Correction du TD : Calcul des Valeurs Propres

Correction de l'Exercice 1

1. **Démonstration par récurrence :** Nous voulons montrer que pour tout entier $n \geq 1$, $x_n = \frac{A^n x_0}{\|A^n x_0\|}$.
   — *Initialisation :* Pour $n = 1$, l'algorithme donne directement $x_1 = \frac{Ax_0}{\|Ax_0\|}$. La propriété est vraie.
   — *Hérédité :* Supposons la propriété vraie pour un certain entier $n \geq 1$. Calculons x_{n+1} :

$$x_{n+1} = \frac{Ax_n}{\|Ax_n\|} = \frac{A\left(\frac{A^n x_0}{\|A^n x_0\|}\right)}{\left\|A\left(\frac{A^n x_0}{\|A^n x_0\|}\right)\right\|}$$

   Par linéarité de l'application matricielle A et par les propriétés de la norme, on peut sortir le scalaire positif $\frac{1}{\|A^n x_0\|}$ au numérateur et au dénominateur :

$$x_{n+1} = \frac{\frac{1}{\|A^n x_0\|} A^{n+1} x_0}{\frac{1}{\|A^n x_0\|}\|A^{n+1} x_0\|} = \frac{A^{n+1} x_0}{\|A^{n+1} x_0\|}$$

   L'hérédité est donc prouvée.

2. **Condition de convergence :** Puisque A est diagonalisable, il existe une base de vecteurs propres $(u_1, \ldots, u_N)$ associés aux valeurs propres $(\lambda_1, \ldots, \lambda_N)$. Décomposons le vecteur initial x_0 dans cette base :

$$x_0 = \sum_{i=1}^{N} \alpha_i u_i = \alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_N u_N$$

   En appliquant A^n, on obtient :

$$A^n x_0 = \sum_{i=1}^{N} \alpha_i \lambda_i^n u_i = \lambda_1^n \left(\alpha_1 u_1 + \sum_{i=2}^{N} \alpha_i \left(\frac{\lambda_i}{\lambda_1}\right)^n u_i \right)$$

   Puisque λ_1 est la valeur propre strictement dominante, on a $\left|\frac{\lambda_i}{\lambda_1}\right| < 1$ pour tout $i > 1$. Ainsi, le terme $\sum_{i=2}^{N} \alpha_i \left(\frac{\lambda_i}{\lambda_1}\right)^n u_i$ tend vers le vecteur nul quand $n \to +\infty$. Pour que le vecteur $A^n x_0$ s'aligne avec la direction de u_1, **il est indispensable que** $\alpha_1 \neq 0$. Autrement dit, le vecteur initial x_0 ne doit pas appartenir à l'hyperplan engendré par les autres vecteurs propres.

3. **Estimation de λ_1 avec le quotient de Rayleigh :** Une fois que la suite (x_n) a suffisamment convergé, x_n est une bonne approximation d'un vecteur propre unitaire associé à λ_1. On estime λ_1 en utilisant le quotient de Rayleigh : $\mu_n = x_n^T A x_n$.

4. **Implémentation en Python pour $N = 5$:** Voici le script Python modélisant la méthode de la puissance itérée pour la matrice demandée.

```python
import numpy as np

# Construction de la matrice A (N=5)
A = np.array([
    [ 2, -1,  0,  0,  0],
    [-1,  2, -1,  0,  0],
    [ 0, -1,  2, -1,  0],
    [ 0,  0, -1,  2, -1],
    [ 0,  0,  0, -1,  2]
])

# Initialisation
np.random.seed(42)
x = np.random.rand(5)
x = x / np.linalg.norm(x) # x_0 unitaire

# Algorithme de la puissance iteree
iterations = 20
for i in range(iterations):
    Ax = np.dot(A, x)
    x = Ax / np.linalg.norm(Ax)

    # Estimation de la valeur propre dominante (Quotient de Rayleigh)
    lambda_1_est = np.dot(x.T, np.dot(A, x))

print(f"Valeur propre dominante estimee : {lambda_1_est:.4f}")
print("Vecteur propre associe :", x)
```

5. **Adaptation pour la plus petite valeur propre (Puissance inverse) :** On suppose maintenant que l'on cherche la valeur propre λ_1 telle que $|\lambda_1| < |\lambda_i|$ pour $i > 1$. Pour cela, on applique la méthode de la puissance à la matrice inverse A^{-1}.

