Algorithmique Numérique

saida.bouakaz@univ-lyon1.fr

Plan du Cours

- **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 linaires 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

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

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

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

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

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

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

Cas générique : à l'étape k

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

$$A^{(k-1)} = \begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \cdots & a_{1k-1}^{(0)} & a_{1k}^{(0)} & \cdots & a_{1n}^{(0)} \\ 0 & a_{22}^{(1)} & \cdots & a_{2k-1}^{(1)} & a_{2k}^{(1)} & \cdots & a_{2n}^{(1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a_{k-1}^{(k-2)} & a_{k-1\,k}^{(k-2)} & \cdots & a_{k-1\,n}^{(k-2)} \\ 0 & 0 & \cdots & 0 & a_{k\,k}^{(k-1)} & \cdots & a_{k\,n}^{(k-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & 0 & a_{n\,k}^{(k-1)} & \cdots & a_{n\,n}^{(k-1)} \end{bmatrix} \qquad B^{(k-1)} = \begin{bmatrix} b_1^{(0)} \\ b_2^{(1)} \\ \vdots \\ b_{k-1}^{(k-2)} \\ b_k^{(k-1)} \\ \vdots \\ b_n^{(k-1)} \end{bmatrix}$$

Expression générale

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$$

$$b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)}$$

avec $\begin{cases} i = k+1, \dots, n \\ j = k+1, \dots, n \end{cases}$

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

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 → $[A. A^{-1} \mid A^{-1} .I]$

- 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 → (L . U) X= B

A=L .U → L .(U X)= B 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

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

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

$$\text{pour} \quad i = k+1,\ldots,n \quad \begin{cases} a_{ij}^{(k+1)} \leftarrow a_{ij}^{(k)} - \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}} a_{kj}^{(k)} & \text{pour } j = k+1,\ldots,n \\ b_i^{(k+1)} \leftarrow b_i^{(k)} - \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}} b_k^{(k)} \end{cases}$$

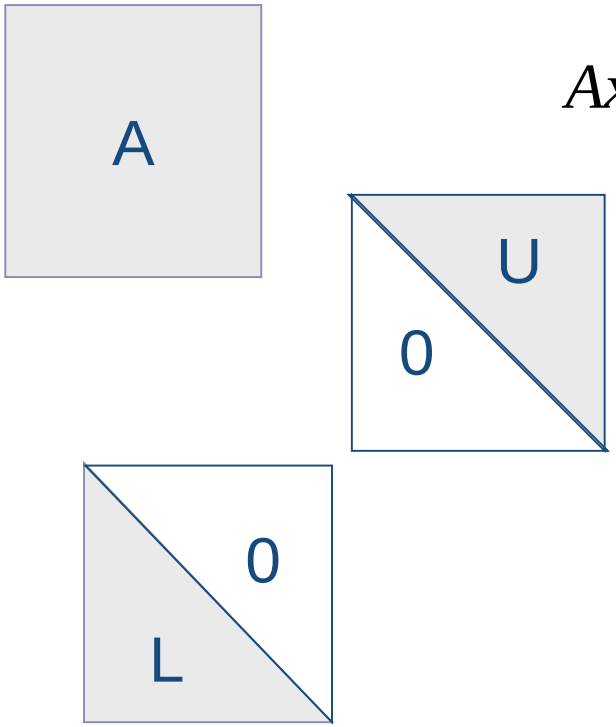
$$\text{notation matricielle} : A^{(k+1)} = M^{(k)} A^{(k)}; \quad b^{(k+1)} = M^{(k)} b^{(k)};$$

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 Let 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 $M^{(k)}$

$$A^{(k+1)} = M^{(k)} A^{(k)} \qquad A^{(1)} = A \quad \text{et} \quad A^{(n)} = U$$

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

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

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

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

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

donc $L = M^{-1}$

LU : récapitulatif

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

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

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

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

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

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

C'est la matrice l_{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

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

Pivot =1
lig2 ←lig2 -(-1)lig1
lig3 ←lig3 -(1)lig1
lig4 ←lig4 -(1)lig1

$$\begin{pmatrix} 1 & 1 & -1 & 2 \\ 0 & \textcircled{3} & 0 & 3 \\ 0 & -1 & 2 & -3 \\ 0 & -2 & 1 & 0 \end{pmatrix}$$

Pivot =3
lig3 ←lig3 -(-1/3)lig2
lig4 ←lig4 -(-2/3)lig2

Etape 2

$$\begin{pmatrix} 1 & 1 & -1 & 2 \\ 0 & 3 & 0 & 3 \\ 0 & 0 & \textcircled{2} & -2 \\ 0 & 0 & 1 & 2 \end{pmatrix}$$

Pivot =2
lig4 ←lig4 -(1/2)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 = 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

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

      pour $j = k+1$ jusqu'à n

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

      fait

    fait

  sinon "problème"

fait

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

Pour calculer L et U, il suffit de remarquer que

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

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

En prenant les équations obtenues dans le bons ordres (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.

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

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

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} ← a_{i,j}
      pour k de 1 à i − 1 faire
        r_{i,j} ← 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} ← a_{i,j}
      pour k de 1 à j − 1 faire
        l_{i,j} ← l_{i,j} − l_{i,k} r_{k,j}
      fin pour
         l_{i,j} ← 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 . X = B***
- Principe du calcul d'un point fixe : limite de la suite construite.
- Procédé → transformer ***A . X = B*** ⟷ en une égalité

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

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

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

On définie la suite récurrente par :

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

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

- Si la suite converge (k vers + ∞), 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

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

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

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

Sous cette forme $AX = (-E + \boldsymbol{D} - F)X$

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

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 :

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

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

$$\vdots$$

$$x_n^1 = \frac{1}{a_{nn}}(b_n - a_{n1}x_1^0 - a_{n2}x_2^0 - \cdots - a_{nn-1}x_{n-1}^0)$$

Méthode de Gauss-Seidel

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

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

Le calcul effectif peut se faire par un calcul matriciel
En calculant : $(D - E)^{-1}$:
$M = (D - E)^{-1}F$ 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 sou 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 :

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

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

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

$$x_n^1 = \frac{1}{a_{nn}}(b_n - a_{n1}x_1^0 - a_{n2}x_2^0 - \cdots - a_{nn-1}x_{n-1}^0)$$

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 λi 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 :

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

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 $(D-E)^{-1}$

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

1ère 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

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

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

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

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

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

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

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

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

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

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

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

Résolution par expressions équationnelles

Le calcul effectif se fait de la façon suivante

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

Soit :

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

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

Résolution par expressions équationnelles -suite

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

1ère itération

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

2ème itération

$$x_1^{(2)} = \frac{1}{2}\left(6 - x_2^{(1)} - 0x_3^{(1)}\right) = \frac{1}{2}\left(6 - 6 - (-\frac{1}{2})\right) = \frac{1}{4} = 0.25$$
$$x_2^{(2)} = \frac{1}{1}\left(3 + x_1^{(2)} - 2x_3^{(1)}\right) = \left(3 + \frac{1}{4} - 2 \times (-\frac{1}{2})\right) = \frac{17}{4} = 4.25$$
$$x_3^{(3)} = \frac{1}{2}\left(2 - x_1^{(2)} - 0x_2^{(2)}\right) = \frac{1}{2}\left(2 - \frac{1}{4} - 0 \times \frac{17}{4}\right) = \frac{7}{8} = 0.875$$

3ère itération

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

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

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

4ème itération

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

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

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

Retour aux méthodes : méthode de Jordan

- Méthode : basée sur une diagonalisation
- Utilise une suite de combinaison linaires entre les différentes lignes, travaille sur la matrice élargie (voir méthode de gauss)
- Utilisation particulière de Gauss

- $AX=B \rightarrow LX=B^{(k)}$ 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 : $R^n \rightarrow R^p$.

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

?

$f : R \quad R.$

Théorème (zéro d'une fonction)
Soit f une fonction continue
$f: [a, b] \rightarrow R$
$si\ f(a)f(b) \leq 0, \qquad alors$

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

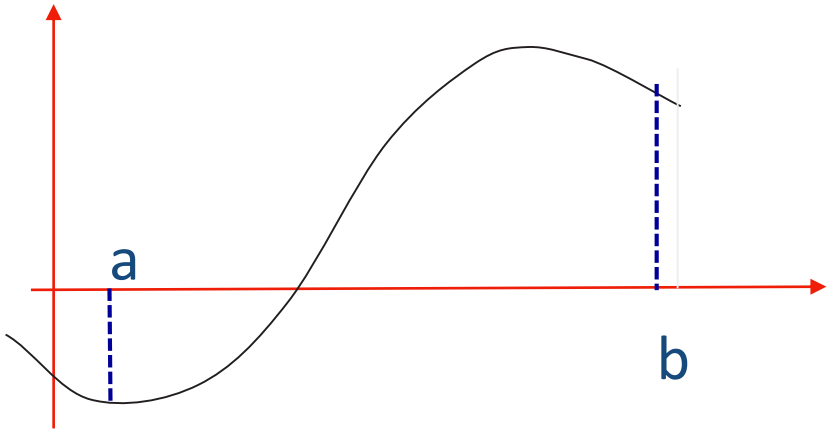


Schéma général de l'approche pour la résolution

$f : R \rightarrow R$.

