Université Savoie Mont Blanc, UFR SceM
VISI601, L3 CMI INFO, L3 MATH-INFO

4 mai 2023
Durée : 1h30

Examen, INFO601_CMI – Algorithmique numérique, Session 1

Documents autorisés : tous documents du cours/td/tp, notes manuscrites (nb : pas de livres)

Les exercices sont indépendants. Le barème est indicatif. Il dépasse volontairement 20 pour que vous ayez le choix dans les exercices.

Exercice 1. Problème bien posé, conditionnement (/4)

On définit la droite (D) comme l'ensemble des points $\mathbf{x}$ du plan qui satisfont $\mathbf{n} \cdot \mathbf{x} = a$, avec $\mathbf{n}$ un vecteur de longueur 1 (qui est orthogonal à la droite) et a un réel.

a [/1] Montrez que $|\mathbf{n} \cdot \mathbf{p} - a|$ est la distance du point $\mathbf{p}$ à la droite (D).

> Soit $\mathbf{q}$ le projeté orthogonal de $\mathbf{p}$ sur D. On sait que $\mathbf{n} \cdot (\mathbf{q} - \mathbf{p}) = \|\mathbf{n}\| \|\mathbf{q} - \mathbf{p}\| \cos \alpha$ avec α l'angle entre les 2 vecteurs. Mais $\mathbf{n}$ comme $\mathbf{q} - \mathbf{p}$ sont orthogonaux à (D) donc $\alpha = 0$ ou $\alpha = \pi$. Comme $\|\mathbf{n}\| = 1$, on arrive à $|\mathbf{n} \cdot (\mathbf{q} - \mathbf{p})| = \|\mathbf{q} - \mathbf{p}\|$.
> On conclut car $\mathbf{n} \cdot (\mathbf{q} - \mathbf{p}) = \mathbf{n} \cdot \mathbf{q} - a$ car $\mathbf{q} \in (D)$, et $\mathbf{q} - \mathbf{p}$ est bien la distance de $\mathbf{p}$ à (D).

b [/1] On s'intéresse au problème de déterminer la distance entre un point quelconque $\mathbf{p}$ et la droite (D) définie par $\mathbf{n}$ et a. Formaliser ce problème en écrivant sa résolvante $G(\mathbf{p}, \mathbf{n}, a)$. Quel est le domaine valide des données ? Est-ce que ce problème est bien posé sur ce domaine ?

> $$G(\mathbf{p}, \mathbf{n}, a) := |\mathbf{n} \cdot \mathbf{p} - a|$$
> le domaine est $\mathbb{R}^2 \times S^1 \times \mathbb{R}$ où est S^1 est le cercle de rayon 1. Le problème est bien posé car la solution existe toujours, est unique (de façon évidente), et varie continûment en fonction de $\mathbf{n}$, $\mathbf{p}$ et a (fonctions linéaires).

c [/2] Montrez que le conditionnement absolu est borné par une fonction ne dépendant que de $\|\mathbf{p}\|$ sauf lorque $\mathbf{p} \in (D)$ tandis que le conditionnement relatif peut exploser autour de la droite (D).

> On trouve, avec $\mathbf{d} = (\mathbf{p}, \mathbf{n}, a)$:
> $$K_{abs}(\mathrm{d}) = \|G'(\mathbf{d})\| = \sqrt{1 + \|\mathbf{n}\|^2 + \|\mathbf{p}\|^2} \leq \sqrt{2} + \|\mathbf{p}\|$$
> $$K_{rel}(\mathrm{d}) = \frac{\|G'(\mathbf{d})\| \|\mathbf{d}\|}{|G(\mathbf{d})|} = \frac{\sqrt{(2 + \|\mathbf{p}\|^2)(1 + a^2 + \|\mathbf{p}\|^2)}}{|\mathbf{n} \cdot \mathbf{p} - a|} \geq \frac{\sqrt{2}}{|\mathbf{n} \cdot \mathbf{p} - a|}.$$
> Si $\mathbf{p}$ se rapproche de (D), le dénominateur tend vers 0 alors que le numérateur ne descend jamais en dessous de $\sqrt{2}$.

Exercice 2. Approximation de dérivées (/7)

On cherche à approcher les dérivées premières d'une fonction f, supposée suffisamment lisse (au moins C^3), en des points (x_i), pour i de 0 à n, espacés régulièrement d'un pas h. On connait les valeurs de f en ces points x_i, que l'on pourra noter f_i pour $f(x_i)$.

a [/2] Proposez une formule qui approche $f'(x)$ à l'ordre 2, i.e. erreur de l'ordre de $O(h^2)$, valide pour tous les points x_i internes, i.e. i de 1 à $n - 1$. Justifiez cette formule.

