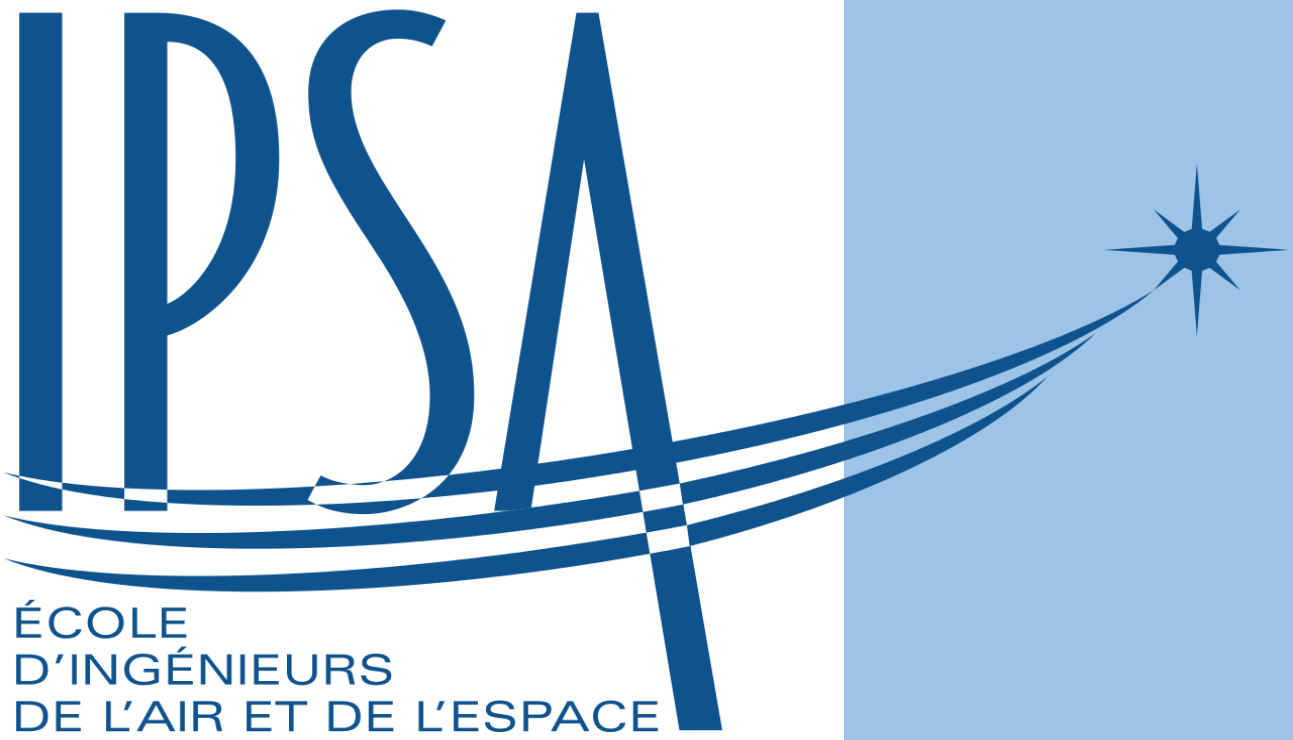
TP2-Décomposition QR, algorithme de Gram-Schmidt

Compte rendu MA313



EL ZENARY RAMEL
GINET PAUL
VINCIENNE—CORREGE MATHILDE
AERO 3, CLASSE B1

Table des matières

Introduction

À travers ce TP2, nous allons expérimenter des fonctions simples permettant d'appliquer la méthode de Gram-Schmidt pour résoudre des systèmes linéaires.

Historique :

Jørgen Pedersen Gram et Erhard Schmidt sont deux mathématiciens du milieu du XIX ème siècle. Ils sont célèbres pour leur méthode de résolution des systèmes d'équations linéaires. La méthode de Gram-Schmidt a été publiée par Jørgen Pedersen Gram en 1883 et reformulée par Erhard Schmidt en 1907, mais on la trouve déjà dans des travaux de 1816 de Laplace.

La méthode de Gram-Schmidt consiste pour une matrice A carrée, inversible à déterminer une matrice Q qui est orthogonale et R une matrice triangulaire. Un des avantages de la décomposition A = QR est que la matrice Q est très bien conditionnée (conditionnement égal à 1).