   Preuve que le spectre de A^{-1} est l'inverse de celui de A : Soit λ une valeur propre de A et $x \neq 0$ un vecteur propre associé. On a donc :

$$Ax = \lambda x$$

   Puisque l'on suppose que l'on peut inverser la matrice (et que 0 n'est pas valeur propre), on multiplie à gauche par A^{-1} :

$$A^{-1}(Ax) = A^{-1}(\lambda x)$$

$$x = \lambda A^{-1}x$$

   En divisant par λ (qui est non nul car A est inversible), on obtient :

$$A^{-1}x = \frac{1}{\lambda}x$$

   Ainsi, $\frac{1}{\lambda}$ est bien valeur propre de A^{-1}, associée au même vecteur propre x.

   Si λ_1 est la valeur propre de A de **plus petit** module, alors $\frac{1}{\lambda_1}$ sera la valeur propre de A^{-1} de **plus grand** module. On applique donc l'algorithme à A^{-1} : on résout le système linéaire $Ay_n = x_n$, puis on normalise $x_{n+1} = \frac{y_n}{\|y_n\|}$.

```python
# Algorithme de la puissance iteree INVERSE en Python
x_inv = np.random.rand(5)
x_inv = x_inv / np.linalg.norm(x_inv)
```

```python

for i in range(iterations):
    # On resout A * y = x_inv plutot que d'inverser A explicitement
    y = np.linalg.solve(A, x_inv)
    x_inv = y / np.linalg.norm(y)

# Quotient de Rayleigh pour l'estimation
lambda_min_est = np.dot(x_inv.T, np.dot(A, x_inv))
print(f"Plus petite vp estimee : {lambda_min_est:.4f}")
```

Exercice 2 : Orthogonalisation et décomposition QR

1. **Base de Gram-Schmidt :** Le procédé de Gram-Schmidt construit itérativement des vecteurs orthogonaux. On pose $q_1 = a_1/\|a_1\|$, ce qui donne $a_1 = \|a_1\|q_1$ (donc $r_{11} = \|a_1\|$). Pour l'étape j, on projette a_j sur l'espace engendré par $(q_1, \dots, q_{j-1})$:

$$\tilde{q}_j = a_j - \sum_{i=1}^{j-1} \langle a_j, q_i \rangle q_i$$

   Puis on normalise $q_j = \tilde{q}_j/\|\tilde{q}_j\|$. En réarrangeant, on obtient bien $a_j = \sum_{i=1}^{j} r_{ij} q_i$ avec $r_{ij} = \langle a_j, q_i \rangle$ pour $i < j$, et $r_{jj} = \|\tilde{q}_j\|$.
2. **Décomposition $A = QR$:** En écrivant l'égalité précédente sous forme matricielle, les colonnes a_j forment la matrice A. Les vecteurs q_i forment une matrice Q dont les colonnes sont orthonormées (donc Q est unitaire/orthogonale). Les coefficients r_{ij} forment une matrice R. Comme r_{ij} n'apparaît que pour $i \leq j$, la matrice R est triangulaire supérieure.

Correction de l'Exercice 3 : Calcul des valeurs propres suivantes (Déflation)

1. **Orthogonalité de y_1 et v_1 :** Calculons le produit scalaire entre v_1 et y_1, qui s'écrit matriciellement $v_1^T y_1$:

$$v_1^T y_1 = v_1^T (Ax_0 - (v_1^T Ax_0)v_1) = v_1^T Ax_0 - (v_1^T Ax_0)(v_1^T v_1)$$

   Comme v_1 est un vecteur propre unitaire par hypothèse, on a $\|v_1\|^2 = v_1^T v_1 = 1$. L'expression se simplifie donc immédiatement :

$$v_1^T y_1 = v_1^T Ax_0 - v_1^T Ax_0 = 0$$

   On en conclut que le vecteur y_1 est bien orthogonal à v_1.

   Remarque : On peut aussi noter que si x_0 a bien été construit pour être orthogonal à v_1, alors $v_1^T Ax_0 = (Av_1)^T x_0 = (\lambda_1 v_1)^T x_0 = \lambda_1 v_1^T x_0 = 0$. Donc théoriquement, $y_1 = Ax_0$. La soustraction sert surtout en pratique algorithmique à corriger la dérive due aux erreurs d'arrondis numériques.
2. **Algorithme itératif pour λ_2 :** L'idée est d'appliquer la méthode de la puissance itérée tout en forçant le vecteur courant à rester dans l'orthogonal du sous-espace propre dominant engendré par v_1.

