

# Algorithmique Numérique

saida.bouakaz@univ-lyon1.fr

- **Rappel sur les matrice**
  - Définitions
  - Opérations sur les matrices
  - Déterminant & méthode de cramer
- **Résolution de système linéaire**
  - Méthodes directes
    - Triangulation de Gauss
    - Décomposition LU
  - Méthodes itératives
    - Méthode de Jacobi
    - Méthode de Seidel
- **Racines de fonctions  $F(x)=0$** 
  - Introduction
  - Méthode de Newton
  - Méthode de la sécante
  - Méthode de dichotomie

- **Interpolation**

- Interpolation linéaire et quadratique
- Formule de Lagrange, polynôme de Newton,
- Différences finis
- Splines

- **Approximation polynomiale**

- Méthode des moindres carrés, moindres carrés pondérées
- Polynômes de Chebychev

- **Intégration numérique**

- Introduction
- Méthode des trapèzes
- Méthode de Simpson
- Méthodes améliorées

# Chapitre 2

## Résolution de systèmes linéaires

- Méthode de Gauss: basée sur la triangulation
- Méthode de factorisation : LU
- Méthodes itératives

# Méthode de Gauss

- Idée : méthode basée sur la triangulation
- Utilise une suite de combinaison linéaires entre les différentes lignes, travaille sur la matrice élargie.
- $AX=B \longrightarrow A^{(k)}X=B^{(k)}$  avec  $A^{(k)}$  triangulaire.
- Complexité
  - Complexité de la résolution du système triangulaire en  $O(n^2)$  :
  - Complexité de la triangulation en  $O(n^3)$  :

# Méthode de Gauss

- Procédé du pivot avec normalisation de la diagonale

Le principe consiste à transformer le système  $A X = B$  en un système triangulaire équivalent

$$T \times X = C \equiv \begin{cases} x_1 + t_{1,2}x_2 + \cdots + t_{1,n}x_n & = & c_1 \\ & x_2 + \cdots + t_{2,n}x_n & = & c_2 \\ & & \vdots \\ & & x_n & = & c_n \end{cases}$$

La solution se calcule par remontée.

- La transformation de  $A$  en  $T$  se compose de deux étapes itérées  $n$  fois.

A l'étape  $i$  :

- normalisation : on divise la ligne  $i$  par  $a_{i,i}$  (le *pivot*)  
si  $a_{i,i} \neq 0$  pour obtenir  $a_{i,i} = 1$ ,
- annulation sous la diagonale : pour  $i + 1 = k \rightarrow n$ ,  
on soustrait la ligne du pivot multipliée par  $a_{k,i}$  à la ligne  $k$  pour obtenir  $a_{k,i} = 0$

# Méthode de Gauss - form

## Procédé du pivot sans normalisation de la diagonale

*On garde le principe de transformer le système  $A X = B$  en un système équivalent.*

*On travaille tjrs avec la matrice élargie.*

On note par  $m_{i1} = \frac{a_{i1}}{a_{11}}$  pour  $1 \leq i \leq n$  d'où

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ & & (a_{22} - m_{21}a_{12})x_2 & + & \cdots & + & (a_{2n} - m_{21}a_{1n})x_n & = & b_2 - m_{21}b_1 \\ & & \vdots & & & & \vdots & & \\ & & (a_{i2} - m_{i1}a_{12})x_2 & + & \cdots & + & (a_{in} - m_{i1}a_{1n})x_n & = & b_i - m_{i1}b_1 \\ & & \vdots & & & & \vdots & & \\ & & (a_{n2} - m_{n1}a_{12})x_2 & + & \cdots & + & (a_{nn} - m_{n1}a_{1n})x_n & = & b_n - m_{n1}b_1 \end{array}$$

A l'issue de la première transformation, la matrice du nouveau système est

$$A^{(2)} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} - m_{21}a_{12} & \cdots & a_{2n} - m_{21}a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_{i2} - m_{i1}a_{12} & \cdots & a_{in} - m_{i1}a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2} - m_{n1}a_{12} & \cdots & a_{nn} - m_{n1}a_{1n} \end{pmatrix}$$

le second membre est

$$b^{(2)} = \begin{pmatrix} b_1 \\ b_2 - m_{21}b_1 \\ \vdots \\ b_i - m_{i1}b_1 \\ \vdots \\ b_n - m_{n1}b_1 \end{pmatrix}$$



Le nouveau système s'écrit :

$$A^{(2)}X = b^{(2)}$$

À l'étape  $k$ , on a

$$A^{(k)} = \begin{pmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \cdots & \cdots & a_{1n}^{(k)} \\ 0 & a_{22}^{(k)} & & & a_{2n}^{(k)} \\ \vdots & \vdots & & & \\ 0 & 0 & a_{k-1,k-1}^{(k)} & a_{k-1,k}^{(k)} & \cdots & a_{k,n}^{(k-1)} \\ \vdots & \vdots & 0 & \vdots & & \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{pmatrix}$$

$$b^{(2)} = \begin{pmatrix} b_1 \\ b_2 - m_{21}b_1 \\ \vdots \\ b_i - m_{i1}b_1 \\ \vdots \\ b_n - m_{n1}b_1 \end{pmatrix}$$

## Cas générique : à l'étape k

Étape  $(K-1)$  :  $A^{(K-1)} X = B^{(K-1)}$

$$A^{(k-1)} = \begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \dots & a_{1k-1}^{(0)} & a_{1k}^{(0)} & \dots & a_{1n}^{(0)} \\ 0 & a_{22}^{(1)} & \dots & a_{2k-1}^{(1)} & a_{2k}^{(1)} & \dots & a_{2n}^{(1)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{k-1,k-1}^{(k-2)} & a_{k-1,k}^{(k-2)} & \dots & a_{k-1,n}^{(k-2)} \\ 0 & 0 & \dots & 0 & a_{kk}^{(k-1)} & \dots & a_{kn}^{(k-1)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & a_{nk}^{(k-1)} & \dots & a_{nn}^{(k-1)} \end{bmatrix}$$

$$B^{(k-1)} = \begin{bmatrix} b_1^{(0)} \\ b_2^{(1)} \\ \vdots \\ b_{k-1}^{(k-2)} \\ b_k^{(k-1)} \\ \vdots \\ b_n^{(k-1)} \end{bmatrix}$$

## Expression générale

$$\begin{aligned} a_{ij}^{(k+1)} &= a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)} \\ b_i^{(k+1)} &= b_i^{(k)} - m_{ik} b_k^{(k)} \end{aligned} \quad \text{avec} \begin{cases} i = k + 1, \dots, n \\ j = k + 1, \dots, n \end{cases}$$

$$\text{où : } m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \quad i = k + 1, \dots, n$$

# Méthode de Gauss

Algorithme de triangulation sans normalisation de la diagonale

Ecrire l'algorithme

# Méthode de Gauss

- Algorithme de résolution (après triangulation de la matrice)

Ecrire l'algorithme

# Pivot de gauss : technique pratique pour inverser une matrice

Technique : elle s'appuie sur :  $A \cdot A^{-1} = I$

- la matrice A et la matrice identité I sont juxtaposée (on parle de matrice augmentée  $[A | I]$ )
- On applique une série de transformation aux ligne de façon à obtenir une matrice identité à la place de A, la matrice situé à droite sera la matrice inverse  $\rightarrow [A \cdot A^{-1} | A^{-1} \cdot I]$
- La méthode du pivot de gauss permet d'obtenir cette matrice

# Méthode de factorisation $LU$ (ou $LR$ )

- Méthode : basée sur une factorisation  $A$
- Le principe de cette méthode de recherche de solution consiste à décomposer

la matrice  $A$  sous forme d'un produit  $A = L . U$



$$A=L .U \rightarrow (L . U ) X= B$$

$$A=L .U \rightarrow L .( U X)= B \text{ si on pose } UX=Y$$

$$AX= B \Rightarrow \begin{cases} LY = B \\ UX = Y \end{cases}$$

# Méthode de factorisation $LU$ (ou LR)

Si on peut décomposer la matrice  $A$  en le produit de 2 matrices  $A=L.U$  (ou  $A= L.R$ )

- $L$  : Triangulaire inférieure (L pour Lower triangular matrix )
- $U$  : Triangulaire supérieure (U pour Upper triangular matrix )
- $AX = B \Leftrightarrow (L.U)X = B \Leftrightarrow L.(UX) = B$
- On pose  $UX = Y$  d'où  $LY = B$

3 étapes :

1. Trouver les matrices  $L$  et  $U$
2. Résolution du système  $LY = B$  ( $L$  triangulaire inférieure)
3. Résolution du système  $UX = Y$  ( $U$  triangulaire supérieure)

## Remarque

LR :  $L$  pour Left triangular matrix et  $R$  pour Right triangular matrix



- L est une matrice triangulaire inférieure avec diagonale unité
- U est une matrice triangulaire supérieure.
- On utilisera la méthode LU lorsque l'on veut résoudre une famille de systèmes de la forme

$$A \cdot X = B_i$$

- où seul le vecteur  $B_i$  (les données) varie, le modèle (matrice A) reste la même. le calcul de L et R est totalement indépendant de B

Comment déterminer  $L$  et  $U$  et quelle est la complexité de la décomposition (en ?? opérations).

- Deux méthodes :
  - décomposition de Gauss
  - Algorithme de Crout (identification)

# Représentation matricielle de l'élimination de Gauss

$$AX = B \Rightarrow \begin{cases} LY = B \\ UX = Y \end{cases}$$

Rappelle : à chaque étape de l'algorithme de gauss...

$$\text{pour } i = k + 1, \dots, n \quad \left\{ \begin{array}{l} a_{ij}^{(k+1)} \leftarrow a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} a_{kj}^{(k)} \quad \text{pour } j = k + 1, \dots, n \\ b_i^{(k+1)} \leftarrow b_i^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} b_k^{(k)} \end{array} \right.$$

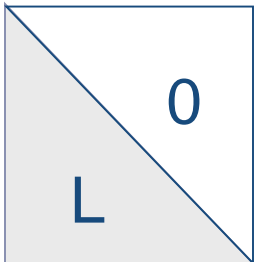
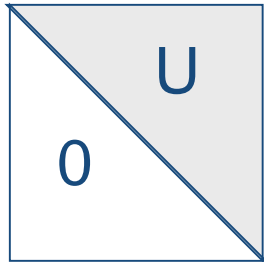
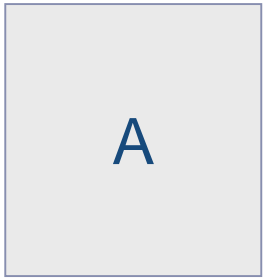
notation matricielle :  $A^{(k+1)} = M^{(k)} A^{(k)}$ ;  $b^{(k+1)} = M^{(k)} b^{(k)}$ ;

# LU : principe

Il est si facile le résoudre un système « triangulaire » !

$$A = LU$$

$$Ax = b \Leftrightarrow \begin{cases} (1) & Ly = b \\ (2) & Ux = y \end{cases}$$



Comment construire L et U ?

**idée :**

reprendre l'étape de triangularisation  
de la méthode de Gauss

# De Gauss à LU (ou LR)

Représentons une étape de la triangularisation par la multiplication de  $A$  par une matrice  $M^{(k)}$

$$A^{(k+1)} = M^{(k)} A^{(k)}$$

$$A^{(1)} = A \quad \text{et} \quad A^{(n)} = U$$

$$\ell_{i,k} = \frac{a_{ik}}{a_{kk}} = -m_{i,k}$$

$$\begin{cases} a_{ij} \leftarrow a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj} \\ b_i \leftarrow b_i - \frac{a_{ik}}{a_{kk}} b_k \end{cases}$$

$$M^{(k)} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \ddots & \boxed{-\ell_{k+1,k}} & \ddots & \vdots \\ 0 & \cdots & \boxed{-\ell_{n,k}} & 0 & 1 \end{pmatrix}$$

$$U = M^{(n-1)} \dots M^{(k)} \dots M^{(1)} A = MA$$

$$A = M^{-1}U = LU$$

$$\text{donc } L = M^{-1}$$

# LU : récapitulatif

Les matrices élémentaires  $M^{(k)}$  sont **inversibles**  
et leurs inverses sont les matrices  $L^{(k)}$  triangulaires inférieures  
telles que :

$$L^{(k)} = \begin{cases} l_{ii} = 1 & i = 1, n \\ l_{ik} = \ell_{ik} & i = k+1, n \\ l_{ij} = 0 & \text{sinon} \end{cases}$$

$$L^{(k)} = I - (M^{(k)} - I)$$

$$M^{(k)} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \ddots & -\ell_{k+1,k} & \ddots & \vdots \\ 0 & \cdots & -\ell_{n,k} & 0 & 1 \end{pmatrix}$$

$$L^{(k)} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \ddots & \ell_{k+1,k} & \ddots & \vdots \\ 0 & \cdots & \ell_{n,k} & 0 & 1 \end{pmatrix}$$

$$L = L^{(n-1)} \dots L^{(k)} \dots L^{(1)}$$

C'est la matrice  $\ell_{ik}$

# Exemple

$$A = \begin{pmatrix} 1 & 1 & -1 & 2 \\ -1 & 2 & 1 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 2 \end{pmatrix}$$

La décomposition de  $A=LU$  donne :

$$\begin{pmatrix} 1 & 1 & -1 & 2 \\ -1 & 2 & 1 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 2 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 1 & -\frac{1}{3} & 1 & 0 \\ 1 & -\frac{2}{3} & \frac{1}{2} & 1 \end{pmatrix}}_L \times \underbrace{\begin{pmatrix} 1 & 1 & -1 & 2 \\ 0 & 3 & 0 & 3 \\ 0 & 0 & 2 & -2 \\ 0 & 0 & 0 & 3 \end{pmatrix}}_U$$

Détail de la décomposition

$$\begin{pmatrix} \textcircled{1} & 1 & -1 & 2 \\ -1 & 2 & 1 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 2 \end{pmatrix} \xrightarrow{\substack{\text{Pivot}=1 \\ \text{lig2} \leftarrow \text{lig2} - (-1)\text{lig1} \\ \text{lig3} \leftarrow \text{lig3} - (1)\text{lig1} \\ \text{lig4} \leftarrow \text{lig4} - (1)\text{lig1}}} \begin{pmatrix} 1 & 1 & -1 & 2 \\ 0 & \textcircled{3} & 0 & 3 \\ 0 & -1 & 2 & -3 \\ 0 & -2 & 1 & 0 \end{pmatrix} \xrightarrow{\substack{\text{Pivot}=3 \\ \text{lig3} \leftarrow \text{lig3} - (-1/3)\text{lig2} \\ \text{lig4} \leftarrow \text{lig4} - (-2/3)\text{lig2}}}$$

Etape 2

$$\begin{pmatrix} 1 & 1 & -1 & 2 \\ 0 & 3 & 0 & 3 \\ 0 & 0 & \textcircled{2} & -2 \\ 0 & 0 & 1 & 2 \end{pmatrix} \xrightarrow{\substack{\text{Pivot}=2 \\ \text{lig4} \leftarrow \text{lig4} - (1/2)\text{lig3}}} \begin{pmatrix} 1 & 1 & -1 & 2 \\ 0 & 3 & 0 & 3 \\ 0 & 0 & 2 & -2 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

# L'algorithme de décomposition

Fonction  $L, U = \text{décompose}(A)$

pour  $k = 1$  jusqu'à  $n - 1$

$pivot \leftarrow a_{kk}$  (\* stratégie de pivot \*)

si  $pivot \neq 0$  alors

$\ell_{kk} \leftarrow 1$

pour  $i = k + 1$  jusqu'à  $n$

$\ell_{ik} \leftarrow \frac{a_{ik}}{pivot}$

pour  $j = k + 1$  jusqu'à  $n$

$a_{ij} \leftarrow a_{ij} - \ell_{ik} a_{kj}$

fait

fait

fait sinon "problème"



# Calcul des matrices L et U (ou L et R) par identification : Algorithme de Crout

Pour calculer L et U, il suffit de remarquer que

$$\begin{pmatrix} 1 & 0 & 0 \\ l_{2,1} & 1 & 0 \\ l_{3,1} & l_{3,2} & 1 \end{pmatrix} \times \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ 0 & 0 & u_{3,3} \end{pmatrix}$$

$$= \begin{pmatrix} \boxed{u_{1,1}} & \boxed{u_{1,2}} & \boxed{u_{1,3}} \\ \boxed{l_{2,1}} \boxed{u_{1,1}} & l_{2,1} u_{1,2} + \boxed{u_{2,2}} & l_{2,1} u_{1,3} + \boxed{u_{2,3}} \\ \boxed{l_{3,1}} \boxed{u_{1,1}} & l_{3,1} u_{1,2} + \boxed{l_{3,2}} \boxed{u_{2,2}} & l_{3,1} u_{1,3} + l_{3,2} u_{2,3} + \boxed{u_{3,3}} \end{pmatrix}$$

