Chapitre III

Méthodes numériques de base

Le but de ce chapitre est de fournir, de manière succincte, les principes de base des méthodes numériques les plus utilisées. Même si désormais le premier contact avec les méthodes numériques ne se fait plus au travers d'une programmation de ces méthodes dans les langages de bas-niveau, mais plutôt par l'utilisation d'outils tels que `Matlab, Scilab, Maple, Mathematica...` permettant la manipulation de concepts mathématiques évolués, il reste indispensable de connaître les fondements des principales méthodes pour les utiliser de façon pertinente.

1 Résolution des équations du type $f(x) = 0$

Nous nous restreignons, par souci de simplicité, à la recherche de racines réelles, zéros de fonctions réelles continues.

L'existence et une première localisation des solutions utilisent le théorème des valeurs intermédiaires.

Théorème 1.1 (Théorème des valeurs intermédiaires) **:** *Si f est une fonction continue sur $[a, b]$, et si $f(a)f(b) \leq 0$, alors il existe au moins un point $c \in [a, b]$ tel que $f(c) = 0$.*

Si de plus f est strictement monotone sur $[a, b]$, la racine est unique dans $[a, b]$.

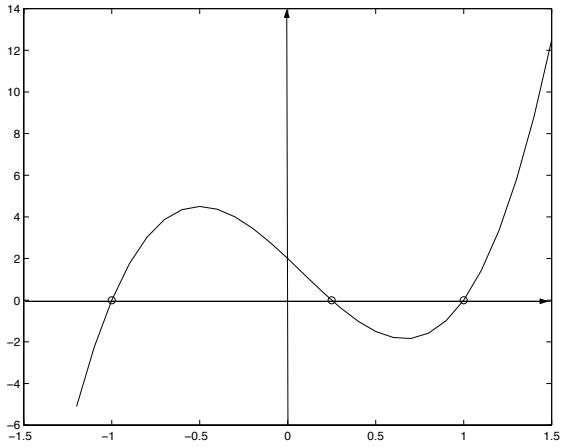

FIGURE III.1 – Fonction présentant 3 racines.

1.1 Méthode de dichotomie

La méthode de dichotomie est un procédé systématique de raffinement de la localisation d'une racine. Le mot dichotomie (dicho = deux, tomie = coupe) exprime clairement le principe de la méthode.

Soit $[a, b]$ un intervalle initial de localisation de la racine cherchée s. Supposons que l'on ait $f(a)f(b) < 0$, l'algorithme de la dichotomie s'écrit :

```
Tant que( abs( b-a ) > epsilon)  ! test d'arret
    m=(a+b)/2
    si (f(a) * f(m)) < )
        b=m             ! s est dans  [a,m]
    sinon
        a = m           ! s est dans  [m,b]
    Fin si
Fin tant que
```

Cet algorithme réduit à chaque pas l'amplitude de la localisation d'un facteur 2. L'erreur est donc réduite d'un facteur 2 à chaque itération. En 20 itérations, par exemple l'erreur sera 10^{-6} fois l'erreur initiale. Cette méthode

est relativement lente. Par contre elle converge dans tous les cas où la fonction change de signe au voisinage de la racine (ce qui exclut les racines de multiplicités paires). C'est une méthode que nous qualifierons de méthode tout-terrain, lente mais quasiment infaillible.

1.2 Méthodes de point-fixe

Les méthodes de point-fixe permettent de construire des algorithmes plus rapides que la dichotomie (parfois) mais surtout des algorithmes qui se généralisent simplement au cas de problèmes en dimension supérieure à un. On ramène l'équation $f(x) = 0$ à une équation équivalente de forme point-fixe

$$x = g(x)$$

Ceci nous permettra d'obtenir simplement une méthode itérative de la forme

$$\begin{cases} x_0 \text{ donné :} & \text{initialisation} \\ x_{n+1} = g(x_n) \end{cases} \tag{III.1}$$

Si cette itération converge, elle converge vers le point-fixe de g, donc de

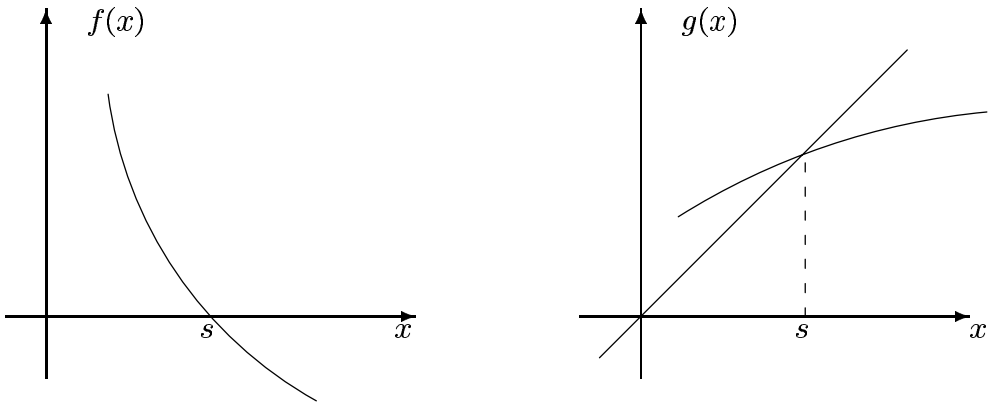

FIGURE III.2 – Forme $f(x) = 0$ et forme point-fixe équivalente d'une équation.

manière équivalente vers le zéro recherché de f. La condition de convergence essentielle est une condition de contraction sur la fonction g.

Déf. 1.1 (Application contractante) **:** *On dit qu'une application définie de* $[a, b]$ *dans* $[a, b]$ *est contractante, ou que c'est une contraction, s'il existe un nombre* $0 \leq k < 1$ *tel que, pour tout couple de points distincts* (x_1, x_2) *de* $[a, b]$*, on ait :*

$$|g(x_1) - g(x_2)| \leq k|x_1 - x_2|$$

k est le facteur de contraction. Il donne la vitesse de convergence de l'itération.

Dans le cas où g est dérivable, la condition de contraction se ramène à la condition suivante sur la dérivée : $|g'(x)| \leq k < 1 \quad \forall x \in [a, b]$

Remarque 1.1 : *Les notions de contraction et le théorème de convergence des itérations associé peuvent s'écrire et se démontrer dans le cadre général des espaces vectoriels normés (espaces de Banach). Cette possibilité de généralisation très large est un des intérêts principaux des méthodes de point-fixe (voir la démonstration générale du théorème* **??***).*

1.3 Vitesse de convergence et ordre d'une méthode itérative

Nous nous plaçons dans le cas d'une itération $x_{n+1} = g(x_n)$ convergente et nous supposons la fonction g suffisamment dérivable. L'application de la formule de Taylor au voisinage de la racine s donne :

$$x_{n+1} - s = g(x_n) - g(s) = (x_n - s)g'(s) + \frac{(x_n - s)^2}{2}g''(s) + O((x_n - s)^3)$$

Méthodes d'ordre un

Si $g'(s) \neq 0$, la limite du rapport des écarts est :

$$\lim_{n \to \infty} \frac{|x_{n+1} - s|}{|x_n - s|} = |g'(s)|$$

L'écart au pas $n+1$ est donc du même ordre que l'écart au pas n. Le facteur de réduction d'erreur est asymptotiquement donné par $|g'(s)|$. Plus petite sera la valeur de $|g'(s)|$, plus vite se fera la convergence.

Méthodes d'ordre deux

Si $g'(s) = 0$, l'erreur au pas $n+1$ est un infiniment petit d'ordre ≥ 2 par rapport à l'erreur au pas n. On obtient en effet :

$$\lim_{n\to\infty} \frac{|x_{n+1} - s|}{|x_n - s|^2} = \frac{1}{2}|g''(s)|$$

La convergence est dite quadratique. La réduction du nombre des itérations est spectaculaire dès l'ordre 2. À partir d'une erreur de 0.1, on obtient 10^{-8} en trois itérations.

On peut essayer, pour chaque problème particulier, de construire une itération de point-fixe convergente. Il est évidemment plus intéressant d'utiliser des méthodes générales applicables pour toute équation $f(x) = 0$. Voici une famille de méthodes classiques très utiles dans la pratique.

1.4 Méthode de Newton et Quasi-Newton

On obtient évidemment une forme point-fixe équivalente à $f(x) = 0$ en considérant la famille $x = x - \lambda(x)f(x)$, avec λ définie et non nulle sur un intervalle $[a, b]$ contenant la racine s. Parmi tous les choix possibles pour λ, le choix qui conduit à la convergence la plus rapide est $\lambda = \dfrac{1}{f'}$ (f' dérivée de f). La méthode obtenue ainsi est la méthode de Newton. Il est facile de vérifier que l'itération

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{III.2}$$

est d'ordre deux si elle converge. Évidemment la méthode de Newton n'est plus efficace si f' s'annulle, donc dans le cas de racines multiples. Il existe des résultats de convergence globale de Newton. Ils supposent des hypothèses de monotonie et de concavité de signe constant sur f (pas de point d'inflexion). La méthode de Newton est très classique. Elle s'interprète géométriquement comme méthode de la tangente.

La méthode de Newton nécessite le calcul des dérivées $f'(x_n)$. C'est un inconvénient dans la pratique où l'on ne dispose pas toujours d'expression analytique pour la fonction f.

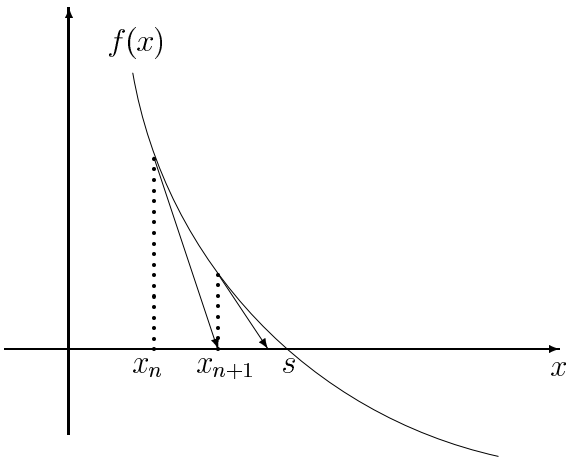

FIGURE III.3 – Interprétation géométrique de la méthode de Newton.

Une solution simple est fournie par la *méthode de la sécante ou de fausse-position* dans laquelle on remplace le calcul de $f'(x_n)$ par l'approximation

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

Ce qui donne

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \tag{III.3}$$

Les procédures de résolution d'équations de type $f(x) = 0$ que l'on trouve dans les outils génériques d'algorithmique (Matlab par exemple), combinent en général une première approche de la racine par quelques pas de dichotomie, suivis, pour la convergence fine, par une méthode rapide afin d'obtenir une grande précision en peu d'itérations. L'algorithme proposé par Matlab est dû à Dekker (1969). La méthode rapide utilisée est une interpolation quadratique inverse, assurant l'ordre deux de convergence (comme Newton), sans calcul de la dérivée de f.