On transforme la forme de l'équation:

$f(x) = 0 \Leftrightarrow \varphi(x) = x$ on construit la suite :

$X^{k+1} = \varphi(X^k)$ et $\lim_{k \to \infty} X^k = x$

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

La solution est déterminée avec une précision ε 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

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

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

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 : $\left|X^{(k)} - X^{(k-1)}\right| \leq \epsilon$
    - ✓ $|f(X^{(k)})| \leq \epsilon$

❖ 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 → 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 $f(x_1)*f(x_2) \leq 0$
  - 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] → R, continue sur [a, b], dérivable sur]a, b[, il existe c ∈]a, b[tel que

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

- Cqce du Th. des accroissement finis : φ 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 φ n'est définie que sur un domaine D, il faut choisir x(0) dans D et vérifier que $\varphi(D) \subset D$.

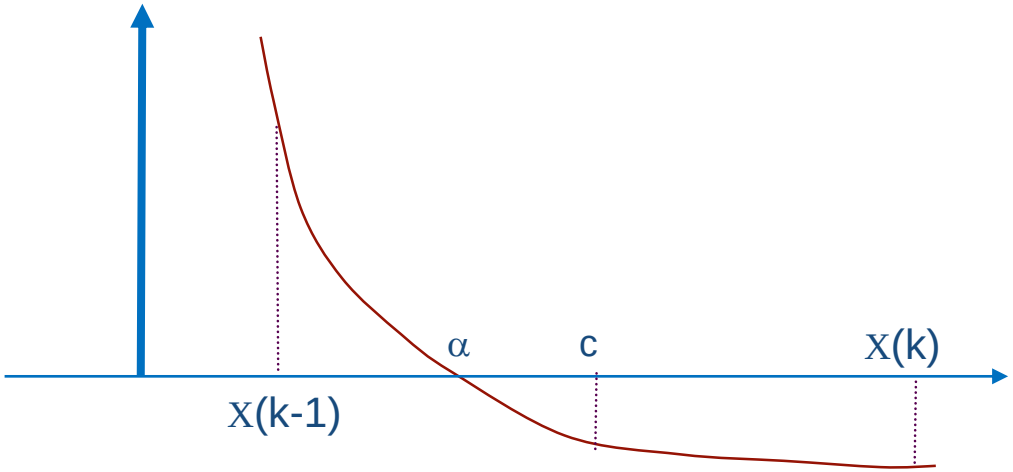
- *Ordre de convergence : Soit $x^{(*)}$, un point fixe de φ*
  - si pour tout $x^{(k)}$ dans le voisinage de x^*, on a la relation :

$$|x^{(k+1)} - x^*| \leq C \cdot |x^{(k+1)} - x^*|^p$$

  pour tout $k \geq 0$, avec $C < 1$ si $p \geq 1$; on dit que φ est d'ordre au moins **p** pour déterminer $x^{(*)}$.

- p = 1 : convergence linéaire
- p = 2 : convergence quadratique

Méthode de la bissection (dichotomie)



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

La règle de production

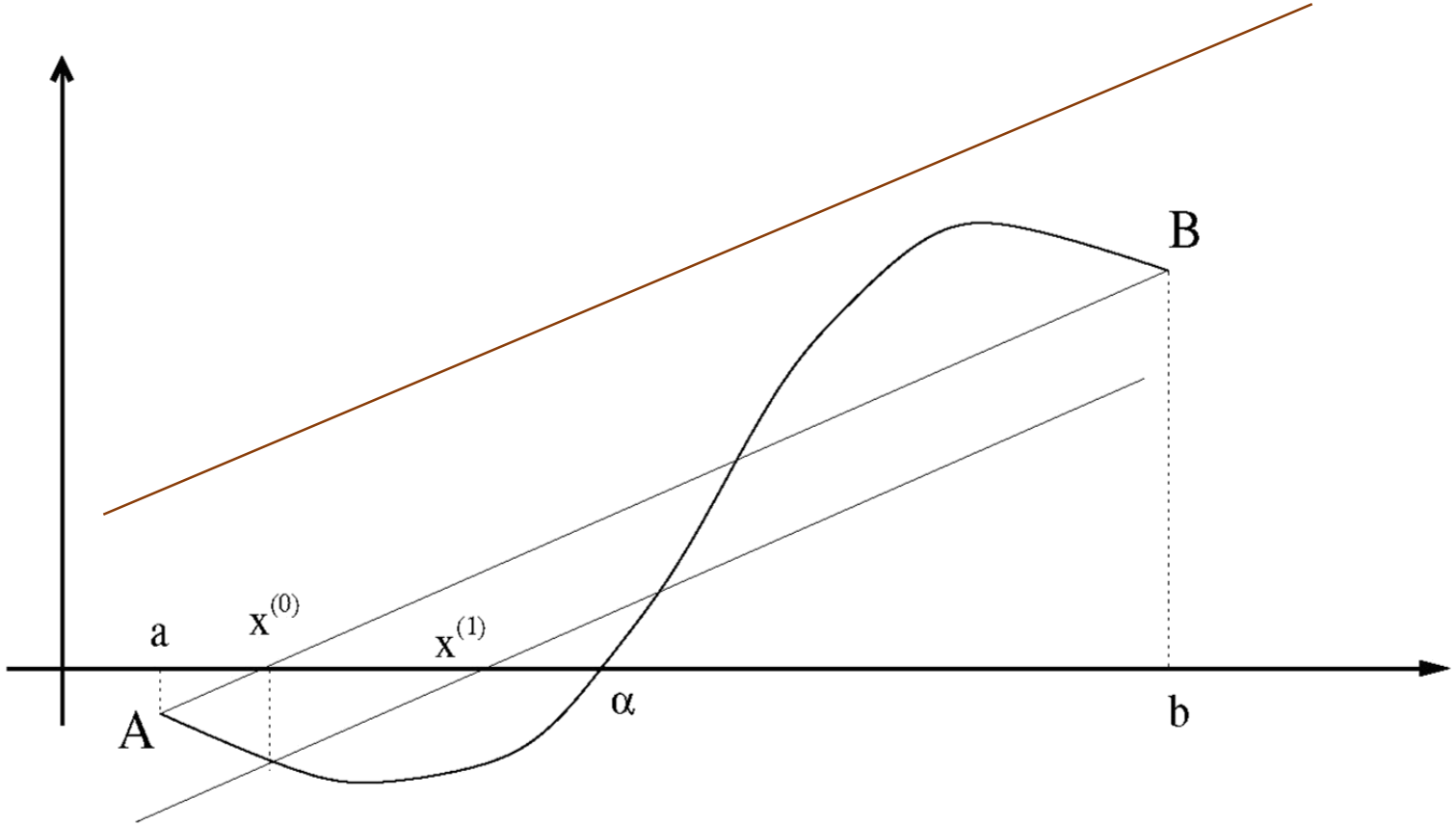
Algorithme : méthode de dichotomie

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

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

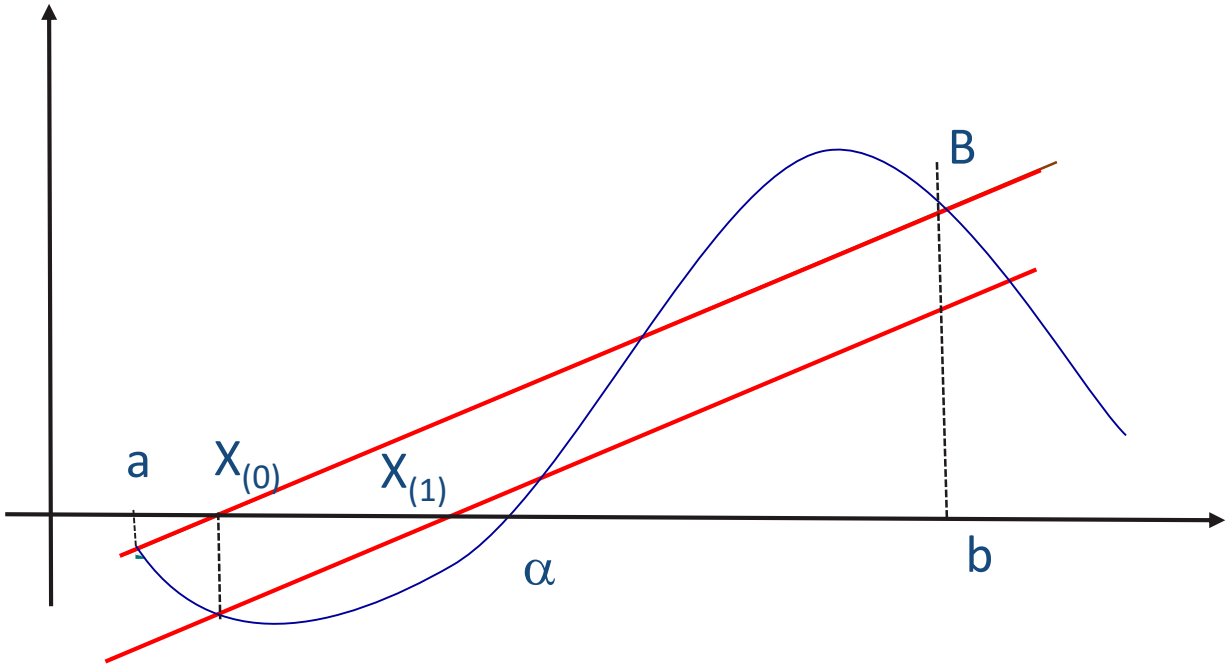
- Si $f(x^{(k)}) = 0$ alors $x^{(k)}$ est la racine α.
- Si $f(x^{(k)})f(a^{(k)}) < 0$
  - $a^{(k+1)} = a^{(k)}$, $b^{(k+1)} = x^{(k)}$
- Si $f(x^{(k)})f(b^{(k)}) < 0$
  - $a^{(k+1)} = x^{(k)}$, $b^{(k+1)} = b^{(k)}$
- $x^{(k+1)} = \frac{a^{(k)}+b^{(k)}}{2}$

