# Introduction à la Méthode des Différences Finies

## Contents


**1 Introduction à la Méthode des Différences Finies** **2**
- 1.1 Principes de Base : Discrétisation de l'Espace et du Temps . . . 2
  - 1.1.1 Schéma Forward (Avancé) . . . 2
  - 1.1.2 Schéma Backward (Retardé) . . . 2
  - 1.1.3 Schéma Centré . . . 3
  - 1.1.4 Exemple . . . 3
- 1.2 Classification des Équations Différentielles Partielles . . . 3

**2 Application de la Méthode aux Équations Elliptiques** **4**
- 2.1 Équation Elliptique 2D : Conduction Stationnaire . . . 4
  - 2.1.1 Description Physique et Formulation Mathématique . . . 4
  - 2.1.2 Discrétisation par Différences Finies . . . 4
- 2.2 Conditions aux Limites . . . 4

**3 Méthodes de Résolution** **4**
- 3.1 Solution Directe : Méthode de Gauss . . . 4
- 3.2 Solutions Itératives . . . 6
  - 3.2.1 Méthode de Jacobi . . . 6
  - 3.2.2 Méthode de Gauss-Seidel . . . 7
  - 3.2.3 Méthode de Sur-Relaxation Successive (SOR) . . . 8
  - 3.2.4 Méthodes à Pas Multiples et Stationnarisation de Douglas-Rachford . . . 9
- 3.3 Introduction à la Stationnarisation . . . 9
- 3.4 Techniques pour Accélérer la Convergence . . . 9
- 3.5 Comparaison des Performances . . . 9
- 3.6 Optimisation de la Convergence . . . 9

**4 Analyse de Convergence** **9**
- 4.1 Influence de la Discrétisation (Taille de Maillage) . . . 9
- 4.2 Application aux Systèmes Non Linéaires . . . 9

**5 Exemples Pratiques avec MATLAB** **9**
- 5.1 Simulation de la Conduction Thermique en 2D . . . 9
- 5.2 Comparaison des Différentes Méthodes de Résolution . . . 10
- 5.3 Analyse de Convergence sur Différents Maillages . . . 12

# 1 Introduction à la Méthode des Différences Finies

La Méthode des Différences Finies est une technique numérique utilisée pour résoudre les équations différentielles en approximant les dérivées à l'aide de valeurs discrètes prises à des points spécifiques sur une grille. Elle est largement utilisée dans divers domaines de l'ingénierie et des sciences pour modéliser des phénomènes physiques, tels que la conduction thermique, la diffusion et les vibrations.

## 1.1 Principes de Base : Discrétisation de l'Espace et du Temps

Le principe de base de la méthode des différences finies est de remplacer les dérivées continues par des différences finies. Par exemple, pour une fonction $u(x)$, la dérivée première peut être approximée en utilisant un schéma forward, backward ou centré.

### 1.1.1 Schéma Forward (Avancé)

Le schéma forward utilise une approximation de la dérivée en utilisant les points actuels et futurs. La dérivée première d'une fonction $u(x)$ est approximée par :

$$\frac{du}{dx} \approx \frac{u(x+\Delta x) - u(x)}{\Delta x}$$

Dérivation Pour dériver cette approximation, nous utilisons la série de Taylor :

$$u(x+\Delta x) = u(x) + \Delta x \frac{du}{dx} + \frac{(\Delta x)^2}{2!}\frac{d^2u}{dx^2} + O((\Delta x)^3)$$

En négligeant les termes d'ordre supérieur :

$$u(x+\Delta x) \approx u(x) + \Delta x \frac{du}{dx}$$

En réarrangeant pour isoler $\frac{du}{dx}$, nous obtenons :

$$\frac{du}{dx} \approx \frac{u(x+\Delta x) - u(x)}{\Delta x}$$

### 1.1.2 Schéma Backward (Retardé)

Le schéma backward utilise une approximation de la dérivée en utilisant les points actuels et passés. La dérivée première d'une fonction $u(x)$ est approximée par :

$$\frac{du}{dx} \approx \frac{u(x) - u(x-\Delta x)}{\Delta x}$$

Dérivation De même, nous utilisons la série de Taylor :

$$u(x-\Delta x) = u(x) - \Delta x \frac{du}{dx} + \frac{(\Delta x)^2}{2!}\frac{d^2u}{dx^2} + O((\Delta x)^3)$$

