4M053 : Grands systèmes linéaires
C.Guichard, X.Claeys, P.Marchand & B.Thierry

Projet

Il est possible de réaliser le projet en binôme. Celui-ci devra être rendu, au plus tard le dimanche 13 mai 2017 à minuit, par email à xavier.claeys@ann.jussieu.fr, pierre.marchand@upmc.fr et thierry@ljll.math.upmc.fr

Le projet sera envoyé sous forme d'archive .zip contenant les éléments suivant :
- les codes sources du projet,
- un fichier readme résumant le contenu des différents fichiers, et la manière de s'en servir,
- un fichier makefile permettant de compiler le projet.

Il n'est pas nécessaire de rédiger de réponse aux questions théoriques qui sont posées comme un guide. Il sera attendu à l'oral que les réponses, et surtout les démonstrations correspondant aux réponses à ces questions soient connues.

Vous êtes invités à réutiliser le travail déjà effectué au cours du semestre en TP. Par ailleurs, on attend que vous mettiez en place des tests de votre propre initiative pour vous assurer du bon fonctionnement de votre code et de sa conformité au cahier des charges imposé. Vous serez interrogés sur les tests mis en place.

1 Matrices denses et creuses

Question 1
En reprenant le travail effectué en TP, écrivez une classe nommée `DenseMatrix` modélisant les matrices denses à coefficients réel (dont les entréres seront donc des `double`). Cette classe sera pourvue
- d'une surcharge de l'opérateur << de flux de sortie permettant un affichage dans le terminal de la matrice
- une fonction admettant pour entête `void Load(char* const filename)` et aura pour effet de lire le fichier nommé `filename` pour récupérér les valeurs des coefficients, et mettra à jour la matrice en conséquence. Le format attendu pour ce fichier est décrit en annexe.

Question 2
En reprenant le travail effectué en TP, écrivez une classe nommée `SparseMatrix` modélisant les matrices creuses stockées au format Coordonnées ou CSR. Cette classe comprendra
- une surcharge de l'opérateur << de flux de sortie permettant un affichage dans le terminal de la matrice
- une fonction membre nommée `MVProd` réalisant les produits matrice-vecteurs rapides, en complexité quasi-linéaire.

2 Perturbation de rang faible

Dans cette partie du projet on s'intéresse à modéliser les matrices $\mathrm{A} \in \mathbb{R}^{n\times n}$ qui sont la somme d'une matrice creuse et d'une matrice de rang faible, c'est-à-dire prenant la forme

$$\mathrm{A} = \mathrm{D} + \sum_{j=1}^{r} s_j \boldsymbol{u}_j \boldsymbol{v}_j^\top \tag{1}$$

où $s_j \in \mathbb{R}$, et $\boldsymbol{u}_j, \boldsymbol{v}_j \in \mathbb{R}^n$ sont des vecteurs et $\mathrm{D} \in \mathbb{R}^{n\times n}$ est une matrice creuse. On est particulièrement intéressé à la situation $r \ll n$ et même $r = \mathcal{O}(1)$. Ces matrices s'apparentent à des versions en dimension finie d'opérateurs de type Fredholm. Nous les appellerons donc *matrices de Fredholm*. Dans toute la suite, on supposera que les matrices creuses ont $\mathcal{O}(1)$ termes par ligne.

Question 3
Implémentez une classe appelée `FredholmMatrix` modélisant les matrices de Fredholm. Cette classe répondra au cahier des charges suivant en terme de données et fonctions membres.

Données membres

- un `int` noté `size` représentant le nombre n de lignes et de colonnes de la matrice
- une `SparseMatrix` noté `diag` représentant la contribution D dans (1)
- un `vector<double>` noté `sigma` tel que `sigma[j]` représente s_j
- deux `vector< vector<double> >` notés `lsv` et `rsv` représentant les deux collections de vecteur $\{\boldsymbol{u}_j\}_{j=1}^r$ et $\{\boldsymbol{v}_j\}_{j=1}^r$ dans (1)

Fonctions membres

- Un constructeur ayant pour entête

```
1    FredholmMatrix(const int& n0)
```

  initialisant `size` à `n0`. La partie creuse D ainsi que k seront considérés nuls.
- Un constructeur par recopie, et un opérateur d'affectation par recopie.
- Une fonction membre permettant d'ajouter une contribution dans la partie creuse de la matrice. Elle admettra l'entête

```
1    void Insert(const int& j, const int& k, const double& val)
```

  et aura un effet équivalent à $\mathrm{D}_{j,k} = \mathrm{D}_{j,k} + \mathrm{val}$.
- Une fonction membre pour ajouter une contribution de rang faible. Cette fonction admettra pour entête

```
1    void Insert(const double& s, const vector<double>& u,
2                const vector<double>& v)
```

  et aura pour effet de stocker `s`, `u` et `v` dans les vecteurs `sigma`, `lsv` et `rsv`. Ceci incrémentera donc au passage la taille de ces vecteurs de 1.
- Une surcharge de l'opérateur d'accès `( , )` prenant deux `const int&` en argument d'entrée de sorte que si `A` est une instance de `FredholmMatrix`, alors l'appel `A(j,k)` renvoie la valeur du coefficient de A situé à la j^{eme} ligne et la k^{eme} colonne.

Fonctions amies

- L'opérateur de flux de sortie admettant pour entête

```
1     friend ostream& operator<<(ostream& os, const FredholmMatrix& m)
```

- Une fonction modélisant le produit matrice-vecteur et admettant pour entête

```
1     friend void MVProd(vector<double>& b, const FredholmMatrix& A,
2                        const vector<double>& x)
```

permettant d'écrire dans `b` le résultat du produit matrice-vecteur de `A` par `x`. On contrôlera que la taille `x` est adaptée à cette opération, et on redimensionnera `b` le cas échéant.

