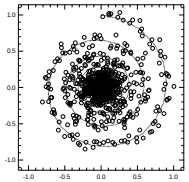
Algèbre linéaire
Applications à la modélisation des données

MNCS – Jean Hare, Noé Lahaye, Jacques Lefrère

Université Pierre et Marie CURIE

Méthodes numériques pour le calcul scientifique

14 et 15 février 2019



Sommaire I

Sommaire II

Algèbre linéaire

- Applications très diverses (physique linéaire ou quasi-linéaire)
- **Optimisation, modélisation**
- **EDP linéaires** (Thème 4)
- Indispensable aussi pour certains calculs *non-linéaires*
- Problèmes spécifiques :
  - très gros volumes de données (ressources en N^2)
  - matrices creuses, matrices bandes
  - multiplication : en N^3
  - précision après un très grand nombre d'opérations
  - tableaux 2-D en C
- Solutions spécifiques :
  - Procédures de base fortement optimisées (BLAS)
  - **Bibliothèques** diverses (LAPACK)
  - Vectorisation, calcul parallèle (non abordé ici)

LAPACK et BLAS sont intégrés dans Python, Scilab/MatLab, Maple *etc*

Matrices rectangulaires

Ici A est une matrice à M lignes et N colonnes, ou de profil (M, N) (en anglais, *shape*).

$$\mathsf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdot & a_{1j} & \cdots a_{1N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i1} & a_{i2} & \cdot & a_{ij} & \cdots a_{iN} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{M1} & a_{M2} & \cdot & a_{Mj} & \cdots a_{MN} \end{pmatrix}$$

Nota important : **Par convention, en mathématiques,**

- $i \equiv$ premier indice $\equiv$ indice de ligne, de 1 à M
- $j \equiv$ deuxième indice $\equiv$ indice de colonne, de 1 à N

Une matrice carrée est un cas particulier $N = M$.
Un vecteur colonne correspond à $N = 1$, un vecteur ligne à $M = 1$.

Matrices particulières I

Réelle symétrique : matrice qui coïncide avec sa transposée ${}^{\mathrm{t}}\mathsf{A} = \mathsf{A}$ soit `A(i,j)==A(j,i)` ;

Hermitienne matrice complexe qui coïncide avec son adjointe (auto-adjointe) $\mathsf{A} = \mathsf{A}^* = \overline{{}^{\mathrm{t}}\mathsf{A}}$, soit `A(i,j)==CONJG(A(j,i))` ;

Orthogonale matrice réelle vérifiant ${}^{\mathrm{t}}\mathsf{A} = \mathsf{A}^{-1}$;

Unitaire matrice complexe vérifiant $\mathsf{A}^* = \mathsf{A}^{-1}$;

(Symétrique) définie positive matrice dont les valeurs propres sont strictement positives ;

À diagonale dominante $|\mathsf{A}(i,i)| \geqslant \sum_{\substack{j=1 \\ j\neq i}}^{n} |\mathsf{A}(i,j)| \quad \forall i = 1, 2, \ldots n$

À diagonale strictement dominante $|...| > \sum \quad \Rightarrow$ inversible

Matrices particulières II

Creuse matrice contenant beaucoup de zéros...

Diagonale matrice dont les éléments non-diagonaux sont nuls
Cas particulier : matrice *scalaire* $\mathsf{A} = \lambda\mathbb{1}$;

Diagonale par bande seuls ses éléments tels que $|i - j| \leqslant p$ ($p \ll N$) sont non-nuls
Cas particulier : matrice *tridiagonale* pour $p = 1$;

Triangulaire supérieure/inférieure tous les éléments au-dessous de la diagonale ($i > j$) /au-dessus de la diagonale ($i < j$) sont nuls ;

Hessenberg tous les éléments en dessous (au dessus) de la première sous-diagonale (sur-diagonale) sont nuls ;

Diagonale par blocs matrice nulle en dehors d'une diagonale de blocs carrés ;

Norme des matrices carrées

Exemples de normes des vecteurs

Euclidienne ou L_2 $\|\vec{u}\|_2 = \sqrt{\sum_{i=1}^{n} u_i^2}$
$\|\vec{u}\|_2 \leqslant 1$ définit une sphère

Sup L_∞ $\|\vec{u}\|_\infty = \underset{1\leqslant i\leqslant n}{\mathrm{Max}} |u_i|$
$\|\vec{u}\|_\infty \leqslant 1$ définit un hypercube

normes équivalentes $\|\vec{u}\|_\infty \leqslant \|\vec{u}\|_2 \leqslant \sqrt{n}\|\vec{u}\|_\infty \quad \forall \vec{u}$

Distance entre deux vecteurs $\|\vec{u} - \vec{v}\|$

Normes induites d'une matrice carrée

$$\|\mathsf{A}\| = \underset{\|\vec{u}\|=1}{\mathrm{Max}} \|\mathsf{A}u\| = \underset{\vec{u}\neq 0}{\mathrm{Max}} \frac{\|\mathsf{A}\vec{u}\|}{\|\vec{u}\|}$$

Propriété $\|\mathsf{A}\|_\infty = \underset{1\leqslant i\leqslant n}{\mathrm{Max}} \sum_{j=1}^{n} |a_{ij}|$

Conditionnement des matrices

Définition

Si A est inversible, on définit le conditionnement de A par :

$$\gamma(A) = \|A\| \, \|A^{-1}\| \geqslant 1$$

Système mal conditionné

Une matrice « proche » d'une matrice singulière possède un déterminant proche de zéro et un conditionnement élevé.
La résolution du système $A\vec{x} = \vec{b}$ est alors très sensible aux erreurs d'arrondi. On le qualifie de système « mal conditionné ».
Exemple : calcul du point d'intersection de deux droites presque parallèles.

Conditionnement et rayon spectral des matrices

Spectre des matrices carrées

Spectre d'une matrice = ensemble de ses valeurs propres
Rayon spectral d'une matrice

$$\rho(A) = \operatorname{Max}_i |\lambda_i|$$

où les λ_i sont les valeurs propres de A.

Propriété

$$\|A\|_2 = \sqrt{\rho(A^t A)} \quad \text{et si A est symétrique} \quad \|A\|_2 = \rho(A)$$

Donc si A est hermitienne, son conditionnement est :

$$\gamma(A) = \frac{|\lambda|_{\max}}{|\lambda|_{\min}}$$

Matrices convergentes

Définition

Matrice A convergente si

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

Propriétés

Matrice A convergente équivalent à

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

ou à

$$\lim_{k\to\infty} (A^k \vec{u}) = 0 \quad \forall \vec{u}$$

ou encore à

$$\rho(A) < 1$$

Stockage des matrices

En Fortran : Vrais Tableaux 2-D, `A(i,j)`,
indexés par défaut de `1` à `M` et de `1` à `N` comme les matrices

- statiques, déclarés avec leurs dimensions : `real :: a(3,5)`
- dynamiques, alloués avec `allocate`, libérés avec `deallocate` : `real,allocatable::a(:,:); m=3; n=5; allocate(a(m,n))`
- automatiques (moins utiles qu'en C).

En C : Tableaux de tableaux (pseudo 2-D) : `A[i][j]`,
indexés de `0` à `M-1` et `0` à `N-1`, pas comme les matrices.

- statiques, déclarés avec leurs dimensions : `float A[3][5];`
- dynamiques, alloués avec `calloc` (ou `malloc`), libérés avec `free` (eg via la bibliothèque `mnitab` : `float ** A; A=float2d(3,5)`)
- automatiques grâce à une déclaration tardive en C99 : `M=3; N=5; float A[M][N];`

La portée des tableaux automatiques est toujours limitée à la procédure.

NB: Pour utiliser des tableaux automatiques, réserver suffisamment de place sur la pile (stack) avec : Linux : `ulimit -s 524288` (en kbytes).

Ordre des éléments

En Fortran

Les éléments de la même colonne (second indice) sont contigus :

$$\underbrace{a_{11}\,|\,a_{21}\,|\cdots|\,a_{M1}}_{\text{colonne 1}}\;\underbrace{a_{12}\,|\,a_{22}\,|\cdots|\,a_{M2}}_{\text{colonne 2}}\;\cdots\;\underbrace{a_{1N}\,|\,a_{2N}\,|\cdots|\,a_{MN}}_{\text{colonne } N}$$

On dit que l'indice de ligne (premier indice) est l'indice rapide, ou que le stockage est en « Column Major ».

En C

Les éléments de la même ligne (premier indice) sont contigus :

$$\underbrace{a_{11}\,|\,a_{12}\,|\cdots|\,a_{1N}}_{\text{ligne 1}}\;\underbrace{a_{21}\,|\,a_{22}\,|\cdots|\,a_{2N}}_{\text{ligne 2}}\;\cdots\;\underbrace{a_{M1}\,|\,a_{M2}\,|\cdots|\,a_{MN}}_{\text{ligne } M}$$

On dit que l'indice de colonne (second indice) est l'indice rapide, ou que le stockage est en « Row Major ».

Matrice aplatie

Dans les deux langages, on peut aussi utiliser un tableau 1-D `Ap` pour représenter la matrice `A(M,N)` « aplatie ». C'est en particulier nécessaire pour l'usage de BLAS et de LAPACK en langage C :

- En Fortran, pour accéder à a_{ij}, remplacer `A(i,j)` par `Ap(i+M*(j-1))` car ColMajor ;
  avancer de `M` cases (le nombre de lignes) quand on incrémente `j`
  Leading Dimension = `M`
- En C, pour accéder à a_{ij}, remplacer `A[i-1][j-1]` par
  - `Ap[(i-1)+M*(j-1)]` en ColMajor, Leading Dimension= `M`
  - ou `Ap[N*(i-1)+(j-1)]` en RowMajor, Leading Dimension = `N`

⇒ Écrire une fonction qui calcule l'indice informatique `k` dans le tableau aplati de l'élément d'indices mathématiques `i` et `j` en représentation 2D. Elle utilise la Leading Dimension

Produit lignes par colonnes

Définition : Composition des applications linéaires : si A est une matrice de profil (M, N) et B est une matrice de profil (N, P), leur produit est une matrice de profil (M, P).