En prenant les équations obtenues dans le bon ordre (les colonnes de gauche à droite et les lignes de haut en bas) on remarque que l'on obtient un système à résoudre où à chaque étape, il n'y a qu'une seule inconnue.

$$u_{11} = a_{11} ; u_{12} = a_{12} ; u_{13} = a_{13} ; \quad l_{21} = a_{21} / u_{11} ; \quad l_{31} = a_{31} / u_{11}$$

$$u_{22} = a_{22} - l_{21} u_{12} ; \quad l_{32} = (a_{32} - l_{31} u_{12}) / u_{22} ; \quad u_{33} = a_{33} - l_{31} u_{13} - l_{32} u_{23}$$

# Algorithme de Crout

```
pour j de 1 à n faire
  pour i de 1 à j faire      // Calcul des  $r_{i,j}$ 
     $r_{i,j} \leftarrow a_{i,j}$ 
    pour k de 1 à i - 1 faire
       $r_{i,j} \leftarrow r_{i,j} - l_{i,k}r_{k,j}$ 
    fin pour
  fin pour
  pour i de j + 1 à n faire  // Calcul des  $l_{i,j}$ 
     $l_{i,j} \leftarrow a_{i,j}$ 
    pour k de 1 à j - 1 faire
       $l_{i,j} \leftarrow l_{i,j} - l_{i,k}r_{k,j}$ 
    fin pour
     $l_{i,j} \leftarrow l_{i,j} / r_{j,j}$ 
  fin pour
fin pour
```

# Méthodes itératives

- L'idée construire une suite de vecteurs qui converge vers le vecteur  $(X^{(k)})$ , solution du système  $A \cdot X = B$
- Principe du calcul d'un point fixe : limite de la suite construite.
- Procédé  $\rightarrow$  transformer  $A \cdot X = B$   $\longleftrightarrow$  en une égalité

$$A \cdot X = B \quad \Leftrightarrow \quad X = \varphi(X) = MX + N$$

- On est alors ramené à un problème de recherche de point fixe :

$$X^* = \varphi(X^*)$$

On définit la suite récurrente par :

- $X^{(0)}$  (vecteur initial fixé).
- la règle de récurrence pour  $(X^{(k+1)})_{k \in \mathbb{N}}$  :

$$X^{(k+1)} = \varphi(X^{(k)}) = MX^{(k)} + N \quad : \text{un système linéaire}$$

- Si la suite converge ( $k$  vers  $+\infty$ ), alors sa limite est solution du système

Si on écrit  $A$  sous la forme  $A = -E + D - F$  (une somme de matrices)

$$AX = B \Rightarrow (-E + D - F)X = B$$

$$DX = B + EX + FX$$

$$X = D^{-1}(B + EX + FX)$$

On choisit  $D$  pour qu'elle soit facilement inversible

$$A = \underbrace{\begin{pmatrix} 0 & \cdots & 0 \\ a_{21} & \ddots & \vdots \\ \vdots & & \\ a_{n1} & \cdots & a_{n(n-1)} & 0 \end{pmatrix}}_{(-E)} + \underbrace{\begin{pmatrix} a_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{nn} \end{pmatrix}}_{(D)} + \underbrace{\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & a_{n-1n} \\ 0 & \cdots & & 0 \end{pmatrix}}_{(-F)}$$

On constate que la matrice  $D^{-1}$  est facile à calculer

$$D^{-1} = \left( \frac{1}{a_{ii}} \right)_{i=1 \dots n} \text{ où } a_{ii} \neq 0$$

Sous cette forme  $AX = (-E + D - F)X$

Les méthodes Jacobi, Gauss-Seidel se distinguent dans la façon de répartir :  $D$ ,  $-E$  et  $-F$

## Méthode de Jacobi

On pose :  $M = D^{-1}(+E + F)$  et  $N = D^{-1}B$

$$AX = b \Rightarrow X = D^{-1}(B + EX + FX)$$

$$\begin{cases} X^{(0)} : \text{(vecteur initial fixé)} \\ X^{(k+1)} = D^{-1}(B + EX^{(k)} + FX^{(k)}) \end{cases}$$

En écrivant le système sous forme d'équations on a :

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

A l'étape 1 on a :

$$\begin{aligned} x_1^1 &= \frac{1}{a_{11}} (b_1 - a_{12} x_2^0 - \dots - a_{1n} x_n^0) \\ x_2^1 &= \frac{1}{a_{22}} (b_2 - a_{21} x_1^0 - a_{23} x_3^0 - \dots - a_{2n} x_n^0) \\ &\quad \vdots \\ x_n^1 &= \frac{1}{a_{nn}} (b_n - a_{n1} x_1^0 - a_{n2} x_2^0 - \dots - a_{nn-1} x_{n-1}^0) \end{aligned}$$

## Méthode de Gauss-Seidel

A partir de  $A = D - E - F$  on répartit  $D; E; F$   
 $M = (D - E)$  et  $N = (D - E)^{-1}F$

$$AX = b \Rightarrow X = (D - E)^{-1}X + (D - E)^{-1}B$$

Le calcul effectif peut se faire par un calcul matriciel

En calculant :  $(D - E)^{-1}$  :

$$M = (D - E)^{-1}F \text{ et } N = (D - E)^{-1}B$$

$$\begin{cases} X^{(0)} : (\text{vecteur initial fixé}) \\ X^{(k+1)} = (D - E)^{-1}FX^{(k)} + (D - E)^{-1}B \end{cases}$$

Ce calcul suppose le calcul de  $(D - E)^{-1}$

En général on passe par la formulation sous forme d'équation (plus simple à calculer) **C'est cette méthode qu'on adoptera ici**

Le calcul effectif se fait de la façon suivante

$$\begin{cases} X^{(0)} : \text{(vecteur initial fixé)} \\ (D - E)X^{(k+1)} = (B + FX^{(k)}) \end{cases}$$

Soit :

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



En écrivant le système sous forme d'équations on a :

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

A l'étape 1 on a :

$$x_1^1 = \frac{1}{a_{11}} (b_1 - a_{12} x_2^0 - \dots - a_{1n} x_n^0)$$

$$x_2^1 = \frac{1}{a_{22}} (b_2 - a_{21} x_1^1 - a_{23} x_3^0 - \dots - a_{2n} x_n^0)$$

$$x_n^1 = \frac{1}{a_{nn}} (b_n - a_{n1} x_1^1 - a_{n2} x_2^1 - \dots - a_{nn-1} x_{n-1}^1)$$

# Condition de convergence

- Une matrice  $A$  est dite à diagonale dominante si

$$\forall i, |a_{i,i}| > \sum_{j \neq i} |a_{i,j}|.$$

- **Théorème (CS)** : Les méthodes de Jacobi et Gauss-Seidel s'appliquent sur  $(A.X=B)$  et convergent si  $A$  est à diagonale dominante.
- Soit  $\rho(M) = \sup\{ |\lambda_i| \}$  où les  $\lambda_i$  sont les valeurs propres de la matrice
- $\rho(M)$  est appelé rayon spectral de  $M$
- **Théorème (CNS)** : si  $P = M^{-1} \times N$  est diagonalisable, alors pour tout  $X^{(0)}$ , la suite  $(X^{(k)})$  converge ssi  $\rho(M) < 1$ .

# Conditions d'arrêt

- Condition d'arrêt  
en général :

$$\frac{\|AX^{(k)} - B\|}{\|B\|} < \varepsilon$$

ou bien :

$$\|X^{(k+1)} - X^{(k)}\| < \varepsilon$$

# Complexité

- Chaque itération nécessite  $n(2n - 1)$  opérations, et plus précisément :
  - $n$  divisions
  - $n(n - 1)$  soustractions
  - $n(n - 1)$  multiplications
- Remarque 1: plus on fait d'itérations, plus le résultat est précis.
- Remarque 2 : Ces méthodes sont particulièrement intéressantes lorsqu'il s'agit de très grandes matrices ( $n > 100$ ) et on se contente dans ce cas d'une dizaine d'itérations.

Exemple : méthode Gauss-Seidel : passage par inversion de  $(D - E)^{-1}$

$$A = \begin{pmatrix} 2 & 1 & 1 \\ -1 & 1 & 2 \\ 1 & 0 & 2 \end{pmatrix}; B = \begin{pmatrix} 6 \\ 3 \\ 2 \end{pmatrix} \text{ on suppose } X^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

1<sup>ère</sup> méthode de résolution : calcul de  $(D - E)^{-1}$

On part de :  $(-E + D - F)X = B \Rightarrow (D - E)X = B + FX$

$$X = (D - E)^{-1}(B + FX)$$

On construit la suite récurrente  $X^{(k)}$  comme suit

$$\begin{cases} X^{(0)} \text{ Valeur initial} \\ X^{(k+1)} = (D - E)^{-1}(B + FX^{(k)}) \\ \text{Condition d'arrêt} \end{cases}$$

$$(D - E)^{-1} = \begin{pmatrix} 1/2 & 0 & 0 \\ 1/2 & 1 & 0 \\ -1/4 & 0 & 1/2 \end{pmatrix} ;$$

$$(D - E)^{-1}F = \begin{pmatrix} 1/2 & 0 & 0 \\ 1/2 & 1 & 0 \\ -1/4 & 0 & 1/2 \end{pmatrix} \begin{pmatrix} 0 & -1 & -1 \\ 0 & 0 & -2 \\ 0 & 0 & 0 \end{pmatrix}$$

$$(D - E)^{-1}F = \begin{pmatrix} 0 & -1/2 & -1/2 \\ 0 & -1/2 & -5/2 \\ 0 & 1/4 & 1/4 \end{pmatrix} ; (D - E)^{-1}B = \begin{pmatrix} 3 \\ 6 \\ -1/2 \end{pmatrix} ;$$

$$X^0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$X^{(1)} = \begin{pmatrix} 0 & -0.5 & -0.5 \\ 0 & -0.5 & -2.5 \\ 0 & 0.25 & 0.25 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 3 \\ 6 \\ -0.5 \end{pmatrix} = \begin{pmatrix} 3 \\ 6 \\ -0.5 \end{pmatrix}$$

$$X^{(2)} = \begin{pmatrix} 0 & -0.5 & -0.5 \\ 0 & -0.5 & -2.5 \\ 0 & 0.25 & 0.25 \end{pmatrix} \times \begin{pmatrix} 3 \\ 6 \\ -0.5 \end{pmatrix} + \begin{pmatrix} 3 \\ 6 \\ -0.5 \end{pmatrix} = \begin{pmatrix} 0.25 \\ 4.25 \\ 0.875 \end{pmatrix}$$

$$X^{(3)} = \begin{pmatrix} 0 & -0.5 & -0.5 \\ 0 & -0.5 & -2.5 \\ 0 & 0.25 & 0.25 \end{pmatrix} \times \begin{pmatrix} 0.25 \\ 4.25 \\ 0.875 \end{pmatrix} + \begin{pmatrix} 3 \\ 6 \\ -0.5 \end{pmatrix} = \begin{pmatrix} 0.4375 \\ 1.6875 \\ 0.7813 \end{pmatrix}$$

$$X^{(4)} = \begin{pmatrix} 0 & -0.5 & -0.5 \\ 0 & -0.5 & -2.5 \\ 0 & 0.25 & 0.25 \end{pmatrix} \times \begin{pmatrix} 0.4375 \\ 1.6875 \\ 0.7813 \end{pmatrix} + \begin{pmatrix} 3 \\ 6 \\ -0.5 \end{pmatrix} = \begin{pmatrix} 1.7656 \\ 3.203 \\ 0.1172 \end{pmatrix}$$

## Résolution par expressions équationnelles

Le calcul effectif se fait de la façon suivante

$$\begin{cases} X^{(0)} : \text{(vecteur initial fixé)} \\ (D - E)X^{(k+1)} = (B + FX^{(k)}) \Rightarrow DX^{(k+1)} = B + EX^{(k+1)} + FX^{(k)} \end{cases}$$

Soit :

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

$$A = \begin{pmatrix} 2 & 1 & 1 \\ -1 & 1 & 2 \\ 1 & 0 & 2 \end{pmatrix}; B = \begin{pmatrix} 6 \\ 3 \\ 2 \end{pmatrix} \text{ on a } X^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$



$$x_1^{(k+1)} = \frac{1}{2} \left( 6 - x_2^{(k)} - 0x_3^{(k)} \right)$$

$$x_2^{(k+1)} = \frac{1}{2} \left( 3 + x_1^{(k+1)} - 2x_3^{(k)} \right)$$

$$x_3^{(k+1)} = \frac{1}{2} \left( 2 - x_1^{(k+1)} - 0x_2^{(k+1)} \right)$$

## 1<sup>ère</sup> itération

$$x_1^{(1)} = \frac{1}{2} \left( 6 - x_2^{(0)} - 0x_3^{(0)} \right) = \frac{1}{2} (6 - 0 - 0) = 3$$

$$x_2^{(1)} = \frac{1}{2} \left( 3 + x_1^{(1)} - 2x_3^{(0)} \right) = (3 + 3 - 2 \times 0) = 6$$

$$x_3^{(1)} = \frac{1}{2} \left( 2 - x_1^{(1)} - 0x_2^{(1)} \right) = \frac{1}{2} (2 - 3 - 0 \times 0) = -\frac{1}{2}$$

## 2<sup>ème</sup> itération

$$x_1^{(2)} = \frac{1}{2} \left( 6 - x_2^{(1)} - 0x_3^{(1)} \right) = \frac{1}{2} \left( 6 - 6 - \left(-\frac{1}{2}\right) \right) = \frac{1}{4} = 0.25$$

$$x_2^{(2)} = \frac{1}{2} \left( 3 + x_1^{(2)} - 2x_3^{(1)} \right) = \left( 3 + \frac{1}{4} - 2 \times \left(-\frac{1}{2}\right) \right) = \frac{17}{4} = 4.25$$

$$x_3^{(3)} = \frac{1}{2} \left( 2 - x_1^{(2)} - 0x_2^{(2)} \right) = \frac{1}{2} \left( 2 - \frac{1}{4} - 0 \times \frac{17}{4} \right) = \frac{7}{8} = 0.875$$

### 3<sup>ème</sup> itération

$$x_1^{(3)} = \frac{1}{2} \left( 6 - x_2^{(2)} - 0x_3^{(2)} \right) = \frac{1}{2} \left( 6 - \frac{17}{4} - \frac{7}{8} \right) = \frac{7}{16} = 0.4375$$

$$x_2^{(3)} = \frac{1}{1} \left( 3 + x_1^{(3)} - 2x_3^{(2)} \right) = \left( 3 + \frac{7}{16} - 2 \times \frac{7}{8} \right) = \frac{27}{16} = 1.6875$$

$$x_3^{(3)} = \frac{1}{2} \left( 2 - x_1^{(3)} - 0x_2^{(3)} \right) = \frac{1}{2} \left( 2 - \frac{7}{16} \right) = -\frac{25}{32} = 0.7813$$

### 4<sup>ème</sup> itération

$$x_1^{(4)} = \frac{1}{2} \left( 6 - x_2^{(3)} - 0x_3^{(3)} \right) = \frac{1}{2} \left( 6 - \frac{27}{16} - \frac{25}{32} \right) = \frac{113}{64} = 1.7656$$

$$x_2^{(4)} = \frac{1}{1} \left( 3 + x_1^{(4)} - 2x_3^{(3)} \right) = \left( 3 + \frac{113}{64} - 2 \times \left( -\frac{25}{32} \right) \right) = \frac{205}{64} = 3.2031$$

$$x_3^{(4)} = \frac{1}{2} \left( 2 - x_1^{(4)} - 0x_2^{(4)} \right) = \frac{1}{2} \left( 2 - \frac{113}{64} - 0 \times \frac{205}{64} \right) = \frac{15}{128} = 0.1172$$

# Retour aux méthodes : méthode de Jordan

- Méthode : basée sur une diagonalisation
  - Utilise une suite de combinaison linéaires entre les différentes lignes, travaille sur la matrice élargie (voir méthode de Gauss)
  - Utilisation particulière de Gauss
- 
- $AX=B \xrightarrow{\quad} L X=B^{(k)}$  avec L matrice diagonale.
  - Complexité (globalement la même que Gauss)
  - Complexité de la résolution du système triangulaire en ?
  - Complexité de la triangulation en ?

# Chapitre 3

Racines de fonctions  $F(x)=0$   
 $F$  : fonction non linéaire

## Problème général

Soit une fonction  $f : \mathbb{R}^n \rightarrow \mathbb{R}^p$ .

Le problème est de trouver (en temps fini) par une méthode approchée, des solutions de l'équation  $f(x) = 0$

□

$f : \mathbb{R} \rightarrow \mathbb{R}$ .