   Initialisation : Choisir un vecteur aléatoire $s \in \mathbb{R}^n$. Poser $x_0 = s - (v_1^T s)v_1$, puis normaliser $x_0 \leftarrow \frac{x_0}{\|x_0\|}$.

   Itérations : Pour $k = 0, 1, 2, \dots$ faire :

— Calcul de l'image : $y_{k+1} = Ax_k$
— Orthogonalisation par rapport à v_1 (Déflation ponctuelle) : $\tilde{y}_{k+1} = y_{k+1} - (v_1^T y_{k+1})v_1$
— Normalisation : $x_{k+1} = \frac{\tilde{y}_{k+1}}{\|\tilde{y}_{k+1}\|}$
— Estimation de la valeur propre par le quotient de Rayleigh $\mu_{k+1} = x_{k+1}^T A x_{k+1}$
La suite (μ_k) convergera vers λ_2.

3. **Généralisation à l'ensemble du spectre :** Pour calculer la k-ième valeur propre λ_k (après avoir déterminé $\lambda_1, \ldots, \lambda_{k-1}$ et les vecteurs propres orthonormés associés $v_1, \ldots, v_{k-1}$), on généralise le principe de la déflation de Hotelling.

   À chaque itération de la méthode de la puissance, il faut s'assurer que le vecteur courant est orthogonal à **tous** les vecteurs propres précédents. L'étape d'orthogonalisation de l'algorithme devient une projection de Gram-Schmidt généralisée :

$$\tilde{y}_{m+1} = y_{m+1} - \sum_{i=1}^{k-1} (v_i^T y_{m+1}) v_i$$

   Lien avec le cours : Construire cette boucle revient exactement à appliquer la méthode de la puissance classique sur la matrice déflatée $A_{k-1} = A - \sum_{i=1}^{k-1} \lambda_i v_i v_i^T$. Le spectre de cette nouvelle matrice a été modifié de sorte que les anciennes valeurs propres dominantes soient remplacées par 0, laissant λ_k comme nouvelle valeur propre dominante.

Exercice 4 : Décalage spectral (Shift)

1. **Algorithme :** On utilise la méthode de la puissance itérée inverse avec décalage. On pose $B = A - \mu I$. Les valeurs propres de B sont $\lambda_i - \mu$. Les valeurs propres de B^{-1} sont $\frac{1}{\lambda_i - \mu}$. L'algorithme consiste à choisir x_0 unitaire, puis à chaque itération résoudre le système linéaire $(A - \mu I)y_n = x_n$, et normaliser $x_{n+1} = \frac{y_n}{\|y_n\|}$.
2. **Conditions de convergence :** Pour que la méthode converge vers le vecteur propre associé à λ_k, la valeur propre $\frac{1}{\lambda_k - \mu}$ doit être l'unique valeur propre strictement dominante de B^{-1}. Cela signifie que le dénominateur $|\lambda_k - \mu|$ doit être strictement plus petit que tous les autres $|\lambda_j - \mu|$. Il faut donc que μ soit strictement plus proche de λ_k que de n'importe quelle autre valeur propre de A.

Exercice 5 : Localisation par les disques de Gerschgorin

1. **Démonstration du théorème :** Soit $\lambda \in Sp(A)$ et $x \neq 0$ un vecteur propre associé. Soit k l'indice tel que $|x_k| = \max_j |x_j| > 0$. La k-ième ligne de $Ax = \lambda x$ s'écrit $\sum_{j=1}^{N} a_{kj} x_j = \lambda x_k$. En isolant le terme diagonal : $(\lambda - a_{kk})x_k = \sum_{j \neq k} a_{kj} x_j$. En passant au module et en utilisant l'inégalité triangulaire :

$$|\lambda - a_{kk}||x_k| \leq \sum_{j \neq k} |a_{kj}||x_j| \leq \sum_{j \neq k} |a_{kj}||x_k|$$

   En divisant par $|x_k| > 0$, on obtient $|\lambda - a_{kk}| \leq \sum_{j \neq k} |a_{kj}| = R_k$, ce qui prouve que $\lambda \in D_k$, et donc $\lambda \in \cup_i D_i$.
2. **Disques par colonnes :** On sait qu'une matrice A et sa transposée A^T ont exactement le même spectre (les mêmes valeurs propres). En appliquant le théorème de Gerschgorin à A^T, les sommes sur les lignes de A^T correspondent aux sommes sur les colonnes de A, définissant exactement les disques D'_i. D'où $\lambda \in \cup_i D'_i$.