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

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

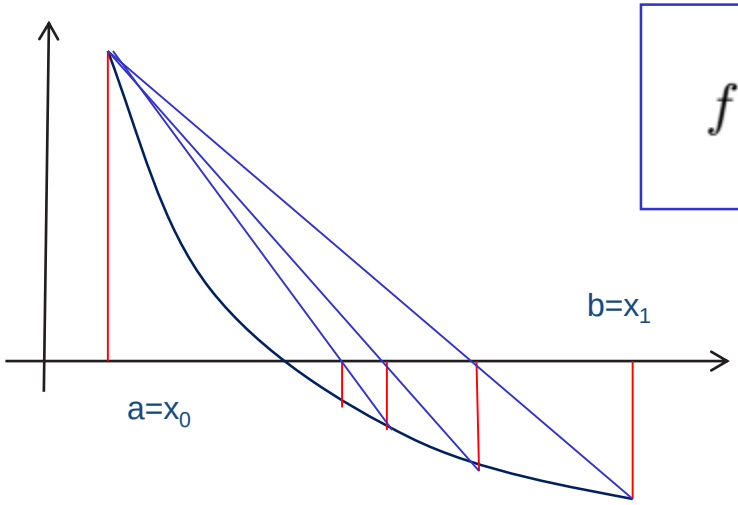
la condition de convergence locale $|\phi'(\alpha)| < 1$ est équivalente à

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

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. $(f(a).f(b) < 0\ ;\quad f\ continue$

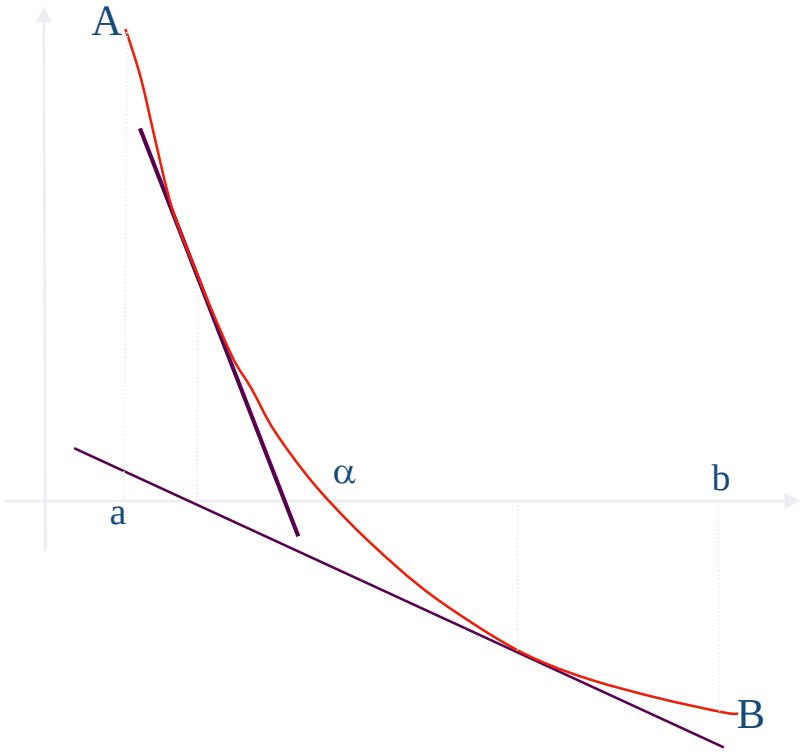


$$f(x_n) + \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}(x_{n+1} - x_n) = 0$$

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 α 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 :

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

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

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

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

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

Chapitre 4

Interpolation et approximation

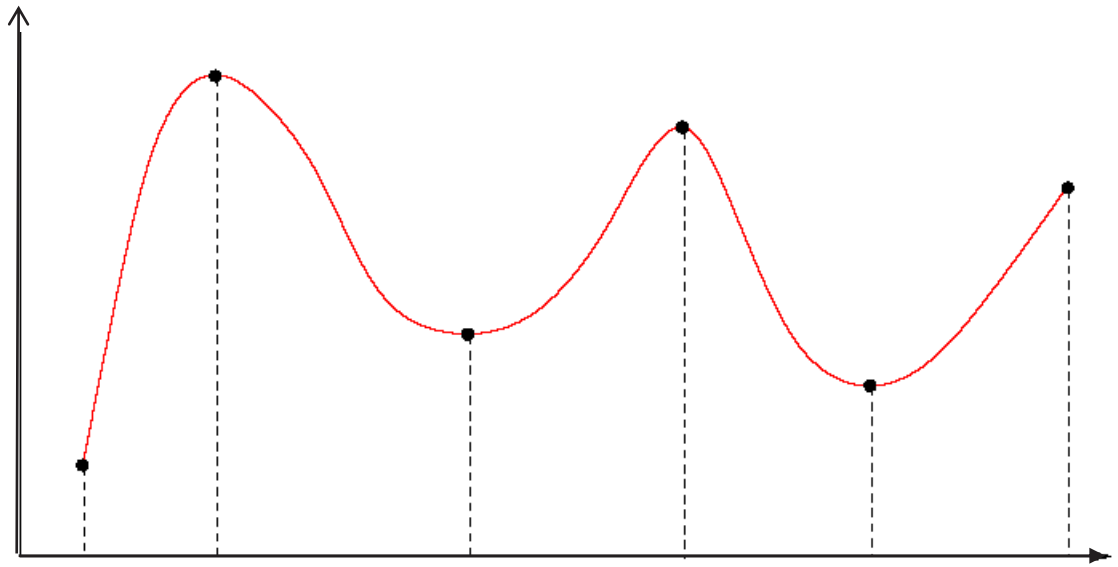
Problème

- Données :
  - un ensemble de points connus (x_i, Y_i) ; ou $Y_i \in R^p$
    - 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: R → R inconnue et continue sur un intervalle [a, b].
- un ensemble de points connus (x_i, y_i), $i \in [0, n]$.
  - $\{x_0, x_1, \ldots, x_n\}$ *est le support de l'interpolation*

On cherche

une fonction φ : R → R telle que

$\varphi(x_i) = f(x_i)$, $i \in [0, n]$.

- En pratique, φ est une somme de fonctions

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

vérifiant

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

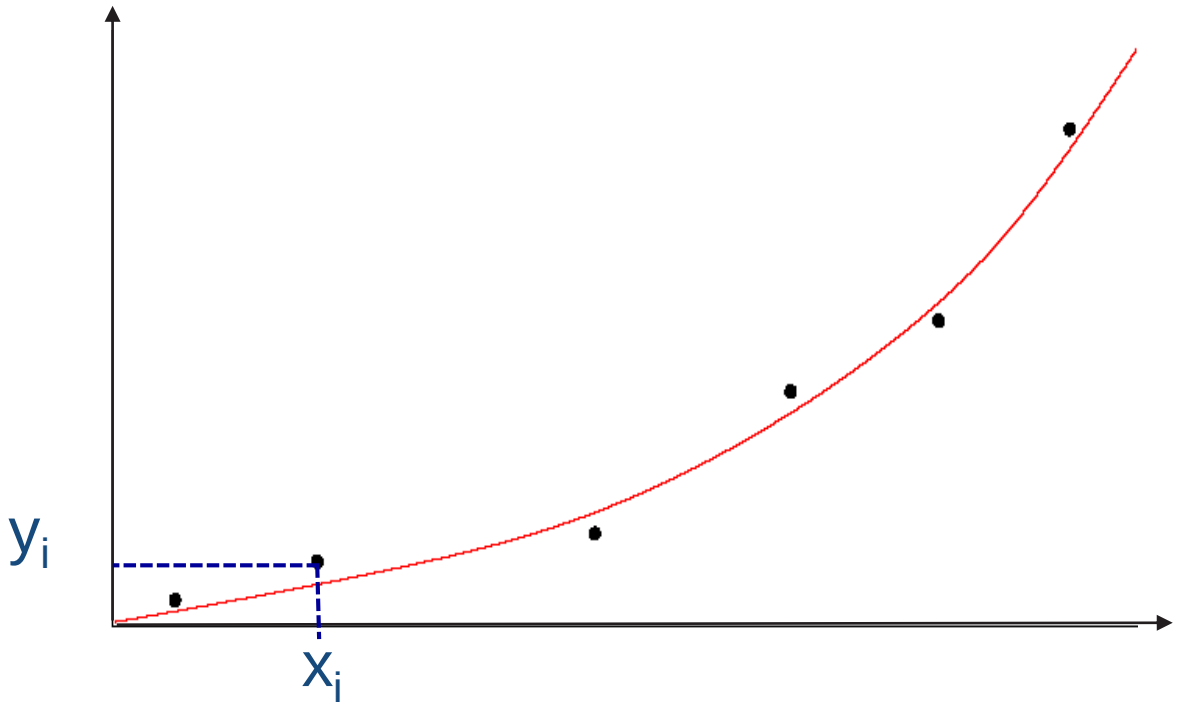
φ_j : fonctions de la base dans laquelle on exprime f ; φ_i doit se prêter aux traitements numériques courants.