1.5 Méthode de Newton et Quasi-Newton pour les systèmes

La méthode de Newton se généralise en dimension supérieure à un. On peut en effet montrer que le choix du $n+1^e$ itéré x_{n+1} est tel que

$$f(x_{n+1}) = O(|x_{n+1} - x_n|^2)$$

En effet un développement simple donne :

$$f(x_{n+1}) = f(x_n) + (x_{n+1} - x_n)f'(x_n) + O(|x_{n+1} - x_n|^2)$$

On voit donc que le choix

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

assure bien

$$f(x_{n+1}) = O(|x_{n+1} - x_n|^2)$$

On retrouve ainsi l'ordre 2 de la méthode de Newton.

Dans le cas d'un système de N équations non-linéaires, on peut écrire

$$F(X_{n+1}) = F(X_n) + F'(X_n)(X_{n+1} - X_n) + O(\|X_{n+1} - X_n\|^2)$$

la même idée conduit au choix suivant pour l'itération vectorielle :

$$X_{n+1} = X_n - \{F'(X_n)\}^{-1} F(X_n) \tag{III.4}$$

où $F'(X_n)$ désigne la matrice jacobienne de coefficients $\dfrac{\partial F_i}{\partial x_j}(X_N)$. Pratiquement le $n+1^e$ itéré est calculé à chaque itération par résolution de systèmes linéaires

$$F'(X_n)[X_{n+1} - X_n] = -F(X_n) \tag{III.5}$$

La matrice $F'(X_n)$ doit être assemblée et recalculée et le système doit être résolu à chaque itération. Ceci rend la méthode de Newton très coûteuse en temps de calcul.

Pour éviter ces inconvénients, on utilise des méthodes dites de *Quasi-Newton* dans lesquelles sont proposées des approximations de l'inverse de la matrice Jacobienne. Une méthode classique et efficace de ce type est la méthode BFGS (Broyden-Fletcher-Goldfarb-Shanno).

1.5.1 Application à l'optimisation

Dans un contexte d'optimisation, la recherche du minimum d'une fonctionnelle J peut être ramenée à la recherche du point qui annule son gradient $\nabla J(x)$, x désignant la paramétrisation du problème (voir chapitre **??**). On pourra alors utiliser les méthodes de type Newton ou Quasi-Newton (nous renvoyons à la littérature pour les méthodes d'optimisation en dimension un du type de la méthode de la section dorée qui ne nécessite pas le calcul des dérivées). Ces méthodes demandent le calcul exact ou approché de la matrice des dérivées partielles secondes ou matrice Hessienne. La méthode BFGS permet de construire directement une approximation de l'inverse de la Hessienne $\mathbf{H}$, en démarrant de la matrice identité ($\mathbf{H}_0 = Id$) et en appliquant l'itération suivante :

$$\mathbf{H}_{p+1} = \mathbf{H}_p + \left(1 + \frac{{\gamma_p}^{\mathrm{T}} \mathbf{H}_p \gamma_p}{\delta {x_p}^{\mathrm{T}} \gamma_p}\right) \frac{\delta x_p \delta {x_p}^{\mathrm{T}}}{\delta {x_p}^{\mathrm{T}} \gamma_p} - \frac{1}{2} \frac{\delta {x_p}^{T} (\mathbf{H}_p + (\mathbf{H}_p)^T) \gamma_p}{\delta x_p \gamma_p} \quad \text{(III.6)}$$

où p indique l'itération d'optimisation, $\delta x_p = x_p - x_{p-1}$ la variation du paramètre et $\gamma_p = \nabla J\left(x_{p+1}\right) - \nabla J\left(x_p\right)$. Voir chapitre **??** pour des développements sur l'optimisation.

2 Interpolation

Une collection de valeurs notées y_i étant données pour un ensemble d'abscisses x_i, pour $i = 0$ à n, l'interpolation est le procédé qui consiste à déterminer une fonction, d'une famille choisie a priori, prenant les valeurs données aux abscisses correspondantes. Le choix de fonctions polynomiales est le plus classique. Dans ce cas, le polynôme d'interpolation est le polynôme de degré minimal passant par les $n + 1$ points donnés. Ce polynôme est unique et il est de degré inférieur ou égal à n. Si l'on exprime le polynôme recherché dans la base canonique sous la forme

$$P(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$$

on doit résoudre un système linéaire de $n + 1$ équations à $n + 1$ inconnues. Sous cette forme le calcul est donc coûteux. La solution de ce système est très sensible aux erreurs d'arrondis.

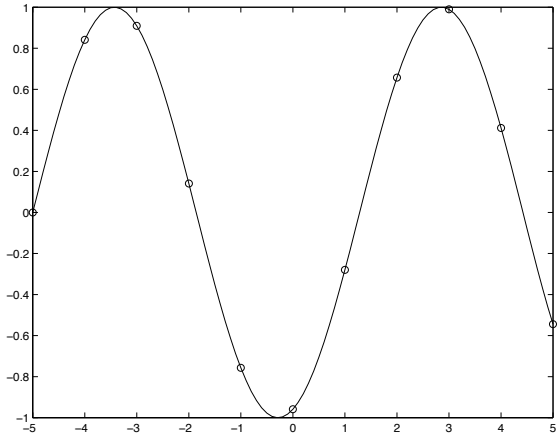

FIGURE III.4 – Interpolation polynomiale de degré 10 pour une fonction sinusoïde.

2.1 Polynômes de Lagrange

Une solution simple, élégante et économique de ce problème est fournie par l'utilisation de la base des polynômes de Lagrange.

On considère les $n+1$ polynômes L_i de degré $\leq n$ qui vérifient, pour tout i et j compris entre 0 et n, les égalités :

$$\begin{cases} L_i(x_i) = 1 \\ L_i(x_j) = 0 \end{cases} \tag{III.7}$$

Les polynômes L_i sont déterminés de façon unique par les $n+1$ équations ci-dessus. Il est facile de montrer qu'ils forment une base de l'espace des polynômes de degré inférieur ou égal à n et qu'ils s'écrivent :

$$L_i(x) = \prod_{j=0 j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \tag{III.8}$$

Exprimé dans cette nouvelle base, le polynôme d'interpolation s'écrit

$$P(x) = \sum_{i=0}^{n} y_i \, L_i(x) \tag{III.9}$$

La relation ci-dessus, facile à vérifier, explique l'intérêt de la base de Lagrange. Les coefficients du polynôme d'interpolation cherché sont, dans cette base, tout simplement les valeurs y_i données. Exprimé autrement, le changement de base, de la base canonique à la base de Lagrange, a transformé le système à résoudre en un système à matrice identité.

2.2 Limites de l'interpolation polynomiale

L'interpolation polynomiale est la base de nombreuses techniques numériques, en particulier les techniques d'intégration approchée. Elle se généralise de façon naturelle aux cas de dimension supérieure à un.

Cependant elle a des limites :

- théoriques : on n'est pas assuré de la convergence du polynôme d'interpolation vers la fonction interpolée lorsque l'on fait tendre le nombre de points d'interpolation (et donc le degré du polynôme) vers l'infini (voir le phénomène de Runge pour la fonction $f(x) = \frac{1}{1+x^2}$) ;
- numériques : même dans le cas où la convergence théorique est assurée, les instabilités de calcul provenant de l'accumulation des erreurs d'arrondis, auxquelles le procédé d'interpolation polynomiale est particulièrement sensible, limite l'usage de cette technique dès que le nombre de points d'interpolation dépasse la dizaine ;

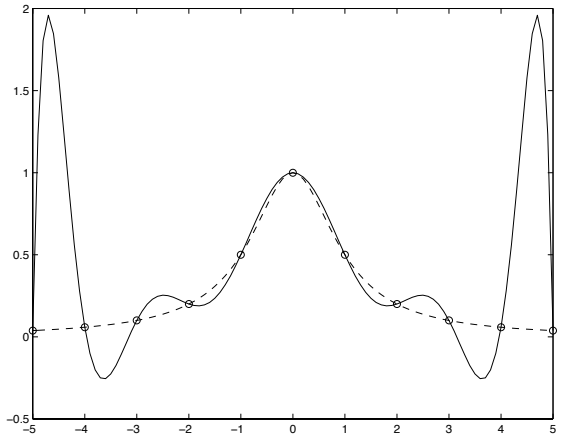

FIGURE III.5 – Divergence de l'interpolation polynomiale pour la fonction $y = \frac{1}{1+x^2}$. Phénomène de Runge. En pointillés : la fonction, en traits pleins : le polynôme d'interpolation de degré 10 construit sur 11 points régulièrement espacés.

– pratiques : remarquons que dans de nombreux cas, les valeurs données résultent d'expériences ou de calculs préalables. Ces valeurs sont donc approximatives. Le problème réel n'est alors plus un problème d'interpolation, mais plutôt un problème de meilleure approximation pour lequel les méthodes de moindres carrés, présentées plus bas, sont mieux adaptées.

2.3 Interpolation par des splines

Pour éviter l'inconvénient, signalé plus haut, de l'augmentation du degré du polynôme et de l'instabilité qui en résulte, lorsque le nombre de points est grand, tout en restant dans un procédé d'interpolation, on subdivise l'ensemble des points donnés en plusieurs sous-ensembles. On réalise les interpolations sur ces petits sous-ensembles, ce qui permet de se limiter à des polynômes de bas degré. Les fonctions polynomiales par morceaux obtenues sont à la base des éléments finis de Lagrange.

Les interpolations ci-dessus produisent des fonctions globalement continues mais non continûment dérivables.

Les *splines cubiques* d'interpolation sont des fonctions cubiques par mor-

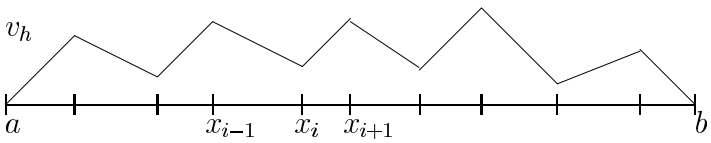

FIGURE III.6 – Une fonction affine par morceaux.

