Institut Universitaire Professionnalisé d'Évry

Génie Électrique et Informatique Industrielle 3ème Année

MT31- Méthodes numériques

Saïd Mammar

1999

© **Institut Universitaire Professionnalisé d'Évry**

Table des matières

Table des figures

Avant propos

L'objectif de ce polycopié est de fournir à l'étudiant les bases des méthodes de résolution numériques appelées aussi analyse numérique.

Par souci de synthèse, nous avons volontairement omis la plupart des démonstrations. Des théorèmes donnant les conditions de convergence ne sont pas toujours spécifiés. Le lecteur devra donc se référer à la littérature pour vérifier la validité de ses choix.

Nous donnons autant que possible les algorithmes de calcul des différentes méthodes décrites, dans leurs grandes lignes sinon dans le détail.

Ce polycopié est organisé comme suit :

- Première Partie
  - Le chapitre 1 donne quelques rappels d'analyse et d'algèbre matricielle. Il traite la représentation générale des nombres puis le cas de la représentation sur calculateur et les problèmes de précision qui peuvent surgir.
  - Les chapitres 2 et 3 traitent respectivement la résolution des équations linéaires et non-linéaires.
  - Le chapitre 4 développe les méthodes d'interpolation polynômiale et spline.
  - Les chapitres 5 et 6 sont consacrés aux problèmes d'optimisation linéaire et non-linéaire
  - Le méthodes des moindres carrés linéaires et non-linéaires sont respectivement développées dans les chapitres 7 et 8.
  - Le chapitre 9 donne les différentes méthodes d'intégration numériques alors que les chapitres 10 et 11 sont respectivement consacrés aux équations différentielles ordinaires et aux dérivées partielles .
- Les textes de travaux dirigés et pratiques se trouvent respectivement en deuxième et troisième partie de ce polycopié.
- La quatrième partie donne le mode d'utilisation de quelques fonctions de Matlab permettant le calcul des quantités définies dans la première partie ainsi que la réalisation pratique de certaines méthodes et algorithmes.

Première partie

Éléments de cours

Chapitre 1 Rappels mathématiques

1 Rappels d'analyse

1.1 Calcul d'une fonction en un point x voisin de c

Le problème de base dans le calcul par ordinateur (calculateur au sens général du terme) est de pouvoir trouver une approximation de la valeur d'une fonction f en un point x donné. Pour cela, on suppose que l'on dispose de la valeur de la fonction et de ses dérivées successives en un point c. Par exemple, on cherche à calculer $\cos(0.1)$ connaissant $\cos(0) = 1$.

Développement en séries de Taylor Dans tout ce qui suit, on se limitera au cas des fonctions d'une variable réelle et à valeurs réelles.

Théorème 1 (Séries de Taylor) *Soit f une fonction C^∞, c une constante et x un point appartenant au voisinage de c, alors*

$$\begin{aligned} f(x) &\approx f(c) + f'(c)(x-c) + \frac{f''(c)}{2!}(x-c)^2 + \frac{f^{(3)}(c)}{3!}(x-c)^3 + ... \\ &= \sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!}(x-c)^k \end{aligned} \tag{1}$$

Ce théorème signifie que la valeur de f en x se déduit de la valeur de f et de toutes des dérivées au point c. Appliquer la formule de Taylor dans le cas d'un calculateur pose cependant un certain nombre de difficultés :

- Comment choisir c ?
- Le calcul des dérivées $f^{(k)}(c)$ est-il facile ?
- A quel ordre peut-on arrêter la sommation ? Que vaut alors l'erreur effectuée sur la valeur calculée de $f(x)$?

Exemple 1 $f(x) = \cos x$ $\quad c = 0$
$f^{(k)}(x) = \cos(x + \frac{k\pi}{2})$ $\quad k = 0, 1, .., n$
$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - ...$
alors *pour* $k = 1$ $\quad \cos(0.1) \approx 0.995$
pour $k = 2$ $\quad \cos(0.1) \approx 0.99500416$

Théorème 2 (Théorème de Taylor) *Soit f une fonction C^{n+1} sur un intervalle $[a, b]$, soient x et c appartenant à $[a, b]$, alors il existe $\xi(x)$ appartenant à un ouvert entre x et c tel que*

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(c)}{k!}(x-c)^k + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-c)^{n+1} \tag{2}$$

Dans cette formule, le premier terme est une approximation de $f(x)$, le deuxième évalue l'erreur entre la valeur vraie et cette approximation. L'intérêt de ce théorème par rapport au précédent est qu'il permet d'obtenir un majorant de l'erreur connaissant un majorant de $f^{(n+1)}$.

Exemple 2 $f(x) = e^x, |x| < \infty \qquad c = 0$
$f^{(k)}(x) = e^x \qquad \forall k$
$e^x = \sum_{k=0}^{n} \frac{x^k}{k!} + \frac{e^{\xi(x)}}{(n+1)!}x^{n+1}$, *le terme d'erreur tend vers 0 quand* $n \to \infty$, *donc :*

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...$$

Exemple 3 $f(x) = \cos x, |x| < \infty \qquad c = 0$
$f^{(k)}(x) = \cos(x + \frac{k\pi}{2}) \qquad \forall k$
$\cos x = \sum_{k=0}^{n} \frac{\cos(\frac{k\pi}{2})}{k!}x^k + \frac{\cos\left(\xi(x)+\frac{(n+1)\pi}{2}\right)}{(n+1)!}x^{n+1}$, *le terme d'erreur tend vers 0 quand* $n \to \infty$, *car le cosinus est borné. De plus les termes impairs sont nuls donc :*

$$\begin{aligned} \cos x &= \sum_{l=0}^{\infty} \frac{\cos\left(\frac{\pi(2l)}{2}\right)}{l!}x^{2l} = \sum_{k=0}^{\infty}(-1)^k \frac{x^{2k}}{(2k)!} \\ &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - ... \end{aligned}$$

Certaines fonctions peuvent poser des problèmes ; celles pour lesquelles, on ne peut garantir la convergence du terme d'erreur vers 0 que sur un intervalle restreint. C'est le cas de la fonction $\ln x$ traitée dans l'exemple ci-dessous.

Exemple 4 $f(x) = \ln x, 0 < x \leq 2 \qquad c = 1$
$f^{(k)}(x) = (-1)^{k-1}\frac{(k-1)!}{x^k} \qquad \forall k \geq 1$
$\ln x = \sum_{k=1}^{n}(-1)^{k-1}\frac{(x-1)^k}{k} + (-1)^n \frac{1}{n+1}\frac{(x-1)^{n+1}}{(\xi(x))^{n+1}}$,
La convergence vers 0 de $\left(\frac{x-1}{\xi(x)}\right)^{n+1}$ *quand* $n \to \infty$, *ne peut être garantie que pour* x *dans l'intervalle* $\left[\frac{1}{2}, 2\right]$.
Pour cela considérons que x *appartienne à l'intervalle* $[x_m, x_M]$. *Deux cas se présentent :*

- $x \geq 1$, *alors*

$$0 \leq x - 1 \leq x_M - 1$$

De plus $\xi(x)$ *appartient à* $]1, x[$. *On en déduit que*

$$\frac{x-1}{\xi(x)} \leq x_M - 1$$

La condition de convergence s'exprime donc sous la forme

$$x_M - 1 \leq 1$$

qui permet d'obtenir la valeur de x_M :

$$x_M = 2$$

- $x < 1$, *alors*

$$x_m - 1 \leq x - 1 < 0$$

De plus $\xi(x)$ appartient à $]x, 1[$. (avec $x > 0$). On en déduit

$$\frac{x-1}{\xi(x)} > \frac{x_m - 1}{x_m}$$

La condition de convergence s'exprime donc sous la forme

$$\frac{x_m - 1}{x_m} \geq -1$$

qui permet d'obtenir la valeur de x_m :

$$x_m = \frac{1}{2}$$

C'est dans cet intervalle uniquement, on pourra écrire :

$$\begin{aligned} \ln x &= \sum_{k=1}^{\infty} (-1)^{k-1} \frac{(x-1)^k}{k} \\ &= (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - ... \end{aligned}$$

Ce dernier exemple situe l'importance du choix de c (par rapport au point étudié x) dans la méthode. Nous allons approfondir ce point.

Théorème 3 *Soit a_n, le terme générique d'une suite vérifiant $a_n > 0$, $a_n \geq a_{n+1}, \forall n \geq 0$ et $a_n \to 0$ alors la somme*

$$\sum_{n=0}^{\infty} (-1)^n a_n \tag{3}$$

est convergente.

On peut appliquer le théorème précédent à la fonction $\ln x$ en posant

$$a_n = \frac{(x-1)^n}{n}$$

alors

$$\left| \frac{a_{n+1}}{a_n} \right| = \left| (x-1) \frac{n}{n+1} \right| < 1 \text{ pour } 0 < x < 2$$

- pour $x = 2$, $\ln 1 = 1 - \frac{1}{2} + \frac{1}{3} - ...$ est convergente
- pour $x = 0$, $\ln 1 = 1 - \frac{1}{2} - \frac{1}{3} - ...$ est divergente

On peut donc assurer la convergence sur l'intervalle $]0, 2]$

1.1.1 Influence du choix de x et de c

Supposons que l'on veuille calculer $\ln 2$. Nous disposons pour cela de plusieurs méthodes

solution 1 prendre la fonction $\ln x$, poser $c = 1$ et $x = 1$

$$\ln 2 = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \frac{1}{7} - ...$$

solution 2 prendre la fonction $\ln(1 + x)$, poser $c = 0$ et $x = 1$. Ceci donne le même résultat que précédemment.

solution 3 prendre la fonction $\ln\left(\frac{1+x}{1-x}\right)$, poser $c = 0$ et $x = \frac{1}{3}$. Le développement en série de Taylor donne

$$\ln 2 = 2\left(3^{-1} + \frac{3^{-3}}{3} + \frac{3^{-5}}{5} + \frac{3^{-7}}{7} + ...\right)$$

Les résultats sont les suivants

solutions 1 et 2 Les huit premiers termes donnent 0.63452

solution 3 Les quatre premiers termes donnent 0.69313

Le résultat de la solution 3 est plus proche du meilleur approximant (0.69315) avec moins de termes retenus. On le voit, le choix de x et de c est donc primordial.

1.1.2 Approximations polynomiales et deuxième forme du théorème de Talyor

Théorème 4 (Théorème de Taylor, deuxième forme) *Soit f une fonction C^{n+1}sur un intervalle $[a, b]$, soit x et h tels que $x \in [a, b]$ et $(x + h) \in [a, b]$. Alors, il existe $\xi(h)$ appartenant à un ouvert entre x et $(x + h)$ tel que :*

$$f(x+h) = \sum_{k=0}^{n} \frac{f^{(k)}(x)}{k!} h^k + \frac{f^{(n+1)}(\xi(h))}{(n+1)!} h^{n+1} \tag{4}$$

Cette deuxième forme du théorème de Taylor permet d'introduire la notion d'infiniment petit et de vitesse de convergence. On notera l'erreur E. Elle satisfait :

$$E_{n+1} = O(h^{n+1}) \Leftrightarrow |E_{n+1}| \leq C\,|h|^{n+1} \tag{5}$$

avec

$$C := \left| \max_{\xi(h)} \frac{f^{(n+1)}(\xi(h))}{(n+1)!} \right| \tag{6}$$

On remarquera également que le théorème de Taylor est le cas général du théorème de la valeur moyenne qui s'énonce comme suit :

Théorème 5 (Théorème de la valeur moyenne) *Soit f une fonction C^1(dérivable à dérivée continue) sur l'intervalle $[a, b]$ alors, il existe ξ appartenant à l'intervalle ouvert entre $]a, b[$, tel que*

$$f(b) = f(a) + (b - a)f'(\xi) \tag{7}$$

ou

$$f'(\xi) = \frac{f(b) - f(a)}{b - a} \tag{8}$$

1.2 Différentiation numérique

La formule usuelle d'approximation de la dérivée d'une fonction f en un point x est la formule des différences finies. Elle s'écrit :

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{9}$$

avec h suffisamment petit.