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

Approximation

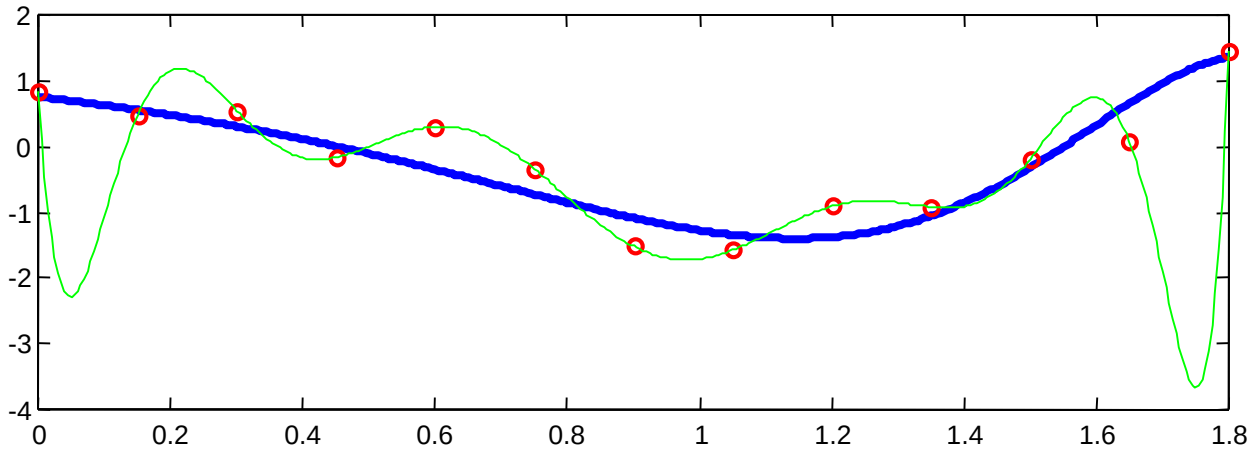
les (x_i, y_i) sont des mesures données

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 :

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

plus ε, 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 --> $(x_0, y_0 = f(x_0)), \ldots, (x_i, y_i = f(x_i)), \ldots, (x_i, y_i = f(x_i))$
- 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

Interpolation polynomiale

la combinaison linéaire de polynômes est un polynôme

$$(x_0, y_0 = f(x_0)), \ldots, (x_i, y_i = f(x_i)), \ldots, (x_i, y_i = f(x_i))$$

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

→ Idée de Lagrange

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

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

ainsi $P(x_i) = y_0 P_0(x_i) + \ldots + y_i P_i(x_i) + y_n P_n(x_i)$

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$0 \qquad 1 \qquad 0$$

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

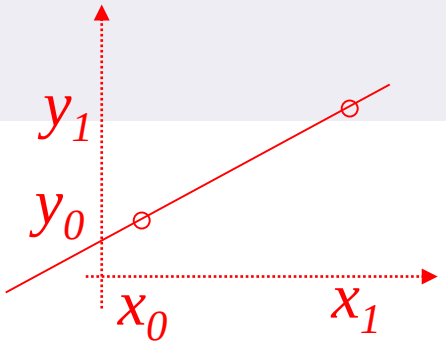
- Soient *n+1* points distincts x_i réels et *n+1* réels y_i, il existe un unique polynôme $p \in P_n$ tel que $p(x_i) = y_i$ pour $i = 0$ à n

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**
  - $L_i(x_i) = 1$
  - $L_i(x_i) = 0 \quad (j \neq i)$

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

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

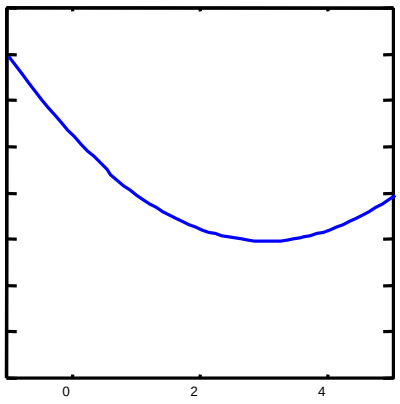
$$y = y_0\frac{x - x_1}{x_0 - x_1} - y_1\frac{x - x_0}{x_0 - x_1} = y_0\underbrace{\frac{x - x_1}{x_0 - x_1}}_{L_0(x)} + y_1\underbrace{\frac{x - x_0}{x_1 - x_0}}_{L_1(x)}$$

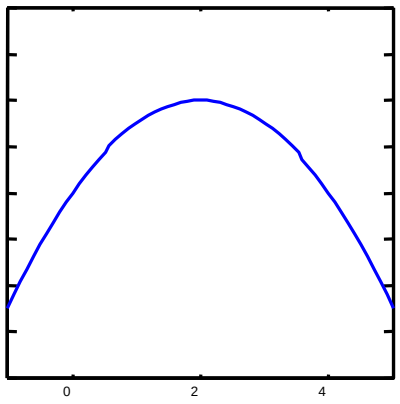
Lagrange : degré 2

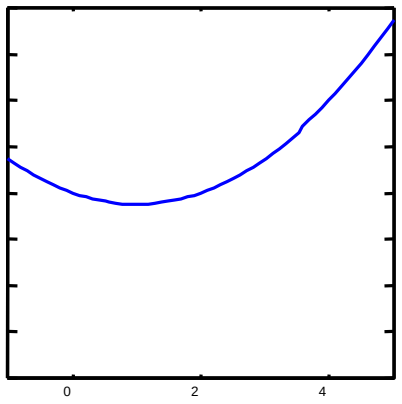
- Exemple avec *n=2*
  - on connaît 3 points (*0,1*), (*2,5*) et (*4,17*)
  - polynômes de Lagrange associés :

    → Espace vectoriel : avec $\{L_i\}$ base de l'interpolation

$$L_0(x) = \frac{(x-2)(x-4)}{8} \qquad L_1(x) = \frac{x(x-4)}{-4} \qquad L_2(x) = \frac{x(x-2)}{8}$$







Lagrange : degré 2

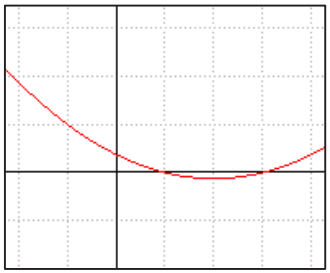
- Exemple avec *n=2*
  - on connaît 3 points (*-1,1*), (*1,4*) et (*3,16*)
  - polynômes de Lagrange associés :

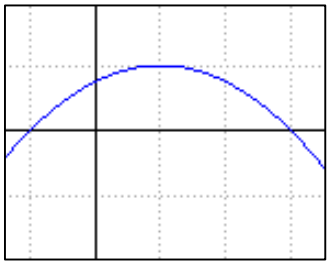
    → Espace vectoriel : avec $\{L_i\}$ base de l'interpolation

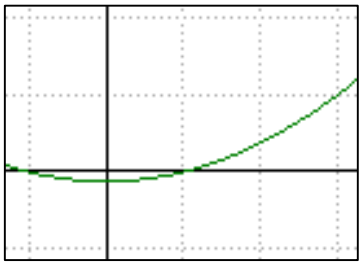
$$L_0(x) = \frac{(x-1)(x-3)}{8}$$

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

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







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 $\mathrm{p(x)} = \frac{9}{8}x^2 + \frac{3}{2}x + \frac{11}{8}$