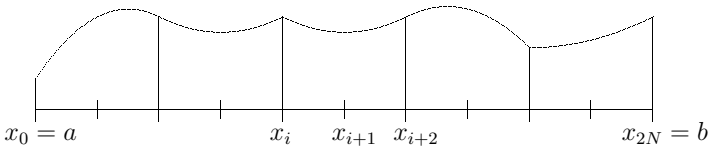

FIGURE III.7 – Une fonction polynomiale de degré deux par morceaux.

ceaux, globalement C^2. On obtient leur expression analytique, segment par segment, en imposant les conditions suivantes aux points x_i d'interpolation

$$s(x_i) = y_i \text{ donné pour } i = 0, ..n, \quad s' \text{ et } s'' \text{ continues}$$

Les inconnues du problème sont alors les dérivées secondes C_i de la spline

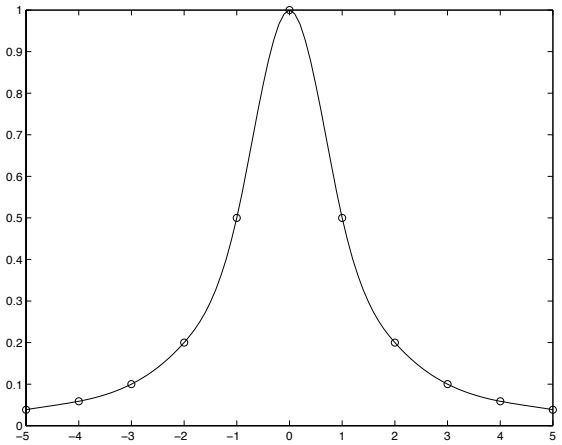

FIGURE III.8 – Spline cubique d'interpolation pour $y = \frac{1}{1+x^2}$.

aux points x_i. On suppose la dérivée seconde de la spline affine par intervalles. On intègre deux fois en prenant en compte les conditions de continuité de la dérivée et les valeurs données y_i aux points x_i. On en déduit les expressions suivantes de la spline sur chaque intervalle $[x_i, x_{i+1}]$:

$$S_i(x) = \frac{C_i}{6}[\frac{(x_{i+1}-x)^3}{h_i} - h_i(x_{i+1}-x)] + \frac{C_{i+1}}{6}[\frac{(x-x_i)^3}{h_i} - h_i(x-x_i)] + y_i\frac{(x_{i+1}-x)}{h_i} + y_{i+1}\frac{(x-x_i)}{h_i} \quad \text{(III.10)}$$

avec $h_i = x_{i+1} - x_i$, et où les C_i sont solutions du système tridiagonal :

$$\frac{h_{i-1}}{6}C_{i-1} + \frac{h_{i-1}+h_i}{3}C_i + \frac{h_i}{6}C_{i+1} = \frac{y_{i+1}-y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}$$
$$\text{pour } i = 1, ..n-1, \text{ complété, en général, par } C_0 = C_n = 0.$$

Voici par exemple (Figure III.8) la spline cubique d'interpolation de la fonction $f(x) = \dfrac{1}{1+x^2}$ sur 10 intervalles. On observe la stabilité de cette interpolation par contraste avec le résultat obtenu (Figure III.5) par interpolation polynomiale.

3 Approximation au sens des moindres carrés

L'instabilité du procédé d'interpolation polynomiale lorsque le nombre de points augmente, d'une part, l'incertitude des résultats de mesure, d'autre part, conduisent à préférer à l'interpolation des méthodes d'approximation. Ainsi il est clair que l'expérimentateur qui relèvera 100 points quasiment alignés sera plus intéressé par la droite passant " au mieux " par ces 100 points plutôt que par le polynôme de degré 99 réalisant l'interpolation exacte.

La plus célèbre et la plus utile des méthodes d'approximation est la méthode des moindres carrés. La formalisation de l'idée intuitive d'une droite représentant "au mieux" un nuage de points au sens des moindres carrés se fait de la manière suivante.

3.1 Droite des moindres carrés

Soient N valeurs $y_1, y_2, ...y_i, ...y_N$ données aux N abscisses $x_1, x_2, ...x_i, ...x_N$. Le polynôme P de degré un : $P(x) = a_0 + a_1 x$ (représenté par une droite) qui

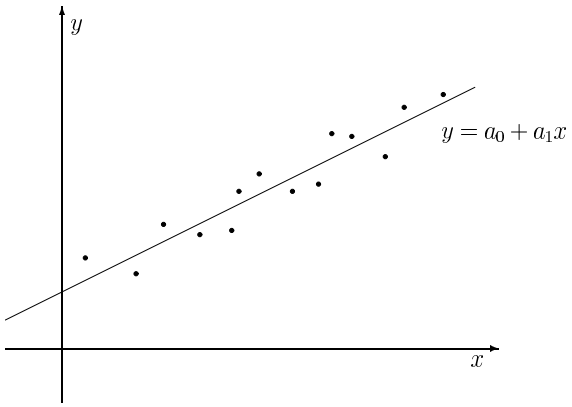

FIGURE III.9 – Droite des moindres carrés

réalise la meilleure approximation au sens des moindres carrés des valeurs y_i données aux points x_i est celui qui minimise la somme des carrés des écarts entre les y_i et les $P(x_i)$, soit

$$S(a_0, a_1) = \sum_{i=1}^{N} [y_i - (a_0 + a_1 x_i)]^2 \tag{III.11}$$

S apparaît comme le carré de la norme euclidienne du vecteur de composantes $y_i - (a_0 + a_1 x_i)$. La minimisation de S s'interprète donc comme la recherche du vecteur le plus proche du vecteur $Y \in \mathbb{R}^N$ de composantes y_i, dans le sous-espace de dimension deux engendré par les vecteurs U, de composantes toutes égales à 1, et X, de composantes x_i. Comme la norme utilisée est la norme euclidienne, le vecteur le plus proche est le projeté orthogonal. On obtient ses composantes a_0 et a_1 en écrivant les relations d'orthogonalité :

$$\begin{cases} (Y - a_0 U - a_1 X | U) = \displaystyle\sum_{i=1}^{N} [y_i - (a_0 + a_1 x_i)] \; 1 = 0 \\ (Y - a_0 U - a_1 X | X) = \displaystyle\sum_{i=1}^{N} [y_i - (a_0 + a_1 x_i)] \; x_i = 0 \end{cases} \tag{III.12}$$

Ceci conduit au système dit des équations normales pour a_0 et a_1, coefficients de la droite des moindres carrés (ou de régression) cherchée.

$$\begin{pmatrix} N & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 \end{pmatrix} \begin{pmatrix} a_0 \\ \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} x_i y_i \end{pmatrix} \tag{III.13}$$

3.2 Généralisation : polynôme des moindres carrés

Il est facile de généraliser le calcul précédent au cas de la recherche du polynôme de degré $\leq m$, avec $m << N$, qui réalise la meilleure approximation au sens des moindres carrés des y_i. Ce polynôme minimise

$$S(a_0, a_1, ...a_m) = \sum_{i=1}^{N} [y_i - (a_0 + a_1 x_i + a_2 x_i^2 + ...a_m x_i^m)]^2 \tag{III.14}$$

On obtient les relations d'orthogonalité :

$$\begin{cases} (Y - a_0 U - a_1 X - ...a_m X^m | U) = 0 \\ (Y - a_0 U - a_1 X - ...a_m X^m | X) = 0 \\ .... \\ (Y - a_0 U - a_1 X - ...a_m X^m | X^m) = 0 \end{cases} \tag{III.15}$$

où l'on a noté X^m le vecteur de composantes x_i^m. Les valeurs des coefficients a_k du polynôme des moindres carrés s'en déduisent après résolution du système linéaire déduit de (III.15).

Remarque 3.1 : *Le système des moindres carrés ci-dessus est mal conditionné (il est de plus en plus sensible aux erreurs d'arrondis à mesure que m augmente). On se limite habituellement à des polynômes de degré peu élevé. La résolution pratique des problèmes de moindres carrés se fait par des algorithmes spécifiques d'élimination pour des systèmes rectangulaires surdéterminés. Ces algorithmes utilisent la factorisation QR (technique de Householder).*

On peut envisager, comme dans le cas de l'interpolation, la recherche d'une meilleure approximation par des splines en découpant l'ensemble des points en sous-ensembles plus petits. Cette idée de meilleure approximation au sens des moindres carrés par des splines est à la base de nombreuses techniques d'approximation et de représentation de données.

4 Intégration numérique

La plupart des formules d'intégration numérique proviennent de méthodes d'interpolation polynomiales. Le calcul de l'intégrale d'une fonction est approché par le calcul exact de l'intégrale d'un polynôme interpolant cette fonction en certains points x_k pour $k = 1, ..p$. On obtient ainsi une forme générale pour les quadratures numériques

$$\int_a^b f(x)dx = \sum_{k=1}^{p} A_k f(x_k) \tag{III.16}$$

Les x_k sont alors les points d'intégration et les coefficients A_k les poids de la formule de quadrature.

Le coût d'une formule est mesuré par le nombre de calculs de f nécessaires, donc par le nombre de points d'intégration. Le critère d'efficacité choisi est le degré maximal de l'espace des polynômes pour lequel la formule est exacte. La précision d'une formule d'intégration numérique est mesurée par son ordre.

Déf. 4.1 : *On dit qu'une formule est d'ordre k si k est le plus grand entier pour lequel elle donne l'intégrale exacte de tout polynôme de degré $\leq k$. On montre que l'erreur d'intégration est alors, pour toute fonction suffisamment régulière, un infiniment petit d'ordre k du pas d'intégration h.*

4.1 Formules des rectangles

Ce sont les formules à un point d'intégration qui proviennent d'interpolation par des polynômes constants.

$$\int_a^b f(x)dx \approx (b-a)f(\alpha) \tag{III.17}$$

Le coût d'une telle formule à un point est celui d'une évaluation de f. Les choix classiques sont $\alpha = a, \quad \alpha = b$ et le choix donnant la meilleure formule dite *formule du point-milieu* (exacte pour les fonctions affines) est $\alpha = \frac{a+b}{2}$.

4.2 Formule des trapèzes

On considère cette fois l'interpolation par un polynôme de degré un construit sur les points a et b. On obtient la formule classique des trapèzes :

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\,[f(a)+f(b)] \tag{III.18}$$

Cette formule, à deux points, est clairement exacte pour les polynômes de degré ≤ 1.

4.3 Formule de Simpson

En utilisant l'interpolation sur les trois points $a, \frac{a+b}{2}, b$, on obtient la formule de Simpson, que l'on vérifie être exacte pour les polynômes de degré ≤ 3.

$$\int_a^b f(x)dx \approx \frac{b-a}{6}[f(a) + 4\, f(\frac{a+b}{2}) + f(b)] \tag{III.19}$$

Cette formule à trois points nécessite donc trois évaluations de f par segment (voir cependant, paragraphe 4.5 dans le cas d'un segment global subdivisé en sous-intervalles, les formules composites de calcul d'intégrales par les méthodes des trapèzes et de Simpson).