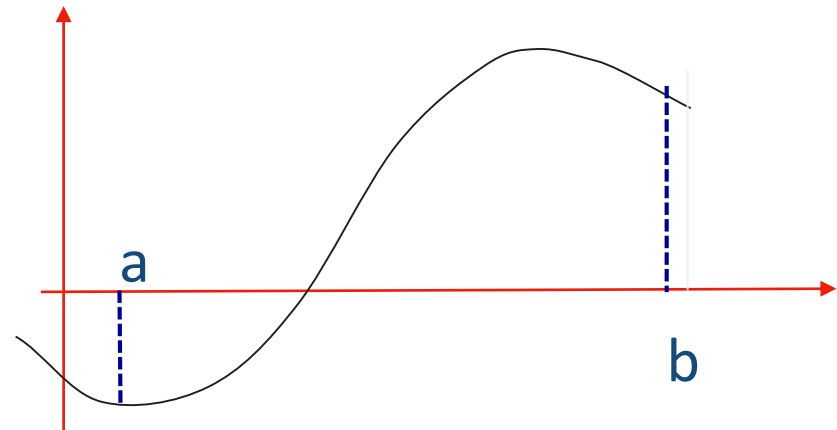
Théorème (zéro d'une fonction)

Soit  $f$  une fonction continue

$f : [a, b] \rightarrow \mathbb{R}$

si  $f(a)f(b) \leq 0$ , alors

$\exists \alpha \in ]a, b[$  tel que  $f(\alpha) = 0$



# Schéma général de l'approche pour la résolution

$$f : R \rightarrow R.$$

On transforme la forme de l'équation:

$$f(x) = 0 \Leftrightarrow \varphi(x) = x \quad \text{on construit la suite :}$$

$$X^{k+1} = \varphi(X^k) \quad \text{et} \quad \lim_{k \rightarrow \infty} X^k = X$$

on s'appuie sur le principe du point fixe :  $X^*$  tq :  $\varphi(X^*) = X^*$

La solution est déterminée avec une précision  $\varepsilon$  donnée :

$$|\varphi(x^{(k)}) - x^{(k-1)}| \leq \varepsilon$$

On passe par des méthodes itératives ; il faut avoir :

- un point de départ  $x^{(0)}$  → initialisation
- la fonction  $\varphi(x) = x$  pour chaque méthode (règle de l'itération).
- définir les conditions d'arrêt de l'itération

$$\text{Fonction d'itération} \begin{cases} x^{(1)} = \varphi(x^{(0)}) \\ x^{(2)} = \varphi(x^{(1)}) \\ x^{(k)} = \varphi(x^{(k-1)}) \end{cases} \text{ on suppose } x^{(k-1)} \text{ connu}$$

➤ Si la suite  $x^{(k)}$  converge une limite  $x^*$  lorsque  $k \rightarrow \infty$

Alors  $x^*$  est solution de l'équation  $x = \varphi(x)$

➤ critère d'arrêt :  $x^{(k)}$  proche d'une solution de l'équation  $x = \varphi(x)$ .

❖ Par exemple :

✓ la suite  $X^{(n)}$  devient stationnaire :  $|X^{(k)} - X^{(k-1)}| \leq \epsilon$

✓  $|f(X^{(k)})| \leq \epsilon$

## ❖ Récapitulatif

On considère l'équation (1)  $f(x) = 0$  :  $f$  continue et dérivable.

Résoudre le problème (1)  $\Leftrightarrow$  répondre aux 3 points suivants :

- Définir une suite itérative  $x^{(k+1)} = \varphi(x^{(k)})$  (trouver une méthode adaptée).
- Trouver un point de départ  $x^{(0)}$  (voir conditions de convergence).
- Déterminer un critère d'arrêt (précision).

Temps fini  $\Rightarrow$  la vitesse de convergence de la suite  $(x^{(k)})$ .

Remarques : Convergence  $\rightarrow$  existence de la solution + choix de  $x^{(0)}$ .

Propagation d'erreur peut entraîner une divergence



- Condition d'existence : *théorème des valeurs intermédiaires*
  - Si :  $f$  est continue sur  $[x_1, x_2]$  et  $f(x_1) \cdot f(x_2) \leq 0$
  - Alors  $\exists x_0 \in [x_1, x_2] : f(x_0) = 0$
- Méthode d'itération : Théorème du point fixe ( $f$  continue) :
  - $f(x) = 0 \Leftrightarrow \varphi(x) = x \rightarrow$  on construit la suite
  - $x^{(k+1)} = \varphi(x^{(k)})$  et  $\lim x^{(k)} = x^* \Rightarrow \varphi(x^*) = x^*$
- Condition de convergence : application du théorème des accroissements finies

Rappel du Théorème des accroissements finis

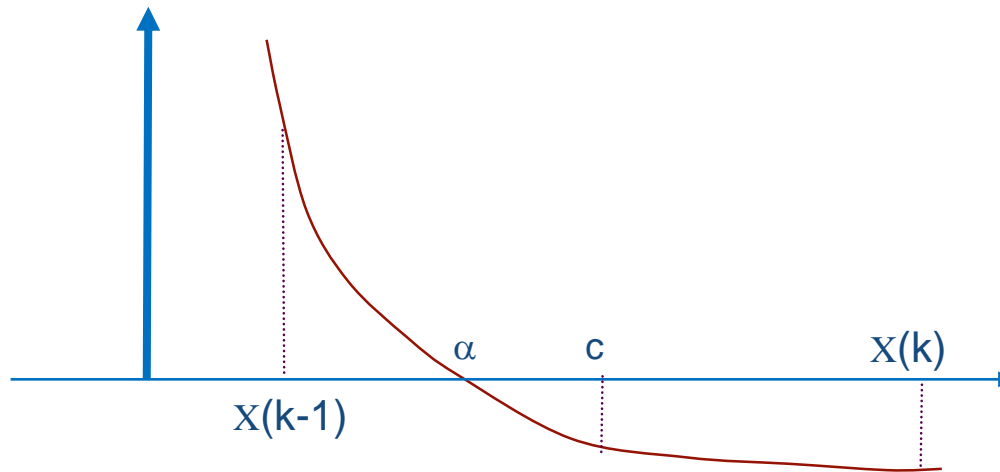
$f : [a, b] \rightarrow \mathbb{R}$ , continue sur  $[a, b]$ , dérivable sur  $]a, b[$ , il existe  $c \in ]a, b[$  tel que

$$f'(c) = \frac{f(b) - f(a)}{b - a}$$

- Cqce du Th. des accroissement finis :  $\varphi$  contractante ssi :
  - $|\varphi(x) - \varphi(y)| \leq c|x-y|$
  - $x = x(k), y = x(k-1) \Rightarrow |x(k+1) - x(k)| \leq c|x(k) - x(k-1)| \leq c^k |x(1) - x(0)|$
- Si  $\varphi$  n'est définie que sur un domaine  $D$ , il faut choisir  $x(0)$  dans  $D$  et vérifier que  $\varphi(D) \subset D$ .

- *Ordre de convergence : Soit  $x^{(*)}$ , un point fixe de  $\varphi$* 
  - si pour tout  $x^{(k)}$  dans le voisinage de  $x^*$ , on a la relation :
$$|x^{(k+1)} - x^*| \leq C \cdot |x^{(k)} - x^*|^p$$
pour tout  $k \geq 0$ , avec  $C < 1$  si  $p \geq 1$ ; on dit que  $\varphi$  est d'ordre au moins **p** pour déterminer  $x^{(*)}$  .
- $p = 1$  : convergence linéaire
- $p = 2$  : convergence quadratique

# Méthode de la bisection (dichotomie)



$$x_{n+1} = \varphi(x_n) = \frac{x_n + x_{n-1}}{2}$$

La règle de production

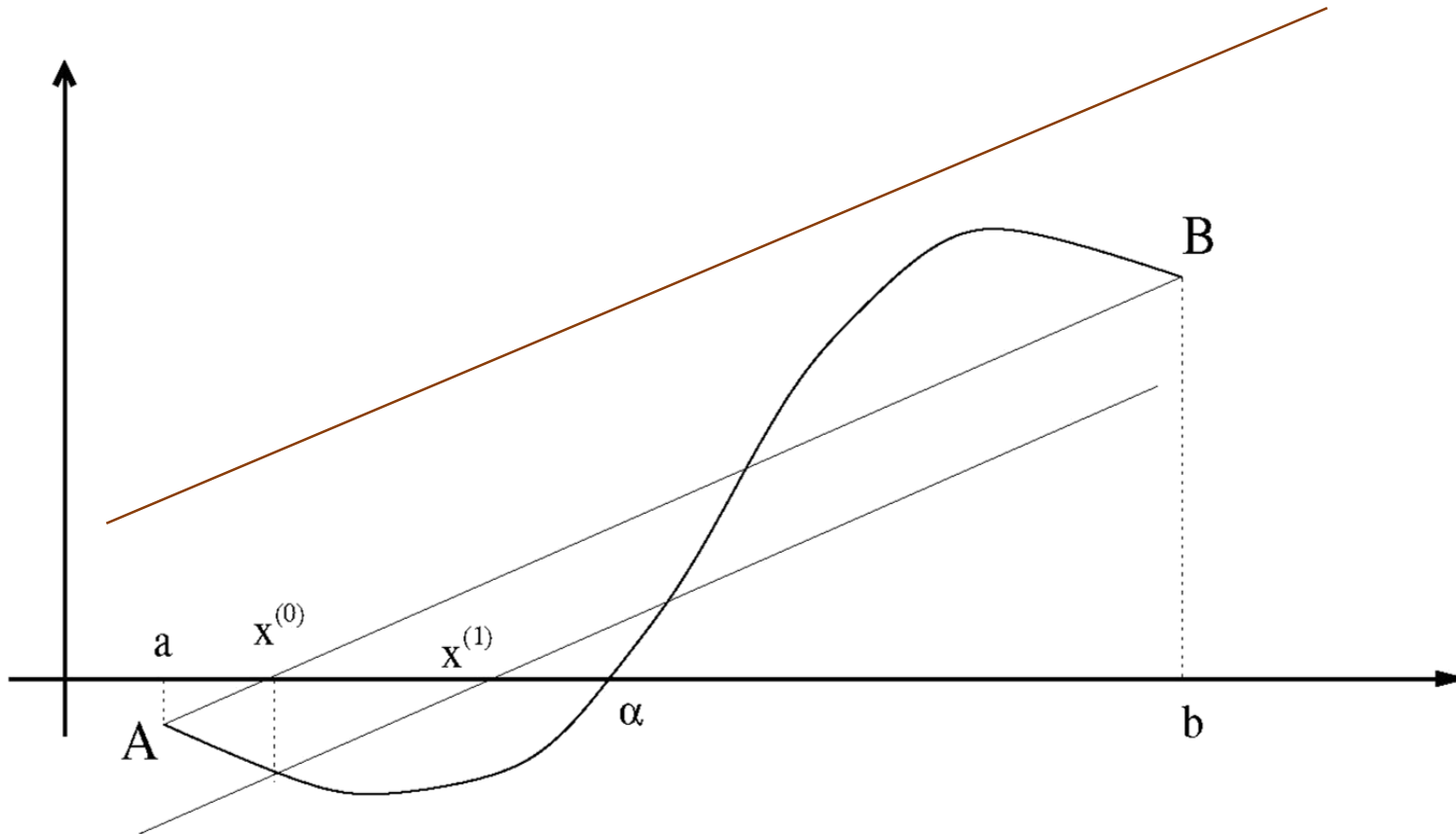
## Algorithme : méthode de dichotomie

$$a^{(0)} = a, b^{(0)} = b, \text{ et } x^{(0)} = \frac{a^{(0)} + b^{(0)}}{2}.$$

Pour  $k \geq 0$  et tant que  $|I_k| = |b^{(k)} - a^{(k)}| > \epsilon$

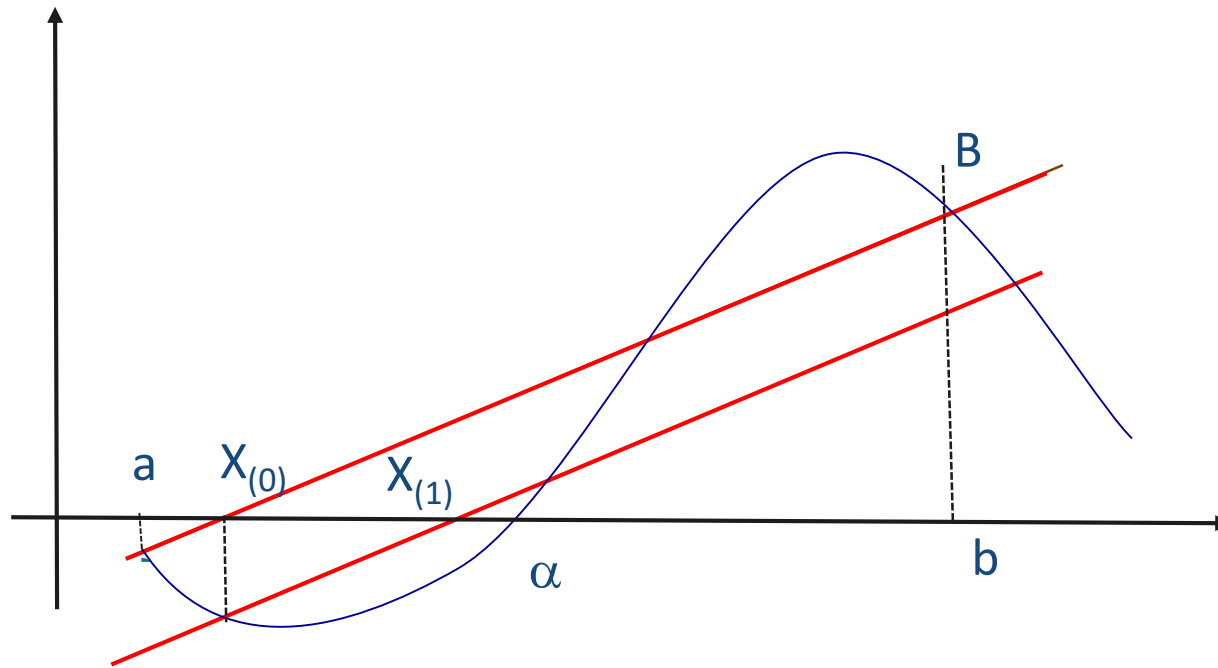
- Si  $f(x^{(k)}) = 0$  alors  $x^{(k)}$  est la racine  $\alpha$ .
- Si  $f(x^{(k)})f(a^{(k)}) < 0$ 
  - $a^{(k+1)} = a^{(k)}, b^{(k+1)} = x^{(k)}$
- Si  $f(x^{(k)})f(b^{(k)}) < 0$ 
  - $a^{(k+1)} = x^{(k)}, b^{(k+1)} = b^{(k)}$
- $x^{(k+1)} = \frac{a^{(k)} + b^{(k)}}{2}$

# Méthode de la corde



Si la méthode converge, elle converge avec un ordre  $p = 1$ .

## Méthode de la corde (ou la sécante)



$$f(x_n) + \frac{f(b) - f(a)}{b - a}(x_{n+1} - x_n) = 0$$

On peut exprimer la suite recherchée par:

$$\varphi(x_n) = x_{n+1} = x_n - \frac{b - a}{f(b) - f(a)} f(x_n)$$

La méthode de la corde peut être écrite sous la forme d'itération de point fixe  $x_{n+1} = \varphi(x_n)$  où

$$\phi(x) = x - \frac{b-a}{f(b)-f(a)}f(x)$$

Puisque

$$\phi'(x) = 1 - \frac{b-a}{f(b)-f(a)}f'(x)$$

la condition de convergence locale  $|\phi'(\alpha)| < 1$  est équivalente à

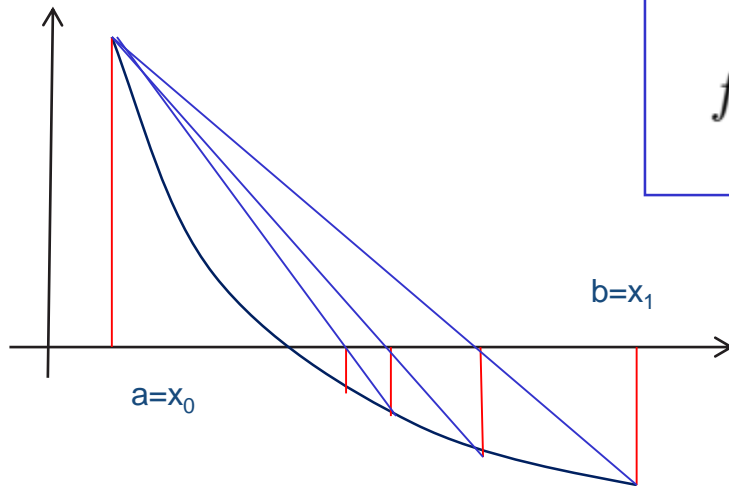
$$0 < \frac{b-a}{f(b)-f(a)}f'(\alpha) < 2$$

Sauf le cas exceptionnel où  $\phi'(\alpha) = 0$ , la convergence est **linéaire**.