On peut utiliser les différences centrées :

$$f'(x_i) = \frac{f_{i+1} - f_i}{2h} + O(h^2),$$

Il suffit d'écrire les développement de Taylor de f en $x_i + h$ et $x_i - h$:

$$f(x_i + h) = f(x_i) + hf'(x_i) + \frac{h^2}{2} f''(x_i) + O(h^3) \tag{1}$$
$$f(x_i - h) = f(x_i) - hf'(x_i) + \frac{h^2}{2} f''(x_i) + O(h^3) \tag{2}$$

On écrit (1) - (2), ce qui donne (en remarquant que $x_{i+1} = x_i + h$ et $x_{i-1} = x_i - h$.

$$f_{i+1} - f_{i-1} = 2hf'(x_i) + O(h^3).$$

b [/2] En utilisant un développement de Taylor à l'ordre 3 en x_0 pour les pas h et $2h$, trouvez une formule qui approche $f'(x_0)$ à l'ordre 2, et qui utilise f_0, f_1, f_2 et h.

On trouve la formule suivante : :

$$f'(x_0) = \frac{-f_2 + 4f_1 - 3f_0}{2h} + O(h^2),$$

Il suffit d'écrire les développement de Taylor de f en $x_0 + 2h$ et $x_0 + h$:

$$f(x_0 + 2h) = f(x_0) + 2hf'(x_0) + 2h^2 f''(x_0) + O(h^3) \tag{3}$$
$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2} f''(x_0) + O(h^3) \tag{4}$$

On écrit $(3) - 4 \times (4)$, ce qui donne (en remarquant que $x_1 = x_0 + h$ et $x_2 = x_0 + 2h$.

$$f_2 - 4f_1 = -3f_0 - 2hf'(x_0) + O(h^3).$$

c [/1] Procédez similairement pour obtenir la formule pour approcher $f'(x_n)$ à l'ordre 2.

On trouve la formule suivante : :

$$f'(x_n) = \frac{f_{n-2} - 4f_{n-1} + 3f_n}{2h} + O(h^2),$$

Il suffit d'écrire les développement de Taylor de f en $x_n - 2h$ et $x_n - h$:

$$f(x_n - 2h) = f(x_n) - 2hf'(x_0) + 2h^2 f''(x_n) + O(h^3) \tag{5}$$
$$f(x_n - h) = f(x_n) - hf'(x_0) + \frac{h^2}{2} f''(x_n) + O(h^3) \tag{6}$$

On écrit $(3) - 4 \times (4)$, ce qui donne (en remarquant que $x_{n-1} = x_n - h$ et $x_{n-2} = x_n - 2h$.

$$f_{n-2} - 4f_{n-1} = -3f_n + 2hf'(x_n) + O(h^3).$$

d [/2] Si maintenant les valeurs de f en ces points (x_i) forment le vecteur $\mathbf{f} = \begin{bmatrix} f(x_0) & \cdots & f(x_n) \end{bmatrix}^T$, écrivez la matrice D d'ordre $n+1$ telle que $D\mathbf{f}$ est le vecteur approchant les dérivées de f en tous les points, d'ordre 2 partout.

$$D = \frac{1}{2h}\begin{bmatrix} -3 & 4 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 0 & 1 & 0 & \ddots & \ddots & \vdots \\ 0 & -1 & 0 & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & -1 & 0 & 1 & 0 \\ \vdots & \ddots & \ddots & 0 & -1 & 0 & 1 \\ 0 & \cdots & \cdots & 0 & 1 & -4 & 3 \end{bmatrix}$$

Exercice 3. Conditionnement d'un système linéaire (/5)

Soit e un nombre positif (que l'on prendra petit non nul). Soit $\mathbf{A}_e = \begin{bmatrix} 1 & -e \\ 1 & e \end{bmatrix}$. On rappelle que la norme de Frobenius $\|\mathbf{A}_e\|$ de la matrice $\mathbf{A}_e$ est simplement la racine carrée de la somme des carrés de tous les coefficients de $\mathbf{A}_e$.

— (/0,5) Calculez la norme $\|\mathbf{A}_e\|$. Que vaut-elle si $|e|$ est proche de 0 ?

$$\|\mathbf{A}_e\| = \sqrt{2+2e^2} \approx \sqrt{2}.$$

— (/1,5) Montrez que $\mathbf{A}_e$ est inversible et calculez son inverse.

$\mathbf{A}_e$ est inversible car son déterminant est différent de 0 :

$$\det(\mathbf{A}_e) = e + e = 2e \neq 0.$$

Son inverse est : $\mathbf{A}_e^{-1} = \frac{1}{2e}\begin{bmatrix} e & e \\ -1 & 1 \end{bmatrix}$.

— (/0,5) Calculez la norme $\|\mathbf{A}_e^{-1}\|$.

$$\|\mathbf{A}_e^{-1}\| = \frac{1}{2e}\sqrt{2+2e^2} \approx \frac{\sqrt{2}}{2e}.$$

— (/0,5) Quel est environ le conditionnement de $\mathbf{A}_e$?

Le conditionnement de $\mathbf{A}_e$ est $K_{\mathbf{A}_e} = \|\mathbf{A}_e\|\|\mathbf{A}_e^{-1}\| = \frac{1}{2e}(2+2e^2) \approx 1/e$.

— (/1,5) Quelle précision peut-on attendre sur la solution (x, y) du système

$$\begin{bmatrix} 1 & -0.000001 \\ 1 & 0,000001 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

en supposant une précision numérique $\mathfrak{u} = 10^{-16}$.