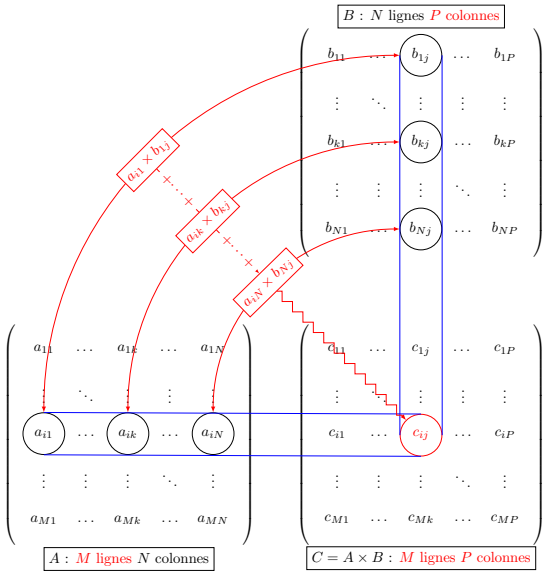
$$\mathsf{C} = \mathsf{A.B} \iff (\mathsf{C})_{ij} = (\mathsf{A.B})_{ij} = \sum_{k=1}^{N} A_{ik} B_{kj}$$

L'élément C_{ij} est le *produit scalaire* de la ligne i de la matrice A avec la colonne j de la matrice B, d'où le nom.

Cas particulier $P = 1$: action de A sur le vecteur-colonne B,
ou $M = 1$: action (à gauche) de la matrice B sur le vecteur-ligne A.

En Fortran : On utilise la fonction standard : `C = MATMUL (A, B)`.
En C : Cette fonction doit être implémentée avec des boucles.

Multiplication matricielle



Produit matriciel en C

On utilise *trois* boucles imbriquées, les indices extérieurs étant les indices libres i et j, la boucle interne formant la somme sur k.
Par exemple, pour des matrices aplaties en *RowMajor* on peut écrire :

```
void matmul_naive(int M, int N, int P,          // RowMajor
            float A[M*N], float B[N*P], float C[M*P]) {
    int i,j,k;
    float tmpij;
    for (i=0; i<M; ++i) {
        for (j=0; j<P; ++j) {
            tmpij = 0.0f;
            for (k=0; k<N; ++k) {tmpij += A[i*N+k] * B[k*P+j];}
            C[i*P+j] = tmpij;
        }
    }
}
```

Ces trois boucles imbriquées impliquent un nombre $M \times N \times P \sim N^3$ de multiplications, avec autant d'accès non-contigus dans les tableaux.

Autres manipulations

Le Fortran fournit encore de nombreuses fonctions de manipulation de matrices : `PRODUCT`, `SUM`, `TRANSPOSE`, `RESHAPE`, `DOT_PRODUCT`... enrichies par la notion de section de tableau et l'argument optionnel `DIM`.

En C toutes ces fonctions doivent être codées à l'aide de boucles imbriquées.

Pour une meilleure efficacité (vitesse d'exécution), surtout lorsqu'on doit traiter des matrices de taille importante, il devient essentiel d'utiliser des codes soigneusement optimisés. C'est, pour les deux langages, le principal intérêt des bibliothèques mathématiques, et nous présenterons dans la suite de ce cours l'utilisation d'une de ces bibliothèques, le couple BLAS/LAPACK.

Matrices opérant sur des matrices

Pour réaliser les inversions, diagonalisations et factorisations, on modifie de façon graduelle une matrice, par exemple en permutant des lignes et/ou des colonnes, en effectuant des combinaisons linéaires (CL) des lignes et/ou colonnes, *etc*.
Ces manipulations sont des *multiplications* par des matrices creuses.
On notera le résultat qui peut sembler contre-intuitif :

- La multiplication à gauche opère sur les lignes :
  les nouvelles lignes sont des CL des anciennes avec des poids définis par la ligne considérée de la matrice de gauche ;
- La multiplication à droite opère sur les colonnes :
  les nouvelles colonnes sont des CL des anciennes avec des poids définis par la colonne considérée de la matrice de droite ;

Ces règles sont utiles pour décrire les opérations mathématiques, et pour éviter de faire un produit complet (en N^3) lorsque la matrice « opérante » est creuse, comme c'est le cas de l'inversion par la méthode de Gauss, ou de la diagonalisation par l'algorithme de Jacobi (*cf* infra).

Inversion : position du problème

Définition : L'inverse d'une matrice *carrée* A de profil (N, N) est une matrice A^{-1} de même dimension telle que $A.A^{-1} = A^{-1}.A = \mathbb{1}$.

C'est un problème important notamment car il permet de résoudre les systèmes linéaires de N équations à N inconnues. Il permet aussi de déterminer le déterminant d'une matrice.

Le problème posé peut sembler plus ardu que la multiplication, mais sa complexité algorithmique et le nombre d'opérations impliquées ne sont pas beaucoup plus grands. Par contre, il implique des divisions flottantes, plus longues que des multiplications, et qui peuvent poser des problèmes de précision (matrice mal conditionnée).

Il existe deux types de méthodes pour calculer l'inverse d'une matrice, ou montrer que la matrice est singulière :

- les méthodes directes (pivot de Gauss et ses variantes)
- les méthodes itératives (Jacobi, Gauss-Seidel, Successive Over-Relaxation = SOR, ...)

Méthode du pivot de Gauss

Définition : C'est une forme systématique de la méthode « d'élimination », où l'on parcourt successivement toutes les lignes. À l'étape p, elle utilise des combinaisons linéaires de lignes pour annuler tous les éléments de la colonne p, à l'exception de l'élément diagonal a_{pp}, appelé « pivot ».
Matrice singulière : Si à un moment donné le(s) pivot(s) est (sont) nul(s) la matrice est singulière, mais le système réduit fournit une équation du sous-espace vectoriel des solutions.
Déterminant : C'est tout simplement le produit des pivots. Cette méthode en N^3 est très économique par rapport au développement par rapport à une ligne qui serait en $N!$, soit pour $N = 10$ un gain d'un facteur 3000 environ !
Décomposition LU : Si, au lieu d'annuler tous les éléments non-diagonaux d'une colonne, on se limite à ceux qui sont *sous* la diagonale, on obtient une matrice triangulaire *supérieure*, et les éliminations étant effectuées en multipliant successivement à gauche par des matrices triangulaires *inférieures*, on réalise la décomposition LU.

Gauss-Jordan : principe de base

À l'étape n°p (où $p = 1, \cdots, N$) on effectue les opérations suivantes :

1. éventuellement : recherche de l'élément max dans la colonne p sur les lignes en dessous de p ($q \geqslant p$), et permutation des lignes p et q
2. division de la ligne p par le pivot a_{pp} : $a_{pp} \leftarrow 1$;
3. $q \neq p$: $\mathsf{L}_q \leftarrow \mathsf{L}_q - a_{qp}\mathsf{L}_p$ (soit $a_{qr} \leftarrow a_{qr} - a_{qp} \times a_{pr}$)
   $\Rightarrow a_{pq} = 0$ pour $q \neq p$.

Chaque opération sur les lignes = multiplication à gauche de la matrice à inverser A par une matrice « opérant sur les lignes ».
Ensemble de ces opérations sur A $\Rightarrow \mathbb{1}_N$
Ensemble de ces opérations sur $\mathbb{1}_N \Rightarrow \mathsf{B} = \mathsf{A}^{-1}$

Un exemple avec pivotage partiel (1/3)

Considérons par exemple la matrice M ci-dessous. *Le premier pivot* est l'élément m_{11}, et on forme la matrice d'élimination L_1 :

$$\mathsf{M} = \begin{pmatrix} \mathbf{1} & 2 & 3 & 2 \\ -1 & 2 & -2 & -1 \\ 0 & 3 & -1 & 1 \\ -1 & 3 & -2 & 0 \end{pmatrix} \Rightarrow \mathsf{L}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix},$$

dont on déduit les matrices $\mathsf{M}' = \mathsf{L}_1 \cdot \mathsf{M}$ et $\mathsf{N}' = \mathsf{L}_1 \cdot \mathbb{1}$:

$$\mathsf{M}' = \mathsf{L}_1 \cdot \mathsf{M} = \begin{pmatrix} 1 & 2 & 3 & 2 \\ 0 & 4 & 1 & 1 \\ 0 & 3 & -1 & 1 \\ 0 & \mathbf{5} & 1 & 2 \end{pmatrix}, \quad \mathsf{N}' = \mathsf{L}_1 \cdot \mathbb{1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Le *second pivot* est l'élément $m'_{52} = 5$, que l'on fait remonter à la seconde ligne à l'aide de matrice de permutation $\mathsf{S}(2,4)$ pour échanger les lignes.

