وزارة التعليم العالي والبحث العلمي

Université 20 Aout 1955 de Skikda
Faculté des Sciences
Département de Mathématiques

جامعة 20 أوت 1955 ، سكيكدة
كلـــــــية العلـــــوم
قسم الرياضيات

N^0 : U.S/F.S/D.M/……/2024

Faculté des Sciences
Département de Mathématiques

Mémoire

Présenté en vue de l'obtention du diplôme de
Master en Mathématiques

UNE MÉTHODE DE RELAXATION STOCHASTIQUEMENT ADAPTÉE.

Option : Analyse fonctionnelle appliquée

Par :

1. Magrous chaima

Encadré par : Dr. C. Benatmane **MCB** U. SKIKDA

Devant le jury :

Président :Dr. A. Laallouche	MCB	U. SKIKDA
Examinateur : Dr. A. Bouadi	MCB	U. SKIKDA
Dr. Y. Bedrani	MCB	U. SKIKDA

Année : 2023/2024

TABLE DES MATIÈRES

Résumé en Anglais

In numerical analysis, there are several methods for solving systems of linear equations, primarily divided into two categories : direct methods and iterative methods. Direct methods, such as Gaussian elimination and LU decomposition, aim to obtain the exact solution in a finite number of steps. In contrast, iterative methods, like the Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR) methods, progressively approach the solution through a series of iterations.

In this thesis, we delve deeply into the concept of iterative methods, particularly in the context of stochastic processes, to develop a new algorithm aimed at solving linear equations with increased accuracy. Stochastic processes, which incorporate random elements into calculations, can offer new and potentially more efficient perspectives for optimizing iterative methods.

Our research focuses on integrating these stochastic processes with traditional iterative methods to enhance the convergence and accuracy of solutions. We present the theoretical foundations of our approach, followed by a series of experiments and simulations to evaluate the performance of our algorithm.

Finally, we conclude this study with a detailed comparison of the approxi-

mate solutions obtained by our new algorithm with those provided by classical methods. This comparison highlights the advantages and disadvantages of our approach, as well as the conditions under which it may outperform existing methods. We also discuss the implications of our results for the field of numerical analysis and future prospects for research and development in iterative algorith

Keywords :

Résumé en Français

En analyse numérique, il existe plusieurs méthodes pour résoudre les systèmes d'équations linéaires, se divisant principalement en deux catégories : les méthodes directes et les méthodes itératives. Les méthodes directes, telles que la méthode de Gauss et la décomposition LU, visent à obtenir la solution exacte en un nombre fini d'étapes. En revanche, les méthodes itératives, comme les méthodes de Jacobi, Gauss-Seidel et sur-relaxation successive (SOR), approchent progressivement la solution par une série d'itérations.

Dans ce mémoire, nous explorons en profondeur le concept des méthodes itératives, en particulier dans le contexte des processus stochastiques, afin de développer un nouvel algorithme destiné à résoudre les équations linéaires avec une précision accrue. Les processus stochastiques, qui incorporent des éléments aléatoires dans les calculs, peuvent offrir des perspectives nouvelles et potentiellement plus efficaces pour l'optimisation des méthodes itératives.

Notre recherche se concentre sur l'intégration de ces processus stochastiques avec les méthodes itératives traditionnelles pour améliorer la convergence et la précision des solutions. Nous présentons les fondements théoriques de notre approche, suivis d'une série d'expérimentations et de simulations pour évaluer la performance de notre algorithme.

Enfin, nous concluons cette étude par une comparaison détaillée des solutions approximatives obtenues par notre nouvel algorithme avec celles fournies par les méthodes classiques. Cette comparaison permet de mettre en évidence les avantages et les inconvénients de notre approche, ainsi que les conditions sous lesquelles elle pourrait surpasser les méthodes existantes. Nous discutons également des implications de nos résultats pour le domaine de l'analyse numérique et des perspectives futures pour la recherche et le développement d'algorithmes itératifs.

Mots clés : Méthode itérative, Méthode Jacobi, Méthode Gauss-Seidal, Méthode de relaxation (SOR), Processus stochastiques, Chaine de Markov,

ملخص

في التحليل العددي، توجد عدة طرق لحل أنظمة المعادلات الخطية، وتنقسم بشكل رئيسي إلى فئتين: الطرق المباشرة والطرق التكرارية. تهدف الطرق المباشرة، مثل طريقة غاوس وتحليلLU ، إلى الحصول على الحل الدقيق في عدد محدد من الخطوات. في المقابل، تقترب الطرق التكرارية، مثل طرق جاكوبي وجاوس-سيدل والاسترخاء المتعاقب(SOR) ، تدريجياً من الحل عبر سلسلة من التكرارات.

في هذا البحث، نستكشف بعمق مفهوم الطرق التكرارية، خصوصاً في سياق العمليات العشوائية، بهدف تطوير خوارزمية جديدة لحل المعادلات الخطية بدقة أكبر. يمكن أن توفر العمليات العشوائية، التي تتضمن عناصر عشوائية في الحسابات، رؤى جديدة وربما أكثر فعالية لتحسين الطرق التكرارية.

تركز بحثنا على دمج هذه العمليات العشوائية مع الطرق التكرارية التقليدية لتحسين التقارب ودقة الحلول. نقدم الأسس النظرية لنهجنا، تليها سلسلة من التجارب والمحاكاة لتقييم أداء خوارزميتنا.

أخيراً، نختم هذه الدراسة بمقارنة مفصلة للحلول التقريبية التي تم الحصول عليها عبر خوارزميتنا الجديدة مع تلك التي توفرها الطرق التقليدية. تسمح هذه المقارنة بتسليط الضوء على مزايا وعيوب نهجنا، وكذلك الشروط التي يمكن أن يتفوق فيها على الطرق الحالية. نناقش أيضاً تداعيات نتائجنا على مجال التحليل العددي وآفاق البحث والتطوير المستقبلي للخوارزميات التكرارية.

Remerciements

En premier lieu, je tiens à remercier mon encadreur, **Dr Chaabane Benatmane**, pour la confiance qu'il m'a témoignée en me proposant ce sujet, ainsi que pour ses encouragements et sa patience. Les discussions scientifiques qu'il a initiées, ses remarques et ses suggestions ont été essentielles pour finaliser ce modeste travail. Son encadrement précieux, sa disponibilité constante et ses conseils avisés tout au long de ce travail ont été une source d'inspiration et de motivation inestimable.

Je tiens à exprimer ma profonde gratitude aux membres du jury, **Dr. Abdallah Laallouche**, **Dr. Abdelkader Bouaadi**, et **Dr. Yacine Bedrani**, pour avoir accepté d'évaluer ce mémoire. Leurs commentaires constructifs, leurs questions pertinentes et leurs suggestions éclairées ont grandement contribué à améliorer la qualité de ce travail. Leur expertise et leur rigueur académique ont été d'une grande aide, et je les remercie sincèrement pour le temps et l'attention qu'ils ont consacrés à cette évaluation.

Je ne peux pas clôturer mes remerciements sans se retourner vers les êtres que me sont le plus cher ; ma famille qui ont eu rôle essentiel et continu dans ma réussite.

Introduction générale

Effectivement, les méthodes numériques directes peuvent rencontrer des défis dans la résolution des systèmes linéaires de grande taille en raison de la complexité des opérations arithmétiques et de la probabilité accrue d'erreurs cumulatives. Cela signifie que dans certains cas, la résolution du système linéaire en utilisant des méthodes directes peut être peu pratique en raison de la consommation importante de temps et de ressources informatiques, et parfois même impossible en raison de la taille des données.

Pour résoudre ce problème, les méthodes numériques itératives ou auto-itératives entrent en jeu, utilisant des estimations qui convergent progressivement vers la solution précise à chaque itération. Ces méthodes sont souvent plus efficaces pour les grands systèmes linéaires, permettant de contrôler ou de réduire l'erreur cumulative. Il convient de noter que les méthodes numériques itératives présentent plusieurs avantages, tels que la capacité à traiter de grands ensembles de données et à fournir des stratégies pour gérer l'erreur cumulative.