# Méthode de fausse position (Regula falsi)

Cette méthode combine les possibilités de la dichotomie et la méthode de la sécante. On considère un intervalle  $[a, b]$  qui contient un zéro de la fonction  $f$ .  $(f(a) \cdot f(b) < 0 ; f \text{ continue})$

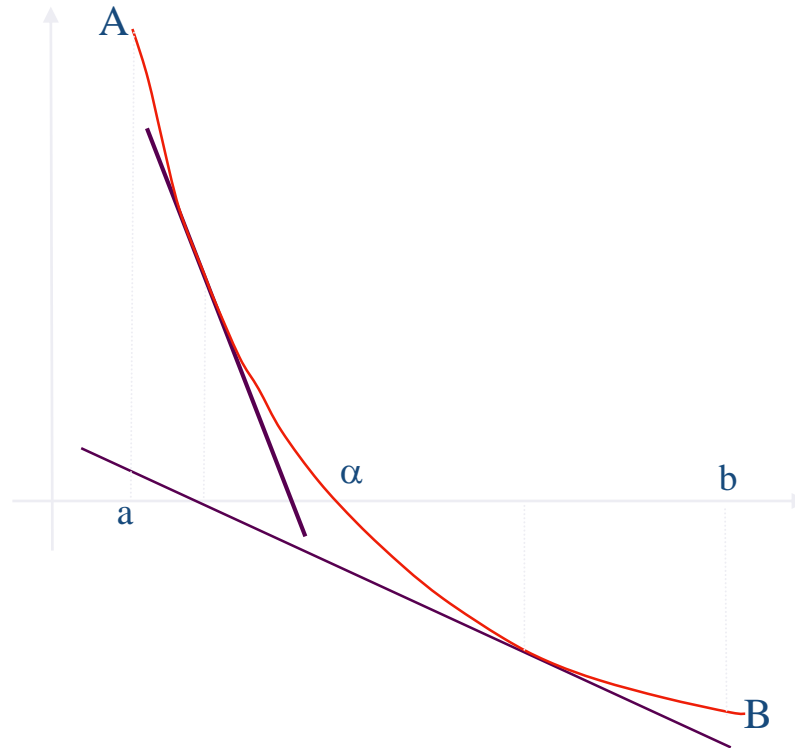


$$f(x_n) + \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}(x_{n+1} - x_n) = 0$$

Ce qui donne :

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n)$$

# Méthode de Newton (-Raphson)



*Convergence locale:* si  $x^{(0)}$  est assez proche de  $\alpha$  et  $f'(\alpha) \neq 0$ , la méthode converge avec un ordre  $p = 2$ .

## Méthode de Newton : expression de la suite $(x_n)$

Pour la méthode de Newton on utilise le développement de Taylor à l'ordre 1 au voisinage de  $(x_n)$  on obtient :

$$f(x_{n+1}) = f(x_n) + (x_{n+1} - x_n)f'(x_n)$$

D'où, si on cherche le point  $(x_{n+1})$  tel que  $f(x_{n+1}) = 0$   
Obtient :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad \text{avec } f'(x_n) \neq 0$$

$$\text{donc ici } \varphi(x) = x - \frac{f(x)}{f'(x)} \quad f'(x) \neq 0$$

Pour la convergence

En supposant  $f'(\alpha) \neq 0$  on obtient

$$\phi'(x) = 1 - \frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2} \Rightarrow \phi'(\alpha) = 0$$

La méthode est convergente localement. On peut montrer qu'elle est convergente d'ordre  $p = 2$ .

## A propos de la convergence

- $|I_0| = |b - a|$
- $|I_k| = |b^{(k)} - a^{(k)}| = \frac{|I_0|}{2^k} = \frac{|b-a|}{2^k}$  pour  $k \geq 0$
- En notant  $e^{(k)} = \alpha - x^{(k)}$  l' *erreur absolue* à l'étape  $k$ , on déduit que

$$|e^{(k)}| = |\alpha - x^{(k)}| \leq \frac{|I_k|}{2} = \frac{|b-a|}{2^{k+1}} \quad \text{pour } k \geq 0$$

ce qui entraîne

$$\lim_{k \rightarrow \infty} |e^{(k)}| = 0$$

- Donc la méthode de la bisection est *globalement convergente*.

# Chapitre 4

## Interpolation et approximation

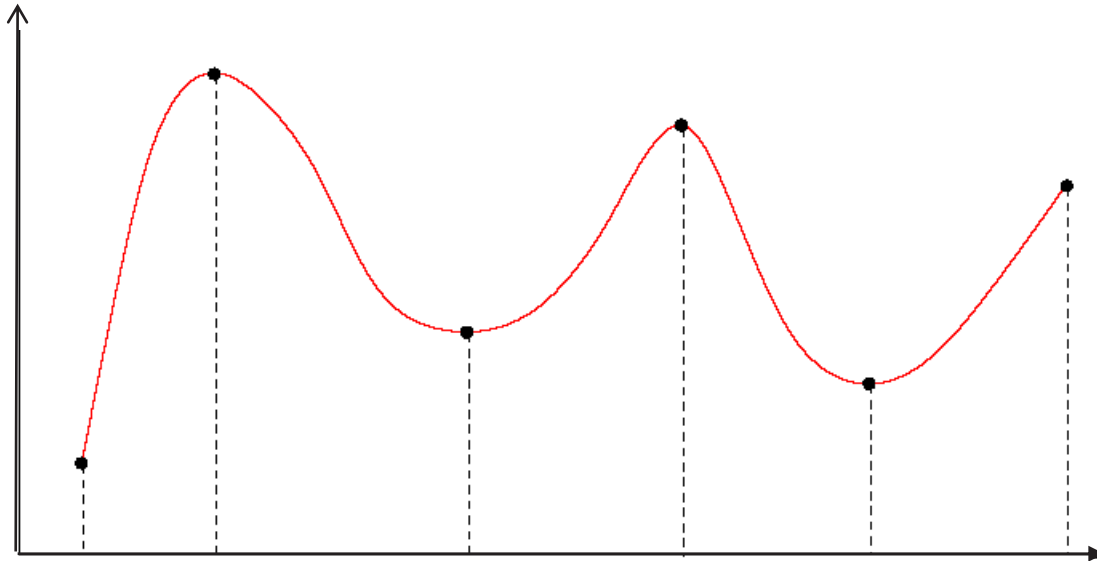
# Problème

- Données :
  - un ensemble de points connus  $(x_i, Y_i)$  ; ou  $Y_i \in \mathbb{R}^p$ 
    - Obtenus par un ensemble de mesures( relevés terrains)
    - ou bien calculé par l'estimation  $(x_i, f(x_i))$  d'une fonction  $f$  au points  $x_i$
- But : déterminer un "modèle" mathématique pour  $f$ 
  - réduire  $f$  en une expression simple (exemple : polynôme)
  - bonnes propriétés : dérivabilité, etc.
- Dans quels cas ?
  - définir un modèle mathématique à partir d'un nombre discret de mesures
  - analyser un phénomène étudié de manière empirique
  - remplacer une équation de courbe "compliquée" par une fonction polynomiale par exemple.

# Interpolation

les  $x_i$ , sont des mesures exactes

On veut que la courbe passe par tous les  $(x_i, f(x_i))$





## On se donne

- une fonction  $f: \mathbb{R} \rightarrow \mathbb{R}$  inconnue et continue sur un intervalle  $[a, b]$ .
- un ensemble de points connus  $(x_i, y_i), i \in [0, n]$ .
  - $\{x_0, x_1, \dots, x_n\}$  est le support de l'interpolation

## On cherche

une fonction  $\varphi : \mathbb{R} \rightarrow \mathbb{R}$  telle que  
 $\varphi(x_i) = f(x_i), i \in [0, n]$ .

- En pratique,  $\varphi$  est une somme de fonctions

$$\varphi(x) = \sum_{i=0}^n a_i \varphi_i(x)$$

vérifiant

$$f(x_i) = \varphi(x_i) \text{ avec } (x_i) \in R^n \quad (1)$$

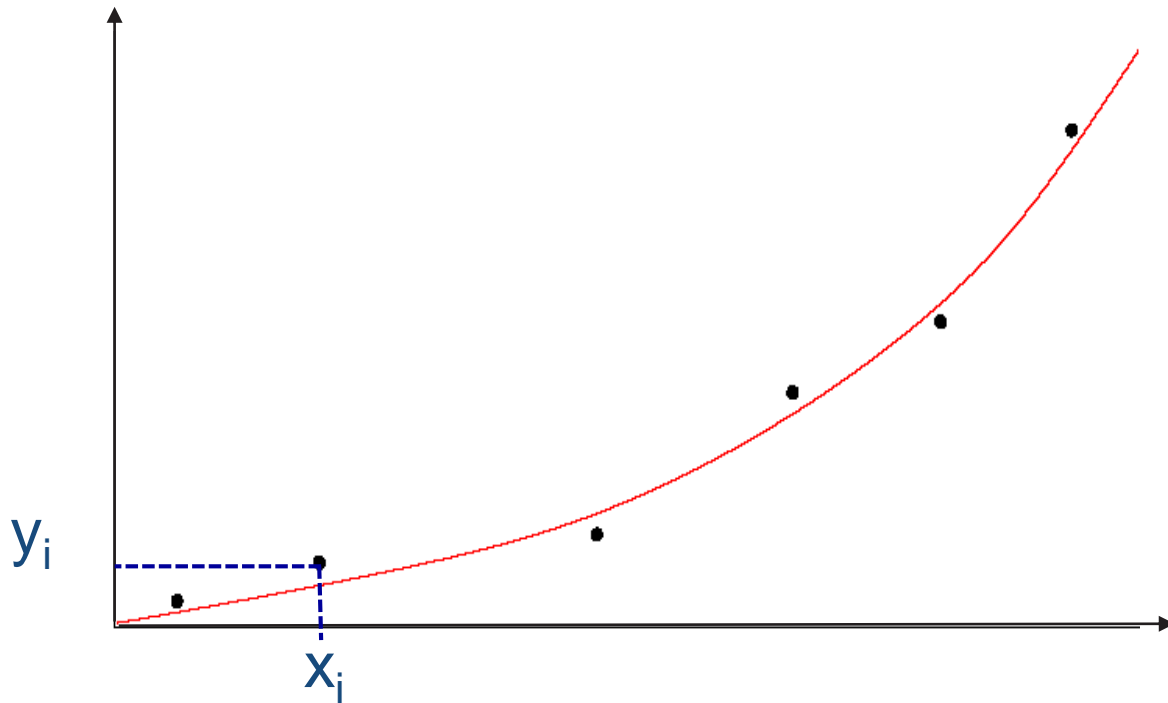
$\varphi_j$ : fonctions de la base dans laquelle on exprime  $f$  ;  $\varphi_i$  doit se prêter aux traitements numériques courants.

Problème : déterminer les  $a_i$  pour vérifier (1) et assurer l'unicité de la solution donc de  $a_i$

# Approximation

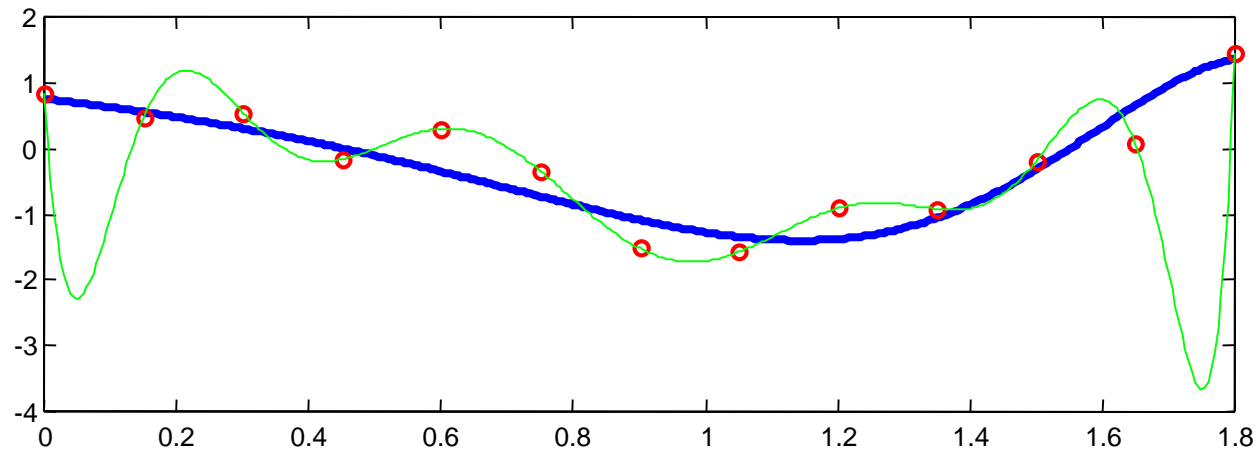
les  $(x_i, y_i)$  sont des mesures données

Objet de l'étude : déterminer la courbe s'approchant au mieux des points  $(x_i, f(x_i))$



# Approximation

- En général, on se restreint à une famille de fonctions connues
  - polynômes,
  - exponentielles, logarithme
  - fonctions trigonométriques...



# Quelques méthodes d'interpolation

- Interpolation polynomiale
  - polynômes de degré au plus  $n$ 
    - ▶ polynômes de Lagrange
    - ▶ différences finies de Newton
- Interpolation par splines
  - polynômes par morceaux
- Interpolation d'Hermite (ce chapitre ne sera pas traité)
  - informations sur les dérivées de la fonction à approcher

# Théorème de Weierstrass

soit  $f$  fct continue sur  $[a, b]$

Alors,  $\forall \varepsilon > 0$ , il existe un polynôme  $P(x)$ , défini sur  $[a, b]$   
tel que :

$$|f(x) - P(x)| < \varepsilon \quad \forall x \in [a, b]$$

plus  $\varepsilon$ , est petit,  
plus l'ordre du polynôme est grand

Interpolation :

$n + 1$  points,  $n + 1$  contraintes,  $n + 1$  équations,  
 $n + 1$  inconnues: ordre du polynôme  $n$

# Interpolation polynomiale

- Le problème : Solution recherchée
- Données -->  $(x_0, y_0 = f(x_0)), \dots, (x_i, y_i = f(x_i)), \dots, (x_n, y_n = f(x_n))$
- Solution -->  $P(x)$  tel que  $P(x_i) = f(x_i), i = 0, n$
- mauvaise solution : résoudre le système linéaire

$$P(x) = \sum_{i=0}^n a_i x^i$$

- la combinaison linéaire de polynômes est un polynôme

# Ch 4 - suite

## Interpolation de Lagrange

## Newton



# Interpolation polynomiale

la combinaison linéaire de polynômes est un polynôme

$$(x_0, y_0 = f(x_0)), \dots, (x_i, y_i = f(x_i)), \dots, (x_n, y_n = f(x_n))$$

$$P(x) \text{ tel que } P(x_i) = f(x_i), \quad i = 0, n$$

→ Idée de Lagrange

$$P(x) = y_0 P_0(x) + \dots + y_i P_i(x) + \dots + y_n P_n(x)$$

$$\text{tel que } P_i(x_i) = 1 \quad \text{et} \quad P_i(x_j) = 0 \quad j \neq i$$

$$\text{ainsi } P(x_i) = y_0 P_0(x_i) + \dots + y_i P_i(x_i) + \dots + y_n P_n(x_i)$$

↓

↓

↓

0

1

0

# Méthode de Lagrange pour l'interpolation polynômiale

➔ Idée changer de base pour les polynômes

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n+1} \frac{(x - x_j)}{(x_i - x_j)}$$

$$L(x) = \sum_{i=0}^n y_i L_i$$

L est un polynôme d'ordre n

- Théorème

- Soient  $n+1$  points distincts de coordonnée  $(x_i, y_i)$   
avec  $x_i, y_i$  réels

il existe un unique polynôme  $p \in P_n$  tel que  $p(x_i) = y_i$  pour  $i = 0$  à  $n$

## ● Théorème

- Soient  $n+1$  points distincts  $x_i$  réels et  $n+1$  réels  $y_i$ , il existe un unique polynôme  $p \in P_n$  tel que  $p(x_i) = y_i$  pour  $i = 0$  à  $n$