Exemple pivot de Gauss (2/3)

Les matrices $\mathsf{S}(2,4)$, $\mathsf{S} \cdot \mathsf{M}'$ et L_2 sont données par :

$$\mathsf{S} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \ \mathsf{S}\cdot\mathsf{M}' = \begin{pmatrix} 1 & 2 & 3 & 2 \\ 0 & \mathbf{5} & 1 & 2 \\ 0 & 3 & -1 & 1 \\ 0 & 4 & 1 & 1 \end{pmatrix}, \ \mathsf{L}_2 = \begin{pmatrix} 1 & -2/5 & 0 & 0 \\ 0 & 1/5 & 0 & 0 \\ 0 & -3/5 & 1 & 0 \\ 0 & -4/5 & 0 & 1 \end{pmatrix}$$

d'où $\mathsf{M}'' = \mathsf{L}_2.\mathsf{S}(2,4).\mathsf{M}'$, $\mathsf{N}'' = \mathsf{L}_2.\mathsf{S}(2,4).\mathsf{N}'$:

$$\mathsf{M}'' = \begin{pmatrix} 1 & 0 & 13/5 & 6/5 \\ 0 & 1 & 1/5 & 2/5 \\ 0 & 0 & \mathbf{-8/5} & -1/5 \\ 0 & 0 & 1/5 & -3/5 \end{pmatrix}, \quad \mathsf{N}'' = \begin{pmatrix} 3/5 & -2/5 & 0 & 0 \\ 1/5 & 1/5 & 0 & 0 \\ -3/5 & -3/5 & 1 & 0 \\ 1/5 & -4/5 & 0 & 1 \end{pmatrix}.$$

Le *troisième pivot* est alors $m'_{33} = -8/5$ (pas de permutation).

Exemple pivot de Gauss (3/3)

On obtient alors respectivement L_3 , $M'''= L_3 \cdot M''$ et $N''' = L_3 \cdot N''$:

$$\begin{pmatrix} 1 & 0 & -13/8 & 0 \\ 0 & 1 & -1/8 & 0 \\ 0 & 0 & -5/8 & 0 \\ 0 & 0 & -1/8 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 7/8 \\ 0 & 1 & 0 & 3/8 \\ 0 & 0 & 1 & 1/8 \\ 0 & 0 & 0 & \mathbf{-5/8} \end{pmatrix}, \begin{pmatrix} -3/8 & 0 & 13/8 & -11/8 \\ 1/8 & 0 & 1/8 & 1/8 \\ 3/8 & 0 & -5/8 & 3/8 \\ 1/8 & 1 & 1/8 & -7/8 \end{pmatrix}.$$

Le *quatrième pivot* est alors $m'_{44} = -5/8$, donc L_4, $M'''' = \mathbb{1}$, et $N'''' \equiv M^{-1}$ sont

$$\begin{pmatrix} 1 & 0 & 0 & -7/5 \\ 0 & 1 & 0 & -3/5 \\ 0 & 0 & 1 & -1/5 \\ 0 & 0 & 0 & -8/5 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ et } M^{-1} = \frac{1}{5}\begin{pmatrix} -1 & 7 & 9 & -13 \\ 1 & 3 & 1 & -2 \\ 2 & 1 & -3 & 1 \\ -1 & -8 & -1 & 7 \end{pmatrix}.$$

Le déterminant est le produit des pivots : $\det(M) = 1 \times 5 \times \frac{-8}{5} \times \frac{-5}{8} = 5$ comme un calcul plus ordinaire mais plus laborieux nous l'aurait donné.

Décompositions usuelles

Bien des opérations reposent sur différentes factorisations de la matrice, dont les facteurs se prêtent à une inversion simple :

- Factorisation LU : met la matrice sous la forme $A = P \cdot L \cdot U$, où
  - P est une matrice de permutation (*cf* exemple 2 ci-dessus)
  - L (lower) est triangulaire inférieure dont les éléments diagonaux sont égaux à 1,
  - U (upper) est triangulaire supérieure.

  Elle s'obtient par une modification minime, mais efficace de la méthode de Gauss.
- Factorisation de Cholesky : $A = L \cdot L^*$ ou $A = U^* \cdot U$, et autres variantes, si A est symétrique définie positive.
- Factorisation QR : $A = Q \cdot R$ où Q est une matrice orthogonale (ou unitaire si A est complexe) et R une matrice triangulaire supérieure.

et bien d'autres méthodes fondées sur la décomposition en valeurs singulières, ou la diagonalisation.

Introduction aux méthodes itératives

Principe

Itérer une application affine de matrice M et de vecteur de translation C sur le vecteur X :

$$X^{(k)} = MX^{(k-1)} + C$$

telle que le point fixe soit solution du système $AX^{(\infty)} = B$
Critère d'arrêt : $\|X^{(k)} - X^{(k-1)}\| \leqslant \eta \, \|X^{(k-1)}\|$

Série géométrique de raison M^k : Convergence si $\rho(M) < 1$
Les méthodes se distinguent par le choix de la matrice M et du vecteur C.
La matrice A du système est décomposée en $A = D + L + U$

- D diagonale,
- L triangulaire inférieure (stricte),
- U triangulaire supérieure (stricte).

D est supposée inversible ($a_{ii} \neq 0 \; \forall i$), donc $D + L$ est inversible.

Méthode de Jacobi

En utilisant la i^{e} équation du système $AX = B$,

$$a_{i1}x_1 + \cdots + a_{i,i-1}x_{i-1} + a_{ii}\boxed{x_i} + a_{i,i+1}x_{i+1} + \cdots + a_{i,n}x_n = b_i$$

calculer chaque composante i du nouveau vecteur $X^{(k)}$ en fonction des autres composantes ($j \neq i$) du vecteur antérieur $X^{(k-1)}$.

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

$$X^{(k)} = -D^{-1}(L + U)X^{(k-1)} + D^{-1}B$$
$$M_J = -D^{-1}(L + U) \quad \text{et} \quad C = D^{-1}B$$

On vérifie que $X^{(\infty)} = -D^{-1}(L + U)X^{(\infty)} + D^{-1}B$ implique $DX^{(\infty)} = -(L + U)X^{(\infty)} + B$, soit $AX^{(\infty)} = B$

Méthode de Gauss-Seidel

La variante de Gauss-Seidel consiste à utiliser, dans la i^{e} équation, les composantes déjà calculées ($j < i$) du nouveau vecteur pour déterminer la composante i du nouveau vecteur.

$$x_i^{(k)} = -\frac{1}{a_{ii}} \sum_{j=1}^{j=i-1} a_{ij} x_j^{(k)} - \frac{1}{a_{ii}} \sum_{j=i+1}^{j=n} a_{ij} x_j^{(k-1)} + \frac{b_i}{a_{ii}} \qquad (2)$$

$$\mathrm{X}^{(k)} = -(\mathrm{D}+\mathrm{L})^{-1}\mathrm{U}\mathrm{X}^{(k-1)} + (\mathrm{D}+\mathrm{L})^{-1}\mathrm{B}$$

$$\mathrm{M}_G = -(\mathrm{D}+\mathrm{L})^{-1}\mathrm{U} \quad \text{et} \quad \mathrm{C} = (\mathrm{D}+\mathrm{L})^{-1}\mathrm{B}$$

Convergence plus rapide que Jacobi car $\rho(\mathrm{M}_G) = \rho^2(\mathrm{M}_J)$

On vérifie que $\mathrm{X}^{(\infty)} = -(\mathrm{D}+\mathrm{L})^{-1}\mathrm{U}\mathrm{X}^{(\infty)} + (\mathrm{D}+\mathrm{L})^{-1}\mathrm{B}$ implique

$(\mathrm{D}+\mathrm{L})\mathrm{X}^{(\infty)} = -\mathrm{U}\mathrm{X}^{(\infty)} + \mathrm{B}$, soit $\mathrm{A}\mathrm{X}^{(\infty)} = \mathrm{B}$

Diagonalisation : Position du problème

Définition : La diagonalisation d'une matrice *carrée* A de profil (N, N) est une décomposition de la matrice sous la forme $\mathrm{A} = \mathrm{V}\mathrm{D}\mathrm{V}^{-1}$ où D est une matrice diagonale et V une matrice inversible, dite « matrice de passage ».

Éléments spectraux Lorsqu'une telle décomposition est réalisée, les éléments diagonaux de D sont les valeurs propres de A et les colonnes de V les vecteurs propres correspondants.

Existence Cette décomposition n'existe pas toujours, mais on montre que les matrices réelles symétriques sont diagonalisables de même que les matrices complexes hermitiques. De plus les matrices définies positives (ou négatives) sont toujours diagonalisables avec une matrice de passage qui est orthogonale réelle ou unitaire dans le cas complexe.

Décomposition de Schur Dans le cas général, une matrice n'est pas diagonalisable, et on se contente de cette décomposition $\mathrm{A} = \mathrm{U} \cdot \mathrm{T} \cdot \mathrm{U}^*$ où U est unitaire et T est triangulaire supérieure avec les valeurs propres de A sur sa diagonale. La forme canonique de Jordan en est un cas particulier.

L'algorithme de Jacobi

Une méthode de diagonalisation élémentaire des matrices symétriques est la méthode de Jacobi. C'est une méthode *itérative* qui annule successivement les éléments non-diagonaux.

En dimension 2, on peut retrancher à la matrice A à diagonaliser la matrice scalaire $\mathrm{tr}(\mathrm{A})\mathbb{1}/2$, sans changer ses vecteurs propres. La matrice A′ ainsi obtenue est proportionnelle à une matrice de symétrie

$\mathrm{S}(\theta) = \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{pmatrix}$, que l'on sait diagonaliser en utilisant la matrice de rotation d'angle θ.

En dimension n On utilise la même approche, en annulant successivement les éléments non diagonaux a_{pq}. En considérant la sous-matrice

$\mathrm{A}_{pq} = \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix}$, on peut construire comme en dimension 2 une rotation d'angle θ qui n'affecte que les vecteurs de base X_p et X_q.