3. **Application :** - Pour A : D_1 est de centre 3 et rayon 1. D_2 est de centre 4 et rayon 2. - Pour A^T : D'_1 est de centre 3 et rayon 2. D'_2 est de centre 4 et rayon 1. - Valeurs propres exactes : Le polynôme caractéristique est $\chi_A(\lambda) = (3-\lambda)(4-\lambda)-2 = \lambda^2-7\lambda+10 = (\lambda-5)(\lambda-2)$. Les valeurs propres sont 2 et 5. On vérifie aisément que $2 \in D_1 \cap D'_1$ et $5 \in D_2 \cap D'_1$, validant le théorème.

Exercice 6

1. Montrer que la matrice $G(i,j,\theta)$ est une matrice orthogonale.

Par définition, la matrice de rotation de Givens $G = G(i,j,\theta)$ est identique à la matrice identité de taille $N \times N$, sauf aux intersections des lignes et colonnes i et j où l'on a : $G_{ii} = \cos\theta$, $G_{jj} = \cos\theta$, $G_{ij} = -\sin\theta$ et $G_{ji} = \sin\theta$.

Pour étudier cette matrice rigoureusement, introduisons une matrice de permutation P. Soit P la matrice associée à une permutation des vecteurs de la base canonique ramenant l'indice i en position 1, l'indice j en position 2, et décalant les autres indices ensuite. Puisque P est une matrice de permutation, elle est orthogonale : $P^TP = PP^T = I_N$.

$$P = \left(\underbrace{e_i}_{\text{col 1}} \;\middle|\; \underbrace{e_j}_{\text{col 2}} \;\middle|\; \underbrace{e_1 \;\ldots\; \hat{e_i} \;\ldots\; \hat{e_j} \;\ldots\; e_N}_{\text{autres colonnes dans l'ordre croissant}} \right)$$

La notation $\hat{e_k}$ signifie que e_k est absent de cette suite.

En appliquant ce changement de base, la matrice semblable $G' = P^TGP$ prend une forme diagonale par blocs très simple, avec le bloc de rotation isolé en haut à gauche :

$$G' = P^TGP = \begin{pmatrix} J & 0 \\ 0 & I_{N-2} \end{pmatrix}$$

où I_{N-2} est la matrice identité de taille $N-2$, et J est la sous-matrice de rotation 2×2 :

$$J = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Vérifions d'abord l'orthogonalité du bloc J :

$$J^TJ = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} \cos^2\theta + \sin^2\theta & -\cos\theta\sin\theta + \sin\theta\cos\theta \\ -\sin\theta\cos\theta + \cos\theta\sin\theta & \sin^2\theta + \cos^2\theta \end{pmatrix}$$

Puisque $\cos^2\theta + \sin^2\theta = 1$, on obtient $J^TJ = I_2$.

Calculons maintenant le produit $(G')^TG'$ en utilisant la multiplication par blocs :

$$(G')^TG' = \begin{pmatrix} J^T & 0 \\ 0 & I_{N-2}^T \end{pmatrix} \begin{pmatrix} J & 0 \\ 0 & I_{N-2} \end{pmatrix} = \begin{pmatrix} J^TJ & 0 \\ 0 & I_{N-2} \end{pmatrix} = \begin{pmatrix} I_2 & 0 \\ 0 & I_{N-2} \end{pmatrix} = I_N$$

La matrice G' est donc orthogonale.

Enfin, revenons à notre matrice initiale G. Sachant que $G' = P^TGP$, on a :

$$(P^TGP)^T(P^TGP) = I_N \implies P^TG^TPP^TGP = I_N$$

Comme $PP^T = I_N$, l'équation se simplifie en :

$$P^TG^TGP = I_N$$

En multipliant à gauche par P et à droite par P^T, on obtient :

$$G^TG = PI_NP^T = PP^T = I_N$$

La matrice $G(i,j,\theta)$ vérifie bien $G^TG = I_N$, elle est donc orthogonale.

2. Annulation de l'élément en position (i, c) Soit A une matrice $N \times N$ et i, c des entiers tels que $c < i \leq N$. Lors du produit matriciel à gauche $G(i-1, i, \theta)A$, seules les lignes $i-1$ et i de la matrice A sont modifiées.