4.4 Formules de Gauss

Dans les formules précédentes le choix des points d'intégration était fixé (extrémités et/ou milieu des intervalles d'intégration). Dans les formules de type Gauss, les points d'intégration sont choisis de manière à obtenir la précision la plus élevée.
La **Formule de Gauss Legendre** à 2 points, est exacte pour les polynômes de degré ≤ 3 :

$$\int_a^b f(x) \approx \frac{b-a}{2}[f(\xi_1) + f(\xi_2)]. \tag{III.20}$$

$$\text{avec} \quad \xi_1 = \frac{a+b}{2} - \frac{b-a}{2}\frac{\sqrt{3}}{3} \quad \text{et} \quad \xi_2 = \frac{a+b}{2} + \frac{b-a}{2}\frac{\sqrt{3}}{3}$$

4.5 Formules composites, maillages et méthodes adaptatives

Toutes les formules présentées ci-dessus sont des formules de base, utilisables sur de petits éléments en dimension un, deux ou trois. Pour faire un calcul réel, il faut préalablement découper le domaine d'intégration global en un ensemble de petits sous-domaines élémentaires. Voici, pour fixer les

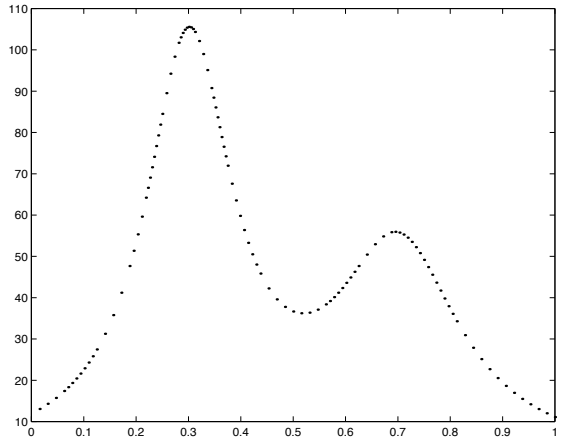

FIGURE III.10 – Choix optimal de points d'intégration par Matlab pour la fonction $f(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.7)^2+0.02}$.

idées, le calcul global, de l'intégrale d'une fonction f, par la méthodes des trapèzes, sur un intervalle $[a, b]$ découpé uniformément en N sous-intervalles de longueur h (le pas d'intégration) :

$$\int_a^b f(x)dx \approx \frac{h}{2}(f(a) + f(b)) + h \sum_{i=1}^{N-1} f(a + ih)$$

et le même calcul par Simpson

$$\int_a^b f(x)dx \approx \frac{h}{3}(f(a) + f(b)) + \frac{2h}{3} \sum_{i=1}^{P-1} f(a + 2ih) + \frac{4h}{3} \sum_{i=1}^{P} f(a + (2i - 1)h)$$

où $P = N/2$.

Cependant, un découpage géométrique uniforme en sous-intervalles égaux n'est pas optimal. L'opération de discrétisation géométrique ou "maillage" que l'on retrouvera dans le contexte des méthodes de différences, d'éléments ou de volumes finis est déterminante pour la précision du résultat. Il faut mettre plus de points d'intégration là où la fonction présente des variations relatives rapides. Dans les outils génériques comme Matlab, on trouve des procédures de quadrature adaptatives qui choisissent automatiquement les

points d'intégration (ou la taille des mailles) selon les variations de la fonction à intégrer. Typiquement, ces méthodes utilisent des indicateurs d'erreurs locaux basés sur l'écart entre deux calculs effectués avec deux méthodes de base différentes, trapèzes et Simpson par exemple (voir Fig III.10).

5 Résolution des équations différentielles

Nous limiterons l'exposé au cas simple d'équations différentielles ordinaires de la forme

$$\left\{ \begin{array}{ll} \text{Trouver la fonction } y : & x \to y(x) \quad \text{telle que :} \\ y'(x) = f(x, y(x)) & \forall x \in [a, b] \\ y(a) = y_0 & \end{array} \right. \tag{III.21}$$

Les problèmes différentiels de ce type sont appelés problèmes de Cauchy ou problèmes à valeurs initiales. Si f est continue et si elle vérifie une condition de Lipschitz par rapport à la deuxième variable (Il existe $L > 0$ tel que $\forall x \in [a, b]$ et $\forall y_1$, y_2 on ait : $|f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2|$), alors le problème admet une solution unique y pour toute valeur initiale. On dit qu'il est "bien posé". On a vu, lors du chapitre précédent, que certains problèmes différentiels bien posés du point de vue théorique pouvaient s'avérer impossibles à résoudre numériquement car instables. Numériquement, il faudra en effet considérer l'ensemble des solutions voisines de la solution exacte cherchée, solutions voisines correspondant à de petites perturbations des conditions initiales. Si ces solutions ne s'écartent pas trop de la solution de référence exacte, on aura un problème stable et on pourra construire des approximations numériques convenables.

En généralisant l'écriture de l'équation (III.21) au cas d'une fonction inconnue vectorielle, on pourra traiter des systèmes différentiels et des équations ou systèmes d'ordre supérieur à un par des extensions naturelles des techniques que nous présentons ci-dessous dans le cas de la dimension un par souci de simplicité.

5.1 Principe général des méthodes numériques

La solution exacte d'une équation différentielle du type (III.21) est une fonction continue. Les ordinateurs ne peuvent fournir qu'un nombre fini de résultats numériques. Tout commence donc par un choix préalable d'un nombre fini de points x_i sur $[a, b]$. Ceci s'appelle une discrétisation ou un maillage du domaine géométrique (ici le segment $[a, b]$). On limitera le calcul au calcul approché de la solution en ces points.

Le choix des points x_i est évidemment crucial pour la qualité de la solution numérique obtenue. Le maillage doit permettre de représenter de façon

précise la solution. Comme cette solution est au départ inconnue, on procède par des techniques d'adaptation de maillage a posteriori. On calcule une première solution sur un premier maillage. On déduit de ce premier calcul les zones de forte variation de la solution. On raffine le maillage dans ces zones.

Encore une fois dans un souci de simplicité, nous présenterons ici les méthodes numériques dans le cas de maillage à pas uniformes. Le problème de l'adaptation de maillage sera traité dans un cadre général chapitre **??**.

On peut construire des schémas d'intégration d'équations différentielles de diverses manières. Par exemple :

- les schémas d'intégration à un pas peuvent être obtenus en utilisant des formules d'intégration numérique approchée. En introduisant des sous-pas, on obtient les schémas de Runge et Kutta qui sont les plus pratiques et les plus utilisés ;
- les schémas multi-pas peuvent être construits par développement de Taylor.

Deux critères principaux gouvernent le choix d'une méthode :

- l'ordre, qui mesure la précision du schéma ou erreur de troncature faite en remplaçant l'équation différentielle exacte par son approximation. L'ordre de la méthode donnera, à convergence, l'ordre de l'erreur commise en fonction du pas de discrétisation.
- la stabilité, qui concerne le comportement de la solution approchée discrète et la propagation des erreurs d'arrondis dans le cas d'un calcul réel pour un pas de discrétisation fixé. Le schéma est stable si la solution discrète reste bornée quel que soit le nombre de pas de discrétisation.

Remarque 5.1 : *Un critère supplémentaire et important de choix des schémas concerne la facilité de mise en œuvre, notamment lors d'un redémarrage des calculs. Imaginons un calcul instationnaire ne pouvant faire l'objet d'un calcul complet, soit en raison de la modélisation (comme en météorologie par exemple, où de nouvelles conditions initiales et aux limites doivent être assimilées chaque jour par le modèle), soit en raison de l'implémentation et de la durée du calcul. Par exemple, sur un calculateur parallèle, le temps d'attente est lié au temps de calcul et à la quantité de mémoire requise. Dans le premier cas, comme dans le second, on aura recours aux approches à un pas de type Runge et Kutta pour assurer une précision élevée, plutôt qu'aux schémas multi-pas. En effet, quelle que soit la précision demandée, les méthodes de Runge et Kutta ne nécessitent que le stockage d'un seul état pour un redémarrage des calculs.*

5.2 Méthodes à un pas

Dans ces méthodes la valeur approchée y_{n+1} de la fonction inconnue y pour l'abscisse x_{n+1} est calculée en fonction des seules valeurs de l'abscisse précédente x_n, de l'approximation y_n et du pas de discrétisation h.
Si y_{n+1} s'obtient par une formule explicite de la forme

$$y_{n+1} = y_n + \Phi(x_n, y_n, h)$$

on dit que la méthode est *explicite*.
Si par contre y_{n+1} est donnée par une relation de la forme générale

$$y_{n+1} = y_n + \Phi(x_n, y_n, y_{n+1}, h)$$

on ne pourra l'obtenir que par la résolution d'une équation. On dit que la méthode est *implicite*.

La fonction Φ définit la méthode utilisée.

Ces schémas sont obtenus, par exemple, en intégrant l'équation différentielle et en utilisant des formules d'intégration numérique pour le second membre. L'ordre du schéma sera égal au degré du polynôme pour lequel l'intégration est exacte + 1.

$$\int_{x_n}^{x_{n+1}} y'(x)\,dx = y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} f(x, y(x))\,dx$$

À titre d'exemple, on obtient les schémas suivants :

- A l'ordre 1, avec une formule exacte pour les polynômes constants par morceaux, on obtient le **schéma d'Euler explicite :**

$$y_{n+1} - y_n = hf(x_n, y_n)$$

- Toujours à l'ordre 1, mais en utilisant le point d'arrivée, on obtient le **schéma d'Euler implicite :**

$$y_{n+1} - y_n = hf(x_{n+1}, y_{n+1})$$

– A l'ordre 2, en utilisant la méthode des trapèzes, on obtient le **schéma des trapèzes ou de Crank-Nicolson :**

$$y_{n+1} - y_n = \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_{n+1})].$$

En général un schéma explicite a une complexité plus faible mais impose des conditions de stabilité plus restrictives (voir plus loin).

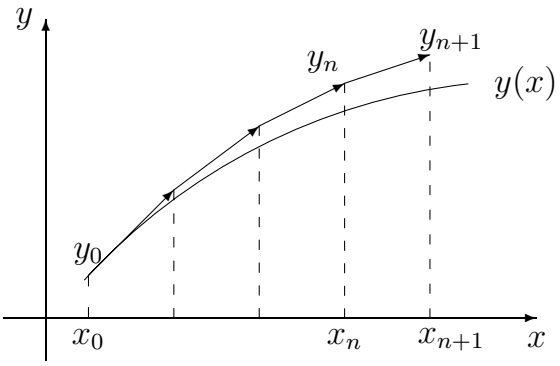

FIGURE III.11 – Interprétation de la méthode d'Euler comme méthode de la tangente. En chaque point x_n, utiliser un développement de Taylor à l'ordre un revient à remplacer la courbe solution par sa tangente.

5.3 Interprétations de la méthode d'Euler explicite

La méthode d'Euler est le prototype le plus simple des méthodes numériques de résolution des équations différentielles. Pour l'équation

$$\begin{cases} y'(x) = f(x, y(x)) & \forall x \in [a, b] \\ y(a) = y_0 \end{cases}$$

elle s'écrit :

$$\begin{cases} y_0 \text{ donné} \\ y_{n+1} = y_n + hf(x_n, y_n) \end{cases} \tag{III.22}$$

C'est la plus simple des méthodes explicites à un pas.

