TP. Résolution des systèmes linéaires. 3ème partie

1. La méthode de Gauss avec pivotement partiel et mise à l'échelle des lignes

Voici un exemple de code MATLAB qui utilise **la méthode de Gauss avec pivotement partiel et mise à l'échelle des lignes** pour résoudre un système d'équations linéaires :

```
function x = gaussPivotScale(A, b)
n = length(b);
x = zeros(n, 1); % Initialisation du vecteur solution
s = zeros(n, 1); % Vecteur pour stocker la plus grande valeur de chaque ligne
index = 1:n; % Index pour suivre les changements de lignes
% Mise à l'échelle: Trouver la plus grande valeur de chaque ligne
for i = 1:n
s(i) = max(abs(A(i,:)));
end
% Élimination de Gauss avec pivotement partiel et mise à l'échelle
for k = 1:n-1
% Pivotement partiel avec mise à l'échelle
[pivot, i_max] = max(abs(A(index(k:n),k)) ./ s(index(k:n)));
i_max = i_max + k - 1;
if pivot == 0, error('La matrice est singulière!'); end
% Échanger les lignes si nécessaire
if i_max ~= k
index([k, i_max]) = index([i_max, k]);
end
% Élimination
for i = k+1:n
factor = A(index(i), k) / A(index(k), k);
A(index(i), k+1:n) = A(index(i), k+1:n) - factor * A(index(k), k+1:n);
b(index(i)) = b(index(i)) - factor * b(index(k));
end
end
% Substitution arrière
for i = n:-1:1
x(i) = (b(index(i)) - A(index(i), i+1:n) * x(i+1:n)) / A(index(i), i);
end
end
```

Pour utiliser cette fonction, vous devez fournir la matrice A et le vecteur b qui représentent votre système d'équations linéaires. La fonction retournera le vecteur solution x.

Exemple 1:

```
A = [3 -0.1 -0.2; 0.1 7 -0.3; 0.3 -0.2 10];
b = [7.85; -19.3; 71.4];
x = gaussPivotScale(A, b);
disp(x);
```

2. La méthode de Gauss avec pivot complet

La méthode d'élimination de Gauss avec pivotement complet est une variante de l'élimination de Gauss avec pivotement partiel. Elle comprend le pivotement complet, qui considère à la fois les lignes et les colonnes pour le pivot, au lieu de se limiter aux lignes comme dans le pivotement partiel. Cela garantit la sélection de l'élément le plus grand en valeur absolue dans toute la sous-matrice restante comme pivot à chaque étape, minimisant encore davantage les erreurs d'arrondi.

Voici les étapes de l'élimination de Gauss avec pivotement complet :

1. **Pivotement Complet :**
   - À chaque étape k, parmi les éléments restants dans la sous-matrice $A[k : n, k : n]$, identifiez l'élément avec la valeur absolue la plus élevée.
   - Échangez les lignes et les colonnes pour amener cet élément à la position (k, k) de la matrice A. Cela nécessite de garder une trace des échanges de colonnes car cela change l'ordre des variables dans les solutions.
2. **Élimination :**
   - Comme dans l'élimination de Gauss standard, utilisez l'élément pivot pour réduire les éléments en dessous de lui à zéro. Cela se fait en soustrayant les multiples appropriés de la ligne du pivot des lignes en dessous.
   - Continuez ce processus jusqu'à ce que la matrice de coefficients soit transformée en une matrice triangulaire supérieure.
3. **Substitution Arrière :**
   - Après avoir obtenu une matrice triangulaire supérieure, résolvez le système d'équations par substitution arrière pour obtenir les valeurs des inconnues.
   - N'oubliez pas de réordonner les variables à la fin si des échanges de colonnes ont eu lieu.

Voici comment vous pourriez implémenter cela en MATLAB :

```
function x = gaussPivotComplet(A, b)
% Cette fonction résout Ax=b en utilisant la méthode de Gauss avec
pivot complet.
% A : Matrice des coefficients
% b : Vecteur colonne des termes constants
% x : Solution du système
[m,n] = size(A);
if m ~= n
error('La matrice A doit être carrée');
end
% Extension de A avec b pour simplifier les opérations
Ab = [A b];
```

```
for k = 1:n-1
% Recherche du pivot complet
[maxval, maxpos] = max(abs(Ab(k:end, k:end-1)), [], 'all', 'linear');
[p, q] = ind2sub([n-k+1, n-k], maxpos);
p = p + k - 1;
q = q + k - 1;
% Permutation des lignes et des colonnes pour placer le pivot
Ab([k p], :) = Ab([p k], :); % Échange des lignes
Ab(:, [k q]) = Ab(:, [q k]); % Échange des colonnes
% Élimination de Gauss
for i = k+1:n
factor = Ab(i,k) / Ab(k,k);
Ab(i, k+1:end) = Ab(i, k+1:end) - factor * Ab(k, k+1:end);
Ab(i, k) = 0; % Pour améliorer la lisibilité
end
end
% Substitution arrière
x = zeros(n, 1);
x(n) = Ab(n,end) / Ab(n,n);
for i = n-1:-1:1
x(i) = (Ab(i,end) - Ab(i,i+1:n)*x(i+1:n)) / Ab(i,i);
end
end
```

Remarques importantes :

- Ce code nécessite que la matrice A soit carrée et que le système $Ax = b$ ait une solution unique.
- La fonction effectue des permutations des lignes et des colonnes de la matrice étendue $[A \mid b]$ pour placer le pivot. Cela signifie que l'ordre des variables dans le vecteur solution x correspond à l'ordre des colonnes modifié de la matrice A après les étapes de pivot.
- La méthode du pivot complet est plus stable numériquement que le pivot partiel, mais elle est aussi plus coûteuse en termes de calculs car elle recherche le maximum dans toute la sous-matrice restante à chaque étape.
- Ce code ne gère pas les cas où la matrice A est singulière ou proche de l'être. Dans de tels cas, une vérification de la condition du pivot (par exemple, si **'Ab(k,k)'** est très proche de zéro) peut être ajoutée pour gérer ces situations.

Pour utiliser cette fonction, définissez votre matrice A et votre vecteur b, puis appelez la fonction **'gaussPivotComplet(A, b)'**.

Pour illustrer cela avec deux exemples, prenons le système suivant :

Exemple 1:

```
% Définition de la matrice A et du vecteur b
A = [3 2 1; 1 3 2; 1 1 3];
b = [10; 13; 14];
% Résolution du système Ax = b
x = gaussPivotComplet(A, b);
% Affichage de la solution
disp('La solution du système est :');
```

```
disp(x);
```

Exemple 2:

```
% Définition de la matrice A et du vecteur b
A = [2 3 -1 4; 4 4 2 -2; -2 3 -1 3; 5 -3 2 2];
b = [20; 28; 8; 10];
% Résolution du système Ax = b
x = gaussPivotComplet(A, b);
% Affichage de la solution
disp('La solution du système est :');
disp(x);
```