En négligeant les termes d'ordre supérieur :

$$u(x-\Delta x) \approx u(x) - \Delta x \frac{du}{dx}$$

En réarrangeant pour isoler $\frac{du}{dx}$, nous obtenons :

$$\frac{du}{dx} \approx \frac{u(x) - u(x-\Delta x)}{\Delta x}$$

### 1.1.3 Schéma Centré

Le schéma centré utilise une approximation de la dérivée en utilisant les points passés et futurs. La dérivée première d'une fonction $u(x)$ est approximée par :

$$\frac{du}{dx} \approx \frac{u(x+\Delta x) - u(x-\Delta x)}{2\Delta x}$$

Dérivation Utilisons la série de Taylor pour les points en avant et en arrière :

$$u(x+\Delta x) = u(x) + \Delta x \frac{du}{dx} + \frac{(\Delta x)^2}{2!}\frac{d^2u}{dx^2} + O((\Delta x)^3)$$

$$u(x-\Delta x) = u(x) - \Delta x \frac{du}{dx} + \frac{(\Delta x)^2}{2!}\frac{d^2u}{dx^2} + O((\Delta x)^3)$$

En soustrayant les deux équations :

$$u(x+\Delta x) - u(x-\Delta x) = 2\Delta x \frac{du}{dx} + O((\Delta x)^3)$$

En négligeant les termes d'ordre supérieur et en réarrangeant pour isoler $\frac{du}{dx}$ :

$$\frac{du}{dx} \approx \frac{u(x+\Delta x) - u(x-\Delta x)}{2\Delta x}$$

### 1.1.4 Exemple

Considérons la fonction $u(x) = x^2$ et calculons sa dérivée en $x = 1$ avec $\Delta x = 0.1$.

Schéma Forward

$$\frac{du}{dx} \approx \frac{u(1+0.1) - u(1)}{0.1} = \frac{1.1^2 - 1^2}{0.1} = \frac{1.21 - 1}{0.1} = 2.1$$

Schéma Backward

$$\frac{du}{dx} \approx \frac{u(1) - u(0.9)}{0.1} = \frac{1^2 - 0.9^2}{0.1} = \frac{1 - 0.81}{0.1} = 1.9$$

Schéma Centré

$$\frac{du}{dx} \approx \frac{u(1+0.1) - u(1-0.1)}{2\Delta x} = \frac{1.1^2 - 0.9^2}{2 \times 0.1} = \frac{1.21 - 0.81}{0.2} = 2$$

La dérivée exacte de $u(x) = x^2$ en $x = 1$ est $2x = 2$, donc le schéma centré donne l'approximation la plus précise.

## 1.2 Classification des Équations Différentielles Partielles

Les équations différentielles partielles (EDP) peuvent être classées en trois types principaux : elliptique, parabolique et hyperbolique. Voici une description détaillée de chaque type avec des exemples :

- **Elliptique** : Les équations elliptiques correspondent à des solutions stationnaires, où les conditions ne changent pas avec le temps. Un exemple typique est l'équation de Laplace, utilisée pour décrire la conduction thermique stationnaire, les potentiels électriques et d'autres phénomènes similaires.

$$\Delta T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = f(x,y)$$

  Où $f(x,y)$ est un terme de source :

- **Parabolique** : Les équations paraboliques décrivent des phénomènes qui évoluent avec le temps. Un exemple classique est l'équation de la chaleur, utilisée pour modéliser la diffusion de la chaleur dans un matériau au fil du temps.

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

  où $\alpha$ est le coefficient de diffusion thermique telle que ($\alpha = \frac{k}{\rho C_p}$).

- **Hyperbolique** : Les équations hyperboliques modélisent la propagation des ondes et des vibrations. Un exemple courant est l'équation des ondes, qui décrit la propagation des ondes sonores, électromagnétiques et autres types d'ondes.

$$\frac{\partial^2 p_a}{\partial t^2} = c^2 \frac{\partial^2 p_a}{\partial x^2}$$

où $c$ est la vitesse de propagation de l'onde.

# 2 Application de la Méthode aux Équations Elliptiques

## 2.1 Équation Elliptique 2D : Conduction Stationnaire

### 2.1.1 Description Physique et Formulation Mathématique

Considérons l'équation de conduction stationnaire en deux dimensions :

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = f(x, y)$$

Cette équation décrit la distribution de la température $T$ dans une plaque en fonction de la position.