Cette formule est très pratique numériquement puisqu'elle ramène le problème de la dérivation à des opérations élémentaires. Cependant, quelle est alors l'erreur commise lors de cette approximation ? La deuxième forme du théorème de Taylor nous permet d'affirmer qu'il existe ξ dans $]x, x+h[$ tel que

$$f(x+h) = f(x) + f'(x)h + \frac{f''(\xi)}{2} h^2 \tag{10}$$

on en déduit

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(\xi)}{2}h \tag{11}$$

L'erreur commise est donc un infiniment petit d'ordre 1 ($O(h)$)

De la même manière, une simple réécriture du théorème de Taylor en $(x+h)$ et en $(x-h)$ permet d'obtenir une approximation à l'ordre 2. En effet

$$f(x \pm h) = f(x) \pm f'(x)h + \frac{f''(x)}{2!}h^2 \pm \frac{f'''(x)}{3!}h^3 + \frac{f^{(4)}(x)}{4!}h^4 \pm \frac{f^{(5)}(x)}{5!}h^5 + ... \tag{12}$$

d'où

$$f(x+h) - f(x-h) = 2f'(x)h + 2\frac{f'''(x)}{3!}h^3 + 2\frac{f^{(5)}(x)}{5!}h^5 + ...$$

et finalement

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{f'''(x)}{6}h^2 - ... \tag{13}$$

On réalité, on peut encore faire mieux et obtenir un ordre 4. C'est le cas avec la méthode de Richardson qui reprend la formule précédente avec plus de termes sous la forme

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + a_2h^2 + a_4h^4 + a_6h^6 + ... \tag{14}$$

en notant

$$\phi(h) = \frac{f(x+h) - f(x-h)}{2h} \tag{15}$$

on a alors

$$\phi(h) = f'(x) - a_2h^2 - a_4h^4 - a_6h^6 - ... \tag{16}$$

et

$$\phi\left(\frac{h}{2}\right) = f'(x) - a_2\left(\frac{h}{2}\right)^2 - a_4\left(\frac{h}{2}\right)^4 - a_6\left(\frac{h}{2}\right)^6 - ... \tag{17}$$

ce qui permet d'écrire

$$\phi(h) - 4\phi\left(\frac{h}{2}\right) = -3f'(x) - \frac{3}{4}a_4h^4 - \frac{15}{16}a_6h^6 - ... \tag{18}$$

On obtient donc la formule d'approximation de la dérivée avec un infiniment petit d'ordre 4

$$f'(x) = \phi\left(\frac{h}{2}\right) + \frac{1}{3}\left[\phi\left(\frac{h}{2}\right) - \phi(h)\right] + O(h^4) \tag{19}$$

Remarque 1 *On peut procéder de la même manière pour obtenir une approximation de la dérivée seconde de la fonction, il suffit de reprendre 12 et d'écrire*

$$f(x+h) + f(x-h) = 2f(x) + f''(x)h^2 + O(h^4)$$

et donc

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2) \tag{20}$$

2 Rappels d'algèbre

2.1 Quelques définitions

Soit $A = [a_{ij}]_{(i=1,..,n,j=1,...,m)}$ une matrice $n \times m$. On appelle transposée de la matrice A la matrice de dimension $(m \times n)$ notée A^T telle que

$$\left[A^T\right]_{i,j} = a_{ji} \tag{21}$$

Dans le cas où la matrice A est à éléments dans le corps des nombres complexes, on parle de matrice transposée conjuguée qui est notée A^H.

Une matrice carrée $A = [a_{i,j}]_{(i,j=1,...,n)}$ de dimension $n \times n$ est dite

- symétrique si $A^T = A$
- hermitienne si $A^H = A$, ^{H} signifie transposée conjuguée
- unitaire si $A^H = A^{-1}$
- orthogonale si $A^T A = AA^T = I$
- normale si $AA^H = A^H A$
- semblable à la matrice B s'il existe une matrice inversible M telle que $B = M^{-1}AM$
- définie positive (resp. semi définie positive) $(A > 0)$ (resp. $A \geq 0$) si et seulement si

$$\forall u \in \mathbb{C}^n, u^H Au > 0 \qquad (\text{resp.} \quad \geq 0) \tag{22}$$

Un nombre complexe λ est valeur propre de A si la condition suivante est vérifiée

$$\exists u \in \mathbb{C}^n, u \neq 0 \quad / \quad Au = \lambda u \tag{23}$$

On appelle déterminant de A, le scalaire noté $\det(A)$ ou encore $|A|$ donné par l'une ou l'autre des expressionns ci-dessous

$$\begin{array}{ll} \det(A) = \sum_{k=1}^{n} (-1)^{i+k} a_{ik}\Delta_{ik} & i = 1, ..., n \\ \det(A) = \sum_{i=1}^{n} (-1)^{i+k} a_{ik}\Delta_{ik} & k = 1, ..., n \end{array} \tag{24}$$

où Δ_{ik} est le déterminant de la matrice de dimension $((n-1) \times (n-1))$ obtenue en suprimant la $i^{ème}$ ligne et la $k^{ème}$ colonne de A.

Les formules précedentes indiquent que le déterminant de la matrice est indépendant de la ligne ou de la colonne choisie pour le développement. La quantité $(-1)^{i+k}\Delta_{ik}$ est appelée cofacteur de a_{ik}.

$$\det(AB) = \det(A).\det(B) \tag{25}$$

Théorème 6 *Les propriétés suivantes sont équivalentes :*

- A est inversible
- A est non singulière
- les lignes de A sont linéairement indépendantes
- Les colonnes de A sont linéairement indépendantes
- le rang de A vaut n
- A ne possède aucune valeur propre nulle
- $\det(A) \neq 0$

On appelle nombre condition associé de A aux valeurs propres la quantité

$$\kappa_\lambda(A) = \frac{\max_i |\lambda_i(A)|}{\min_j |\lambda_j(A)|} \tag{26}$$

On appelle valeurs singulières d'une matrice A les racines carrées des valeurs propres non nulles de la matrices $A^H A$.

$$\sigma_i(A) = \sqrt{\lambda_i \left(A^H A\right)} \tag{27}$$

On démontre que toute matrice peut se décomposer sous la forme $A = U\Sigma V^T$, où U et V sont des matrices unitaires et Σ une matrice diagonale contenant les σ_i par ordre décroissant.

On appelle nombre condition de A la quantité

$$\kappa(A) = \frac{\bar{\sigma}(A)}{\underline{\sigma}\,(A)} \tag{28}$$

avec $\bar{\sigma}(A) = \max_i (\sigma_i) = \sigma_1$ et $\underline{\sigma}(A) = \min(\sigma_i) = \sigma_r$, où r est le rang de la matrice A.

On remarque que $\kappa(A) \geq 1$. Si $\kappa(A)$ est grand, on dit que la matrice A est mal conditionnée. Les matrices mal conditionnées fournissent généralement des résultats de calcul peu stables.

2.2 Les normes

2.2.1 Norme de vecteurs

Soit $x = [x_1, ..., x_n]^T$ un vecteur. On définit :

- la norme l_p du vecteur x par

$$\|x\|_p = \left[\sum_{i=1}^{n} (x_i)^p\right]^{1/p} \tag{29}$$

- la norme 2 ou la norme euclidienne (c'est un cas particulier de la norme l_p)

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} (x_i)^2} \tag{30}$$

- la norme infinie

$$\|x\|_\infty = \max_i |x_i| \tag{31}$$

2.2.2 Normes de matrices

Soit A une matrice quelconque. On définit :

- la norme 1 de la matrice comme étant la plus grande somme de la valeur absolue des éléments pris colonne par colonne

$$\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}| \tag{32}$$

- la norme 2 ou la norme euclidienne

$$\|A\|_2 = \max_i \sigma_i(A) \tag{33}$$

– la norme infinie comme étant la plus grande somme de la valeur absolue des éléments pris ligne par ligne

$$\|A\|_\infty = \max_i \sum_{j=1}^{m} |a_{ij}| \tag{34}$$

– la norme de Frobenius

$$\|A\|_F = \sqrt{\sum_{i=1}^{n} (A^T A)_{ii}}. \tag{35}$$

2.3 Inversion de matrice

On démontre les propriétés suivantes sous condition d'existence des différentes quantités :

– $\left(A^{-1}\right)^T = \left(A^T\right)^{-1}$

– $(ABCD...)^{-1} = ...D^{-1}(BC)^{-1}A^{-1}$

– Lemme d'inversion : si $A = B + XRY$ alors

$$A^{-1} = B^{-1} - B^{-1}X\left(R^{-1} + YB^{-1}X\right)^{-1}YB^{-1} \tag{36}$$

–

$$\det(A^{-1}) = \frac{1}{\det(A)} \tag{37}$$

2.4 Opérations sur les lignes et colonnes

2.4.1 Permutations de lignes et de colonnes

Soit $A = \{a_{ij}\}_{1\leq i\leq n, 1\leq j\leq m}$ une matrice de dimension $n \times m$. On peut écrire A sous forme de matrice par blocs de vecteurs lignes :

$$A = \begin{bmatrix} l_1 \\ l_2 \\ : \\ l_n \end{bmatrix} \tag{38}$$

où

$$l_i = \begin{bmatrix} a_{i1} & a_{i2} & ... & a_{im} \end{bmatrix} \qquad i = 1, ..., n \tag{39}$$

est la $i^{ème}$ ligne de A.

La permutation de la $i^{ème}$ ligne avec la $j^{ème}$ ligne équivaut à la multiplication à gauche de A par la matrice de permutation

$$P_{ij} = \{p_{kl}\}_{1\leq k,l\leq n} \tag{40}$$

définie de la manière suivante

$$\begin{array}{llll} p_{ii} = 0 & p_{jj} = 0 & p_{ll} = 1 & (l \neq j, l \neq i) \\ p_{ij} = 1 & p_{ji} = 1 & p_{kl} = 0 & \text{sinon} \end{array}$$

Exemple 5

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} = \begin{bmatrix} 9 & 10 & 11 & 12 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Remarque 2 *La multiplication à droite de A par la même matrice P_{ij} entraîne la permutation des $i^{ème}$ et $j^{ème}$ colonnes.*

Remarque 3 *La matrice de permutation P_{ij}, vérifie l'égalité*

$$P_{ij}P_{ij} = I \tag{41}$$

Ceci se démontre simplement en remarquant que la multiplication à gauche de P_{ij} par P_{ij} entraîne la permutation des $i^{ème}$ et $j^{ème}$ ligne de P_{ij} qui devient donc la matrice identité.

Remarque 4 *On démontre sans difficulté les propriétés suivantes*

- $P_{ij} = P_{ij}^{-1}$
- $P_{ij}^T = P_{ji}$
- $\det(P_{ij}) = \pm 1$

2.4.2 Opérations linéaires sur les lignes et colonnes

La multiplication d'une ligne l_i de la matrice A par un nombre α, équivaut à la multiplication à gauche par la matrice diagonale $M_i(\alpha)$ de dimension compatible définie par :

$$\begin{array}{l} m_{jj} = 1 \quad (i \neq j) \\ m_{ii} = \alpha \end{array} \tag{42}$$

Il va de soit que la multiplication à droite par la même matrice $M_i(\alpha)$ revient à multiplier la colonne i par ce même nombre α.

Les combinaisons linéaires de lignes ou de colonnes suivent le même principe. A titre d'exemple remplacer la ligne l_i par la ligne $(\alpha l_i + \beta l_j)$ équivaut à la multiplication à gauche par la matrice $Q_{ij}(\alpha, \beta)$ définie par :

$$\begin{array}{ll} q_{kk} = 1 & (i \neq k) \\ q_{ii} = \alpha & \\ q_{ij} = \beta & \\ q_{kl} = 0 & \text{sinon} \end{array} \tag{43}$$

3 Représentation des nombres

3.1 Introduction

Dans l'informatique en totalité et l'électronique en grande partie, l'information est représentée sous forme de mots constitués de symboles qui ne peuvent prendre que deux valeurs : 0 ou 1. C'est pour cette raison que l'on utilise, improprement, le terme de traitement numérique pour désigner l'ensemble des traitements informatiques.

L'étude des procédés de traitement de la représentation binaire s'appelle la logique. On parle alors de systèmes logiques pour qualifier les éléments qui affectent les traitements portant sur les informations logiques. Les plus simples sont les portes logiques.

La synthèse d'un circuit logique revient à assembler des briques élémentaires plus ou moins complexes. On classe par degré de complexité croissante ces éléments logiques :

- les portes logiques
- les circuits moyens : compteurs, registres, décodeurs,....
- les circuits complexes : les microprocesseurs et les mémoires.

3.2 Ecriture des nombres et Changement de base

Tout nombre N peut s'écrire dans une base B quelconque :

$$N = a_n B^n + a_{n-1} B^{n-1} + ... + a_0 B^0 + a_{-1} B^{-1} + ... \tag{44}$$

où les nombres a_i sont compris entre 0 et $(B-1)$.

Ceci s'écrit également, sous la forme condensée, comme suit : $(N)_B = a_n a_{n-1}.....a_0, a_{-1}.....$

Le passage d'une base B à la base décimale s'effectue simplement par le calcul de l'expression. Le passage de la base décimale à une base B quelconque se fait par division successive par B du nombre N écrit dans la base décimale. Les nombres a_i sont alors les restes de ces divisions successives.

$$N = B.Q + a_0 \tag{45}$$

Généralement pour un nombre comportant une partie fractionnaire, on commence d'abord par traiter la partie entière puis la partie fractionnaire. Le traitement de la partie entière se fait comme précédemment. Le traitement de la partie fractionnaire se fait comme suit :

$$F = a_{-1} B^{-1} + a_{-2} B^{-2} + ... + a_{-p} B^{-p} \tag{46}$$

$$B.F = a_{-1} + F_1 \tag{47}$$

a_{-1} est donc la partie entière de $B.F$. On répète cette opération pour déterminer les autres coefficients.

