8) Autres méthodes numériques avec Matlab

8.1) Système d'équations linéaires

1) Forme

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \quad \vdots \quad \ddots \quad \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \end{cases}$$

2) Forme matricielle

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \qquad X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Connue *Inconnue* *Connue*

(Matrice des coefficients)

- S'écrit: A × X = B
- Si $m = n$:
  - ✓ (système déterminé)
  - ✓ une solution unique (si $det(A) \neq 0$)
  - ✓ solution peut être obtenue par différentes méthodes

3) Méthode Matricielle (matrice inverse)

- Si A^{-1} *est la matrice inverse de A*
  - ✓ *le système* $A \times X = B$ s'écrit aussi : $X = A^{-1} \times B$
- La connaissance de $A{-}1$ permet de calculer directement la solution X par produit matriciel
- Exemple:

$$A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & -1 & 0 \\ 5 & 4 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 6 \\ 0 \\ -1 \end{pmatrix}$$

Méthode Matricielle (matrice inverse)-Exemple

- Solution par Matlab

```
>> A=[1 -2 3; 1 -1 0;  3 4 2];
B=[3 0 -1]';
x=inv(A)*B
%ou bien x =A^(-1)*B
%ou bien x = A\B
x =
   -0.3913
   -0.3913
    0.8696
```

- Alors la solution est la matrice X

$$X = \begin{pmatrix} -0.3913 \\ -0.3913 \\ 0.8693 \end{pmatrix}$$

4) Méthode de Gauss-Seidel

- Est une méthode itérative pour le calcul de la solution d'un système linéaire
- Construit une suite de vecteurs : $x^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$

  convergent vers le vecteur solution exacte $x = (x_1, x_2, \ldots, x_n)$ *pour tout vecteur initiale* $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$ lorsque k tend vers ∞.

- Soit un système à 3 inconnus:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases}$$

  ✓ peut s'écrit comme suit:

$$\begin{cases} x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\ x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\ x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \end{cases}$$

Méthode de Gauss-Seidel -Principe

- Vecteur initial $x_0 = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})$
- A la première itération:
  - ✓ à partir du vecteur x_0, on calcule:

$$\begin{cases} x_1^{(1)} = (b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)})/a_{11} \\ x_2^{(1)} = (b_2 - a_{21}x_1^{(1)} - a_{23}x_3^{(0)})/a_{22} \\ x_3^{(1)} = (b_3 - a_{31}x_1^{(1)} - a_{32}x_2^{(1)})/a_{33} \end{cases}$$

- On continue jusqu'à aboutir à une précision suffisante.

- Exemple
  - ✓ Considérons le système linéaire:

$$\left(\begin{array}{lcl} 4x_1 + 2x_2 + x_3 & = & 4 \\ -x_1 + 2x_2 & = & 2 \\ 2x_1 + x_2 + 4x_3 & = & 9 \end{array}\right)$$

  - ✓ On peut le mettre sous la forme:

$$\left(\begin{array}{lcl} x_1 & = & (4 - 2x_2 - x_3)/4 \\ x_2 & = & (2 + x_1)/2 \\ x_3 & = & (9 - 2x_1 - x_2)/4 \end{array}\right)$$

  - ✓ Soit le vecteur initial:

$$x^{(0)} = (0, 0, 0)$$

Méthode de Gauss-Seidel -Exemple

- **Itération 1:** En partant de $x^{(0)} = (0, 0, 0)$

$$\left(\begin{array}{lcll} x_1 & = & (4 - 2(0) - (0))/4 & = & 1 \\ x_2 & = & (2 + 1)/2 & = & 3/2 \\ x_3 & = & (9 - 2(1) - 3/2)/4 & = & 11/8 \end{array}\right) \quad x^{(1)} = (1, \tfrac{3}{2}, \tfrac{11}{8})$$

- **Itération 2:** En partant de $x^{(1)} = (1, \frac{3}{2}, \frac{11}{8})$

$$\left(\begin{array}{lcll} x_1 & = & (4 - 2(\frac{3}{2}) - (\frac{11}{8}))/4 & = & \frac{-3}{32} \\ x_2 & = & (2 - \frac{-3}{32})/2 & = & \frac{61}{64} \\ x_3 & = & (9 - 2(\frac{-3}{32}) - (\frac{61}{64}))/4 & = & \frac{527}{256} \end{array}\right) x^{(2)} = (\tfrac{-3}{32}, \tfrac{61}{64}, \tfrac{527}{256})$$

- **Itération 3:** En partant de $x^{(2)} = (\frac{-3}{32}, \frac{61}{64}, \frac{527}{256})$

$$\left(\begin{array}{lcll} x_1 & = & (4 - 2(\frac{61}{64}) - (\frac{527}{256}))/4 & = & \frac{9}{1024} \\ x_2 & = & (2 + \frac{9}{1024})/2 & = & \frac{2057}{2048} \\ x_3 & = & (9 - 2(\frac{9}{1024}) - (\frac{2057}{2048}))/4 & = & \frac{16339}{8192} \end{array}\right)$$

$$x^{(3)} = (\tfrac{9}{1024}, \tfrac{2057}{2048}, \tfrac{16339}{8192}) \cong (0.0087, 1.0043, 1.9945)$$