### 2.1.2 Discrétisation par Différences Finies

En utilisant les différences finies centrées, on peut approximiser les dérivées comme suit :

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2}$$

$$\frac{\partial^2 T}{\partial y^2} \approx \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2}$$

Ainsi, l'équation discrétisée devient :

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} = f(x_i, y_j)$$

## 2.2 Conditions aux Limites

- **Dirichlet** : Valeurs fixes aux frontières. Par exemple, $T = 0$ sur les bords d'une plaque.
- **Neumann** : Dérivées normales aux frontières. Par exemple, $\frac{\partial T}{\partial n} = 0$ sur les bords d'une plaque isolée.
- **Mixtes** : Combinaison des deux. Par exemple, une plaque avec une partie isolée et une partie à température fixe.

# 3 Méthodes de Résolution

## 3.1 Solution Directe : Méthode de Gauss

L'élimination de Gauss permet de résoudre directement les systèmes linéaires en transformant la matrice en une forme triangulaire. Voici un exemple détaillé :

Considérons le système d'équations suivant :

$$\begin{cases} 2x + 3y + z = 1 \\ 4x + y - 2z = -2 \\ -2x + y + 2z = 7 \end{cases}$$

Écrivons la matrice augmentée :

$$\left[\begin{array}{ccc|c} 2 & 3 & 1 & 1 \\ 4 & 1 & -2 & -2 \\ -2 & 1 & 2 & 7 \end{array}\right]$$

Utilisons l'élimination de Gauss étape par étape : 1. Diviser la première ligne par 2 :

$$\left[\begin{array}{ccc|c} 1 & 1.5 & 0.5 & 0.5 \\ 4 & 1 & -2 & -2 \\ -2 & 1 & 2 & 7 \end{array}\right]$$

2. Soustraire 4 fois la première ligne de la deuxième ligne :

$$\left[\begin{array}{ccc|c} 1 & 1.5 & 0.5 & 0.5 \\ 0 & -5 & -4 & -4 \\ -2 & 1 & 2 & 7 \end{array}\right]$$

3. Ajouter 2 fois la première ligne à la troisième ligne :

$$\left[\begin{array}{ccc|c} 1 & 1.5 & 0.5 & 0.5 \\ 0 & -5 & -4 & -4 \\ 0 & 4 & 3 & 8 \end{array}\right]$$

4. Diviser la deuxième ligne par -5 :

$$\left[\begin{array}{ccc|c} 1 & 1.5 & 0.5 & 0.5 \\ 0 & 1 & 0.8 & 0.8 \\ 0 & 4 & 3 & 8 \end{array}\right]$$

5. Soustraire 4 fois la deuxième ligne de la troisième ligne :

$$\left[\begin{array}{ccc|c} 1 & 1.5 & 0.5 & 0.5 \\ 0 & 1 & 0.8 & 0.8 \\ 0 & 0 & -0.2 & 4.8 \end{array}\right]$$

6. Diviser la troisième ligne par -0.2 :

$$\left[\begin{array}{ccc|c} 1 & 1.5 & 0.5 & 0.5 \\ 0 & 1 & 0.8 & 0.8 \\ 0 & 0 & 1 & -24 \end{array}\right]$$

Substitution arrière :

$$z = -24$$

$$y + 0.8z = 0.8 \Rightarrow y + 0.8(-24) = 0.8 \Rightarrow y = 20$$

$$x + 1.5y + 0.5z = 0.5 \Rightarrow x + 1.5(20) + 0.5(-24) = 0.5 \Rightarrow x = -17.5$$

Solutions :

$$\begin{cases} x = -17.5 \\ y = 20 \\ z = -24 \end{cases}$$

## Code MATLAB

```
A = [2 3 1; 4 1 -2; -2 1 2];
b = [1; -2; 7];
Ab = [A b];
for k = 1:length(b)-1
    for i = k+1:length(b)
        factor = Ab(i,k) / Ab(k,k);
        Ab(i,k:end) = Ab(i,k:end) - factor * Ab(k,k:end);
    end
end
x = zeros(length(b),1);
x(end) = Ab(end,end) / Ab(end,end-1);
for i = length(b)-1:-1:1
    x(i) = (Ab(i,end) - Ab(i,i+1:end-1) * x(i+1:end)) / Ab(i,i);
end
```

## 3.2 Solutions Itératives

### 3.2.1 Méthode de Jacobi

La méthode de Jacobi est une technique itérative pour résoudre les systèmes d'équations linéaires en utilisant des approximations successives. Elle met à jour chaque point indépendamment des autres dans chaque itération.

