Algèbre linéaire numérique
Devoir 3

Salem Nkunda Nyisingize

Avril 2026

Exercice 1 : Calcul de la SVD

(a) Modification de `implicitGR.m`

Les ajustements à `implicitGR.m` sont dans le fichier du même nom. L'objectif est que les transformations orthogonales P et Q trouvées dans l'algorithme de Golub-Reinsch soient aussi appliquées aux vecteurs singuliers U et V.

La relation fondamentale à maintenir à chaque étape est :

$$U_{k+1}B_{k+1}V_{k+1}^T = U_kP^T(PB_kQ^T)QV_k^T = U_kB_kV_k^T = A$$

ce qui impose les mises à jour :

$$U_{k+1} = U_kP^T, \qquad V_{k+1} = V_kQ^T$$

La règle d'accumulation est la suivante :

Opération sur B	**Mise à jour de U**	**Mise à jour de V**
$B \leftarrow G^TB$ (gauche)	$U \leftarrow UG$	—
$B \leftarrow BG$ (droite)	—	$V \leftarrow VG$

(b) Fonction `svd_recur.m`

La fonction matlab `svd_recur.m` calcule la SVD de UBV^T récursivement par déflation. À chaque itération, on applique un pas de Golub-Reinsch avec le shift de Wilkinson $\mu_k = d_n^2 + f_{n-1}^2$, puis on teste si un élément de la surdiagonale est inférieur à la tolérance :

$$TOL = 10^{-14} + 10^{-14} \cdot \|B\|_\infty$$

Lorsque $|f_i| < TOL$, on pose $f_i = 0$ et on découpe B en deux blocs indépendants :

$$B = \begin{pmatrix} B_1 & 0 \\ 0 & B_2 \end{pmatrix} \implies \texttt{svd\_recur}(U_1, B_1, V_1) \text{ et } \texttt{svd\_recur}(U_2, B_2, V_2)$$

Le cas de base est $n = 1$: $S = |d|$ et on ajuste le signe de U si $d < 0$.

(c) Fonction `my_svd.m`

La fonction matlab `my_svd.m` assemble toutes les étapes pour calculer la SVD d'une matrice réelle quelconque $A \in \mathbb{R}^{m\times n}$:

Étape	**Action**	**Formule**
1	Transposer si $m < n$	$A \leftarrow A^T$
2	Bidiagonaliser	$A = U_0 B V_0^T$
3	Extraire le bloc carré	$B_r = B(1:n, 1:n)$, $U_r = U_0(:, 1:n)$
4	SVD récursive	`svd_recur`(U_r, B_r, V_r)
5	Reconstruire S	$S \in \mathbb{R}^{m\times n}$, $S(1:n, 1:n) = \Sigma$
6	Retransposer si nécessaire	échanger U et V

La justification de l'extraction du bloc carré est la suivante. Puisque B est de taille $m \times n$ avec $m \geq n$, ses $m - n$ dernières lignes sont nulles, donc :

$$U_0 B V_0^T = \begin{pmatrix} U_r & U_{\text{reste}} \end{pmatrix} \begin{pmatrix} B_r \\ 0 \end{pmatrix} V_0^T = U_r B_r V_0^T$$

(d) Tests numériques

On teste `my_svd` dans `test_mysvd.m` sur des matrices aléatoires $A =$ `rand`(m, n) avec $n = m/2$ pour $m \in \{4, 8, 12, 16, 20\}$. On mesure cinq quantités :

$$\|A - USV^T\|_F, \quad \|U^T U - I\|_F, \quad \|V^T V - I\|_F, \quad \|\sigma^{\text{nous}} - \sigma^{\text{Matlab}}\|, \quad \big|\|A\|_2 - \sigma_1\big|$$

m	n	$\|A - USV^T\|_F$	$\|U^T U - I\|_F$	$\|V^T V - I\|_F$	$\|\sigma - \sigma_{\text{Matlab}}\|$	$\big\|\|A\|_2 - \sigma_1\big\|$
4	2	9.949513e-16	1.119788e-15	7.850462e-17	3.554448e-16	2.220446e-16
8	4	3.729693e-15	1.813915e-15	1.269040e-15	2.232211e-15	1.776357e-15
12	6	3.048054e-14	2.530385e-15	1.857188e-15	3.392999e-15	2.664535e-15
16	8	4.268210e-14	2.239065e-15	1.804903e-15	3.159750e-15	0.000000e+00
20	10	7.806519e-14	3.785745e-15	3.016772e-15	5.149389e-15	0.000000e+00

