

# **TP4 – Résolution de systèmes tridiagonaux par la méthode de Thomas**

Réalisé par :

- Mathéo BEAUDOUIN

Encadré par :

- Mr. Cyrille Bertelle

## I) Introduction

Dans le cadre du TP4, notre travail de la résolution de systèmes linéaires en Java se poursuit avec le développement de nouvelles fonctionnalités au sein du package "AlgLin". L'objectif principal de ce travail était de mettre en œuvre la méthode de Thomas pour la résolution de systèmes tridiagonaux, telle que décrite dans nos exercices de TD.

Nous avons débuté par la création de la classe Mat3Diag, dérivée de la classe Matrice, permettant la manipulation de matrices carrées tridiagonales. Cette classe a été conçue pour garantir que le tableau-attribut des coefficients soit nécessairement un tableau à 3 lignes et n colonnes, représentant respectivement la sous-diagonale, la diagonale et la sur-diagonale du système. Nous avons enrichi cette classe en ajoutant des constructeurs appropriés et une méthode statique pour effectuer le produit d'une matrice tridiagonale par un vecteur.

Ensuite, nous avons défini la classe Thomas, dérivée de la classe abstraite SysLin, pour décrire un système tridiagonal d'ordre n. Nous avons également intégré une méthode main dans la classe Thomas pour réaliser des tests et des validations.

Dans ce compte rendu, nous présenterons les résultats des tests effectués sur les méthodes et les classes développées.

## II) Résultats et tests

### A) La classe Mat3Diag

Les trois tests effectués dans la méthode main() ont permis de valider le bon fonctionnement de cette classe. Pour le test 1, une matrice tridiagonale  $3 \times 3$  a été créée avec les coefficients  $\{0, 1, 2\}$ ,  $\{4, 5, 6\}$ ,  $\{7, 2, 0\}$ . Un vecteur  $\{1, 2, 3\}$  a également été créé. Le produit de la matrice par le vecteur a été calculé et le résultat obtenu est conforme aux attentes. Ensuite, pour le test 2, une autre matrice tridiagonale  $3 \times 5$  a été créée avec des coefficients différents. Un autre vecteur a été créé et le produit de la matrice par ce vecteur a été calculé avec succès. Pour finir, pour le test 3 une matrice tridiagonale à coefficients négatifs a été utilisée pour tester le comportement de la classe.

Voici le résultat de ces tests lors de l'exécution du programme :

```
Matrice 1
0.0 1.0 2.0
4.0 5.0 6.0
7.0 2.0 0.0

Vecteur 1
1.0
2.0
3.0

Résultat Test 1:
18.0
17.0
22.0

Matrice 2
0.0 1.0 2.0 3.0 1.0
4.0 5.0 6.0 7.0 8.0
2.0 9.0 10.0 11.0 0.0

Vecteur 2
1.0
3.0
5.0
7.0
9.0

Résultat Test 2:
10.0
61.0
106.0
163.0
79.0

Matrice 3
0.0 -1.0 -1.0 -1.0
2.0 2.0 2.0 2.0
-1.0 -1.0 -1.0 0.0

Vecteur 3
1.0
2.0
3.0
4.0

Résultat Test 3:
0.0
0.0
0.0
5.0
```

Nous constatons que les produits matrice-vecteur sont correctement calculés, conformément aux propriétés des matrices tridiagonales. Les valeurs obtenues pour chaque résultat sont cohérentes et correspondent aux attentes.

## B) La classe Thomas

Elle permet de trouver rapidement et précisément la solution à des systèmes tridiagonaux, où la matrice du système est tridiagonale, ce qui signifie qu'elle a des éléments non nuls uniquement sur la diagonale principale et les diagonales immédiatement supérieure et inférieure.