Lagrange : Algorithme

Fonction y = Lagrange (x, x_i, y_i)

pour $i = 1$ à n

  pour $j = 1$ à $n, j \neq i$

    $l \leftarrow l * \dfrac{x - x_i(j)}{x_i(i) - x_i(j)}$

  fin pour

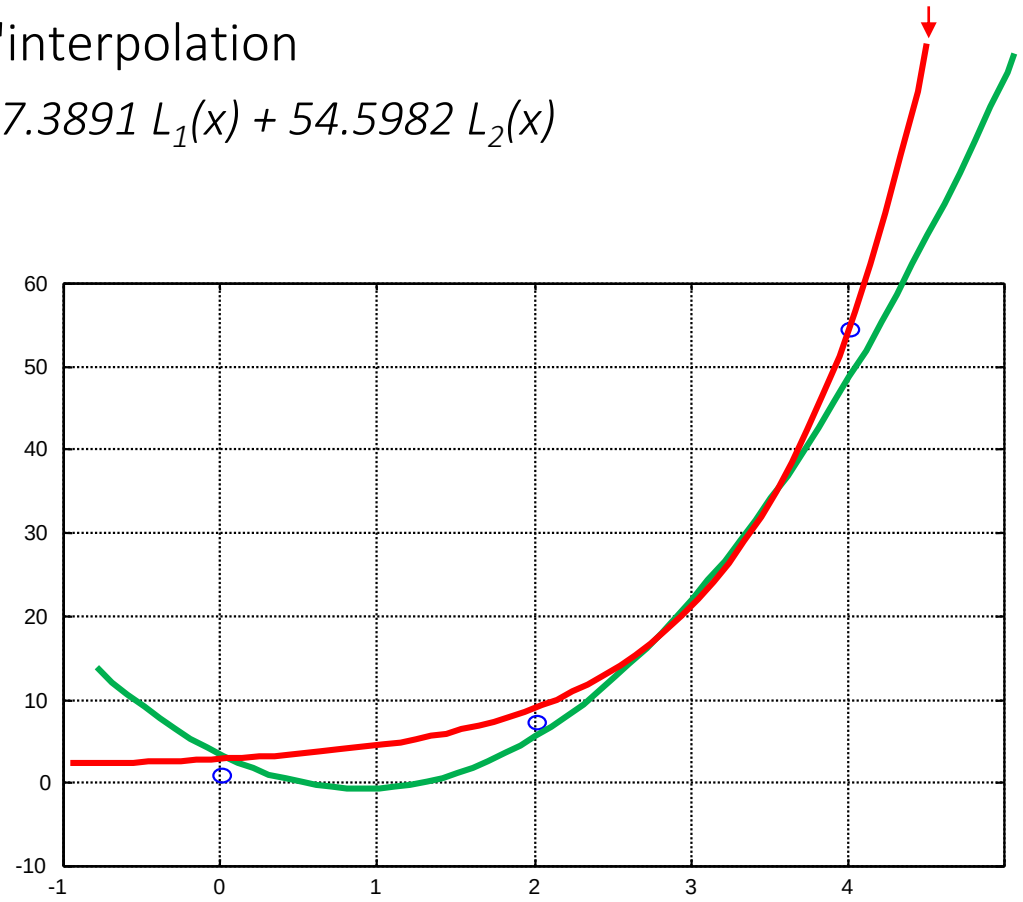
  $y \leftarrow y + y_i * l$

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
  - $\phi(x)=(x-x_0)(x-x_1)\ldots(x-x_n)$
- alors, il existe $\xi \in I$ tel que

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

- NB : ξ 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

- Supposons que l'on possède un nb élevé de points pour approcher f … faut-il tous les utiliser ?
  - (calculs lourds)

- Méthode de Neville :
  - on augmente progressivement n
  - on calcule des L_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.

$$p_{i,0}(x) = y_i, \qquad 0 \leq i \leq n,$$

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

Algorithme de Neville-Aitken

- Application

$$
\begin{array}{lllll}
p_{0,0}(x) = y_0 & & & & \\
 & p_{0,1}(x) & & & \\
p_{1,0}(x) = y_1 & & p_{0,2}(x) & & \\
 & p_{1,1}(x) & & p_{0,3}(x) & \\
p_{2,0}(x) = y_2 & & p_{1,2}(x) & & \boxed{p_{0,4}(x)} \\
 & p_{2,1}(x) & & p_{1,3}(x) & \\
p_{3,0}(x) = y_3 & & p_{2,2}(x) & & \\
 & p_{3,1}(x) & & & \\
p_{4,0}(x) = y_4 & & & &
\end{array}
$$

x_i	1	2	3	5
y_i	1	4	2	5

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

L'algorithme de Neville

Fonction y = 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

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

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 = $\{1, (x-x_0), (x-x_0)(x-x_1), \ldots, (x-x_0)(x-x_1)\ldots(x-x_{n-1})\}$
  - on peut ré-écrire $p(x)$:

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

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

- ✓ $k=0 \quad f[a] = f(a)$
- ✓ $k=1 \quad f[a,b] = (f[b] - f[a]) / (b - a)$
- ✓ $k=2 \quad f[a,b,c] = (f[a,c] - f[a,b]) / (c - b)$

…

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

$a \notin X$, $b \notin X$, $a \neq b$

Newton : différences divisées

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

Théorèmes

calcul des coeficients de newton

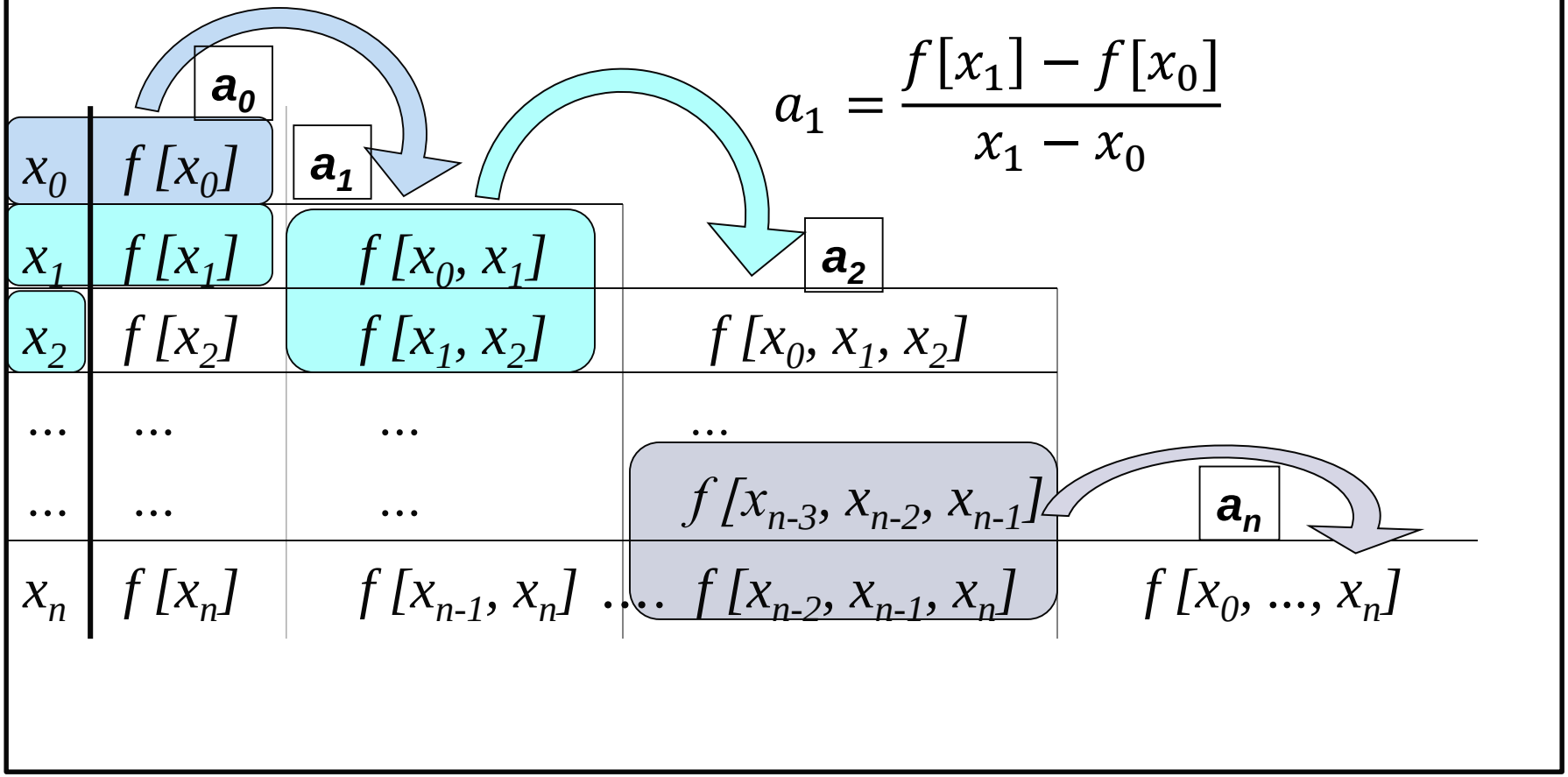
$$a_k = f[x_0, x_1, \ldots, x_k] \qquad \text{avec } k = 0 \ldots n$$