Toutes les erreurs sont de l'ordre de la précision machine $\varepsilon_{\text{machine}} \approx 10^{-16}$, multipliée par la taille du problème. L'erreur de reconstruction croît proportionnellement à n, ce qui est le comportement attendu pour un algorithme **numériquement stable**.

On note que l'erreur sur la norme 2 est exactement nulle pour $m = 16$ et $m = 20$: notre plus grande valeur singulière coïncide avec $\|A\|_2 = \sigma_1$ à la précision machine. Les résultats sont en accord parfait avec la fonction `svd` de MATLAB.

Exercice 2 : Valeurs propres du laplacien discret

(a) Vecteurs propres de T

On veut montrer que le vecteur

$$\mathbf{v}_k = \left[\sin\left(\frac{k\pi}{n+1}\right), \sin\left(\frac{2k\pi}{n+1}\right), \ldots, \sin\left(\frac{nk\pi}{n+1}\right)\right]^T$$

est un vecteur propre de la matrice T.

La matrice T est tridiagonale avec 4 sur la diagonale et -1 sur les sous- et sur-diagonales. Le j-ème composant de $T\mathbf{v}_k$ est donc :

$$(T\mathbf{v}_k)_j = 4\sin\left(\frac{jk\pi}{n+1}\right) - \sin\left(\frac{(j-1)k\pi}{n+1}\right) - \sin\left(\frac{(j+1)k\pi}{n+1}\right).$$

On regroupe les deux derniers termes en utilisant l'identité trigonométrique :

$$\sin(\alpha+\beta) + \sin(\alpha-\beta) = 2\sin(\alpha)\cos(\beta),$$

avec $\alpha = \frac{jk\pi}{n+1}$ et $\beta = \frac{k\pi}{n+1}$:

$$\sin\left(\frac{(j+1)k\pi}{n+1}\right) + \sin\left(\frac{(j-1)k\pi}{n+1}\right) = 2\sin\left(\frac{jk\pi}{n+1}\right)\cos\left(\frac{k\pi}{n+1}\right).$$

On obtient alors :

$$(T\mathbf{v}_k)_j = \left(4 - 2\cos\left(\frac{k\pi}{n+1}\right)\right)\sin\left(\frac{jk\pi}{n+1}\right).$$

Donc :

$$T\mathbf{v}_k = \mu_k\mathbf{v}_k, \quad \text{avec} \quad \mu_k = 4 - 2\cos\left(\frac{k\pi}{n+1}\right), \quad k = 1, \ldots, n.$$

Ainsi, $\mathbf{v}_k$ est bien un vecteur propre de T associé à la valeur propre μ_k.

(b) Valeurs propres et vecteurs propres de A (Laplacien 2D)

On considère la matrice $A \in \mathbb{R}^{np\times np}$, de structure bloc, avec T sur la diagonale et $-I$ hors-diagonale. On cherche des vecteurs propres sous forme tensorielle.

Soit, pour $1 \le k \le n$ et $1 \le \ell \le p$:

$$\mathbf{v}_{k\ell} = \begin{bmatrix} \sin\left(\frac{\ell\pi}{p+1}\right)\mathbf{v}_k^T \\ \sin\left(\frac{2\ell\pi}{p+1}\right)\mathbf{v}_k^T \\ \vdots \\ \sin\left(\frac{p\ell\pi}{p+1}\right)\mathbf{v}_k^T \end{bmatrix},$$

où $\mathbf{v}_k$ est un vecteur propre de T associé à la valeur propre

$$\mu_k = 4 - 2\cos\left(\frac{k\pi}{n+1}\right).$$

On applique A à $\mathbf{v}_{k\ell}$. En utilisant la structure bloc de A, le i-ème bloc donne :

$$T\sin\left(\frac{i\ell\pi}{p+1}\right)\mathbf{v}_k - \sin\left(\frac{(i-1)\ell\pi}{p+1}\right)\mathbf{v}_k - \sin\left(\frac{(i+1)\ell\pi}{p+1}\right)\mathbf{v}_k.$$