Parmi les plus courantes, la méthode de Jacobi et la méthode de Gauss-Seidel mettent à jour successivement les variables en utilisant des valeurs de l'itération précédente ou actuelle, la méthode de Gauss-Seidel étant plus ra-

pide pour les matrices à diagonale dominante. La méthode SOR (Successive Over-Relaxation) améliore encore cette convergence en introduisant un facteur de relaxation w. Pour les systèmes linéaires symétriques et définis positifs, la méthode des gradients conjugués est particulièrement efficace, minimisant une forme quadratique associée au système. D'autres méthodes comme Richardson, GMRES (Generalized Minimal Residual Method), BiCGSTAB (Biconjugate Gradient Stabilized), et SSOR (Symmetric Successive Over-Relaxation) utilisent des sous-espaces de Krylov ou combinent des itérations pour améliorer la convergence, même pour des matrices non symétriques. Le choix de la méthode dépend de la nature du système, du conditionnement de la matrice, et des critères de convergence spécifiques à chaque problème.

La méthode SOR est particulièrement utile dans les problèmes de grandes dimensions où les méthodes directes sont impraticables. Elle est largement utilisée dans des domaines tels que l'ingénierie, la physique et les mathématiques appliquées pour résoudre des systèmes linéaires provenant de la discrétisation d'équations aux dérivées partielles.

Le facteur de relaxation w joue un rôle crucial dans l'efficacité de la méthode SOR. Si $w = 1$, la méthode se réduit à la méthode de Gauss-Seidel. Un w optimal, généralement dans l'intervalle $]0, 2[$, peut améliorer considérablement la vitesse de convergence. Toutefois, un w mal choisi peut ralentir la convergence voire rendre la méthode divergente.

Les processus stochastiques discrets sont des outils puissants pour modéliser des systèmes où les événements se produisent à des intervalles de temps distincts. Leur analyse repose sur des concepts de probabilité et de statistiques, tels que la stationnarité, l'indépendance, et l'ergodicité. Les applications de ces processus sont vastes et couvrent de nombreux domaines,

allant de la finance à la biologie en passant par la physique et l'informatique.

CHAPITRE 1

Un bref aperçu sur Matlab

1.1 Qu'est-ce que Matlab ?

Matlab, abréviation de MATrix LABoratory, est un environnement de programmation et un langage de programmation spécialisé dans le calcul numérique et la visualisation. Il est largement utilisé dans les domaines scientifiques et d'ingénierie pour la modélisation, l'analyse et la simulation de systèmes complexes.

1.2 Caractéristiques principales de Matlab.

1.2.1 Calculs Numériques.

Matlab est particulièrement efficace pour effectuer des opérations sur des matrices et des tableaux numériques, facilitant ainsi la résolution de problèmes mathématiques complexes.

1.2.2 Visualisation.

Il offre des outils puissants pour la création de graphiques et de visualisations, ce qui est essentiel pour analyser et présenter des données.

1.2.3 Programmation.

Matlab est un langage de script interprété, ce qui signifie que vous pouvez écrire des scripts et des fonctions pour automatiser des tâches répétitives ou pour effectuer des calculs complexes.

1.2.4 Intégration.

Matlab permet l'intégration avec d'autres langages de programmation, comme C/C++, Fortran, Python, etc., ce qui en fait un outil polyvalent dans un environnement de développement.

1.3 Environnement Matlab.

Lorsque vous lancez Matlab, vous êtes accueilli par l'interface utilisateur de Matlab, appelée l'Environnement de développement intégré (IDE) de Matlab. Voici quelques éléments clés de cet environnement :

Command Window (fenêtre de commande) : C'est là où vous pouvez entrer des commandes directement et voir les résultats immédiats.

Editor (éditeur) : Utilisé pour écrire, éditer et exécuter des scripts et des fonctions Matlab.

Workspace (espace de travail) : Affiche les variables actuellement définies ainsi que leurs valeurs.

Current Folder (dossier actuel) : Montre les fichiers et dossiers de votre répertoire de travail actuel.

Toolboxes (boîtes à outils) : Collections de fonctions supplémentaires étendues que vous pouvez utiliser pour des applications spécifiques comme la statistique, le traitement du signal, etc.

1.4 Exemple de Script Matlab :

Voici un exemple simple de script Matlab qui calcule et affiche la somme de deux nombres :

```
% Définition des variables
a = 5;
b = 7;
% Calcul de la somme
somme = a + b;
% Affichage du résultat
disp(['La somme de ', num2str(a), ' et ', num2str(b), ' est ', num2str(somme)]);
```

Commentaires : Les lignes commençant par % sont des commentaires.

disp : Fonction pour afficher du texte et des variables.

num2str : Fonction pour convertir des nombres en chaînes de caractères.

1.5 Syntaxe de base.

La syntaxe de base de Matlab est essentielle pour écrire des scripts et des fonctions qui effectuent des calculs numériques, manipulent des données et créent des visualisations.

1.5.1 Déclaration de variables.

En Matlab, vous pouvez déclarer des variables pour stocker des valeurs numériques, des vecteurs, des matrices, des chaînes de caractères, etc. La déclaration se fait simplement en utilisant le nom de la variable suivi d'un signe égal = et de la valeur ou de l'expression à assigner. Par exemple :

```
a = 5 ; % Déclaration d'une variable scalaire
v = [1, 2, 3, 4, 5] ; % Déclaration d'un vecteur
A = [1, 2, 3 ; 4, 5, 6 ; 7, 8, 9] ; % Déclaration d'une matrice
```

1.5.2 Opérations arithmétiques

Les opérations arithmétiques de base peuvent être effectuées de manière intuitive en Matlab, notamment l'addition (+), la soustraction (-), la multiplication (*), et la division (/). Matlab prend également en charge les opérations sur les matrices et les vecteurs de manière efficace. Par exemple

```
b = a + 3 ; % Addition
c = a * b ; % Multiplication
d = sin(a) ; % Fonction trigonométrique
```

1.5.3 Fonctions intégrées.

Matlab propose un large éventail de fonctions intégrées pour effectuer des opérations mathématiques avancées, des transformations de données et des manipulations de matrices. Ces fonctions couvrent des domaines tels que l'algèbre linéaire, le traitement du signal, l'optimisation, et bien d'autres/ Par exemples

```
sin(a) ; %
cos(a) ; %
```

exp(a) ; %

log(a) ; %

sqrt(a) ; %

det(A) ; % Calcul du déterminant d'une matrice

eig(A) ; % Calcul des valeurs propres et vecteurs propres d'une matrice

reshape(b, 1, 3) ; % Réorganisation d'un vecteur en une matrice

1.6 Manipulation de matrices et de vecteurs

1.6.1 Accès aux éléments

A(2, 3) ; % Accès à l'élément à la deuxième ligne, troisième colonne de A

V(1 :3) ; % Accès aux trois premiers éléments du vecteur V

1.6.2 Opérations matricielles

B = A' ; % Transposition de la matrice A

C = A * B ; % Multiplication matricielle

1.7 Fonctions et scripts :

1.7.1 Fonctions.

function y = myFunction(x)

y = x^2 + 3*x + 1 ;

end

1.7.2 Scripts.

Les scripts Matlab sont des fichiers textes avec l'extension .m qui contiennent une série de commandes Matlab organisées de manière séquentielle. Ces scripts permettent aux utilisateurs de regrouper des instructions Matlab pour automatiser des tâches répétitives ou pour réaliser des calculs complexes de manière cohérente.

- Les scripts Matlab peuvent définir des variables pour stocker des données numériques, des tableaux, des chaînes de caractères, etc.
- Les lignes de code dans un script Matlab sont exécutées dans l'ordre où elles apparaissent, de haut en bas.
- Les scripts peuvent contenir des définitions de fonctions Matlab pour encapsuler des algorithmes spécifiques et améliorer la réutilisation du code.
- Les commentaires commencent par le symbole % et peuvent être utilisés pour documenter le code, fournir des explications ou désactiver temporairement des lignes de code.
- Les scripts Matlab peuvent inclure des instructions pour gérer les erreurs ou les cas particuliers, améliorant ainsi la robustesse du programme.

Les scripts sont utiles pour automatiser des séquences d'opérations, ce qui permet de gagner du temps et d'assurer la cohérence des résultats. Ils facilitent le prototypage rapide d'algorithmes et de méthodes, en permettant aux utilisateurs de tester et de modifier rapidement des idées. Les scripts peuvent appeler des fonctions et des scripts externes pour une modélisation et une simulation complexes.