1. La méthode d'Euler provient du développement de Taylor d'ordre un de y au voisinage de x_n. On peut montrer que, lorsque cette méthode converge, l'erreur est un infiniment petit d'ordre un en h.
2. Géométriquement, elle revient à remplacer localement en chaque point x_n, la courbe solution par sa tangente. On voit donc qu'au cours du processus numérique, on va passer, à chaque pas d'une courbe solution à une courbe voisine correspondant à une condition initiale légèrement différente. Si le problème est stable, on pourra obtenir la convergence. Par contre, les résultats peuvent vite devenir catastrophiques dans un cas instable (exemple **??** du chapitre 1).
3. Enfin, en utilisant l'équivalence entre un problème différentiel et une équation intégrale, on peut interpréter, on l'a vu plus haut, la méthode d'Euler comme le résultat de l'application de la formule des rectangles basée sur le point x_n

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(t, y(t))dt \Rightarrow y_{n+1} = y_n + hf(x_n, y_n)$$

5.4 Méthodes de Runge et Kutta

Dans les méthodes de résolution des problèmes à valeurs initiales, le processus de calcul est un processus fini. On avance de N pas, à partir du temps initial jusqu'au temps final et on s'arrête. Chaque valeur est donc calculée une fois pour toutes. Sauf technique plus complexe d'adaptation de maillages, il n'y a pas de réitération pour améliorer le résultat. Il faudra donc utiliser des méthodes suffisamment précises. Ceci explique le recours à des méthodes d'ordre élevé. Les méthodes de Runge et Kutta sont les généralisations de la méthode d'Euler à des ordres supérieurs à un. Elles s'obtiennent à partir de formules d'intégration numériques plus précises que la formule des rectangles.

Considérons tout d'abord l'utilisation de la formule des trapèzes. Elle conduit à la méthode

$$\begin{cases} y_0 \text{ donné} \\ y_{n+1} = y_n + \dfrac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \end{cases} \tag{III.23}$$

Cette méthode est une méthode implicite. Le calcul de la nouvelle valeur y_{n+1} nécessite la résolution d'une équation. Si l'on veut obtenir une méthode

explicite du même ordre, on peut procéder de la manière suivante :

$$\begin{cases} y_0 \text{ donné} \\ {y_{n+1}}^* = y_n + hf(x_n, y_n) \\ y_{n+1} = y_n + \dfrac{h}{2}[f(x_n, y_n) + f(x_{n+1}, {y_{n+1}}^*)] \end{cases} \tag{III.24}$$

Ceci peut s'interpréter comme une itération de point-fixe (limitée ici à un pas) pour résoudre l'équation du schéma implicite des trapèzes (voir plus bas **??**). On obtient ainsi la méthode de **Runge et Kutta d'ordre 2 : RK2**.

De même l'utilisation de la formule d'intégration de Simpson est à la base de la formule de **Runge et Kutta d'ordre 4 : RK4**. C'est l'une des formules les plus utilisées, elle s'écrit :

$$\begin{cases} y_0 \text{ donné, puis pour } n \geq 0 \\ k_1 = hf(x_n, y_n) \\ k_2 = hf(x_n + h/2, y_n + k_1/2) \\ k_3 = hf(x_n + h/2, y_n + k_2/2) \\ k_4 = hf(x_n + h, y_n + k_3) \\ y_{n+1} = y_n + \dfrac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4] \end{cases} \tag{III.25}$$

Pour réduire la complexité en stockage de ce schéma (pas de stockage des coefficients k_i), on peut utiliser le schéma de Runge-Kutta sans stockage suivant :

$$y_{n+1} = y_n + \theta_p h f(y_{n+1}), p = 1...q, \quad \theta_p \in]0, 1]$$

où l'on passe de n à $n+1$ après q sous-itérations, sans stocker les valeurs intermédiaires. Les coefficients θ_p doivent être calés pour réaliser une intégration exacte à un ordre donné pour une fonction f donnée (ce qui est une limitation de cette technique).

Considérons l'équation $y'(x) = \lambda y(x)$, et prenons $q = 2$, le schéma s'écrit alors (on introduit pour la compréhension les v_i intermédiaires) :

$$\begin{cases} v_0 = y_n \\ v_1 = v_0 + h\theta_1 \lambda v_0 \\ v_2 = v_0 + h\theta_2 \lambda v_1 \\ y_{n+1} = v_2 \end{cases}$$

on a donc :

$$y_{n+1} = y_n + h\theta_2 \lambda y^n + h^2 \theta_1 \theta_2 \lambda^2 y_n$$

Or $y''(x) = \lambda^2 y(x)$ d'où par identification : $\theta_2 = 1$ et $\theta_1 \theta_2 = \frac{1}{2}$.

5.5 Application aux systèmes différentiels

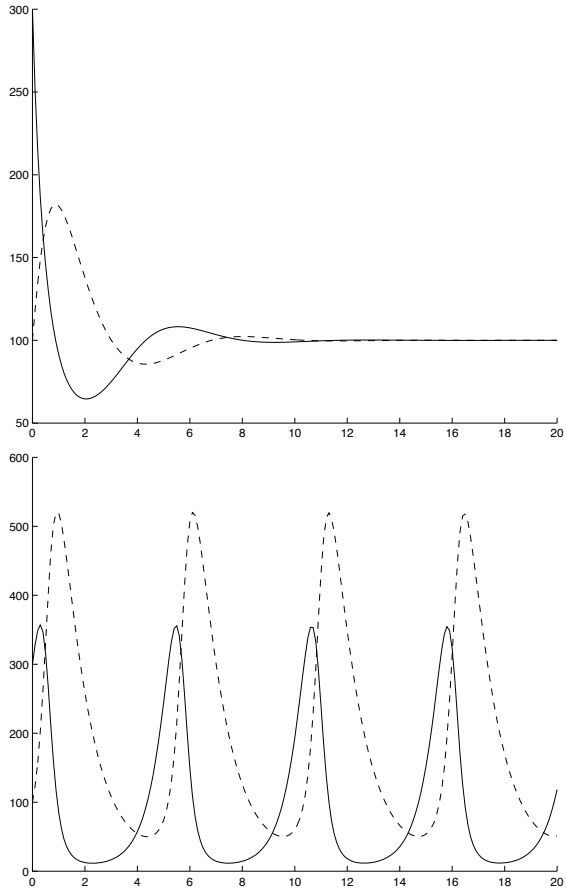

FIGURE III.12 – Système proie-prédateur : les proies sont représentées en trait continu, les prédateurs en pointillés. En haut le cas $b = 0.01$: proies en milieu fini, en bas le cas $b = 0$: pas de limite au développement de la population des proies, hormis la présence de prédateurs.

On peut généraliser simplement l'application des méthodes de Runge et

Kutta aux systèmes. Pour un système de deux équations couplées de la forme

$$\begin{cases} y_0, z_0 \text{ donnés} \\ y'(x) = f(x, y(x), z(x)) \\ z'(x) = g(x, y(x), z(x)) \end{cases} \tag{III.26}$$

on obtient l'algorithme suivant pour Runge et Kutta d'ordre 4 :

$$\begin{cases} y_0, z_0 \text{ donnés, puis pour } n \geq 0 & \\ k_1 = hf(x_n, y_n, z_n) & l_1 = hg(x_n, y_n, z_n) \\ k_2 = hf(x_n + h/2, y_n + k_1/2, z_n + l_1/2) & l_2 = hg(x_n + h/2, y_n + k_1/2, z_n + l_1/2) \\ k_3 = hf(x_n + h/2, y_n + k_2/2, z_n + l_2/2) & l_3 = hg(x_n + h/2, y_n + k_2/2, z_n + l_2/2) \\ k_4 = hf(x_n + h, y_n + k_3, y_n + l_3) & l_4 = hg(x_n + h, y_n + k_3, y_n + l_3) \\ y_{n+1} = y_n + \dfrac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4] & z_{n+1} = z_n + \dfrac{1}{6}[l_1 + 2l_2 + 2l_3 + l_4] \end{cases} \tag{III.27}$$

Voici trois applications classiques

1. Le système proies-prédateurs :

$$\begin{cases} y_1' = (\, a - by_1 - cy_2 \,)\, y_1 \\ y_2' = (\, -\alpha + \gamma y_1 \,) y_2 \\ y_1(0) = 300 \quad y_2(0) = 150 \\ a = 2, b = 0.01 \text{ (ou } b = 0), \, c = 0.01, \alpha = 1, \gamma = 0.01 \end{cases} \tag{III.28}$$

dont voici le programme en langage Matlab utilisant le schéma RK2.

```
x=0:0.1:20;
h=0.1;
a= 2;
b=0.01;
c=0.01;
alpha=1;
gamma=0.01;

y(1)=300;
```

```
z(1)= 100;

for i=1:200
   k1=h*(a -b*y(i)-c*z(i))*y(i);
   l1=h*(-alpha +gamma*y(i))*z(i);
   k2=h*(a-b*(y(i)+k1) -c*(z(i)+l1))*(y(i)+k1);
   l2=h*(-alpha +gamma*(y(i)+k1))*(z(i)+l1);
   y(i+1)=y(i)+(k1+k2)/2;
   z(i+1)=z(i)+(l1+l2)/2;
end
hold on
plot(x,y)
plot(x,z,'--')
hold off
```

2. L'équation du pendule amorti :

$$\begin{cases} \theta''(t) + \alpha\theta'(t) + k^2\, sin\,(\theta(t)) = 0 \qquad \text{sur} \quad [0, 4\pi] \\ \theta(0) = \dfrac{\pi}{2} \text{ et } \theta'(0) = 0 \end{cases} \tag{III.29}$$

   $k = 5$ et $\alpha = 0.1$

3. Le système dynamique de Lorentz :

$$\dot{x} = \sigma(y - x), \quad \dot{y} = \rho x - y - xz, \quad \dot{z} = xy - \beta z \tag{III.30}$$

   dont les conditions initiales sont précisées dans le programme ci-dessous :

```
! Etude du systeme dynamique de Lorentz

 x1=x0=-10.;
 y1=y0=20.;
 z1=z0=-5.;
 epsilon=1.e-2  ! perturbation 1 pourcent
 sigma=10.*(1.+epsilon)
 ro=28.*(1.+epsilon)
 beta=2.6667*(1.+epsilon)
```

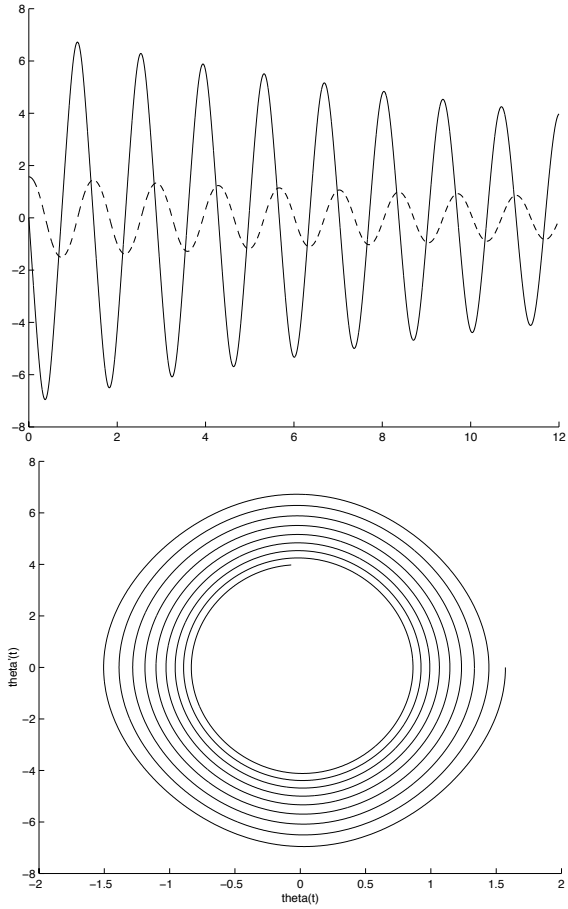

FIGURE III.13 – Oscillation du pendule amorti : les angles sont représentés en pointillés, leurs dérivées en trait continu. En bas le diagramme de phase, angles / vitesses angulaires

```
irkmax=4  ! Runge et Kutta a 4 pas
dt=1.e-3  ! pas de temps
 do kt=1,10000  ! boucle en temps
   do irk=1,irkmax ! boucle Runge et Kutta sans stockage
     alpha=1./(irkmax+1-irk);
     x1=x0+dt*alpha*(sigma*(y1-x1));
     y1=y0+dt*alpha*(ro*x1-y1-x1*z1);
```

```
      z1=z0+dt*alpha*(x1*y1-beta*z1);
    enddo
      x0=x1;  y0=y1;   z0=z1;
  enddo
```