Pour le test 1, le produit de la matrice tridiagonale d'origine par la solution obtenue est pratiquement identique au second membre, avec une différence négligeable. Les normes L1, L2 et Linfini de la différence entre le produit et le second membre sont extrêmement proches de zéro, indiquant une solution précise.

```
matrice1:  
0.0 2.0 1.0  
1.0 3.0 5.0  
7.0 4.0 0.0
```

```
vecteur 1:  
7.0  
15.0  
8.0
```

```
La solution du systeme trouvée :  
3.7966101694915246  
0.4576271186440679  
1.5084745762711866
```

```
Produit de la matrice * solution :  
7.0  
15.0  
8.0000000000000002
```

```
Sachant que le second membre était :  
7.0  
15.0  
8.0
```

```
Différence (produit - second membre) :  
0.0  
0.0  
1.7763568394002505E-15
```

```
norme L1 de la différence : 1.7763568394002505E-15  
norme L2 de la différence : 1.7763568394002505E-15  
norme Linfini de la différence : 1.7763568394002505E-15
```

Pour le test 2, le produit de la matrice tridiagonale d'origine par la solution obtenue correspond quasiment au second membre. Les normes L1, L2 et Linfini de la différence

entre le produit et le second membre sont également extrêmement proches de zéro, confirmant la précision de la solution.

```
matrice 2 :  
0.0 1.0 2.0 3.0  
1.0 2.0 3.0 4.0  
6.0 4.0 1.0 0.0  
  
vecteur 2:  
5.0  
14.0  
23.0  
14.0  
  
solution du systeme trouvée :  
-15.382352941176471  
3.397058823529412  
5.647058823529412  
-0.7352941176470589  
  
produit matrice * solution :  
5.0  
14.0000000000000002  
23.0  
14.0  
  
Sachant que le second membre était  
5.0  
14.0  
23.0  
14.0  
  
différence (produit - second membre) :  
0.0  
1.7763568394002505E-15  
0.0  
0.0  
  
Norme de la différence : 1.7763568394002505E-15  
norme de la différence : 1.7763568394002505E-15  
Norme de la différence : 1.7763568394002505E-15
```

---

Les tests effectués démontrent que la classe Thomas est capable de résoudre avec précision des systèmes linéaires tridiagonaux. Les résultats obtenus confirment la fiabilité de l'algorithme de Thomas implémenté dans cette classe.

### III) Conclusion

En conclusion, ce TP4 a été une exploration dans la résolution de systèmes linéaires tridiagonaux en Java, en mettant en œuvre l'algorithme de Thomas.

Nous avons enrichi notre bibliothèque "AlgLin" avec une nouvelle classe Mat3Diag, et Thomas permettant de résoudre efficacement ce type de systèmes.

Les tests réalisés ont démontré l'efficacité de l'algorithme de Thomas dans la résolution de systèmes linéaires tridiagonaux, produisant des résultats précis.

En examinant les résultats obtenus, nous avons confirmé la fiabilité de notre implémentation.