1.8 Outils et boîtes à outils (Toolboxes).

Matlab propose un écosystème riche en fonctionnalités grâce à ses boîtes à outils spécialisées, qui étendent significativement ses capacités de base. Il est livré avec plusieurs boîtes à outils (toolboxes) qui fournissent des fonctions et des outils spécifiques à divers domaines scientifiques et techniques. Ces boîtes à outils permettent aux utilisateurs d'approfondir et d'élargir leur utilisation de Matlab dans des applications spécialisées.

La boîte à outils de l'algèbre linéaire **(Linear Algebra Toolbox)** de Matlab est fondamentale pour les opérations sur les matrices et les vecteurs. Elle inclut une gamme étendue de fonctions pour la résolution de systèmes linéaires, la factorisation de matrices, les valeurs propres et vecteurs propres, ainsi que d'autres opérations algébriques avancées. Quelques exemples de fonctions clés incluent :

inv : Calcul de l'inverse d'une matrice.

det : Calcul du déterminant d'une matrice.

eig : Calcul des valeurs propres et vecteurs propres d'une matrice.

lu, qr, svd : Factorisations LU, QR et SVD de matrices.

les utilisateurs peuvent tirer parti de la puissance de Matlab pour résoudre une gamme diversifiée de problèmes impliquant des matrices, qu'il s'agisse de calculs d'algèbre linéaire, de traitement de signaux, d'analyse statistique avancée, ou de manipulation d'images. Cette approche intégrée permet aux professionnels de résoudre efficacement des problèmes complexes tout en capitalisant sur la polyvalence et la robustesse de Matlab comme plateforme de calcul numérique.

1.9 Ressources supplémentaires :

Documentation officielle Matlab : Une ressource précieuse pour apprendre et référencer les fonctions et syntaxes de Matlab.

Communauté Matlab : Forums et groupes en ligne où vous pouvez poser des questions et trouver des solutions.

CHAPITRE 2

Matrices et systèmes d'équations linéaires

Sommaire

Dans ce chapitre, nous définissons et introduisons les outils fonctionnels de base nécessaires à la résolution des systèmes linéaires.

2.1 Quelques notions sur les matrices.

2.1.1 Définitions et Notations

Les matrices sont des tableaux de nombres organisés en lignes et colonnes, et elles sont utilisées pour représenter et résoudre des systèmes d'équations

linéaires, ainsi que pour modéliser divers phénomènes dans les sciences et l'ingénierie.

Matrice et dimensions

Une matrice de dimension $n \times m$ est un tableau rectangulaire de nombres, organisés en n lignes et en m colonnes. Chaque élément de la matrice est désigné par un double indice $a_{i,j}$, où i indique la ligne et j indique la colonne. Une matrice peut être notée de manière générale comme suit :

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & & a_{1,m} \\ a_{21} & & & \\ & & & \\ & & & \\ a_{n,1} & & & a_{n,m} \end{pmatrix} = (a_{i,j})_{\substack{i=1;n \\ j=1;m}}$$

Un vecteur ligne est une matrice composée d'une seule ligne et de plusieurs colonnes. Il est noté

$$v = (v_1, v_2, ..., v_m) = (v_{1j})_{j=1;m}$$

Un vecteur colonne est une matrice composée d'une seule colonne et de plusieurs lignes. Il est noté

$$u = \begin{pmatrix} u_1 \\ u_2 \\ . \\ . \\ u_n \end{pmatrix} = (u_{i1})_{i=1;n}$$

Types de matrices

– Matrice Carrée : Une matrice est dite carrée si le nombre de lignes est égal au nombre de colonnes $(n = m)$.

- Matrice Diagonale : Une matrice carrée est dite diagonale si tous ses éléments en dehors de la diagonale principale sont nuls.
- Matrice Identité : Une matrice identité (notée I) est une matrice carrée où tous les éléments de la diagonale principale sont égaux à 1, et tous les autres éléments sont nuls.
- Matrice Nulle : Une matrice nulle est une matrice dans laquelle tous les éléments sont égaux à zéro.
- Matrice Triangulaire : Une matrice carrée est dite triangulaire supérieure (notée U) si tous les éléments situés en dessous de la diagonale principale sont nuls, et triangulaire inférieure (notée L) si tous les éléments situés au-dessus de la diagonale principale sont nuls.

Opérations sur les Matrices

Addition et soustraction de matrices Deux matrices peuvent être additionnées ou soustraites si elles ont les mêmes dimensions. L'addition ou la soustraction se fait élément par élément.

$$\begin{aligned} A + B &= (a_{i,j})_{\substack{i=1;n \\ j=1;m}} + (b_{i,j})_{\substack{i=1;n \\ j=1;m}} \\ &= (a_{i,j} + b_{i,j})_{\substack{i=1;n \\ j=1;m}} \end{aligned}$$

$$\begin{aligned} A - B &= (a_{i,j})_{\substack{i=1;n \\ j=1;m}} - (b_{i,j})_{\substack{i=1;n \\ j=1;m}} \\ &= (a_{i,j} - b_{i,j})_{\substack{i=1;n \\ j=1;m}} \end{aligned}$$

Multiplication par un Scalaire Chaque élément de la matrice est multiplié par le même scalaire

$$\lambda A = \lambda \, (a_{i,j})_{\substack{i=1;n \\ j=1;m}} = (\lambda a_{i,j})_{\substack{i=1;n \\ j=1;m}}$$

Multiplication de Matrices. Deux matrices A et B peuvent être multipliées si le nombre de colonnes de A est égal au nombre de lignes de B. Le produit de A et B est une nouvelle matrice C où chaque élément est obtenu en multipliant les éléments des lignes de A par les éléments des colonnes de B et en additionnant les produits.

$$(a_{i,j})_{\substack{i=1;n\\j=1;m}} (b_{i,j})_{\substack{i=1;m\\j=1;l}} = \left(\sum_{k=1}^{n} a_{i,k} b_{k,j}\right)_{\substack{i=1;n\\j=1;l}}$$

Transposition d'une Matrice La transposition d'une matrice A est une nouvelle matrice A^T obtenue en échangeant les lignes et les colonnes de A.

$$a^T_{i,j} = a_{j,i}$$

2.1.2 Propriétés des Opérations Matricielles

- Commutativité de l'Addition

$$A + B = B + A$$

- Associativité de l'Addition

$$A + (B + C) = (A + B) + C$$

- Distributivité de la Multiplication par un Scalaire

$$\lambda (A + B) = \lambda A + \lambda B$$

- Associativité de la Multiplication

$$A (BC) = (AB) C$$

- Distributivité de la Multiplication sur l'Addition

$$A (B + C) = AB + AC$$

– Transposition d'une Somme

$$(A+B)^T = A^T + B^T$$

– Transposition d'un Produit

$$(AB)^T = B^T A^T$$

2.1.3 Matrices Carrées

– Pour une matrice carrée A de dimension $n \times n$, si une matrice B existe telle que $AB = BA = I$, où I est la matrice identité de dimension $n \times n$, alors B est l'inverse de A, et on note $B = A^{-1}$.

– Pour calculer l'inverse d'une matrice, plusieurs méthodes sont utilisées, notamment la méthode par adjointe et la méthode par réduction de Gauss-Jordan.

– Une matrice est dite inversible si elle possède un inverse.

– Les matrices inversibles sont également appelées matrices non singulières ou régulières, tandis que les matrices non inversibles sont dites singulières.

– Une matrice symétrique est une matrice carrée qui est égale à sa transposée

$$A = A^T$$

– Une matrice antisymétrique est une matrice carrée pour laquelle la transposée est égale à l'opposée de la matrice d'origine.

$$A = -A^T$$

– Une matrice orthogonale est une matrice carrée pour laquelle la transposée est égale à l'inverse de la matrice elle-même.

$$AA^T = A^T A = I$$

2.1.4 Déterminant et Trace

Déterminant.

Le déterminant est une valeur scalaire associée à une matrice carrée. Pour une matrice A de dimension $n \times n$, le déterminant $det(A)$ est calculé comme une combinaison linéaire des éléments de la matrice selon certaines règles. Le déterminant mesure l'extensibilité de la matrice, c'est-à-dire comment les transformations linéaires effectuées par cette matrice étirent ou compressent l'espace dans lequel elles opèrent.