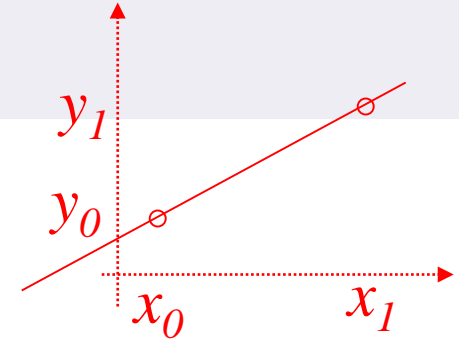
### Idée de démonstration

- **Construction de  $p$  :**  
avec  $L_i$  polynôme de Lagrange

$$p(x) = \sum_{i=0}^n y_i L_i(x)$$

- **Propriétés de  $L_i$** 
  - ▶  $L_i(x_i) = 1$
  - ▶  $L_i(x_j) = 0 \quad (j \neq i)$

# Lagrange : degré 1



## Exemple avec $n=1$

- on connaît 2 points  $(x_0, y_0)$  et  $(x_1, y_1)$
- on cherche la droite  $y=ax+b$  (polynôme de degré 1) qui passe par les 2 points :

►  $y_0 = a x_0 + b$

►  $y_1 = a x_1 + b$

*en résolvant le système*

$$a = (y_0 - y_1) / (x_0 - x_1)$$

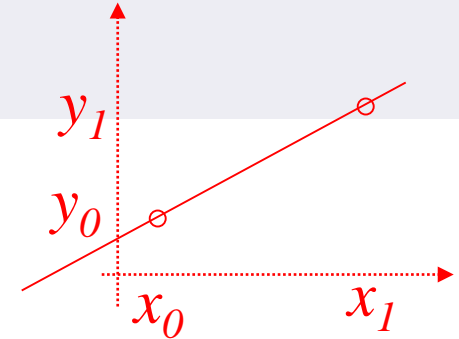
$$b = (x_0 y_1 - x_1 y_0) / (x_0 - x_1)$$

*en passant par l'expression de Lagrange*

$$y = \frac{y_0 - y_1}{x_0 - x_1} x + \frac{x_0 y_1 - x_1 y_0}{x_0 - x_1}$$

$$y = y_0 \frac{x - x_1}{x_0 - x_1} - y_1 \frac{x - x_0}{x_1 - x_0} = y_0 \underbrace{\frac{x - x_1}{x_0 - x_1}}_{L_0(x)} + y_1 \underbrace{\frac{x - x_0}{x_1 - x_0}}_{L_1(x)}$$

# Lagrange : degré 1



## • Exemple avec $n=1$

- on connaît 2 points  $(x_0, y_0)$  et  $(x_1, y_1)$
- on cherche la droite  $y=ax+b$  (polynôme de degré 1) qui passe par les 2 points :

$$\diamond y_0 = a x_0 + b$$

$$\diamond y_1 = a x_1 + b$$

*en résolvant le système*

$$a = (y_0 - y_1) / (x_0 - x_1)$$

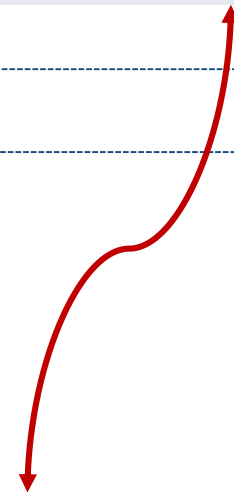
$$b = (x_0 y_1 - x_1 y_0) / (x_0 - x_1)$$

$$y = \frac{y_0 - y_1}{x_0 - x_1} x + \frac{x_0 y_1 - x_1 y_0}{x_0 - x_1}$$

*en passant par l'expression de Lagrange*

$$y = y_0 \underbrace{\frac{x - x_1}{x_0 - x_1}}_{L_0(x)} + y_1 \underbrace{\frac{x - x_0}{x_1 - x_0}}_{L_1(x)}$$

$$\text{Après réduction : } y = \frac{y_0 - y_1}{x_0 - x_1} x + \frac{x_0 y_1 - x_1 y_0}{x_0 - x_1}$$

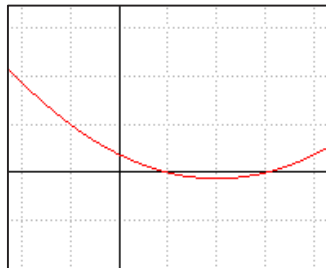


# Lagrange : degré 2

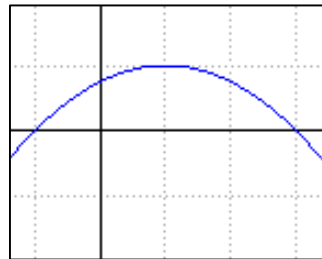
- Exemple avec  $n=2$

- on connaît 3 points  $(-1,1)$ ,  $(1,4)$  et  $(3,16)$
- polynômes de Lagrange associés :
  - Espace vectoriel : avec  $\{L_i\}$  base de l'interpolation

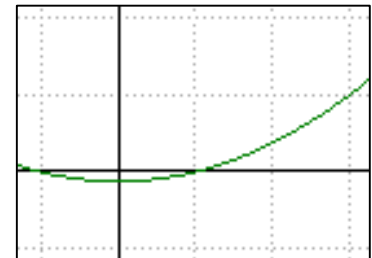
$$L_0(x) = \frac{(x-1)(x-3)}{8}$$



$$L_1(x) = \frac{(x+1)(x-3)}{-4}$$



$$L_2(x) = \frac{(x+1)(x-1)}{8}$$



## Lagrange : degré 2

- calcul du polynôme d'interpolation

*points :  $(-1,1)$ ,  $(1,4)$  et  $(3,16)$*

►  $p(x) = l_0(x) + 4l_1(x) + 16l_2(x)$

► 
$$p(x) = \frac{(x-1)(x-3)}{(-1-1)(-1-3)} + 4 \frac{(x+1)(x-3)}{(1-(-1))(1-3)} + 16 \frac{(x+1)(x-1)}{(3+1)(3-1)}$$

► en développant, on trouve  $p(x) = \frac{9}{8}x^2 + \frac{3}{2}x + \frac{11}{8}$

# Lagrange : Algorithme

Fonction  $y = \text{Lagrange}(x, x_i, y_i)$

```
pour  $i = 1$  à  $n$ 
  pour  $j = 1$  à  $n, j \neq i$ 
     $l \leftarrow l * \frac{x - x_i(j)}{x_i(i) - x_i(j)}$ 
  fin pour
   $y \leftarrow y + y_i * l$ 
fin pour
```

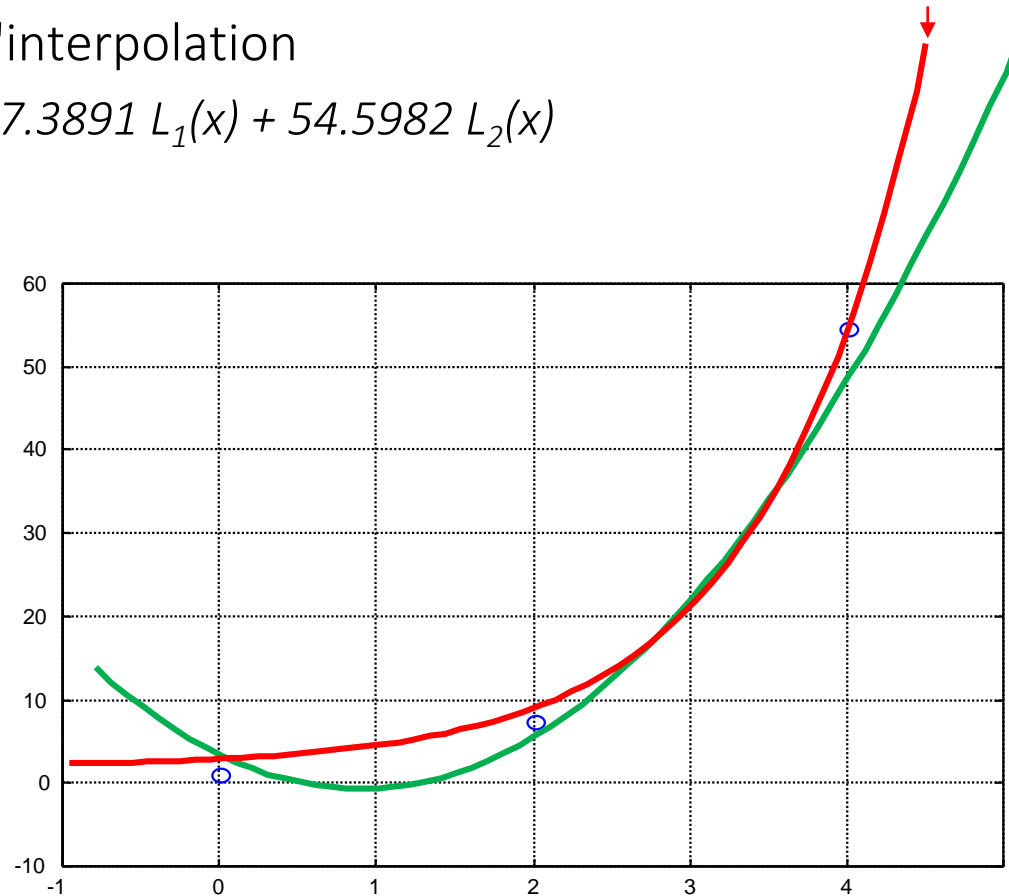
Donner la complexité de l'algorithme !



## Lagrange : exemple n°3

○ Exemple avec  $n=2$  (fonction à approcher  $y=e^x$ )

- on connaît 3 points  $(0,1)$ ,  $(2,7.3891)$  et  $(4,54.5982)$
- Polynôme d'interpolation
- ▶  $p(x) = L_0(x) + 7.3891 L_1(x) + 54.5982 L_2(x)$



# Lagrange : estimation de l'erreur d'interpolation

- Erreur d'interpolation  $e(x) = \|f(x) - p(x)\|$

## □ Théorème :

- si  $f$  est  $n+1$  dérivable sur  $[a,b]$ ,  $\forall x \in [a,b]$ , notons :
  - $I$  le plus petit intervalle fermé contenant  $x$  et les  $x_i$
  - $\phi(x) = (x-x_0)(x-x_1)\dots(x-x_n)$
- alors, il existe  $\xi \in I$  tel que

$$e(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \phi(x)$$

- NB :  $\xi$  est dans le voisinage de  $x$
- Utilité = on contrôle l'erreur d'interpolation donc la qualité de l'interpolation (voir exercice fait en TD)

## Lagrange : choix de $n$ (nombre de points ou nœuds)

- La méthode de Lagrange présente l'inconvénient de ne pas être incrémentale ; si on veut passer du degré  $n$  au degré  $n+1$ , il faut refaire les calculs → amélioration schéma de Neville-Aitken
- Méthode de Neville :
  - on augmente progressivement  $n$
  - on calcule les  $p_i$  de manière récursive
  - on arrête dès que l'erreur est inférieure à un seuil  
(d'où l'utilité du calcul de l'erreur)

# La méthode de Neville

Méthode récursive du calcul de la valeur du polynôme d'interpolation en un point donné, il est aisé d'ajouter des points d'interpolation au fur et à mesure.

$$p_{i,0}(x) = y_i, \quad 0 \leq i \leq n,$$
$$p_{i,j+1}(x) = \frac{(x_i - x)p_{i+1,j}(x) + (x - x_{i+j+1})p_{i,j}(x)}{x_i - x_{i+j+1}}, \quad 0 \leq j < n \text{ et } 0 \leq i < n - j.$$

# Algorithme de Neville-Aitken

- Application

$$p_{0,0}(x) = y_0$$

$$p_{0,1}(x)$$

$$p_{1,0}(x) = y_1$$

$$p_{0,2}(x)$$

$$p_{1,1}(x)$$

$$p_{0,3}(x)$$

$$p_{2,0}(x) = y_2$$

$$p_{1,2}(x)$$

$$p_{0,4}(x)$$

$$p_{2,1}(x)$$

$$p_{1,3}(x)$$

$$p_{3,0}(x) = y_3$$

$$p_{2,2}(x)$$

$$p_{3,1}(x)$$

$$p_{4,0}(x) = y_4$$

$x_i$	1	2	3	5
$y_i$	1	4	2	5

$i \backslash j$	0	1	2	3
0	1	$3x - 2$	$-\frac{5}{2}x^2 + \frac{21}{2}x - 7$	$\frac{11}{12}x^3 - 8x^2 + \frac{247}{12}x - \frac{25}{2}$
1	4	$-2x + 8$	$\frac{7}{6}x^2 - \frac{47}{6}x + 15$	
2	2	$\frac{3}{2}x - \frac{5}{2}$		
3	5			

# L'algorithme de Neville

Fonction  $y = \text{Neville}(x, x_i, y_i)$

pour  $i = 1$  à  $n$

$Q(i,0) \leftarrow y_i(i)$

fin pour

pour  $i = 1$  à  $n$

pour  $j = 1$  à  $i$

$$Q(i,j) \leftarrow \frac{(x - x_i(i-j))Q(i,j-1) - (x - x_i(i))Q(i-1,j-1)}{x_i(i) - x_i(i-j)}$$

fin pour

$y \leftarrow Q(n,n)$

fin pour

Vérifier : complexité du calcul :  $n^2$

# Méthode de Newton pour l'interpolation polynomiale :

- Polynômes de Newton :

- Base =  $\{1, (x - x_0), (x - x_0)(x - x_1), \dots, (x - x_0)(x - x_1) \dots (x - x_{n-1})\}$

- on peut ré-écrire  $p(x)$  :

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \dots + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1})$$

- calcul des  $a_k$  : méthode des différences divisées



# Newton : différences divisées

## ○ Définition :

Soit une fonction  $f$  dont on connaît les valeurs en des points distincts  $a, b, c, \dots$

On appelle différence divisée d'ordre  $0, 1, 2, \dots, n$  les expressions définies par récurrence sur l'ordre  $k$  :

$$\blacktriangleright k = 0 \quad f[a] = f(a)$$

$$\blacktriangleright k = 1 \quad f[a, b] = (f[b] - f[a]) / (b - a)$$

$$\blacktriangleright k = 2 \quad f[a, b, c] = (f[a, c] - f[a, b]) / (c - b)$$

$\blacktriangleright \dots$

$$\blacktriangleright f[X, a, b] = (f[X, b] - f[X, a]) / (b - a)$$

$a \quad X, \quad b \quad X, \quad a \quad b$

# Newton : différences divisées

- Détermination des coefficients de  $p(x)$  dans la base de Newton :

## Théorèmes

calcul des coefficients de newton

$$a_k = f[x_0, x_1, \dots, x_k] \quad \text{avec } k = 0 \dots n$$

Calcul de l'erreur d'interpolation

$$e(x) = f[x_0, x_1, \dots, x_n, x] \phi(x)$$

# Newton : différences divisées

- Calcul pratique des coefficients :  $a_1 = \frac{f[x_1] - f[x_0]}{x_1 - x_0}$

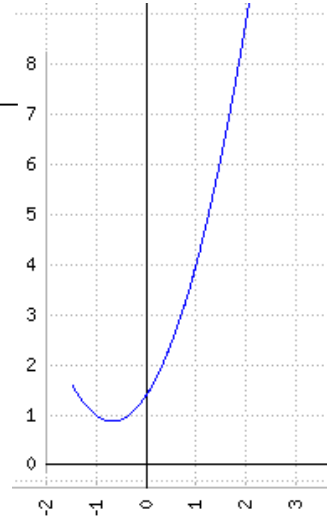
$$a_2 = \frac{f[x_0, x_1] - f[x_1, x_2]}{x_2 - x_0}$$

	$a_0$			
$x_0$	$f[x_0]$	$a_1$		
$x_1$	$f[x_1]$	$f[x_0, x_1]$	$a_2$	
$x_2$	$f[x_2]$	$f[x_1, x_2]$	$f[x_0, x_1, x_2]$	
...	...	...	...	
...	...	...	$f[x_{n-3}, x_{n-2}, x_{n-1}]$	$a_n$
$x_n$	$f[x_n]$	$f[x_{n-1}, x_n]$	$f[x_{n-2}, x_{n-1}, x_n]$	$f[x_0, \dots, x_n]$

## Newton : exemple

- Retour sur l'exercice :  $n=2$  avec  $(-1,1)$ ,  $(1,4)$  et  $(3,16)$

$x_i$	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$
-1	$f[x_0]=1$	$a_1$	$a_2$
1	$f[x_1]=4$	$f[x_0, x_1] = \frac{1-4}{1+1} = \frac{3}{2}$	
3	$f[x_2]=16$	$f[x_0, x_1] = \frac{16-4}{3-1} = 6$	



$$p(x) = 1 + \frac{3}{2}(x+1) + \frac{9}{8}(x+1)(x-1)$$

$$\text{et on retombe sur } p(x) = \frac{9}{8}x^2 + \frac{3}{2}x + \frac{11}{8} \quad (\text{voir Lagrange})$$

## Newton : l'algorithme

Fonction  $a = \text{Newton}(x_i, y_i)$

```
pour  $i = 1$  jusqu'à  $n$ 
     $F(i, 0) \leftarrow y_i(i)$ 
fait
pour  $i = 1$  jusqu'à  $n$ 
    pour  $j = 1$  jusqu'à  $i$ 
        
$$F(i, j) \leftarrow \frac{F(i, j-1) - F(i-1, j-1)}{x_i(i) - x_i(i-j)}$$

    fait
fait
pour  $i = 1$  jusqu'à  $n$ 
     $a(i) \leftarrow F(n, i)$ 
fait
```

Vérifier que la complexité est de :  $n^2$

## Exemple

Soit la fonction  $f(x) = x^4 - 2x^3 - x^2 + x - 1$  définie sur  $\mathbb{R}$ , on souhaite créer un polynôme  $p$  de degré 2 qui interpole  $f$ . Pour ceci nous utilisons les points d'abscisse  $x = \{0, 1, 2\}$ .