Ci-dessus le pas de temps a été fixé a priori, à la suite d'essais numériques. Nous pouvons, par une analyse de stabilité, proposer un critère pour son choix. Cependant, l'analyse de stabilité s'avère souvent difficile pour les systèmes couplés. Dans ce cas, on effectue l'analyse pour chaque équation, en laissant invariantes les autres variables. Le pas de temps sera alors le minimum des pas de temps produits par ces analyses. Par exemple, dans le cas du système de Lorentz on obtient :

$$dt = \min(\frac{2}{\sigma}|\frac{x}{x-y}|, \quad 2|\frac{y}{-\rho x + y + xz}|, \quad 2|\frac{z}{\beta z - xy}|)$$

5.6 Méthodes à pas multiples

Les méthodes de Runge et Kutta sont des méthodes à un pas. Pour obtenir des méthodes d'ordre de précision élevé, on peut aussi augmenter le nombre de pas. Dans ce cas, la solution au point $n+1$ sera fonction des solutions aux p pas précédents avec $p > 1$. La construction de ces schémas peut se faire en utilisant la dérivation numérique et la précision du schéma sera donnée par la précision de cette dérivation. Ainsi, pour un schéma à l'ordre 2, on peut combiner les expressions ci-dessous :

$$y(x_{n+1}) = y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(x_n) + ...$$

$$y(x_{n-1}) = y(x_n) - hy'(x_n) + \frac{h^2}{2}y''(x_n) + ...$$

pour aboutir au schéma "saute-mouton" (leap-frog en anglais) :

$$\frac{y_{n+1} - y_{n-1}}{2h} = f(x_n, y_n) \tag{III.31}$$

Un autre schéma à deux pas largement utilisé est le schéma implicite d'ordre deux de Gear :

$$\frac{3}{2}y_{n+1} - 2y_n + \frac{1}{2}y_{n-1} = hf(x_{n+1}, y_{n+1}) \tag{III.32}$$

On trouvera dans les ouvrages spécialisés un grand nombre de formules multipas d'ordre élevé (voir cependant la remarque 5.1 sur les avantages pratiques des méthodes à un pas de type Runge-Kutta). Remarquons toutefois, qu'un schéma d'ordre élevé n'a d'intérêt que si l'on est assuré de la régularité suffisante de la solution. Pour une solution discontinue ou peu régulière des méthodes d'ordre bas sont mieux adaptées. D'autre part, il y a, en général, sauf évidemment dans le cas de schémas implicites, antagonisme entre ordre élevé et stabilité.

5.7 Stabilité

Par opposition à l'ordre de précision qui utilise la solution du problème continu, le concept de **stabilité** est basé sur la solution discrète. Il rend compte du comportement réel de la solution approchée pour une valeur pratique, donc non nulle, du pas h.

Lors d'un calcul réel, les erreurs d'arrondis, inévitables, s'accumulent. Ceci est particulièrement évident dans le processus de résolution d'une équation différentielle où l'on progresse pas à pas à partir d'une valeur initiale. Il existe diverses conditions de stabilité. Tout d'abord la solution numérique doit rester bornée. Cette exigence minimale de stabilité peut se révéler insuffisante dans la pratique, la borne obtenue étant souvent une exponentielle de la durée qui donc croît infiniment lorsque celle-ci augmente.

On introduit alors des critères de stabilité plus exigeants afin que la solution numérique reproduise le comportement physique de la solution exacte. Par exemple, pour des problèmes dissipatifs, on imposera des conditions de stabilité permettant d'obtenir une solution de norme décroissante. Le concept de A-stabilité, sous sa forme la plus simple, est basé sur l'analyse du comportement, selon les valeurs du pas h, des solutions numériques de l'équation modèle

$$y'(t) = -ay(t) \qquad \text{avec } y(0) \text{ donné et } a \text{ réel} > 0 \tag{III.33}$$

dont la solution exacte est $y(t) = y(0)e^{-at}$.

Étudions le cas du schéma d'Euler explicite. On obtient $y_{n+1} = y_n - ahy_n$, soit

$$y_n = (1 - ah)^n y_0$$

Si l'expression exacte e^{-at} est toujours décroissante en temps et positive, ce n'est pas le cas de l'expression approchée $(1-ah)^n$, qui selon les valeurs du pas $h>0$, peut tendre vers l'infini, vers zéro ou prendre alternativement les valeurs 1 et -1. Pour que la solution approchée reproduise le comportement de la solution exacte, donc reste positive et décroissante, il faut imposer une condition de stabilité sur le pas h. On doit avoir $h < \frac{1}{a}$. Pour des systèmes différentiels de type $X'(t) + AX(t) = F$ (A est supposée symétrique définie positive, donc à valeurs propres réelles positives), la condition de stabilité imposera, pour toute valeur propre λ_i de la matrice A, les conditions : $h < \frac{1}{\lambda_i}$. Donc $h < \frac{1}{max(\lambda_i)}$. Si certaines valeurs propres sont grandes, ceci imposera des pas h très petits, et donc des difficultés pour le calcul des solutions sur de longues durées. On dit, dans ce cas, que le système différentiel est *raide* (stiff en anglais).

Étudions maintenant le cas d'un schéma implicite, le schéma d'Euler implicite :

$$y_{n+1} = y_n + f(x_{n+1}, y_{n+1})$$

Son application à (III.33) donne :

$$(1+ah)^n y_n = y_0 \Longrightarrow y_n = \frac{y_0}{(1+ah)^n}$$

Dans ce cas, quel que soit $h>0$, la solution numérique est bornée, positive et décroissante au cours du temps. On dit que le schéma est inconditionnellement stable.

Remarque 5.2 : *On retrouve ici les approximations stables et instables de la fonction exponentielle présentées au chapitre précédent.*

6 Méthodes de résolution des systèmes linéaires

Les systèmes linéaires sur-déterminés (plus grand nombre d'équations que d'inconnues) sont résolus, en général, en utilisant les techniques de moindres carrés. Nous nous limitons donc ici au cas des sytèmes carrés comportant autant d'équations que d'inconnues.

On considère le système suivant :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ...a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + ...a_{2n}x_n = b_2 \\ \qquad\qquad ..... \\ \qquad\qquad ..... \\ \qquad\qquad ..... \\ a_{n1}x_1 + a_{n2}x_2 + ...a_{nn}x_n = b_n \end{cases}$$

qui s'écrit matriciellement

$$AX = B$$

Soit ϕ l'application linéaire représentée par la matrice A, si X représente x et B, b, $AX = B$ correspond à $\phi(x) = b$.

6.1 Existence et unicité de la solution

Théorème 6.1 (Systèmes de Cramer) **:** *Une condition nécessaire et suffisante pour qu'un système linéaire de n équations à n inconnues admette une solution unique pour tout second membre $B \in \mathbb{R}^n$ est de manière équivalente que*

- $det(A) \neq 0$
- $Ker(\phi) = \{0\}$
- $rg(\phi) = dim(Im(\phi)) = n$
- *les vecteurs lignes ou les vecteurs colonnes de A sont indépendants*
- *la seule solution de* $AX = 0$ *est* $X = 0$

6.2 Méthodes directes

Les méthodes directes de résolution des systèmes linéaires sont des méthodes dans lesquelles la solution est obtenue de façon exacte en un nombre fini d'opérations. De façon exacte s'entend, sur un ordinateur, aux erreurs d'arrondis machine près. Le prototype de méthode directe est la méthode du pivot de Gauss. Cette méthode permet de ramener la résolution d'un système

général à la résolution d'un système triangulaire supérieur. La triangularisation de Gauss consiste à annuler, par étape, colonne par colonne, les coefficients sous-diagonaux de la matrice par combinaison des lignes avec une ligne de référence ou ligne "pivot". À la première étape, la ligne pivot est la première ligne, puis la k^e à l'étape k et ainsi de suite jusqu'à l'étape $n-1$. Le coefficient diagonal $A_{k,k}$ de la ligne de référence s'appelle le pivot. L'élimination se fait selon

$$A_{i,j}^{(k+1)} = A_{i,j}^{(k)} - \frac{A_{i,k}^{(k)}}{A_{k,k}^{(k)}} A_{k,j}^{(k)}$$

pour $k = 1...N-1,\ i = k+1...N$ et $j = k+1...N$, sans oublier de combiner de la même façon les composantes du second-membre. La rencontre de pivot nul peut nécessiter la permutation de lignes du système. Cependant pour certaines classes de matrices, en particulier les matrices symétriques définies positives, on est assuré de pouvoir trianguliariser le système par Gauss sans permutation. La méthode du pivot équivaut alors à une factorisation de type

$$A = LU$$

de la matrice A. L est une matrice triangulaire inférieure à diagonale unité et U une matrice triangulaire supérieure. Une fois obtenu le système triangulaire supérieur équivalent au système initial, sa résolution se fait ensuite explicitement par un processus de remontée. On commence par calculer la dernière composante du vecteur inconnu en utilisant la dernière équation et on remonte équation par équation en déterminant les composantes correspondantes.

Dans le cas d'une matrice A symétrique, on peut utiliser la symétrie pour obtenir une factorisation de Crout :

$$A = LDL^T$$

avec D diagonale.

Dans le cas d'une matrice A symétrique définie positive, la méthode de Choleski conduit à une factorisation :

$$A = LL^T$$

On trouve la matrice L, qui cette fois n'est plus à diagonale unité, par un algorithme d'identification de coefficients.

De

$$A_{ij} = \sum_{k=1,i} L_{ik} L_{jk} \quad \text{pour} \quad j \leq i$$

on déduit pour tout i :

$$L_{ii} = \sqrt{A_{ii} - \sum_{k=1,i-1} L_{ik}^2}$$

et pour tout $j < i$

$$L_{ij} = \frac{(A_{ij} - \sum_{k=1,j-1} L_{ik} L_{jk})}{L_{jj}}$$

Les méthodes directes présentent l'avantage de fournir la solution exacte (aux erreurs d'arrondis machine près) en un nombre fini d'opérations (de l'ordre de $\frac{n^3}{3}$ pour Gauss). La méthode du pivot de Gauss s'applique à tout système inversible. Par contre les méthodes directes ont un coût important en stockage mémoire (bien que des techniques de stockage minimal associées à des algorithmes de numérotation optimale des inconnues permettent de le réduire sensiblement). Ceci rend leur application pratiquement impossible, en l'état actuel de la technologie, pour de gros systèmes à plus de 10^5 inconnues, et donc en particulier pour la résolution de problèmes industriels en dimension 3 d'espace.