Le coefficient en position (i, c) de la nouvelle matrice résultante s'écrit comme la combinaison linéaire suivante :

$$(G(i-1, i, \theta)A)_{i,c} = G_{i,i-1}A_{i-1,c} + G_{i,i}A_{i,c}$$

En remplaçant par les coefficients de la rotation de Givens, cela donne :

$$(G(i-1, i, \theta)A)_{i,c} = \sin(\theta)A_{i-1,c} + \cos(\theta)A_{i,c}$$

Pour que cet élément soit nul, nous devons résoudre l'équation :

$$(G(i-1, i, \theta)A)_{i,c} = \sin(\theta)A_{i-1,c} + \cos(\theta)A_{i,c} = 0$$

Deux cas se présentent :
— Si $A_{i-1,c} = 0$, il suffit de choisir $\theta = \frac{\pi}{2}$.
— Si $A_{i-1,c} \neq 0$:

$$\tan\theta = -\frac{A_{i,c}}{A_{i-1,c}}$$

Puisque la fonction tangente est une bijection de $]-\frac{\pi}{2}, \frac{\pi}{2}[$ vers $\mathbb{R}$, il existe toujours un angle θ permettant d'annuler ce coefficient.

3. Algorithme itératif pour la décomposition QR La décomposition QR vise à exprimer A sous la forme $A = QR$, où Q est une matrice orthogonale et R une matrice triangulaire supérieure. Le principe algorithmique utilisant les rotations de Givens est le suivant :

1. On parcourt la matrice A colonne par colonne, pour un indice c allant de 1 à $N-1$.
2. Pour chaque colonne c, on applique successivement des rotations de Givens $G(i-1, i, \theta)$ en remontant de la dernière ligne vers la ligne juste sous la diagonale (pour i allant de N en décroissant jusqu'à $c+1$).
3. Chaque rotation annule le coefficient en position (i, c) sans détruire les zéros créés précédemment à sa gauche ou en dessous.

   Preuve de la conservation des zéros :
   Soit c la colonne courante dont on veut annuler les coefs sous la diagonale et soit $i > c$.

   Soit A' la matrice obtenue après la rotation : $A' = G(i-1, i, \theta)A$. Cette opération ne modifie que les lignes $i-1$ et i. Ainsi pour montrer que les zeros des colonnes précédentes ne sont pas modifiés, il faut et il suffit de monter que $A'_{i-1,c'} = A'_{i-1,c'} = 0$ avec $c' < c$. On a :

$$A'_{i-1,c'} = \cos\theta \cdot A_{i-1,c'} - \sin\theta \cdot A_{i,c'}$$
$$A'_{i,c'} = \sin\theta \cdot A_{i-1,c'} + \cos\theta \cdot A_{i,c'}$$

   Conservation à gauche ($c' < c$) : Comme $i > c$ et $c' < c$ alors $i > c'$ donc par les itérations précédentes $A_{i,c'} = A_{i-1,c'} = 0$. Par conséquent :

$$A'_{i-1,c'} = A'_{i,c'} = 0$$

4. Après avoir traité toutes les colonnes, la matrice transformée est triangulaire supérieure, notons-la $R = G_k \dots G_2 G_1 A$.
5. En posant $Q^T = G_k \dots G_2 G_1$, on obtient $A = QR$. La matrice Q est orthogonale car elle est la transposée (et l'inverse) d'un produit de matrices orthogonales.

4. Application à la matrice A Nous disposons de la matrice :

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}$$

L'objectif est d'annuler le coefficient en position $(3,1)$ en appliquant la matrice de rotation $G(2,3,\theta)$ (rotation agissant sur les lignes 2 et 3). Les éléments de la première colonne impliqués sont $A_{2,1} = 1$ et $A_{3,1} = 1$.

D'après la formule établie à la question 2, nous cherchons θ tel que :

$$\sin(\theta)A_{2,1} + \cos(\theta)A_{3,1} = 0$$

$$\sin(\theta) \cdot 1 + \cos(\theta) \cdot 1 = 0 \implies \tan(\theta) = -1$$

Nous choisissons l'angle $\theta = -\frac{\pi}{4}$. Les coefficients trigonométriques sont :

$$\cos\left(-\frac{\pi}{4}\right) = \frac{\sqrt{2}}{2} \quad \text{et} \quad \sin\left(-\frac{\pi}{4}\right) = -\frac{\sqrt{2}}{2}$$

L'application de la matrice G modifie uniquement les lignes $L_2 = (1,2,3)$ et $L_3 = (1,1,1)$ de la matrice A selon les relations suivantes :

— **Nouvelle ligne 2 :**

$$L_2' = \cos(\theta)L_2 - \sin(\theta)L_3 = \frac{\sqrt{2}}{2}(1,2,3) - \left(-\frac{\sqrt{2}}{2}\right)(1,1,1)$$

$$L_2' = \frac{\sqrt{2}}{2}(1+1, 2+1, 3+1) = \left(\sqrt{2}, \frac{3\sqrt{2}}{2}, 2\sqrt{2}\right)$$