# Annexe

Pour Mat3diag.java :

```
package AlgLin;
public class Mat3Diag extends Matrice {
    // constructeur prenant les dimensions de la matrice
    public Mat3Diag(int dim1, int dim2) throws IrregularSysLinException {
        super(dim1, dim2);

        // verification que dim1 vaut 3 pour une matrice tridiagonale
        if (dim1 != 3) {
            throw new IllegalArgumentException("dim1 doit etre egal à 3");
        }
        if (dim2 <= 0) {
            throw new IllegalArgumentException("dim2 doit etre strictement positif");
        }
    }

    // constructeur prenant un tableau de coefficients
    public Mat3Diag(double[][] tableau) throws IrregularSysLinException {
        super(tableau);
        // verification que le tableau a 3 lignes
        if (tableau.length != 3) {
            throw new IllegalArgumentException("Le tableau doit avoir 3 lignes");
        }
        if (tableau[0].length <= 0) {
            throw new IllegalArgumentException("Le tableau doit avoir au moins une colonne");
        }
    }

    // constructeur pour allouer en memoire un tableau à 3 lignes et dim colonnes pour une matrice tridiagonale
    // d ordre n
    public Mat3Diag(int dim) throws IrregularSysLinException {
        super(3, dim);
        if (dim <= 0) {
            throw new IllegalArgumentException("la taille de la matrice doit être strictement positive");
        }
    }

    // methode statique pour le produit d'une matrice tridiagonale par un vecteur
    public static Vecteur produit(Mat3Diag tridiagonale, Vecteur vecteur) {
        // Vérification des dimensions compatibles
        if (tridiagonale.nbColonne() != vecteur.taille()) {
            throw new IllegalArgumentException("dimensions incompatibles pour le produit");
        }

        // Calcul du produit :
        Vecteur resultat = new Vecteur(vecteur.taille());
    }
}
```

```
        for (int i = 0; i < vecteur.taille(); i++) {
            double somme = 0.0;
            // Diagonale principale
            somme += tridiagonale.getCoef(1, i) * vecteur.getCoef(i);
            // Sous-diagonale
            if (i > 0) {
                somme += tridiagonale.getCoef(0, i) * vecteur.getCoef(i - 1);
            }
            // Sur-diagonale
            if (i < vecteur.taille() - 1) {
                somme += tridiagonale.getCoef(2, i) * vecteur.getCoef(i + 1);
            }

            resultat.remplacerCoefficient(i, somme);
        }
        return resultat;
    }

    public static void main(String[] args) throws IrregularSysLinException {
        //Test 1 :
        double[][] matrice1 = {
            {0, 1, 2},
            {4, 5, 6},
            {7, 2, 0}
        };

        Mat3Diag mat1 = new Mat3Diag(matrice1);
        Vecteur vec1 = new Vecteur(new double[]{1, 2, 3});
        Vecteur res1 = Mat3Diag.produit(mat1, vec1);
        System.out.println("Test :");
        System.out.println("Matrice 1 :\n" + mat1);
        System.out.println("Vecteur 1 :\n" + vec1);
        System.out.println("Résultat 1 :\n" + res1);

        // Test 2 :
        double[][] matrice2 = {
            {0, 1, 2, 3, 1},
            {4, 5, 6, 7, 8},
            {2, 9, 10, 11, 0}
        };

        Mat3Diag mat2 = new Mat3Diag(matrice2);
        Vecteur vec2 = new Vecteur(new double[]{1, 3, 5, 7, 9});
        Vecteur res2 = Mat3Diag.produit(mat2, vec2);
        System.out.println("Test 2");
        System.out.println("Matrice 2 :\n" + mat2);
        System.out.println("Vecteur 2:\n" + vec2);
        System.out.println("Résultat 2:\n" + res2);
        // Test 3 :
        double[][] matrice3 = {
            { 0, -1, -1, -1},
            { 2, 2, 2, 2 },
            { -1, -1, -1, 0 }
        };
    }
```



```

    };

    Mat3Diag mat3 = new Mat3Diag(matrice3);
    Vecteur vec3 = new Vecteur(new double[]{1, 2, 3, 4});
    Vecteur res3 = Mat3Diag.produit(mat3, vec3);
    System.out.println("Test 3 : ");
    System.out.println("Matrice 3:\n" + mat3);
    System.out.println("Vecteur 3 :\n" + vec3);
    System.out.println("Résultat 3:\n" + res3);
}
}

```

Pour Thomas.java :

```

package AlgLin;

public class Thomas extends SysLin {
    public Thomas(Mat3Diag matrice, Vecteur secondMembre) throws
    IrregularSysLinException {
        super(matrice, secondMembre);
        // Vérification que la matrice est bien tridiagonale
        if (!(matrice instanceof Mat3Diag)) {
            throw new IrregularSysLinException("la matrice du système doit
être de type Mat3Diag");
        }
    }

    @Override
    public Vecteur resolution() {

        int n = this.getMatriceSystem().nbColonne();
        Vecteur p = new Vecteur(n); // pk
        Vecteur q = new Vecteur(n); // qk
        Vecteur x = new Vecteur(n); // la solution
        Mat3Diag matrice = (Mat3Diag) this.getMatriceSystem();
        Vecteur d = this.getSecondMembre();
        // initialisation des premiers coefficients p1 et q1 :
        p.remplacerCoefficient(0, -matrice.getCoeff(2, 0) / matrice.getCoeff(1,
0));
        q.remplacerCoefficient(0, d.getCoeff(0, 0) / matrice.getCoeff(1, 0));
        // Calcul des pk et qk
        for (int k = 1; k < n - 1; k++) {
            double beta = matrice.getCoeff(0, k) * p.getCoeff(k - 1, 0) +
matrice.getCoeff(1, k);
            p.remplacerCoefficient(k, -matrice.getCoeff(2, k) / beta);
            q.remplacerCoefficient(k, (d.getCoeff(k, 0) - matrice.getCoeff(0, k)
* q.getCoeff(k - 1, 0)) / beta);
        }
        // Calcul de xn, dernier element du vecteur solution
        x.remplacerCoefficient(n - 1, (d.getCoeff(n - 1, 0) -
matrice.getCoeff(0, n - 1) * q.getCoeff(n - 2, 0)) / (matrice.getCoeff(0, n - 1)
* p.getCoeff(n - 2, 0) + matrice.getCoeff(1, n - 1)));
        // remontée pour calculer xk
        for (int k = n - 2; k >= 0; k--) {

