Université du Sud Toulon-Var
ISITV - 1ère année

Analyse Numérique

Paola GOATIN

Table des matières

Version provisoire. Merci de me signaler les erreurs !

1 Calculs numériques approchés.

L'objet de cette section est de mettre en évidence les principales difficultés liées à la pratique des calculs numériques sur ordinateur.

1.1 Représentation décimale approchée des nombres réels.

La capacité mémoire d'un ordinateur étant finie, il est nécessaire de représenter les nombres réels sous forme approchée. La notation la plus utilisée est la *représentation avec virgule flottante* : si x est un nombre réel, on pose

$$x \simeq \pm m \cdot b^p,$$

où b est la *base de numération*, m la *mantisse*, et p l'*exposant*. (Les calculs sont généralement effectués en base $b = 2$, les résultats affichés sont traduits en base 10.)
La mantisse m est un nombre écrit avec virgule fixe et possédant un nombre maximum N de chiffres significatifs (imposé par la mémoire de l'ordinateur) :

$$m = 0, a_1 a_2 \dots a_N = \sum_{k=1}^{N} a_k b^{-k}, \qquad b^{-1} \leq m < 1.$$

La *précision relative* dans l'approximation d'un nombre réel est donc donnée par

$$\frac{\Delta x}{x} = \frac{\Delta m}{m} \leq \frac{b^{-N}}{b^{-1}} = b^{1-N}.$$

L' exposant p est compris entre deux nombres entiers, $L \leq p \leq U$, ce qui veut dire que les nombres réels qui peuvent être représentés en machine sont compris entre deux valeurs x_{min} et x_{max} :

$$x_{min} = b^{L-1} \leq |x| \leq b^U = x_{max}.$$

Voici en résumant la repartition des n positions de mémoire pour la représentations des nombres réels dans l'ordinateur : une position pour le signe, N positions pour les chiffres significatifs, et les restantes $n - N - 1$ positions pour l'exposant.

Exemple. La même écriture peut représenter des nombre différents dans des bases différentes : $425,33$ en base 10 représente le nombre $x = 4{\cdot}10^2{+}2{\cdot}10{+}5{+}3{\cdot}10^{-1}{+}3{\cdot}10^{-2}$, en base 6 il représente le nombre $y = 4{\cdot}6^2{+}2{\cdot}6{+}5{+}3{\cdot}6^{-1}{+}3{\cdot}6^{-2}$. D'autre part, le même nombre peut avoir un nombre fini de chiffres dans une base, et un nombre infini dans un'autre base : $x = 1/3$ donne $x_3 = 0,1$ en base 3 et $x_{10} = 0,\overline{3}$ en base 10.

1.2 Non-associativité des opérations arithmétiques.

Supposons par exemple que les réels soient calculés avec $N = 3$ chiffres significatifs et arrondis à la décimale la plus proche. Soient

$$x = 8,22 = 0,822{\cdot}10, \quad y = 0,00317 = 0,317{\cdot}10^{-2}, \quad z = 0,00432 = 0,432{\cdot}10^{-2}.$$

On veut calculer la somme $x + y + z$.
$(x + y) + z$ donne :

$$x + y = 8,22317 \simeq 0,822 \cdot 10$$
$$(x + y) + z \simeq 8,22432 \simeq 0,822 \cdot 10$$

$x + (y + z)$ donne :

$$y + z = 0,00749 \simeq 0,749 \cdot 10^{-2}$$
$$x + (y + z) = 8,22749 \simeq 0,823 \cdot 10$$

L'addition est donc non associative par suite des erreurs d'arrondi. *En générale, dans une sommations de réels, l'erreur a la tendance à être minimisé lorsqu'on somme en premier les termes ayant la plus petite valeur absolue.*

1.3 Phénomènes de compensation.

Lorsqu'on tente d'effectuer des soustractions de valeurs très voisines, on peut avoir des pertes importantes de précision.

Exemple. On veut résoudre l'équation $x^2 - 1634x + 2 = 0$ en effectuant les calculs avec $N = 10$ chiffres significatifs. On obtient

$$\Delta' = 667487, \quad \sqrt{\Delta'} = 816,9987760,$$
$$x_1 = 817 + \sqrt{\Delta'} \simeq 1633,998776,$$
$$x_2 = 817 - \sqrt{\Delta'} \simeq 0,0012240.$$

On voit qu'on a une perte de 5 chiffres significatifs sur x_2. Ici le remède est simple : il suffit d'observer que $x_1 \cdot x_2 = 2$, d'où

$$x_2 = \frac{2}{x_1} \simeq 1,223991125 \cdot 10^{-3}.$$

C'est donc l'algorithme numérique utilisé qui doit être modifié.

2 Systèmes linéaires.

On appelle système linéaire d'ordre n (n entier positif), une expression de la forme

$$A\mathbf{x} = \mathbf{b},$$

où $A = (a_{ij})$, $1 \leq i, j \leq n$, désigne une matrice de taille $n \times n$ de nombres réels ou complexes, $\mathbf{b} = (b_i)$, $1 \leq i \leq n$, un vecteur colonne réel ou complexe et $\mathbf{x} = (x_i)$, $1 \leq i \leq n$, est le vecteur des inconnues du système. La relation précédente équivaut aux équations

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, \ldots, n.$$

La matrice A est dite *régulière* (ou *non singulière*) si $detA \neq 0$; on a existence et unicité de la solution $\mathbf{x}$ (pour n'importe quel vecteur $\mathbf{b}$ donné) si et seulement si la matrice associée au système linéaire est régulière.
Théoriquement, si A est non singulière, la solution est donnée par la *formule de Cramer* :

$$x_i = \frac{det(A_i)}{det(A)}, \qquad i = 1, \ldots, n,$$

où A_i est la matrice obtenue en replaçant la i-ème colonne de A par le vecteur $\mathbf{b}$. Cependant l'application de cette formule est inacceptable pour la résolution pratique des systèmes, car son coût est de l'ordre de $(n+1)!$ *floating-point operations* (*flops*). En fait, le calcul de chaque déterminant par la formule

$$\det(A) = \sum_{\sigma} (-1)^{\epsilon(\sigma)} \prod_{i=1}^{n} a_{i,\sigma(i)}$$

(où la somme est étendue à toutes les permutations σ sur n objets) requiert $n!$ flops. Par example, sur un ordinateur effectuant 10^9 flops par seconde il faudrait $9,6 \cdot 10^{47}$ années pour résoudre un système linéaire de seulement 50 équations. Il faut donc développer des algorithmes alternatives avec un coût raisonnable. Dans les sections suivantes plusieurs méthodes sont analysées.
On appelle méthode de résolution **directe** d'un système linéaire un algorithme qui, si l'ordinateur faisait des calculs exacts, donnerait la solution en un nombre fini d'opérations. Il existe aussi des méthodes **itératives** qui consistent à construire une suite de vecteurs $\mathbf{x}_n$ convergeant vers la solution $\mathbf{x}$.

2.1 Méthodes directes.

Pour résoudre $A\mathbf{x} = \mathbf{b}$, on cherche à écrire $A = LU$ où
- L est une matrice triangulaire inférieure avec des 1 sur la diagonale,
- U est une matrice triangulaire supérieure.

La résolution de $A\mathbf{x} = \mathbf{b}$ est alors ramenée aux résolutions successives des systèmes échelonnés $L\mathbf{y} = \mathbf{b}$ et $U\mathbf{x} = \mathbf{y}$.

2.1.1 Résolution des systèmes triangulaires.

Une matrice $A = (a_{ij})$ est *triangulaire supérieure* si

$$a_{ij} = 0 \quad \forall i, j \ : \ 1 \leq j < i \leq n$$

et *triangulaire inférieure* si

$$a_{ij} = 0 \quad \forall i, j \ : \ 1 \leq i < j \leq n.$$

Suivant ces cas, le système à résoudre est dit *système triangulaire supérieur ou inférieur*. Si la matrice A est régulière et triangulaire alors, comme $\det(A) = \prod a_{ii}$, on en déduit que $a_{ii} \neq 0$, pour tout $i = 1, \ldots, n$.
Si A est triangulaire inférieure on a

$$x_1 = b_1/a_{11},$$

et pour $i = 2, 3, \ldots, n$

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

Cet algorithme est appelé *méthode de descente*.
Si A est triangulaire supérieure on a

$$x_n = b_n/a_{nn},$$

et pour $i = n-1, n-2, \ldots, 2, 1$

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

Cet algorithme est appelé *méthode de remontée*.
Le nombre de multiplications et de divisions nécessaires dans cet algorithme est de $n(n+1)/2$ et le nombre d'additions et de soustraction est de $n(n-1)/2$, donc l'algorithme nécessite de n^2 flops.

2.1.2 Méthode d'elimination de Gauss et décomposition LU.

Soit $A = (a_{ij})$ une matrice non singulière $n \times n$. L'*algorithme de Gauss* est le suivant : on pose $A^{(1)} = A$, c'est-à-dire $a_{ij}^{(1)} = a_{ij}$ pour $i, j = 1, \ldots, n$; pour $k = 1, \ldots, n$ on calcule

$$\begin{aligned} l_{ik} &= \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \qquad i = k+1, \ldots, n; \\ a_{ij}^{(k+1)} &= a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}, \qquad i = k+1, \ldots, n. \end{aligned}$$

A la fin de ce procédé, les éléments de la matrice U sont définis par

$$u_{ij} = a_{ij}^{(i)},$$

tandis que les éléments de la matrice L sont les termes l_{ij} engendrés par l'algorithme de Gauss. Le coût de cette méthode de factorisation est de l'ordre de $2n^3/3$ flops. En fait, pour passer de $A^{(k)}$ à $A^{(k+1)}$ on modifie touts les elements de $A^{(k)}$ sauf les premières k lignes et les premières k colonnes, qui ne changent

pas ; pour chaque élément de $A^{(k+1)}$ il faut faire une multiplication et une soustraction ; on a donc $2(n-k)^2$ opérations à faire. Au total, pour fabriquer $A^{(n)}$ il faut

$$\sum_{k=1}^{n-1} 2(n-k)^2 = 2\sum_{j=1}^{n-1} j^2 = 2\frac{(n-1)n(2n-1)}{6} \sim \frac{2n^3}{3}.$$

Remarque. *Pour que l'algorithme de Gauss puisse terminer, il faut que tous les termes $a_{kk}^{(k)}$, qui correspondent aux termes diagonaux $u_{kk}^{(k)}$ de la matrice U et qu'on appelle* ***pivots****, soient non nuls.*

Le fait que la matrice A ait entrées non nulles ne prévient pas l'apparition de pivot nuls, comme on remarque dans l'exemple qui suit :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix} \qquad A^{(2)} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & -1 \\ 0 & -6 & -12 \end{bmatrix}$$

Cependant on a :

Proposition 2.1 *La factorisation LU de la matrice carrée A de taille n par la méthode de Gauss existe si et seulement si les sous-matrices principales $A_i = (a_{hk})$, $h, k = 1, \ldots, i$ sont non-singulières. Cette propriété est satisfaite en particulier dans les cas qui suivent :*
1. A est une matrice symétrique définie positive ;
2. A est une matrice à diagonale dominante stricte par lignes, c'est-à-dire pour tout $i = 1, \ldots, n$

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|.$$

3. A est une matrice à diagonale dominante stricte par colonnes, c'est-à-dire pour tout $j = 1, \ldots, n$

$$|a_{jj}| > \sum_{i=1, j\neq i}^{n} |a_{ij}|.$$

Si A est non singulière, sa factorisation LU est unique.