Algorithme de Jacobi en dimension n

A chaque étape, on doit ainsi multiplier A à gauche par la matrice R_{pq} ci-dessous, et à droite par sa transposée, avec les notations précédentes, en posant $c = \cos\theta$ et $s = \sin\theta$:

$$\mathrm{A} = \begin{array}{c} \\ 1 \\ \vdots \\ p \\ \\ q \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 \cdots & & p & \cdots & q & \cdots n \end{array} \\ \begin{pmatrix} & & & & & \\ & a_{pp} & & a_{pq} & & \\ & & & & & \\ & a_{qp} & & a_{qq} & & \end{pmatrix} \end{array} \Rightarrow \mathrm{R}_{pq} = \begin{array}{c} \begin{array}{ccccccc} 1 \cdots & & p & \cdots & q & \cdots n \end{array} \\ \begin{pmatrix} \mathbb{1} & \begin{matrix} 0 \\ \vdots \end{matrix} & \mathbb{0} & \begin{matrix} 0 \\ \vdots \end{matrix} & \mathbb{0} \\ 0 \cdots & c & \cdots & s & \cdots 0 \\ \mathbb{0} & \vdots & \mathbb{1} & \vdots & \mathbb{0} \\ 0 \cdots & -s & \cdots & c & \cdots 0 \\ \mathbb{0} & \begin{matrix} \vdots \\ 0 \end{matrix} & \mathbb{0} & \begin{matrix} \vdots \\ 0 \end{matrix} & \mathbb{1} \end{pmatrix} \end{array}$$

Bien sûr on ne calcule pas les produits de matrices, mais simplement :

– 2 CL de lignes pour le produit à gauche

– 2 CL de colonnes pour le produit à droite.

Itérations et convergence

- Cette opération ne fait qu'annuler deux éléments non-diagonaux : on doit l'itérer, par exemple en choisissant à chaque pas l'élément non-diagonal le plus grand (en valeur absolue).
- On peut montrer que le caractère non-diagonal, $\mathcal{E}(\mathrm{A}) = 2\sum_{i<j} a_{ij}^2$ diminue de $2a_{pq}^2$ à chaque itération, et en déduire que $\mathcal{E}$ tend vers zéro plus vite qu'une suite géométrique de raison $\rho < 1$, dépendant bien sûr de la matrice A.
- Le processus est donc convergent, on l'arrête en utilisant un *critère de convergence* défini par $\mathcal{E}(\mathrm{A}) \leqslant \eta^2 \mathcal{N}(\mathrm{A})$ où $\mathcal{N}(\mathrm{A}) = \sum_{i,j} a_{ij}^2$, et la tolérance η est choisie en fonction de la précision recherchée, mais aussi de façon à éviter d'accumuler des erreurs d'arrondi.
- En définitive, la matrice A est devenue quasi-diagonale, et la matrice de passage est le produit des rotations utilisées.

Bibliothèques : présentation

- Les codes fondés sur des algorithmes et des procédures d'optimisation bien établis sont « consolidés » dans des bibliothèques.
- Il existe un nombre considérable de bibliothèques, dédiées à de très nombreux types de calcul : AL, FFT et ondelettes, analyse spectrale, intégration, EDO, EDP et éléments finis, fonctions spéciales, traitement d'images, CAO dans différents domaines *etc*.
- On en recense près de 80 sur l'algèbre linéaire `http://www.netlib.org/utk/people/JackDongarra/la-sw.html`
- Chaque fabriquant de processeurs ou éditeur de compilateurs orientés « calcul scientifique » propose, moyennant finances, une bibliothèque mathématique qui intègre bien souvent des bibliothèques « open source », optimisées pour leur compilateur ou leurs microprocesseurs.

Voir `http://en.wikipedia.org/wiki/List_of_numerical_analysis_software`

Le couple BLAS/LAPACK

Choix d'une bibliothèque (*library*) du domaine public :

- la bibliothèque LAPACK (Linear Algebra PACKage), fondée elle-même
- sur la librairie BLAS (Basic Linear Algebra Subprograms), qui fournit les manipulations élémentaires « de bas niveau ».

Avantages de ces bibliothèques :

- gratuites ⇒ intégrées dans de très nombreux programmes comme Python, Maple, Scilab, Matlab, Octave *etc*
- aussi des versions propriétaires de Intel, Sun, HP, NEC, Irix, NAG, *etc*
- Écrites au départ en Fortran 77, elles ont été adaptées à de nombreux langages : Fortran 95, C, C++, Java, Python *etc*
- Un programme fondé sur LAPACK est aisément optimisé pour une nouvelle architecture (vectorisation, calcul parallèle, calcul distribué) en changeant uniquement la version de BLAS sous-jacente.
- Il existe aussi des extensions pour des types de matrices non pris en compte (matrices creuses ou par blocs *etc*).

Appels en Fortran 95 et en C

Mise en garde : pour économiser la mémoire, les procédures travaillent souvent « en place », en écrivant le résultat à la place des données d'entrée.

Les bibliothèques écrites en Fortran 77 sont directement utilisables depuis le Fortran 95 avec tous ses avantages grâce à des « wrappers » :

- Le module `blas95` sécurise l'appel des subroutines de BLAS en fournissant leur interface et simplifie les passages de tableaux ;
- Le package `lapack95` simplifie l'appel des subroutines de LAPACK, en utilisant de plus la généricité et les paramètres optionnels.

Leur utilisation en C rencontre plusieurs difficultés :

- l'inter-opération du C et du Fortran (types de variables, passage des paramètres, absence de généricité *etc*) ;
- l'absence de vrais tableaux 2-D au sens où ils existent en Fortran ;
- l'ordre de rangement des éléments dans les tableaux de tableaux (`A[i][j]`) diffère de celui du Fortran.

Le problème spécifique du C

La solution aux problèmes spécifiques du C passe par :

- la représentation des matrices dans leur forme aplatie (1-D) ;
- L'ajout, dans toutes les fonctions d'un paramètre `CblasRowMajor/RowMajor` ou `CblasColumnMajor/ColumnMajor` indiquant dans quel ordre sont rangés les éléments ;
- Une version de BLAS réécrite en C, `OpenBlas`, qui est de plus optimisée automatiquement à la compilation ;
- Une interface dédiée pour le langage C dénommée `lapacke` bien que moins puissante que celle du Fortran 95, elle décharge l'utilisateur de certaines tâches de bas niveau.

Dans le cas où on l'on utilise `RowMajor` – à éviter – `lapacke` se charge des transpositions nécessaires avec un algorithme optimisé.
Notons qu'il faut utiliser soit des tableaux dynamiques 1D, soit les tableaux automatiques du C99, pour que les éléments soient contigus en mémoire.

BLAS

BLAS est structuré en trois niveaux, selon la nature des opérations :

1. vectorielles du type : $y \leftarrow \alpha\, x + y$, ou $x \cdot y$, *etc*
2. matrices-vecteurs du type : $y \leftarrow \alpha\, \mathrm{A}.x + \beta\, y$ et les résolutions de systèmes ;
3. matrices-matrices du type : $\mathrm{C} \leftarrow \alpha \mathrm{A}.\mathrm{B} + \beta \mathrm{C}$, les inversions *etc*

Le nom des fonctions comme `sgemm` par exemple est formé par `S-GE-MM` où les lettres désignent :

1. le type de données : `S` = réel simple précision, `D` =double précision, `C` : complexe simple, `Z` complexe double
2. le type de matrice concerné : `GE`=general, `GB`=general band, `SY(SB/SP)`=symmetric (band/packed), `HE(HB/HP)`=hermitian (band/packed), `TR(TB/TP)`=triangular (band/packed)
3. La nature de l'opération : `AXPY`=somme de vecteurs, `MV`=produit matrice-vecteur, `MM`=produit de matrices, `SV/SM` résolution matrice-vecteur/matrice-matrice (`S` comme solve).

Exemple de multiplication avec GEMM en fortran 95

En fortran 77, l'interface de `SGEMM` comporte 13 paramètres :
`SGEMM(TRANSA,TRANSB,M,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC)`
réalise l'opération $\mathrm{C} \leftarrow \alpha\mathrm{A}.\mathrm{B} + \beta\mathrm{C}$, où

- `TRANSA` et `TRANSB` (des caractères prenant la valeurs `'N'` ou `'T'`) indiquent s'il faut travailler sur `A` et `B` ou leur transposée ;
- `M,N,K` sont les tailles des matrices `A(M,N)`, `B(N,K)`, `C(M,K)`,
- `LDA, LDB, LDC` sont aussi des tailles :
  LD=leading dimension=nbre de lignes car ColMajor en fortran

En fortran 95 avec `GEMM`, seuls les 3 paramètres `A`, `B` et `C` sont obligatoires.

```
use la_precision, only : WP=>SP          ! la precision
use blas95, only : gemm                  ! la fonction
integer, parameter :: m=..,n=.., p=..    ! les tailles
real(kind=WP) :: A(m,n), B(n,p), C(m,p) ! les matrices
! ... remplissage des matrices A et B ...
call GEMM(A, B, C)                       ! générique sans les étendues
! en F77 :  call sgemm('N','N',M,P,N,1.0,A,M,B,N,0.0,C,M)
```


Exemple de multiplication avec xGEMM

En C : on retrouve les difficultés habituelles, plus celles qui tiennent à ce que la bibliothèque est en Fortran, notamment :

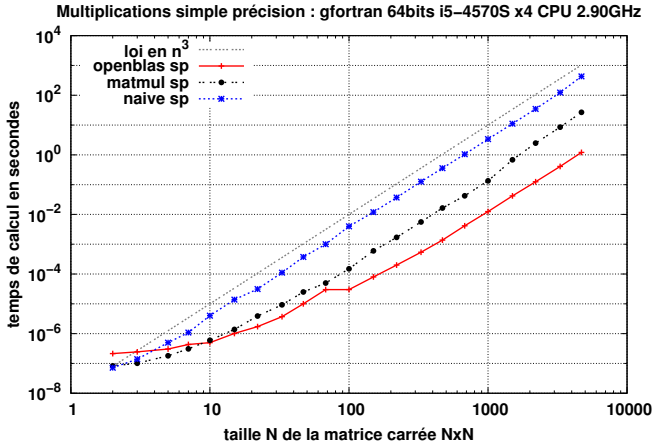
- Il faut utiliser la représentation aplatie,
- La « Leading Dimension » dépend de l'ordre choisi pour les éléments : ce n'est pas toujours le nombre de lignes, mais la taille de l'enregistrement de base dans la matrice aplatie. En `CblasRowMajor`, c'est donc la longueur d'une ligne, soit le nombre de colonnes !
  ⇒ Préférer `CblasColMajor` où Leading Dimension = nbre de lignes