```

```

        x.remplacerCoefficient(k, p.getCoef(k, 0) * x.getCoef(k + 1, 0) +
q.getCoef(k, 0));
    }
    return x;
}

public static void main(String[] args) {
    try {

        // Test 1:
        // Définition de la matrice tridiagonale
        double[][] matrice = {{0, 2, 1}, {1, 3, 5}, {7, 4, 0}};
        Mat3Diag matrice1 = new Mat3Diag(matrice);
        System.out.println("matrice1: \n" + matrice1);
        // Définition du second membre
        double[] secondMembre = {7, 15, 8};
        Vecteur secondMembre1 = new Vecteur(secondMembre);
        System.out.println("vecteur 1: \n" + secondMembre1);
        // Résolution du système
        Thomas system1 = new Thomas(matrice1, secondMembre1);
        Vecteur solution1 = system1.resolution();
        System.out.println("La solution du systeme trouvée : \n" +
solution1);

        // Calcul du produit de la matrice tridiagonale d'origine par la
solution trouvée
        Vecteur produit1 = Mat3Diag.produit(matrice1, solution1);
        System.out.println("Produit de la matrice * solution : \n" +
produit1);
        System.out.println("Sachant que le second membre était : \n" +
secondMembre1);

        // Calcul de la différence entre le produit et le second membre
        Vecteur difference1 = Vecteur.soustraction(produit1,
secondMembre1);
        System.out.println("Différence (produit - second membre) : \n" +
difference1);
        double norme1 = difference1.normeL1(); // Norme L1 de la
différence
        System.out.println("norme L1 de la différence : " + norme1);
        double norme2 = difference1.normeL2(); // Norme L2 de la
différence
        System.out.println("norme L2 de la différence : " + norme2);
        double norme3 = difference1.normeLinfini(); // Norme Linfini de la
différence
        System.out.println("norme Linfini de la différence : " + norme3 +
"\n\n");
        System.out.println("\n test 2\n");

        // Test 2:
        // Définition de la seconde matrice tridiagonale
        double[][] matrices = {{0, 1, 2, 3}, {1, 2, 3, 4}, {6, 4, 1, 0}};
        Mat3Diag matrice2 = new Mat3Diag(matrices);

```

```
System.out.println("matrice 2 : \n" + matrice2);
// Définition du second membre pour le test 2
double[] secondMembre4 = {5, 14, 23, 14};
Vecteur secondMembre2 = new Vecteur(secondMembre4);
System.out.println("vecteur 2: \n" + secondMembre2);

// Résolution du système
Thomas system2 = new Thomas(matrice2, secondMembre2);
Vecteur solution2 = system2.resolution();
System.out.println("solution du systeme trouvée : \n" +
solution2);

// Calcul du produit de la seconde matrice tridiagonale d'origine
par la solution trouvée
Vecteur produit2 = Mat3Diag.produit(matrice2, solution2);
System.out.println("produit matrice * solution : \n" + produit2);
System.out.println("Sachant que le second membre était \n" +
secondMembre2);
// Calcul de la différence entre le produit et le second membre
Vecteur difference2 = Vecteur.soustraction(produit2,
secondMembre2);
System.out.println("différence (produit - second membre) : \n" +
difference2);
double norme4 = difference2.normeL1(); // Norme L1 de la
différence
System.out.println("Norme de la différence : " + norme4);
double norme5 = difference2.normeL2(); // Norme L2 de la
différence
System.out.println("norme de la différence : " + norme5);
double norme6 = difference2.normeLinfini(); // Norme Linfini de la
différence
System.out.println("Norme de la différence : " + norme6);
} catch (IrregularSysLinException e) {
    System.err.println(e.getMessage());
}
}
```