— **Nouvelle ligne 3 :**

$$L_3' = \sin(\theta)L_2 + \cos(\theta)L_3 = -\frac{\sqrt{2}}{2}(1,2,3) + \frac{\sqrt{2}}{2}(1,1,1)$$

$$L_3' = \frac{\sqrt{2}}{2}(-1+1, -2+1, -3+1) = \left(0, -\frac{\sqrt{2}}{2}, -\sqrt{2}\right)$$

La matrice résultante, où le coefficient en position $(3,1)$ est bien annulé, est donc :

$$G\left(2,3,-\frac{\pi}{4}\right)A = \begin{pmatrix} 0 & 1 & 1 \\ \sqrt{2} & \frac{3\sqrt{2}}{2} & 2\sqrt{2} \\ 0 & -\frac{\sqrt{2}}{2} & -\sqrt{2} \end{pmatrix}$$

Exercice 7

Soit A une matrice inversible de taille $n \times n$ et $Q = I - 2uu^T$ une matrice de réflexion de Householder avec $||u|| = 1$. Cet exercice explore les propriétés de ces matrices pour aller plus loin.

1. Symétrie et orthogonalité de Q **Symétrie :** Pour montrer que la matrice de Householder Q est symétrique, on transpose l'expression en utilisant la linéarité de la transposition et la propriété $(AB)^T = B^T A^T$:

$$Q^T = (I - 2uu^T)^T = I^T - 2(u^T)^T u^T = I - 2uu^T = Q$$

La matrice Q est donc bien symétrique.

Orthogonalité : Pour montrer qu'elle est orthogonale, calculons $Q^T Q$. Sachant que Q est symétrique, $Q^T Q = Q^2$:

$$Q^2 = (I - 2uu^T)(I - 2uu^T) = I - 2uu^T - 2uu^T + 4(uu^T)(uu^T)$$

Puisque le produit matriciel est associatif, on peut réécrire le terme de droite : $(uu^T)(uu^T) = u(u^T u)u^T$. Or, le vecteur u est unitaire ($||u|| = 1$), donc le scalaire $u^T u$ vaut 1.

$$Q^2 = I - 4uu^T + 4u(1)u^T = I - 4uu^T + 4uu^T = I$$

Ainsi, $Q^T Q = I$, ce qui prouve l'orthogonalité.

2. Annulation des coefficients d'un vecteur (Raisonnement par équivalence) Soit $x \in \mathbb{R}^n$ un vecteur non nul. Nous cherchons l'existence d'un vecteur unitaire u (avec $\|u\| = 1$) et d'un scalaire α tels que $Qx = \alpha e_1$.

Raisonnons par équivalence pour déterminer les conditions nécessaires sur α et u.

$$\begin{aligned} Qx = \alpha e_1 &\iff (I - 2uu^T)x = \alpha e_1 \\ &\iff x - 2(u^T x)u = \alpha e_1 \\ &\iff 2(u^T x)u = x - \alpha e_1 \quad (*) \end{aligned}$$

Cette dernière égalité vectorielle $(*)$ nous impose deux contraintes simultanées : une sur la norme et une sur la direction.

1. **Condition sur la norme (détermination de α) :**
   Puisque Q est orthogonale (d'après la question 1), elle préserve la norme euclidienne. En prenant la norme de l'équation initiale :

   $$\|Qx\| = \|\alpha e_1\| \iff \|x\| = |\alpha| \cdot \|e_1\|$$

   Comme $\|e_1\| = 1$, on a nécessairement $|\alpha| = \|x\|$. Il existe donc deux choix possibles pour le scalaire : $\alpha = \pm\|x\|$.
2. **Condition sur la direction (détermination de u) :**
   L'équation $(*)$ montre que le vecteur u doit être colinéaire au vecteur $v = x - \alpha e_1$. En effet, le terme $2(u^T x)$ est un scalaire λ. L'équation s'écrit $\lambda u = v$. Pour que u existe, il faut que le vecteur $v = x - \alpha e_1$ ne soit pas nul.
   — Si x est déjà colinéaire à e_1 (i.e., $x = \|x\|e_1$), on choisit $\alpha = -\|x\|$ pour que $v = 2\|x\|e_1 \neq 0$.
   — Sinon, on peut choisir indifféremment $\alpha = \|x\|$ ou $\alpha = -\|x\|$.
   Une fois α fixé tel que $v \neq 0$, la condition de colinéarité et la contrainte $\|u\| = 1$ imposent :

   $$u = \frac{x - \alpha e_1}{\|x - \alpha e_1\|}$$

Conclusion : Il est toujours possible de choisir un vecteur unitaire u vérifiant la condition demandée en posant :

$$u = \frac{x - \alpha e_1}{\|x - \alpha e_1\|} \quad \text{avec} \quad \alpha = -\text{signe}(x_1)\|x\|$$