Remarque 5 *Dans le cas des nombres rationnels, cette suite est soit finie soit périodique. Dans le cas périodique, il suffit de s'arrêter lorsque la période est atteinte. Pour les nombres irrationnels* $(\pi, \sqrt{2}, ...)$*, cette suite est non périodique. Il faudrait donc une infinité de chiffres pour les écrire.*

3.3 Les systèmes usuels

3.3.1 Le système binaire (B = 2)

Le système binaire est le système de la base 2. Il ne comprend que les chiffres 0 et 1. Le passage de la base décimale à la base 2 s'effectue par division successives par 2.

Exemple 6 $(67)_2 = 1000011$

Les poids sont les puissances de 2.

Soit a et b deux chiffres binaires. Ils peuvent donc prendre la valeur 0 ou 1. Nous avons 4 cas possibles pour chaque opération.

a	b	produit	somme	retenue
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	1	0	1

Le produit vaut 1 si les deux chiffres valent 1 simultanément. Elle est notée $P = a.b$

La somme vaut 1 si un seulement des deux chiffres vaut 1. On l'appelle dilemme ou encore "OU exclusif". Elle est notée : $S = a \oplus b$. La retenue est identique au produit : $R = a.b$

3.3.2 Le système octal (B = 8)

Le système octal est le système de la base 8. Il utilise donc les chiffres compris entre 0 et 7. Le passage du système binaire au système octal est immédiat. Il suffit de diviser le nombre binaire en tranches de trois chiffres à partir de la droite ($2^3 = 8$).

Exemple 7 *Le nombre écrit en binaire*

$$1|011|100|111|010$$

donne

$$1|3|4|7|2$$

en octal

La conversion inverse s'effectue selon le même procédé. C'est très rapide. L'intérêt de ce système réside justement dans cette rapidité de conversion.

3.3.3 Système hexadécimal ($B = 16$)

Le système hexadécimal est un système de numération de base 16, donc à 16 chiffres de 0 à 15 notés : 0 1 2 3 4 5 6 7 8 9 A B C D E F

Exemple 8 *$(31)_{10}$ = 16 + 15 = 1F H (H pour hexadécimal) ou $1F*

Passage de la base 2 à l'hexadécimal Pour passer de la base 2 à la base 16, il suffit de regrouper les éléments binaires du nombre à convertir 4 par 4 (2^4 =16) en partant de la droite.

Exemple 9 *$(111101010)_2$ = 1EA H*

Passage de l'hexadécimal au binaire Le passage du système hexadécimal au système binaire s'effectue de façon inverse à la précédente

Exemple 10 *4EBH = 100 1110 1011*

3.4 Notions sur les codes binaires

Les codes ne sont pas des systèmes de numération, c'est une simple convention d'écriture.

3.4.1 Code binaire naturel

C'est le code de la base 2. Ce code est pondéré c'est à dire qu'à chaque bit est associé un poids ou équivalent décimal. La soustraction binaire suit les règles suivantes :

$$\begin{aligned} 0-0 &= 0 \\ 1-0 &= 1 \\ 1-1 &= 0 \\ 10-1 &= 1 \end{aligned}$$

$$\begin{array}{rcr} 17 & & 10001 \\ - \ 10 & - & 1010 \\ \hline 7 & & 111 \end{array}$$

Lorsqu'on soustrait un nombre d'un plus petit, on effectue l'opération inverse et on affecte le résultat d'un signe -. D'autres méthodes de soustraction existent, elles utilisent le complément à 1 ou le complément à 2 du nombre pour transformer la soustraction en addition.

– Complément à 1

Le complément à 1 d'un nombre binaire s'obtient en remplaçant les 0 par des 1 et inversement. ex : 1101100 → 0010011

– Complément à 2

Le complément à 2 du nombre s'obtient en ajoutant 1 au complément à 1 de ce même nombre. ex : 1101100 → 0010011 + 1 →0010100

3.4.2 Méthode du complément à 1

On additionne le complément à 1 du nombre à soustraire puis on ajoute la retenue au résultat.

	17		10001
-	10	+	10101
	———		—————
	7		100110
			00111

S'il n'y a pas de retenue le résultat est donc négatif et il est sous sa forme de complément à 1.

3.4.3 Méthode du complément à 2

La méthode est la même que la précédente sauf que cette fois-ci on ne tiens pas compte de la retenue (compl à 2 = compl à 1 + 1). S'il n'y a pas de retenue, le résultat est encore négatif et il est sous la forme de complément à 2.

3.5 Notions de compléments

3.5.1 Complément vrai

Le complément à B d'un nombre A écrit dans la base B est donné par : $A' = B^n - A$(n désigne la puissance de B immédiatement supérieure à A). Ce complément est aussi appelé complément vrai.

3.5.2 Complément restreint

On appelle le complément à $(B-1)$ d'un nombre écrit dans la base B, le nombre A" défini par : $A" = B^n - 1 - A$. Ce complément est aussi appelé complément restreint du nombre A.

3.5.3 Représentation de l'opposé d'un nombre binaire

Le problème en informatique est de représenter les nombres signés. La solution qui est généralement adoptée est celle du complément à 2 du nombre non signé. En effet :

$$A' + A = 2^n \tag{48}$$

d'où modulo 2^n

$$A' + A = 0$$

et finalement, toujours modulo 2^n

$$-A = A^{'} \tag{49}$$

3.6 Notation scientifique des nombres

La notation scientifique d'un nombre utilise un signe, une mantisse r ($\frac{1}{10} \leq r < 1$) et un exposant n

$$\pm r \times 10^n \tag{50}$$

Exemple 11

$$32.213 = 0.32213 \times 10^2$$

Le nombre de chiffres alloués à la mantisse étant fini, il y aura forcément des erreurs d'arrondi ou de troncature. Certaines précautions sont alors nécessaires quand on effectue des opérations d'addition et de soustraction sur les nombres.

- Il faut additionner les nombres dans l'ordre croissant. A titre d'exemple supposant que l'on alloue 5 chiffres significatifs à la mantisse alors les deux opérations suivantes donnent des résultats très différents :

$$\begin{aligned} 1000000 + \underbrace{1+1+...+1}_{1 \text{ million de fois}} &= \left(0.1 \times 10^7 + 0.0000 \times 10^7\right) + 1 + 1 + ...1 \\ &= 1000000 \end{aligned}$$

  alors que

$$\underbrace{1+1+...+1}_{1 \text{ million de fois}} + 1000000 = 2000000$$

  On voit que la première manière d'additionner donne un résultat faux.

- Il faut éviter de soustraire des nombres très peu différents. Prenons le cas de la fonction $x - \sin x$ pour x proche de zéro :

$$\begin{aligned} x &= \frac{1}{15} rad \\ x &= 0.6666666667 \times 10^{-1} \\ \sin x &= 0.6661729492 \times 10^{-1} \\ x - \sin x &= 0.0004937175 \times 10^{-1} \\ &= 0.4937175000 \times 10^{-4} \end{aligned}$$

  Plusieurs artifices existent pour éviter ces soustractions :

  - pour la fonction $\sqrt{x^2+1} - 1$ au voisinage de 0, on utilisera l'égalité

$$\sqrt{x^2+1} - 1 = \frac{x^2}{\sqrt{x^2+1}+1} \tag{51}$$

  - pour la fonction $x - \sin x$, on utilisera le développement en séries de Taylor
  - idem pour la fonction $e^x - e^{-2x}$ au voisinage de 0 (ajouter les puissances identiques)
  - pour la fonction $\cos^2 x - \sin^2 x$ pour $x \approx \frac{\pi}{4}$, on utilisera l'égalité

$$\cos^2 x - \sin^2 x = \cos 2x \tag{52}$$

  - pour la fonction $\ln x - 1$ pour $x \approx e$, on utilisera la fonction $\ln \frac{x}{e}$

4 Tests d'arrêts

Les algorithmes numériques nécessitent généralement des tests d'arrêts permettant de juger de la convergence de l'algorithme utilisé. Le problème est donc de juger quant à la proximité de deux nombres a et b. Le test le plus souvent utilisé est

$$|a - b| < \varepsilon \tag{53}$$

Ce test n'est pas forcément le meilleur comme le montre la suite suivante

$$s_{n+1} = \frac{K}{n} \tag{54}$$

- pour $K = 1$ et $\varepsilon = 10^{-3}$, le test est vérifié à partir de $n = 31$
- pour $K = 0.01$ et $\varepsilon = 10^{-3}$, le test est vérifié à partir de $n = 2$

En réalité ce type de test est à proscrire, il est préférable de considérer des tests relatifs du type

- $\left|\frac{a-b}{a}\right|$, ce test ne convient pas pour une suite qui converge vers 0
- $\left|\frac{a-b}{a+b}\right|$, ce test ne convient pas pour une suite alternée
- $\frac{|a-b|}{|a|+|b|}$, ce test est très robuste.

Chapitre 2 Résolution des systèmes d'équations linéaires

1 Introduction

Le but de ce chapitre est de passer en revue quelques méthodes de résolution des systèmes d'équations linéaires. On supposera que dans ce cas on dispose d'autant d'équations que d'inconnues. On examinera par la suite le cas où cette condition n'est pas satisfaite.

Considérons le système d'équations linéaires suivant que l'on cherche à résoudre :

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

Ce système peut aussi s'écrire :

$$A.x = b \tag{2}$$

avec $A = [a_{ij}\ ]_{(i,j=1...n)}$, $x^T = [x_1, ..., x_n]$ et $b^T = [b_1, ..., b_n]$.

Théorème 7 *Le système linéaire précédent admet une solution unique si et seulement si la matrice A est inversible (on dit aussi matrice non singulière). La solution est donnée par $x = A^{-1}.b$.*

Il existe différentes méthodes de résolution de ces systèmes. Nous examinerons les avantages et les inconvénients de chacune d'elles.

2 Méthode de Gauss

La méthode de Gauss est basée sur le constat suivant : le système linéaire reste invariant pour les trois opérations suivantes effectuées dans n'importe quel ordre et un nombre de fois indéterminé :

1. Permutation de lignes de la matrice A (et donc de b)
2. Multiplication d'une ligne par une constante non nulle
3. Addition d'une ligne à une autre ligne

2.1 Exposé de la méthode

En combinant les opérations précédentes, on peut transformer la matrice A en une matrice triangulaire supérieure par le processus itératif suivant :

1. On garde la première ligne inchangée
2. On suppose $a_{11} \neq 0$. On soustrait à la $i^{ème}$ ligne de A et de b $(i = 2, ..., n)$ la première ligne multipliée par la quantité (a_{i1}/a_{11}). Les matrices A et b prennent alors la forme :

$$A^{(1)} = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ .... & ... & ... & ... \\ 0 & a_{n2}^{(1)} & a_{n3}^{(1)} & a_{nn}^{(1)} \end{bmatrix} \qquad b^{(1)} = \begin{bmatrix} b1 \\ b_2^{(1)} \\ . \\ b_n^{(1)} \end{bmatrix} \tag{3}$$

avec

$$a_{i,j}^{(1)} = a_{ij} - a_{i1}\frac{a_{1i}}{a_{11}} \qquad i \geq 2 \quad j \geq 1 \tag{4}$$

$$b_i^{(1)} = b_i - a_{i1}\frac{b_1}{a_{11}} \qquad i \geq 2 \tag{5}$$

On répète l'étape précédente en supposant cette fois-ci $a_{22}^{(1)} \neq 0$. On garde alors les deux premières lignes inchangées et on retranche $(a_{i1}^{(1)}/a_{22}^{(1)}) \times (2^{ème}$ ligne) à la $i^{ème}$ ligne $(i = 3, ..., n)$. En tout le processus est effectué $(n-1)$ fois. La matrice A devient alors triangulaire supérieure.

3 Soit U la matrice triangulaire supérieure obtenue et c la transformée de la matrice b. Nous avons :

$$\begin{aligned} u_{11}x_1 + u_{12}x_2 + ... + u_{1n}x_n &= c_1 \\ 0 + u_{22}x_2 + ... + u_{2n}x_n &= c_2 \\ &. \\ 0 + 0 + ... + u_{nn}x_n &= c_n \end{aligned}$$

4 Finalement la solution est obtenue par simple substitution :

$$x_i = \frac{\left(-\sum_{k=i+1}^{n} u_{ik}x_k + c_i\right)}{r_{ii}} \qquad (i = n, n-1, ..., 2, 1) \tag{6}$$

Remarque 6 *Il est possible de donner une interprétation matricielle de l'algorithme de Gauss comme nous allons le voir plus loin. A titre d'exemple, les équations 4 et 5 qui correspondent à la transformation de la matrice A en la matrice $A^{(1)}$ équivaut à la multiplication matricielle*

$$A^{(1)} = \left[\prod_{i=2}^{n} Q_{i1}\left(1, -\frac{a_{i1}}{a_{11}}\right)\right] A \tag{7}$$

de même

$$b^{(1)} = \left[\prod_{i=2}^{n} Q_{i1}\left(1, -\frac{a_{i1}}{a_{11}}\right)\right] b \tag{8}$$

Il est alors facile de voir que les matrices $A^{(k)}$ et $b^{(k)}$ au pas k de l'algorithme s'écrivent :

$$A^{(k)} = \left[\prod_{i=k+1}^{n} Q_{ik}\left(k, -\frac{a_{ik}}{a_{kk}}\right)\right] A^{(k-1)} \tag{9}$$

de même

$$b^{(k)} = \left[\prod_{i=k+1}^{n} Q_{ik}\left(k, -\frac{a_{ik}}{a_{kk}}\right)\right] b^{(k-1)} \tag{10}$$

pour k allant de 1 à $(n-1)$, en admettant que $A^{(0)} = A$ et $b^{(0)} = b$.