Grace à la factorisation LU on peut calculer le determinant d'une matrice carrée avec $O(n^3)$ operations, vu que

$$\det(A) = \det(L)\det(U) = \det(U) = \prod_{k=1}^{n} u_{kk},$$

puisque le determinant d'une matrice triangulaire est le produit des termes diagonaux. La commande MATLAB `det(A)` exploite ce procédé.

2.1.3 Elimination avec recherche de pivot.

Si au cours de l'algorithme d'élimination de Gauss on trouve un pivot nul à l'étape k, le procédé s'arrête. On peut alors permuter la ligne k avec la ligne $r > k$. La matrice qui permute les lignes k et r d'une matrice quelconque est la

matrice qui a des éléments égaux à 1 partout sur la diagonale sauf $p_{rr} = p_{kk} = 0$ et des éléments égaux à 0 partout ailleurs, sauf $p_{rk} = p_{kr} = 1$:

$$P^{(k,r)} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ \vdots & & & & \vdots \\ \dots & 0 & \dots & 1 & \dots \\ \vdots & & & & \vdots \\ \dots & 1 & \dots & 0 & \dots \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix} \begin{matrix} 1 \\ \vdots \\ k \\ \vdots \\ r \\ \vdots \\ n \end{matrix}$$

On aboutit alors à la factorisation d'une nouvelle matrice, c'est-à-dire

$$LU = PA,$$

où P est le produit de toutes les matrices de permutation (des lignes) qu'on a utilisées.
Supposons donc qu'on a trouvé une factorisation du type $PA = LU$. Alors le vecteur $\mathbf{x}$, solution du système $A\mathbf{x} = \mathbf{b}$, vérifie également $PA\mathbf{x} = \tilde{\mathbf{b}}$, où $\tilde{\mathbf{b}} = P\mathbf{b}$. Ce deuxième système s'écrit $LU\mathbf{x} = \tilde{\mathbf{b}}$. Donc, comme auparavant, on résout par $L\mathbf{y} = \tilde{\mathbf{b}}$ et $U\mathbf{x} = \mathbf{y}$.

Si un pivot est très petit, sans pour autant être nul, le procédé d'élimination ne s'arrête pas, mais le résultat peut être entaché d'erreurs considérables, comme on va le voir dans l'exemple ci-dessous.
Soit ϵ un petit nombre réel et considérons le système de deux équations à deux inconnues

$$\begin{aligned} \epsilon x + y &= 1, \\ x + y &= 2. \end{aligned}$$

Par élimination sans recherche de pivot, nous obtenons le système équivalent suivant

$$\begin{aligned} \epsilon x + y &= 1, \\ \left(1 - \frac{1}{\epsilon}\right) y &= 2 - \frac{1}{\epsilon}. \end{aligned}$$

On a donc immédiatement la valeur de y :

$$y = \frac{1 - 2\epsilon}{1 - \epsilon} \simeq 1.$$

Par substitution, on obtient x :

$$x = \frac{1 - y}{\epsilon} \simeq 0.$$

Essayons maintenant l'élimination après avoir échangé les lignes dans le système :

$$\begin{aligned} x + y &= 2, \\ \epsilon x + y &= 1. \end{aligned}$$

Le même procédé d'élimination que précédemment conduit au système équivalent suivant

$$\begin{aligned} x + y &= 2, \\ (1-\epsilon)y &= 1 - 2\epsilon, \end{aligned}$$

qui devient, en flottants,

$$\begin{aligned} x + y &= 2, \\ y &= 1. \end{aligned}$$

La solution, tout à fait acceptable cette fois-ci, est

$$x = 1 \text{ et } y = 1.$$

L'erreur qu'on faisait avec la première élimination venait de ce qu'on divisait un nombre par le petit pivot ϵ, ce qui amplifie considérablement les erreurs.

L'algorithme de décomposition de Gauss avec *pivotation par lignes* permute deux lignes de la matrice $A^{(k)}$ à chaque pas de la décomposition afin que l'élément diagonale $a_{kk}^{(k)}$ de la matrice permutée soit maximal (en valeur absolue). Précisément, au pas k de la décomposition on trouve l'index r, avec $r \geq k$, tel que

$$|a_{rk}^{(k)}| = \max_{s \geq k} |a_{sk}^{(k)}|$$

et on échange les lignes r et k entre elles. Le logiciel MATLAB implémente ce dernier algorithme dans la commande `lu`.

2.1.4 Matrices tridiagonales.

On considère le cas particulier où la matrice A est non singulière et tridiagonale, donnée par

$$\begin{bmatrix} a_1 & c_1 & & 0 \\ b_2 & a_2 & \ddots & \\ & \ddots & & c_{n-1} \\ 0 & & b_n & a_n \end{bmatrix}.$$

Dans ce cas, les matrices L et U de la factorisation LU de A sont des matrices bidiagonales de la forme :

$$L = \begin{bmatrix} 1 & & & 0 \\ \beta_2 & 1 & & \\ & \ddots & \ddots & \\ 0 & & \beta_n & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} \alpha_1 & c_1 & & 0 \\ & \alpha_2 & \ddots & \\ & & \ddots & c_{n-1} \\ 0 & & & \alpha_n \end{bmatrix}.$$

Les coefficients α_i et β_i peuvent être calculés par les relations :

$$\alpha_1 = a_1,\ \beta_2\alpha_1 = b_2 \Rightarrow \beta_2 = \frac{b_2}{\alpha_1},\ \beta_2 c_1 + \alpha_2 = a_2 \Rightarrow \alpha_2 = a_2 - \beta_2 c_1, \ldots$$

Donc, on a

$$\alpha_1 = a_1,\ \beta_i = \frac{b_i}{\alpha_{i-1}},\ \alpha_i = a_i - \beta_i c_{i-1}, \qquad i = 2, \ldots, n.$$

Cet algorithme est connu sous le nom de *algorithme de Thomas*, et le nombre d'opérations effectuées est de l'ordre de n.
Avec l'algorithme de Thomas, résoudre un système tridiagonal $A\mathbf{x} = \mathbf{f}$ revient à résoudre 2 systèmes bidiagonaux $L\mathbf{y} = \mathbf{f}$ et $U\mathbf{x} = \mathbf{y}$ avec

$$\begin{aligned} L\mathbf{y} = \mathbf{f} \quad &\leftrightarrow \quad y_1 = f_1,\ y_i = f_i - \beta_i y_{i-1}, \qquad i = 2, \ldots, n, \\ U\mathbf{x} = \mathbf{y} \quad &\leftrightarrow \quad x_n = \frac{y_n}{\alpha_n},\ x_i = \frac{y_i - c_i x_{i+1}}{\alpha_i}, \qquad i = n-1, \ldots, 1. \end{aligned}$$

Le coût de calcul ici est de $3(n-1)$ opérations pour la factorisation et de $5n-4$ opérations pour la substitution, pour une totale de $8n-7$ opérations.

2.1.5 L'inverse d'une matrice.

Si A est une matrice $n \times n$ non singulière, notons $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(n)}$ les colonnes de sa matrice inverse A^{-1}, i.e. $A^{-1} = (\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(n)})$.
La relation $AA^{-1} = I$ se traduit par les n systèmes suivants :

$$A\mathbf{v}^{(k)} = \mathbf{e}^{(k)}, \qquad 1 \le k \le n, \tag{1}$$

où $\mathbf{e}^{(k)}$ est le vecteur colonne ayant tous les éléments 0 sauf celui sur la ligne k qui est 1,

$$\mathbf{e}^{(k)} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} 1 \\ \vdots \\ k \\ \vdots \\ n \end{matrix} .$$

Une fois connues les matrices L et U qui factorisent la matrice A, résoudre les n systèmes (1) gouvernés par la même matrice A, coûte n^3 opérations.
Si la matrice A est triangulaire, la solution brute de ce système s'avère inefficient puisque A^{-1} a beaucoup d'entrées nulles. En effet, le système precedent équivaut à

$$A^{(j)}\mathbf{v}'_j = \mathbf{1}_j,$$

où $A^{(j)}$ est la sous-matrice principale d'ordre j, $\mathbf{v}'_j = (v_{1j}, \ldots, v_{jj})^T$ et $\mathbf{1}_j = (0, 0, \ldots, 0, 1)^T$ sont des vecteurs de longueur j.

2.1.6 Considération sur la précision des méthodes directes pour les systèmes linéaires.

Il y a des cas où les méthodes qu'on vient de voir ne marchent pas trop bien.

Définition. *On définit le* ***conditionnement*** *d'une matrice M symétrique définie positive comme le rapport entre la valeur maximale et la valeur minimale de ses valeurs propres, i.e.*

$$K_2(M) = \frac{\lambda_{max}(M)}{\lambda_{min}(M)}.$$

On peut montrer que plus le conditionnement de la matrice est grand, plus la solution du système linéaire obtenue par une méthode directe peut être mauvaise.
Par exemple, considérons un système linéaire

$$A\mathbf{x} = \mathbf{b}.$$

Si on résout ce système avec un ordinateur, à cause des erreurs d'arrondis on ne trouve pas la solution exacte mais une solution approchée $\hat{\mathbf{x}}$. On s'attend, alors, à ce que la solution $\hat{\mathbf{x}}$ soir très proche de $\mathbf{x}$. Toutefois, on peut montrer la relation suivante :

$$\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq K_2(A) \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}$$

où $\mathbf{r}$ est le résidu $\mathbf{r} = \mathbf{b} - A\hat{\mathbf{x}}$; on a noté $\|\cdot\|$ la norme euclidienne d'un vecteur. On remarque que, si le conditionnement de A est grande, la distance $\|\mathbf{x} - \hat{\mathbf{x}}\|$ entre la solution exacte et celle calculée numériquement peut être très grande même si le résidu est très petit.
Avec MATLAB on peut calculer le conditionnement d'une matrice avec la commande `cond`.

2.2 Méthodes itératives.

L'idée des méthodes itératives est de construire une suite de vecteurs $\mathbf{x}^{(k)}$ qui converge vers le vecteur $\mathbf{x}$, solution du système $A\mathbf{x} = \mathbf{b}$,

$$\mathbf{x} = \lim_{k \to \infty} \mathbf{x}^{(k)}.$$

L'intérêt des méthodes itératives, comparées aux méthodes directes, est d'être simples à programmer et de nécessiter moins de place en mémoire. En revanche le temps de calcul est souvent plus long.