L'un des objectifs de ce TP est de tester la méthode de Gram-Schmidt pour résoudre un système AX=B mais nous allons également s'intéresser à son efficacité et à sa rapidité par rapport à d'autres méthodes. Pour cela, nous allons effectuer la méthode de Cholesky et celle de Gauss (vu précédemment en Aéro 2) et la méthode Householder que nous comparerons avec la méthode de Gram-Schmidt par le temps des calculs, par les incertitudes d'erreurs et leurs complexités.

1ère Partie : Décomposition QR par la méthode de Gram-Schmidt

Généralité :
La méthode de décomposition QR est une des nombreuses alternatives servant à simplifier la résolution de systèmes matricielles de la forme Ax = B.
Cette méthode consiste à décomposer la matrice A en une matrice Q orthogonale tel que A=QR et une matrice R étant triangulaire supérieur tel que : $Q^tA = Q^tQR = IR = R$
La méthode de décomposition auxquels l'ont va s'intéresser est celle de Gram-Schmidt.
Le protocole de décomposition est le suivant (i désignant la ligne et j la colonne) :

1. Le calcul, pour chaque i avec i < j, des coefficients de R par la formule rij =< aj , qi >.
   Tout d'abord, nous cherchons à calculer les coefficients de R. Pour cela, nous calculons les rij à l'aide de la formule ci-dessus.

2. Le calcul d'un vecteur $wj = aj - \sum_{k=1}^{j-1} r$kj*qk
   Ensuite, nous calculons le vecteur wij. Pour cela, nous créons une autre boucle dans laquelle nous réaliserons la différence entre aj et la somme des termes du produit $rkj * q$. On va poser initialement S qui correspond à la somme des termes du produit $rkj * q$. Une fois que S sera calculé, nous calculerons le vecteur wj par leur différence à l'aide de la formule :

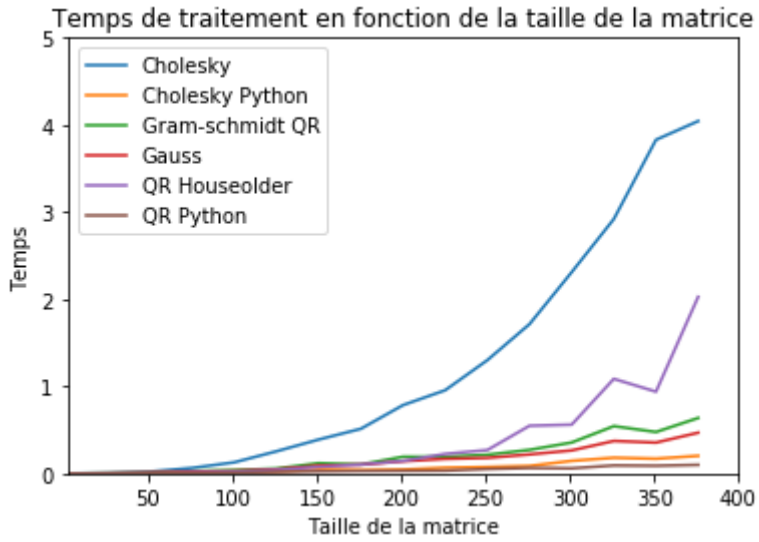
3. Le calcul de rjj =$\|w_j\|$.
   Nous calculerons les termes diagonaux de R en faisant la norme du vecteur wj trouvé précédemment.

4. Le calcul de qj = (1 /rjj)* wj .
   Puis nous calculerons les termes diagonaux de Q en faisant une division de wj par rjj tous deux calculé précédemment.

En prenant bien soin de comprendre que la première fois que l'on effectue ces itérations, on doit commencer à partir de l'étape 3 et traiter aj à la place de wj, la matrice intermédiaire.

NB : Pour la génération des courbes qui suivront nous avons dû générer des matrices symétriques définis positive au lieu de simple matrice inversible, pour pouvoir appliquer la méthode de Cholesky afin d'effectuer une comparaison.