On sait que la précision sera de l'ordre de $K_{\mathbf{A}_e}\|\mathbf{b}\|\mathfrak{u}$. Donc ici, $e = 10^{-6}$, $\|\mathbf{b}\| = \sqrt{2}$, donc erreur de l'ordre de $10^6 \times \sqrt{2} \times 10^{-16} \approx 1.4e - 10$.

— (/0,5) Quelle erreur sur le résidu peut-on espérer avec un algorithme de type pivot de Gauss (PLU) ?

L'erreur sur le résidu $\|A_e\mathbf{x} - \mathbf{b}\|$ reste faible de l'ordre de $2\mathfrak{u}$.

Exercice 4. Trace d'une matrice creuse (/4)

Soit A une matrice creuse carrée de taille $n \times n$ en format CSR. Ecrivez la fonction Python `Trace(A)` qui calcule la trace la matrice A, c'est-à-dire la somme de ses éléments diagonaux, de façon le plus efficace possible selon vous.

Quelle est la complexité de cette fonction en fonction de n et/ou de n_{nz} le nombre de coefficients non nuls de A ?

```python
# Complexité en O(n log n). Si on balaie les indptr, la complexité est en O(
    n_{nz})
def trace( A ):
    n = rows( A )
    t = 0
    for i in range( n ):
        a = A.indptr[ i   ]
        b = A.indptr[ i+1 ]
        while a < b:
            m = (a+b)//2
            if A.indices[ m ] == i :
                t += A.data[ m ]
                break
            elif A.indices[ m ] < i :
                a = m
            else:
                b = m
    return t
```

Exercice 5. Soustraction de deux matrices CSR (/6)

Soit A et B deux matrices carrées au format CSR, de tailles $n \times n$. Ecrire une fonction `Soustraction` (A, B) qui retourne une nouvelle matrice au format CSR qui représente $A - B$. Attention certains coefficients peuvent s'annuler dans la soustraction.

Quelle est la complexité de `Soustraction`, en fonction de n et/ou n_A et n_B le nombre de coefficients non nuls de A et B respectivement.

```python
def Soustraction( A, B ):
    indptr  = []
    indices = []
    data    = []
    n       = rows( A )
    k       = 0
    for i in range( n ):
        indptr.append( k )
        ka = A.indptr[ i ]
        kb = B.indptr[ i ]
        while ( ka < A.indptr[ i+1 ] ) and ( kb < B.indptr[ i+1 ] ):
            if ( A.indices[ ka ] < B.indices[ kb ] ) :
                indices.append( A.indices[ ka ] )
                data.append( A.data[ ka ] )
                ka += 1
                k  += 1
            elif ( A.indices[ ka ] > B.indices[ kb ] ) :
                indices.append( B.indices[ kb ] )
                data.append( -B.data[ kb ] )
                kb += 1
                k  += 1
            else: # equality
                v   = A.data[ ka ] - B.data[ kb ]
                if ( v != 0 ):
                    indices.append( A.indices[ ka ] )
                    data.append( v )
                    k += 1
                ka += 1
                kb += 1
        while ( ka < A.indptr[ i+1 ] ):
            indices.append( A.indices[ ka ] )
            data.append( A.data[ ka ] )
            ka += 1
            k  += 1
        while ( kb < B.indptr[ i+1 ] ):
            indices.append( B.indices[ kb ] )
            data.append( -B.data[ kb ] )
            kb += 1
            k  += 1
    indptr.append( k )
    return sp.sparse.csr_matrix((data, indices, indptr), shape = (n,n) )
```

Rappels et notations matrices creuses

On rappelle que les matrices creuses en mode CSR (compressed sparse row) sont représentés à l'aide des informations suivantes (A est une matrice creuse CSR) :

- `A.rows` et `A.cols` donnent le nombre de lignes et colonnes de A,
- `A.data` est le tableau contenant les n_{nz} coefficients non nuls de A,
- `A.indices` est le tableau des colonnes de chaque coefficient non nul de A,
- `A.indptr` est le tableau stockant le numéro du premier coefficient sur chaque ligne,
- `x[i]` retourne la i-ème valeur du vecteur $\mathbf{x}$ (i.e. x_i),
- `x.size()` retourne la taille du vecteur $\mathbf{x}$ (i.e. m),
- `vector( n )` crée et retourne un vecteur de taille n, rempli de zéros.
- `ndarray( (m,n))` crée et retourne une matrice pleine de taille $m \times n$
- `coo_matrix( (data, (rows,cols)), shape=(m,n))` crée et retourne une matrice COO de taille $m \times n$, remplie avec les coefficients `data[k]`, à la ligne `rows[k]` et la colonne `cols[k]`.
- `csr_matrix( (m,n))` crée et retourne une matrice CSR de taille $m \times n$, vide.

— `csr_matrix( (data, indices, indptr), shape=(m,n))` crée et retourne une matrice CSR de taille $m \times n$, remplie exactement avec les tableaux donnés correspondant au format.

— `A.tocoo()` retourne une copie de A mais dans le format COO

— `A.tocsr()` retourne une copie de A mais dans le format CSR