Une stratégie est de considérer la relation de récurrence linéaire

$$\mathbf{x}^{(k+1)} = B\mathbf{x}^{(k)} + \mathbf{g}, \tag{2}$$

où B est la *matrice d'itération* de la méthode itérative (dépendant de A) et $\mathbf{g}$ est un vecteur (dépendant de $\mathbf{b}$), tels que

$$\mathbf{x} = B\mathbf{x} + \mathbf{g}.$$

Étant $\mathbf{x} = A^{-1}\mathbf{b}$, on obtient $\mathbf{g} = (I - B)A^{-1}\mathbf{b}$; la méthode itérative (2) est donc complètement définie par la matrice B.
En définissant l'erreur au pas k comme

$$\mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)},$$

on obtient la relation de récurrence

$$\mathbf{e}^{(k)} = B\mathbf{e}^{(k-1)}, \quad \text{et donc} \quad \mathbf{e}^{(k)} = B^{(k)}\mathbf{e}^{(0)}, \quad k = 0, 1, \ldots$$

On démontre que $\lim_{k \to \infty} \mathbf{e}^{(k)} = 0$ pour tout $\mathbf{e}^{(0)}$ (et donc pour tout $\mathbf{x}^{(0)}$) si et seulement si $\rho(B) < 1$, où $\rho(B)$ est le *rayon spectral* de la matrice B, défini comme

$$\rho(B) = \max |\lambda_i(B)|,$$

et $\lambda_i(B)$ sont les valeur propres de la matrice B.

Une technique générale pour construire des méthodes itératives est basée sur une décomposition (*splitting*) de la matrice A sous la forme $A = P - N$, où P et N sont des matrices à déterminer avec P non singulière. La matrice P est appelée *matrice de préconditionnement*. Plus précisément, $\mathbf{x}^{(0)}$ étant donné, on peut calculer $\mathbf{x}^{(k)}$, pour $k \geq 1$, en résolvant le système

$$P\mathbf{x}^{(k+1)} = N\mathbf{x}^{(k)} + \mathbf{b}, \quad k \geq 0. \tag{3}$$

Clairement, la solution exacte $\mathbf{x}$ satisfait $P\mathbf{x} = N\mathbf{x} + \mathbf{b}$ et donc $A\mathbf{x} = \mathbf{b}$.
Le système (3) peut être écrit également sous la forme (2), avec $B = P^{-1}N$, et $\mathbf{g} = P^{-1}\mathbf{b}$.
Une relation de récurrence équivalente à (3) est la suivante :

$$P(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = \mathbf{r}^{(k)}, \quad k \geq 0, \tag{4}$$

où $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$ est le *résidu* à l'itération k. On peut généraliser la relation précédente comme

$$P(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = \alpha_k\mathbf{r}^{(k)}, \quad k \geq 0, \tag{5}$$

où on a introduit un paramètre α_k (qui peut être différent à chaque itération k) afin d'accélérer la convergence. Cette méthode est dite de *Richardson*.
Les relations (3), (4) et (5) montrent qu'on doit résoudre un système linéaire de matrice P à chaque itération ; donc P doit être telle que la résolution du système ait un coût raisonnable. Par exemple, on pourra choisir P diagonale ou triangulaire.

2.2.1 La méthode de Jacobi.

On remarque que si les éléments diagonaux de A sont non nuls, le système linéaire $A\mathbf{x} = \mathbf{b}$ est équivalent à :

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j\right), \qquad i = 1, \ldots, n.$$

Pour une donnée initiale $\mathbf{x}^{(0)}$ choisie, on calcule $\mathbf{x}^{(k+1)}$ par

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

Cela permet d'identifier le *splitting* suivant pour A :

$$\begin{array}{lcl} A & = & D - E - F \\ D & : & \text{diagonale de } A \\ E & : & \text{triangulaire inférieure avec des 0 sur la diagonale} \\ F & : & \text{triangulaire supérieure avec des 0 sur la diagonale} \end{array}$$

La matrice d'itération de la méthode de Jacobi est donnée par :

$$B_J = D^{-1}(E + F) = I - D^{-1}A.$$

L'algorithme de Jacobi nécessite le stockage des deux vecteurs $x_j^{(k)}$ et $x_j^{(k+1)}$.

2.2.2 La méthode de Gauss-Seidel.

Pour cette méthode on a

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

Il s'écrit aussi

$$\mathbf{x}^{(k+1)} = (D - E)^{-1} \big(b + F \mathbf{x}^{(k)} \big)$$

Dans ce cas, le *splitting* de A est

$$P = D - E, \quad N = F,$$

et la matrice d'itération associée est

$$B_{GS} = (D - E)^{-1} U.$$

L'algorithme de Gauss-Seidel ne nécessite qu'un vecteur de stockage, $\mathbf{x}^{(k)}$ étant remplacé par $\mathbf{x}^{(k+1)}$ au cours de l'itération. Il est en général plus rapide que l'algorithme de Jacobi, donc préférable.

2.2.3 Convergence des méthodes de Jacobi et de Gauss-Seidel.

On a les résultats suivants

– *Si A est une matrice à diagonale dominante stricte par lignes, alors les méthodes de Jacobi et de Gauss-Seidel sont convergentes.*

Démonstration. Nous prouvons la partie concernant la méthode de Jacobi. Soit $\mathbf{x}$ la solution du système linéaire, on a

$$x_i^{(n+1)} - x_i = \frac{-1}{a_{ii}} \sum_{k \neq i} a_{ik} \left(x_k^{(n)} - x_k \right)$$

donc

$$\begin{aligned} \max_i \left| x_i^{(n+1)} - x_i \right| &\leq \frac{1}{a_{ii}} \sum_{k \neq i} |a_{ik}| \max_k \left| x_k^{(n)} - x_k \right| \\ &\leq K \max_k \left| x_k^{(n)} - x_k \right| \text{ avec } 0 \leq K < 1 \end{aligned}$$

car A est à diagonale strictement dominante. De plus $\max_i |y_i| = \|y\|_\infty$ est une norme sur R^N. On a ainsi montré que

$$\left\| x^{(n+1)} - \mathbf{x} \right\|_\infty \leq K \left\| x^{(n)} - \mathbf{x} \right\|_\infty$$

ce qui prouve la convergence.□

Remarque : en pratique la stricte dominance n'est pas indispensable. L'inégalité large suffit pour la plupart des matrices inversibles.

– *Si A est une matrice symétrique définie positive, alors la méthode de Gauss-Seidel converge (la méthode de Jacobi pas forcément).*

2.2.4 Méthode du gradient à pas optimal.

Considérons la méthode itérative (5) avec $\alpha_k \neq 1$. Si P et A sont symétriques définies positives, alors on a un critère optimal pour le choix de α_k :

$$\alpha_k = \frac{\langle \mathbf{r}^{(k)}, \mathbf{z}^{(k)} \rangle}{\langle A\mathbf{z}^{(k)}, \mathbf{z}^{(k)} \rangle}, \qquad k \geq 0,$$

où $\mathbf{z}^{(k)} = P^{-1}\mathbf{r}^{(k)}$. Cette méthode est appelée du *gradient préconditionné* (du gradient lorsque $P = I$). On a noté par $\langle \mathbf{x}, \mathbf{y} \rangle$ le produit scalaire entre les vecteurs $\mathbf{x}$ et $\mathbf{y}$.

Démonstration. D'une part on a

$$\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)} = A\big(\mathbf{x} - \mathbf{x}^{(k)}\big) = A\mathbf{e}^{(k)}$$

où $\mathbf{e}^{(k)}$ représente l'erreur à l'étape k, et d'autre part

$$\begin{aligned} \mathbf{e}^{(k+1)} &= \mathbf{e}^{(k+1)}(\alpha) = \big(I - \alpha P^{-1}A\big)\mathbf{e}^{(k)}, \\ \mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)}(\alpha) - \alpha A\mathbf{z}^{(k)}. \end{aligned}$$

Ainsi, en notant avec $\|\cdot\|_A$ la norme vectorielle issue du produit scalaire $\langle \mathbf{x}, \mathbf{y} \rangle_A = \langle A\mathbf{x}, \mathbf{y} \rangle$, c'est-à-dire $\|\mathbf{x}\|_A = \langle A\mathbf{x}, \mathbf{x} \rangle^{1/2}$ on tire que

$$\begin{aligned} \|\mathbf{e}^{(k+1)}\|_A^2 &= \langle A\mathbf{e}^{(k+1)}, \mathbf{e}^{(k+1)} \rangle = \langle \mathbf{r}^{(k+1)}, \mathbf{e}^{(k+1)} \rangle \\ &= \langle \mathbf{r}^{(k)}, \mathbf{e}^{(k)} \rangle - \alpha\big(\langle \mathbf{r}^{(k)}, P^{-1}A\mathbf{e}^{(k)} \rangle + \langle A\mathbf{z}^{(k)}, \mathbf{e}^{(k)} \rangle\big) + \alpha^2 \langle A\mathbf{z}^{(k)}, P^{-1}A\mathbf{e}^{(k)} \rangle. \end{aligned}$$

Maintenant on choisi α comme le α_k qui minimise $\|\mathbf{e}^{(k+1)}\|_A$, c'est-à-dire

$$\frac{d}{d\alpha} \|\mathbf{e}^{(k+1)}\|_A^2 \bigg|_{\alpha = \alpha_k} = 0.$$

On obtient donc

$$\alpha_k = \frac{\langle \mathbf{r}^{(k)}, \mathbf{z}^{(k)} \rangle}{\langle A\mathbf{z}^{(k)}, \mathbf{z}^{(k)} \rangle}.$$

□

On peut démontrer que la suite $\big\{\mathbf{x}^{(k)}\big\}$ donnée par la méthode du gradient converge vers $\mathbf{x}$ lorsque $k \to \infty$, et

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}\|_A \leq \left(\frac{K_2(P^{-1}A) - 1}{K_2(P^{-1}A) + 1}\right)^k \|\mathbf{x}^{(0)} - \mathbf{x}\|_A, \qquad k \geq 0, \qquad (6)$$

En général, on choisit P de façon à avoir

$$K_2(P^{-1}A) << K_2(A).$$

En conséquence,

$$\frac{K_2(P^{-1}A) - 1}{K_2(P^{-1}A) + 1} << \frac{K_2(A) - 1}{K_2(A) + 1}.$$

2.2.5 Méthode du gradient conjugué.

Une méthode encore plus rapide dans le cas où P et A sont symétriques définies positives est celle du *gradient conjugué préconditionné*. On pose cette fois-ci