En `CblasRowMajor`, déconseillé, on écrirait par exemple :

```
#include "cblas.h"                    // en RowMajor !
int M=.., N=.., P=..;                 // les tailles
float A[M*N], B[N*P], C[M*P];         // matrices aplaties
// ... remplissage des matrices A et B ...
cblas_sgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans,
            M, P, N, 1.0F, A, N, B, P, 0.0F, C, P);
```

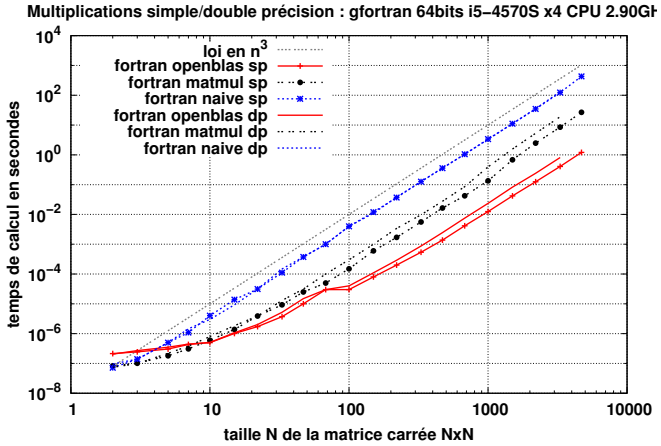
Gain de temps sur la multiplication avec MATMUL et GEMM

Temps de calcul mesuré sur un I5 (R) quad-core à 2.9 GHz sous linux 64 bits, pour le produit de 2 matrices aléatoires N×N en simple précision, via le produit « naïf », avec MATMUL et avec GEMM de openblas en fortran.



Gains simple et double précision avec MATMUL et xGEMM

Temps de calcul pour le produit de 2 matrices aléatoires N×N en simple et double précision, à l'aide du produit « naïf », avec MATMUL et avec GEMM de openblas en fortran. Le C donne les mêmes performances.



LAPACK : principe

LAPACK offre des procédures de plus haut niveau en combinant les possibilités offertes par BLAS.
Les fonctions de LAPACK sont elles-mêmes de 2 niveaux :

- les procédures de calcul (computational),
- et les procédures pilotes (driver), qui utilisent les précédentes.

Par exemple

- la procédure pilote `xGESV(N,NRHS,A,LDA,IPIV,B,LDB,INFO)` résout le système de `N` équations à `N` inconnues A.X = B, où l'inconnue X et le second membre B ont `NRHS` colonnes ;
- `xGESV` appelle les procédures de calcul `xGETRF` pour la factorisation LU de `A`, puis `xGETRS` pour résoudre le système triangulaire ;
- chacune des deux procédures `xGETRF` elles-mêmes, utilisent les procédures LAPACK `xGETF2` et `xLASWP` ainsi que la procédure BLAS `xTRSM`.

LAPACK : exemple de résolution de système

En Fortran :

```
use la_precision, only: WP=>SP     ! precision
use F95_LAPACK, only: LA_GESV      ! procedure
integer, parameter :: m=.., p=..   ! tailles
real(WP):: A(m,m), B(m,p)          ! matrice, seconds membres
integer :: info,  ipiv(m)          ! erreurs,permutations
! ... remplissage de A et B...
call LA_GESV(A, B, INFO=info, IPIV=ipiv)
```

En C :

```
#include "lapacke.h"
int m=..,  p=.. ;                  // les tailles
float A[m*m], B[m*p] ;             // matrice, seconds membres
int info, ipiv[m];                 // erreurs, permutations
// ... remplissage de A et B ...
info = LAPACKE_sgesv(CblasRowMajor, M, P, A, M, ipiv, B, M);
```

Attention : `A` remplacée par sa décomposition LU et `B` par les solutions !

Conclusion : intérêt des bibliothèques

L'utilisation des bibliothèques permet :

- d'écrire du code compact en s'appuyant sur des méthodes générales éprouvées, et de bénéficier d'une optimisation logicielle et matérielle maximale, pour se concentrer sur ce qui est spécifique,
- de choisir parmi tout un arsenal la méthode la plus adaptée, notamment selon le type de la matrice (bandes, triangulaires, symétriques ou hermitiennes) : nous n'avons utilisé dans les exemples que les procédures générales `xGEyy`, mais selon le problème considéré on pourra faire un choix optimal.

Ainsi, pour diagonaliser la matrice L_{1D} du TE, on peut choisir :
- soit la méthode générale `xGEEV` ,
- soit `xSYEV` qui exploite son caractère symétrique,
- soit `xSTEV` utilisant son caractère symétrique et tridiagonal.

Optimisation et ajustement

La problématique de l'optimisation est une thématique transversale des mathématiques, qui trouve des applications dans toutes les disciplines, en sciences comme en économie, ou en recherche opérationnelle. Dans ce domaine, il peut s'agir de tenir compte des certaines contraintes techniques et certains coûts variables et/ou incompressibles (production, promotion, distribution) pour fixer un ensemble de paramètres afin maximiser le bénéfice sur divers produits.

Un exemple usuel est celui d'une raffinerie de pétrole qui doit optimiser sa production de différentes fractions en fonction :

- de l'approvisionnement et de la composition pétroles bruts disponibles
- de la demande et du prix de vente des différents sous-produits
- des contraintes techniques de fonctionnement (chimie, stockage)

Dans ce cours nous nous limiterons au problème physique des ajustements de données expérimentales par des fonctions appelées « modèle », et utilisant les méthodes de l'algèbre linéaire.

Position du problème : ajustement

Exemple : On dispose d'un ensemble de N points « expérimentaux »
$(x_i, y_i, \sigma_i)_{i=1,N}$.
qui ont été obtenus avec une précision limitée, caractérisée par la déviation standard σ_i (barre d'erreur sur y_i).
Mais les x_i supposés connus exactement

On cherche à vérifier une loi physique (ou autre) et à déterminer aussi bien que possible M paramètres $(p_j)_{j=1,M}$ qui caractérisent cette loi.

On suppose que les points devraient vérifier le modèle :

$$y(x) = f(x, p_1 \dots p_M) = f(x, \vec{p}) \ , \qquad (3)$$

où $\vec{p}$ représente le vecteur à M composantes des paramètres.

NB : La fonction modèle f est souvent notée y_{th}, voire y tout court.

Équations sur-déterminées

On pourrait penser que les points de mesure sont autant de déterminations plus ou moins exactes des paramètres $\vec{p}$ du modèle. Cela pose un problème de principe, dans la mesure où l'on a affaire à un nombre N d'équations en général très supérieur au nombre M de paramètres : les équations sont sur-déterminées, et le plus souvent incompatibles.

Par exemple dans le cas linéaire, on va devoir résoudre un système de N équations à M inconnues, de la forme :

$$\mathrm{A} \cdot \mathrm{p} = \mathrm{b} \quad \text{où A est une matrice rectangulaire } N \times M$$

qui n'a en général pas de solution.

La « moins mauvaise » solution est obtenue en cherchant non plus à annuler $\mathrm{Ap} - \mathrm{b}$, mais à minimiser sa norme $\|\mathrm{A} \cdot \mathrm{p} - \mathrm{b}\|$ (qui dépend de la loi), par des méthodes directes ou itératives selon la nature du problème.

Cette approche doit être justifiée par une analyse statistique du problème. Il s'agit d'estimer les paramètres à partir des mesures.

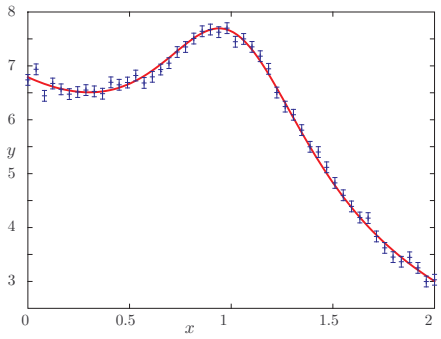
Déviations

Typiquement, on suppose qu'il existe un jeu de valeurs $\vec{p}^{\,v}$ des paramètres tel que :

$$y_i = f(x_i, \vec{p}^{\,v}) + \varepsilon_i \qquad (4)$$

où les déviations ε_i résultent

- de bruit sur les signaux,
- d'un léger écart au modèle (effets non pris en compte),
- des incertitudes introduites dans le processus de mesure.



NB : Les abscisses de mesure x_i sont supposées connues parfaitement.

Probabilité conditionnelle et vraisemblance

Hypothèses : déviations ε_i sur les y_i = variables aléatoires

- indépendantes
- gaussiennes centrées, d'écart-type σ_i.

Probabilité conditionnelle des mesures y_i connaissant les paramètres :

$$\mathcal{P}(y_1, \cdots y_N | \vec{p}) = \prod_{i=1}^{N} \mathcal{P}(y_i | \vec{p}) \propto \exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i - f(x_i, \vec{p})}{\sigma_i}\right)^2\right].$$

Probabilité conditionnelle des paramètres connaissant les mesures = fonction de vraisemblance (*likelihood*) : $\mathcal{L}(\vec{p} | y_1, \cdots y_N)$

Problème d'estimation des paramètres $\vec{p}$ au vu des mesures :
$\Rightarrow$ choisir l'estimateur du maximum de vraisemblance

$$\text{Max}\left[\mathcal{L}(\vec{p} | y_1, \cdots y_N)\right]$$

Vraisemblance maximale et erreur quadratique minimale

On admet que

$$\text{Max}\left[\mathcal{L}(\vec{p} | y_1, \cdots y_N)\right] \Longleftrightarrow \text{Max}\left[\mathcal{P}(y_1, \cdots y_N | \vec{p})\right]$$

Justifié lorsque la fonction f est linéaire par rapport aux variables $\vec{p}$, (et il n'est alors pas nécessaire que les ε_i suivent une loi normale).