(Le choix du signe opposé à la première composante x_1 est préféré numériquement pour maximiser la norme du dénominateur et éviter les erreurs d'arrondi).

3. Transformation en matrice de Hessenberg supérieure L'objectif est d'appliquer des transformations de similitude du type $A \leftarrow QAQ^T$ pour préserver les valeurs propres, tout en introduisant des zéros sous la première sous-diagonale pour obtenir une matrice de Hessenberg supérieure H (où $H_{ij} = 0$ pour $i > j + 1$).

Étape 1 : Le problème de la méthode naïve

Si l'on cherche à mettre des zéros sous le tout premier coefficient de la colonne 1 de A, on trouve (selon la question 2) une matrice de Householder Q de taille $n \times n$ telle que la première colonne de QA soit colinéaire à e_1. Cependant, pour conserver la similitude, il faut ensuite calculer la matrice $(QA)Q^T$. La multiplication à droite par Q^T (qui est pleine) va combiner linéairement toutes les colonnes de QA. Cela va immanquablement détruire les zéros que nous venons de créer dans la première colonne.

Étape 2 : L'astuce de la matrice par blocs

Pour que la multiplication à droite ne détruise pas nos zéros, l'astuce consiste à ne pas toucher à la première ligne ni à l'élément diagonal de la première colonne. Pour cela, on décompose la matrice A par blocs :

$$A = \begin{pmatrix} a_{11} & l^T \\ x & A' \end{pmatrix}$$

où a_{11} est un scalaire, l et x sont des vecteurs de $\mathbb{R}^{n-1}$, et A' est une sous-matrice de taille $(n-1) \times (n-1)$.

D'après la question 2, il existe une matrice de Householder Q_1 de taille $(n-1) \times (n-1)$ telle que $Q_1 x = \alpha e_1^{(n-1)}$ (c'est-à-dire un vecteur avec α sur sa première composante, et des zéros partout en dessous). On construit alors une matrice de taille $n \times n$ en la bordant par l'identité :

$$P_1 = \begin{pmatrix} 1 & 0 \\ 0 & Q_1 \end{pmatrix}$$

Puisque Q_1 est symétrique et orthogonale, P_1 l'est également ($P_1^T = P_1$ et $P_1^2 = I$).

Étape 3 : Application de la similitude et préservation des zéros

Regardons d'abord l'effet de la multiplication à gauche par P_1 :

$$P_1 A = \begin{pmatrix} 1 & 0 \\ 0 & Q_1 \end{pmatrix} \begin{pmatrix} a_{11} & l^T \\ x & A' \end{pmatrix} = \begin{pmatrix} a_{11} & l^T \\ Q_1 x & Q_1 A' \end{pmatrix} = \begin{pmatrix} a_{11} & l^T \\ \alpha e_1^{(n-1)} & Q_1 A' \end{pmatrix}$$

À ce stade, la première colonne possède les zéros souhaités sous le terme de la première sous-diagonale.

Appliquons maintenant la multiplication à droite par P_1^T (qui vaut P_1) :

$$A_1 = (P_1 A)P_1 = \begin{pmatrix} a_{11} & l^T \\ \alpha e_1^{(n-1)} & Q_1 A' \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & Q_1 \end{pmatrix} = \begin{pmatrix} a_{11} & l^T Q_1 \\ \alpha e_1^{(n-1)} & Q_1 A' Q_1 \end{pmatrix}$$

C'est ici que la magie opère : grâce au 1 et aux 0 dans la première colonne de P_1, la première colonne de notre matrice n'est pas modifiée par la multiplication à droite ! Les zéros sont donc préservés.

Étape 4 : Itération

La matrice A_1 obtenue possède des zéros sous la position $(2, 1)$. On répète alors ce même processus pour la sous-matrice $Q_1 A' Q_1$ en ciblant sa première colonne, à l'aide d'une nouvelle matrice $P_2 = \begin{pmatrix} I_2 & 0 \\ 0 & Q_2 \end{pmatrix}$. En itérant ce procédé $n-2$ fois, on amène des zéros sous toute la première sous-diagonale. Le produit final $H = P_{n-2} \dots P_1 A P_1 \dots P_{n-2}$ est donc bien une matrice de Hessenberg supérieure.