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}, \quad k \geq 0,$$

avec $\mathbf{p}^{(k)} = \mathbf{z}^{(k)} - \beta_k \mathbf{p}^{(k-1)}$.
Soit $\mathbf{x}^{(0)}$ une donnée initiale; on calcule $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$, $\mathbf{z}^{(0)} = P^{-1}\mathbf{r}^{(0)}$, $\mathbf{p}^{(0)} = \mathbf{z}^{(0)}$, puis pour $k \geq 0$,

$$\begin{aligned}
\alpha_k &= \frac{\langle \mathbf{r}^{(k)}, \mathbf{p}^{(k)} \rangle}{\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \rangle} \\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)} \\
\mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} - \alpha_k A\mathbf{p}^{(k)} \\
\mathbf{z}^{(k+1)} &= P^{-1}\mathbf{r}^{(k+1)} \\
\beta_{k+1} &= \frac{\langle A\mathbf{p}^{(k)}, \mathbf{z}^{(k+1)} \rangle}{\langle A\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \rangle} \\
\mathbf{p}^{(k+1)} &= \mathbf{z}^{(k+1)} - \beta_k \mathbf{p}^{(k)}.
\end{aligned}$$

Dans ce cas la formule (6) devient

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}\|_A \leq 2 \left(\frac{\sqrt{K_2(P^{-1}A)} - 1}{\sqrt{K_2(P^{-1}A)} + 1} \right)^k \|\mathbf{x}^{(0)} - \mathbf{x}\|_A, \qquad k \geq 0.$$

2.3 Exercices.

Exercice 1. On veut résoudre le système linéaire $A\mathbf{x} = \mathbf{b}$ où

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 5 \\ 4 & 6 & 8 \end{bmatrix} \qquad \text{et} \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 5 \end{bmatrix}$$

par la méthode d'élimination de Gauss.
a) Vérifier que l'algorithme de Gauss sans pivoting ne peut pas être exécuté jusqu'au bout.
b) Trouver une matrice de permutation P telle que la matrice PA soit factorisable.
c) Calculer la factorisation LU de la matrice PA.
d) Résoudre le système linéaire $A\mathbf{x} = \mathbf{b}$ en remplaçant PA par LU et en utilisant les algorithmes de substitution progressive et rétrograde.

Exercice 2. 1) Pour une matrice quelconque A montrer que $\rho(A) \leq \|A\|$ pour toute norme matricielle. Donner l'expression de $\|A\|_\infty$.
2) On suppose A symétrique définie positive. Montrer que $\rho(A) = \|A\|_2$.

Soit

$$A = \begin{bmatrix} 1 & a & a \\ a & 1 & a \\ a & a & 1 \end{bmatrix}$$

3) Pour quelles valeurs de a A est-elle définie positive ?

4) Écrire la matrice J de l'itération de Jacobi.
5) Pour quelles valeurs de a la méthode de Jacobi converge-t-elle ?
6) Écrire la matrice G de l'itération de Gauss-Seidel.
7) Calculer $\rho(G)$. Pour quelles valeurs de a cette méthode converge-t-elle plus vite que celle de Jacobi ?

Exercice 3. Soit A une matrice décomposée en $A = D - E - F$

$$A = \begin{pmatrix} \ddots & & -F \\ & D & \\ -E & & \ddots \end{pmatrix}$$

pour résoudre $Ax = b$ on propose la méthode

$$\left(\frac{D}{\omega} - E\right) x^{(k+1)} = \left(\frac{1-\omega}{\omega} D + F\right) x^{(k)} + b \quad \omega \in R^*.$$

a) Vérifier que si la méthode converge, elle converge vers une solution de $Ax = b$.
b) Donner la matrice d'itération L_ω de cette méthode.
c) Calculer $\det(L_\omega)$
d) En déduire que $\rho(L_\omega) \geq |1 - \omega|$. Conclusion ?

Exercice 4. On considère la matrice

$$A = \begin{bmatrix} 3 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix}$$

On se donne également un vecteur b de R^3. On note $\overline{x}$ l'unique solution du système $Ax = b$.
a) Montrer que la méthode de Jacobi pour résoudre le système linéaire $Ax = b$ est ici convergente.
b) Soit x_n le $n^{\text{ième}}$ itéré de la méthode de Jacobi en partant de $x_0 = 0$. Déterminer la matrice d'itération M de la méthode de Jacobi (c'est la matrice qui vérifie $x_{n+1} - \overline{x} = M(x_n - \overline{x})$).
c) Calculer les valeurs propres de A. En déduire les valeurs propres de M. Déterminer s le module de la plus petite valeur propre en module de A, et S le module de la plus grande valeur propre en module de M. (Vérifier que $S < 1$!)
d) Montrer que $|\overline{x}| \leq \frac{1}{s} |b|$ où $|u|$ désigne la norme euclidienne usuelle du vecteur u dans R^3.
e) Montrer que $|x_n - \overline{x}| \leq \frac{S^n}{s} |b|$.

3 Zeros de fonctions non-linéaires.

Soit $f : \Omega$ fermé $\subset R^p \to R^p$. On cherche $\overline{x} \in \Omega$ tel que $f(\overline{x}) = 0$. Nous supposerons qu'un moyen (une étude graphique, une raison physique, un théorème, l'intuition ou ... la chance !) nous ont permis de voir que $\overline{x}$ était l'unique solution dans Ω. Nous supposerons aussi que f est au moins continue sur Ω.
En général, les méthodes de résolution de $f(\overline{x}) = 0$ sont des méthodes itératives. Elles consistent à construire une suite x_n convergente (le plus rapidement possible) vers $\overline{x}$.

3.1 Méthode de dichotomie (ou de la bissection).

Cette méthode est utilisée pour calculer les zéros d'une fonction continue $f : \mathbb{R} \to \mathbb{R}$. S'il existe a, b, $a < b$, avec $f(a)f(b) < 0$, on sait alors qu'il existe au moins un zéro $\overline{x}$ de f dans l'intervalle (a, b). Alors on pose $a_0 = a$, $b_0 = b$, $I_0 = (a_0, b_0)$ et $x_0 = (a_0 + b_0)/2$. Pour $n \geq 1$, on choisit le sous-intervalle $I_n = (a_n, b_n)$ de l'intervalle $I_{n-1} = (a_{n-1}, b_{n-1})$ de la façon suivante :
a) soit $x_{n-1} = (a_{n-1} + b_{n-1})/2$;
b) si $f(x_{n-1}) = 0$, alors $\overline{x} = x_{n-1}$ et la méthode termine ;
c) si $f(x_{n-1}) \neq 0$, alors
– si $f(a_{n-1}) \cdot f(x_{n-1}) < 0$, on pose

$$a_n = a_{n-1}, \quad b_n = x_{n-1};$$

– si $f(x_{n-1}) \cdot f(b_{n-1}) < 0$, on pose

$$a_n = x_{n-1}, \quad b_n = b_{n-1};$$

d) on définit $I_n = (a_n, b_n)$; on augmente k de 1 et on recommence du point a).

3.2 Méthode de Newton.

Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction donnée et x_0 un point donné. On considère la droite $y(x)$ qui passe par le point $(x_n, f(x_n))$ et qui a comme pente $f'(x_n)$:

$$y(x) = f'(x_n)(x - x_n) + f(x_n).$$

On définit x_{n+1} comme étant le point où cette droite intersecte l'axe $0x$, c'est à dire $y(x_{n+1}) = 0$. On en déduit que :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n \geq 0. \tag{7}$$

3.3 Méthode de la sécante.

Dans certaines situations, la dérivée de f' est très compliquée ou même impossible à expliciter. On ne peut alors utiliser telle quelle la méthode de Newton. L'idée est de remplacer f' par le taux d'accroissement de f sur un petit intervalle :

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

On notera que l'algorithme itératif ne peut démarrer que si on dispose déjà de deux valeurs approchées x_0, x_1 de $\overline{x}$.

3.4 Méthode de la corde.

Cette méthode est obtenue en replaçant $f'(x_n)$ par un q fixé dans la formule de Newton :

$$x_{n+1} = x_n - \frac{f(x_n)}{q}, \quad k \geq 0.$$

On peut prendre par exemple $q = f'(x_0)$, ou bien $q = \dfrac{f(b) - f(a)}{b - a}$, dans le cas où l'on cherche un zéro dans l'intervalle $[a, b]$.

3.5 Méthode de point fixe.

Un procédé général pour trouver les racines d'une équation non-linéaire $f(x) = 0$ consiste en la transformer en un problème équivalent $x = \phi(x)$, où la fonction auxiliaire $\phi : [a, b] \to \mathbb{R}$ doit avoir la propriété suivante :

$$\alpha = \phi(\alpha) \quad \text{ssi} \quad f(\alpha) = 0.$$

3.5.1 Le théorème du point fixe.

Soit (E, d) un espace métrique complet et $\phi : E \to E$ une application continue. On dit que $\alpha \in E$ est un *point fixe* de ϕ si $\phi(\alpha) = \alpha$. On dit que ϕ est *contractante* si ϕ est lipschitzienne de rapport $K < 1$, c'est-à-dire s'il existe $K < 1$ tel que

$$\forall x, y, \in E, \quad d(\phi(x) - \phi(y)) \leq K d(x - y).$$

Théorème. *Soit $\phi : E \to E$ une application contractante d'un espace métrique complet dans lui-même. Alors ϕ admet un point fixe unique $\alpha \in E$. De plus, pour tout point initial $x_0 \in E$ la suite itérée $\{x_n\}$ définie par $x_{n+1} = \phi(x_n)$ converge vers α.*

Démonstration.

1. Unicité : soient x_1 et x_2, deux solutions. On a :

   $$d(x_1 - x_2) \leq K d(x_1 - x_2)$$

   et $K < 1 \Rightarrow x_1 = x_2$.

2. Existence : puisque ϕ est $K-$Lipschitzienne, on a

   $$d(x_{n+1} - x_n) \leq K d(x_n - x_{n-1})$$

   donc

   $$d(x_{n+1} - x_n) \leq K^n d(x_1 - x_0) \tag{8}$$

   Calculons maintenant, pour $p > n$

   $$\sum_{k=n+1}^{p} x_k - x_{k-1} = x_p - x_n$$

   par (8), on obtient

   $$\begin{aligned} d(x_p - x_n) &\leq \sum_{k=n+1}^{p} K^{k-1} d(x_1 - x_0) \leq K^n d(x_1 - x_0) \sum_{k=0}^{\infty} K^k \\ &= \frac{K^n}{1 - K} d(x_1 - x_0) \to 0 \text{ quand } (n, p) \to \infty. \end{aligned}$$

La suite $\{x_n\}$ est donc de Cauchy dans E, donc convergente dans E. De plus, soit α la limite de cette suite ; par continuité de ϕ, $\phi(\alpha) = \alpha$, d'où l'existence du point fixe.□

Il est important de disposer d'un critère pratique assurant qu'une fonction ϕ est K-Lipschitzienne, avec $K < 1$.