Si on considère une distribution a priori des paramètres uniforme, le théorème de Bayes justifie cette identification :

$$\mathcal{P}(\vec{p} | y_1, \cdots y_N) = \mathcal{P}(y_1, \cdots y_N | \vec{p}) \times \mathcal{P}(\vec{p}) / \mathcal{P}(y_1, \cdots y_N)$$

$\Rightarrow$ Minimiser l'erreur quadratique moyenne pondérée :

$$\text{Min}\left[\sum_{i=1}^{N}\left(\frac{y_i - f(x_i, \vec{p})}{\sigma_i}\right)^2\right].$$

Moindres carrés pondérés

Les meilleures valeurs de $\vec{p}$, notées $\vec{p}^{\,a}$, maximisent la vraisemblance $\mathcal{L}$, soit encore minimisent l'erreur quadratique pondérée totale :

$$\mathcal{E}(\vec{p}) = \sum_{i=1}^{N}\left(\frac{\varepsilon_i}{\sigma_i}\right)^2 = \sum_{i=1}^{N}\left(\frac{y_i - f(x_i, \vec{p})}{\sigma_i}\right)^2. \qquad (5)$$

- Dans l'hypothèse où les déviations sont des variables normales centrées, elle a la structure d'un χ^2.
- Elle est très souvent abusivement appelée « chi2 » (χ^2), alors qu'elle n'y correspond – au mieux – qu'au point $\vec{p}^{\,a}$ pour lequel on suppose que la valeur moyenne des déviations normalisées est nulle.

Équations normales

Le minimum recherché $\vec{p}^a$ est un point où le gradient de $\mathcal{E}$ s'annule.
Par dérivation, on en déduit les M équations suivantes : $(k = 1 \dots M)$:

$$\sum_{i=1}^{N} \left(\frac{y_i - f(x_i, \vec{p}^a)}{\sigma_i^2} \right) \frac{\partial f(x_i, \vec{p}^a)}{\partial p_k} = 0 \qquad (6)$$

appelées équations normales

La résolution de ces équations donne les valeurs de $\vec{p}^a$ qui,
si le modèle est valide,
rendent le mieux compte des données, et sont susceptibles de constituer une détermination de la « vraie » valeur $\vec{p}^v$.

On parle alors d'« ajustement » des données par les moindres carrés pondérés ou en anglais "weighted least squares fit" ou fit tout court.

Linéaire vs non-linéaire

Deux situations doivent être distinguées :

1. Le cas particulier où la fonction $f(x, \vec{p})$ est linéaire vis à vis des paramètres p_j. Alors les dérivées $\frac{\partial f(x_i, \vec{p}^a)}{\partial p_k}$ sont des constantes (par rapport à $\vec{p}$)
   et le système des équations normales (6) est linéaire et peut être résolu simplement.
2. Le cas général où $f(x, p_1 \dots p_M)$ est une fonction non-linéaire des paramètres p_j. Le système (6) est non-linéaire, et il faut trouver une autre méthode,
   qui sera fondamentalement une méthode itérative.

Régression linéaire : équations normales

Le cas le plus simple d'ajustement et le plus important est celui d'un modèle linéaire à la fois vis à vis des $(M = 2)$ paramètres et des données :

$$f(x, \vec{p}) = a + b\,x \qquad \text{où} \qquad \vec{p} = (a, b) \qquad (7)$$

Alors

$$\cancel{\chi^2(\vec{p})} = \mathcal{E}(\vec{p}) = \sum_{i=1}^{N} \left(\frac{y_i - a - b x_i}{\sigma_i} \right)^2 \qquad (8)$$

$$\text{Min}\,[\mathcal{E}] \Rightarrow \begin{cases} \dfrac{\partial \mathcal{E}}{\partial a} = 0 \Rightarrow \displaystyle\sum_{i=1}^{N} \dfrac{y_i - a - b\,x_i}{\sigma_i^2} = 0 \\ \dfrac{\partial \mathcal{E}}{\partial b} = 0 \Rightarrow \displaystyle\sum_{i=1}^{N} x_i \dfrac{y_i - a - b\,x_i}{\sigma_i^2} = 0 \end{cases} \qquad (9)$$

Régression linéaire : résolution

En développant, on définit les 6 sommes

$$S = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \qquad S_x = \sum_{i=1}^{N} \frac{x_i}{\sigma_i^2} \qquad S_y = \sum_{i=1}^{N} \frac{y_i}{\sigma_i^2}$$
$$S_{xy} = \sum_{i=1}^{N} \frac{x_i y_i}{\sigma_i^2} \qquad S_{xx} = \sum_{i=1}^{N} \frac{x_i^2}{\sigma_i^2} \qquad S_{yy} = \sum_{i=1}^{N} \frac{y_i^2}{\sigma_i^2}$$

qui, à S près, sont des moyennes, variances et covariances.

$$\text{alors} \quad \begin{cases} a S + b S_x = S_y \\ a S_x + b S_{xx} = S_{xy} \end{cases} \quad \text{d'où} \quad \begin{aligned} a &= \frac{S_{xx} S_y - S_x S_{xy}}{\Delta} \\ b &= \frac{S S_{xy} - S_x S_y}{\Delta} \end{aligned} \qquad (10)$$

où $\Delta = S\,S_{xx} - (S_x)^2$ est le déterminant du système
et Δ / S^2 est la « variance » des x_i (déterministes !).

Qualité de la régression : variances/covariances de a et b

Les déviations obtenues pour les valeurs qui minimisent l'erreur sont appelées les « résidus ». La somme pondérée de leurs carrés est le χ^2 à (plus) proprement parler, à $N - M = N - 2$ degrés de liberté. On évalue les variances et covariances de a et b (démonstration plus bas) :

$$\mathrm{Var}(a) = \frac{S_{xx}}{\Delta} \qquad \mathrm{Var}(b) = \frac{S}{\Delta} \qquad \mathrm{Cov}(a,b) = -\frac{S_x}{\Delta} \ . \tag{11}$$

La covariance de a et b est généralement rapportée à la variance de a et de b pour définir le coefficient de corrélation de a et b :

$$R_{ab} = \frac{\mathrm{Cov}(a,b)}{\sigma_a\,\sigma_b} = -\frac{S_x}{\sqrt{S\,S_{xx}}}$$

$|R_{ab}| \leqslant 1$ doit être aussi petit que possible
$R_{ab} = \pm 1$ annulent Δ et font que a et b ne sont plus définis.

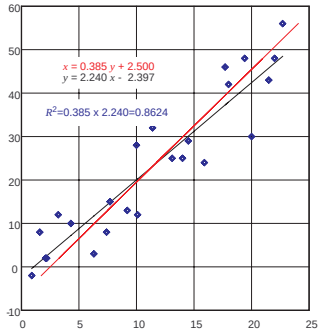
Qualité de la régression : coefficient de corrélation linéaire

La qualité de la régression est souvent évaluée en écrivant aussi le modèle :

$$x = a' + b'\,y$$

Moyennant certaines hypothèses simplificatrices sur les σ_i, on obtient :

$$b' = \frac{S\,S_{xy} - S_x\,S_y}{S\,S_{yy} - S_y^2} \ .$$



Les deux droites sont confondues si et seulement si $b \times b' = 1$, c'est à dire si le coefficient de corrélation linéaire :

$$r^2 = b \times b' = [\mathrm{Cov}(x,y)]^2 \,/(\sigma_x^2\,\sigma_y^2) = 1$$

La corrélation linéaire est donc appréciée qualitativement selon que r^2 est plus ou moins proche de ± 1.

Cas linéaire général

Un modèle $y = f(x, \vec{p})$ qui dépend linéairement *des paramètres* $\vec{p}$ est nécessairement de la forme :

$$f(x,\vec{p}) = \sum_{k=1}^{M} p_k\,F_k(x) \ , \tag{12}$$

où $F_1(x) \cdots F_M(x)$ sont M fonctions quelconques de x.