Nous renvoyons à la littérature pour plus de précisions sur les méthodes directes (voir Lascaux-Théodor, Ciarlet, Lucquin-Pironneau).

6.3 Méthode de Gauss-Seidel ou de relaxation

Soit encore

$$AX = B$$

le système linéaire à résoudre. La méthode de Gauss-Seidel correspond aussi à la décomposition de la matrice A sous la forme

$$A = D - E - F$$

avec D matrice diagonale constituée des éléments diagonaux a_{ii} de A,

$-E$, matrice triangulaire inférieure stricte, constituée des éléments strictements sous-diagonaux de A : a_{ij} pour $i > j$, et

$-F$, matrice triangulaire supérieure stricte, constituée des éléments strictements sur-diagonaux de A : a_{ij} pour $i < j$.

Mais on définit cette fois la méthode de Gauss-Seidel comme la méthode itérative :

$$\begin{cases} X^{(0)} \quad \text{donné} \\ \\ X^{(k+1)} = (D-E)^{-1}F\ X^{(k)} + (D-E)^{-1}B \end{cases} \tag{III.34}$$

La matrice d'itération de Gauss-Seidel est donc

$$\mathcal{L} = (D-E)^{-1}F$$

et la condition de convergence s'exprime par

$$\rho(\mathcal{L}) < 1$$

On observe que la méthode de Gauss-Seidel correspond à l'écriture ligne par ligne suivante :

$$x_i^{k+1} = \frac{b_i - \sum_{j<i} a_{ij}x_j^{k+1} - \sum_{j>i} a_{ij}x_j^k}{a_{ii}} \tag{III.35}$$

Elle utilise les informations les plus récentes sur les composantes des itérés. La méthode de Gauss-Seidel converge en général plus vite que la méthode de Jacobi. Par contre, Gauss-Seidel n'est pas adaptée à la vectorisation.

Les méthodes de **relaxation** sont des extrapolations de la méthode de Gauss-Seidel dépendant d'un paramètre ω. Ce paramètre est déterminé pour obtenir une convergence plus rapide. Les méthodes de relaxation s'écrivent :

$$\begin{cases} X^{(0)} \quad \text{donné} \\ \\ X^{(k+1)} = (D-\omega E)^{-1}(\omega F + (1-\omega)D)\ X^{(k)} + \omega(D-\omega E)^{-1}B \end{cases} \tag{III.36}$$

La méthode de relaxation s'écrit ligne par ligne comme une extrapolation de la composante obtenue par Gauss-Seidel. On a :

$$X_i^{k+1}(relax) = \omega\, X_i^{k+1}(GS) + (1-\omega)\, X_i^k(relax)$$

La méthode de Gauss-Seidel correspond au cas $\omega = 1$. Dans le cas de matrices A symétriques définies positives, les méthodes de relaxation convergent pour $0 < \omega < 2$.

6.4 Méthodes de descente - Méthode du gradient

Les méthodes de descente sont des méthodes itératives qui utilisent l'équivalence entre les problèmes suivants (voir thÈorËme **??**) :

$$\begin{cases} \text{Trouver } X \in I\!R^N \quad \text{tel que} \\ \\ A\,X = B \end{cases} \tag{III.37}$$

et

$$\begin{cases} \text{Trouver } X \in I\!R^N \quad \text{tel que} \\ \\ J(X) = \dfrac{1}{2}(A\,X, X) - (B, X) \;\; \text{soit minimal} \end{cases} \tag{III.38}$$

et qui sont donc limitées au cas des systèmes linéaires dont la matrice A est symétrique définie positive. Elles se généralisent, par contre, au cas non-linéaire de la minimisation de fonctionnelles strictement convexes quelconques (voir chapitre **??**).

Les méthodes de descente sont basées sur le calcul de la solution comme limite d'une suite minimisante de la forme quadratique J. Cette suite est construite comme une suite récurrente :

$$\begin{cases} X^{(0)} \quad \text{donné} \\ \\ X^{(k+1)} = X^{(k)} - \alpha_k\, d^{(k)} \end{cases} \tag{III.39}$$

avec $d^{(k)} \in I\!R^N$ vecteur donnant la direction de descente à l'étape k et $\alpha_k \in I\!R$ coefficient déterminé de manière à minimiser J dans la direction $d^{(k)}$.

$$J(X^{(k)} - \alpha_k\, d^{(k)}) \leq J(X^{(k)} - \alpha\, d^{(k)}) \quad \forall \alpha \in I\!R$$

Après développement, on obtient α_k comme valeur annulant la dérivée de J par rapport à α, soit :

$$\alpha_k = \frac{(G^{(k)}, d^{(k)})}{(Ad^{(k)}, d^{(k)})}$$

avec

$$G^{(k)} = grad(J(X^{(k)})) = AX^{(k)} - B$$

On peut montrer que la méthode de Gauss-Seidel est la méthode de descente correspondant aux choix des axes de coordonnées comme directions successives de descente. Dans le cas de la **méthode du gradient**, on choisit comme direction de descente la direction du vecteur gradient de J au point $X^{(k)}$. Cette direction est la direction de variation maximale de J. On dit encore direction de plus profonde descente, d'où le nom :"steepest descent method" , employé dans certains ouvrages en anglais.

L'itération de la méthode du gradient s'écrit :

$$\begin{cases} X^{(0)} \quad \text{donné} \\ \\ X^{(k+1)} = X^{(k)} - \alpha_k \, G^{(k)} \end{cases} \tag{III.40}$$

avec

$$\alpha_k = \frac{\|G^{(k)}\|^2}{(AG^{(k)}, G^{(k)})}$$

6.4.1 Vitesse de convergence. Conditionnement.

À partir de $G^{(k+1)} = AX^{(k+1)} - B$ et de $X^{(k+1)} = X^{(k)} - \alpha_k \, G^{(k)}$, on obtient la récurrence suivante sur les gradients :

$$G^{(k+1)} = G^{(k)} - \alpha_k \, AG^{(k)}$$

On en déduit $\|G^{(k+1)}\|^2 = \|G^{(k)}\|^2 - 2\alpha_k(AG^{(k)}, G^{(k)}) + \alpha_k^2 \|AG^{(k)}\|^2$

$$\text{avec} \quad \alpha_k = \frac{\|G^{(k)}\|^2}{(AG^{(k)}, G^{(k)})}$$

on obtient :

$$\|G^{(k+1)}\|^2 = \|G^{(k)}\|^2 \big[\frac{\|G^{(k)}\|^2 \|AG^{(k)}\|^2}{(AG^{(k)}, G^{(k)})^2} - 1\big]$$

Et en utilisant les valeurs propres et les vecteurs propres de la matrice A supposée définie positive, on déduit :

$$\|G^{(k+1)}\|^2 \leq \|G^{(k)}\|^2 \big[\frac{\lambda_{max}}{\lambda_{min}} - 1\big]$$

Déf. 6.1 : *Pour une matrice symétrique réelle le nombre*

$$K(A) = \frac{\lambda_{max}}{\lambda_{min}}$$

rapport des plus grandes et plus petites valeurs propres de A est appelé nombre de conditionnement de la matrice A.

Plus il est proche de 1, plus vite la méthode du gradient converge. Les techniques de préconditionnement ont, entre autres, pour but de rapprocher les valeurs propres extrêmes afin d'accélerer la convergence des méthodes itératives.

Remarque 6.1 : *Le nombre de conditionnement est, en particulier, égal à 1 pour la matrice identité. D'ailleurs, dans ce cas, l'expression à minimiser décrirait, en dimension deux, un ensemble de cercles concentriques. Les gradients ayant la direction d'un rayon, on obtiendrait la solution en une itération. Cette remarque est à la base de l'idée de la méthode du gradient conjugué.*

6.5 Méthode du gradient conjugué

Dans le cas d'une matrice symétrique définie positive quelconque A, les iso-J sont des hyper-ellipses. Elles redeviennent des cercles pour le produit scalaire définie par A :

$$X, Y \longrightarrow (X, Y)_A = (AX, Y)$$

D'où l'idée de remplacer les directions de descente dans la méthode du gradient, par des directions conjuguées, c'est à dire orthogonales au sens du produit scalaire $(.,.)_A$.

On obtient alors l'algorithme :

$$\begin{cases} X^0 \in I\!R^N \quad \text{donné} \\ d^0 = G^0 = AX^0 - B \\ \text{puis pour } k = 0, 1... \\ \alpha_k = \dfrac{\|G^k\|^2}{(Ad^k, d^k)} \\ X^{k+1} = X^k - \alpha_k \, d^k \\ G^{k+1} = G^k - \alpha_k \, Ad^k \\ \beta_{k+1} = \dfrac{\|G^{k+1}\|^2}{\|G^k\|^2} \\ d^{k+1} = G^{k+1} + \beta_{k+1} \, d^k \end{cases} \tag{III.41}$$

Nous renvoyons à la littérature pour une étude exhaustive de la méthode du gradient conjugué. Retenons que la propriété de A-orthogonalité entraîne une convergence théorique en N itérations. Ce qui en ferait une méthode directe. Cependant, elle serait alors plus chère que Choleski. De plus, les erreurs d'arrondis font que les propriétés d'orthogonalité ne sont pas vérifiées exactement. Il faut donc considérer le gradient conjugué comme une méthode itérative. On montre que sa vitesse de convergence dépend également du conditionnement de la matrice A. On est conduit à préconditionner A pour obtenir des performances intéressantes. Parmi les préconditionnements classiques, on citera le préconditionnement SSOR (O. Axelsson) et par Choleski incomplet (Meijerink et Van der Vorst).

Dans le cas où la matrice du système n'est pas symétrique définie positive, il est plus difficile d'obtenir des méthodes itératives performantes. Mentionnons la méthode GMRES (Saad et Schultz), dont il existe également une version pour la résolution de systèmes non-linéaires.

6.6 Application des méthodes de gradient au cas non-linéaire

Dans le cas plus général de minimisation d'une fonctionnelle J strictement convexe non nécessairement quadratique, le gradient ∇J est non-linéaire. Cependant les méthodes de gradient peuvent s'appliquer (on peut même les

appliquer sans garantie de convergence, car dans tous les cas on réduira J, voir chapitre **??**). La détermination du pas optimal α_k à l'itération k se fait alors par une méthode de recherche de l'argument α_k qui minimise la fonction $J(X^k - \alpha \nabla J(X^k))$. On est ramené à un problème en dimension un pour lequel diverses techniques existent, en particulier les algorithmes de recherche de type section dorée.