Théorème. *Soit ϕ une fonction differentiable sur $[a,b] \subset \mathbb{R}$. Si $\|\phi'(x)\| \leq K$ pour tout x dans $[a,b]$, alors ϕ est K-Lipschitzienne sur $[a,b]$.*

Démonstration. Soient x et y dans $[a,b]$. Puisque le segment $[x,y]$ est inclus dans $[a,b]$, on a

$$\phi(y) - \phi(x) = \int_0^1 \phi'(x + t(x-y))(x-y)dt$$

donc

$$\begin{aligned} \|\phi(y) - \phi(x)\| &\leq \int_0^1 \left\|\phi'(x + t(x-y))(x-y)\right\| dt \\ &\leq \int_0^1 \left\|\phi'(x + t(x-y))\right\| \|(x-y)\| dt \\ &\leq K \|x-y\| . \end{aligned}$$

□

3.5.2 Point fixes attractifs et répulsifs.

Soit $[a,b] \subset \mathbb{R}$ et $\phi : [a,b] \to [a,b]$ une application de classe $\mathcal{C}^1$. Soit $\alpha \in [a,b]$ un point fixe de ϕ. On peut distinguer trois cas :

1. $|\phi'(\alpha)| < 1$.

   Soit K tel que $|\phi'(\alpha)| < K < 1$. Par continuité de ϕ', il existe un intervalle $E = [\alpha - h, \alpha + h]$ sur lequel $|\phi'| \leq K$, donc ϕ est contractante de rapport K sur E ; on a nécessairement $\phi(E) \subset E$ et par conséquent

   $$\forall x_0 \in E, \quad \lim_{n \to +\infty} x_n = \alpha.$$

   On dit que α est un *point fixe attractif*. Dans ce cas la convergence de la suite $\{x_n\}$ est au moins exponentiellement rapide : $|x_n - \alpha| \leq K^n |x_0 - \alpha|$.

   Cas particulier : $|\phi'(\alpha)| = 0$.
   Supposons de plus que $\phi \in \mathcal{C}^2$ et que $|\phi''| \leq M$ sur E. La formule de Taylor donne

   $$\begin{aligned} \phi(x) &= \phi(\alpha) + (x-\alpha)\phi'(\alpha) + \frac{(x-\alpha)^2}{2!}\phi''(c) \\ &= \alpha + \frac{1}{2}\phi''(c)(x-\alpha)^2, \quad c \in]\alpha, x[, \end{aligned}$$

   d'où $|\phi(x) - \alpha| \leq \frac{1}{2}M|x-\alpha|^2$, soit encore $\frac{1}{2}M|\phi(x) - \alpha| \leq \left[\frac{1}{2}M|x-\alpha|\right]^2$. Par récurrence on déduit successivement

   $$\begin{aligned} \frac{1}{2}M|x_n - \alpha| &\leq \left[\frac{1}{2}M|x_0 - \alpha|\right]^{2^n}, \\ |x_n - \alpha| &\leq \frac{2}{M}\left[\frac{1}{2}M|x_0 - \alpha|\right]^{2^n}. \end{aligned}$$

La convergence est donc ici extraordinairement rapide. Ce phénomène est appelé *convergence quadratique*, et α est alors appelé parfois *point fixe superattractif*.

2. $|\phi'(\alpha)| > 1$.

   Il existe alors un voisinage $[\alpha - h, \alpha + h]$ tel que

   $$\forall x \in [\alpha - h, \alpha + h] \setminus \{\alpha\}, \quad |\phi(x) - \alpha| > |x - \alpha|.$$

   On dit alors que α est un *point fixe répulsif*. Dans ce cas, ϕ' est de signe constant au voisinage de α, donc il existe $h > 0$ tel que la restriction $\phi|_{[\alpha-h,\alpha+h]}$ admette une application réciproque ϕ^{-1} définie sur $\phi([\alpha - h, \alpha + h])$, qui est un intervalle contenant $\phi(\alpha) = \alpha$. L'équation $\phi(x) = x$ peut se récrire $x = \phi^{-1}(x)$ au voisinage de α, et comme $(\phi^{-1})'(\alpha) = 1/\phi'(\alpha)$, le point α est un point fixe attractif pour ϕ^{-1}.

3. $|\phi'(\alpha)| = 1$.

   On est ici dans un cas douteux, comme le montrent les deux exemples suivants dans lesquels $\alpha = 0$, $\phi'(\alpha) = 1$:

   Exemple 1. $\phi(x) = \sin x$, $x \in [0, \pi/2]$. On a ici $\sin x < x$ pour tout $x \in]0, \pi/2]$. Pour tout $x_0 \in]0, \pi/2]$ la suite $\{x_n\}$ est strictement décroissante minorée, donc convergente vers $l = \sin l$, d'où $l = 0$.

   Exemple 2. $\phi(x) = \sinh x$, $x \in [0, +\infty]$. Comme $\sinh x > x$ pour tout $x > 0$, on voit que le point fixe 0 est répulsif.

En général, la méthode du point fixe, quand elle converge, est une méthode d'ordre 1. C'est à dire que, en gros, à chaque itération l'erreur est multipliée par un coefficient (plus petit que un). Voici un énoncé plus précis.

Proposition. *Supposons que la méthode du point fixe converge et que ϕ est de classe C^2, alors*

$$x_{n+1} - \overline{x} = \phi'(\overline{x})(x_n - \overline{x}) + O(\|x_n - \overline{x}\|^2)$$

en particulier, en dimension un d'espace, si $x_n \neq \overline{x}$,

$$\frac{x_{n+1} - \overline{x}}{x_n - \overline{x}} \to \phi'(\overline{x}).$$

Nous sommes conduits à la définition suivante.

Definition. *Une méthode du point fixe est d'ordre p ssi*

$$\frac{x_{n+1} - \overline{x}}{(x_n - \overline{x})^p} \to l \text{ avec } 0 < |l| < +\infty$$

Si $p = 1$ on parle de convergence *linéaire*, si $p = 2$ la convergence est dite *quadratique*.

3.6 Encore à propos de la méthode de Newton.

La méthode de Newton est une méthode de point fixe pour la fonction

$$\psi(x) = x - \frac{f(x)}{f'(x)}.$$

Soit α un zéro pour la fonction f. On remarque que $\phi'(\alpha) = 0$ lorsque $f'(\alpha) \neq 0$.

Théorème. *Si $f \in \mathcal{C}^2$, $f(\alpha) = 0$ et $f'(\alpha) \neq 0$, alors il existe $\delta > 0$ telle que, si $|x_0 - \alpha| \leq \delta$, la suite définie dans (7) converge vers α. De plus, la convergence est quadratique :*

$$\lim_{n\to\infty} \frac{x_{n+1} - \alpha}{(x_n - \alpha)^2} = \frac{f''(\alpha)}{2f'(\alpha)}.$$

Définition. *On dit que un zéro α de f est de multiplicité p, $p \in \mathbb{N}$, si*

$$f(\alpha) = \ldots = f^{(p-1)}(\alpha) = 0 \quad et \quad f^{(p)}(\alpha) \neq 0.$$

Remarque. Si $f'(\alpha) = 0$, la convergence de la méthode de Newton est seulement linéaire. On considère alors la méthode de Newton modifiée :

$$x_{n+1} = x_n - p\frac{f(x_n)}{f'(x_n)}, \quad n \geq 0$$

avec p la multiplicité de α.

Quand faut-il terminer les itérations de l'algorithme de Newton ? Un bon critère d'arrêt est le contrôle de l'*incrément* : les itérations s'achèvent dés que

$$|x_{n+1} - x_n| < \epsilon \tag{9}$$

où ϵ est une tolérance fixée. En fait, si on note $e_n = \alpha - x_n$ l'erreur à l'itération n, on a

$$e_{n+1} = \alpha - x_{n+1} = \phi(\alpha) - \phi(x_n) = \phi'(\xi_n)e_n,$$

avec ξ_n entre x_n et α, et

$$x_{n+1} - x_n = \alpha - x_n - \alpha + x_{n+1} = e_n - e_{n+1} = (1 - \phi'(\xi_n))e_n.$$

Or, si n est suffisamment grand, $\phi'(\xi_n) \approx \phi'(\alpha) = 0$ et donc

$$e_n \approx x_{n+1} - x_n.$$

L'erreur qu'on commet lorsque l'on adopte le critère (9) est donc plus petite que la tolérance fixée.

3.7 Le cas des systèmes.

On considère le système de deux équations non-linéaires suivant :

$$\begin{cases} f_1(x_1, x_2) = 0, \\ f_2(x_1, x_2) = 0. \end{cases}$$

Ce système peut s'écrire sous la forme $\mathbf{f} = \mathbf{0}$, où $\mathbf{f} = (f_1, f_2)$. On veut généraliser la méthode de Newton à ce cas. Pour cela on définit la matrice Jacobienne du vecteur $\mathbf{f}$:

$$D\mathbf{f}(\mathbf{x} = (x_1, x_2)) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}.$$

Alors la méthode de Newton pour systèmes non linéaires s'écrit : *étant donné* $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)})$, *on calcule pour* $n \geq 0$ *:*

$$[D\mathbf{f}(\mathbf{x}^{(n)})](\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}) = -\mathbf{f}(\mathbf{x}^{(n)}).$$

Les mêmes résultats que dans le cas scalaire sont valables dans le cas vectoriel. On se ramène ainsi à la résolution des systèmes linéaires gouvernés par des matrices $A_n = D\mathbf{f}(\mathbf{x}^{(n)})$.

4 Approximation polynômiale.

Les fonctions les plus faciles à évaluer numériquement sont les polynômes. Il est donc important de savoir approximer une fonction arbitraire par des polynômes. Dans la suite, on désignera par $\mathcal{P}_n$ l'espace vectoriel des fonctions polynômes sur $\mathbb{R}$ à coefficients réels, de degré inférieur ou égal à n. On a donc $\dim\mathcal{P}_n = n+1$.

Soit $f \in \mathcal{C}^0([a,b])$ et $x_0, x_1, \ldots, x_n \in [a,b]$ $n+1$ points distincts ($n \geq 0$ est un nombre entier). On cherche un polynôme p de degré n tel que

$$p(x_i) = f(x_i), \qquad 0 \leq i \leq n.$$

dans le cas affirmatif, on note $p = p_n$ et on appelle p_n le *polynôme d'interpolation* de f aux points $x_0, x_1, \ldots, x_n$.

4.1 Méthode d'interpolation de Lagrange.

On considère les polynômes ϕ_i, $i = 0, \ldots, n$ de degré n tels que

$$\phi_i(x_j) = \delta_{ij}, \quad i,j = 0, \ldots, n,$$

où $\delta_{ij} = 1$ si $i = j$ et $\delta_{ij} = 0$ si $i \neq j$. Explicitement on a

$$\phi_i(x) = \prod_{i \neq j} \frac{(x - x_j)}{(x_i - x_j)}, \quad 0 \leq i \leq n.$$

Alors le polynôme d'interpolation de f aux points $x_0, x_1, ldots, x_n$ s'écrit

$$p_n(x) = \sum_{i=0}^{n} f(x_i)\phi(x). \tag{10}$$

On montre maintenant que p_n est le seul polynôme de degré n interpolant f aux points x_i. Soit, en effet, $q_n(x)$ un autre polynôme d'interpolation. Alors on a

$$q_n(x_i) - p_n(x_i) = 0, \quad i = 0, \ldots, n.$$

Donc $q_n - p_n$ est un polynôme de degré n qui s'annule un $n+1$ points distincts ; il en suit que $q_n = p_n$, d'où l'unicité du polynôme interpolant. En conclusion, on a le théorème suivant.