- Le déterminant d'une matrice est nul si et seulement si la matrice est singulière

$$A \text{ inverssible} \Leftrightarrow \det(A) \neq 0$$

- Le déterminant est une fonction linéaire alternée des colonnes (ou des lignes) de la matrice.
- Si A et B sont deux matrices carrées de même dimension, alors

$$\det(AB) = \det(A)\det(B)$$

Trace

La trace d'une matrice est la somme de ses éléments diagonaux. Pour une matrice carrée A de dimension $n \times n$ est

$$Tra\left((a_{i,j})_{i,j=1;n}\right) = \sum_{i=1}^{n} a_{ii}$$

- La trace est une opération linéaire

$$Tra(A+B) = Tra(A) + Tra(B)$$

- Si A et B sont deux matrices carrées de même dimension, alors

$$Tra(AB) = Tra(BA)$$

2.2 Systèmes d'équations linéaires

Un système d'équations linéaires est un ensemble d'équations linéaires impliquant des variables inconnues. Ces systèmes sont couramment rencontrés en mathématiques, en ingénierie, en physique et dans d'autres domaines où des relations linéaires entre des quantités sont présentes.

2.2.1 Définitions

Définition 2.2.1 On appelle n système d'équations linéaires à m inconnues (x_j) tout système d'équations de la forme :

$$\begin{cases} a_{11}x_1 & +a_{12}x_2 & +\cdots & +a_{1m}x_m & = & b_1 \\ a_{12}x_1 & +a_{22}x_2 & +\cdots & +a_{2m}x_m & = & b_2 \\ \cdots & \cdots & \cdots & & = & \ldots \\ a_{n1}x_1 & +a_{n2}x_2 & +\cdots & +a_{nm}x_m & = & b_n \end{cases} \tag{2.1}$$

a_{ij} sont les coefficients de système, b_i sont les composants de seconde nombre (constante) pour $i = 1; n$ et $j = 1; m$.

Un système d'équations linéaires peut être représenté sous forme matricielle. Considérons un système de n équations linéaires à m inconnues. Il peut être écrit sous forme matricielle comme suit

$$Ax = b$$

où

A est une matrice $n \times m$ des coefficients des variables

$$A = (a_{ij})_{\substack{i=1;n \\ j=1;m}}$$

, x est un vecteur colonne $m \times 1$ des variables inconnues

$$x = (x_j)_{j=1;m}$$

, b est un vecteur colonne $n \times 1$ des termes constants de chaque équation.

$$b = (b_i)_{i=1;n}$$

Remarque 2.2.1 Les systèmes linéaires peuvent être représentés graphiquement en utilisant des droites, des plans ou des hyperplans en fonction du nombre de variables.

Définition 2.2.2 La solution x du système 2.1 est un vecteur colonne contenant les valeurs des variables inconnues qui satisfont toutes les équations simultanément.

Remarque 2.2.2 Un système d'équations linéaires peut avoir une unique solution, aucune solution

2.2.2 Classification des systèmes linéaires

Les systèmes linéaires peuvent être classifiés en fonction du nombre de solutions qu'ils possèdent.

- Un système d'équations linéaires est dit compatible déterminé s'il possède une unique solution (A inversible).
- Un système d'équations linéaires est dit compatible indéterminé s'il possède une infinité de solutions(A non inversible et de rang maximal).
- Un système d'équations linéaires est dit incompatible s'il n'a pas de solution (A non inversible et de non rang maximal)

2.3 Décomposition Matricielle

2.3.1 Décomposition LU

La décomposition LU (Lower-Upper) est une méthode de factorisation d'une matrice A en deux matrices L (triangulaire inférieure) et U (triangulaire supérieure), telles que

$$A = LU.$$

Cette décomposition est souvent utilisée pour résoudre des systèmes d'équations linéaires et inverser des matrices de manière efficace.

Algorithme

$$\begin{aligned} A &= \left(\prod_{k=1}^{n-1} L_k^{-1}\right) A^{n-1} \\ &= LU \end{aligned}$$

où

$$\begin{cases} A^{(0)} = A \\ A^{(k)} = L_k A^{(k-1)} \end{cases}$$

où

$$L_k = \begin{pmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & & & \\ & & 1 & & \\ & & -l_{k+1,k} & & \\ & & -l_{k+2,k} & & \\ & & & 1 & \\ 0 & & -l_{n,k} & & 1 \end{pmatrix}, k = 1, n-1$$

où

$$l_{i,k} = \frac{a_{i,k}^{(k-1)}}{a_{k,k}^{(k-1)}}$$

- Sachant que l'inverse d'une matrice triangulaire inférieure est aussi une matrice triangulaire inférieure et que le produit de deux matrices triangulaires inférieures est encore une matrice triangulaire inférieure, L est donc une matrice triangulaire inférieure.
- Il est nécessaire que $a_{k,k}^{(k-1)} \neq 0$ à chaque itération, Si, au cours du calcul, ce cas de figure venait à se produire, il faut intervertir à la nième ligne avec une autre pour pouvoir continue (il est toujours possible de trouver un élément non nul sur la colonne qui pose un problème car la matrice est inversible).

2.3.2 Décomposition $M - N$

La décomposition $A = M - N$ est une approche générale utilisée dans les méthodes itératives pour résoudre les systèmes d'équations linéaires $Ax = b$. Cette décomposition est utile pour analyser et construire les matrices d'itération pour différentes méthodes telles que Jacobi, Gauss-Seidel et SOR.

La décomposition $A = M - N$ consiste à séparer la matrice A en deux matrices M et N de manière à faciliter le processus itératif. La forme générale de l'itération est donnée par :

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b$$

La matrice d'itération est donc

$$M_I = M^{-1}N$$

la convergence de la méthode itérative dépend du rayon spectral de M_I

2.4 Valeur propre, rayon spectral

2.4.1 Valeur propre

Les valeurs propres d'une matrice jouent un rôle central dans de nombreux domaines des mathématiques appliquées, notamment dans la résolution des systèmes d'équations linéaires, les transformations linéaires, et les analyses spectrales.

Définition 2.4.1 Soit A une matrice carrée $n \times n$. Un scalaire λ est une valeur propre de A si et seulement si il existe un vecteur non nul v (appelé vecteur propre) tel que

$$Av = \lambda v$$

Autrement dit, λ est une valeur propre de A s'il existe une solution non triviale v à l'équation

$$(A - \lambda I)\, v = 0$$

où I est la matrice identité de même dimension que A.

Les valeurs propres de A peuvent être trouvées en résolvant l'équation caractéristique

$$\det(A - \lambda I) = 0$$

Cette équation est un polynôme en λ de degré n. Les solutions de ce polynôme sont les valeurs propres de A.

Remarque 2.4.1 Une matrice $n \times n$ a n valeurs propres (en comptant les multiplicités).

Remarque 2.4.2 Si A a n vecteurs propres linéairement indépendants, alors A est diagonalisable.

2.4.2 Rayon spectral

Le rayon spectral d'une matrice est une mesure de la convergence des méthodes itératives pour résoudre des systèmes d'équations linéaires. Il joue un rôle crucial dans l'analyse de la convergence des méthodes numériques. En assurant que le rayon spectral de la matrice d'itération est inférieur à 1, on peut garantir la convergence de la méthode JOR. Le calcul et l'analyse du rayon spectral permettent de choisir des paramètres optimaux pour les méthodes itératives et d'assurer une convergence rapide et efficace.

Définition 2.4.2 Le rayon spectral $\rho(A)$ d'une matrice A est défini comme le maximum des valeurs absolues de ses valeurs propres.

$$\rho(A) = \max\{|\lambda| : \lambda \text{ est une valeur propre de } A\}$$

2.4.3 Matrices d'itération et convergence

considérons une méthode itérative générique pour résoudre le système $Ax = b$. L'itération peut être exprimée sous la forme

$$x^{(k+1)} = Mx^{(k)} + c$$

où M est la matrice d'itération.

Proposition 2.1 *La suite des approximations $x^{(k)}$ converge vers la solution exacte x^* si*

$$\rho(M) < 1$$

Le fait que $\rho(M) < 1$ pour une matrice strictement diagonale dominante découle de la propriété que ces matrices assurent que l'effet de l'itération réduit l'erreur à chaque étape.

CHAPITRE 3

Méthodes itératives pour résoudre des systèmes linéaires.