La minimisation de $$\mathcal{E}(\vec{p}) = \sum_{i=1}^{N} \left(\frac{y_i - \sum_{k=1}^{M} p_k\,F_k(x_i)}{\sigma_i} \right)^2 \tag{13}$$

donne les M équations normales :

$$\sum_{i=1}^{N} \frac{y_i - \sum_{j=1}^{M} p_j\,F_j(x_i)}{\sigma_i} \times \frac{F_k(x_i)}{\sigma_i} = 0 \text{ pour } k = 1, \cdots M. \tag{14}$$

Équations normales linéaires

En développant comme dans le cas 2-D, on obtient sous forme matricielle :

$$\sum_{j=1}^{M} \mathsf{A}_{kj} p_j = \mathsf{b}_k \quad \text{soit} \quad \mathsf{A} \cdot \mathsf{p} = \mathsf{b} \tag{15}$$

où les coefficients A_{kj} de la matrice $M \times M$ symétrique A et les composantes du vecteur-colonne b sont :

$$\mathsf{A}_{kj} = \sum_{i=1}^{N} \frac{F_k(x_i)\,F_j(x_i)}{\sigma_i^2} \quad \text{et} \quad \mathsf{b}_k = \sum_{i=1}^{N} \frac{y_i\,F_k(x_i)}{\sigma_i^2} \tag{16}$$

Le vecteur des paramètres p peut alors être obtenu avec l'une des méthodes standard de résolution des systèmes linéaires, sous la forme :

$$\mathsf{p} = \mathsf{A}^{-1} \cdot \mathsf{b}$$

Structure de la matrice A et du vecteur b

A_{ij} et b_i peuvent se calculer à partir de $G_{ik} = F_i(x_k)/\sigma_k$ et $Y_k = y_k/\sigma_k$:

$$\begin{cases} A_{ij} = \sum_{k=1,N} G_{ik} G_{jk} \\ b_i = \sum_{k=1,N} G_{ik} Y_k \end{cases} \quad \text{soit} \quad \begin{cases} A = G\,{}^tG \\ b = G\,Y \end{cases} \quad \text{donc} \quad p = \left(G\,{}^tG\right)^{-1} \cdot G \cdot Y$$

de façon plus visuelle :

$$A = \underbrace{\begin{bmatrix} G_{11} & \cdots\cdots & G_{1N} \\ \vdots & & \vdots \\ G_{M1} & \cdots\cdots & G_{MN} \end{bmatrix}}_{\text{taille} N \gg M} \underbrace{\begin{bmatrix} G_{11} .. G_{M1} \\ \vdots \quad \vdots \\ \vdots \quad \vdots \\ \vdots \quad \vdots \\ G_{1N} .. G_{MN} \end{bmatrix}}_{\text{taille} M \ll N} \quad \text{et} \quad b = \underbrace{\begin{bmatrix} G_{11} & \cdots\cdots & G_{1N} \\ \vdots & & \vdots \\ G_{M1} & \cdots\cdots & G_{MN} \end{bmatrix}}_{\text{taille} N \gg M} \underbrace{\begin{bmatrix} Y_1 \\ \vdots \\ \vdots \\ \vdots \\ Y_N \end{bmatrix}}_{\text{taille} 1}$$

NB : La matrice A ($M \times M$) est symétrique et définie positive (somme des carrés des ε_i), ce qui assure l'existence d'un minimum unique.

Variances-covariances : méthode probabiliste

Une fois obtenue la solution, il faut estimer sa qualité. Une première chose est de voir à quel point elle est contrainte, en évaluant la co-variance des paramètres : c'est la matrice $C = A^{-1}$ qui les donne.

En effet, de $p = C\,G\,Y \Rightarrow p_i = \sum_{m,k} C_{im} G_{mk} Y_k$ on tire :

$$\Rightarrow \mathrm{Cov}(p_i, p_j) = \sum_{m,n=1}^{M} \sum_{k,l=1}^{N} C_{im} G_{mk} C_{jn} G_{nl} \underbrace{\mathrm{Cov}(Y_k, Y_l)}_{\delta_{kl}}$$

$$= \sum_{m,n} C_{im} C_{jn} \underbrace{\sum_k G_{nk} G_{mk}}_{A_{mn}} = [C\,A\,C]_{ij} = C_{ij} \ ,$$

en utilisant successivement l'hypothèse d'indépendance des déviations $\{\varepsilon_k\}$ et la symétrie de A et donc de C.

Ce résultat s'appliquera de la même façon dans le cas non linéaire.

Variances-covariances : méthode algébrique

On peut revenir à l'expression de la vraisemblance $\mathcal{L}$ autour du point $\vec{p}^{\,a}$: en notant $\delta\vec{p} = \vec{p} - \vec{p}^{\,a}$, on a :

$$\mathcal{L}(\delta\vec{p}) \propto \exp\left(-\tfrac{1}{2}\ {}^t\delta p \cdot A \cdot \delta p\right)$$

qui décroît d'autant plus vite que A est grand, correspondant à un minimum très piqué

On diagonalise A sous la forme $A = {}^tU \cdot D \cdot U$ (où U est orthogonale, et D est diagonale), et on obtient des paramètres q_i, tels que $\vec{q} = U\delta\vec{p}$, qui sont indépendants et de moyenne nulle. On a alors :

$$\mathcal{L} \propto \prod_{i=1}^{M} \exp\left(-\tfrac{1}{2} D_{ii}\, q_i^2\right) \qquad \Rightarrow \qquad \mathrm{Var}(q_i) = 1/D_{ii} \ .$$

En revenant aux paramètres p_i, et en utilisant l'indépendance des q_i, on obtient :

$$\langle \delta p_i\ \delta p_j \rangle = \left[{}^tU \cdot D^{-1} \cdot U\right]_{ij} = C_{ij}$$

qui est le résultat énoncé précédemment.

Cas non-linéaire : approche générale

- Le cas non-linéaire est le cas général : penser à titre d'exemple à une lorentzienne ou une gaussienne dont la largeur w est un paramètre de l'ajustement, ou à la fréquence d'une sinusoïde.
- Le problème vient de ce que dans les équations normales :

$$\sum_{i=1}^{N} \left(\frac{y_i - f(x_i, \vec{p}^{\,a})}{\sigma_i^2}\right) \frac{\partial f(x_i, \vec{p}^{\,a})}{\partial p_k} = 0$$

les dérivées partielles $\partial f(x_i, \vec{p})/\partial p_k$ dépendent du point $\vec{p}$ considéré dans l'espace des paramètres.

- Les différentes méthodes existantes sont des méthodes itératives qui diminuent progressivement l'erreur quadratique pondérée $\mathcal{E}(\vec{p})$ jusqu'à obtenir un minimum local.

Méthode itérative : principe

On se donne la valeur initiale $\vec{p}^{(0)}$ ("guess" en anglais) et on détermine successivement $\vec{p}^{(1)}, \cdots \vec{p}^{(n)}$ jusque qu'à ce que $\mathcal{E}(\vec{p}^{(n)})$ cesse de diminuer. Il faut définir un critère de convergence comme :

$$0 \leqslant \mathcal{E}\left(\vec{p}^{n}\right) - \mathcal{E}(\vec{p}^{n+1}) \leqslant \eta_{\text{tol.}} \quad \text{où} \quad \eta_{\text{tol.}} \ll \mathcal{E}\left(\vec{p}^{(n)}\right) . \qquad (17)$$

Trois difficultés se présentent :

- On peut converger vers un minimum local dénué de sens : il faut choisir une valeur initiale $\vec{p}^{(0)}$ pertinente, et une méthode bien adaptée s'il y a beaucoup de minima locaux possibles.
- La surface $\mathcal{E}(\vec{p})$ peut présenter des vallées dans lesquelles on descend très vite, et où ensuite la progression est très lente (méandres).
- La méthode itérative peut nécessiter un nombre très élevé d'évaluations de la fonction f et de ses dérivées, très coûteuses en temps de calcul si N est important.

Cela est d'autant plus probable que le nombre M de paramètres est élevé.
On procède alors par étapes en faisant varier certains paramètres

La méthode du gradient

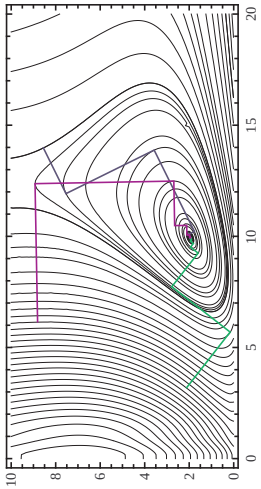
La méthode la plus simple consiste à faire un déplacement $\delta\vec{p} = \vec{p}^{n+1} - \vec{p}^{n}$ dans la direction du gradient de $\mathcal{E}$, qui définit la plus grande pente ("steepest descent" en anglais) sur la « surface » représentée par $\mathcal{E}(\vec{p})$:

$$\delta\vec{p} = -\gamma\,\nabla\mathcal{E}\left(\vec{p}^{n}\right) \qquad (\gamma \in \mathbb{R}^{+}) ,$$

où le gradient est défini par ses composantes $\partial\mathcal{E}/\partial p_k$ qui sont les termes qui figurent à gauche dans les équations normales (à un facteur -2 près).

Méthode assez robuste si le pas est assez petit, mais dans ce cas elle converge très lentement.

Problème : comment choisir γ ?

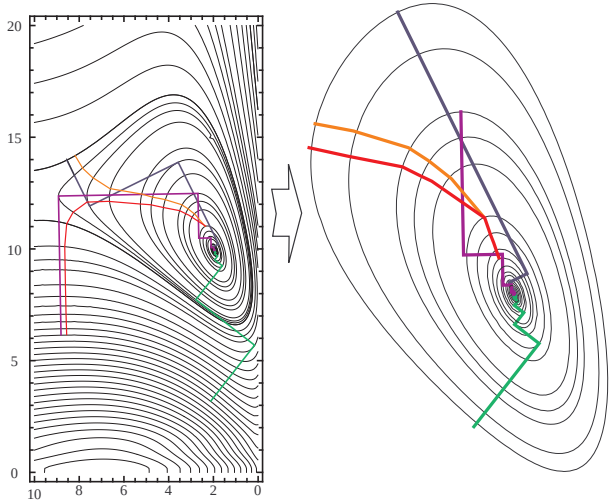


Gradient : le choix du pas

Ni la minimisation en ligne (direction fixe tant qu'on descend donc cheminement avec angles droits)

ni la méthode à petits pas (changements de direction faibles)

ne sont efficaces.



La matrice hessienne

Pour aller plus vite, c'est-à-dire plus directement vers le minimum, il est utile de considérer la courbure locale de la surface, afin de mieux choisir la direction de $\delta\vec{p}$. Celle-ci est accessible via la matrice de Hess ou hessienne des dérivées secondes de $\mathcal{E}(\vec{p})$:

$$\begin{aligned}[\mathsf{H}(\vec{p})]_{kl} &= \frac{\partial^2\mathcal{E}(\vec{p})}{\partial p_k \partial p_l} \\ &= 2\sum_{i=N}^{M}\frac{1}{\sigma_i^2}\left(\frac{\partial f(x_i,\vec{p})}{\partial p_k}\frac{\partial f(x_i,\vec{p})}{\partial p_l} - (y_i - f(x_i,\vec{p}))\,\frac{\partial^2 f(x_i,\vec{p})}{\partial p_k \partial p_l}\right)\end{aligned}$$

C'est son anisotropie qui est le principal responsable de la lenteur de convergence de la méthode du gradient.

