DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MATHÉMATIQUES DE L'APPRENTISSAGE PROFOND
DEVOIR 2

31 octobre 2023

Directives : *Ce devoir doit être complété pour mardi le 21 novembre. Il doit être déposé sur le site Moodle du cours en un seul fichier, en format pdf. Vous devez utiliser le langage Julia.*

Modules Julia

Pour ce devoir, vous aurez besoin des modules `Plots`, `LinearAlgebra`, `MatrixDepot`, `IterativeSolvers`, `Images` et `LinearMaps`

Fonctions Julia

Vous aurez besoin de la fonction `arnoldi.jl`, qui contient une implémentation de la méthode d'Arnoldi. Elle est disponible sur le site du cours. Pour y accéder, utiliser la commande

```
include("arnoldi.jl");
```

Pour l'appeler, utiliser la commande

```
Q,H = arnoldi(A,u,m)
```

Avec ce que nous avons vu en classe, et avec les commentaires en début de fonction, l'utilisation de cette fonction ne devrait pas vous causer de problèmes.

1. On veut comparer la base de Krylov avec la base calculée par la méthode d'Arnoldi. Nous allons travailler avec le système linéaire suivant :

   ```
   lambda = @. 10 + (1:100)
   A = triu(rand(100,100),1) + diagm(0=>lambda)
   b = rand(100);
   ```

   A est une matrice triangulaire supérieure de dimensions $n \times n = 100 \times 100$.

   Comme nous avons vu en classe, nous ne voulons pas utiliser la base de Krylov complète. Nous voulons plutôt travailler avec un sous-espace de dimension $m \ll n$.

   (a) Construire les matrices de Krylov K_m pour $m = 1, \ldots, 30$. En fait, vous n'avez qu'à construire K_{30}, puis vous n'utiliserez que les colonnes voulues. Puisqu'il peut y avoir de grandes variations dans l'ordre de grandeur des termes $A^k \boldsymbol{b}$, normalisez chaque colonne calculée afin d'éviter les instabilités numériques.

   (b) On veut ensuite exprimer la solution de $A\boldsymbol{x} = \boldsymbol{b}$ dans les bases formées par les colonnes des K_m. On doit donc approcher $\boldsymbol{x}$ au sens des moindres carrés :

$$\min_{\boldsymbol{x} \in \boldsymbol{C}(K_m)} \|A\boldsymbol{x} - \boldsymbol{b}\|^2 = \min_{\boldsymbol{z} \in \mathbb{R}^m} \|(AK_m)\boldsymbol{z} - \boldsymbol{b}\|^2.$$

Faites la résolution au sens des moindres carrés de ce problème pour $\boldsymbol{x} \in \boldsymbol{C}(K_m)$, où $m = 1, \ldots, 30$. L'opérateur « \ » (la barre oblique inversée) de Julia bascule vers la résolution au sens des moindres carrés pour les matrices rectangulaires. Calculer la norme du résidu $\boldsymbol{b} - A\boldsymbol{x}$ pour chaque m, que vous illustrerez dans un graphique à l'aide de la commande `plot`, dans un repère semi-logarithmique. Qu'observez-vous ?

(c) Reprendre la sous-question précédente, mais en utilisant la base orthonormale donnée par la fonction du fichier `arnoldi.jl`. On se rendra à une base de dimension $m = 60$. Qu'observez-vous ?

2. Pour de grandes valeurs de n, le coût de la méthode GMRES en calculs et en espace mémoire utilisé devient prohibitif. Dans ce cas, on peut utiliser la version « redémarrée » (« *restarted* ») de la méthode, GMRES(k). Après k étapes de la méthode GMRES, l'algorithme est redémarré en utilisant $\boldsymbol{x}_k$ comme approximation initiale.

   (a) Comparer le nombre d'opérations (on veut un ordre de grandeur) et l'utilisation de la mémoire des méthodes GMRES et GMRES(k) pour k fixé, pour des valeurs de n de plus en plus grandes.

   (b) On veut expérimenter avec l'utilisation de la version redémarrée de la méthode GMRES. On va d'abord chercher une matrice provenant d'une discrétisation de l'équation de Poisson chez « Matrix Depot », puis on construit le système linéaire associé :

```
d = 50;
A = d^2*matrixdepot("poisson",d);
n = size(A,1);
b = ones(n);
```

   Si vous éprouvez des problèmes avec « Matrix Depot », la fonction `poisson(d)`, disponible sur le site du cours, pourra être utilisée. Vous pourriez être intéressés de voir le profil de la matrice avec la commande `spy(A)`.