Calcul de l'erreur d'interpolation

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

Newton : différences divisées

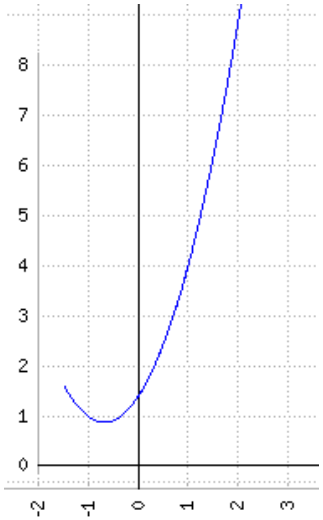
- Calcul pratique des coefficients :



$$a_1 = \frac{f[x_1] - f[x_0]}{x_1 - x_0}$$

Newton : exemple

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



		a_0		a_1		a_2
0	$f[x_0]=1$					
2	$f[x_1]=5$	$f[x_0, x_1]$ $=(1-4)/(1+1)=3/2$				
4	$f[x_2]=17$	$f[x_1, x_2]$ $=(16-4)/(3-1)=6$		$f[x_0, x_1, x_2]$ $= (6-3/2)/(3+1)=9/8$		

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

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

Newton : l'algorithme

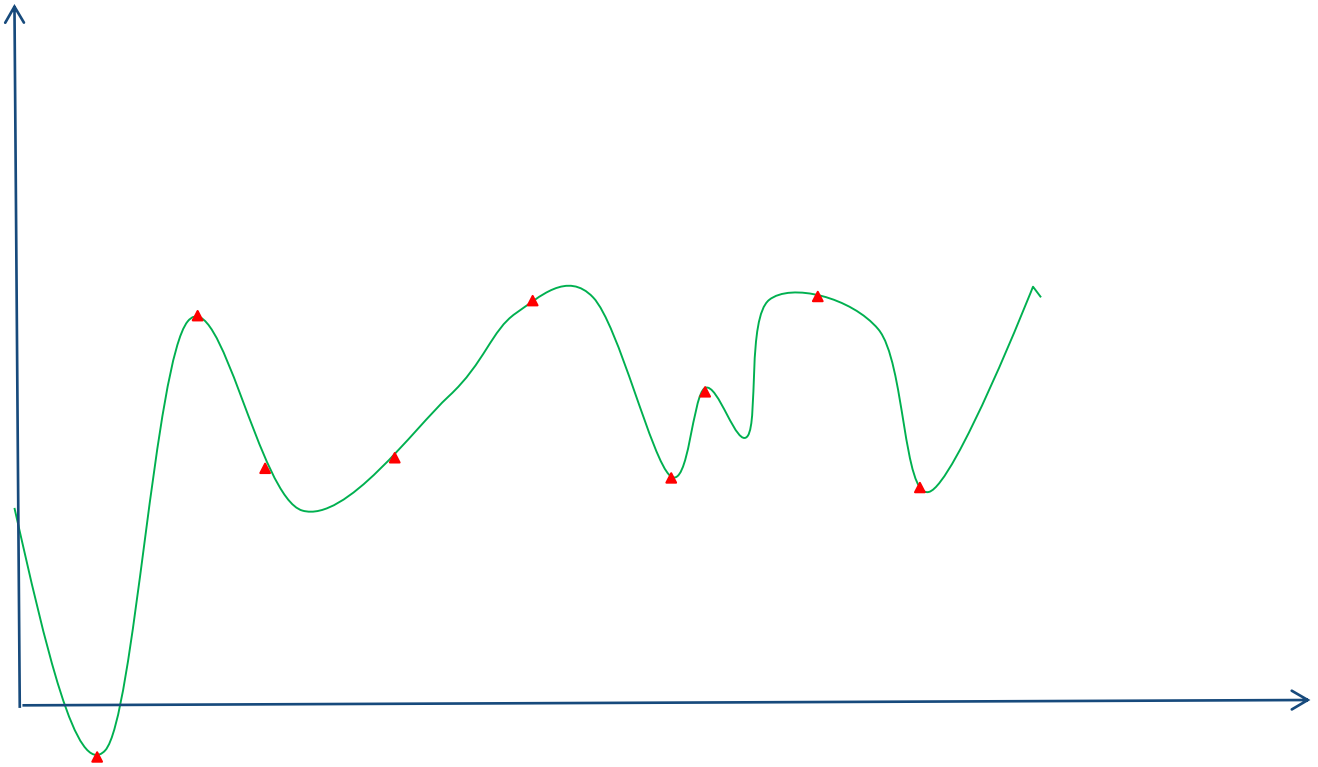
Fonction a = Newton(x_i,y_i)

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
        $F(i,j) \leftarrow \dfrac{F(i,j-1) - F(i-1,j-1)}{x_i(i) - x_i(i-j)}$
    fait
fait
pour $i = 1$ jusqu'à n
    $a(\mathrm{i}) \leftarrow F(n,i)$
fait

Vérifier que la complexité est de : n^2

Si le nombre de points est élevé

- entre les points, le polynôme fait ce qu'il veut !!!
  et plus son degré est élevé plus il est susceptible d'osciller !
- en dehors de l'intervalle des points d'interpolation la fonction tend vers ($\pm \infty$)



Interpolation par splines cubiques

Principe :

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

Comment

– on décompose l'espace de définition (des points) en un ensemble contigu d'intervalles sur lesquels on applique des interpolations polynômiale de degré 3

➔ Résultat un ensemble de polynômes définis de façon continue

Splines cubiques : définition

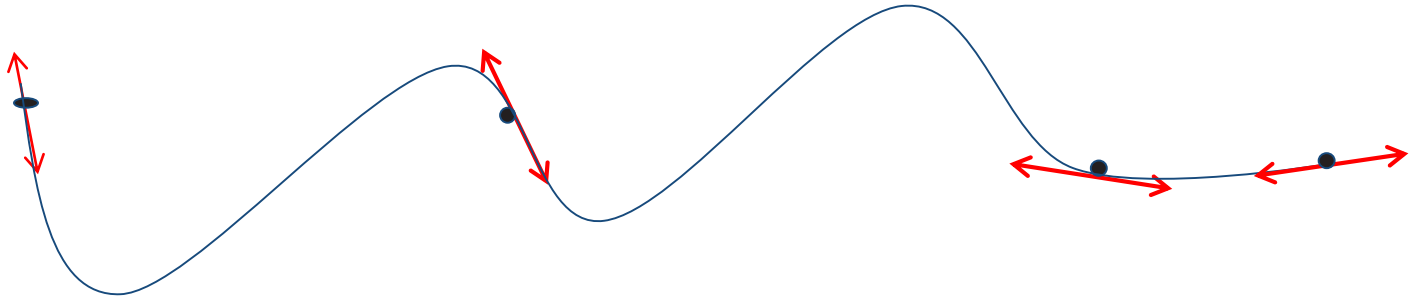
- Définition :
  - On appelle spline cubique (d'interpolation) une fonction notée *g*, qui vérifie les propriétés suivantes :
    - $g \in C^2[a;b]$ (*g* est deux fois continûment dérivable),
    - *g* coïncide sur chaque intervalle $[x_i; x_{i+1}]$ avec un polynôme de degré inférieur ou égal à 3,
 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

Splines cubiques : définition

- En plus :
  - Il faut des conditions supplémentaires pour définir la spline d'interpolation de façon unique
  - Ex. de conditions supplémentaires : conditions aux limites
    - $g''(a) = g''(b) = 0$ spline naturelle.

- Remarques :
  - Ces conditions permettent d'avoir une courbe continue et d'aspect lisse
  - Forme ≡ forme d'une barre souple soumise à des contraintes physiques

Splines : illustration



$$P_1(x)=\alpha_1 x^3+\beta_1 x^2+\chi_1 x+\delta_1$$

$$= a_1 (x\text{-}x_1)^3+b_1 (x\text{-}x_1)^2+c_1 (x\text{-}x_1)+d_1$$

$$P_2(x)=a_2 (x\text{-}x_2)^3+b_2 (x\text{-}x_2)^2+c_2 (x\text{-}x_2)+d_2$$

Splines cubiques : détermination

- Détermination de la spline d'interpolation
  - *g* coïncide sur chaque intervalle $[x_i; x_{i+1}]$ avec un polynôme de degré inférieur ou égal à 3

  - *g"* est de degré 1 et est déterminé par 2 valeurs:
    - $m_i = g''(x_i)$ et $m_{i+1} = g''(x_{i+1})$ (moment au noeud n°*i*)

  - Notations :
    - $h_i = x_{i+1} - x_i$ pour $i = 0 \ldots n-1$
    - $\delta_i = [x_i; x_{i+1}]$
    - $g_i(x)$ le polynôme de degré 3 qui coïncide avec *g* sur l'intervalle δ_i

Splines cubiques : équations

$g''_i(x)$ est linéaire : on peut l'estimer par la méthode de Lagrange

- $\forall\, x \in \delta_i$

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

- on intègre

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

(a_i constante)

Splines cubiques : équations

- On continue (b_i constante)