Question 1.1.A Calculer le polynôme  $p$  par la méthode d'interpolation de Lagrange.

Dans la méthode d'interpolation de Lagrange le polynôme est de la forme :

$P(x) = p_0(x) + \dots + p_n(x)$  avec  $n + 1$  points de support et

Réponse :

$$p_i(x) = y_i \prod_{\substack{k=0 \\ k \neq i}}^n \frac{x - x_k}{x_i - x_k}$$

Les points de support sont : (0,-1) (1,-2) et (2,-3).

Donc ici nous avons :

$$p_0(x) = y_0 \frac{x - x_1}{x_0 - x_1} \frac{x - x_2}{x_0 - x_2} = -1 \frac{x - 1}{0 - 1} \frac{x - 2}{0 - 2} = -\frac{1}{2}x^2 + \frac{3}{2}x - 1$$

$$p_1(x) = y_1 \frac{x - x_0}{x_1 - x_0} \frac{x - x_2}{x_1 - x_2} = -2 \frac{x - 0}{1 - 0} \frac{x - 2}{1 - 2} = 2x^2 - 4x$$

$$p_2(x) = y_2 \frac{x - x_0}{x_2 - x_0} \frac{x - x_1}{x_2 - x_1} = -3 \frac{x - 0}{2 - 0} \frac{x - 1}{2 - 1} = -\frac{3}{2}x^2 + \frac{3}{2}x$$

Finalement on obtient :

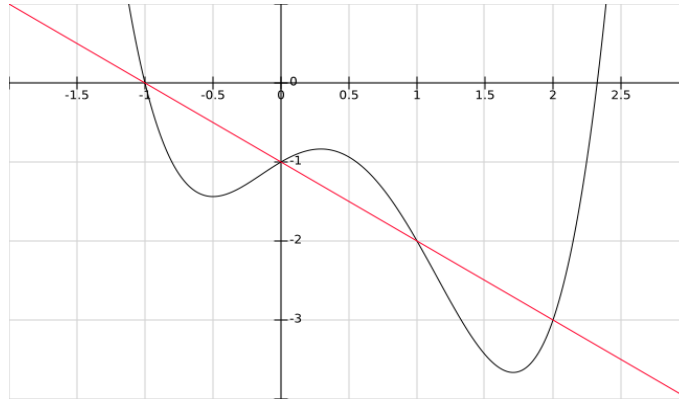
$$P(x) = p_0(x) + p_1(x) + p_2(x) = -x - 1$$

## Exemple (suite)

Question 1.1.B Que constatez-vous ? Ce résultat était-il prévisible dès le départ (justification) ?

Réponse :

Le polynôme se réduit en fait à une droite le résultat est prévisible car les trois points de support sont alignés



Question 1.2 [0.25 point] Calculer et expliquer l'erreur d'interpolation commise en utilisant  $p$  pour le point d'abscisse  $x=1$ .

Réponse :

Pour le point d'abscisse  $x = 1$ , nous avons  $p(1) = -1 - 1 = -2$  qui donne une erreur d'interpolation nulle ce qui est normal, puisque ce point est un point de support.

**Question 1.3 :** Calculer l'erreur d'interpolation commise en utilisant  $p$  pour le point d'abscisse  $x=1/2$ .

**Réponse :** Calculons la valeur de l'interpolation pour  $x = \frac{1}{2}$ . Nous avons  $p\left(\frac{1}{2}\right) = -\frac{1}{2} - 1 = -\frac{3}{2}$   
Alors que nous avons la valeur originale  $f\left(\frac{1}{2}\right) = -\frac{15}{16} = -0.9375$   
L'erreur commise est donc  $e = \left|f\left(\frac{1}{2}\right) - p\left(\frac{1}{2}\right)\right| = \frac{9}{16} = 0.5625$

**Question 1.4 :** Donner l'expression de l'erreur théorique d'interpolation de  $f$  par  $p$  dans l'intervalle  $[0,2]$  (il n'est pas demandé de calculer sa majoration). Et expliquer si l'erreur obtenue en question 1.3 est plus grande ou plus petite que l'erreur théorique maximale.

**Réponse :** Si  $p$  est le polynôme d'interpolation d'une fonction  $f$  pour support les points  $\{x_0, x_1, \dots, x_n\}$ , l'erreur commise en remplaçant la valeur  $f(x)$  par  $p(x)$  est donnée en fonction de  $\xi$  par l'expression suivante :

$$e(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x - x_0) \cdots (x - x_n)$$

Ici, avec nos trois points de support :

$$e(x) = \frac{f^{(3)}(\xi)}{6} (x - x_0)(x - x_1)(x - x_2) = \frac{24\xi - 12}{6} x(x - 1)(x - 2) \quad \text{avec } \xi \in [0,2]$$

L'erreur obtenue à la question 1.3 est nécessairement plus petite que l'erreur théorique maximale, puisque cette erreur est donnée pour un point particulier (i.e.  $-\frac{1}{2}$ ) alors que l'erreur théorique maximale est la pire des erreurs possible pour l'ensemble de l'intervalle.



### Question 1.4 : Donner une majoration de l'erreur théorique.

*Réponse :*

Nous avons obtenu

$$e(x) = \frac{f^{(3)}(\xi)}{6} (x - x_0)(x - x_1)(x - x_2) = \frac{24\xi - 12}{6} x(x - 1)(x - 2) \quad \text{avec } \xi \in [0,2]$$

Il nous faut trouver un majorant de  $e(x)$  pour tout  $x \in [0,2]$ .

$$(x - x_0) \cdots (x - x_n)$$

Comme  $\frac{24\xi - 12}{6}$  est croissante par rapport à  $\xi$ , on prend  $\xi = 2$ . Il reste à étudier

$g(x) = x(x - 1)(x - 2)$  pour en trouver un majorant. On a  $g'(x) = 3x^2 - 6x + 2$  qui s'annule en  $1 + \frac{\sqrt{3}}{3}$  et  $1 - \frac{\sqrt{3}}{3}$  donnant une majoration  $|g(x)| \leq \frac{2\sqrt{3}}{9}$ .

D'où :

$$|e(x)| \leq \frac{24 \times 2 - 12}{6} \times \frac{2\sqrt{3}}{9} = \frac{4\sqrt{3}}{3} \approx 2.31$$

$$|e(x)| \leq 2.31$$

**Question 1.4 :** Donner le polynôme d'interpolation par la méthode de Newton.

**Réponse :**

Dans la méthode d'interpolation de Newton le polynôme est de la forme avec  $(n + 1)$  noeuds

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1) \cdots (x - x_n)$$

En prenant les points de support (ou noeuds) les points : (0,-1) (1,-2) et (2,-3).

Donc ici nous avons :

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$$

Il faut calculer  $a_k$

$x_i$	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$
<b>0</b>	-1 = $a_0$		
<b>1</b>	-2	$\frac{f[x_1] - f[x_0]}{x_1 - x_0} = \frac{-2 + 1}{1 - 0} = -1$ $= a_1$	
<b>2</b>	-3	$\frac{f[x_2] - f[x_1]}{x_2 - x_1} = \frac{-3 + 2}{2 - 1} = -1$	$\frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{-1 + 1}{1 - 0} = 0 = a_2$

Remarque : i le nombre de points est élevé

– **Attention !**

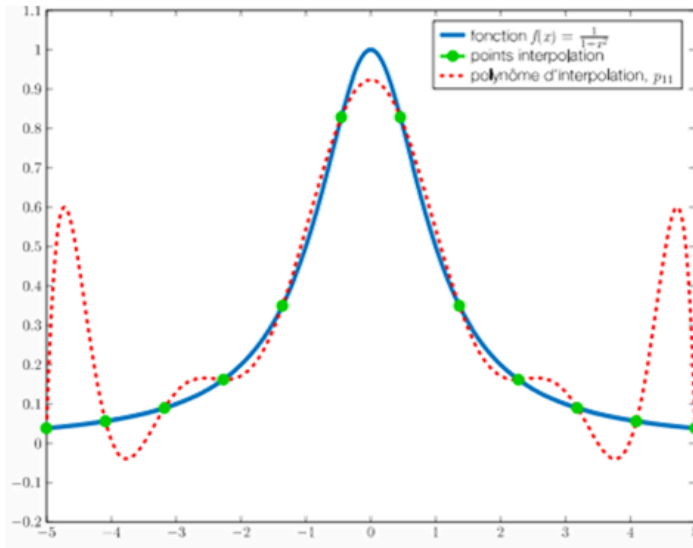
On n'obtient pas de meilleurs résultats en augmentant le degré du polynôme d'interpolation. Supposons que nous cherchions à approcher la fonction  $f(x) = 1/(1 + 25x^2)$  sur l'intervalle  $[-1, 1]$ . Un polynôme d'interpolation de degré trop élevé sur des points équidistants conduit à de fortes oscillations aux bords de l'intervalle (cf. Figure 1.1). Ce phénomène est connu sous le nom de phénomène de Runge. En pratique, on effectue des interpolations avec des polynômes de degrés faibles sur des petits intervalles plutôt que des polynômes de degrés élevés sur de grands intervalles.

# Ch 4 - suite

## Splines cubiques d'Interpolation

## Si le nombre de points est élevé

- entre les points, le polynôme fait ce qu'il veut ; on ne peut pas l'empêcher d'osciller !!!  
et plus son degré est élevé plus il est susceptible d'osciller !
- Le phénomène de Runge Kutta



Phénomène Runge-Kutta : fonction  $f(x) = \frac{1}{1+x^2}$

Interpolation 12 points (noeuds) polynôme de degré 11

Trouver d'autres alternatives !

# Interpolation par splines cubiques

## Principe :

- on approche la courbe par morceaux (localement)
- on prend des polynômes de degré faible (3) pour éviter les oscillations

## Comment

- on décompose l'espace de définition (des points) en un ensemble contigu d'intervalles sur lesquels on applique des interpolations polynômiales de degré 3
- ➔ Résultat un ensemble de polynômes définis de façon continue et « lisse »

# Splines cubiques : définition

## ● Définition :

- On appelle spline cubique (d'interpolation) une fonction notée  $\mathcal{S}$ , qui vérifie les propriétés suivantes :
  - ▶  $\mathcal{S} \in C^2[a, b]$  ( $g$  est deux fois continument dérivable),
  - ▶  $\mathcal{S}$  coïncide sur chaque intervalle  $[x_i, x_{i+1}]$  avec un polynôme de degré inférieur ou égal à 3,
  - ▶  $\mathcal{S}(x_i) = y_i$  pour  $i = 0 \dots n$
  - ▶  $\mathcal{S}'$  est continue sur  $[x_0, x_n]$
  - ▶  $\mathcal{S}''$  est continue sur  $[x_0, x_n]$

# Splines cubiques : définition

- Remarque (voir plus loin):
  - Il faut des conditions supplémentaires pour définir la spline d'interpolation **de façon unique**
  - Ex. de conditions supplémentaires : conditions aux limites
    - ▶  $\mathcal{S}''(a) = \mathcal{S}''(b) = 0$  spline naturelle.
- Remarques :
  - Sur le plan pratique, ces conditions permettent d'avoir une courbe continue et d'aspect lisse ;
  - Forme  $\equiv$  forme d'une barre souple soumise à des contraintes physiques
  - Sur le plan technique, cela permet de poser les équations qui permettent d'obtenir l'expression mathématique de la fonction.



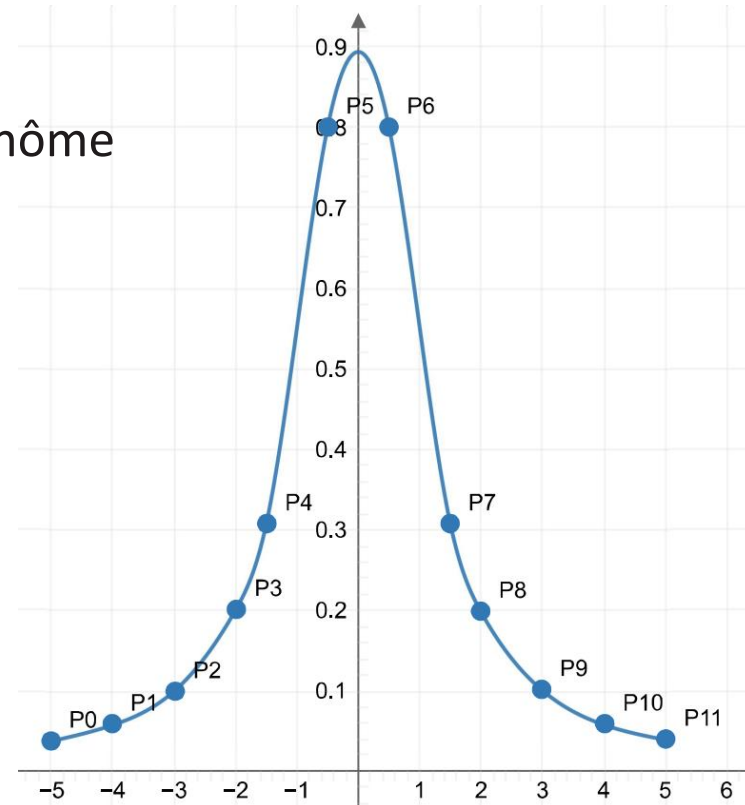
# Splines : illustration

Fonction formée par plusieurs morceaux de polynôme

$$P_1(x) = a_1 x^3 + b_1 x^2 + c_1 x + d_1$$

⋮

$$P_{11}(x) = a_{11} x^3 + b_{11} x^2 + c_{11} x + d_{11}$$



# Splines cubiques : équations

- Déterminer la spline d'interpolation
  - $\mathcal{S}$  coïncide sur chaque intervalle  $[x_i; x_{i+1}]$  avec un polynôme de degré inférieur ou égal à 3
  - $\mathcal{S}''$  est de degré 1 et est déterminé par 2 valeurs:
    - ▶  $m_i = \mathcal{S}''(x_i)$  et  $m_{i+1} = \mathcal{S}''(x_{i+1})$
  - Notations :
    - ▶  $h_i = x_{i+1} - x_i$  pour  $i = 0 \dots n-1$
    - ▶  $\delta_i = [x_i; x_{i+1}]$
    - ▶  $S_i(x)$  le polynôme de degré 3 qui coïncide avec  $\mathcal{S}$  sur l'intervalle  $\delta_i$

# Splines cubiques : équations

Sur  $\delta_i = [x_i; x_{i+1}]$ , le polynôme  $S_i(x)$  vérifie :

$S''_i(x)$  est linéaire : on détermine son équation par la méthode de Lagrange

►  $\forall x \in \delta_i$

$$S''_i(x) = m_{i+1} \frac{x - x_i}{h_i} + m_i \frac{x_{i+1} - x}{h_i}$$

► Pour obtenir  $S'_i(x)$  on intègre :  $S'_i(x) = \int S''_i(x) dx$

$$S'_i(x) = m_{i+1} \frac{(x - x_i)^2}{2h_i} - m_i \frac{(x_{i+1} - x)^2}{2h_i} + a_i$$

( $a_i$  constante d'intégration)

# Splines cubiques : équations

- Pour calculer  $S_i(x)$ , on intègre  $S'_i(x)$

$$S_i(x) = m_{i+1} \frac{(x - x_i)^3}{6h_i} + m_i \frac{(x_{i+1} - x)^3}{6h_i} + a_i(x - x_i) + b_i$$

( $b_i$  constante d'intégration)

- $S_i(x_i) = y_i \longrightarrow y_i = \frac{m_i h_i^2}{6} + b_i$  ①

- $S_i(x_{i+1}) = y_{i+1} \longrightarrow y_{i+1} = \frac{m_{i+1} h_i^2}{6} + a_i h_i + b_i$  ②