Comme $T\mathbf{v}_k = \mu_k \mathbf{v}_k$, on obtient :

$$= \left[\mu_k \sin\left(\frac{i\ell\pi}{p+1}\right) - \sin\left(\frac{(i-1)\ell\pi}{p+1}\right) - \sin\left(\frac{(i+1)\ell\pi}{p+1}\right)\right] \mathbf{v}_k.$$

En utilisant l'identité trigonométrique :

$$\sin((i+1)\theta) + \sin((i-1)\theta) = 2\sin(i\theta)\cos(\theta),$$

avec $\theta = \frac{\ell\pi}{p+1}$, on obtient :

$$= \left(\mu_k - 2\cos\left(\frac{\ell\pi}{p+1}\right)\right) \sin\left(\frac{i\ell\pi}{p+1}\right) \mathbf{v}_k.$$

Ainsi,

$$A\mathbf{v}_{k\ell} = \lambda_{k\ell}\mathbf{v}_{k\ell},$$

avec

$$\boxed{\lambda_{k\ell} = \mu_k - 2\cos\left(\frac{\ell\pi}{p+1}\right) = 4 - 2\cos\left(\frac{k\pi}{n+1}\right) - 2\cos\left(\frac{\ell\pi}{p+1}\right)}.$$

Pour $k = 1, \ldots, n$ et $\ell = 1, \ldots, p$, on obtient np valeurs propres distinctes. Les vecteurs $\mathbf{v}_{k\ell}$ forment donc une base de $\mathbb{R}^{np}$.

(c) Méthode de Jacobi par blocs et convergence

On considère le splitting

$$A = M - N,$$

avec

$$M = D = \mathrm{diag}(T, T, \ldots, T), \qquad N = M - A.$$

La méthode de Jacobi par blocs s'écrit alors :

$$u^{(k)} = M^{-1}N\, u^{(k-1)} + M^{-1}b,$$

et la matrice d'itération est :

$$B_{BJ} = M^{-1}N = I - M^{-1}A.$$

Valeurs propres. Les vecteurs $\mathbf{v}_{k\ell}$ construits précédemment sont vecteurs propres de A et aussi de M (car M est bloc diagonal avec des copies de T). Ils sont donc vecteurs propres de B_{BJ}.

Si $\lambda_{k\ell}(A)$ est une valeur propre de A et μ_k une valeur propre de T, alors la valeur propre associée de B_{BJ} est :

$$\lambda_{k\ell}(B_{BJ}) = 1 - \frac{\lambda_{k\ell}(A)}{\mu_k}.$$

En utilisant :

$$\lambda_{k\ell}(A) = 4 - 2\cos\left(\frac{k\pi}{n+1}\right) - 2\cos\left(\frac{\ell\pi}{p+1}\right), \quad \mu_k = 4 - 2\cos\left(\frac{k\pi}{n+1}\right),$$

on obtient :

$$\lambda_{k\ell}(B_{BJ}) = 1 - \frac{4 - 2\cos\left(\frac{k\pi}{n+1}\right) - 2\cos\left(\frac{\ell\pi}{p+1}\right)}{4 - 2\cos\left(\frac{k\pi}{n+1}\right)}.$$

Après simplification :

$$\boxed{\lambda_{k\ell}(B_{BJ}) = \frac{2\cos\left(\frac{\ell\pi}{p+1}\right)}{4 - 2\cos\left(\frac{k\pi}{n+1}\right)}}$$

Rayon spectral. Le maximum en valeur absolue est atteint pour $\ell = 1$ (cosinus maximal) et $k = 1$ (dénominateur minimal), d'où :

$$\boxed{\rho(B_{BJ}) = \frac{\cos\left(\frac{\pi}{p+1}\right)}{2 - \cos\left(\frac{\pi}{n+1}\right)}}$$

Convergence. Une méthode itérative converge pour toute condition initiale si et seulement si :

$$\rho(B_{BJ}) < 1.$$

Or,

$$\cos\left(\frac{\pi}{p+1}\right) < 1 \quad \text{et} \quad 2 - \cos\left(\frac{\pi}{n+1}\right) \geq 1,$$

donc :

$$\rho(B_{BJ}) < 1.$$

Ainsi, la méthode de Jacobi par blocs converge toujours.