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

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

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

Splines cubiques : équations

- *g′(x)* est continue : $g_i'(x_i) = -m_i \frac{h_i}{2} + a_i = m_i \frac{h_{i-1}}{2} + a_{i-1} = g_{i-1}'(x_i)$ (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)

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

(4)

Splines cubiques : équations

- Rappel : on cherche les m_i (n+1 inconnues)
  - on a seulement n-1 équations grâce aux données
  - il faut rajouter 2 conditions → par exemple condition aux limites
    - $m_0 = m_n = 0$ *(spline naturelle)*

Splines cubiques : calcul des coefficients

4

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

- Ex de résolution avec $h_i = x_{i+1} - x_i$ (h_i constant) :
  - $$m_{i-1} + 4m_i + m_{i+1} = \frac{1}{h^2}(y_{i-1} - 2y_i + y_{i+1}) = f_i$$
  - Forme matricielle

$$Tm = f \qquad \begin{pmatrix} 4 & 1 & & & 0 \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{pmatrix} \begin{pmatrix} m_1 \\ \\ \cdots \\ \\ m_{n-1} \end{pmatrix} = \begin{pmatrix} f_1 \\ \\ \cdots \\ \\ f_{n-1} \end{pmatrix}$$

  - T inversible (diagonale strictement dominante)

Splines cubiques : algorithme

pour $i = 2; n-1$

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

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

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

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

fin pour

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

$m \leftarrow T/f$

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

pour $i = 1; n-1$

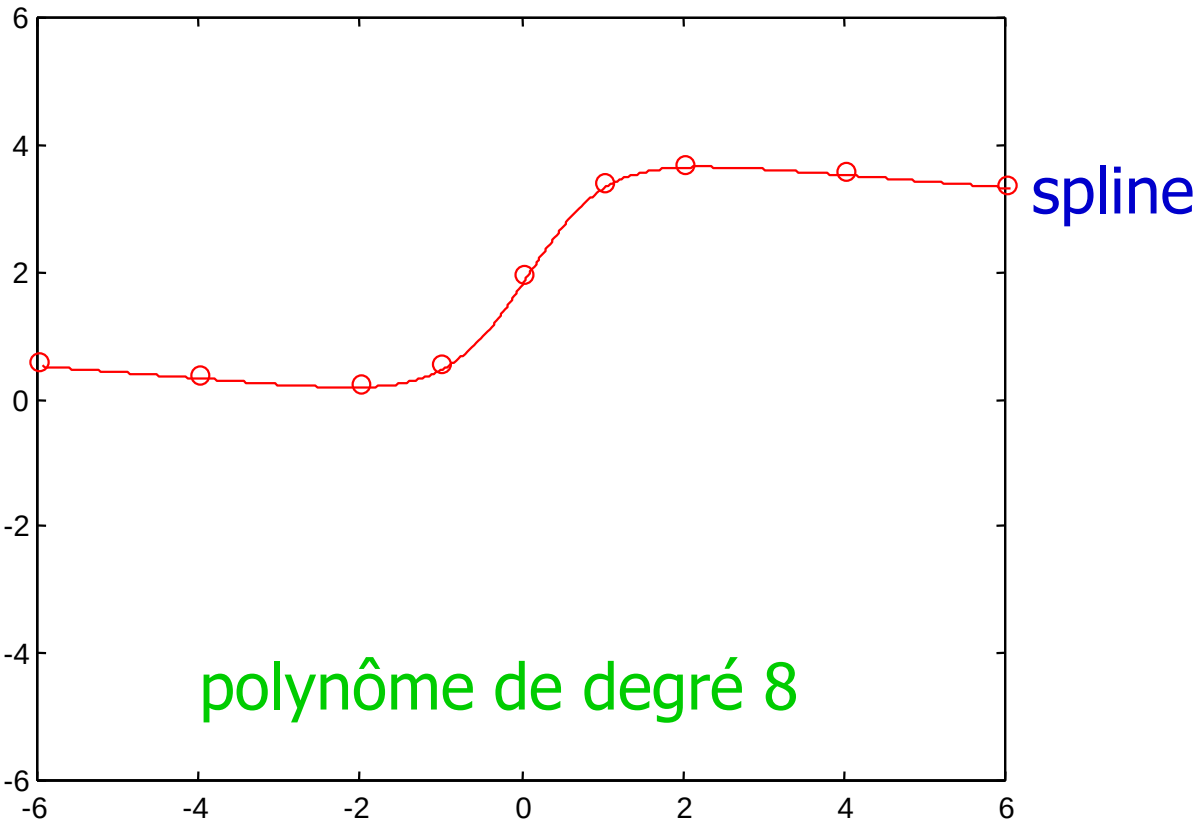
$a(i)$
$\leftarrow \frac{1}{h_i}(y_{i+1} - y_i) - \frac{h_i}{6}(m_{i+1} - m_i)$

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

fin pour

Splines cubiques : exemple

- Ex : avec 9 points → voir une interpolation générale ??



Conclusion

- Interpolation polynomiale
  - évaluer la fonction en un point : Polynôme de Lagrange -> méthode de Neville
  - *compiler* la fonction : Polynôme de Newton

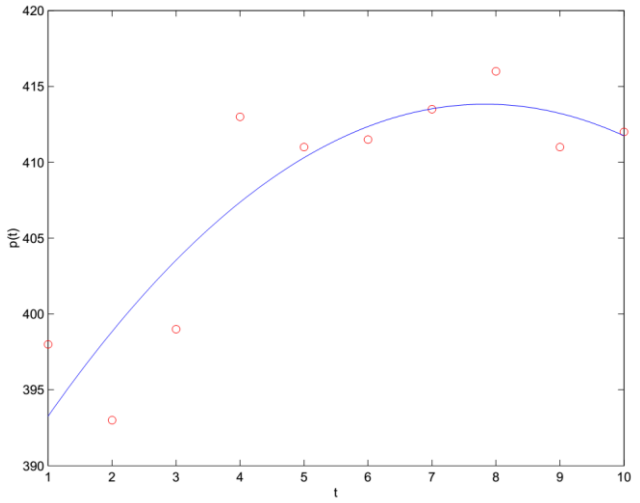
- Interpolation polynomiale par morceau : splines
  - spline cubique d'interpolation : passage par les nœuds (points d'interpolation), mais on limite les oscillations.
  - spline cubique d'approximation : on régule mieux la fonction, mais minimise la distance aux nœuds (les points de passage)

Approximation aux moindres carrés

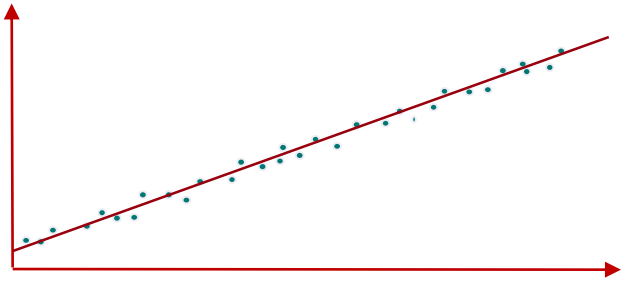
- Exemples

(1) Typiquement, on suppose disposer d'un jeu mesure (x_i, y_i)

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



(2) $g(x, a0, a1) = a_0 + a_1 x$



- **Données** : un ensemble de points $(x_0, y_0),(x_1, y_1), \ldots, (x_r, y_r)$.
- **On cherche** : 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).
  - $f(x, a_0, a_1, \ldots, a_r)$
  - $a_0, a_1, \ldots, a_r$ sont des constantes à régler en fonction des points de l'ensemble.
- but : minimiser la distance entre f et l'ensemble des points.

Cas d'un polynôme

- **Données** : $(x_0, y_0),(x_1, y_1), \ldots, (x_n, y_p)$.

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

$$f(x) = \sum_{k=0}^{n} a_k x^k$$

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

(n+1) équations à (n+1) inconnues)

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

Cas d'un polynôme (suite)

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

Si $n < p$: on a une approximation

(p équations à n inconnues : plus d'équations que d'inconnues)

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

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 $(x_i ; y_i)$ et bi sont souvent des résultats d'expériences ou de mesures, pouvant être entaches d'erreurs.

Cas d'un polynôme (suite)

$$f(x_i) = \sum_{k=0}^{n} a_k x_i^k \qquad pour\ (i = 0, \cdots, p)$$

n le degré du polynôme. En développant les équations on obtient :

$$\begin{cases} a_0 x_0^0 + a_1 x_0^1 + a_2 x_0^2 + \cdots + a_n x_0^n = y_O \\ a_0 x_1^0 + a_1 x_1^1 + a_2 x_1^2 + \cdots + a_n x_1^n = y_1 \\ \vdots \\ a_0 x_j^0 + a_1 x_j^1 + a_2 x_j^2 + \cdots + a_n x_j^n = y_j \\ \vdots \\ a_0 x_p^0 + a_1 x_p^1 + a_2 x_p^2 + \cdots + a_n x_p^n = y_p \end{cases}$$

Cas d'un polynôme (suite)

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

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

$$\varphi(A) = \sum_{i=0}^{p} \left(\sum_{k=0}^{n} (a_k x_i^k - y_i) \right)^2$$

$$A^* = \underset{A}{\text{argument}}\, \varphi(A) \quad \Leftrightarrow \frac{\partial \varphi}{\partial a_k}(A^*) = 0 \quad k = 0, \cdots, n$$

Le minimum est atteint au point où les dérivées s'annulent.

