(t)$. Donnez un algorithme qui, à partir des fonctions A et b va déterminer $\partial_t x(t)$. Justifiez en détail que celui-ci fonctionne.

Question 5. Dans un fichier `q2`, implémentez une fonction (ou méthode) `deriv` qui prend en arguments les fonctions A et b plus, si nécessaire, les dérivées de celles-ci, ainsi que t et retourne (une estimation de) la valeur de $\partial_t x(t)$. Les opérations matricielles doivent être implémentées en utilisant (efficacement) les routines de BLAS.[3]

Question 6. Écrivez un programme `q3` qui prend t comme seul argument sur la ligne de commande et retourne une approximation de $\partial_t x(t)$ pour les fonctions

$$A : \mathbb{R} \to \mathbb{R}^{M\times N} : t \mapsto A(t) = \begin{pmatrix} \sin t & t \\ t & e^t \\ \arctan t & t \end{pmatrix},$$
$$b : \mathbb{R} \to \mathbb{R}^N : t \mapsto b(t) = \begin{pmatrix} t \\ t^2 \\ 1/(1+t^2) \end{pmatrix}.$$

[3] De nouveau, ces routines sont accessibles différemment selon la langage utilisé. Voir la documentation de `numpy.linalg` ou `Lacaml`.