Sommaire

Les méthodes itératives sont des techniques utilisées pour trouver des solutions approximatives aux systèmes d'équations linéaires. Elles sont particulièrement avantageuses pour les grands systèmes où les méthodes directes, telles que la décomposition LU et l'élimination de Gauss, deviennent inefficaces. Dans ce chapitre, nous allons détailler ces méthodes, ce qui nous sera utile dans le dernier chapitre.

3.1 Méthode de Jacobi

La méthode de Jacobi est une méthode itérative pour résoudre des systèmes d'équations linéaires de la forme $Ax = b$. Cette méthode repose sur la décomposition de la matrice A et sur des approximations successives pour atteindre la solution.

3.1.1 Principe de la Méthode de Jacobi

La méthode de Jacobi n'est utilisable que si $a_{ii} \neq 0$ pour $i = 1, n$, A partir d'un vecteur $x^{(0)}$ la méthode de Jacobi consiste à construire la suite de vecteurs $x^{(k)}$ de la manière suivante :

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right), 1 \leq i \leq n \tag{3.1}$$

où :

- $x^{(k)}$ est le vecteur solution après k itérations.
- $a_{i,j}$ sont les éléments de la matrice A.
- b_i sont les éléments du vecteur b.

Pour écrire la relation (3.1) sous forme matricielle

$$x^{(k+1)} = D^{-1} (E + F) x^{(k)} + D^{-1} b \tag{3.2}$$

, on décompose la matrice A sous la forme

$$A = D - E - F \tag{3.3}$$

où D, E et F sont les matrices définies par

$$\begin{cases} D = (d_{i,j}) : \begin{cases} d_{i,j} = 0 & si \quad i \neq j \\ d_{i,i} = a_{i,i} & si \quad i = j \end{cases} \\ E = (e_{i,j}) : \begin{cases} e_{i,j} = -a_{i,j} & si \quad i > j \\ e_{i,j} = 0 & si \quad i \leq j \end{cases} \\ F = (f_{i,j}) : \begin{cases} f_{i,j} = 0 & si \quad i \geq j \\ e_{i,j} = -a_{i,j} & si \quad i < j \end{cases} \end{cases} \tag{3.4}$$

La matrice M_J de la méthode de Jacobi est donnée par

$$M_J = D^{-1}(E + F)$$

où D, E et F sont définies par (3.4). . Les coefficients de J sont donc donnés par

$$(M_J)_{i,j} = \begin{cases} -\frac{a_{ij}}{a_{ii}} & si \quad i \neq j \\ 0 & si \quad i = j \end{cases}$$

3.1.2 Convergence de la Méthode de Jacobi

Théorème 3.1.1 *Soit A une matrice à diagonale strictement dominante, alors la méthode de Jacobi pour la résolution d'un système linéaire de matrice A est convergente.*

Preuve. La matrice D définie par (3.4) est inversible puisque les coefficients diagonaux de la matrice A sont non nuls. En effet A est à diagonale strictement dominante par hypothèse d'où d'après la définition d'une matrice à diagonale strictement dominante ■

3.2 Méthode de Gauss-Seidel

La méthode de Gauss-Seidel est une méthode itérative pour résoudre les systèmes d'équations linéaires de la forme $Ax = b$, ce qui signifie qu'elle

génère une suite qui converge vers une solution de cette équation, lorsque les conditions de convergence sont satisfaites. Elle est une amélioration de la méthode de Jacobi et utilise les valeurs les plus récentes des variables pour accélérer la convergence.

3.2.1 Principe de la Méthode de Gauss-Seidel

La méthode de Gauss-Seidel met à jour chaque composante de la solution approximative x en utilisant la formule suivante :

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), 1 \leq i \leq n \tag{3.5}$$

où :

- $x^{(k)}$ est le vecteur solution après k itérations.
- $a_{i,j}$ sont les éléments de la matrice A.
- b_i sont les éléments du vecteur b.

La forme matricielle de la méthode de Gauss-Seidal est

$$x^{(k+1)} = (D - E)^{-1} F x^{(k)} + (D - E)^{-1} b \tag{3.6}$$

La matrice de la méthode de Gauss-Seidal est

$$M_{GS} = (D - E)^{-1} F$$

3.2.2 Convergence de la Méthode de Gauss-Seidel

Proposition 3.1 *La méthode de Gauss-Seidel converge si la matrice A est strictement diagonale dominante c-à-d*

$$|a_{i,i}| < \sum_{i=1}^{n} \sum_{j \neq i} |a_{i,j}|$$

Preuve. Les coeficients diagonaux de la matrice A sont non nuls car A est à diagonale strictement dominante. Il en résulte que $D-E$ où D et E sont les matrices définies par (**??**), est inversible.
La matrice M_{GS} de la matrice A est donnée par

$$M_{GS} = (D-E)^{-1}F$$

Raisonnons par l'absurde en supposant qu'il existe une valeur propre λ de M_{GS} avec $|\lambda| \geq 1$, on a

$$\det(\lambda I - M_{GS}) = \lambda^n \det\left((D-E)^{-1}\right) \det\left((D-E) - \frac{1}{\lambda}F\right) \tag{3.7}$$

On va montrer que pour $|\lambda| \geq 1$ la matrice $(D-E) - \frac{1}{\lambda}F$ est à diagonale strictement dominante.
En effet, on a d'aprés (**??**)

$$\left((D-E) - \frac{1}{\lambda}F\right)_{i,j} = \begin{cases} a_{i,j} & si \quad i \geq j \\ \frac{a_{i,j}}{\lambda} & si \quad i < j \end{cases} \tag{3.8}$$

et par conséquent, puisque la matrice A est à diagonale strictement dominante et $|\lambda| \geq 1$

$$\begin{aligned} \sum_{i>j} |a_{i,j}| + \sum_{i<j} \left|\frac{a_{i,j}}{\lambda}\right| &\geq \sum_{i>j} |a_{i,j}| + \sum_{i<j} |a_{i,j}| \\ &= \sum_{i \neq j} |a_{i,j}| \\ &> |a_{i,i}| \end{aligned}$$

On en déduit d'après (3.4) que la matrice $(D-E) - \frac{1}{\lambda}F$ est strictement dominante. La matrice $(D-E) - \frac{1}{\lambda}F$ est donc inversible pour $|\lambda| \geq 1$, on en déduit d'après (3.4) que

$$\det\left((D-E) - \frac{1}{\lambda}F\right) \neq 0$$

Ce qui est en contradiction avec le fait que λ est valeur propre de M_{GS}. En conclusion, on ne peut avoir une valeur propre λ telle que $|\lambda| \geq 1$, donc nécéssairement $\rho(M_{GS}) < 1$ et la méthode de Gauss-Seidel converge. ■

Proposition 3.2 *La méthode de Gauss-Seidel converge également si la matrice A est symétrique et définie positive. c-à-d*

$$v^T Av > 0, \forall v$$

pour tout vecteur de ligne non nul v.

Preuve. Pour une matrice A symétrique et définie positive, on peut utiliser les propriétés spectrales de A. Une matrice symétrique définie positive a des valeurs propres réelles et strictement positives. La convergence de la méthode de Gauss-Seidel peut être établie en montrant que le spectre du schéma itératif, qui dépend des valeurs propres de A, se situe dans une région qui garantit la convergence. ■

3.3 Méthode de relaxation (JOR)

La méthode de relaxation JOR (Jacobi Over-Relaxation) est une extension de la méthode de Jacobi utilisée pour résoudre les systèmes d'équations linéaires. Elle introduit un facteur de relaxation pour accélérer la convergence. La méthode JOR améliore la méthode de Jacobi en introduisant un facteur de relaxation w $(0 < \omega < 1)$

3.3.1 Principe de la méthode JOR.

La formule itérative de la méthode JOR est la suivante

$$x_i^{(k+1)} = (1-w)\, x_i^{(k)} + \frac{w}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right), 1 \leq i \leq n \tag{3.9}$$

où sous forme matriciel

$$x^{(k+1)} = (1-w)\, x^{(k)} + wD^{-1} \left(b - Rx^{(k)} \right)$$

$$R = A - D = -(E+F)$$

Cette formule peut être interprétée comme une combinaison linéaire entre la solution précédente $x^{(k)}$ et la mise à jour proposée par la méthode de Jacobi standard $x_J^{(k+1)}$

3.3.2 Convergence de la méthode (JOR)

La convergence de la méthode JOR dépend du choix du facteur de relaxation w et des propriétés de la matrice A

Proposition 3.3 *Une condition suffisante pour la convergence de la méthode JOR est que la matrice A soit strictement diagonale dominante.*

Proposition 3.4 *Si A est symétrique et définie positive, la méthode JOR converge pour un facteur de relaxation w approprié.*

3.4 Méthode de relaxation (SOR)

La méthode de relaxation successive (Successive Over-Relaxation, SOR) est une amélioration de la méthode de Gauss-Seidel pour résoudre les systèmes d'équations linéaires de la forme $Ax = b$. Elle introduit un facteur de relaxation w qui peut accélérer la convergence du processus itératif.