L'extension de la méthode du gradient conjugué au cas non-linéaire nécessite, de plus, la définition des directions de descente "conjuguées". L'algorithme le plus efficace est celui de Polak-Ribière. Les directions de descente successives sont données par

$$d^{k+1} = \nabla J(X^{k+1}) + \beta_{k+1}\, d^k$$

avec cette fois

$$\beta_{k+1} = \frac{(\nabla J(X^{k+1}), \nabla J(X^{k+1}) - \nabla J(X^k))}{\|\nabla J(X^k)\|^2}$$

7 Calcul des valeurs et vecteurs propres

Les méthodes efficaces de calcul des éléments propres d'une matrice, dont on verra plus loin des applications, en vibrations de structures, par exemple, utilisent toujours des techniques itératives. Commençons par exposer la méthode la plus simple, la méthode de la **puissance**.

7.1 La méthode de la puissance

La méthode de la puissance est basée sur le fait suivant. Si la matrice A admet une seule valeur propre de plus grand module, l'itération vectorielle

$$V^{k+1} = \frac{AV^k}{\|AV^k\|}$$

lorsqu'elle converge, admet comme limite un vecteur propre de la matrice A associé à cette valeur propre. La division par la norme a pour but d'éviter d'atteindre des valeurs trop grandes (ou trop petites) des composantes lors des itérations vectorielles. Un cas pratique important où l'on peut démontrer la convergence est celui des matrices symétriques réelles dont la valeur propre de plus grand module est unique. En effet, une matrice symétrique réelle

admet une base de vecteurs propres orthonormés $\{e_i\}$. Considérons l'initialisation :

$$V^0 = \sum_{i=1}^{N} \alpha_i e_i$$

et supposons que e_1 soit vecteur propre associé à la valeur propre λ_1 de plus grand module. Il est alors facile d'obtenir :

$$V^{k+1} = \frac{\sum_{i=1}^{N} \lambda_i^{k+1} \alpha_i e_i}{(\sum_{i=1}^{N} (\lambda_i^{k+1} \alpha_i)^2)^{\frac{1}{2}}}$$

On en déduit :

$$\lim_{k \to \infty} V^k = \frac{\lambda_1^{k+1} \alpha_1}{|\lambda_1^{k+1} \alpha_1|} \, e_1 = \pm e_1$$

La valeur propre de module maximal s'obtenant simplement par

$$\lambda_1 = \lim_{k \to \infty} \frac{(AV^k, V^k)}{(V^k, V^k)}$$

Les limites de cette méthode sont les suivantes :

- elle ne converge efficacement que dans le cas de valeurs propres simples de plus grand module. La vitesse de convergence dépend de la vitesse à laquelle les rapports $(\frac{\lambda_i}{\lambda_1})^k$ tendent vers zéro quand k augmente. La convergence sera donc d'autant plus rapide que l'écart entre la plus grande valeur propre et les suivantes est grand. Dans le cas de valeurs propres multiples, la convergence est ralentie. Dans le cas de valeurs propres distinctes, mais de module égal (valeurs propres opposées ou complexes conjuguées) l'algorithme se complique, on ne peut obtenir directement les valeurs propres et les vecteurs propres associés ;
- elle ne fournit qu'une valeur propre. Les techniques de "déflation", pour en déduire les suivantes, sont peu efficaces et trop sensibles aux erreurs d'arrondis.

Voici ses avantages :

- elle est facile à implémenter ;
- elle permet de calculer efficacement les vecteurs propres et d'améliorer sensiblement la précision des valeurs propres, si l'on a obtenu, par une autre méthode, des approximation des valeurs propres ;
- enfin, il est facile de l'adapter à la recherche d'une autre valeur propre que la plus grande, comme on va le voir plus loin.

Remarque 7.1 : *On pourrait penser que le cas $\alpha_1 = 0$, toujours possible théoriquement si l'on choisit, par hasard, un vecteur initial V^0 orthogonal au vecteur propre e_1 recherché, pose des problèmes. En réalité, voici une situation où paradoxalement, les erreurs d'arrondis sont bénéfiques. Dans la pratique la composante α_1 sera toujours non-nulle. Et même si elle est faible, l'itération finira par converger vers le vecteur propre e_1 associé à la valeur propre de plus grand module.*

7.1.1 Méthode de la puissance inverse

Il est clair que si l'on peut obtenir la plus grande valeur propre, on peut obtenir la plus petite (et le vecteur propre associé) à partir d'une itération utilisant la matrice inverse. On sait, en effet, que les valeurs propres de A^{-1} sont les inverses des valeurs propres de A. Ceci conduit à l'itération

$$\left\{\begin{array}{l} V^0 \text{ donné,} \\ \text{ puis pour } k = 0, 1, ... \\ A\,\tilde{V}^{(k+1)} = V^{(k)} \\ \mu_{k+1} = |\tilde{V}^{(k+1)}| \\ V^{(k+1)} = \frac{\tilde{V}^{(k+1)}}{\mu_{k+1}} \end{array}\right. \tag{III.42}$$

Les $V^{(k)}$ convergent vers le vecteur propre recherché, la plus petite valeur propre en module de A s'obtient comme $\lim\limits_{k\to\infty} \dfrac{1}{\mu_k}$. La matrice A est en général factorisée une fois pour toutes, la résolution du système se réduit à celle de deux systèmes triangulaires.

Remarque 7.2 : *Dans les applications aux vibrations, la plus petite valeur propre correspond au mode fondamental. C'est donc souvent celle que l'on recherche. Il arrive également que l'on soit intéressé par les modes de fréquences proches d'une fréquence donnée. C'est en particulier le cas lorsque l'on cherche à éviter des phénomènes de résonance. Dans ce cas la valeur propre recherchée est la plus proche d'une valeur μ donnée. Il est facile d'utiliser alors la méthode de la puissance inverse sur la matrice décalée $A - \mu I$, dont la plus petite valeur propre, en module, donnera bien la valeur propre de A la plus proche de μ.*

7.2 Méthodes des sous-espaces

Afin de calculer simultanément plusieurs valeurs propres et leurs vecteurs propres associés, on itère sur plusieurs vecteurs au lieu d'un seul. Bien sûr, il faut empêcher tous les vecteurs de converger vers le même vecteur propre (celui associé à la plus grande valeur propre). Pour cela, la méthode des sous-espaces consiste à initialiser l'itération par un ensemble de m vecteurs orthonormés de $\mathbb{R}^N$, puis à réorthonormaliser régulièrement les vecteurs itérés. La réorthonormalisation est une opération coûteuse en temps de calcul. On évite de la faire à chaque itération. Elle utilise une factorisation QR (Q matrice orthogonale, c'est à dire telle que $Q^T Q = I$, et R matrice triangulaire supérieure de taille $m \times m$) de type Gram-Schmidt que l'on peut implémenter par la technique de Householder (voir Lascaux-Théodor). En pratique, si l'on veut obtenir p valeurs propres et vecteurs propres avec une précision acceptable, il faut itérer sur un ensemble de $m > 2p$ vecteurs.

7.3 Méthode QR

Pour obtenir l'ensemble des valeurs propres et des vecteurs propres d'une matrice, la principale méthode utilisée est la méthode QR. C'est une itération de type sous-espace, avec comme sous-espace, l'espace $\mathbb{R}^N$ tout entier. L'algorithme QR s'écrit :

$$\left\{ \begin{array}{l} A_0 = A \quad \text{! initialisation} \\ \text{puis pour } k = 0, 1... \\ \quad Q_k \, R_k = A_k \quad \text{! factorisation } QR \\ \quad A_{k+1} = R_k \, Q_k \quad \text{! itération par calcul du produit} \end{array} \right. \tag{III.43}$$

Il est facile de vérifier qu'à chaque itération $A_{k+1} = Q_k^T \, A_k \, Q_k$, donc que les matrices A_k obtenues sont semblables à A pour tout k.

La méthode QR est un des algorithmes les plus coûteux de l'analyse numérique matricielle. En pratique, on n'itère pas sur la matrice initiale, mais sur une matrice semblable H de forme Hessenberg (ses coefficients $H_{i,j}$ sont nuls pour $i > j + 1$) ou tridiagonale dans le cas symétrique. Ce qui réduit sensiblement le nombre d'opérations. De nombreux travaux ont été menés pour démontrer et aussi pour accélerer la convergence de la méthode QR. On peut citer en particulier les méthodes avec "shift" dues à Wilkinson. Le cas de matrices A symétriques réelles se rencontre dans de nombreux problèmes intéressants en pratique comme en vibrations de structures par

exemple. On a montré la convergence de la suite de matrices tridiagonales symétriques vers une matrice diagonale dont les coefficients sont donc les valeurs propres recherchées (Wilkinson, Parlett, Saad).

7.4 Méthode de Lanczos

Si l'on ne recherche qu'une partie des valeurs propres et des vecteurs propres on utilisera la méthode de Lanczos. Cette méthode permet de réduire la dimension du problème avant de poursuivre par une méthode générale, QR par exemple. La méthode de Lanczos est une méthode de calcul de valeurs propres de type sous-espace appliquée au cas de matrices symétriques réelles. Elle permet de construire itérativement, colonne par colonne, une matrice tridiagonale de taille réduite (matrice de Rayleigh) dont les éléments propres sont des approximations d'un sous-ensemble d'éléments propres de la matrice A. On obtient les valeurs propres de plus grand module par itération directe, et comme pour la méthode de la puissance, celles de plus petit module par itération inverse. Elle s'interprète en fait comme une méthode de projection. On projette le problème $AV = \lambda V$ dans un sous-espace de dimension $m << N$, selon $(AV - \lambda V, q_i) = 0$ pour $i = 1...m$. Voici l'algorithme de Lanczos pour la recherche des m plus grandes valeurs propres et des vecteurs propres associés d'une matrice A symétrique réelle $N \times N$:

$$\left\{\begin{array}{l} \text{Soit } q_0 = 0 \text{ et } q_1 \in I\!R^N \text{ tel que } \|q_1\|_2 = 1 \\ \text{On calcule successivement pour } k = 1, 2...m-1 \\ w_k = Aq_k - \beta_{k-1}q_{k-1} \quad \text{! itération directe} \\ \alpha_k = (w_k, q_k) \quad \text{! produit scalaire} \\ v_k = w_k - \alpha_k q_k \quad \text{! orthogonalisation} \\ \beta_k = \|v_k\|_2 \quad \text{! Calcul de la norme euclidienne} \\ q_{k+1} = v_k/\beta_k \quad \text{! normalisation} \end{array}\right. \tag{III.44}$$

On obtient ainsi m vecteurs orthonormés q_i et une matrice T tridiagonale symétrique $m \times m$ (matrice de Rayleigh) constituée d'éléments diagonaux $T_{i,i} = \alpha_i$ et extradiagonaux $T_{i,i+1} = T_{i+1,i} = \beta_i$. Les valeurs propres et vecteurs propres de T se calculent par la méthode QR. On obtient ainsi les m plus grandes valeurs propres de A et les composantes des vecteurs propres de A dans la base des q_i. Cet algorithme est assez délicat. Les erreurs d'arrondis produisent, comme pour le gradient conjugué, des défauts d'orthogonalité. Il est nécessaire de réorthogonaliser de temps en temps les q_i (voir Lascaux-Théodor et Parlett pour plus de détails).