4. Cas d'une matrice symétrique de départ Si la matrice A initiale est symétrique ($A^T = A$), regardons la transposée de la matrice H obtenue :

$$H^T = (P_{n-2} \dots P_1 A P_1^T \dots P_{n-2}^T)^T = P_{n-2} \dots P_1 A^T P_1^T \dots P_{n-2}^T = H$$

La matrice H est donc symétrique. Puisque H est de Hessenberg supérieure ($H_{ij} = 0$ pour $i > j+1$), sa symétrie impose que $H_{ij} = H_{ji} = 0$ pour $j > i+1$. Les seuls éléments potentiellement non nuls se situent sur la diagonale, la première sous-diagonale et la première sur-diagonale. La matrice de Hessenberg H obtenue est donc tridiagonale symétrique.

5. Conservation de la forme de Hessenberg (Stabilité de l'algorithme QR) Soit H une matrice de Hessenberg supérieure (c'est-à-dire $H_{ij} = 0$ pour $i > j + 1$). Notons $H = QR$ sa décomposition QR et $H' = RQ$ la matrice obtenue après une itération. Montrons que H' conserve la forme de Hessenberg.

- **Étape 1 : Structure de la matrice Q.** Puisque R est triangulaire supérieure inversible, son inverse R^{-1} l'est aussi. On peut écrire $Q = HR^{-1}$. Le coefficient général de Q est $Q_{ij} = \sum_{k=1}^{n} H_{ik}(R^{-1})_{kj}$. Puisque R^{-1} est triangulaire supérieure, $(R^{-1})_{kj} = 0$ si $k > j$. La somme s'arrête donc à $k = j$. Si on considère un indice $i > j+1$, alors pour tout $k \leq j$, on a $k < i - 1$, ce qui implique $H_{ik} = 0$ (car H est de Hessenberg). Ainsi, $Q_{ij} = 0$ pour tout $i > j + 1$. La matrice Q est donc elle-même une matrice de Hessenberg supérieure.
- **Étape 2 : Structure de la matrice H'.** On calcule $H' = RQ$. Le coefficient général est $H'_{ij} = \sum_{k=1}^{n} R_{ik}Q_{kj}$. Puisque R est triangulaire supérieure, $R_{ik} = 0$ si $k < i$. La somme ne commence donc qu'à $k = i$. Supposons que $i > j + 1$. Dans la somme, on a toujours $k \geq i$, donc $k > j + 1$. Or, nous avons montré que Q est de Hessenberg, donc $Q_{kj} = 0$ pour $k > j + 1$. Tous les termes de la somme sont nuls.

Conclusion : On a bien $H'_{ij} = 0$ pour tout $i > j + 1$. L'algorithme QR préserve la structure de Hessenberg, ce qui est crucial pour maintenir le coût de calcul faible au fil des itérations.

6. Intérêt pour l'algorithme QR L'intérêt de cette transformation préalable est d'accélérer l'algorithme QR vu en cours. Le point crucial à comprendre est que l'algorithme QR est **itératif** : la décomposition n'est pas effectuée une seule fois, mais répétée K fois dans une boucle pour converger vers les valeurs propres.

Voici la différence de coût algorithmique :

- **Sans étape préliminaire :** Chaque itération de l'algorithme QR sur une matrice pleine demande $O(n^3)$ opérations. Pour K itérations, le coût total est donc de $K \times O(n^3)$.
- **Avec l'étape préliminaire de Householder :** On paye le coût de la réduction en matrice de Hessenberg **une seule fois** au début (qui est bien de $O(n^3)$). L'avantage majeur est que la structure de Hessenberg (ou tridiagonale symétrique si la matrice de départ l'était) est conservée lors des itérations QR suivantes. Sur une telle structure, le coût d'une itération QR chute drastiquement à $O(n^2)$ opérations (et même $O(n)$ dans le cas tridiagonal). Le coût total devient alors $O(n^3) + K \times O(n^2)$.

En résumé, au lieu de payer le prix fort de $O(n^3)$ à chaque itération, on le paye une seule fois au démarrage. Pour des matrices de grande taille (n grand) et un grand nombre d'itérations (K grand), cette astuce permet de gagner énormément de temps de calcul.