# Splines cubiques : équations

- $\mathcal{S}'(x)$  est continue :  $S'_i(x_i) = -m_i \frac{h_i}{2} + a_i = m_i \frac{h_{i-1}}{2} + a_{i-1} = S'_{i-1}(x_i)$

3

- 1 et 2  $a_i = \frac{1}{h_i}(y_{i+1} - y_i) - \frac{h_i}{6}(m_{i+1} - m_i)$

- on remplace les  $a_i$  dans : 3

$$h_{i-1}m_{i-1} + 2(h_i + h_{i-1})m_i + h_im_{i+1} = 6 \left( \frac{1}{h_i}(y_{i+1} - y_i) - \frac{1}{h_{i-1}}(y_i - y_{i-1}) \right)$$

4

## Splines cubiques : équations

- *Récapitulatif :  $n+1$  points  $\rightarrow n$  morceaux de spline  $S_i$*

$$S_0(x_0) = y_0$$

$$S_n(x_n) = y_n$$

- $S_i(x_i) = y_i \quad i = 1, \dots, n-1$

$$S_{i+1}(x_i) = y_i \quad i = 1, \dots, n-1$$

$$S'_i(x_i) = S'_{i+1}(x_i) \quad i = 1, \dots, n-1$$

$$S''_i(x_i) = S''_{i+1}(x_i) \quad i = 1, \dots, n-1$$

$$2 + 2(n-1) + 2(n-1) = 4n - 2 \quad \text{conditions}$$

Or, il faut calculer 4 coefficients pour chaque intervalle .... Il manque 2 conditions

# Splines cubiques : équations

$$2 + 2(n - 1) + 2(n - 1) = 4n - 2$$

- équations grâce aux données
  - ▶ Pour obtenir une solution unique il manque 2 équations :
  - ▶ il faut rajouter 2 conditions → par exemple condition aux limites
    - $S''_0 = S''_n = 0$  (appelée : *spline naturelle*)

Ainsi on complète les

$$2 + 2(n - 1) + 2(n - 1) + 2 = 4n \text{ équations}$$

# Splines cubiques : calcul des coefficients

4

$$h_{i-1}m_{i-1} + 2(h_i + h_{i-1})m_i + h_im_{i+1} = 6 \left( \frac{1}{h_i}(y_{i+1} - y_i) - \frac{1}{h_{i-1}}(y_i - y_{i-1}) \right)$$

- Ex de résolution avec  $h_i = x_{i+1} - x_i$  ( $h_i$  constant) :

▶  $m_{i-1} + 4m_i + m_{i+1} = \frac{1}{h^2}(y_{i-1} - 2y_i + y_{i+1}) = y_i$

- ▶ Forme matricielle

$$S * M = f = \begin{pmatrix} 4 & 1 & & & 0 \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{pmatrix} \begin{pmatrix} m_1 \\ \vdots \\ m_{n-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

- ▶  $S$  matrice inversible, tridiagonale, à diagonale strictement dominante, système facile à résoudre.



# Splines cubiques : algorithme

pour  $i = 2; n - 1$

$$S(i, i) \leftarrow 2(h_i + h_{i-1})$$

$$S(i, i - 1) \leftarrow h_{i-1}$$

$$S(i, i + 1) \leftarrow 2h_i$$

$$f(i - 1) \leftarrow 6 \left( \frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \right)$$

fin pour

$$T \leftarrow S(2:n - 1, 2:n - 1)$$

$$m \leftarrow S/f$$

$$m \leftarrow [0, m, 0]$$

pour  $i = 1; n - 1$

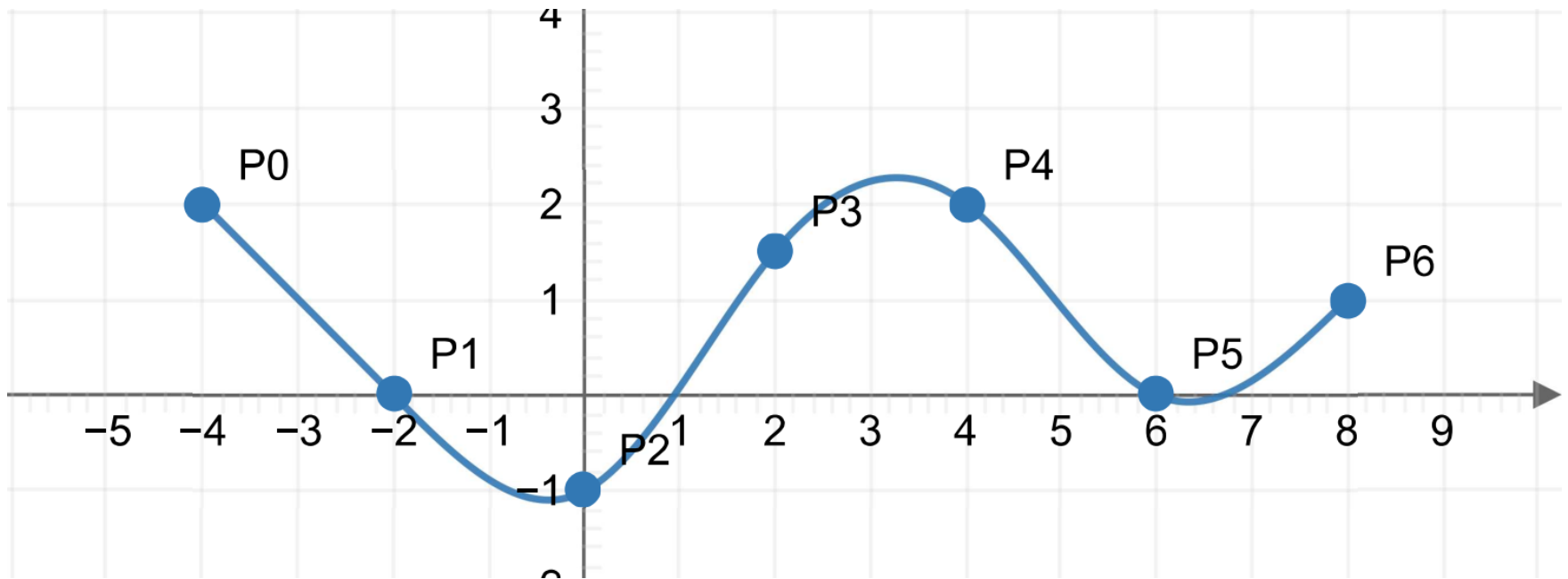
$$a(i) \leftarrow \frac{1}{h_i} (y_{i+1} - y_i) - \frac{h_i}{6} (m_{i+1} - m_i)$$

$$b(i) \leftarrow y(i) - \frac{m_i h_i}{6}$$

fin pour

## Splines cubiques : exemple

- Ex : avec 7 points → spline cubique d'interpolation



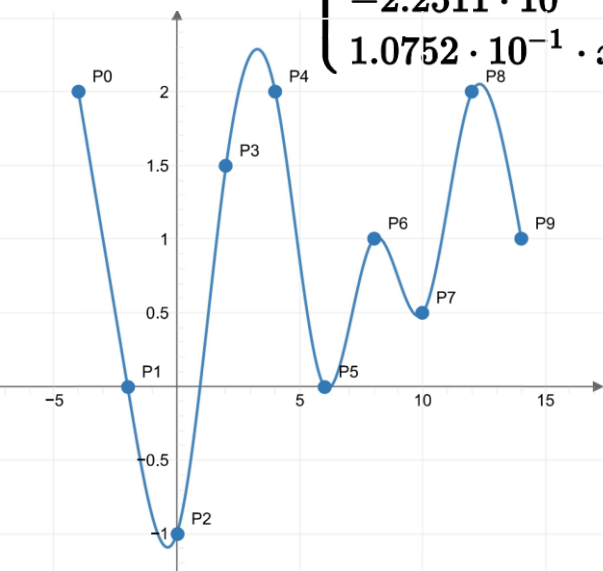
# Splines cubiques : exemple

- Ex : avec 9 points → spline cubique d'interpolation

$P_0(-4|2); P_1(-2|0); P_2(0|-1); P_3(2|1.5); P_4(4|2); P_5(6|0); P_6(8|1); P_7(10|0.5); P_8(12|2); P_9(14|1)$

Equation

$$f(x) = \begin{cases} -4.9636 \cdot 10^{-4} \cdot x^3 + -5.9563 \cdot 10^{-3} \cdot x^2 + -1.0218 \cdot x + -2.0238, & \text{if } x \in [-4, -2], \\ 1.2748 \cdot 10^{-1} \cdot x^3 + 7.6191 \cdot 10^{-1} \cdot x^2 + 5.1390 \cdot 10^{-1} \cdot x + -1.0000, & \text{if } x \in (-2, 0], \\ -1.9693 \cdot 10^{-1} \cdot x^3 + 7.6191 \cdot 10^{-1} \cdot x^2 + 5.1390 \cdot 10^{-1} \cdot x + -1.0000, & \text{if } x \in (0, 2], \\ -2.7258 \cdot 10^{-2} \cdot x^3 + -2.5612 \cdot 10^{-1} \cdot x^2 + 2.5500 \cdot x + -2.3574, & \text{if } x \in (2, 4], \\ 2.4346 \cdot 10^{-1} \cdot x^3 + -3.5048 \cdot x^2 + 1.5545 \cdot 10^1 \cdot x + -1.9684 \cdot 10^1, & \text{if } x \in (4, 6], \\ -2.5910 \cdot 10^{-1} \cdot x^3 + 5.5413 \cdot x^2 + -3.8732 \cdot 10^1 \cdot x + 8.8870 \cdot 10^1, & \text{if } x \in (6, 8], \\ 2.3043 \cdot 10^{-1} \cdot x^3 + -6.2073 \cdot x^2 + 5.5257 \cdot 10^1 \cdot x + -1.6177 \cdot 10^2, & \text{if } x \in (8, 10], \\ -2.2511 \cdot 10^{-1} \cdot x^3 + 7.4589 \cdot x^2 + -8.1405 \cdot 10^1 \cdot x + 2.9377 \cdot 10^2, & \text{if } x \in (10, 12], \\ 1.0752 \cdot 10^{-1} \cdot x^3 + -4.5159 \cdot x^2 + 6.2293 \cdot 10^1 \cdot x + -2.8102 \cdot 10^2, & \text{if } x \in (12, 14]. \end{cases}$$



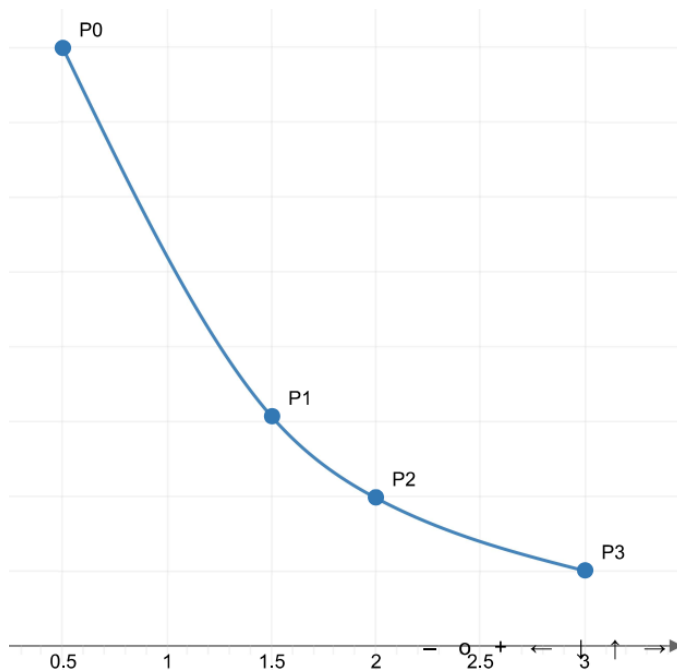
# Exemple

X	0.5	1.5	2.0	3.0
Y	0.7999	0.3107	0.1981	0.1008

- Après résolution du système :

Equation

$$f(x) = \begin{cases} 8.7320 \cdot 10^{-2} \cdot x^3 + -1.3098 \cdot 10^{-1} \cdot x^2 + -5.1463 \cdot 10^{-1} \cdot x + 1.0790, & \text{si } x \in [0.5, 1.5], \\ -1.2328 \cdot 10^{-1} \cdot x^3 + 8.1672 \cdot 10^{-1} \cdot x^2 + -1.9362 \cdot x + 1.7898, & \text{si } x \in (1.5, 2], \\ -2.5680 \cdot 10^{-2} \cdot x^3 + 2.3112 \cdot 10^{-1} \cdot x^2 + -7.6498 \cdot 10^{-1} \cdot x + 1.0090, & \text{si } x \in (2, 3]. \end{cases}$$



# Conclusion

- Interpolation polynomiale
  - évaluer la fonction en un point : Polynôme de Lagrange -> méthode de Neville
  - *compiler* la fonction : Polynôme de Newton
- Interpolation polynomiale par morceau : splines
  - spline cubique d'interpolation : passage par les nœuds (points d'interpolation), mais on limite les oscillations.
  - spline cubique d'approximation : on régule mieux la fonction, mais minimise la distance aux nœuds (les points de passage)

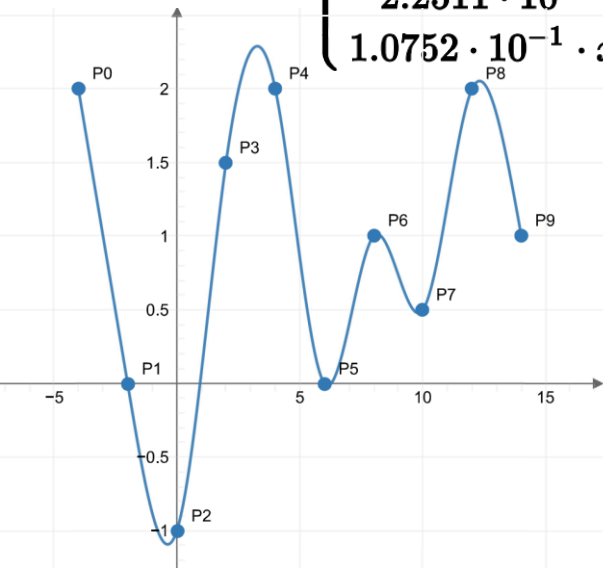
# Splines cubiques : exemple

- Ex : avec 9 points → spline cubique d'interpolation

$P_0(-4|2); P_1(-2|0); P_2(0|-1); P_3(2|1.5); P_4(4|2); P_5(6|0); P_6(8|1); P_7(10|0.5); P_8(12|2); P_9(14|1)$

Equation

$$f(x) = \begin{cases} -4.9636 \cdot 10^{-4} \cdot x^3 + -5.9563 \cdot 10^{-3} \cdot x^2 + -1.0218 \cdot x + -2.0238, & \text{if } x \in [-4, -2], \\ 1.2748 \cdot 10^{-1} \cdot x^3 + 7.6191 \cdot 10^{-1} \cdot x^2 + 5.1390 \cdot 10^{-1} \cdot x + -1.0000, & \text{if } x \in (-2, 0], \\ -1.9693 \cdot 10^{-1} \cdot x^3 + 7.6191 \cdot 10^{-1} \cdot x^2 + 5.1390 \cdot 10^{-1} \cdot x + -1.0000, & \text{if } x \in (0, 2], \\ -2.7258 \cdot 10^{-2} \cdot x^3 + -2.5612 \cdot 10^{-1} \cdot x^2 + 2.5500 \cdot x + -2.3574, & \text{if } x \in (2, 4], \\ 2.4346 \cdot 10^{-1} \cdot x^3 + -3.5048 \cdot x^2 + 1.5545 \cdot 10^1 \cdot x + -1.9684 \cdot 10^1, & \text{if } x \in (4, 6], \\ -2.5910 \cdot 10^{-1} \cdot x^3 + 5.5413 \cdot x^2 + -3.8732 \cdot 10^1 \cdot x + 8.8870 \cdot 10^1, & \text{if } x \in (6, 8], \\ 2.3043 \cdot 10^{-1} \cdot x^3 + -6.2073 \cdot x^2 + 5.5257 \cdot 10^1 \cdot x + -1.6177 \cdot 10^2, & \text{if } x \in (8, 10], \\ -2.2511 \cdot 10^{-1} \cdot x^3 + 7.4589 \cdot x^2 + -8.1405 \cdot 10^1 \cdot x + 2.9377 \cdot 10^2, & \text{if } x \in (10, 12], \\ 1.0752 \cdot 10^{-1} \cdot x^3 + -4.5159 \cdot x^2 + 6.2293 \cdot 10^1 \cdot x + -2.8102 \cdot 10^2, & \text{if } x \in (12, 14]. \end{cases}$$