Remarque : on minimise par rapport aux coefficients a_k

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

Cas d'un polynôme (suite)

Le minimum est atteint au point où les dérivées s'annulent.

Remarque : on minimise par rapport aux coefficients a_k

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

$$\Updownarrow$$

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

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

Cas d'un polynôme (suite)

Détail du calcul :

$$\mathrm{f}\,(x, a_0, \ldots, a_r) = a_0 + a_1 x + \cdots + a_r x^{\mathrm{p}}.$$

Distance :

$$\varphi(a_0, \ldots, a_r) = \sum_{i=0}^{n} (y_i - (a_0 + a_1 x_i + \cdots + a_{\mathrm{p}} x_i^{\mathrm{p}}))^2.$$

Dérivée pour $k = 0, \ldots, r$:

$$\frac{\varphi(a_0, \ldots, a_{\mathrm{p}})}{\partial a_k} = \sum_{i=0}^{n} \left[2 \cdot (y_i - (a_0 + a_1 x_i + \cdots + a_{\mathrm{p}} x_i^{\mathrm{p}}))\,(-x_i^k)\right] = 0$$

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

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

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

$$\left(\sum_{i=0}^{p} x_i^k\right) a_0 + \left(\sum_{i=0}^{p} x_i^{k+1}\right) a_1 + \cdots + \left(\sum_{i=0}^{p} x_i^k\right) a_n$$

On obtient le système linéaire suivant :

$$\begin{pmatrix} \sum i & \sum x_i & \sum x_i^2 & \cdots & \sum x_i^n \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & & \sum x_i^{n+1} \\ & \vdots & \vdots & \cdots & \vdots \\ \sum x_i^n & \sum x_i^{n+1} & \sum x_i^{n+2} & \cdots & \sum x_i^{2n} \end{pmatrix} \times \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \\ a_n \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum y_i x_i \\ \vdots \\ \sum y_i x_i^n \end{pmatrix}$$

Cas d'un polynôme (suite)

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

$X \cdot A = Y$ avec $A = \{a_0, \cdots a_{n-1}\}$

$$\overbrace{\begin{bmatrix} x_0^0 & x_0^1 & x_0^2 & \cdots & x_0^{n-1} \\ x_1^0 & x_1^1 & x_1^2 & \cdots & x_1^{n-1} \\ & & \vdots & & \\ x_j^0 & x_1^1 & x_j^2 & \cdots & x_j^{n-1} \\ & & \vdots & & \\ x_p^0 & x_p^1 & x_1^2 & \cdots & x_p^{n-1} \end{bmatrix}}^{X} \cdot A = Y = \overbrace{\begin{bmatrix} 1 & x_0^1 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1^1 & x_1^2 & \cdots & x_1^{n-1} \\ & & \vdots & & \\ 1 & x_1^1 & x_j^2 & \cdots & x_j^{n-1} \\ & & \vdots & & \\ 1 & x_p^1 & x_p^2 & \cdots & x_p^{n-1} \end{bmatrix}}^{x_j^0 = 1}$$

On a un système Linéaire surdéterminé voir la suite

Cas particulier : régression Linéaire

C'est une approximation par un polynôme de degré 1

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

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 moyenne de $x_i, y_i, \ldots$.

On obtient alors :

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

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

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

Cas général d'un système linéaire

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

$$
\begin{cases}
m_{11}a_1 + a_{12}a_2 + \ldots\ldots + a_{1n}a_n = y_1 + r_1 \\
m_{21}a_1 + m_{22}a_2 + \ldots\ldots + m_{2n}a_n = y_2 + r_2 \\
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
m_{p1}x_1 + m_{p2}a_2 + \ldots\ldots + m_{pn}a_n = y_p + r_p
\end{cases}
$$

m_{ik} : coefficients des équations de 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

$$\begin{bmatrix} m_{11} & m_{12} & \ldots\ldots & m_{1n} \\ m_{21}x_1 & m_{22} & \ldots\ldots & m_{2n} \\ : & : & : & : \\ : & : & : & : \\ m_{m1} & m_{m2} & \ldots\ldots & m_{pn} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ : \\ : \\ a_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ : \\ : \\ y_p \end{bmatrix} + \begin{bmatrix} r_1 \\ r_2 \\ : \\ : \\ r_p \end{bmatrix}$$

La méthode des M.C.

Déterminer a_k tq les sommes des carrés des résidus soit minimales.

Cas d'un système linéaire

En notation Matricielle Ax = Y+R

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

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

Avec $A = [a_0, \cdots, a_n]^t$

La minimisation de S par rapport à a_k avec k = 1, …n entraine les conditions nécessaires : $\frac{\partial S}{\partial a_k} = 0$

Cas 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 !

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

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

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

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

$$\log_b x^p = {}^p \log_b x$$

Régression exponentielle

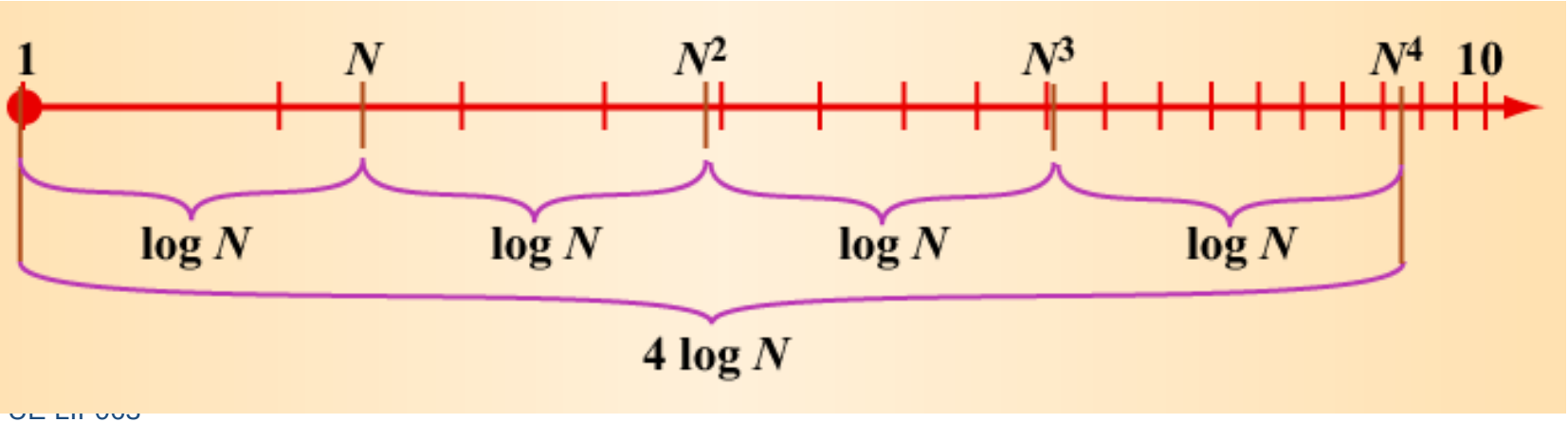
- L'exemple le plus connu est la modélisation de la radioactivité d'un déchet nucléaire ou la modélisation de l'évolution de la population !

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

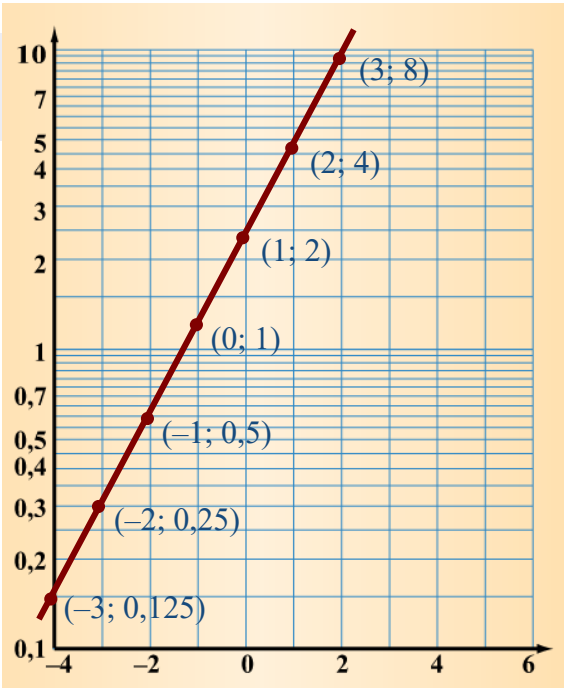
$$\log g(t, a, b) = \log b - at,$$

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 :

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

On revient à un système linéaire

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

$y = AX + B$, où $X = \log x$ et *A et B sont des coefficients réels*.

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.