Exemple 12 *Soit à résoudre le système d'équations*

$$\begin{aligned} 6x_1 - 2x_2 + 2x_3 + 4x_4 &= 16 \\ 12x_1 - 8x_2 + 6x_3 + 10x_4 &= 26 \\ 3x_1 - 13x_2 + 9x_3 + 3x_4 &= -19 \\ -6x_1 + 4x_2 + 1x_3 - 18x_4 &= -34 \end{aligned}$$

En notant le système sous la forme $\left(\begin{array}{cccc|c} & & & & \\ & & A & & b \\ & & & & \end{array}\right)$, *les opérations successives donnent*

$$\left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 12 & -8 & 6 & 10 & 26 \\ 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & -12 & 8 & 1 & -27 \\ 0 & 2 & 3 & -14 & -18 \end{array}\right) \rightarrow$$

$$\left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & 0 & 2 & -5 & -9 \\ 0 & 0 & 4 & -13 & -21 \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & 0 & 2 & -5 & -9 \\ 0 & 0 & 0 & -3 & -3 \end{array}\right)$$

La dernière équation donne $x_4 = 1$ *; on déduit de la troisième (et en remplaçant* x_4 *par sa valeur) la valeur de* $x_3 = -2$. *En procédant de la même manière en obtient* $x_2 = 1$ *puis* $x_1 = 3$.

Théorème 8 *Soit* A *une matrice non singulière. Supposons que l'ensemble des pivots successifs sont non nuls* $(a_{11}, a_{22}^{(1)}, a_{33}^{(2)}, ...)$, *alors il existe deux matrices* L *et* U *respectivement triangulaire inférieure avec des 1 sur la diagonale et triangulaire supérieure telles que*

$$A = LU$$

La matrice triangulaire supérieure U est celle obtenue par l'application de la méthode de Gauss

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & ... & u_{1n} \\ 0 & u_{22} & u_{23} & ... & u_{2n} \\ : & 0 & u_{33} & ... & u_{3n} \\ 0 & ... & : & & : \\ 0 & 0 & 0 & ... & u_{nn} \end{bmatrix} \tag{11}$$

On peut vérifier que la matrice L est constituée des éléments $a_{ik}^{(k-1)}/a_{kk}^{(k-1)}$ permettant d'annuler les éléments de colonne k pour $i > k$. Il suffit pour cela de se rappeler que la soustraction de ligne ou de colonnes d'une matrice équivaut à la multiplication à gauche par une matrice de structure adéquate (cf chapitre 3). On a alors

$$L = \begin{bmatrix} 1 & 0 & 0 & ... & 0 \\ 0 & 1 & 0 & ... & 0 \\ l_{31} & l_{32} & 1 & ... & 0 \\ : & : & : & & : \\ l_{n1} & l_{n2} & l_{n3} & ... & 1 \end{bmatrix} \tag{12}$$

avec

$$l_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} \qquad k \geq 1 \quad i \geq k \tag{13}$$

On peut aussi obtenir L à partir de la matrice U

$$l_{ik} = \frac{\left(a_{ik} - \sum_{j=1}^{k-1} l_{ij} u_{jk}\right)}{u_{kk}} \tag{14}$$

Ce théorème signifie aussi que la résolution du système

$$A.x = b \tag{15}$$

peut se résoudre de la manière suivante en remarquant que le système s'écrit aussi

$$L.U.x = b \tag{16}$$

Définissons une variable auxiliaire y telle que

$$L.y = b \tag{17}$$

La variable y vérifie aussi

$$U.x = y \tag{18}$$

Finalement, la résolution du système de départ s'est simplifiée en la résolution de deux systèmes triangulaires (qui se résolvent par simple substitution).

2.2 Méthode de Gauss à pivot partiel

La quantité $a_{ik}^{(k)}/a_{kk}^{(k)}$ est appelée pivot. Il peut arriver que ce pivot soit nul. Alors, la méthode précédente échoue.

De plus, pour des raisons de stabilité de la méthode, on préfère à chaque étape prendre l'élément $a_{kk}^{(k)}$ de module le plus grand. Au pas k de l'élimination, on examine les $(n-k+1)$ lignes restantes et on détermine la $j^{ème}$ ligne dans l'élément $a_{jj}^{(k)}$ de module le plus grand, on permute alors la $k^{ème}$ ligne avec la $j^{ème}$.

2.3 Méthode de Gauss à pivot total

Au pas k de l'élimination, on examine l'ensemble de la matrice d'ordre $(n-k+1)$. On détermine l'élément $a_{ij}^{(k)}$ de module le plus grand, on permute alors la $k^{ème}$ ligne avec la $i^{ème}$ et la $k^{ème}$ colonne avec la $j^{ème}$. L'ordre des composantes du vecteur x est alors changé du fait de la permutation des colonnes, il ne faudra donc pas oublier de les remettre dans l'ordre initial.

Le théorème ci-dessous donne l'existence de la matrice triangulaire supérieure à laquelle on doit aboutir.

Théorème 9 *Soit A une matrice non singulière alors il existe une matrice de permutation P ($PP = I$) et deux matrices L et U respectivement triangulaire inférieure et triangulaire supérieure telles que*

$$PA = LU \tag{19}$$

Le système se résout alors en résolvant par substitution amont et aval les systèmes

$$L.y = P.b \tag{20}$$
$$U.x = y \tag{21}$$

Ce théorème sera démontré et la méthode illustrée sur un exemple dans le TD 1.

2.4 Nombre d'opérations effectuées

Nous n'allons compter que le nombre d'opérations complexes et coûteuses en temps de calcul, c'est-à-dire les multiplications et les divisions.

Considérons d'abord la décomposition de PA sous la forme LU. Au pas k de l'élimination, on effectue

- $(n-k)$ divisions par a_{ik}
- $(n-k)^2$ multiplications pour obtenir les éléments $a_{ij}^{(k)}$

Le nombre d'opérations total est donc

$$\begin{aligned} N_{LU} &= \sum_{k=1}^{n-1}(n-k) + \sum_{k=1}^{n-1}(n-k)^2 \\ &= \frac{1}{2}n(n-1) + \frac{1}{6}n(n-1)(n-2) \\ &= \frac{1}{3}(n^3 - n) \end{aligned} \tag{22}$$

Le calcul de l'élément y_i du vecteur y à partir de l'équation 20 nécessite $(i-1)$ multiplications. Le nombre d'opérations nécessaires au calcul de y est donc

$$N_y = \sum_{i=1}^{n}(i-1) = \frac{1}{2}n(n-1) \tag{23}$$

De même le calcul de x_i à partir de l'équation 18 nécessite $(n-i+1)$ multiplications. Donc le calcul de x nécessite

$$N_x = \sum_{i=1}^{n}(n-i+1) = \frac{1}{2}n(n+1) \tag{24}$$

Le nombre d'opérations total est donc

$$\begin{aligned} N_{Gauss} &= N_{LU} + N_x + N_y \\ &= \frac{1}{3}n^3 + n^2 - \frac{1}{3}n \end{aligned} \tag{25}$$

Le nombre d'opérations est donc globalement de l'ordre de $\frac{2}{3}n^3$. La méthode présentée ci-après nécessite quant à elle un nombre d'opérations de l'ordre de $\frac{1}{3}n^3$.

3 Méthode de décomposition QR (Méthode De Householder)

On démontre que toute matrice carrée A peut être décomposée de manière unique sous la forme

$$A = QR \tag{26}$$

où Q est une matrice unitaire ($Q^{-1} = Q^T$) et R une matrice triangulaire supérieure telle que ($r_{ii} \geq 0$). Cette décomposition est généralement obtenue en utilisant les transformations de Householder. L'algorithme de Gram-Schmit permet aussi d'obtenir cette décomposition :

- On écrit la matrice A $(n \times n)$ sous la forme de vecteurs colonnes $A = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}$ et on recherche Q sous la forme de vecteurs colonnes $Q = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}$
- On détermine q_1 en normalisant le vecteur a_1 :

$$\begin{cases} \tilde{q}_1 = a_1 \\ q_1 = \frac{\tilde{q}_1}{\|\tilde{q}_1\|} \end{cases} \tag{27}$$

– On itère sur le vecteurs colonnes en faisant varier k de 2 à n, et en enlevant à chaque fois les composantes selon q_j $(j = 1, ..., (k-1))$. Ceci permet d'obtenir un vecteur $\tilde{q}_k$ orthogonal aux vecteurs q_j $(j = 1, ..., (k-1))$, puis on normalise ce vecteur pour obtenir q_k.

$$\begin{cases} \tilde{q}_k = a_k - \left(a_k^T q_{k-1}\right) q_{k-1} - \left(a_k^T q_{k-2}\right) q_{k-2} - ... - \left(a_k^T q_1\right) q_1 \\ q_k = \frac{\tilde{q}_k}{\|\tilde{q}_k\|} \end{cases} \tag{28}$$

– La matrice R est alors simplement calculée par

$$R = Q^T A \tag{29}$$

La résolution du système linéaire est identique à celle de la décomposition LU. Il suffit d'écrire

$$Ax = QR.x = b \tag{30}$$

Définissons une variable auxiliaire y telle que

$$Q.y = b \tag{31}$$

et

$$y = Q^{-1}b = Q^T b \tag{32}$$

Par conséquent, la variable y vérifie aussi

$$R.x = y \tag{33}$$

Ce système est facile à résoudre puisque la matrice R est triangulaire.

Notons que la méthode de décomposition QR est une des plus utilisées car elle est très robuste aux erreurs numériques.

4 Factorisation de Cholesky

Dans le cas où la matrice A est symétrique définie positive, la décomposition LU prend une forme particulière appelée factorisation de Cholesky. La matrice A se décompose sous la forme

$$A = BB^T \tag{34}$$

où B est une matrice triangulaire inférieure telle que $b_{ii} > 0$. Cette décomposition est unique.

Remarque 7 *Il faut noter que les méthodes de décomposition LU, QR et celle de Cholesky sont généralement celles utilisées pour le calcul du déterminant de la matrice et de son inverse. Nous rappelons que le déterminant d'une matrice diagonale ou triangulaire est le produit des éléments diagonaux.*

5 Méthodes itératives

Le principe des méthodes itératives est de trouver une suite de vecteurs $x^{(k+1)}$ de la forme

$$x^{(k+1)} = Mx^{(k)} + v \tag{35}$$

qui converge vers la solution du système

$$Ax = b \tag{36}$$

La convergence de la méthode itérative est directement liée au fait que $\rho(M) < 1$ ou de manière équivalente à $\|M\| < 1$. Le problème revient donc à trouver une matrice M et un vecteur v assurant la convergence de la suite. Différentes méthodes existent.

5.1 Méthode de Jacoby

La $i^{ème}$ équation du système 36 s'écrit

$$a_{i,1}x_1 + a_{i,2}x_2 + ... + a_{i,i}x_i + ... + a_{i,n}x_n = b_i \tag{37}$$

En exprimant x_i en fonction des autres variables, on obtient

$$x_i = \frac{-a_{i,1}x_1 - a_{i,2}x_2 - ... - a_{i,i-1}x_{i-1} - a_{i,i+1}x_{i+1} - ... + a_{i,n}x_n + b_i}{a_{i,i}} \tag{38}$$

On peut bien sûr faire de même avec l'ensemble des inconnues x_i .

Les quantités M et v s'obtiennent en affectant le pas (k) au membre de droite de 38 et le pas $(k+1)$ à son membre de gauche. L'équation 38 se réécrit alors

$$x_i^{(k+1)} = \frac{-a_{i,1}x_1^{(k)} - a_{i,2}x_2^{(k)} - ... - a_{i,i-1}x_{i-1}^{(k)} - a_{i,i+1}x_{i+1}^{(k)} - ... + a_{i,n}x_n^{(k)} + b_i}{a_{i,i}} \tag{39}$$

Une écriture matricielle de l'équation précédente donne

$$M = -D^{-1}(L+U) \tag{40}$$
$$v = D^{-1}b \tag{41}$$

avec

$$D = \begin{bmatrix} a_{1,1} & 0 & ... & 0 \\ 0 & a_{2,2} & 0 & : \\ : & ... & . & 0 \\ 0 & ... & 0 & a_{n,n} \end{bmatrix}$$

$$L = \begin{bmatrix} 0 & 0 & ... & 0 \\ a_{2,1} & 0 & .. & : \\ : & .. & 0 & 0 \\ a_{n,1} & ... & a_{n,n-1} & 0 \end{bmatrix} \qquad U = \begin{bmatrix} 0 & a_{1,2} & ... & a_{1,n} \\ 0 & 0 & .. & : \\ : & .. & 0 & a_{n-1,n} \\ 0 & ... & 0 & 0 \end{bmatrix}$$

5.2 Méthode de Gauss-Seidel

Pour cette méthode, on prend

$$M = -(D+L)^{-1}U \tag{42}$$
$$v = (D+L)^{-1}b \tag{43}$$

Cette formule est issue d'une équation similaire à 39 dans laquelle tous les $x_j^{(k)}$ sont remplacés par $x_j^{(k+1)}$ pour $(j < i)$

$$x_i^{(k+1)} = \frac{-a_{i,1}x_1^{(k+1)} - a_{i,2}x_2^{(k+1)} - ... - a_{i,i-1}x_{i-1}^{(k+1)} - a_{i,i+1}x_{i+1}^{(k)} - ... + a_{i,n}x_n^{(k)} + b_i}{a_{i,i}} \tag{44}$$

Cette méthode permet de diminuer l'espace mémoire nécessaire et accélère la convergence de la suite.