- Une fonction modélisant la résolution d'un système linéaire carré et admettant pour entête

```
1     void Solve(const vector<double>& b, const FredholmMatrix& A,
2                vector<double>& x)
```

La fonction résoudra l'équation $\mathrm{A}\boldsymbol{x} = \boldsymbol{b}$.

3 Decomposition en valeurs singulières

Dans cette partie on considère des matrices dont les valeurs singulières décroissent rapidement, et on souhaite étudier comment obtenir une approximation de rang faible de ces matrices.

Question 4
Dans les classes `FredholmMatrix` et `DenseMatrix`, ajoutez une fonction amie appelée `FrobeniusNorm` permettant de calculer la norme de Frobenius de la matrice.

Question 5
En effectuant un appel à une bibliothèque externe de votre choix (la Gnu Scientific Library (GSL) par example), ajoutez une fonction membre dans la classe `FredholmMatrix` admettant pour entête

```
1   void SVD(const DenseMatrix& B, const int& r)
```

et qui calcule la SVD tronquée à l'ordre r de la matrice B. Le résultat de cette SVD tronquée sera stockée dans l'instance qui appelle la fonction membre `SVD`. Plus précisément, l'instruction `A.SVD(B,r)` aura pour effet que, dans la décomposition (1) de A :

— On impose $\mathrm{D} = 0$
— Les s_j sont les r plus grandes valeurs singulières de B
— $\boldsymbol{u}_j$ (resp. $\boldsymbol{v}_j$) les vecteurs singuliers à gauche (resp. à droite) correspondants.

Question 6
Étant donné $N \geq 0$, on considère la matrice $\mathrm{B} = (\mathrm{B}_{j,k}) \in \mathbb{R}^{N\times N}$ définie par

$$\mathrm{B}_{j,k} = \exp(-(j-k)^2/N^2)$$

En vous aidant d'un logiciel de tracé comme Gnuplot ou Matplotlib, représentez graphiquement la courbe donnant $\log(\|\mathrm{B} - \mathrm{B}_k\|_{\mathrm{F}})$ en fonction de $|\log(k)|$ sur $k = 1 \ldots N$. On prendra une valeur de N la plus élevée possible, et on se contentera de calculer l'erreur pour 10 valeurs de k bien choisies.

Question 7
On cherche maintenant un moyen plus rapide de calculer une SVD tronquée de manière approchée. On se propose de mettre en oeuvre l'algorithme d'approximation en croix dont le principe est décrit ci-dessous.

Étant donné une matrice $\mathrm{B} \in \mathbb{R}^{N\times N}$, notons $\mathrm{B}(:,k) \in \mathbb{R}^{N\times 1}$ la k-ieme colonne de B, et $\mathrm{B}(j,:) \in \mathbb{R}^{1\times N}$ la j-ième ligne de A. On confondra dans la suite $\mathbb{R}^{N\times 1}$ avec $\mathbb{R}^N$. Pour une suite de nombres positifs $r_j, j = 1\dots N$ on notera par ailleurs $(p,q) = \mathrm{argmax}_{j,k=1\dots N}(r_{j,k})$ le couple (p,q) qui réalise le maximum $r_{p,q} = \max_{j,k=1\dots N}(r_{j,k})$. On construit une approximation de B sous la forme $\mathrm{A}_r = \sum_{p=1}^{r} s_p \boldsymbol{u}_p \boldsymbol{v}_p^\top$ (qui se met donc au format (1) avec $\mathrm{D} = 0$).

Algorithme 1 Algorithme d'approximation en croix

R=B
$j_\star = k_\star = 0$
for $p = 1\dots r$ **do**
  $(j_\star, k_\star) = \mathrm{argmax}_{j,k=1\dots N}|\mathrm{R}(j,k)|$
  $\boldsymbol{u}_p = \mathrm{R}(:,k_\star)$
  $\boldsymbol{v}_p = \mathrm{R}(j_\star,:)^\top$
  $s_p = 1/\mathrm{R}(j_\star,k_\star)$
  $\mathrm{R} = \mathrm{R} - s_p \boldsymbol{u}_p \boldsymbol{v}_p^\top$
end for

Ajoutez une fonction membre dans la classe `FredholmMatrix` qui calcule la matrice A_r selon l'algorithme 1 et qui admet pour entête :

```
1   void CrossAppox(const DenseMatrix& B, const int& r)
```

Question 8
Repenez la question 5, et représentez graphiquement la courbe donnant $\log(\|\mathrm{B} - \widetilde{\mathrm{B}}_k\|_\mathrm{F})$ en fonction de $|\log(k)|$ sur $k = 1\dots N$, où $\widetilde{\mathrm{B}}_k$ a cette fois été construit à l'aide de l'algorithme d'approximation en croix.

Annexe : format fichier de matrice

Voici le format du fichier que l'on demande de considérer pour l'implémentation de la fonction membre `load` dans la classe `Mat` :

```
#TAILLE
nr nc
#DONNEES
a_{1,1} a_{1,2} a_{1,3} ... a_{1,nc} a_{2,1} a_{2,2} ... a_{2,nc} ... ... a_{nr,1} ... a_{nr,nc}
```

Donnons un exemple. Dans le cas où la matrice à charger depuis un fichier est la matrice à trois lignes et deux colonnes suivante

$$\mathrm{A} = \begin{bmatrix} 2.7 & 1.5 \\ 0.8 & 2.6 \\ 4.7 & 3.9 \end{bmatrix}$$

le fichier de matrice correspondant s'écrira :

```
#TAILLE
3 2
#DONNEES
2.7 1.5 0.8 2.6 4.7 3.9
```