# Approximation aux moindres carrés

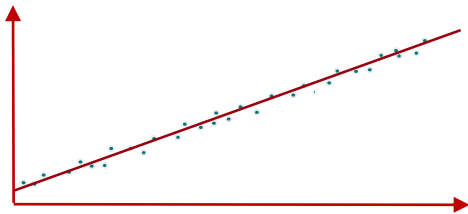
## Ch 4 - suite

# Approximation aux moindres carrés

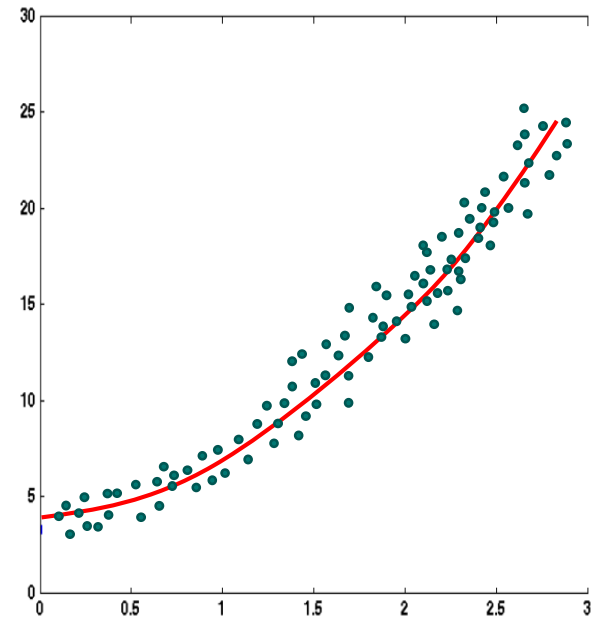
- Exemples

(1) Typiquement, on suppose disposer  
d'un ensemble de mesures  $(x_i, y_i)$

on cherche  $f : f(x, a_0, a_1, \dots, a_n)$



$$g(x, a_0, a_1) = a_0 + a_1x$$





## Approximation aux moindres carrés

- Données : un ensemble de points  $(x_0, y_0), (x_1, y_1), \dots, (x_r, y_r)$ .
- On cherche : une fonction  $f$  dont la courbe approche au mieux tous les points.
  - Le modèle de  $f$  est fixée à l'avance (par exemple un polynôme de degré  $< r$ ).
  - $f(x, a_0, a_1, \dots, a_r) = f(x, A)$  avec  $A = (a_0, a_1, \dots, a_r)^t$
  - But : ajuster les paramètres  $(a_0, a_1, \dots, a_r)$  pour que  $f(x, a_0, a_1, \dots, a_r)$  approche au mieux les données  $(x_i, y_i)$ , (compromis entre toutes les données)
- Comment? minimiser la distance entre  $f$  et l'ensemble des points.

## Cas d'un polynôme

- Rappel : données :  $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_p)$

Si on représente la fonction (théorique) par un polynôme de degré  $n$  ( $n \leq p$ )

$$f(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n = \sum_{k=0}^n a_k x^k$$

**Si  $n = p$**  : on est dans le cas de l'interpolation

( $n+1$ ) équations à ( $n+1$ ) inconnues)

$$f(x_i) = y_i \Leftrightarrow \sum_{k=0}^n a_k x_i^k = y_i \quad i = 1, \dots, p$$

$$f(x_i) = \sum_{k=0}^n a_k x_i^k \quad \text{pour } (i = 0, \dots, p)$$

$n$  le degré du polynôme. Les différentes équations donnent:

$$\left\{ \begin{array}{l} a_0 x_0^0 + a_1 x_0^1 + a_2 x_0^2 + \dots + a_n x_0^n = y_0 \Rightarrow (\text{donné par } (x_0, y_0)) \\ a_0 x_1^0 + a_1 x_1^1 + a_2 x_1^2 + \dots + a_n x_1^n = y_1 \\ \vdots \\ a_0 x_j^0 + a_1 x_j^1 + a_2 x_j^2 + \dots + a_n x_j^n = y_j \\ \vdots \\ a_0 x_p^0 + a_1 x_p^1 + a_2 x_p^2 + \dots + a_n x_p^n = y_p \end{array} \right.$$

$$f(x_i) = y_i \Leftrightarrow \sum_{k=0}^n a_k x_i^k = y_i \quad i = 0, \dots, p$$

Si  $n < p$  : on a  $p$  équations à  $n$  inconnues : plus d'équations que d'inconnues.  
Pas de solution par résolution de système (sauf cas particulier) : incompatibilité.

➔ une approximation on peut écrire :  $f(x_i) = y_i + R_i$  ( où  $R_i$  : résidu ou erreur en  $(x_i, y_i)$  )

➔ **But** : minimiser ces résidus:

$$\min_A \sum_{i=0}^p [R_i]^2 \Leftrightarrow \min_A \sum_{i=0}^p [f(x_i) - y_i]^2$$

On minimise la somme des distances entre les valeurs théoriques  $f(x_i)$  et les  $(p+1)$  données  $y_i$  (les carrés des erreurs)

Ce genre de problème intervient lorsque l'on cherche à modéliser à partir de données, les valeurs  $(x_i ; y_i)$  qui sont souvent des résultats d'expériences ou de mesures, pouvant être entachés d'erreurs.

On note :  $A = \{a_0, \dots a_{n-1}\}$  , on cherche le polynôme :

$$\min_a \sum_{i=0}^p (f(x_i) - y_i)^2 = \min_A \varphi(A)$$

$$\varphi(A) = \sum_{i=0}^p \left( \sum_{k=0}^n (a_k x_i^k - y_i) \right)^2 \rightarrow A^* \text{ (le vecteur solution)}$$

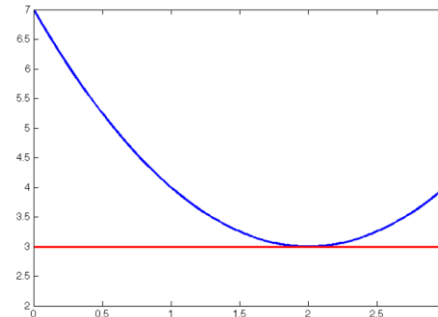
### Rappel

La dérivée permet de connaître les variations d'une fonction, en particulier, si la dérivée est nulle en un point, la tangente est horizontale en ce point et donc la fonction, le minimum (ou le maximum) est atteint au point où les dérivées s'annulent.

On cherche :  $A^* = \underset{A}{\operatorname{Argument}} (\min \varphi(A)) \Leftrightarrow \frac{\partial \varphi}{\partial a_k}(A^*) = 0 \quad k = 0, \dots, n$

**Remarque :** on minimise par rapport aux coefficients  $a_k$ : polynôme de degré 2

$$\frac{\partial \varphi}{\partial a_j} = 2 \sum_{i=0}^n \left( \sum_{k=0}^p (a_k x_i^{k-1} - y_i) \right) x_i^{j-1} = 0$$



Dérivée :

$$\frac{\partial \varphi(a_0, \dots, a_r)}{\partial a_k} = \sum_{i=0}^n [2(y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \dots + a_p x_i^p))(-x_i^k)] = 0$$

**Remarque :** on minimise par rapport aux coefficients  $a_k$

$$\frac{\partial \varphi}{\partial a_j} = 2 \sum_{i=0}^n \left( \sum_{k=0}^p (a_k x_i^{k-1} - y_i) \right) x_i^{j-1} = 0 \quad (*)$$

$$\sum_{k=0}^n a_k \left( \sum_{i=0}^p x_i^{k-1} x_i^{j-1} \right) = \sum_{i=0}^n y_i x_i^{j-1} \quad (**)$$

(\*) et (\*\*) Dérivée d'un polynôme d'ordre 2

On réécrit l'expression de la dérivée en isolant les  $a_j$  (pour  $k = 0, \dots, n$ )

$$\left(\sum_{i=0}^p x_i^k\right) a_0 + \left(\sum_{i=0}^p x_i^{k+1}\right) a_1 + \dots + \left(\sum_{i=0}^p x_i^{k+n}\right) a_p = \left(\sum_{i=0}^p y_i x_i^k\right)$$

On obtient le système linéaire suivant :

$$\underbrace{\begin{pmatrix} \sum x_i^0 & \sum x_i^1 & \sum x_i^2 & \dots & \sum x_i^n \\ \sum x_i^1 & \sum x_i^2 & \sum x_i^3 & \dots & \sum x_i^{n+1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \sum x_i^n & \sum x_i^{n+1} & \sum x_i^{n+2} & \dots & \sum x_i^{2n} \end{pmatrix}}_{\text{Données}} \times \underbrace{\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \\ a_n \end{pmatrix}}_{\text{Paramètres à calculer}} = \underbrace{\begin{pmatrix} \sum y_i \\ \sum y_i x_i \\ \vdots \\ \sum y_i x_i^n \end{pmatrix}}_{\text{Données}}$$



## Cas particulier : régression linéaire

C'est une approximation par un polynôme de degré 1 : modélisation par une droite

$$g(x) = g(x, x_0, x_1) = a_0 + a_1 x$$

Le système devient :

$$\begin{pmatrix} n+1 & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix}$$

Ou encore

$$\begin{pmatrix} n+1 & \sum x_i \\ \bar{x} & \overline{x^2} \end{pmatrix} \times \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \bar{y} \\ \overline{xy} \end{pmatrix}$$

Avec  $\bar{x}$ ,  $\bar{y}$ ,  $\overline{x^2}$ ,  $\overline{xy}$  désigne les moyennes de  $x_i$ ,  $y_i$ , ....

## Cas particulier : régression linéaire (suite)

On obtient alors :

$$a_0 = \frac{\overline{y.x^2} - \bar{x}.\overline{y.x}}{\overline{x^2} - (\bar{x})^2} \quad a_1 = \frac{\overline{x.y} - \bar{x}.\bar{y}}{\overline{x^2} - (\bar{x})^2}$$

On vérifie aisément que la droite passe par le point moyen :

$$\bar{y} = g(\bar{x}, a_0, a_1)$$

## Cas d'un polynôme (suite)

On peut aussi le voir sous forme matricielle on peut écrire :

$X \cdot A = Y$  avec  $A$  vecteur formé des coefficients  $A = \{a_0, \dots, a_{n-1}\}$

$$\begin{bmatrix} x_0^0 & x_0^1 & x_0^2 & \dots & x_0^{n-1} \\ x_1^0 & x_1^1 & x_1^2 & \dots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_j^0 & x_j^1 & x_j^2 & \dots & x_j^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_p^0 & x_p^1 & x_p^2 & \dots & x_p^{n-1} \end{bmatrix} \cdot A = \begin{bmatrix} \mathbf{1} & x_0^1 & x_0^2 & \dots & x_0^{n-1} \\ \mathbf{1} & x_1^1 & x_1^2 & \dots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{1} & x_j^1 & x_j^2 & \dots & x_j^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{1} & x_p^1 & x_p^2 & \dots & x_p^{n-1} \end{bmatrix} \cdot A = Y$$

On a un système Linéaire surdéterminé voir la suite

## Cas d'un polynôme (suite)

Soit à estimer un système linéaires : on a un système surdéterminé : *plus d'équations que d'inconnues* ( $n < P$ )

$$\left\{ \begin{array}{lcl} m_{11}a_1 & + & a_{12}a_2 + \dots + a_{1n}a_n = y_1 \\ & & + r_1 \\ m_{21}a_1 & + & m_{22}a_2 + \dots + m_{2n}a_n = y_2 \\ & & + r_2 \\ : & : & : \\ : & : & : \\ m_{p1}a_1 & + & m_{p2}a_2 + \dots + m_{pn}a_n = y_p \\ & & + r_p \end{array} \right.$$

$a_i$  : inconnues (paramètres du modèle)  $m_{ik}$  : mesures ;

$y_k$  : mesures,

$r_k$  : erreurs de mesures

n données m équations avec  $m > n$

## Cas général d'un système linéaire

*En notation Matricielle  $Ax = Y + R$  avec  $R$  vecteur résidu ou erreur*

$$\begin{bmatrix} m_{11} & m_{12} & \dots & m_{1n} \\ m_{21} & m_{22} & \dots & m_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ m_{m1} & m_{m2} & \dots & m_{pn} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} + \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_p \end{bmatrix}$$

*La méthode des M.C.*

*Déterminer  $a_k$  tq la somme des carrés de tous les résidus soit minimale.*

## Cas général d'un système linéaire

*En notation Matricielle  $Ax = Y+R$*

$$M \cdot A = Y + R \Leftrightarrow M \cdot A - Y = R$$

$$S = R^t R = (M \cdot A - Y)^t (M \cdot A - Y) \quad (*)$$

$$\text{Avec } A = [a_0, \dots, a_n]^t$$

*La minimisation de  $S$  par rapport à  $a_k$  avec  $k = 1, \dots, n$  entraîne les conditions nécessaires fixées avant c'est-à-dire :  $\frac{\partial S}{\partial a_k} = 0$*

## Cas général d'un système linéaire

*En revenant à la notation matricielle (\*), les conditions s'écrivent*

*(voir explication donnée en cours) :*

$$(M^t \cdot M)\tilde{A} = M^t y \quad (**)$$

*avec  $\tilde{A}$  les valeurs de  $A$  minimisant (\*)*

*On arrive à un système cohérent  $n$  équation à  $n$  inconnues. On résout le système en faisant appel aux méthodes de résolution d'un système linéaire (triangulation, QR, LU, .... )*

*Par exemple la solution des moindres carrés  $\tilde{A}$  est donnée :*

$$\tilde{A} = (M^t M)^{-1} M^t Y$$

# Régression exponentielle

- L'exemple le plus connu est la modélisation de la radioactivité d'un déchet nucléaire ou la modélisation de l'évolution de la population !

$$g(t, a, b) = be^{-at}$$

On a à résoudre un système non-linéaire avec des exponentielles.  
Cette résolution peut-être réalisée,  
soit en adaptant une méthode de résolution de système non linéaire dans un cadre multidimensionnel, ou en utilisant que

$$\log_b xy = \log_b x + \log_b y$$

$$\log_a \frac{x}{y} = \log_a x - \log_a y$$

$$\log_b x^p = p \log_b x$$



## Régression exponentielle

$$g(t, a, b) = be^{-at}$$

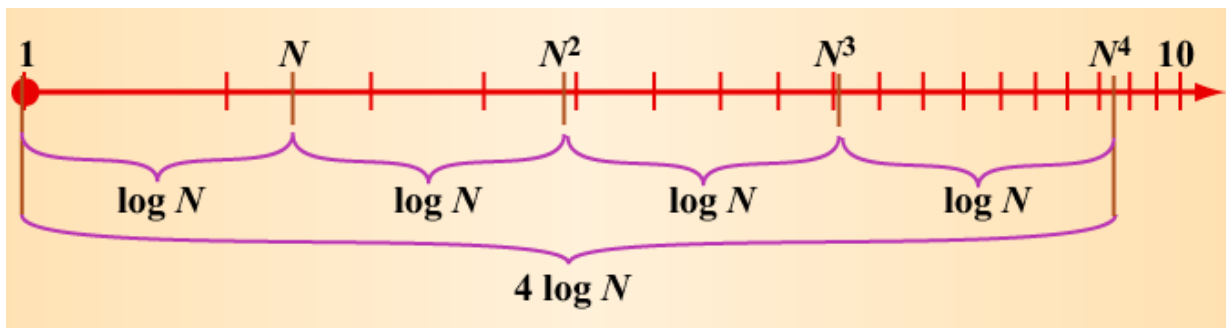
$$\log [g(t, a, b)] = \log(be^{-at}) = \log(b) + \log(e^{-at}) = \log(b) + (-at)$$

$$Y = \log(g(t, a, b)) = c + mt$$

$$\text{Avec } c = \log(b) \text{ et } m = -a$$

et donc, trouver une régression exponentielle pour les points  $(x_i, y_i)$   
Ou bien une régression linéaire pour les points  $(x_i, \log y_i)$ . Les constantes sont alors reliées par

$$b = \log a_0 \text{ et } a = -a_1.$$



## Lien entre variables

### Fonction logarithmique

Une fonction logarithmique est de la forme :

$$y = a \log x + b \text{ (ou } y = a \ln x + b \text{)}$$

On revient à un système linéaire **en posant**  $X = \log x$

On voit directement qu'il doit y avoir une relation affine entre  $y$  et  $\log x$  que l'on peut écrire :

$$y = AX + B, \quad (\text{avec } X = \log x)$$

*A et B sont des coefficients réels.*

**Réciproquement** : on remarque une telle relation sur un repère « semi-log » en représentant la variable  $x$  sur l'échelle logarithmique. Si le nuage forme une droite, le modèle est logarithmique.