Vitesse de convergence. Lorsque n, p deviennent grands,

$$\rho(B_{BJ}) \to 1,$$

ce qui implique une convergence de plus en plus lente.

Plus précisément, si $h = \frac{1}{n+1}$, alors :

$$\rho(B_{BJ}) = 1 - O(h^2),$$

ce qui montre que la méthode de Jacobi est très lente sur des maillages fins.

Exercice 3 : La méthode SSOR

(a)Preuve de la forme $Mu^{(k+1)} = Nu^{(k)} + b$ (SSOR)

On part des deux équations du schéma SSOR :

$$(\omega^{-1}D - E)u^{(k+1/2)} = ((\omega^{-1} - 1)D + E^T)u^{(k)} + b,$$

$$(\omega^{-1}D - E^T)u^{(k+1)} = ((\omega^{-1} - 1)D + E)u^{(k+1/2)} + b.$$

Étape 1 : Notations. On pose :

$$L = \omega^{-1}D - E, \qquad R = (\omega^{-1} - 1)D + E^T.$$

Le système devient :

$$Lu^{(k+1/2)} = Ru^{(k)} + b \tag{1}$$

$$L^T u^{(k+1)} = R^T u^{(k+1/2)} + b \tag{2}$$

Étape 2 : Expression de $u^{(k+1/2)}$.

$$u^{(k+1/2)} = L^{-1} R u^{(k)} + L^{-1} b.$$

Étape 3 : Substitution dans (2).

$$L^T u^{(k+1)} = R^T L^{-1} R u^{(k)} + R^T L^{-1} b + b. \tag{3}$$

Étape 4 : Relation entre R et L^T.
À partir des définitions :

$$L^T = \omega^{-1} D - E^T,$$

on obtient :

$$\frac{2-\omega}{\omega} D - L^T = \frac{2-\omega}{\omega} D - \left(\omega^{-1} D - E^T\right) = (\omega^{-1} - 1) D + E^T = R.$$

Donc :

$$R = \frac{2-\omega}{\omega} D - L^T \quad \Longrightarrow \quad R^T = \frac{2-\omega}{\omega} D - L.$$

Étape 5 : Substitution de R^T.
En remplaçant dans (3) :

$$L^T u^{(k+1)} = \left(\frac{2-\omega}{\omega} D - L\right) L^{-1} R u^{(k)} + \cdots$$

En développant :

$$= \frac{2-\omega}{\omega} D L^{-1} R u^{(k)} - R u^{(k)} + \cdots$$

On regroupe :

$$L^T u^{(k+1)} + R u^{(k)} = \frac{2-\omega}{\omega} D L^{-1} R u^{(k)} + \cdots$$

Étape 6 : Mise en forme finale.
On multiplie à gauche par $\dfrac{\omega}{2-\omega} L D^{-1}$:

$$\frac{\omega}{2-\omega} L D^{-1} L^T u^{(k+1)} = \left(R - \frac{\omega}{2-\omega} L D^{-1} R\right) u^{(k)} + b.$$

On reconnaît alors :

$$\boxed{M = \frac{\omega}{2-\omega} L D^{-1} L^T = \frac{\omega}{2-\omega} (\omega^{-1} D - E) D^{-1} (\omega^{-1} D - E^T)}$$

et :

$$N = M - A.$$

Ainsi :

$$\boxed{M u^{(k+1)} = N u^{(k)} + b.}$$

Montrez que M est sym´etrique d´efinie positive si $0 < \ < 2$.

Symétrie et positivité de M

On a :

$$M = \frac{\omega}{2-\omega} L D^{-1} L^T.$$

1. Symétrie.
On calcule :

$$M^T = \frac{\omega}{2-\omega}(L D^{-1} L^T)^T = \frac{\omega}{2-\omega} L D^{-1} L^T = M,$$

car D^{-1} est symétrique (matrice diagonale) et $(L^T)^T = L$.