2ème Partie : Temps d'exécution des programmes

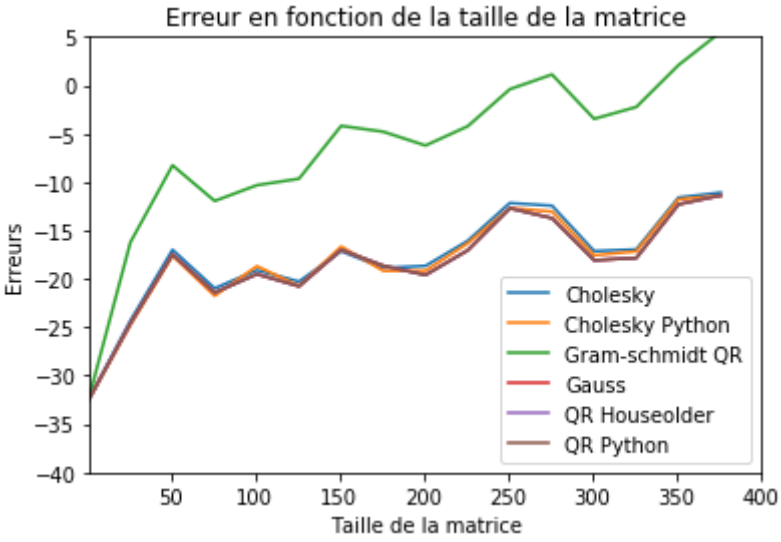


Ce graphique nous montre le temps en seconde qu'il faut pour que le programme réalise les différentes fonctions de résolution des systèmes d'équations linéaires de tailles variées. Le temps d'exécution varie en fonction de la complexité d'un programme, plus le nombre de boucles ou de fonctions tels que les « if » sont important dans le code, plus le programme sera long à exécuter. Par ailleurs, nous observons que le temps d'exécution des six méthodes différentes augmente avec la taille de la matrice. Cela est évident puisque les calculs seront plus longs dû à la taille de la matrice. Cependant, nous remarquons que le traitement de la fonction Cholesky augmente de manière exponentielle pour une taille de matrice supérieur à 50. Ce phénomène s'explique par le nombre de boucle « for » dans la fonction. Nous allons donc expliquer la complexité de la fonction Cholesky(A).

C = (2 + boucle (1) + 4)
Boucle (1) = (2 + boucle (2) + boucle (3)) * ligne = (2 + i + (2 + i) * (ligne – i)) * ligne
Boucle (2) = i
Boucle (3) = (2 + boucle (4)) * (ligne – i) = (2 + i) * (ligne – i)
Boucle (4) = i
C = 6 + (2 + i + (2 + i) * (ligne – i)) * ligne

Par ce calcul de complexité, nous comprenons que plus la taille de la matrice augmente plus la complexité sera grande. En effet, le facteur le plus grand de cette équation est la ligne de la matrice au carré multiplié par i. Et par conséquent l'évolution exponentielle de la courbe est expliquée. A l'inverse, les fonctions QR python et Cholesky python sont les deux fonctions qui s'exécutent le plus rapidement. Cela s'explique par le fait que ces deux fonctions sont des bibliothèques de python et sont donc très optimisées. Ainsi, les méthodes QR houseolder, Gauss et Gram-schmidt QR sont moins optimisées que les méthodes de python avec plus de lignes de calculs mais elles possèdent une complexité moins importante que la méthode de Cholesky. C'est pourquoi elles se différencient des fonctions python sans pour autant avoisiner la fonction Cholesky.

3ème Partie : Erreur en fonction des différentes méthodes de résolutions



Sur ce graphique, nous calculons l'erreur de chacune de ses fonctions étudiées précédemment. Toutes ses fonctions résolvent l'équation A.X = B avec A une matrice carré, B un vecteur de la taille de A , et X un vecteur inconnu. L'erreur est calculée par la différence des normes des deux matrices de part et d'autre de l'inégalité. Plus cette dernière est proche de 0 plus la fonction est fiable, cependant sur le graphique on remarque que le 0 n'est pas le point le plus faible en ordonnée. Cela est dû au fait que l'on a ajouté la fonction log à la norme pour ainsi nous permettre d'avoir un graphe présentable. En effet, sans l'application du logarithme, l'échelle ne serait pas adaptée et par conséquent notre analyse n'aura pas d'intérêt.

