Glody KUTUMBAKANA

Compte Rendu Mat3Diag et Thomas

Année 2024 -2025

Introduction

Dans ce TP on nous demande d'implémenter la matrice Mat3Diag qui représente la matrice tridiagonale d'ordre n et d'implémenter la classe Thomas qui que permet la résolution de la matrice tridiagonale .

Class Mat3Diag

La classe Mat3Diag est une classe qui représente, de façon économique en mémoire, la matrice de notre système tridiagonal.

Elle hérite de la classe Matrice et possède un constructeur qui prend en paramètre un tableau de double à deux dimensions.

Elle possède également un constructeur qui prend en paramètre un entier qui représente la taille de la matrice.

La méthode taille() retourne la taille de la matrice.

La méthode produit() prend en paramètre une matrice tridiagonale et un vecteur et retourne le produit de la matrice et du vecteur

Code :

```
public class Mat3Diag extends Matrice {
  public Mat3Diag(int dim1, int dim2) throws Exception {
    super(dim1, dim2);
    if(dim1 != 3) throw new Exception("La matrice doit être de ligne 3");
  }
  public Mat3Diag(double tab[][]) throws Exception {
    super(tab);
    if(tab.length != 3) throw new Exception("La matrice doit être de ligne 3");
  }
  public Mat3Diag(int dim){
    super(3, dim);
  }
```

```
 public int taille(){
   return this.nbColonne();
 }

 public static Vecteur produit(Mat3Diag mat ,Vecteur vecteur) {
   Vecteur produit = new Vecteur(vecteur.taille());

   produit.remplacecoef(0, mat.getCoef(1, 0) * vecteur.getCoef(0) + mat.getCoef(2, 0) *
vecteur.getCoef(1));

   for(int i = 1; i < mat.nbColonne() - 1; i++){
     produit.remplacecoef(i, mat.getCoef(0, i) * vecteur.getCoef(i - 1) + mat.getCoef(1, i) *
vecteur.getCoef(i) + mat.getCoef(2, i) * vecteur.getCoef(i + 1));
   }

   produit.remplacecoef(mat.nbColonne() - 1, mat.getCoef(0, mat.nbColonne() - 1) *
vecteur.getCoef(mat.nbColonne() - 2) + mat.getCoef(1, mat.nbColonne() - 1) *
vecteur.getCoef(mat.nbColonne() - 1));

   return produit;
 }

 public static void main(String[] args) {
   double[][] coeff = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
   Mat3Diag matrice = null;

   try {
     matrice = new Mat3Diag(coeff);
   } catch (Exception e) {
     e.printStackTrace();

   }
   double[] secondMembre = {3,5,8};
```

```
    Vecteur vecteur = new Vecteur(secondMembre);
    System.out.println("Ordre considéré = " + vecteur.taille() + "\n");
    System.out.println("Matrice = \n" + matrice.toString() + "\n");
    System.out.println("Vecteur = \n" + vecteur.toString() + "\n");
    System.out.println("Produit = \n" + Mat3Diag.produit(matrice, vecteur));
  }
}
```

Class THOMAS

La classe Thomas est classe qui représente les algorithme de résolution de la classe

elle hérite de la classe abstraite SysLin qui possède un constructeur prenant en paramètre une Matrice et un Vecteur

elle redéfini la méthode résolution qui permet de résoudre la matrice de la classe Mat3Diag

Code :

```
public class Thomas extends SysLin {
  public Thomas(Mat3Diag matrice, Vecteur secondMembre) throws IrregularSysLinException{
    super(matrice, secondMembre);
  }
  @Override
  public Vecteur resolution() throws IrregularSysLinException {
    int n = getMatriceSystem().nbColonne();
    double[] p = new double[n - 1];
    double[] q = new double[n - 1];
    Vecteur res = new Vecteur(getMatriceSystem().nbColonne());
      if(Math.abs(getMatriceSystem().getCoef(1, 0)) < Matrice.EPSILON) throw new
IrregularSysLinException("Division par 0");
     p[0] = - getMatriceSystem().getCoef(2, 0) / getMatriceSystem().getCoef(1, 0);
    q[0] = getSecondMembre().getCoef(0) / getMatriceSystem().getCoef(1, 0);

    for(int i = 1; i < n - 1; i++) {
```

```
    double B = getMatriceSystem().getCoef(1, i) + getMatriceSystem().getCoef(0, i) * p[i - 1];

    if(Math.abs(B) < Matrice.EPSILON ) throw new IrregularSysLinException("Division par
0") ;

    p[i] = - getMatriceSystem().getCoef(2, i) / B;
    q[i] = (getSecondMembre().getCoef(i) - getMatriceSystem().getCoef(0, i) * q[i - 1]) / B;
   }

   double up, down;
   up = getSecondMembre().getCoef(n - 1) - getMatriceSystem().getCoef(0, n - 1) * q[n - 2];
   down = getMatriceSystem().getCoef(1, n - 1) +  getMatriceSystem().getCoef(0, n - 1) * p[n - 2];

   if(Math.abs(down) < Matrice.EPSILON ) throw new IrregularSysLinException("Division par
0") ;

   res.remplacecoef(n - 1, up/down);

   for(int i = n - 2 ; 0 <= i ; i--) {
    res.remplacecoef(i, (p[i] * res.getCoef(i + 1) + q[i]));
   }

   return res;
  }

  public static void main(String[] args) throws Exception {
   double[][] coeff = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
   Mat3Diag matrice = null;

   matrice = new Mat3Diag(coeff);

   double[] secondMembre = {3,5,8};
```

```
    Vecteur vecteur = new Vecteur(secondMembre);

    // Calcul du résultat
    Thomas thomas = new Thomas(matrice, vecteur);
    Vecteur res = thomas.resolution();

    // Calcul de la différence
    Matrice diff = Matrice.addition(Mat3Diag.produit(matrice, res), vecteur.produit(-1));
    Vecteur diff_vect = new Vecteur(3);
    for(int i = 0; i < 3; i++)
      diff_vect.remplacecoef(i, diff.getCoef(i, 0));

    // Affichage
    System.out.println("A = \n" + matrice.toString());
    System.out.println("\nd = \n" + vecteur.toString());
    System.out.println("\nx = \n" + res.toString() + "\n");
    System.out.println("la norme L1 de la différence est : " + (diff_vect.normeL1() <
Matrice.EPSILON ? 0 : res.normeL1()));
    System.out.println("la norme L2 de la différence est : " + (diff_vect.normeL2() <
Matrice.EPSILON ? 0 : res.normeL2()));
    System.out.println("la norme Linfini de la différence est : " + (diff_vect.normeLinf() <
Matrice.EPSILON ? 0 : res.normeLinf()));
  }
}
```

Résultat Obtenu

```
A =
1.0 2.0 3.0
4.0 5.0 6.0
7.0 8.0 9.0


d =
-3.0
-5.0
-8.0


x =
-4.266666666666667
2.8666666666666667
-0.1


la norme L1 de la différence est : 0.0
la norme L2 de la différence est : 0.0
la norme Linfini de la différence est : 0.0
```

```
Ordre considéré = 3

Matrice =
1.0 2.0 3.0
4.0 5.0 6.0
7.0 8.0 9.0


Vecteur =
3.0
5.0
8.0


Produit =
47.0
95.0
63.0
```