*__Forme Itérative__

$$x_i^{(k+1)} = \frac{b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}}{a_{ii}}$$

*__Exemple__

Considérons le système d'équations suivant :

$$\begin{cases} 2x + 3y + z = 1 \\ 4x + y - 2z = -2 \\ -2x + y + 2z = 7 \end{cases}$$

Nous choisissons une estimation initiale :

$$x^{(0)} = 0, \quad y^{(0)} = 0, \quad z^{(0)} = 0$$

**__Itération 1__** :

$$\begin{cases} x^{(1)} = \frac{1-3(0)-0}{2} = \frac{1}{2} = 0.5 \\ y^{(1)} = \frac{-2-4(0)+2(0)}{1} = -2 \\ z^{(1)} = \frac{7+2(0)-0}{2} = \frac{7}{2} = 3.5 \end{cases}$$

**__Itération 2__**:

$$\begin{cases} x^{(2)} = \frac{1-3(-2)-3.5}{2} = 1.75 \\ y^{(2)} = \frac{-2-4(0.5)+2(3.5)}{1} = 3 \\ z^{(2)} = \frac{7+2(0.5)-(-2)}{2} = 5 \end{cases}$$

*Code MATLAB

```
A = [2 3 1; 4 1 -2; -2 1 2];
b = [1; -2; 7];
num_iter = 100;
tolerance = 1e-6;
x = zeros(size(b));
for k = 1:num_iter
    x_new = x;
    for i = 1:length(b)
```

```
        sum = 0;
        for j = 1:length(b)
            if j ~= i
                sum = sum + A(i,j) * x(j);
            end
        end
        x_new(i) = (b(i) - sum) / A(i,i);
    end
    if norm(x_new - x, inf) < tolerance
        break;
    end
    x = x_new;
end
```

### 3.2.2 Méthode de Gauss-Seidel

La méthode de Gauss-Seidel est une amélioration de la méthode de Jacobi, utilisant les valeurs les plus récentes disponibles pour les variables à chaque étape d'itération.

*__Forme Itérative__

$$x_i^{(k+1)} = \frac{b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)}}{a_{ii}}$$

*__Exemple__

Considérons le même système d'équations :

$$\begin{cases} 2x + 3y + z = 1 \\ 4x + y - 2z = -2 \\ -2x + y + 2z = 7 \end{cases}$$

Nous choisissons une estimation initiale :

$$x^{(0)} = 0, \quad y^{(0)} = 0, \quad z^{(0)} = 0$$

****Itération 1****:

$$\begin{cases} x^{(1)} = \frac{1-3(0)-0}{2} = \frac{1}{2} = 0.5 \\ y^{(1)} = \frac{-2-4(0.5)+2(0)}{1} = -4 \\ z^{(1)} = \frac{7+2(0.5)-(-4)}{2} = \frac{7+1+4}{2} = 6 \end{cases}$$

****Itération 2****:

$$\begin{cases} x^{(2)} = \frac{1-3(-4)-6}{2} = \frac{1+12-6}{2} = 3.5 \\ y^{(2)} = \frac{-2-4(3.5)+2(6)}{1} = -7 \\ z^{(2)} = \frac{7+2(3.5)-(-7)}{2} = \frac{7+7+7}{2} = 10.5 \end{cases}$$

*Code MATLAB

```
A = [2 3 1; 4 1 -2; -2 1 2];
b = [1; -2; 7];
num_iter = 100;
tolerance = 1e-6;
x = zeros(size(b));
for k = 1:num_iter
    x_old = x;
    for i = 1:length(b)
        sum = 0;
        for j = 1:length(b)
            if j ~= i
                sum = sum + A(i,j) * x(j);
```

```
            end
        end
        x(i) = (b(i) - sum) / A(i,i);
    end
    if norm(x - x_old, inf) < tolerance
        break;
    end
end
```

### 3.2.3 Méthode de Sur-Relaxation Successive (SOR)

La méthode de Sur-Relaxation Successive (SOR) est une amélioration de la méthode de Gauss-Seidel en introduisant un paramètre de relaxation $\omega$ pour accélérer la convergence.