Remarque 8 *Il existe d'autres méthodes qui sont spécifiques à la structure de la matrice A que nous aborderons très brièvement ici :*

- *la méthode de relaxation pour les matrices creuses*
- *La méthode de Choleskey pour les matrices symétriques*
- *les systèmes tridiagonaux*
- *le système de Toeplitz (Algorithme de Levinson)*

6 Résolution des systèmes à structure particulière

6.1 Matrices symétriques : factorisation de Cholesky

Dans le cas où la matrice A est symétrique définie positive, la décomposition LU prend une forme particulière appelée factorisation de Cholesky (voir TD 4). La matrice A se décompose sous la forme

$$A = BB^T \tag{45}$$

où B est une matrice triangulaire inférieure telle que $b_{ii} > 0$. Cette décomposition est unique.

Remarque 9 *Il faut noter que les méthodes de décomposition LU, QR et celle de Cholesky sont généralement celles utilisées pour le calcul du déterminant de la matrice et de son inverse. Nous rappelons que le déterminant d'une matrice diagonale ou triangulaire est le produit des éléments diagonaux.*

6.2 Système tri-diagonal

La résolution du système de la forme

$$\begin{bmatrix} a_{11} & a_{12} & 0 & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & \vdots & \vdots \\ 0 & \cdots & \cdots & \vdots & \vdots \\ \vdots & \cdots & \cdots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{n(n-1)} & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_n \end{bmatrix} \tag{46}$$

se résoud par l'algorithme suivant :

- On pose $z_1 = \frac{a_{21}}{a_{11}}$ et $w_1 = \frac{b_1}{a_{11}}$
- pour k allant de 2 à n, on calcule successivement

$$w_k = \frac{a_{k,k+1}}{a_{kk} - a_{k-1,k} z_{k-1}}$$

et

$$z_k = \frac{b_k - a_{k-1,k} w(k-1)}{a_{kk} - a_{k-1,k} z(k-1)}$$

- On pose $x_n = w\,(n)$
- et pour k allant de $(n-1)$ à 1, on calcule

$$x_k = w_k - z_k x_{k+1}$$

6.3 Système de Toeplitz

On appelle système de Toeplitz un système d'équations de la forme

$$Tx = b$$

où T est une matrice de Toeplitz de la forme

$$T = \begin{bmatrix} t_0 & t_1 & \cdots & t_n \\ t_1 & t_0 & \cdots & t_{n-1} \\ \vdots & \cdots & \cdots & \vdots \\ t_n & \cdots & \cdots & t_0 \end{bmatrix} \tag{47}$$

La résolution se fait par itération sur l'ordre du système. La méthode est appelée algorithme de Levinson.

7 Quantification de l'erreur et de la précision

Les solutions obtenues numériquement (à l'aide d'un calculateur) ne sont que des valeurs approchées de la solution exacte. Soit x la solution exacte du système $A.x = b$ et $\widetilde{x}$ la solution approchée calculée. Nous avons donc les relations suivantes :

$$A.x - b = 0 \tag{48}$$

$$A.\widetilde{x} - b = r \tag{49}$$

où r est appelé le résidu.

Soit z l'erreur commise sur la solution : $x = \widetilde{x} + z$. On montre que z vérifie la relation

$$A.z + r = 0 \tag{50}$$

Ainsi, on peut calculer z à partir du résidu r par la méthode de Gauss par exemple.

Examinons maintenant la sensibilité de la solution vis-à-vis de variations possibles de A et de b. Supposons donc que chacune des quantités A, x et b de l'équation 48 subissent respectivement des variations $\Delta A, \Delta x$ et Δb. L'équation 48 devient

$$(A + \Delta A)\,.\,(x + \Delta x) = (b + \Delta b) \tag{51}$$

On démontre alors que l'erreur relative sur x peut être majorée

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \kappa(A)\frac{\|\Delta A\|}{\|A\|}} \left[\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|}\right] \tag{52}$$

Cette formule montre l'importance du nombre condition de A. De plus ce résultat permet de connaître le nombre de chiffres significatifs nécessaires au calcul de la solution. Supposons qu'on utilise d chiffres significatifs, alors

$$\frac{\|\Delta A\|}{\|A\|} \approx 5 \times 10^{-d} \qquad \frac{\|\Delta b\|}{\|b\|} \approx 5 \times 10^{-d} \tag{53}$$

Supposons de plus que $\kappa(A) \approx 10^{\alpha}$, alors

$$\begin{aligned} \frac{\|\Delta x\|}{\|x\|} &\leq \frac{10^{\alpha}}{1 - 5 \times 10^{-d+\alpha}} \left[5 \times 10^{-d} + 5 \times 10^{-d}\right] \\ &\leq 10^{\alpha - d + 1} \end{aligned} \tag{54}$$

Ce qui veut dire que x ne sera précis qu'après $(d - \alpha - 1)$ chiffres décimaux.

Exemple 13 *Le système avec A et b donnés par*

$$A = \begin{bmatrix} 0.99 & 0.98 \\ 0.98 & 0.97 \end{bmatrix} \qquad b = \begin{bmatrix} 1.97 \\ 1.95 \end{bmatrix}$$

admet pour solution

$$x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Considérons maintenant (avec $d = 5$)

$$A + \Delta A = \begin{bmatrix} 0.990004 & 0.980002 \\ 0.980002 & 0.970003 \end{bmatrix} \qquad b = \begin{bmatrix} -1.969956 \\ -1.949923 \end{bmatrix}$$

Il admet pour solution

$$x = \begin{bmatrix} 0.668925 \\ 1.334401 \end{bmatrix}$$

De plus

$$\kappa(A) = 3.8 \times 10^4$$

donc $\alpha = 4$.

On calcule

$$\|A\| = 1.9601 \qquad \|\Delta A\| = 6.820052 \times 10^{-6} \qquad \|b\| = 2.7719$$
$$\|\Delta b\| = 8.820256 \times 10^{-5} \qquad \|x\| = 1.4142 \qquad \|\Delta x\| = 0.470567$$
$$\frac{\|\Delta x\|}{\|x\|} = 0.332741$$

La formule 52 donne dans ce cas une bonne estimée comme le montre le calcul

$$\frac{\|\Delta x\|}{\|x\|} \leq 1.565407$$

On vérifie donc qu'aucun des chiffres de la solution n'est précis.

8 Inversion de matrice

Trouver l'inverse d'une matrice A, est équivalent à la résolution du système d'équation

$$Ax = y \tag{55}$$

qui donne alors le vecteur x en fonction du vecteur y sous la forme

$$y = A^{-1}x \tag{56}$$

On peut alors envisager un algorithme qui itère sur l'ensemble des composantes x_i de x de la manière suivante :

1. On répète les deux opérations suivantes pour i allant de 1 à n.
2. On calcule x_i en fonction de y_i et des x_j $(j > i)$. A titre d'exemple au premier pas, nous aurons
$$x_1 = \frac{y_1 - \sum_{j=2}^{n} a_{1j}x_j}{a_{11}} \tag{57}$$
3. On remplace x_i par l'expression obtenue dans les équations des y_j $(j > i)$

On voit que le but de cet algorithme est d'échanger les rôles de x et de y. Partant des équations du type

$$y_j = \sum_{i=1}^{n} a_{ji}x_i \tag{58}$$

on aboutit à des équations du type

$$x_i = \sum_{j=1}^{n} \alpha_{ij}y_j$$

Les α_{ij} sont alors les éléments de la matrice A^{-1}.

Remarque 10 *Il peut arriver qu'un des pivots soit nul. Il suffit d'échanger alors x_i avec un autre des y_j restant. Il faudra alors se rappeler de la permutation effectuée.*

Nous donnons ci-dessous le formalisme de l'algorithme dans le cas général où on permute x_q avec y_p

$$\begin{array}{cccccc} & x_1 & \dots & x_q & \dots & x_n \\ y_1 = & a_{11} & \dots & a_{1q} & \dots & a_{1n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_i = & a_{i1} & \dots & a_{iq} & \dots & a_{in} \\ \vdots & & & & & \\ y_p = & a_{p1} & \dots & a_{pq} & \dots & a_{pn} \\ \vdots & \vdots & & \vdots & & \vdots \\ y_n = & a_{n1} & \dots & a_{nq} & \dots & a_{nn} \end{array} \tag{59}$$

Le choix du pivot a_{pq} permet d'exprimer x_q en fonction des x_i $(i \neq q)$ et de y_p

$$x_q = \frac{1}{a_{pq}} \left[y_p - \sum_{i=1, i\neq q}^{n} a_{pi} x_i \right] \tag{60}$$

En remplaçant x_q par l'équation 60 dans les expressions de y_i $(i \neq p)$ et de y_p, on obtient

$$\begin{aligned} y_i &= \sum_{k=1,k\neq q}^{n} a_{i,k} x_k + a_{iq} \left[\frac{1}{a_{pq}} \left[y_p - \sum_{i=1,i\neq q}^{n} a_{pi} x_i \right] \right] \\ &= \frac{a_{iq}}{a_{pq}} y_p + \sum_{k=1,k\neq q}^{n} \left(a_{i,k} - - \frac{a_{iq}}{a_{pq}} a_{pi} \right) x_k \qquad (i \neq p) \end{aligned} \tag{61}$$

On obtient finalement le système ci-dessous où x_q et y_p ont été permutés.

$$\begin{array}{cccccc} & x_1 & \dots & y_p & \dots & x_n \\ y_1 = & a'_{11} & \dots & a'_{1q} & \dots & a'_{1n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_i = & a'_{i1} & \dots & a'_{iq} & \dots & a'_{in} \\ \vdots & & & & & \\ x_q = & a'_{p1} & \dots & a'_{pq} & \dots & a'_{pn} \\ \vdots & \vdots & & \vdots & & \vdots \\ y_n = & a'_{n1} & \dots & a'_{nq} & \dots & a'_{nn} \end{array} \tag{62}$$

avec

$$a'_{pq} = \tfrac{1}{a_{pq}} \tag{63}$$

$$a'_{pk} = -\tfrac{a_{pk}}{a_{pq}}\ , \qquad (k \neq q) \tag{64}$$

$$a'_{iq} = \tfrac{a_{iq}}{a_{pq}} \qquad (i \neq p) \tag{65}$$

$$a'_{ik} = a_{ik} + a_{iq} a'_{pk} \qquad (i \neq p, k \neq q) \tag{66}$$

Chapitre 3 Résolution des équations non-linéaires

L'objectif de ce chapitre est de comparer différentes méthodes de résolution d'équations non linéaires du type :

$$f(x) = 0 \tag{1}$$

dans le cas où une solution analytique ne peut pas être obtenue. Après l'exposé des méthodes applicables aux fonctions à une variable, nous aborderons le cas des systèmes d'équations non linéaires puis le cas particulier des racines d'un polynôme. La particularité de ces méthodes est qu'elles ne permettent de déterminer qu'une seule racine. Il faut alors rechercher les autres racines possibles par itération.

1 Fonctions à une variable, exposé des méthodes

1.1 Méthode de Dichotomie (Bisection)

Cette méthode repose sur le constat que si $f(x)$ est continue et que le produit $f(a).f(b) < 0$ est négatif, alors la fonction f s'annule au moins une fois sur l'intervalle $[a, b]$. Les différentes étapes de la méthode peuvent être résumées comme suit :

1. Choisir un intervalle $[x_0 = a, x_1 = b]$ tel que $f(a).f(b) < 0$.
2. Calculer la valeur de la fonction en $x_2 = (a + b)/2$
3. On retient comme nouvel intervalle $[x_0, x_2]$ ou $[x_2, x_1]$ en respectant la condition du 1. On est alors assuré de toujours encadrer la racine.
4. Répéter les étapes 2 et 3 jusqu'à l'obtention de la précision désirée, c'est à dire jusqu'à ce que $|f(x_2)| < \varepsilon$, ou $|x_{k+1} - x_k| < \varepsilon$, ε étant la précision désirée.