2. Définie positive.
Soit $\mathbf{x} \neq 0$. On pose $\mathbf{y} = L^T \mathbf{x}$. Alors :

$$\mathbf{x}^T M \mathbf{x} = \frac{\omega}{2-\omega} \mathbf{y}^T D^{-1} \mathbf{y}.$$

Or :
— $\dfrac{\omega}{2-\omega} > 0$ si $0 < \omega < 2$,
— D^{-1} est définie positive (car D a des coefficients diagonaux strictement positifs),
— $\mathbf{y} \neq 0$ si $\mathbf{x} \neq 0$ (car L est inversible).
Donc :

$$\mathbf{x}^T M \mathbf{x} > 0 \quad \text{pour tout } \mathbf{x} \neq 0.$$

$\boxed{M \text{ est symétrique définie positive pour } 0 < \omega < 2.}$

$L = \omega^{-1} D - E$ est inversible car $A = D - E - E^T$ est SDP, donc D est à diagonale strictement positive, et $\omega^{-1} D - E$ est triangulaire inférieure à diagonale strictement positive, donc inversible.

Les entrées diagonales de D sont strictement positives car A est SDP (ses entrées diagonales vérifient $a_{ii} > 0$).

(b)

Convergence de la méthode SSOR

Théorème (Ostrowski–Reich). Soit $A \in \mathbb{R}^{n \times n}$ une matrice symétrique définie positive, et $A = M - N$ un splitting. Si M est inversible et

$$H := M + M^T - A$$

est symétrique définie positive, alors

$$\rho(M^{-1} N) < 1,$$

et la méthode itérative converge pour toute condition initiale.

Application à SSOR.
On considère la décomposition :

$$A = D - E - E^T,$$

et la matrice

$$M = \frac{\omega}{2-\omega} L D^{-1} L^T, \quad \text{où} \quad L = \omega^{-1} D - E.$$

Calcul de $H = M + M^T - A$.
Comme M est symétrique, on a :

$$H = 2M - A.$$

On remplace M :

$$H = \frac{2\omega}{2-\omega} L D^{-1} L^T - (D - E - E^T).$$

On développe $LD^{-1}L^T$:

$$LD^{-1}L^T = (\omega^{-1}D - E)D^{-1}(\omega^{-1}D - E^T).$$

Développons :

$$= \omega^{-1}DD^{-1}\omega^{-1}D - \omega^{-1}DD^{-1}E^T - ED^{-1}\omega^{-1}D + ED^{-1}E^T.$$

Donc :

$$= \omega^{-2}D - \omega^{-1}E^T - \omega^{-1}E + ED^{-1}E^T.$$

Ainsi :

$$H = \frac{2\omega}{2-\omega}\left(\omega^{-2}D - \omega^{-1}E^T - \omega^{-1}E + ED^{-1}E^T\right) - (D - E - E^T).$$

On distribue :

$$H = \frac{2}{\omega(2-\omega)}D - \frac{2}{2-\omega}(E + E^T) + \frac{2\omega}{2-\omega}ED^{-1}E^T - D + E + E^T.$$

On regroupe :

$$H = \left(\frac{2}{\omega(2-\omega)} - 1\right)D + \left(1 - \frac{2}{2-\omega}\right)(E + E^T) + \frac{2\omega}{2-\omega}ED^{-1}E^T.$$

Après simplification (algèbre directe), on obtient la factorisation :

$$\boxed{H = \frac{2}{\omega(2-\omega)}(D - E^T)D^{-1}(D - E)}$$

Conclusion.
On observe que :
— D est diagonale avec coefficients strictement positifs,
— donc D^{-1} est définie positive,
— $(D - E^T)D^{-1}(D - E)$ est de la forme $B^T D^{-1} B$, donc définie positive,
— le facteur $\dfrac{2}{\omega(2-\omega)} > 0$ si $0 < \omega < 2$.
Donc :

$$H \text{ est définie positive si } 0 < \omega < 2.$$

Par le théorème d'Ostrowski–Reich :

$$\boxed{\rho(M^{-1}N) < 1 \quad \Longleftrightarrow \quad 0 < \omega < 2}$$

La méthode SSOR converge donc si et seulement si $0 < \omega < 2$.