3.4.1 Principe de la méthode SOR.

La méthode SOR introduit un facteur de relaxation w (où $0 < w < 2$) dans la méthode de Gauss-Seidel. La formule de mise à jour pour chaque composante est :

$$x_i^{(k+1)} = (1-w)\, x_i^{(k)} + \frac{w}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), 1 \leq i \leq n \tag{3.10}$$

ce qui s'écrit vectoriellement

$$\begin{aligned} x^{(k+1)} &= \left[(D - \omega E)^{-1} \left(F + \frac{1-\omega}{\omega} D \right) F \right] x^{(k)} + \left(\frac{D}{\omega} - E \right)^{-1} b \\ &= \left[\left(\frac{D}{\omega} - E \right)^{-1} (\omega F + (1-\omega) D) F \right] x^{(k)} + (D - \omega E)^{-1} b \end{aligned}$$

- Cette formule peut être interprétée comme une combinaison linéaire entre la solution précédente $x^{(k)}$ et la mise à jour proposée par la méthode de Gaus-Seidal standard $x_{GS}^{(k+1)}$
- Pour $w = 1$, on retrouve la méthode de Gauss-Seidel.

3.4.2 Convergence de la méthode SOR.

La convergence de la méthode SOR dépend du choix approprié du facteur de relaxation w. Un choix optimal de w peut accélérer la convergence, tandis qu'un mauvais choix peut rendre la méthode divergente. Généralement, w est choisi expérimentalement ou en fonction des propriétés de la matrice A.

CHAPITRE 4

Processus stochastiques discretes.

Sommaire

Un processus stochastique discret est une séquence de variables aléatoires $(X)_{k\geq 0}$ où k prend des valeurs dans un ensemble discret, souvent $\mathbb{N}$: Chaque variable $X_k$prend ses valeurs dans un espace d'états, souvent $\mathbb{R}$ ou un ensemble fini de valeurs.

4.1 Variables aléatoires

Une variable aléatoire est une fonction qui associe un résultat d'une expérience aléatoire à un nombre réel. Elle permet de quantifier les résultats possibles d'un phénomène aléatoire et de les étudier à l'aide des outils de la

théorie des probabilités. Une variable aléatoire est dite discrète si le nombre de valeurs qu'elle peut prendre est fini ou infini dénombrable.

Définition 4.1.1 La loi de probabilité d'une variable aléatoire discrete est représentée par la probabilité associée à chacune des valeurs possibles de X . Ainsi, la probabilité que la variable aléatoire X prenne la valeur k est donnée par

$$P_X(k) = P\left(X^{-1}(k)\right) = P(\omega \in \Omega \mid X(w) = k)$$

Définition 4.1.2 La fonction de répartition d'une variable aléatoire discrète X est définie pour tout $t \in \mathbb{R}$ par

$$F_X(t) = P(X \leq t)$$

Plus formellement

$$\begin{aligned} F_X(t) &= P(\omega \in \Omega \mid X(\omega) \leq t) \\ &= \sum_{k \leq t} P_X(k) \end{aligned}$$

Définition 4.1.3 (Independence) Deux variables aléatoires X et$Y$ sont indépendantes si

$$P(X \leq t, Y \leq s) = P(X \leq t) P(Y \leq s)$$

Définition 4.1.4 (Espérance) L'espérance mathématique d'une variable aléatoire discrète X à valeurs un ensemble χ est définie par

$$E(x) = \sum_{x \in \chi} xP(X = x)$$

Remarque 4.1.1 (Linéarité de l'espérance)

$$E(\lambda X + \beta Y) = \lambda E(X) + \beta E(Y)$$

Définition 4.1.5 (Variance) La variance d'une variable aléatoire discrète est définie par

$$Var\left(X\right)=\sum_{x\in\chi}\left\{x-E\left(x\right)\right\}^{2}P\left(X=x\right)$$

Remarque 4.1.2 Si X et Y deux variables aléatoitres independentes alors

$$Var\left(aX+b\right)=a^{2}Var\left(X\right)$$

Définition 4.1.6 La covariance entre deux variables aléatoires X et Y est définie par

$$Cov\left(X,Y\right)=E\left(XY\right)-E\left(X\right)E\left(Y\right)$$

Exemple 4.1 (Loi Bernoulli) Prend la valeur 1 avec probabilité p et la valeur 0 avec probabilité $1-p$.

$$P\left(X=k\right)=\begin{cases}p & si\quad k=1\\ 1-p & si\quad k=0\end{cases}$$

Exemple 4.2 (Loi binomiale) Somme de n variables de Bernoulli indépendantes et identiquement distribuées.

$$P\left(X=k\right)=C_{n}^{k}p^{k}\left(1-p\right)^{n-k}$$

Exemple 4.3 (Loi de Poisson)

$$P\left(X=k\right)=\frac{\lambda^{k}}{k!}e^{-\lambda}$$

4.2 Processus stochastiques discrets

Définition 4.2.1 Un processus stochastique discret est une suite de variables aléatoires $\left(X_{k}\right)_{k\in\mathbb{N}}$ où n prend des valeurs dans un ensemble discret,

souvent $\mathbb{N}$. Chaque variable X_k prend ses valeurs dans un espace d'états, souvent $\mathbb{R}$ ou un ensemble fini de valeurs.

$$\begin{array}{rccl} X: & \Omega \times \mathbb{N} & \rightarrow & E \\ & (\omega, n) & \rightarrow & X_n(\omega) = X(n, \omega) \end{array}$$

4.2.1 Propriétés des processus stochastiques discrets

Stationnarité :

La stationnarité est une propriété importante des processus stochastiques, qui se réfère à la constance des propriétés statistiques du processus au cours du temps. Il existe deux principaux types de stationnarité : la stationnarité faible (ou stationnarité au second ordre) et la stationnarité stricte.

Définition 4.2.2 (Stationnarité stricte) Un processus stochastique $(X_n)_n$ est dite strictement stationnaire si la distribution conjointe de $(X_{n_1}, X_{n_2}, X_{n_3}, ..., X_{n_k})$ est la même que celle de $(X_{n_1+m}, X_{n_2+m}, X_{n_3+m}, ..., X_{n_k+m})$ pour tout k, pour tout $n_1, n_2, ..., n_k$,et pour tout décalage m c-à-d

$$\begin{array}{rl} & P(X_{n_1} \leq t_1, X_{n_2} \leq t_2, X_{n_3} \leq t_3, ..., X_{n_k} \leq t_k) \\ = & P(X_{n_1+m} \leq t_1, X_{n_2+m} \leq t_2, X_{n_3+m} \leq t_3, ..., X_{n_k+m} \leq t_m) \end{array}$$

Définition 4.2.3 (Stationnarité faible) Un processus stochastique $(X_n)_n$ est dite faiblement stationnaire (ou stationnaire au second ordre) si les trois conditions suivantes sont satisfaites :

- Espérance constante

$$E(X_n) = m, \forall n$$

- Variance constante

$$Var(X_n) = \sigma^2, \forall n$$

- Covariance dépend seulement du décalage

$$Cov\left(X_n, X_{n+m}\right) = \gamma\left(m\right)$$

Remarque 4.2.1 Pour un processus faiblement stationnaire, l'auto-covariance $\gamma\left(m\right)$ ne dépend que du décalage m, l'auto-corrélation est la normalisation de l'auto-covariance

$$\rho\left(m\right) = \frac{\gamma\left(m\right)}{\gamma\left(0\right)}$$

Remarque 4.2.2 La stationnarité assure que les propriétés statistiques du processus restent constantes au cours du temps.