Nous observons, dans un premier temps, que chaque fonction possède les mêmes pics d'erreurs. Ces pics sont dus à une erreur de calcul matricielle, ils dépendent de la matrice générée. C'est-à-dire que si nous relançons le programme les pics d'erreurs ne seront pas situés à la même taille. En revanche, nous remarquons que la fonction Gram-Schmidt QR se démarque de toutes les autres fonctions qui sont plus ou moins confondues par une plus forte amplitude de l'erreur. Ce phénomène peut être expliqué par la fonction DecompositionGS du programme qui traite chaque ligne avec chaque colonne de la matrice. Ainsi, le nombre d'erreur risque d'être plus conséquent que sur les autres fonctions. Et c'est pourquoi, lorsqu'une légère différence apparait entre le produit A.X et B, la norme de Gram-schmidt QR augmente considérablement comparer aux fonctions qui possèdent moins de calculs aussi complexes.

4ème Partie : Particularité du code

Elaborations du Code :

Pour effectuer les 2 graphiques précédents, La méthode Cholesky, Gauss ainsi que Gram-schmidt ont été utilisé pour les comparer.
Les méthodes comportant « Python » dans leur nom sont composées de fonction interne à Python, à titre d'exemple, QR python fait appel à une fonction interne a python effectuant la décomposition QR et n'est pas coder ligne par ligne.
On peut également remarquer qu'une autre méthode de décomposition, QR est présente, il s'agit de la méthode dite de « Houseolder ».

Elle utilise des symétries vectorielles par rapport à des hyperplans. Avec pour but ici est de trouver les bonnes symétries pour simplifier les calculs.
$H_v = I_n - 2\frac{v \cdot v^T}{\|v\|^2}$ avec v un vecteur réelle non nul.

Le principe de la méthode de Householder est de multiplier A par des matrices de Householder pour faire apparaître une matrice triangulaire.
Soit a1 le premier vecteur colonne de A, et v1 le vecteur de Householder
On réitère l'opération jusqu'à obtenir une matrice triangulaire supérieur A^n
La décomposition QR de A est alors donnée par $Q = H_1 * \ldots * H_n$, et $R = A^n$.

Autre spécificité du code, la boucle de la somme pour la deuxième étape de la décomposition de gram-schmidt à été directement implémenté dans une autre boucle, pour réduire le temps de calcul et optimisé la fonction comme on peut le voir ci-dessous :

```
for j in range (1,n):
    S=0
    for i in range (0,j):
        R[i,j] =  np.vdot(Q[:,i],A[:,j])
        S = S + R[i,j]*Q[:,i]
    W[:,j] = A[:,j]- S
    R[j,j] = np.linalg.norm(W[:,j])
    Q[:,j] = W[:,j]*(1/R[j,j])
```

Pour continuer dans la programmation, nous avons dû commencer à générer des matrices de tailles 4 au lieu de commencer à générer des matrices de tailles 2.
En effet en générant des matrices de tailles 2, l'erreur était égale à 0, et ayant utilisé la fonction logarithme pour pouvoir mieux observer les courbes, log(0) n'étant pas définis nous avons dû commencer avec des matrices de tailles plus élevé afin que le programme puisse fonctionner.

Conclusion

A travers ce TP, nous avons exploité plusieurs méthodes qui permettent de résoudre l'équation AX=B. Nous avons utilisé la décomposition QR par l'algorithme de Gram-Schmidt, la méthode de Gauss et de Cholesky et également Householder.

Il y a plusieurs méthodes directes pour résoudre AX= B comme il y a plusieurs manières de programmer la résolution de celle-ci. Après l'étude du temps de résolutions nous avons constaté que c'était la méthode de Cholesky qui avait le plus long temps d'exécution. Il est donc possible de déterminer les solutions de l'équation de toute matrice A et B de manière plus efficace et rapide avec un gain de temps qui est de l'ordre de quelques secondes.

On en arrive donc à la conclusion que la méthode de Gauss est très intéressante car c'est le programme « fait main » le plus efficace en termes de temps de traitement, tandis qu'en ce qui concerne les erreurs, il fait partie de la moyenne. De plus il n'a pas besoin d'avoir à traiter des matrices particulière, il suffit que celle-ci soit réductible selon la méthode de gauss et non défini positive comme l'exige la méthode de Cholesky par exemple.