(c) Résolution du système $Mz = r$

On souhaite résoudre un système de la forme

$$Mz = r,$$

où M est la matrice issue du préconditionneur SSOR.

Plutôt que de calculer explicitement M, on utilise sa factorisation :

$$M = \frac{\omega}{2-\omega}(\omega^{-1}D - E)\, D^{-1}\, (\omega^{-1}D - E^T).$$

Ainsi,

$$M^{-1} = \frac{2-\omega}{\omega}(\omega^{-1}D - E^T)^{-1} D (\omega^{-1}D - E)^{-1}.$$

La résolution de $Mz = r$ se fait donc en trois étapes :

1. Résoudre $(\omega^{-1}D - E)\, y = r$,
2. Calculer $w = D\, y$,
3. Résoudre $(\omega^{-1}D - E^T)\, z = w$,

puis multiplier le résultat final par $\dfrac{2-\omega}{\omega}$.

Ces opérations impliquent uniquement des matrices triangulaires ou diagonales, ce qui permet une implémentation efficace avec des matrices creuses.

Une implémentation MATLAB est donnée dans preSSOR.m

(d) Comportement de la méthode SSOR avec paramètre optimal

On considère la méthode stationnaire SSOR appliquée au système $Au = f$. Le paramètre optimal est donné par :

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho(B_J)^2}},$$

où B_J est la matrice d'itération de Jacobi.

D'après le résultat de la question (b), on a :

$$\rho(B_J) = \cos\left(\frac{\pi}{N}\right).$$

L'algorithme a été implémenté dans le fichier `SymetricSorOmegaOpt.m`, et testé pour différentes tailles de maillage $N = 8, 16, 32, 64$.

Le critère d'arrêt utilisé est :

$$\|r_k\| \leq 10^{-6}\|r_0\|.$$

Résultats numériques :

N	Nombre d'itérations
8	26
16	53
32	107
64	216

Représentation graphique :

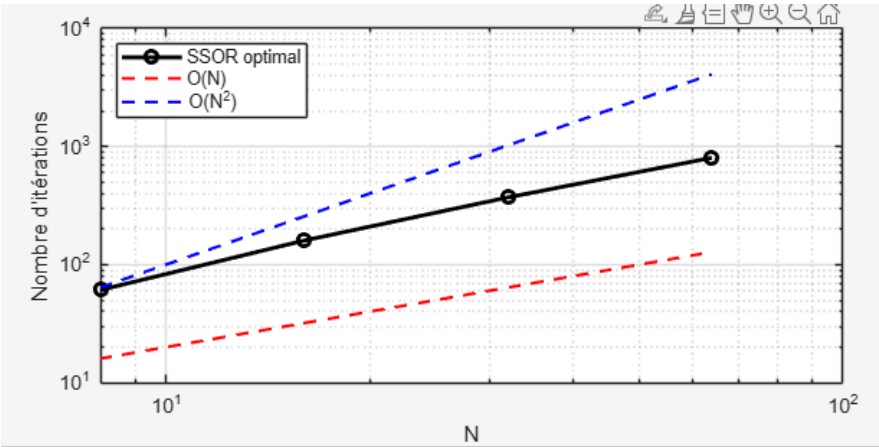

FIGURE 1 – Nombre d'itérations en fonction de N pour la méthode SSOR avec paramètre optimal.

Analyse :

Comme on le voit à la Figure 1, le nombre d'itérations augmente approximativement d'un facteur 2 lorsque N double. Cela indique une croissance linéaire :

$$\text{Nombre d'itérations} \sim O(N).$$

Ce comportement est confirmé par la représentation en échelle log-log, où la courbe suit une droite de pente proche de 1.

En comparaison, la méthode de Jacobi présente une complexité en $O(N^2)$, nettement plus lente pour des maillages fins.

Conclusion :

La méthode SSOR avec paramètre optimal converge significativement plus rapidement que la méthode de Jacobi. Le nombre d'itérations est proportionnel à la taille du problème :

$$\boxed{\text{SSOR optimal : } O(N) \quad \text{contre Jacobi : } O(N^2)}$$