*Forme Itérative

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

*Exemple

Considérons encore le même système d'équations :

$$\begin{cases} 2x + 3y + z = 1 \\ 4x + y - 2z = -2 \\ -2x + y + 2z = 7 \end{cases}$$

Nous choisissons une estimation initiale et un paramètre de relaxation $\omega$ :

$$x^{(0)} = 0, \quad y^{(0)} = 0, \quad z^{(0)} = 0, \quad \omega = 1.25$$

**Itération 1**:

$$\begin{cases} x^{(1)} = (1-1.25)\cdot 0 + \frac{1.25}{2}(1 - 3\cdot 0 - 0) = 0.625 \\ y^{(1)} = (1-1.25)\cdot 0 + 1.25(-2 - 4\cdot 0.625 + 2\cdot 0) = -5 \\ z^{(1)} = (1-1.25)\cdot 0 + \frac{1.25}{2}(7 + 2\cdot 0.625 - (-5)) = 6.875 \end{cases}$$

**Itération 2**:

$$\begin{cases} x^{(2)} = (1-1.25)\cdot 0.625 + \frac{1.25}{2}(1 - 3\cdot(-5) - 6.875) = 6.0625 \\ y^{(2)} = (1-1.25)\cdot -5 + 1.25(-2 - 4\cdot 6.0625 + 2\cdot 6.875) = -13.7813 \\ z^{(2)} = (1-1.25)\cdot 6.875 + \frac{1.25}{2}(7 + 2\cdot 6.0625 - (-13.7813)) = 22.3281 \end{cases}$$

*Code MATLAB

```
A = [2 3 1; 4 1 -2; -2 1 2];
b = [1; -2; 7];
omega = 1.25;
num_iter = 100;
tolerance = 1e-6;
x = zeros(size(b));
for k = 1:num_iter
    x_old = x;
    for i = 1:length(b)
        sum = 0;
        for j = 1:length(b)
            if j ~= i
                sum = sum + A(i,j) * x(j);
            end
        end
        x(i) = (1 - omega) * x(i) + omega * (b(i) - sum) / A(i,i);
    end
```

```
    if norm(x - x_old, inf) < tolerance
        break;
    end
end
```

#### 3.2.4 Méthodes à Pas Multiples et Stationnarisation de Douglas-Rachford

### 3.3 Introduction à la Stationnarisation

Les méthodes à pas multiples visent à améliorer la convergence en utilisant plusieurs étapes pour chaque mise à jour. La stationnarisation de Douglas-Rachford est une technique avancée utilisée pour résoudre les problèmes d'optimisation. Elle combine les avantages des méthodes itératives avec la projection sur des ensembles convexes.

### 3.4 Techniques pour Accélérer la Convergence

Des techniques comme le préconditionnement, la relaxation et l'utilisation de méthodes à pas multiples peuvent être utilisées pour accélérer la convergence des méthodes itératives. Ces techniques sont particulièrement utiles pour les problèmes de grande taille et les systèmes mal conditionnés.

### 3.5 Comparaison des Performances

Comparer les performances des différentes méthodes sur divers types de problèmes permet de déterminer la méthode la plus efficace pour une application donnée. Les critères de comparaison incluent le nombre d'itérations, le temps de calcul et la précision des solutions.

### 3.6 Optimisation de la Convergence

L'optimisation de la convergence peut être réalisée en ajustant les paramètres de relaxation, en utilisant des maillages adaptés et en appliquant des techniques avancées comme la stationnarisation de Douglas-Rachford. L'objectif est de réduire le nombre d'itérations nécessaires pour atteindre une solution acceptable.

## 4 Analyse de Convergence

### 4.1 Influence de la Discrétisation (Taille de Maillage)

La taille du maillage affecte la précision et la convergence des solutions obtenues par les méthodes de différences finies. Une discrétisation fine améliore la précision mais augmente le coût de calcul. Une discrétisation grossière réduit le coût de calcul mais peut introduire des erreurs significatives.

### 4.2 Application aux Systèmes Non Linéaires

Les méthodes de différences finies peuvent être étendues aux systèmes non linéaires en utilisant des linéarisations successives. Cela implique de résoudre une série de problèmes linéaires approchés jusqu'à convergence vers la solution du problème non linéaire.