4.3 Marche aléatoire

La marche aléatoire est un modèle de processus stochastique qui décrit une succession de pas aléatoires sur une ligne, un plan ou dans l'espace, est un modèle simple mais puissant pour représenter des processus évoluant de manière aléatoire.

Définition 4.3.1 Une marche aléatoire est un processus stochastique discrete $\left(W_n\right)_n$, où chaque W_n représente la position après n pas, le processus commence à une position initiale W_0 et chaque pas est défini par une variable aléatoire Z_n représentant le déplacement à l'étape n, la position après n pas est donnée par :

$$W_n = W_{n-1} + Z_n = W_0 + \sum_{k=1}^{n} Z_k$$

CHAPITRE 5

Une méthode de relaxation stochastiquement adaptée.

Sommaire

Les méthodes itératives ont démontré leur efficacité et leur importance dans la résolution de systèmes d'équations linéaires, notamment pour les systèmes de grande taille où les méthodes directes deviennent impraticables. Les méthodes classiques comme Jacobi, Gauss-Seidel, et Successive Over-Relaxation (SOR) offrent des solutions fiables et relativement simples

à implémenter.

La méthode de Jacobi utilise des décompositions simples mais peut souffrir de convergence lente, surtout en l'absence de stricte dominance diagonale. La méthode de Gauss-Seidel améliore Jacobi en utilisant les valeurs les plus récentes à chaque étape, offrant une convergence généralement plus rapide pour les systèmes appropriés. La méthode de relaxation introduit un facteur de relaxation pour optimiser la convergence, souvent plus efficace que Gauss-Seidel en termes de rapidité de convergence.

Tout ce qui a été mentionné dans les chapitres précédents constitue un compte rendu des concepts mathématiques les plus importants que nous avons jugés nécessaires, ainsi qu'une justification pour ce que nous allons approfondir dans ce chapitre. Au cours de celui-ci, qui représente l'essentiel de notre recherche, nous proposerons une méthode itérative différente de celles présentées dans le deuxième chapitre. Nous viserons à combler les lacunes identifiées dans ces méthodes et à améliorer leurs performances en termes de rapidité et de précision.

Les processus stochastiques et les marches aléatoires sont deux concepts de la théorie des probabilités qui introduisent des éléments nouveaux par rapport aux modèles classiques. Pour résoudre des systèmes d'équations linéaires, nous avons trouvé un moyen d'exploiter ces concepts en remplaçant le coefficient de relaxation fixe w par un coefficient variable aléatoirement W_k. Cette approche innovante permet d'adapter dynamiquement le processus itératif, ce qui pourrait potentiellement améliorer la convergence et la précision des solutions obtenues.

5.1 Formule Itérative de la méthode SRA

$$x_i^{(k+1)} = \left(1 - W^{(k)}\right) x_i^{(k)} + \frac{W^{(k)}}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), 1 \leq i \leq n \tag{5.1}$$

ce qui s'écrit vectoriellement

$$\begin{aligned} x^{(k+1)} &= \left[\left(\frac{D}{W_k} - E \right)^{-1} \left(F + \frac{1 - W_k}{W_k} D \right) F \right] x^{(k)} + \left(\frac{D}{\omega} - E \right)^{-1} b x^{(k+1)} \\ &= \left[\left(\frac{D}{W_k} - E \right)^{-1} \left(F + \frac{1 - W_k}{W_k} D \right) F \right] x^{(k)} + \left(\frac{D}{\omega} - E \right)^{-1} b \end{aligned}$$

$$\begin{aligned} M_{SRA} &= \frac{D}{W_k} - E \\ N_{SRA} &= \left(F + \frac{1 - W_k}{W_k} D \right) F \end{aligned}$$

où $(W_k)_k$ est un precessus aléatoire discret (sera définie ulteriorment).

5.2 Principe de la méthode SRA

Initialisation :

- Choisissez une estimation initiale $x^{(0)}$.
- Choisissez un facteur de relaxation w_0 (w_0 est une valeur aléatoire de la variable aléatoire W_0, dont nous supposons qu'elle suit une loi uniforme sur le domaine $]0, 2[$.
- Calculer le résidu

$$R^{(0)} = \left\| b - A x^{(0)} \right\|$$

- Pour chaque itération k, et pour chaque variable i de 1 à n

$$x_i^{(k+1)} = (1 - w_k) x_i^{(k)} + \frac{w_k}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), 1 \leq i \leq n$$

où

– Si la différence entre deux itérations successives $\left\|x_i^{(k+1)} - x^{(k)}\right\|$soit inférieure à un seuil de tolérance ε alors $x_i^{(k+1)}$ est la solution aprocher du système $Ax = b$

– Calculer le résidu :

$$R^{(k+1)} = \left\|b - Ax^{(k+1)}\right\|$$

– Mettre à jour w_k :

$$w_{k+1} = w_k + z_k$$

où z_k est une valeur aléatoire de la variable aléatoire Z_k dont nous supposons qu'elle suit une loi uniforme définie comme suit :

$$Z_k \in \begin{cases} \left]\max_{w_j \leq w_k}(w_j) - w_k, 0\right[& si \quad R^{(k+1)} > R^{(k)} \\ \left]0, \min_{w_j \leq w_k} -w_k\right[& si \quad R^{(k+1)} \leq R^{(k)} \end{cases}$$

5.3 Conditions de Convergence

5.3.1 Matrice symétrique et définie positive

La méthode SRA converge pour toute matrice A symétrique et définie positive pour

$$W_k \in]0, 2[$$

5.3.2 Matrice strictement diagonale dominante

Si A est une matrice strictement diagonale dominante, c'est-à-dire que pour chaque i

$$|a_{ii}| < \sum_{j \neq i} |a_{ij}|$$

alors la méthode SRA converge pour tout

$$W_k \in]1, 2[$$

5.4 Spectre des valeurs propres

La méthode SRA converge également si les valeurs propres du système itératif sont toutes inférieures en module à 1. Le facteur de relaxation aléatoire W joue un rôle crucial dans le positionnement des valeurs propres.

5.5 Analyse de Convergence

La méthode converge si et seulement si le rayon spectral de T_{SRA} est inférieur à 1

$$\rho\left(T_{SRA}\right) = \rho\left(\left(\frac{D}{W_k} - E\right)^{-1}\left(F + \frac{1 - W_k}{W_k}D\right)F\right) < 1$$

5.6 Exemple d'application

Considérons le système d'équations linéaires suivant :

$$\left\{\begin{pmatrix} 4 & 1 & -1 & 1 \\ 2 & 5 & 1 & 1 \\ 0.5 & -1 & 2 & 0 \\ 2 & 1 & 0.75 & -4 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 1 \\ 2.5 \end{pmatrix}\right. \tag{5.2}$$

$$A = \begin{pmatrix} 4 & 1 & -1 & 1 \\ 2 & 5 & 1 & 1 \\ 0.5 & -1 & 2 & 0 \\ 2 & 1 & 0.75 & -4 \end{pmatrix}$$

$$b = \begin{pmatrix} 2 \\ 1 \\ 1 \\ 2.5 \end{pmatrix}$$

$$\begin{cases} D = \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -4 \end{pmatrix} \\ E = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 \\ -0.5 & 1 & 0 & 0 \\ -2 & -1 & -0.75 & 0 \end{pmatrix} \\ F = \begin{pmatrix} 0 & -1 & 1 & -1 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{cases}$$

La solution exacte est :

$$x^* = A^{-1}b = \begin{pmatrix} 0.657\,32 \\ -7.\,134\,1 \times 10^{-2} \\ 0.3 \\ -0.257\,93 \end{pmatrix}$$

Nous allons appliquer la méthode SOR et la méthode SRA adaptative pour résoudre ce système et comparer leurs performances.