Après k itérations, la précision sur la racine s, vaut :

$$|x_k - s| < \frac{b-a}{2^{k+1}} \tag{2}$$

Il est donc possible d'estimer à l'avance le nombre d'itérations nécessaires pour approcher s avec une précision donnée. Si on désire une tolérance τ sur la racine, alors l'inégalité précédente donne le nombre minimal d'itérations n qui doit vérifier

$$\frac{b-a}{2^{n+1}} < \tau \tag{3}$$

On en déduit le minorant de n

$$n > \frac{\log(b-a) - \log 2\tau}{\log 2} \tag{4}$$

Si on choisit le logarithme de base 2 (bits de précision) l'inégalité précédente se simplifie

$$n > \log_2(b-a) - 1 - \log_2 \tau \tag{5}$$

Il faut donc une itération supplémentaire par bit de précision. On remarquera que l'ordre de convergence vaut $p = 1$ et qu'il est indépendant de la fonction.

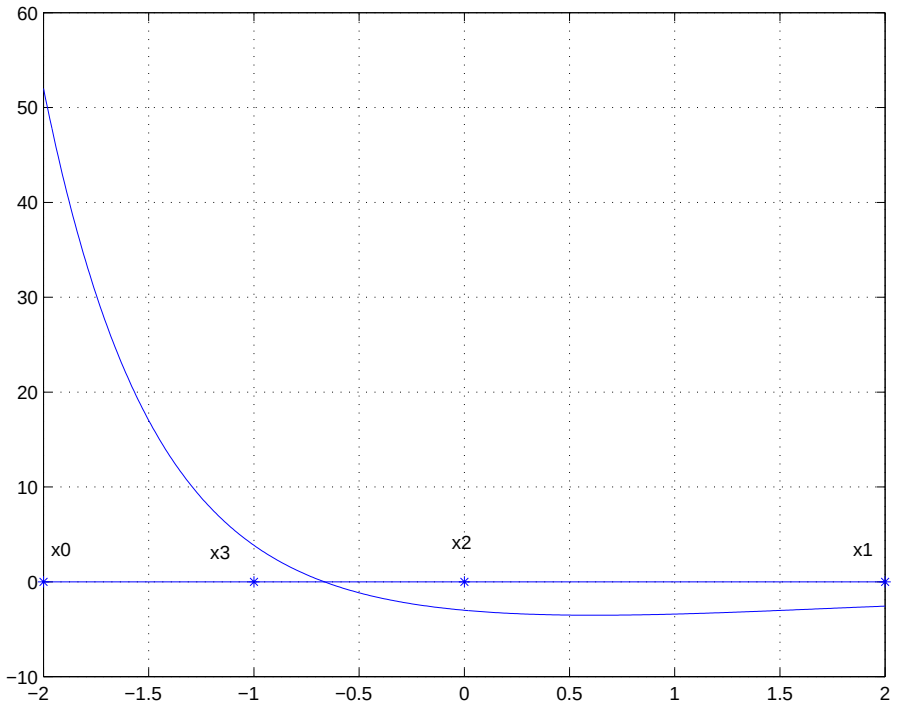

FIG. 1: Méthode de dichotomie

Exemple 14 *Les points successifs pour les premiers pas d'itération apparaissent sur la figure 1 pour la fonction*

$$f(x) = e^{-2x} - \cos x - 3$$

avec $a = -2$ *et* $b = 2$.

1.2 Méthode de Regula Falsi (fausse position)

La première étape de la méthode est identique à celle de la précédente. Au lieu d'utiliser le point médian, on utilise le point d'intersection avec l'axe des abscisses de la droite $(x^{(0)} = a, y^{(0)} = f(a)), (x^{(1)} = b, y^{(1)} = f(b))$. Ce point est donné par l'équation suivante qui se vérifie très simplement :

$$x^{(2)} = x^{(0)} - y^{(0)} \frac{x^{(1)} - x^{(0)}}{y^{(1)} - y^{(0)}} \tag{6}$$

Les étapes 3 et 4 sont alors identiques à celles de la méthode de dichotomie

La figure 2 montre l'évolution des points au cours des itérations. On démontre que si $f'(s)$ et $f''(s)$ sont différents de 0, l'ordre de convergence de cette méthode vaut aussi 1.

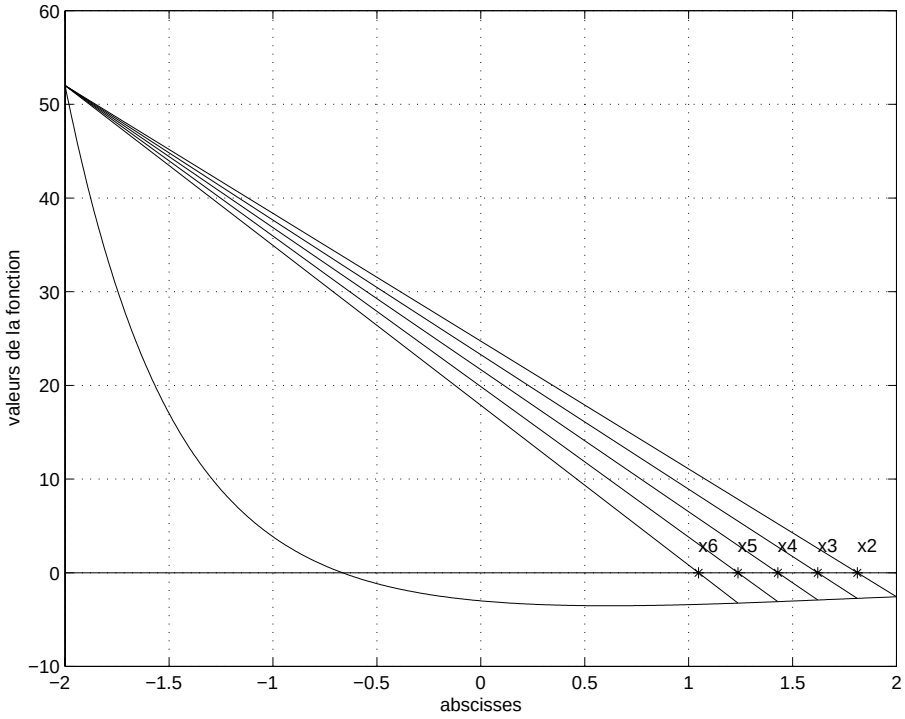

FIG. 2: Méthode de Régula-Falsi

1.3 Méthode de la sécante

Cette méthode est une variante de la méthode de Regula Falsi. La condition de l'étape 1 n'a plus à être satisfaite : la racine n'est pas nécessairement dans l'intervalle $[x^{(k)}, x^{(k+1)}]$. Comme il ne subsiste aucune indication sur le choix initial de l'intervalle $[x^{(0)}, x^{(1)}]$, on effectue un choix arbitraire de cet intervalle et on surveille alors la décroissance de la norme de $y^{(k)} = f(x^{(k)})$.

$$x^{(k+1)} = x^{(k)} - y^{(k)} \frac{x^{(k)} - x^{(k-1)}}{y^{(k)} - y^{(k-1)}} \quad (k = 1, 2, ...) \tag{7}$$

L'ordre de convergence vaut $p = \frac{1+\sqrt{5}}{2} = 1.618$.

La figure 3 montre l'évolution des points en prenant comme intervalle initial, l'intervalle $[-2, -1]$ qui ne contient pas la racine. On pourra vérifier que la méthode diverge si on choisit par exemple l'intervalle $[1, 2]$.

1.4 Méthode de Newton

Dans ce cas, on suppose que la fonction est continûement dérivable et que sa dérivée peut être calculée facilement. La méthode consiste alors à approcher la fonction au point de coordonnées $(x^{(k)}, y^{(k)})$ par sa tangente :

$$y = f(x^{(k)}) + (x - x^{(k)}) f'(x^{(k)}) \tag{8}$$

On obtient alors $x^{(k+1)}$, point d'intersection avec l'axe des abscisses :

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \quad (k = 0, 1, 2, ...) \tag{9}$$

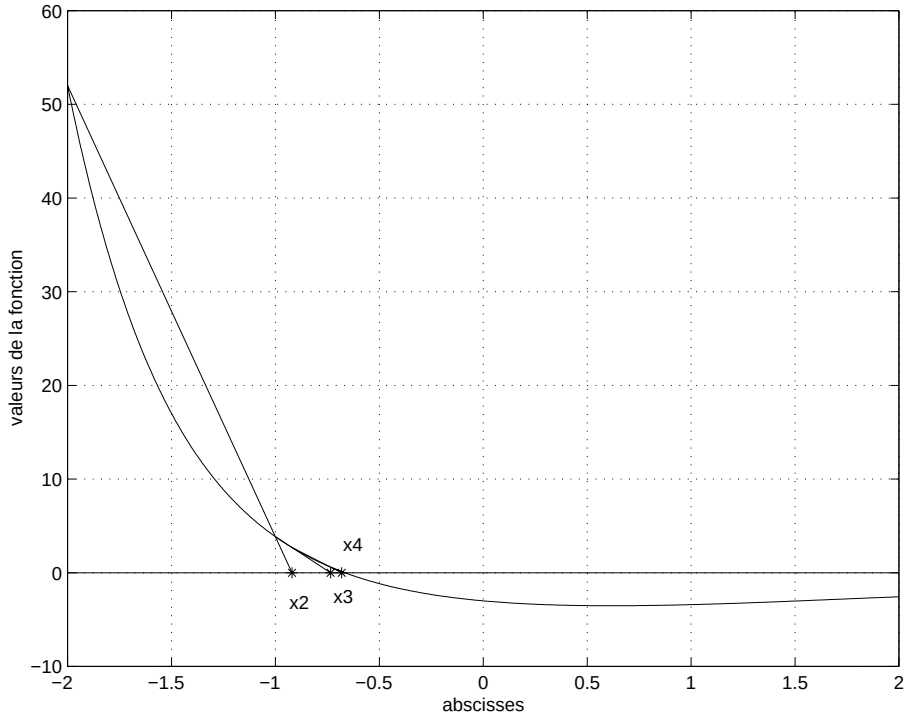

FIG. 3: Méthode de la sécante

La figure 4 montre l'évolution des points au cours des itérations si on prend comme point de départ le point $x_0 = -2$.

Remarque 11 *Bien que plus rapide, la méthode de Newton échoue dans plusieurs cas. Quelques exemples sont rassemblés sur la figure ci-dessous*

1.5 Récapitulatif

méthode	Rapidité	nombre de points	$f(a)f(b) < 0$	$f \in C^2$
dichotomie	non	2	oui	non
Newton	oui	1	non	oui
Sécante	moyen	2	non	non

2 Systèmes d'équations non linéaires

La méthode de newton peut être généralisée à la résolution des systèmes d'équations non-linéaires de la forme :

$$f_i(u) = f_i(x_1, x_2, ..., x_n) = 0 \qquad (i = 1, 2, ..., n) \tag{10}$$

A titre d'exemple, considérons le système d'équations à deux inconnues suivant

$$\begin{cases} f(x, y) = 0 \\ g(x, y) = 0 \end{cases} \tag{11}$$

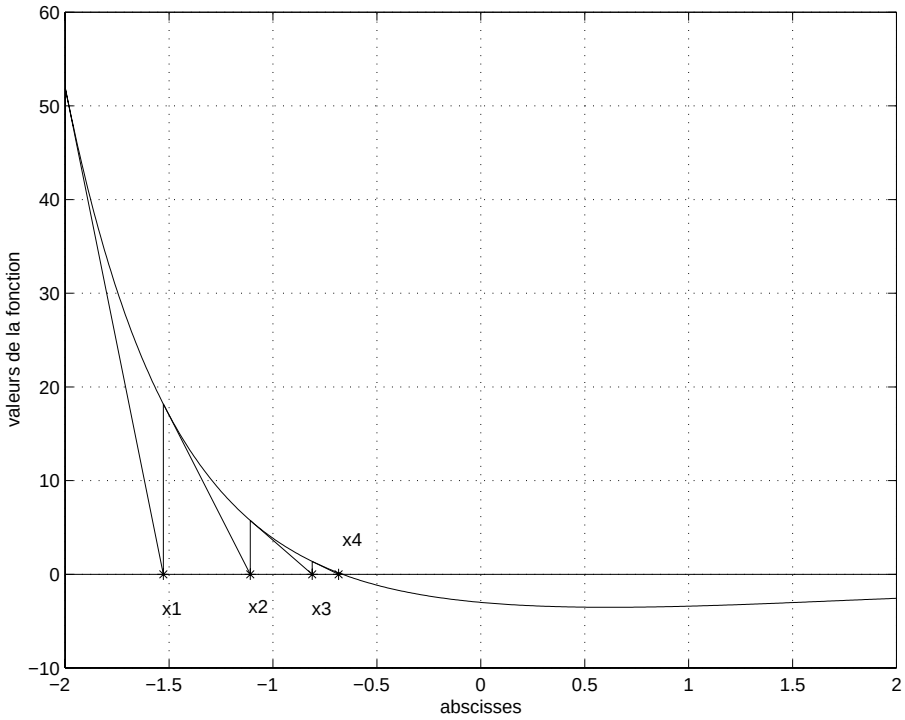

FIG. 4: Méthode de Newton

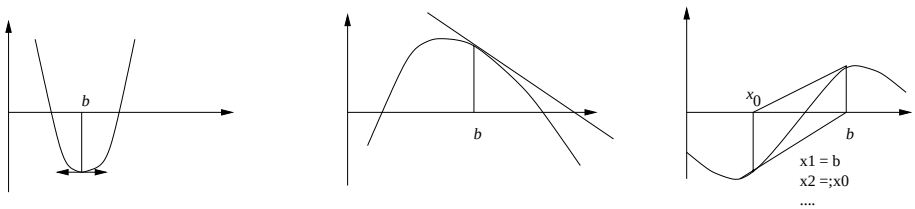

FIG. 5: Cas où la méthode de Newton échoue

On définit le Jacobien du système par la matrice aux dérivées partielles :

$$\Phi(x,y) = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x & f_y \\ g_x & g_y \end{bmatrix} \tag{12}$$