Prise en compte de la courbure

En faisant un développement limité de $\mathcal{E}(\vec{p})$ autour de $\vec{p}^{(n)}$:

$$\mathcal{E}(\vec{p}) \approx \mathcal{E}(\vec{p}^{(n)}) + \nabla\mathcal{E}(\vec{p}^{(n)}) \cdot \delta\vec{p} + \frac{1}{2}\, {}^t\delta\vec{p} \cdot \mathsf{H} \cdot \delta\vec{p} + \cdots , \qquad (18)$$

En dérivant, le gradient au point $\vec{p}^{(n+1)} = \vec{p}^{(n)} + \delta\vec{p}$ s'écrit :

$$\nabla\mathcal{E}(\vec{p}^{(n+1)}) = \nabla\mathcal{E}(\vec{p}^{(n)}) + \mathsf{H} \cdot \delta\vec{p}$$

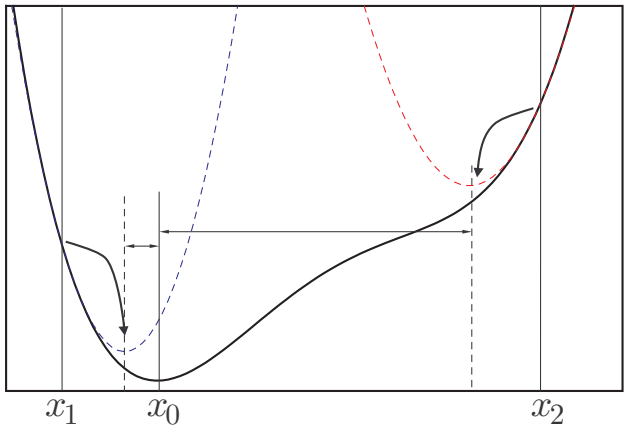
Pour qu'il s'annule, il faut choisir $\delta\vec{p}$ tel que :

$$\nabla\mathcal{E}(\vec{p}^{(n)}) + \mathsf{H} \cdot \delta\vec{p} = 0 \quad \Leftrightarrow \quad \delta\vec{p} = -\mathsf{H}^{-1} \cdot \nabla\mathcal{E}(\vec{p}^{(n)}) . \qquad (19)$$

Si $\mathcal{E}$ était effectivement quadratique, on obtiendrait directement la solution : c'est en fait le cas linéaire, où la matrice H se confond alors avec la matrice 2A, et $\nabla\mathcal{E}$ avec $-$b qui sont indépendantes du point $\vec{p}$ considéré.

Cette solution est bien plus rapide lorsqu'on est assez proche d'un minimum, (*i.e.* au fond de la vallée).

Illustration à une dimension

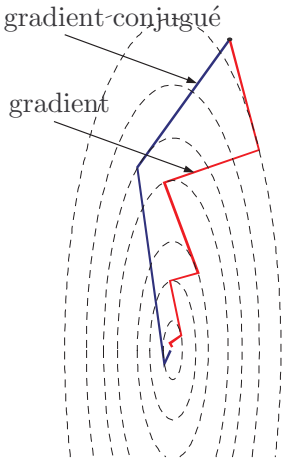


La méthode du gradient conjugué

Elle consiste à faire des « pas » ou incréments $\Delta\vec{p}$ dans la direction du $\delta\vec{p}$ tel que défini par l'équation (19).
Elle est ainsi nommée car la surface de niveau et l'incrément sont « conjugués », *i.e.* ne sont pas orthogonaux pour le produit scalaire normal, mais pour la forme quadratique H, soit :

$$\vec{u} \cdot \nabla\mathcal{E} = 0 \quad \Rightarrow \quad {}^t\delta\vec{p}\ \mathsf{H}\ \vec{u} = 0 .$$

Elle a le mérite de progresser dans la « bonne » direction, et conduit à une convergence rapide si l'approximation quadratique (18) est adaptée.



Comparaison des méthodes du gradient et du gradient conjugué

Taille des pas et choix de la méthode

La courbure nous fournit une réponse à la question du choix de la taille des pas : pour des raisons de dimension et d'ordre de grandeur, dans la méthode du gradient, il est souhaitable de prendre un facteur γ_i différent dans chaque direction i, et $\gamma_i \approx \alpha/\mathsf{H}_{ii}$ est un choix raisonnable, avec $\alpha < 1$ si on veut limiter la taille des « pas ».
Dans ces conditions :

- La méthode du gradient est plus sûre quand on est loin de l'optimum car l'approximation quadratique est alors grossière.
- La méthode du gradient conjugué est plus rapide lorsqu'on s'approche du minimum, car l'approximation quadratique est alors bien meilleure.

NB : On peut donc chercher à combiner les deux pour descendre à coup sûr dans la vallée, et ensuite y progresser rapidement.
C'est l'algorithme de **Levenberg-Marquardt**, utilisé dans la pratique, avec un paramètre permettant de passer continûement d'une méthode à l'autre.

Méthode du gradient conjugué : les équations

- Si les déviations $\epsilon_k = y_k - f(x_k, \vec{p})$ sont petites, indépendantes et dispersées selon une loi normale centrée, la contribution à H des termes croisés en $\dfrac{\partial^2 f}{\partial p_k \partial p_l}$, $k \neq l$ est négligeable.
- On définit alors la matrice A et le vecteur B comme suit :

$$A = G\,{}^tG\,, \quad b = G\,E \quad \text{où} \quad G_{ik} = \frac{1}{\sigma_k}\frac{\partial f(x_k, \vec{p})}{\partial p_i} \text{ et } E_k = \frac{\epsilon_k}{\sigma_k} \qquad (20)$$

- On souligne dans cette approche le parallélisme étroit avec la méthode appliquée dans le cas linéaire. Simplement A et b sont des fonctions du point considéré, mais tendront assez vite vers les valeurs qu'ils prennent au minimum.

Méthode du gradient conjugué : l'algorithme

Les itérations sont alors effectuées comme suit :

1. Choisir une valeur *initiale* $\vec{p}^{(0)}$ et évaluer $\mathcal{E}(\vec{p}^{(0)})$.
2. Au point $\vec{p}^{(n)}$, évaluer A, b et résoudre l'équation A δp = b.
3. Prendre $\vec{p}^{(n+1)} = \vec{p}^{(n)} + \delta\vec{p}$.
4. Tester la convergence : si $\mathcal{E}$ ne diminue plus, c'est à dire si

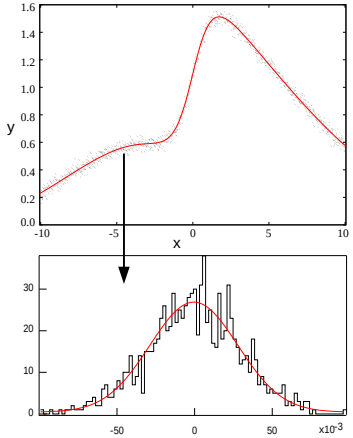
$$0 < \mathcal{E}(\vec{p}^{(n)}) - \mathcal{E}(\vec{p}^{(n+1)}) < \eta \ll 1,$$

on peut arrêter les itérations. Sinon, retourner en (2).

Variante :
On peut rendre la méthode plus robuste en remplaçant $\delta\vec{p}$ par $\gamma\delta\vec{p}$ avec $0 < \gamma < 1$. Pour optimiser la convergence, il peut être utile de contrôler la valeur de γ en fonction de l'évolution de $\mathcal{E}$ à chaque itération. Mais ce raffinement dépasse le cadre du travail proposé en TE.

Critères d'évaluation de la qualité de l'ajustement

- La matrice des variances-covariances est l'inverse de matrice A. Elle rend compte de ce que l'ajustement est plus ou moins contraint. Les variances qu'elle donne sont souvent très faibles et non-significatives. Attention aux covariances $\equiv$ choix des paramètres.
- La dépendance en x et la distribution statistique des résidus renseignent sur la vraisemblance du modèle et la qualité de l'ajustement.



La pertinence du modèle est enfin appréciée par le test du χ^2 (erreur évaluée au minimum !), avec $N - M$ degrés de liberté, dont la fonction de répartition est proportionnelle à la fonction Γ incomplète.

Test du χ^2

On calcule alors :

$$Q\left(\tfrac{N-M}{2}, \tfrac{\chi^2}{2}\right) = 1 - \frac{\Gamma\left(\tfrac{N-M}{2}, \tfrac{\chi^2}{2}\right)}{\Gamma\left(\tfrac{N-M}{2}\right)}\,,$$

qui mesure la probabilité que des résidus, indépendants et distribués selon une loi normale centrée d'écart type σ_i, donnent une valeur supérieure ou égale à la valeur χ^2 obtenue. Les seuils à utiliser dépendent du risque que l'on accepte. Typiquement une valeur $Q \geqslant 0.1$ suggère de faire confiance au modèle, puisque on aurait une chance d'erreur sur 10 en le rejetant. Une valeur de $Q \sim 0.01$ est juste passable puisque cette chance d'erreur est réduite à 1%.
Observons toutefois qu'une valeur trop grande de χ^2, et donc trop faible de Q, provient très souvent de dispersions σ_i sous-évaluées, car on est fréquemment trop optimiste sur ces « barres d'erreur ». Ce peut alors être l'expérimentateur qu'il faut mettre en cause, et non le modèle !