- $x^{(3)}$ converge vers la solution du système $x = (0, 1, 2)$

- Programme Metlab.

```
function x=Gauss_Seidel(A,b,n)
%n représente le nombre d'itérations
x=[0,0,0];
for i=1:n
x(1)=(b(1)-x(3)*A(1,3)-x(2)*A(1,2))/A(1,1);
x(2)=(b(2)-x(3)*A(2,3)-x(1)*A(2,1))/A(2,2);
x(3)=(b(3)-x(2)*A(3,2)-x(1)*A(3,1))/A(3,3);
end
return
```

```
A = [4 2 1; -1 2 0; 2 1 4]
b = [4 2 9] %ou b = [4 2 9]'
x=Gauss_seidel(A,b,10)
```

Méthode de Gauss-Seidel

- Test d'arrêt
  - ✓ Le calcul s'arrête lorsque $(Ax - b) < Tol$
  - ✓ *Tol:* une tolérance précisée à l'avance.

Méthode de Gauss-Seidel- Test d'arrêt

- Nouveau programme Metlab.

```
function [x,n]=Gauss_Seidel2(A,b)
   %n représente le nombre d'itérations
   %b est un vecteur colonne
   n = 0;  x=[0,0,0];
   C=(A*x')-b;
   tol=1e-5;
while abs(C(1))>tol | abs(C(2))>tol |
abs(C(3))>tol
x(1)=(b(1)-x(3)*A(1,3)-x(2)*A(1,2))/A(1,1);
x(2)=(b(2)-x(3)*A(2,3)-x(1)*A(2,1))/A(2,2);
x(3)=(b(3)-x(2)*A(3,2)-x(1)*A(3,1))/A(3,3);
n=n+1;
C=(A*x')-b ;
end
return
```

Méthode de Gauss-Seidel- Test d'arrêt

- Exemple

$$3x_1 + 2x_2 + x_3 = 10$$
$$2x_1 - 3x_2 - 2x_3 = 10$$
$$x_1 - x_2 + 2x_3 = 5$$

```
A = [3 2 1; 2 -3 -2; 1 -1 2]
b = [10 10 5]'
[X,n] = Gauss_seidel2(A,b)
```

8.2) Interpolation de données

- Consiste à trouver l'expression générale d'une fonction à partir d'un nombre limité de points (points d'appuis)

8.2.1) Interpolation polynomiale

- Si la fonction recherchée est un polynôme l'interpolation est dite polynomial.

1) Interpolation polynomiale de **Lagrange**

- Si *f* une fonction donnée définie sur IR
- Interpoler *f* par un polynôme *P* de degré *n* sachant les n+1 points : $(x_0, f(x_0)), (x_1, f(x_1)), \ldots , (x_n, f(x_n))$
  - ✓ consiste à résoudre le problème suivant :Trouver un polynôme P de degré ≤ n tel que : $y_i = P(x_i) = f(x_i),\ 0 \leq i \leq n$
  - ✓ Ce polynôme est donnée par :

$$P(x) = \sum_{j=1}^{n} P_j(x)$$

  - ✓ P_j est un polynôme qui passe par le point $(x_j, f(x_j))$ et qui s'annule dans tous les autres points
  - ✓ P_j est calculé par :

$$P_j(x) = y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}$$

Interpolation de Lagrange

- Construction du polynôme:
- $P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$
- Afin que P(x) passe par l'ensemble des points à interpoler, il faut que les coefficients $a_0 \ldots a_n$ respectent le système d'équation linéaire suivant : $AP = Y$

$$\begin{pmatrix} x_0^n & x_0^{n-1} \ldots & x_0 & 1 \\ x_1^n & x_2^{n-1} \ldots & x_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n^n & x_n^{n-1} \ldots & x_n & 1 \end{pmatrix} \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

✓ Donc pour trouver la formule du polynôme *P* il suffit de résoudre ce système d'équation.