   On veut comparer la méthode GMRES avec la méthode redémarrée GMRES(k) pour $k = 20, 40, 60$ et 120 (le nombre maximal d'itérations qu'on acceptera). On utilisera la commande

```
x,hist = gmres(A,b,reltol=1e-8,restart=k,maxiter=120,log=true,verbose=true);
```

   Faire un graphe des itérées dans un repère semi-logarithmique. Commenter ce que vous observez.

3. Nous allons finalement utiliser une méthode itérative, la méthode GMRES dans notre cas, pour corriger une image floue. Le floutage est un phénomène qui est souvent linéaire. Pour une image nette, associée à une matrice X, un modèle pour la représentation d'une image floue peut être exprimé comme

$$Z = B^k X C^k,$$

où Z est la matrice associée à l'image floue. Les matrices B et C sont de dimensions compatibles avec les dimensions de X et Z. Plus la puissance de k est élevée, plus l'image sera floue.

En pratique, on a la matrice Z et on cherche la matrice X. On veut exprimer le problème sous la forme d'un système d'équations algébriques linéaires

$$A\boldsymbol{x} = \boldsymbol{z},$$

où $\boldsymbol{z} = \mathrm{vec}(Z)$ et $\boldsymbol{x} = \mathrm{vec}(X)$. Donc si Z et X sont de dimensions $m \times n$, $\boldsymbol{z}$ et $\boldsymbol{x}$ sont de longueur mn. La matrice A, de dimensions $mn \times mn$, est associée au modèle de floutage choisi. La méthode itérative sera appliquée à ce système linéaire. Elle nous permettra de calculer une approximation $\tilde{X}$ de X.

Nous allons d'abord lire le fichier « MandrillZ.png » que vous trouverez sur le site Moodle du cours, pour en extraire la matrice associée, à l'aide des commandes :

```
img = load("MandrillZ.png")
Z = @. Float64(Gray(img));
```

Vous remarquerez que cette image est floue.

Nous allons considérer les modèles basés sur les matrices

$$b_{ij} = c_{ij} = \begin{cases} \frac{1}{2} & \text{si } i = j\,; \\ \frac{1}{4} & \text{si } |i-j| = 1\,; \\ 0 & \text{sinon} \end{cases} \qquad \text{et} \qquad b_{ij} = c_{ij} = \begin{cases} \frac{5}{8} & \text{si } i = j\,; \\ \frac{1}{8} & \text{si } |i-j| = 1\,; \\ \frac{1}{16} & \text{si } |i-j| = 2\,; \\ 0 & \text{sinon.} \end{cases}$$

La commande `diagm` nous permet de créer facilement ce type de matrices où on a, par exemple,

```
B = diagm(0=>fill(0.5,m),1=>fill(0.25,m-1),-1=>fill(0.25,m-1))
```

L'opération $\boldsymbol{u} = \mathrm{vec}(U)$ est donnée directement par la commande `u = vec(U)` avec Julia. L'opération inverse, où on passe du vecteur à la matrice, peut être exécutée avec la commande `U = reshape(u,m,n)`.

Finalement, puisque nous ne connaissons pas explicitement la matrice A, nous allons passer par le résultat de l'action des opérations de floutage sur le vecteur $\boldsymbol{x}$. On peut définir cette application linéaire à l'aide d'une commande de la forme

```
T = LinearMap(x -> vec(B^k*reshape(x,m,n)*B^k),m*n);
```

Nous sommes ensuite en mesure d'utiliser la méthode GMRES pour calculer une approximation de la solution du système linéaire $A\boldsymbol{x} = \boldsymbol{z}$:

```
y,hist = gmres(T,vec(Z),reltol=1e-5,restart=k,maxiter=120,log=true,verbose=true)
```

Il faut aussi s'assurer que les valeurs calculées se trouvent bien entre 0 et 1.

(a) Utiliser les modèles proposés pour déflouter l'image du fichier `MandrillZ.png`.

(b) Utiliser une norme matricielle pour comparer l'image défloutée et l'image d'origine du fichier `MandrillX.png`. Afficher aussi l'image calculée, ainsi que les images des fichiers `MandrillZ.png` et `MandrillX.png`.

Steven Dufour