Théorème. *Le problème d'interpolation $p(x_i) = f(x_i)$, $0 \leq i \leq n$, admet une solution et une seule, donnée par la formule* (10).

On peut aussi voir les choses différemment. Si $p(t) = a_0 + a_1 x + \cdots + a_n x^n$, les coefficients de p sont solutions de

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & & & \vdots \\ 1 & x_N & & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}.$$

La matrice de ce système linéaire est une matrice de Vandermonde de déterminant $\prod_{i<j}(x_i - x_j) \neq 0$ car les points x_i sont distincts. La résolution de ce

système conduit bien sur à l'unique polynôme d'interpolation de f. Cependant si l'on tente de résoudre ce système **numériquement** grâce à la méthode LU (avec ou sans pivotage), les résultats obtenus n'ont rien à voir avec la solution exacte. L'erreur peut atteindre, suivant la machine avec laquelle on travaille, plusieurs centaines de milliers ! Le système de Vandermonde appartient à la catégorie (heureusement plutôt rare) des systèmes linéaires **mal conditionnés**. On appelle ainsi un système pour lequel une petite erreur sur les coefficients ou le second membre entraîne une erreur importante sur la solution du système. Il est possible de savoir si un système $Ax = b$ est bien ou mal conditionné en connaissant sont **conditionnement**, défini par

$$cond(A) = \|A\| \left\|A^{-1}\right\|$$

où $\|\cdot\|$ est une norme matricielle quelconque. On a toujours $cond(A) \geq 1$, et plus ce nombre est grand plus la résolution du système est difficile. Si par exemple $x_i = \frac{i}{2}0, i = 0, \ldots, 20$, on a $cond(A) = 356551625901350880$. Par ailleurs, $cond(A) = 1$ ssi A est une matrice orthogonale ($A^{-1} = A^T$). Les systèmes linéaires les mieux conditionnés sont donc ceux dans lesquels intervient une matrice orthogonale.

4.2 Méthode des différences divisées.

Soit p_k le polynôme d'interpolation de f aux points $x_0, x_1, \ldots, x_k$. On désigne par $f[x_0, x_1, \ldots, x_k]$ le coefficient directeur du polynôme p_k.
Alors $p_k - p_{k-1}$ est un polynôme de degré $\leq k$, s'annulant aux points $x_0, x_1, \ldots, x_{k-1}$, et admettant $f[x_0, x_1, \ldots, x_k]$ pour coefficient directeur. Par suite :

$$p_k(x) - p_{k-1}(x) = f[x_0, x_1, \ldots, x_k](x - x_0)\ldots(x - x_{k-1}).$$

Comme $p_0(x) = f(x_0)$, on en déduit la formule

$$p_n(x) = f(x_0) + \sum_{k=1}^{n} f[x_0, x_1, \ldots, x_k](x - x_0)\ldots(x - x_{k-1}). \tag{11}$$

Pour évaluer les coefficients $f[x_0, x_1, \ldots, x_k]$ on utilise une recurrence sur le nombre k de points x_i, en observant que $f[x_0] = f(x_0)$. Pour $k \geq 1$ on a

$$f[x_0, x_1, \ldots, x_k] = \frac{f[x_1, \ldots, x_k] - f[x_0, \ldots, x_{k-1}]}{x_k - x_0}. \tag{12}$$

Démonstration. Si $q_{k-1} \in \mathcal{P}_{k-1}$ est le polynôme d'interpolation de f aux points $x_1, x_2, \ldots, x_k$, on a

$$p_k = \frac{(x - x_0)q_{k-1}(x) - (x - x_k)p_{k-1}(x)}{x_k - x_0}.$$

On obtient la formule (12) en égalant les coefficients de x^k dans l'identité précédente. □

Algorithme pratique. On range les valeurs de $f(x_i)$ dans un tableau TAB, puis on modifie ce tableau en n étapes successives, en procédant par indices

décroissantes :

Tableau	Etape 0	Etape 1	Etape 2	...	Etape n
TAB$[n]$	$f(x_n)$	$f[x_{n-1}, x_n]$	$f[x_{n-2}, x_{n-1}, x_n]$	...	$f[x_0, \ldots, x_n]$
TAB$[n-1]$	$f(x_{n-1})$	$f[x_{n-2}, x_{n-1}]$			
TAB$[n-2]$	$f(x_{n-2})$				
$\vdots$	$\vdots$	$\vdots$	$\vdots$		
TAB[2]	$f(x_2)$	$f[x_1, x_2]$	$f[x_0, x_1, x_2]$		
TAB[1]	$f(x_1)$	$f[x_0, x_1]$			
TAB[0]	$f(x_0)$				

A l'issue de la n-ème étape, la case TAB$[k]$ contient le coefficient $f[x_0, \ldots, x_k]$, et on peut utiliser la formule (11). On construit p_n par récurrence descendante (règle de Hörner)

$$\begin{aligned} u_n &= \text{TAB}[n] \\ u_k &= \text{TAB}[k] + (x - x_k)u_{k+1}, \quad 0 \le k < n, \end{aligned}$$

qui donne $u_0 = p_n(x)$.

4.3 Formule d'erreur.

Théorème. *Soient $x_0, x_1, \ldots, x_n$, $n+1$ points distincts dans $[a, b]$ et soit $f \in \mathcal{C}^{n+1}([a, b])$. Alors pour tout $x \in [a, b]$*

$$f(x) - p_n(x) = \frac{1}{(n+1)!}\pi_{n+1}(x)f^{(n+1)}(\xi)$$

où $\pi_{n+1}(x) = \prod_i (x - x_i)$, $\xi \in [a, b]$.

Démonstration. Si $x = x_i$, on a $\pi_{n+1}(x_i) = 0$, tout point ξ convient. Si x est distinct des points x_i, soit $p_{n+1}(t)$ le polynôme d'interpolation de f aux points $x, x_0, \ldots, x_{n+1}$. Par construction $f(x) - p_n(x) = p_{n+1}(x) - p_n(x)$. Or le polynôme $p_{n+1} - p_n$ est de degré $\le n+1$ et s'annule aux $n+1$ points $x_0, \ldots, x_{n+1}$. On a donc

$$p_{n+1}(t) - p_n(t) = c\pi_{n+1}(t), \quad c \in \mathbb{R}.$$

Considérons la fonction

$$g(t) = f(t) - p_{n+1}(t) = f(t) - p_n(t) - c\pi_{n+1}(t).$$

Cette fonction s'annule en $n+2$ points, donc d'après une extension du théorème de Rolle il existe ξ tel que $g^{(n+1)}(\xi) = 0$. Or

$$p_n^{(n+1)} = 0, \quad \pi_{n+1}^{(n+1)} = (n+1)!$$

On a par conséquent $g^{(n+1)}(\xi) = f^{(n+1)}(\xi) - c(n+1)! = 0$, d'où la valeur de c. □

En prenant la borne supérieure de la valeur absolue des deux membres dans la formule d'erreur, on obtient en particulier :

$$\|f - p_n\| \le \frac{1}{(n+1)!}\|\pi_{n+1}\| \, \|f^{(n+1)}\|.$$

Cela montre que la taille de l'erreur d'interpolation $f(x) - p_n(x)$ dépend à la fois de la quantité $\|f^{(n+1)}\|$, qui peut être grande si f oscille trop vite, et de la quantité $\|\pi_{n+1}\|$, qui est liée à la repartition des points x_i dans l'intervalle $[a, b]$. Dans le cas particulier où les nœdus sont équirepartis on a le résultat suivant

$$\|f - p_n\| \leq \frac{1}{4(n+1)} \left(\frac{b-a}{n}\right)^{n+1} \|f^{(n+1)}\|.$$

Démonstration. On peut montrer que le maximum de $|\pi_{n+1}(x)|$ est atteint toujours dans un de deux intervalles extrêmes $[x_0, x_1]$ ou $[x_{n-1}, x_n]$. On prend alors $x \in [x_0, x_1]$ (l'autre cas est similaire) ; on a

$$|(x - x_0)(x - x_1)| \leq \frac{(x_1 - x_0)^2}{4} = \frac{h^2}{4}$$

où l'on a noté $h = (b-a)/n$. De plus, pour tout $i > 1$

$$|(x - x_i)| \leq ih.$$

Donc

$$\max_{x \in [a,b]} \prod_i |(x - x_i)| \leq \frac{h^2}{4} 2h3h \dots nh = \frac{n!}{4} \left(\frac{b-a}{n}\right)^{n+1}$$

d'où la conclusion. □

Remarque. Le seul fait que

$$\lim_{n \to \infty} \frac{1}{4(n+1)} \left(\frac{b-a}{n}\right)^{n+1} = 0$$

n'implique pas que $\|f - p_n\|$ tend vers zéro quand $n \to \infty$ (voir le *phénomène de Runge*).

Remèdes :
1- Interpolation avec points qui ne sont pas équirepartis (interpolation de Tchebychev).
2- Interpolation par intervalles.

4.4 Interpolation de Tchebychev.

On définit les polynômes de Tchebychev par

$$t_n(x) = \cos(n \arccos x), \quad x \in [-1, 1].$$

En effet, si on pose $\theta = \arccos x$, i.e. $x = \cos\theta$ avec $\theta \in [0, \pi]$, on a

$$\begin{aligned} t_n(x) &= \cos(n\theta), \\ t_{n+1}(x) + t_{n-1}(x) &= \cos((n+1)\theta) + \cos((n-1)\theta) \\ &= 2\cos(n\theta)\cos\theta = 2xt_n(x). \end{aligned}$$

La fonction t_n se calcule donc par les formules de récurrence

$$\begin{aligned} &t_0(x) = 1, \quad t_1(x) = x, \\ &t_{n+1}(x) = 2xt_n(x) - t_{n-1}(x). \end{aligned}$$

Il en résulte que t_n est un polynôme de degré n, dont le coefficient directeur est 2^{n-1} si $n \geq 1$. Les n racines de t_n sont données par

$$\hat{x}_i = \cos \frac{2i+1}{2n}\,\pi \in]-1,1[, \quad 0 \leq i \leq n-1.$$

Les points $\hat{x}_i$ sont répartis symétriquement autour de 0 ($\hat{x}_{n-i} = -\hat{x}_i$) de façon plus dense au voisinage de 1 et -1.
Pour se ramener à un intervalle $[a,b]$ quelconque, on utilisera le changement de variable suivant

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\,\hat{x}_i.$$

Le polynôme π_{n+1} est donné par

$$\pi_{n+1}(x) = \prod_{i=0}^{n}(x - x_i) = \left(\frac{b-a}{2}\right)^{(n+1)} \prod_{i=0}^{n}(\hat{x} - \hat{x}_i)$$

où $\prod_{i=0}^{n}(\hat{x} - \hat{x}_i) = 1/2^n t_{n+1}$ est le polynôme $\pi_{n+1}(\hat{x})$ correspondant à $[-1,1]$. Donc on a ici

$$\|\pi_{n+1}\| = 2\left(\frac{b-a}{4}\right)^{(n+1)}.$$

Cette valeur est plus petite que l'estimation

$$\frac{n!}{4}\left(\frac{b-a}{n}\right)^{n+1} \sim \frac{\sqrt{2\pi}\,e}{4\sqrt{n}}\left(\frac{b-a}{e}\right)^{n+1} \tag{13}$$

obtenue pour $\|\pi_{n+1}\|$ avec des points x_i équidistants, surtout lorsque n est assez grand : le rapport entre les deux est de l'ordre $\mathcal{O}(\sqrt{n}\,(e/4)^{n+1})$, qui tend vers 0 quand $n \to \infty$. L'approximation (13) repose sur la formule de Stirling $n! \sim \sqrt{2\pi n}\,(n/e)^n$.