En posant $u = [x,y]^T$, la suite des points $u^{(k)}$ est obtenue par généralisation de l'équation de Newton pour une fonction à une variable. Elle prend la forme :

$$u^{(k+1)} = u^{(k)} - \Phi^{-1}(u^{(k)})F(u^{(k)}) \quad (k = 0,1,2,...) \tag{13}$$

avec $F^T = [f,g]$.
La forme explicite de l'équation 13 dans ce cas simple est

$$x^{(k+1)} = x^{(k)} + \left.\frac{g.f_y - f.g_y}{f_x g_y - f_y g_x}\right|_{(x^{(k)},y^{(k)})} \tag{14}$$

$$y^{(k+1)} = x^{(k)} + \left.\frac{f.g_x - g.f_x}{f_x g_y - f_y g_x}\right|_{(x^{(k)},y^{(k)})} \tag{15}$$

Dans le cas général de l'équation 10, l'équation 13 reste vraie en définissant :

$$u = \begin{bmatrix} x_1 \\ x_2 \\ : \\ x_n \end{bmatrix} \qquad F = \begin{bmatrix} f_1 \\ f_2 \\ : \\ f_n \end{bmatrix} \qquad \text{et} \quad \Phi = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & ... & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & : & ... & \frac{\partial f_2}{\partial x_n} \\ : & : & ... & : \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & ... & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

3 Zéros de polynômes

La méthode de Newton est aussi bien adaptée au problème de résolution d'équations de type polynomial. Nous nous restreindrons au cas de polynômes à coefficients réels mais la méthode est encore valable dans le cas complexe.

Comme nous l'avons vu, la méthode de newton nécessite, à chaque itération, le calcul de la fonction et de la dérivée au point $x^{(k)}$. Le calcul peut être réalisé de la manière suivante avec un coût en temps de calcul réduit. Considérons pour cela le polynôme de degré n dont on veut la valeur de la dérivée au point $p = x^{(k)}$. Le polynôme de départ étant donné par

$$P_n(x) = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + ... + a_{n-1}x + a_n \tag{16}$$

La division euclidienne de $P(x)$ par $(x-p)$ donne comme quotient le polynôme $P_{n-1}(x)$ de degré $(n-1)$ et le reste constant R_0

$$P_n(x) = (x-p)P_{n-1}(x) + R_0 \tag{17}$$

avec

$$P_{n-1}(x) = b_0 x^{n-1} + b_1 x^{n-2} + ... + b_{n-2}x + b_{n-1} \tag{18}$$

On vérifie que le calcul des b_k et de R_0 peut se faire simplement par l'algorithme suivant

$$b_0 = a_0 \tag{19}$$

$$b_k = a_k + p.b_{k-1} \quad (k = 0,1,2,...) \tag{20}$$

$$R_0 = b_n \tag{21}$$

D'autre part la dérivation de $P_n(x)$ donne

$$P_n(x) = P_{n-1}(x) + (x-p)P'_{n-1}(x) \tag{22}$$

On en déduit donc que

$$P_n^{'}(p) = P_{n-1}(p) \tag{23}$$

Pour avoir la valeur de la dérivée de $P_n(x)$ au point p, il suffit donc de calculer la valeur de $P_{n-1}(x)$ au point p. Pour cela il suffit de considérer la division euclidienne de $P_{n-1}(x)$ par $(x-p)$ qui donne le quotient $P_{n-2}(x)$ et le reste R_1

$$\begin{aligned} P_{n-1}(x) &= (x-p)P_{n-2}(x) + R_1 \\ P_{n-2}(x) &= c_0 x^{n-2} + c_1 x^{n-2} + ... + c_{n-3}x + c_{n-2} \end{aligned} \tag{24}$$

D'après l'équation 24, on remarque que

$$P_n^{'}(p) = P_{n-1}(p) = R_1 \tag{25}$$

Il suffit de calculer R_1 pour avoir la dérivée de P_n. La détermination de R_1 se fait par le calcul des c_k en utilisant le même algorithme que pour les b_k

$$c_0 = b_0 \tag{26}$$

$$c_k = b_k + p.c_{k-1} \qquad (k = 1, 2, ...n-1) \tag{27}$$

$$R_1 = c_{n-1} = P_n^{'}(p) \tag{28}$$

Il suffira maintenant d'inclure cet algorithme de calcul de la dérivée dans l'algorithme de Newton.

Chapitre 4 Interpolation polynomiale et splines

1 Introduction

L'objectif est dans un premier temps de trouver une forme polynomiale passant par $(n+1)$ points. Cette forme sera définie sur le domaine entier ou par morceaux. On imposera alors selon le cas en plus de la continuité de la fonction, la continuité de sa dérivée et de ses dérivées successives.

2 Interpolation polynomiale

Supposons donnés $(n+1)$ points définis par les couples (x_i, y_i), $(i = 0, 1, ..., n)$. On démontre qu'il existe un et un seul polynôme de degré égal à n :

$$P(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n \tag{1}$$

vérifiant $P(x_i) = y_i$

Preuve : Supposons qu'il existe deux polynômes $P(x)$ et $Q(x)$ vérifiant cette propriété. Alors le polynôme de degré au plus égal à n

$$R(x) = P(x) - Q(x) \tag{2}$$

s'annule en $(n+1)$ points distincts x_i. Or le polynôme $R(x)$ étant de degré au plus égal à n ne peut avoir plus de n racines distinctes. On en déduit donc que $R(x)$ est nul et donc que $P(x) = Q(x)$.

2.1 Interpolation de Lagrange

2.1.1 Interpolation linéaire

Supposons donnés deux points (x_0, y_0) et (x_1, y_1). L'interpolation polynomiale de ces deux points est un polynôme de degré 1 qui correspond à la droite passant par les deux points. Ce polynôme est donné par

$$P(x) = y_0 + g(x)(y_1 - y_0) \tag{3}$$

avec

$$g(x) = \frac{x - x_0}{x - x_1} = \begin{cases} 0 & \text{si} \quad x = x_0 \\ 1 & \text{si} \quad x = x_1 \end{cases} \tag{4}$$

Que vaut le polynôme dans le cas général où nous avons $(n+1)$ points $(n > 1)$

2.1.2 Cas général

Supposons donnés $(n+1)$ points définis par les couples (x_i, y_i), $(i = 0, 1, ..., n)$.
On commence par définir $(n+1)$ polynômes de degré n, $l_i(x)$ $(i = 0, 1, ..., n)$ vérifiant

$$l_i(x_j) = \delta_{ij} = \begin{cases} 1 & si\ i = j \\ 0 & si\ i \neq j \end{cases}$$

Il est facile de voir que ces polynômes sont donnés par

$$l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \tag{5}$$

Les $(n+1)$ polynômes $l_i(x)$ forment une base du sous-espace vectoriel des polynômes de degré n. En effet si

$$\lambda_0 l_0(x) + \lambda_1 l_1(x) + ... + \lambda_n l_n(x) = 0 \tag{6}$$

alors

$$\lambda_0 l_0(x_k) + \lambda_1 l_1(x_k) + ... + \lambda_k l_k(x_k) + ... + \lambda_n l_n(x_k) = 0 \Longleftrightarrow \lambda_k l_k(x_k) = 0 \tag{7}$$

ce qui entraine

$$\lambda_k = 0 \qquad (k = 0, 1, 2, ...n) \tag{8}$$

Exemple 15 *Dans le cas $n = 2$, on obtient*

$$l_0(x) = \frac{x - x_1}{x_0 - x_1} \cdot \frac{x - x_2}{x_0 - x_2} \tag{9}$$

$$l_1(x) = \frac{x - x_0}{x_1 - x_0} \cdot \frac{x - x_2}{x_1 - x_2} \tag{10}$$

$$l_2(x) = \frac{x - x_0}{x_2 - x_0} \cdot \frac{x - x_1}{x_2 - x_1} \tag{11}$$

Ces polynômes apparaissent sur la figure 1 pour

$$x_0 = 0, x_1 = 1, x_2 = 2$$

$$\begin{aligned} l_0 &= \frac{1}{2}x^2 - \frac{3}{2}x + 1 \\ l_1 &= -x^2 + 2x \\ l_2 &= \frac{1}{2}x^2 - \frac{1}{2}x \end{aligned}$$

On vérifie alors que le poynôme

$$P(x) = \sum_{i=0}^{n} y_i l_i(x) \tag{12}$$

est le polynôme d'interpolation puisqu'il est de degré n et qu'il vérifie $P(x_i) = y_i$ $(i = 0, 1, ..., n)$. Ce polynôme est appelé polynôme d'interpolation de Lagrange.

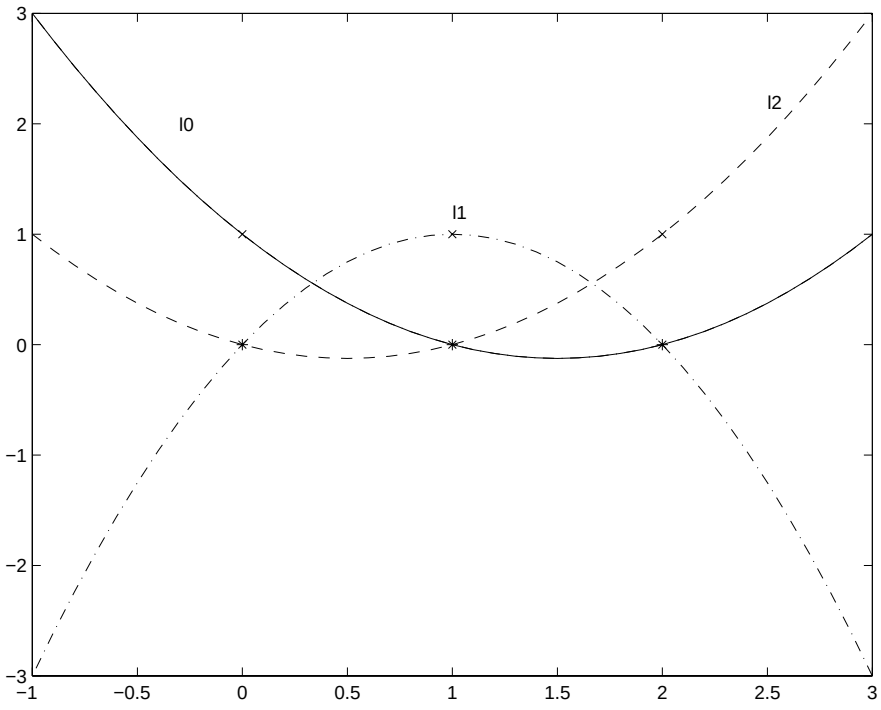

FIG. 1: Polynômes de Lagrange pour 3 points

2.1.3 Algorithme de calcul des coefficients

On démontre que le polynôme précédant peut s'écrire simplement :

$$P_n(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} = \sum_{i=0}^{n} y_i \lambda_i^{(n)} \prod_{j=0, j\neq i}^{n} (x - x_j) \tag{13}$$

Les coefficients $\lambda_i^{(n)}$ peuvent se calculer de manière récursive en utilisant la formule suivante :

$$\lambda_i^{(n+1)} = \lambda_i^{(n)} / (x_i - x_j) \qquad (i = 0, 1, ..., n) \tag{14}$$

On peut montrer par ailleurs l'égalité suivante :

$$\sum_{i=0}^{n} \lambda_i^{(n)} = 0 \tag{15}$$

L'algorithme de calcul des $\lambda_i^{(n)}$ peut se résumer de la manière suivante :

début $\lambda_0^{(0)}$=*1*
Pour $k = 1$ *à* n
    Pour $i = 0$ *à* $k-1$
        $\lambda_i^{(k)} = \lambda_i^{(k-1)} / (x_i - x_k)$
    fin
    $\lambda_k^{(k)} = -\sum_{i=0}^{k-1} \lambda_i^{(k)}$
fin

Les coefficients a_i du polynôme $P_n(x)$ peuvent alors être déduits des coefficients $\lambda_i^{(n)}$.