Pour la méthode SOR, choisissons un facteur de relaxation initial

$$w_0 = 1.2$$

et un poit intial

$$x^{(0)} = \begin{pmatrix} 0.7 \\ 0.2 \\ 0 \\ -1.3 \end{pmatrix}$$

Utilisons l'algorithme SOR pour itérer

$$\begin{cases} x_1^{(k+1)} = (1-w_0)\, x_1^{(k)} + \frac{w_0}{a_{11}} \left(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - a_{14}x_4^{(k)} \right) \\ x_2^{(k+1)} = (1-w_0)\, x_2^{(k)} + \frac{w_0}{a_{22}} \left(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - a_{24}x_4^{(k)} \right) \\ x_3^{(k+1)} = (1-w_0)\, x_3^{(k)} + \frac{w_0}{a_{33}} \left(b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)} - a_{34}x_4^{(k)} \right) \\ x_4^{(k+1)} = (1-w_0)\, x_3^{(k)} + \frac{w_0}{a_{44}} \left(b_4 - a_{41}x_1^{(k)} - a_{42}x_2^{(k)} - a_{43}x_3^{(k)} \right) \end{cases}$$

Pour la méthode SRA adaptative, nous commençons avec un

$$w_0 = 1.2$$

initial et ajustons dynamiquement et aléatoirement w_k à chaque itération. Supposons que nous utilisons un mécanisme simple pour ajuster w basé sur l'erreur relative et un processus stochastique discret suit une loi uniforme.

Avec un point initial

$$x^{(0)} = \begin{pmatrix} 0.7 \\ 0.2 \\ 0 \\ -1.3 \end{pmatrix}$$

À chaque itération, nous calculons $x^{(k+1)}$ en utilisant l'équation SOR avec w_k , puis ajustons w_{k+1} en fonction de la convergence observée.

5.7 Comparaison des Résultats

Les résultats sont présentés après quelques itérations pour les deux méthodes.

5.7.1 Itérations de la Méthode SOR

k	$x^{(k)}$	$\left\|b - Ax^{(k)}\right\|$	$\left\|x^{(k)} - x^{(k-1)}\right\|$
0	$\begin{pmatrix} 0.7 \\ 0.2 \\ 0 \\ -1.3 \end{pmatrix}$	4.394 6	
1	$\begin{pmatrix} 0.79 \\ 0.176 \\ 0.51 \\ -0.01 \end{pmatrix}$	2.160 5	1. 390 3
2	$\begin{pmatrix} 0.545\,2 \\ -0.294\,4 \\ 0.366\,6 \\ -0.106\,45 \end{pmatrix}$	1. 642 7	0.557 74
3	$\begin{pmatrix} 0.721\,20 \\ -2.525\,2 \times 10^{-2} \\ 0.186\,48 \\ -0.407\,43 \end{pmatrix}$	0.780 64	0.475 87
4	$\begin{pmatrix} 0.641\,51 \\ -4.\,809\,8 \times 10^{-2} \\ 0.331\,19 \\ -0.201\,41 \end{pmatrix}$	0.274 63	0.265 06

k	$x^{(k)}$	$\left\|b - Ax^{(k)}\right\|$	$\left\|x^{(k)} - x^{(k-1)}\right\|$
5	$\begin{pmatrix} 0.645\,91 \\ -8.945\,2 \times 10^{-2} \\ 0.312\,45 \\ -0.264\,72 \end{pmatrix}$	$0.141\,06$	$7.803\,1 \times 10^{-2}$
6	$\begin{pmatrix} 0.670\,8 \\ -6.360\,2 \times 10^{-2} \\ 0.290\,07 \\ -0.266\,04 \end{pmatrix}$	$0.101\,48$	$4.231\,2 \times 10^{-2}$

5.7.2 Itérations de la Méthode SRA

k	$x^{(k)}$	w_k	$\left\|b - Ax^{(k)}\right\|$	$\left\|x^{(k)} - x^{(k-1)}\right\|$
0	$\begin{pmatrix} 0.7 \\ 0.2 \\ 0 \\ -1.3 \end{pmatrix}$	1.2	4. 394 6	
1	$\begin{pmatrix} 0.79 \\ 0.176 \\ 0.51 \\ -0.01 \end{pmatrix}$	1.24	2. 160 5	1. 390 3
2	$\begin{pmatrix} 0.537\,04 \\ -0.310\,08 \\ 0.361\,82 \\ -0.109\,67 \end{pmatrix}$	1.28	1. 744 6	0.576 33
3	$\begin{pmatrix} 0.739\,73 \\ 3.307\,5 \times 10^{-3} \\ 0.168\,39 \\ -0.437\,98 \end{pmatrix}$	1.36	1.0036	0.533 38

k	$x^{(k)}$	w_k	$\left\|b - Ax^{(k)}\right\|$	$\left\|x^{(k)} - x^{(k-1)}\right\|$
4	$\begin{pmatrix} 0.618\,74 \\ -5.\,827\,5 \times 10^{-2} \\ 0.370\,12 \\ -0.145\,25 \end{pmatrix}$	1.53	0.514 05	0.267 24
5	$\begin{pmatrix} 0.656\,49 \\ -0.110\,59 \\ 0.287\,59 \\ -0.322\,04 \end{pmatrix}$		0.356 26	0.205 49

5.7.3 Analyse des Résultats

- La méthode SRA montre une légère amélioration en termes de convergence avec l'ajustement dynamique et aléatoire de w
- La méthode SRA offre une solution plus robuste face aux variations dans les paramètres du système.

La méthode SOR est efficace et simple, mais son succès dépend fortement du choix de w. La méthode SRA, bien que plus complexe, ajuste dynamiquement et aléatoirement w, ce qui peut offrir une meilleure performance et une convergence plus rapide. Dans des situations où le choix optimale de w est difficile à déterminer, la méthode SRA peut être une solution plus robuste et efficace.

Conclusion générale et perspectives

Ce mémoire a permis de mettre en lumière les divers aspects des méthodes itératives et des processus stochastiques appliqués à la résolution des systèmes d'équations linéaires. En explorant les méthodes existantes telles que Jacobi, Gauss-Seidel et SOR, nous avons identifié les limitations et les opportunités d'amélioration en termes de précision et de vitesse de convergence.

Nous avons proposé un nouvel algorithme qui intègre des éléments stochastiques dans le processus itératif, visant à surmonter certaines de ces limitations. Les résultats expérimentaux ont montré que notre approche peut offrir des avantages significatifs dans certaines conditions, notamment en améliorant la précision des solutions et en accélérant la convergence dans des systèmes complexes.

Cette étude a également mis en avant l'importance du choix des paramètres, tels que le facteur de relaxation dans les méthodes SOR, et la nécessité d'adapter ces paramètres de manière dynamique pour optimiser la performance de l'algorithme.

Plusieurs axes de recherche pourraient être explorés à partir de ce travail :

– Développer des techniques d'optimisation pour ajuster dynamiquement

les paramètres de relaxation et les éléments stochastiques en fonction des caractéristiques du système d'équations à résoudre.

- Étendre les principes de l'algorithme proposé aux systèmes d'équations non-linéaires et évaluer ses performances dans ce contexte.
- Tester et valider l'algorithme dans des applications pratiques et des problèmes réels, tels que les simulations en ingénierie et les modèles financiers.
- Explorer des techniques avancées de préconditionnement et des méthodes hybrides combinant des approches directes et itératives pour améliorer encore les performances de convergence.
- Mener des études théoriques plus approfondies pour analyser les propriétés de convergence de l'algorithme proposé et établir des critères rigoureux pour son utilisation optimale.

En conclusion, bien que ce mémoire ait fourni une base solide pour l'utilisation des processus stochastiques dans les méthodes itératives, il ouvre également la voie à de nombreuses recherches futures pour affiner et étendre cette approche prometteuse.

BIBLIOGRAPHIE

[1] **Haouam Wala & Lemita Nesrine,** *La convergence des méthodes itératives pour la résolution des systèmes linéaires, Mémoire de Master,* Université Larbi Tébessi - Tébessa (2020)

[2] **P.G.Ciarlet,**Introduction à l'analyse numérique matricielle et à l'optimisation. Dunod1998

[3] **D. Serre**,Les matrices.Dunod, Masson Sciences 2001.

[4] **CLEMENTINE VASSOILLES** *Proposition d'une nouvelle méthode de classification, base de copules Mémoire de Master* Université du Québec à Trois-Rivières (2014)

[5] **Dr. ZEMANI FARAH** *Polycopié Programmation et méthode numérique sous MATLAB* Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf

[1] **Laurent MAZLIAK** (2004-2005) *Elements de la theorie des probabilites :*Cours Préliminaire au DEA de Probabilités.

[10] **Monique Jeanblanc (**2006) *cours de calcul stochastique* M.Duras, L'été 80 Edition de Minuit