4.5 Interpolation par intervalles.

Soient $x_0 = a < x_1 < \ldots < x_N = b$ des points qui divisent l'intervalle $I = [a,b]$ dans une réunion d'intervalles $I_j = [x_j, x_{j+1}]$ de longueur H, où

$$H = \frac{b-a}{N}.$$

Sur chaque sous-intervalle I_j on interpole $f_{|I_j}$ par un polynôme de degré n. Le polynôme par morceaux qu'on obtient est noté p_n^H.

Théorème. *Si $f \in \mathcal{C}^{(n+1)}(I)$ alors*

$$\|f(x) - p_n^H(x)\| \leq \frac{H^{(n+1)}}{4(n+1)n^{n+1}}\|f^{(n+1)}\|.$$

5 Intégration numérique.

On souhaite disposer d'un moyen d'évaluer numériquement $I(f) = \int_a^b f(t)dt$ où f est une fonction continue sur un intervalle $[a, b]$ avec $a < b$. En effet, dans de nombreuses applications, cette intégrale ne se ramène pas à des fonctions simples, et même si c'est le cas, on cherche une méthode simple et utilisable dans le cas général. Pour cela, nous chercherons une **approximation** de $I(f)$, notée $J(f)$ sous la forme

$$J(f) = (b-a)\sum_{i=0}^{n} \omega_i f(\xi_i)$$

où les points ξ_i sont dans l'intervalle $[a, b]$ et $\sum_{i=0}^{n} \omega_i = 1$. Les ξ_i sont appelés les **points d'intégration** et les coefficients ω_i les **poids d'intégration**.

Definition. *On dit qu'un méthode de quadrature est d'ordre N si la formule approchée est exacte pour tout $f \in \mathcal{P}_N$ et inexacte pour au moins un $f \in \mathcal{P}_{N+1}$.*

On observera que les formules sont toujours exactes pour $f(x) = 1$ à cause de l'hypothèse $\sum_{i=0}^{n} \omega_i = 1$. Par linéarité, elles sont donc exactes au moins pour $f \in \mathcal{P}_0$.

5.1 Exemples.

(a) **Un seul point.** On choisit un seul point $\xi \in [a, b]$ et on remplace f sur $[a, b]$ par le polynôme de degré 0 : $p_0(x) = f(\xi)$. On a alors

$$\int_a^b f(x)dx \simeq (b-a)f(\xi).$$

Voici les choix plus courants :
- $\xi = a$: méthode des rectangles à gauche (ordre 0) ;
- $\xi = b$: méthode des rectangles à droite (ordre 0) ;
- $\xi = \frac{a+b}{2}$: *méthode du point milieu* (ordre 1) ;

(b) **Interpolation linéaire.** On choisit $\xi_0 = a$ et $\xi_1 = b$ et on remplace f sur $[a, b]$ par la fonction linéaire p_1 qui interpole f aux points a, b :

$$p_1(x) = \frac{(x-a)f(b) - (x-b)f(a)}{b-a}.$$

On obtient la formule suivante, dite *méthode des trapèzes* (ordre 1) :

$$\int_a^b f(x)dx \simeq (b-a)\frac{f(a)+f(b)}{2}.$$

(b) **Méthodes de Newton-Cotes.** On choisit $n+1$ points équidistants

$$\xi_i = a + i\frac{b-a}{n}.$$

Pour déterminer la formule de quadrature élémentaire, on se ramène par changement de variable à l'intervalle $[-1,1]$, subdivisé par les points $\tau_i = -1 + i\frac{2}{n}$. Le polynôme d'interpolation d'une fonction $f \in \mathcal{C}([-1,1])$ est donné par

$$p_n(x) = \sum_{i=0}^{n} f(\tau_i)\phi_i(x),$$

où ϕ_i est le polynôme de base de Lagrange $\phi_i(x) = \prod_{j\neq i} \frac{x-\tau_j}{\tau_i - \tau_j}$. On a donc

$$\int_{-1}^{1} f(x)dx \simeq \int_{-1}^{1} p_l(x)dx = 2\sum_{i=0}^{n} \omega_i f(\tau_i)$$

avec $\omega_i = \frac{1}{2}\int_{-1}^{1} \phi_i(x)dx$. Par suite de la symétrie des points τ_i autour de 0, on a

$$\tau_{n-i} = -\tau_i, \quad \phi_{n-i}(x) = \phi_i(-x), \quad \omega_{n-i} = \omega_i.$$

Après changement de variable, les coefficients ω_i sont inchangés, donc on obtient la formule

$$\int_a^b f(x)dx \simeq (b-a)\sum_{i=0}^{n} \omega_i f(\xi_i).$$

Si $f \in \mathcal{P}_n$, alors $p_n = f$, donc la méthode de Newton-Cotes de rang n est d'ordre $\geq n$. De plus, lorsque $f \in \mathcal{C}([-1,1])$ est un polynôme impair, on a

$$\int_{-1}^{1} f(x)dx = 0 = 2\sum_{i=0}^{n} \omega_i f(\tau_i).$$

Si n est pair, les formules sont donc encore exactes pour $f(x) = x^{n+1}$, et plus généralement pour $f \in \mathcal{P}_{n+1}$ par linéarité. On démontre en fait le résultat suivant que nous admettrons :

Proposition. *Si n est pair, l'ordre de la méthode de Newton-Cotes de rang n est $n+1$, si n est impair, l'ordre est n.*

Ceci fait que, hormis le cas $n = 1$, les méthodes de Newton-Cotes ne sont utilisées que pour n pair :
– $n = 1$ méthode des trapèzes (ordre 1)

$$\omega_0 = \omega_1 = \frac{1}{2}$$

– $n = 2$ méthode de Simpson (ordre 3)

$$\omega_0 = \omega_2 = \frac{1}{6}, \quad \omega_1 = \frac{2}{3}$$

– $n = 4$ méthode de Boole-Villarceau (ordre 5)

$$\omega_0 = \omega_4 = \frac{7}{90}, \quad \omega_1 = \omega_3 = \frac{16}{45}, \quad \omega_2 = \frac{2}{15}$$

– $n = 6$ méthode de Weddle-Hardy (ordre 7)

$$\omega_0 = \omega_6 = \frac{41}{840}, \quad \omega_1 = \omega_5 = \frac{9}{35}, \quad \omega_2 = \omega_4 = \frac{9}{280}, \quad \omega_3 = \frac{34}{105}.$$

Pour $n \geq 8$ il apparaît des coefficients $\omega_i < 0$, ce qui a pour effet de rendre les formules beaucoup plus sensibles aux erreurs d'arrondis. Les méthodes de Newton-Cotes ne sont donc utilisées en pratique que dans les 4 cas ci-dessus.

5.2 Evaluation de l'erreur. Noyau de Peano.

Théorème. *On suppose que la méthode est d'ordre $N \geq 0$. Si f est de classe $\mathcal{C}^{N+1}$ sur $[a,b]$, alors*

$$E(f) = \int_a^b f(x)dx - (b-a)\sum_{i=0}^{n} \omega_i f(x_i) = \frac{1}{N!}\int_a^b K_N(t) f^{(N+1)}(t)dt,$$

où K_N est une fonction sur $[a,b]$, appelée noyau de Peano associé à la méthode, définie par

$$K_N(t) = E\left(x \mapsto (x-t)_+^N\right), \quad t \in [a,b].$$

Démonstration. On observe d'abord que si $g : (x,t) \mapsto g(x,t)$ est une fonction integrable sur $[a,b] \times I$, le théorème de Fubini implique par ailleurs

$$E\left(x \mapsto \int_I g(x,t)dt\right) = \int_I E\left(x \mapsto g(x,t)\right) dt.$$

La formule de Taylor avec reste intégral donne

$$f(x) = p_N(x) + \int_a^b \frac{1}{N!}(x-t)_+^N f^{(N+1)}(t)dt.$$

Comme $p_N \in \mathcal{P}_N$, on a $E(p_N) = 0$ par hypotèse, d'où

$$\begin{aligned} E(f) &= E\left(x \mapsto \int_a^b \frac{1}{N!}(x-t)_+^N f^{(N+1)}(t)dt\right) \\ &= \int_a^b E\left(x \mapsto \frac{1}{N!}(x-t)_+^N f^{(N+1)}(t)\right) dt \\ &= \int_a^b \frac{1}{N!} f^{(N+1)}(t) \cdot E\left(x \mapsto (x-t)_+^N\right) dt \\ &= \frac{1}{N!}\int_a^b K_N(t) f^{(N+1)}(t)dt. \end{aligned}$$

□

On déduit la majoration

$$E(f) \leq \frac{1}{N!}\|f^{(N+1)}\|_\infty \cdot \int_a^b |K_N(t)|dt.$$

6 Equations différentielles.

On considère une fonction continue $f : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}$. Pour $y_0 \in \mathbb{R}$ donné, on cherche $y : t \in \mathbb{R}_+ \to y(t) \in \mathbb{R}$ qui satisfait le problème suivant, appelé *problème de Cauchy* :

$$\begin{cases} y'(t) = f(t, y(t)) & \text{si } t > 0 \\ y(0) = y_0. \end{cases} \tag{14}$$

Théorème (Cauchy-Lipschitz). *Si f est continue sur $\mathbb{R}_+ \times \mathbb{R}$ et s'il existe une constante $L > 0$ telle que*

$$|f(t, x_1) - f(t, x_2)| \leq L|x_1 - x_2| \quad \forall x_1, x_2 \in \mathbb{R}, \forall t > 0,$$

alors le problème de Cauchy (14) *admet une solution globale (c'est-à-dire pour tout $t > 0$) et elle est unique.*