Interpolation de Lagrange-Construction du polynôme

- Exemple: Polynôme de degré 3
  - ✓ $P(x) = a_3x^3 + a_2x^2 + a_1x + a_0$
  - ✓ Les points d'appuis: (1,1), (2,-1), (3,2), (4,0)
  - ✓ X=[1 2 3 4], Y = [1 -1 2 0]'
  - ✓ Système d'équation:

$$\begin{pmatrix} 1^3 & 1^2 & 1 & 1 \\ 2^3 & 2^2 & 2 & 1 \\ 3^3 & 3^2 & 3 & 1 \\ 4^3 & 4^2 & 4 & 1 \end{pmatrix} \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 2 \\ 0 \end{pmatrix}$$

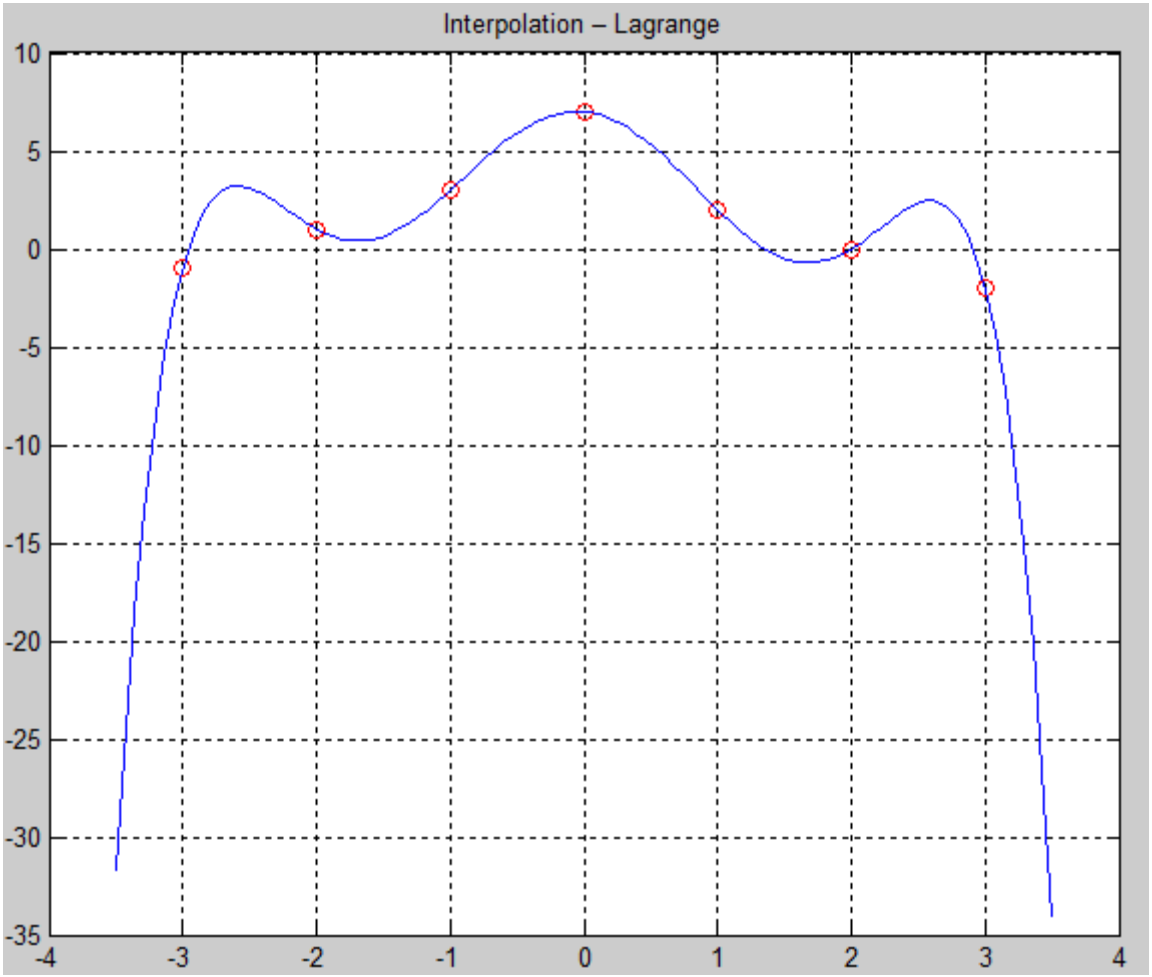
- Programme Metlab.

```
function P = lagrange(X, Y)
      n = length(X)-1;
      A = ones(n+1,n+1);
      for i=1:n+1
            for j=1:n
                  A(i,j) = X(i)^(n+1-j);
            end
      end
      P = A\Y';
return
```

- Appel de la fonction *lagrange*.

```
X1 = [-3,-2,-1,0,1,2, 3];
Y1 = [-1, 1, 3,7,2,0,-2];
P = lagrange(X1,Y1);
%Tracer les points
plot(X1,Y1, 'ro');
%Tracer la courbe de P
X = linspace(-3.5, 3.5, 100)
Y = polyval(P, X);
hold on;
plot(X,Y);
title('Interpolation - Lagrange');
grid on;
```

Interpolation de Lagrange- Programme Metlab



- On peut faire le même travail en utilisant la fonction Matlab *polyfit*

```
figure;
P = polyfit(X1,Y1,6);
X = linspace(-3.5, 3.5, 100)
Y = polyval(P, X);
plot(X,Y);
```

Merci