## 5 Exemples Pratiques avec MATLAB

### 5.1 Simulation de la Conduction Thermique en 2D

Une simulation de conduction thermique en 2D peut être réalisée en utilisant des méthodes itératives pour résoudre l'équation de la chaleur discrétisée. Par exemple :

```
% Paramètres
Lx = 1; % Longueur en x
Ly = 1; % Longueur en y
nx = 20; % Nombre de points en x
ny = 20; % Nombre de points en y
```

```
dx = Lx / (nx - 1);
dy = Ly / (ny - 1);

% Matrice A et vecteur b
A = sparse(nx*ny, nx*ny);
b = zeros(nx*ny, 1);

% Remplissage de A et b
for i = 1:nx
    for j = 1:ny
        % Calcul de l'indice linéaire
        k = i + (j-1) * nx;

        if i == 1 || i == nx || j == 1 || j == ny
            % Conditions aux limites de Dirichlet
            A(k, k) = 1;
            b(k) = 0;
        else
            % Points intérieurs
            A(k, k) = -4;
            A(k, k-1) = 1;
            A(k, k+1) = 1;
            A(k, k-nx) = 1;
            A(k, k+nx) = 1;
            b(k) = f(i*dx, j*dy);
        end
    end
end

% Résolution du système
u = A\b;

% Reshape du vecteur solution en matrice 2D
U = reshape(u, [nx, ny]);

% Affichage du résultat
surf(linspace(0, Lx, nx), linspace(0, Ly, ny), U);
xlabel('x');
ylabel('y');
zlabel('u(x,y)');
title('Conduction thermique en 2D');
```

## 5.2 Comparaison des Différentes Méthodes de Résolution

Pour comparer les différentes méthodes de résolution, nous pouvons mesurer le nombre d'itérations nécessaires pour atteindre une certaine précision, ainsi que le temps de calcul pour chaque méthode.

```
% Comparaison des méthodes de Jacobi, Gauss-Seidel et SOR

methods = {'Jacobi', 'Gauss-Seidel', 'SOR'};
num_iter = 100;
tolerance = 1e-6;

for method = methods
    % Initialisation
    x = zeros(size(b));
    iter_count = 0;
    tic;
```

```
    for k = 1:num_iter
        x_old = x;

        switch method{1}
            case 'Jacobi'
                % Implémentation de Jacobi
                x_new = x;
                for i = 1:length(b)
                    sum = 0;
                    for j = 1:length(b)
                        if j ~= i
                            sum = sum + A(i,j) * x(j);
                        end
                    end
                    x_new(i) = (b(i) - sum) / A(i,i);
                end
                x = x_new;

            case 'Gauss-Seidel'
                % Implémentation de Gauss-Seidel
                for i = 1:length(b)
                    sum = 0;
                    for j = 1:length(b)
                        if j ~= i
                            sum = sum + A(i,j) * x(j);
                        end
                    end
                    x(i) = (b(i) - sum) / A(i,i);
                end

            case 'SOR'
                % Implémentation de SOR
                omega = 1.25;
                for i = 1:length(b)
                    sum = 0;
                    for j = 1:length(b)
                        if j ~= i
                            sum = sum + A(i,j) * x(j);
                        end
                    end
                    x(i) = (1 - omega) * x(i) + omega * (b(i) - sum) / A(i,i);
                end
        end

        iter_count = iter_count + 1;

        if norm(x - x_old, inf) < tolerance
            break;
        end
    end

    elapsed_time = toc;

    fprintf('Méthode: %s, Itérations: %d, Temps: %f\n', method{1}, iter_count, elapsed_time);
end
```

### 5.3 Analyse de Convergence sur Différents Maillages

Pour analyser la convergence sur différents maillages, nous pouvons varier la taille du maillage et observer l'évolution de l'erreur de la solution.

```
% Analyse de convergence sur différents maillages

Lx = 1;
Ly = 1;

maillages = [10, 20, 40, 80];
erreurs = zeros(size(maillages));

for m = 1:length(maillages)
    nx = maillages(m);
    ny = maillages(m);
    dx = Lx / (nx - 1);
    dy = Ly / (ny - 1);

    % Matrice A et vecteur b
    A = sparse(nx*ny, nx*ny);
    b = zeros(nx*ny, 1);

    % Remplissage de A et b
    for i = 1:nx
        for j = 1:ny
            k = i + (j-1) * nx;

            if i == 1 || i == nx || j == 1 || j == ny
                A(k, k) = 1;
                b(k) = 0;
            else
                A(k, k) = -4;
                A(k, k-1) = 1;
                A(k, k+1) = 1;
                A(k, k-nx) = 1;
                A(k, k+nx) = 1;
                b(k) = f(i*dx, j*dy);
            end
        end
    end

    % Solution du système
    u = A\b;

    % Calcul de l'erreur (par exemple, norme L2 de la différence)
    u_exact = exact_solution(linspace(0, Lx, nx), linspace(0, Ly, ny));
    erreurs(m) = norm(u - u_exact(:), 2);
end

% Affichage des résultats
loglog(maillages, erreurs);
xlabel('Taille du maillage');
ylabel('Erreur');
title('Analyse de convergence sur différents maillages');
```