6.1 Dérivée numérique.

Soit $y : [a, b] \to \mathbb{R}$ de classe $\mathcal{C}^1$ et $a = t_0, t_1, \ldots, t_n = b$, $n+1$ nœdus equirépartis dans $[a, b]$. On note $h = (b - a)/n$ la distance entre deux nœdus consécutifs. La dérivée $y'(t_i)$ est donnée par

$$\begin{aligned} y'(t_i) &= \lim_{h \to 0+} \frac{y(t_i + h) - y(t_i)}{h}, \\ &= \lim_{h \to 0+} \frac{y(t_i) - y(t_i - h)}{h}, \\ &= \lim_{h \to 0+} \frac{y(t_i + h) - y(t_i - h)}{2h}. \end{aligned}$$

Soit maintenant $(Dy)_i$ une approximation de $y'(t_i)$. On appelle
– *différence finie progressive* l'approximation

$$(Dy)_i^P = \frac{y(t_{i+1}) - y(t_i)}{h}, \quad i = 0, \ldots, n-1;$$

– *différence finie rétrograde* l'approximation

$$(Dy)_i^R = \frac{y(t_i) - y(t_{i-1})}{h}, \quad i = 1, \ldots, n;$$

– *différence finie centrée* l'approximation

$$(Dy)_i^C = \frac{y(t_{i+1}) - y(t_{i-1})}{2h}, \quad i = 1, \ldots, n-1.$$

Si $y \in \mathcal{C}^2(\mathbb{R})$, pour tout $t \in \mathbb{R}$, il existe un η entre t_i et t tel que l'on a le développement de Taylor

$$y(t) = y(t_i) + y'(t_i)(t - t_i) + \frac{y''(\eta)}{2}(t - t_i)^2.$$

– Pour $t = t_{i+1}$ on obtient pour la différence finie progressive

$$(Dy)_i^P = y'(t_i) + \frac{h}{2} y''(\eta),$$

ce qui conduit à une estimation du type

$$|y'(t_i) - (Dy)_i^P| \leq Ch,$$

où $C = \frac{1}{2}\max_{t\in[t_i,t_{i+1}]} |y''(t)|$.

– Pour $t = t_{i-1}$ on obtient pour la différence finie retrograde

$$(Dy)_i^R = y'(t_i) - \frac{h}{2}y''(\eta),$$

ce qui conduit à une estimation du type

$$|y'(t_i) - (Dy)_i^R| \leq Ch,$$

où $C = \frac{1}{2}\max_{t\in[t_{i-1},t_i]} |y''(t)|$.

– Pour $t = t_{i+1}$ et $t = t_{i-1}$ avec un développement d'ordre 2 (si $y \in \mathcal{C}^3$)

$$\begin{aligned} y(t_{i+1}) &= y(t_i) + y'(t_i)h + \frac{y''(t_i)}{2}h^2 + \frac{y'''(\eta_1)}{6}h^3, \\ y(t_{i-1}) &= y(t_i) - y'(t_i)h + \frac{y''(t_i)}{2}h^2 - \frac{y'''(\eta_2)}{6}h^3, \end{aligned}$$

on obtient

$$(Dy)_i^C = y'(t_i) + \frac{y'''(\eta_1) + y'''(\eta_2)}{12}h^2,$$

et donc l'estimation suivante

$$|y'(t_i) - (Dy)_i^C| \leq Ch^2,$$

où $C = \frac{1}{6}\max_{t\in[t_{i-1},t_{i+1}]} |y'''(t)|$.

Definition. *La différence $|y'(t_i) - (Dy)_i^P|$ (et celles correspondantes aux autres différences finies) est appelée* ***erreur de troncature*** *au point t_i.*

L'erreur de troncature est d'ordre 1 pour les formules progressive et retrograde et d'ordre 2 pour la formule centrée.

6.2 Méthodes d'Euler.

Soient $0 = t_0 < t_1 < \ldots < t_n < t_{n+1} < \ldots$ une suite de nombres réels equirépartis. On note $h = t_{n+1} - t_n$. On notera par

$$y_n \text{ une approximation de } y(t_n).$$

Dans le problème de Cauchy (14), pout $t = t_n$ on a

$$y'(t_n) = f(t_n, y(t_n)).$$

On approche la dérivée $y'(t_n)$ en utilisant des schémas de *dérivation numérique.*

Schéma d'Euler progressif :

$$\begin{cases} \frac{y_{n+1}-y_n}{h} = f(t_n, y_n) \quad \text{pour } n = 0, 1, 2, \ldots \\ y_0. \end{cases}$$

Schéma d'Euler rétrograde :

$$\begin{cases} \frac{y_{n+1}-y_n}{h} = f(t_{n+1}, y_{n+1}) \quad \text{pour } n = 0, 1, 2, \ldots \\ y_0. \end{cases}$$

6.3 Etude générale des méthodes à un pas.

Les méthodes à un pas sont les méthodes de résolution numérique qui peuvent s'écrire sous la forme

$$y_{n+1} = y_n + h_n \Phi(t_n, y_n, h_n), \quad 0 \leq n < N,$$

où Φ est une fonction qu'on supposera continue.

6.3.1 Consistance, stabilité, convergence.

Definition. *L'**erreur de consistance** e_n relative à une solution exacte y est l'erreur*

$$e_n = y(t_{n+1}) - y_{n+1}, \quad 0 \leq n < N,$$

en supposant $y_n = y(t_n)$. On a donc

$$e_n = y(t_{n+1}) - y(t_n) - h_n \Phi(t_n, y(t_n), h_n).$$

*On dit que la méthode est **consistante** si pour toute solution exacte y la somme des erreurs de consistance relatives à y, soit $\sum_n |e_n|$, tend vers 0 quand h_{max} tend vers 0.*

Une autre notion fondamentale est la notion de stabilité. dans la pratique, le calcul récurrent des points y_n est entaché d'erreurs d'arrondis ε_n. Pour que les calculs soient significatifs, il est indispensable que la propagation de ces erreurs reste contrôlable.

Definition. *On dit que la méthode est **stable** s'il existe une constante $S \geq 0$ telle que pour toutes suites (y_n), $(\tilde{y}_n)$ définies par*

$$\begin{aligned} y_{n+1} &= y_n + h_n \Phi(t_n, y_n, h_n), && 0 \leq n < N, \\ \tilde{y}_{n+1} &= \tilde{y}_n + h_n \Phi(t_n, \tilde{y}_n, h_n) + \varepsilon_n, && 0 \leq n < N, \end{aligned}$$

on ait

$$\max_n |\tilde{y}_n - y_n| \leq S \left(|\tilde{y}_0 - y_0| + \sum_n |\varepsilon_n| \right)$$

Une dernière notion importante est la suivante

Definition. *On dit que la méthode est **convergente** si pour toute solution exacte y, la suite (y_n) vérifie*

$$\max_n |y_n - y(t_n)| \to 0$$

quand $y_0 \to y(0)$ et $h_{max} \to 0$.

Posons $\tilde{y}_n = y(t_n)$. Par definition d'erreur de consistance on a

$$\tilde{y}_{n+1} = \tilde{y}_n + h_n \Phi(t_n, \tilde{y}_n, h_n) + e_n.$$

Si la méthode est stable, de constante S, l'erreur globale est donc

$$\max_n |y_n - y(t_n)| \leq S \left(|y_0 - y(0)| + \sum_n |e_n| \right).$$

Corollaire. *Si la méthode est stable et consistante, elle est convergente.*

Nous énonçons sans les démontrer les résultats suivants.

Théorème (CNS de consistance). *La méthode à 1 pas définie par la fonction Φ est consistante ssi*

$$\Phi(t, y, 0) = f(t, y), \quad \forall (t, y) \in \mathbb{R}^+ \times \mathbb{R}.$$

Théorème (CS de stabilité). *Pour que la méthode soit stable, il suffit que la fonction Φ soit Lipschitzienne en y, de constante Λ :*

$$|\Phi(t, y_1, h) - \Phi(t, y_2, h)| \leq \Lambda |y_1 - y_2|,$$

pour tout $t \in [0, T]$, tout $(y_1, y_2) \in \mathbb{R}^2$, tout $h \in \mathbb{R}$. Alors on peut prendre pour constante de stabilité

$$S = e^{\lambda T}$$

On termine ce paragraphe avec une definition.

Definition. *On dit qu'un méthode à 1 pas est d'ordre $\geq p$ si pour toute solution exacte y d'une équation différentielle*

$$y' = f(t, y) \text{ où } f \text{ est de classe } \mathcal{C}^p,$$

il existe une constante $C \geq 0$ telle que l'erreur de consistance relative à y vérifie

$$|e_n| \leq C h_n^{p+1}, \quad 0 \leq n < N.$$

6.4 Méthodes de Runge-Kutta d'ordre 2.

Si on intègre l'équation $y'(t) = f(t, y(t))$ entre t_n et t_{n+1} on obtient :

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t)) dt.$$

En utilisant la formule des trapèzes, on trouve le schéma implicite suivant, appelé *schéma de Crank-Nicolson ou du trapèze* :

$$y_{n+1} - y_n = \frac{h}{2}[f(t_n, y_n) - f(t_{n+1}, y_{n+1})], \quad \forall n \geq 0.$$

Ce schéma est implicite. En le modifiant afin de le rendre explicite, on identifie la *méthode de Heun* :

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

Ce deux méthode son d'ordre 2 par rapport à h.
Si on utilise la méthode du point milieu on trouve

$$y_{n+1} - y_n = h f(t_{n+\frac{1}{2}}, y_{n+\frac{1}{2}}).$$

Si maintenant on approche $y_{n+\frac{1}{2}}$ par

$$y_{n+\frac{1}{2}} = y_n + \frac{1}{2} f(t_n, y_n),$$

on trouve la *méthode d'Euler modifiée* :

$$y_{n+1} - y_n = hf\left(t_{n+\frac{1}{2}}, y_n + \frac{1}{2} f(t_n, y_n)\right).$$

Les méthodes de Heun et d'Euler modifiée sont des cas particuliers dans la famille des méthodes de Runge-Kutta d'ordre 2. Il existe d'autres méthodes plus compliquées, comme par exemple la *méthode de Runge-Kutta d'ordre* 4 suivante, qui est obtenue en considérant la méthode d'intégration de Simpson.

$$y_{n+1} = y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4),$$

où les K_i son calculés comme suit :

$$\begin{aligned} K_1 &= f(t_n, y_n) \\ K_2 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2} K_1) \\ K_3 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2} K_2) \\ K_4 &= f(t_{n+1}, y_n + hK_3) \end{aligned}$$

Références

[1] J.P. Demailly, Analyse numérique et équations différentielles, *Presses universitaires de Grenoble.*

[2] Quarteroni, Sacco, Saleri, Numerical Mathematics, *Sprigler.*

[3] M. Schatzman, Analyse numérique, *Masson.*