Exemple 16 *La figure 2 montre le polynôme d'interpolation de la fonction*

$$f(x) = \log x - 2\frac{x-1}{x} \tag{16}$$

aux points $[1, 2, 4, 8, 10]$

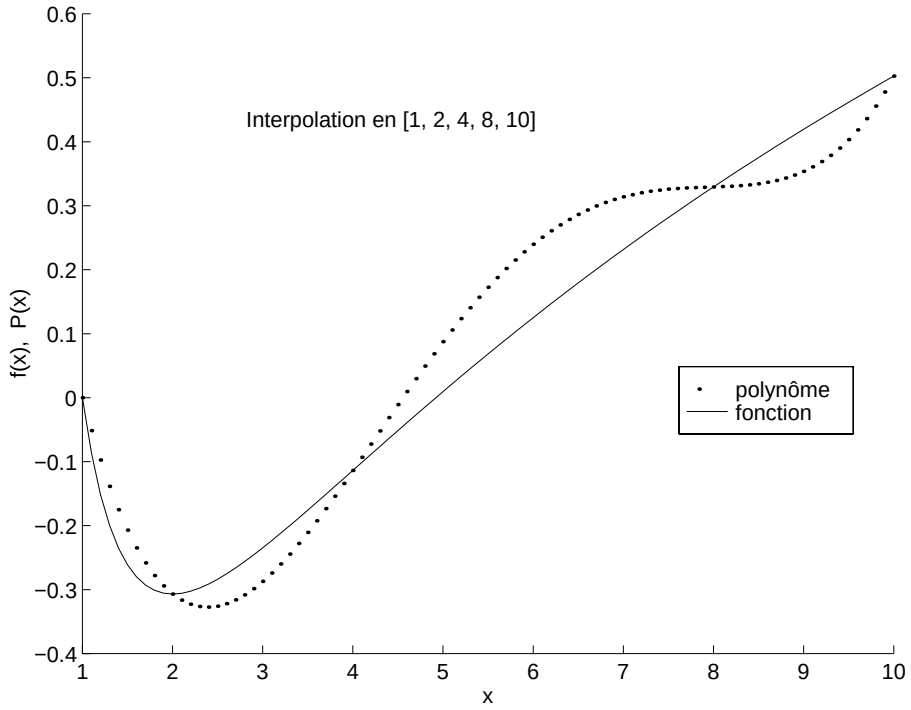

FIG. 2: Interpolation polynomiale

Remarque 12 *Lorsqu'on détermine un polynôme d'interpolation, on espère que le polynôme approche encore la fonction en dehors du domaine d'interpolation (extrapolation). Le tracé de la fonction précédente et de son polynôme d'interpolation sur l'intervalle* $[0.5, 12]$ *montre que ce n'est pas le cas. Il y a divergence rapide du polynôme par rapport à la fonction*

2.2 Interpolation de Newton

Le polynôme d'interpolation de Newton obtenu pour $(n+1)$ points $x_0, x_1, ..., x_n$ prend la forme :

$$P_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + ... + c_n(x - x_0)(x - x_1)...(x - x_{n-1}) \tag{17}$$

Il ne faut pas oublier que le polynôme de Newton sera égal au polynôme de Lagrange (le polynôme d'interpolation étant unique). Il diffèrent simplement dans la forme et par l'algorithme de calcul des coefficients.

Les coefficients c_i peuvent se calculer par la résolution du système d'équations linéaires de forme triangulaire inférieure. La résolution est donc directe par simple substitution comme le montrent les équations ci-dessous

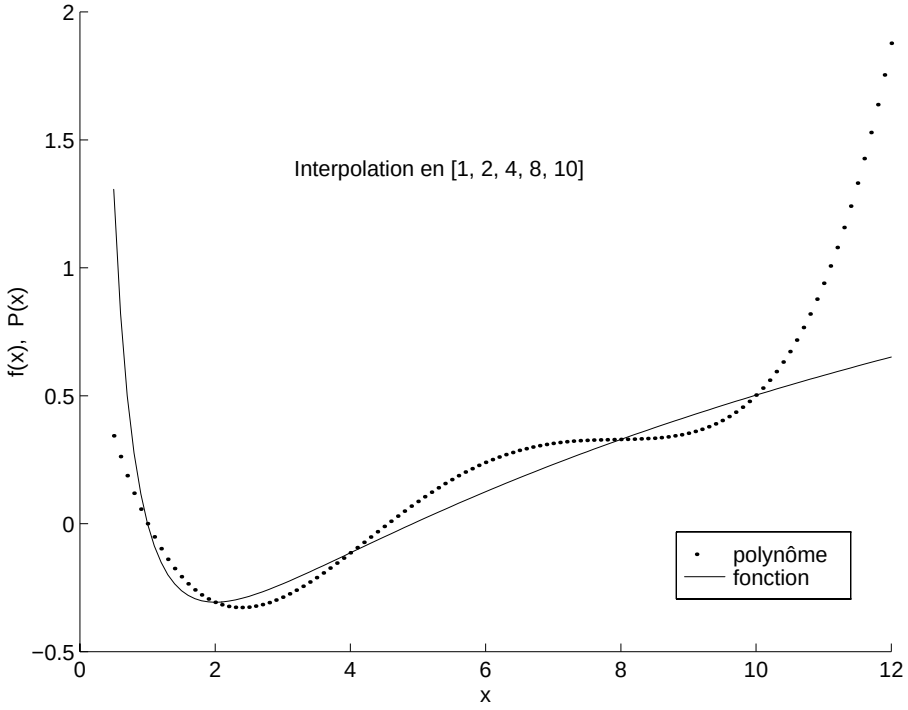

FIG. 3: Interpolation / Extrapolation polynomiale

$$\left[\begin{array}{l} P_n(x_0) = c_0 = y_0 \\ P_n(x_1) = c_0 + c_1(x_1 - x_0)) = y_1 \\ .... \\ P_n(x_n) = c_0 + c_1(x_n - x_0) + c_2(x_n - x_0)(x_n - x_1) + ... + c_n(x_n - x_0)...(x_n - x_{n-1}) = y_n \end{array}\right.$$

Quoi qu'il en soit, une autre méthode permet d'obtenir un algorithme itératif sur le nombre de points. C'est ici une différence majeure avec le polynôme de Lagrange pour lequel, si on ajoute un point, on est obligé de recalculer tous les coefficients du polynôme.

Remarquons d'abord que le terme c_i ne dépend que des x_k avec $(k = 0, ..., i)$.

$$c_0 = y_0 \tag{18}$$

$$c_1 = \frac{y_1 - c_0}{x_1 - x_0} \tag{19}$$

$$c_2 = \frac{y_2 - y_0 - c_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} \tag{20}$$

$$= \frac{y_2 - y_0}{(x_2 - x_0)(x_2 - x_1)} - \frac{y_1 - y_0}{(x_1 - x_0)(x_2 - x_1)} \tag{21}$$

$$= \frac{(y_2 - y_0)(x_1 - x_0) - (y_1 - y_0)(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)(x_1 - x_0)} \tag{22}$$

$$= \frac{\frac{y_1 - y_0}{x_1 - x_0} - \frac{y_2 - y_1}{x_2 - x_1}}{(x_2 - x_0)} \tag{23}$$

La formule précédente se généralise sans difficulté, le polynôme d'interpolation de Newton peut donc être calculé comme suit en construisant le tableau :

$$
\begin{array}{llllllll}
x_0 & f(x_0) & & & & & & \\
 & & f[x_0,x_1] & & & & & \\
x_1 & f(x_1) & & f[x_0,x_1,x_2] & & & & \\
 & & f[x_1,x_2] & & f[x_0,x_1,x_2,x_3] & & & \\
x_2 & f(x_2) & & f[x_1,x_2,x_3] & : & : & & \\
 & & f[x_2,x_3] & : & : & : & : & f[x_0,...,x_n] \\
x_3 & f(x_3) & : & : & : & : & : & \\
: & : & : & : & : & : & & \\
: & : & : & : & f[x_{n-3}x_{n-2},x_{n-1},x_n] & & & \\
x_{n-1} & f(x_{n-1}) & : & f[x_{n-2},x_{n-1},x_n] & & & & \\
 & & f[x_{n-1},x_n] & & & & & \\
x_n & f(x_n) & & & & & &
\end{array}
$$

avec

$$
\begin{aligned}
f(x_i) &= y_i && (24)\\
c_i &= f\left[x_0,x_1,...,x_i\right] && (25)\\
f[x_i,x_j] &= \frac{f(x_j)-f(x_i)}{x_j-x_i} && (26)\\
f[x_i,x_j,x_k] &= \frac{f[x_j,x_k]-f[x_i,x_j]}{x_k-x_i} && (27)
\end{aligned}
$$

2.3 Estimation de l'incertitude

Soit une fonction définie sur $[a,b]$ et $(n+1)$ points $(x_i, y_i = f(x_i))$. Que peut-on dire de la différence $f(x) - P(x)$, où $P(x)$ est le polynôme d'interpolation de f ? Nous avons aux points d'interpolation :

$$P(x_i) = f(x_i) = y_i \tag{28}$$

Cette différence peut être faible ou grande comme le montre la figure 4

Exemple 17

$$f(x) = \tfrac{1}{x} \qquad x \in [1,3] \tag{29}$$

$$x_0 = 1, \quad f(1) = 1 \quad x_1 = 3, \quad f(3) = \tfrac{1}{3} \tag{30}$$

alors

$$P(x) = -\frac{1}{3}x + \frac{4}{3} \tag{31}$$

Dans ce cas on peut dire que la différence est faible mais on remarque que le polynôme $P(x)$ est aussi polynôme d'interpolation de toute fonction vérifiant la condition 30, et la différence peut alors être élevée.

On peut donc dire que, plus la fonction est régulière, meilleure sera l'approximation. Le théorème suivant donne la valeur de cette erreur.

Théorème 10 *si*

$$\begin{cases} f \in C^{n+1}[a,b] \\ a \le x_0 \le x_1 \le ... \le x_n \le b \end{cases} \tag{32}$$

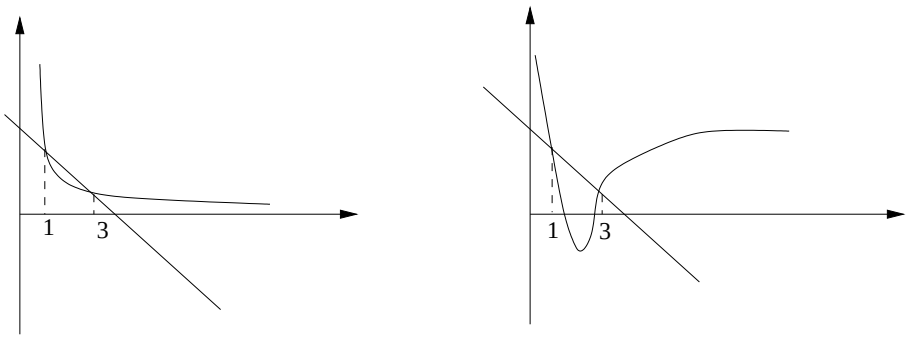

FIG. 4: Ecart entre la fonction et le polynôme d'interpolation

alors

$$f(x) - P(x) = \frac{1}{(n+1)!} \prod_{i=0}^{n} (x - x_i) f^{(n+1)}(\xi) \tag{33}$$

où

$$a \leq \min(x_0, x) < \xi < \max(x, x_n) \leq b \tag{34}$$

Preuve : *non développée ici*

Exemple 18 *L'application de ce théorème dans le cas de l'interpolation linéaire avec*

$$x_0 = a \qquad x_1 = b \tag{35}$$

et donc

$$P(x) = f(a).\frac{x-b}{a-b} + f(b).\frac{x-a}{b-a} \tag{36}$$

donne

$$f(x) - P(x) = \frac{(x-a)(x-b)}{2} f''(\xi) \tag{37}$$

Il faut donc trouver un majorant de $|f''(\xi)|$

2.3.1 Choix des x_i au mieux

D'après le théorème précédent,

$$|f(x) - P(x)| \leq \max_{x \in [a,b]} \left| \frac{f^{(n+1)}(\xi)}{(n+1)!} \right| \max_{x \in [a,b]} \left| \prod_{i=0}^{n} (x - x_i) \right| \tag{38}$$

L'erreur résulte donc de deux termes :

- le terme de dérivée $(n+1)$ ème dépendant de la fonction à interpoler, on ne peut donc rien y changer, il sera simplement d'autant plus faible que la fonction est régulière.
- Par contre le deuxième terme dépend du choix des x_i

$$\max_{x \in [a,b]} \left| \prod_{i=0}^{n} (x - x_i) \right| \tag{39}$$

Le problème est le suivant : comment choisir les x_i pour rendre cette quantité minimale ? On démontre que cette quantité est minimale pour les points x_i annulant les polynômes de Chebychev c'est-à-dire

$$x_i = \frac{b-a}{2} \cos\left(\frac{n-i}{n}\pi\right) + \frac{b+a}{2} \tag{40}$$

Exemple 19 *La figure ci-dessous montre le cas de la fonction de Runge avec des points régulièrement espacés et des points suivant une répartition de Chebychev. Le polynôme de degré 6 en traits interrompus est obtenu aux points d'interpolation*

$$x_0 = -5, x_1 = -3, x_2 = -1, x_3 = 0, x_4 = 1, x_5 = 3, x_6 = 5$$

Le polynôme en traits pointillés est obtenu avec la répartition des points donnés par la formule 40 pour $a = -5$, $b = 5$ *et* $n = 6$. *La répartition des points est donnée sur la figure 6*

$$x'_0 = -5, x'_1 = -4.3301, x'_2 = -2.5, x'_3 = 0, x'_4 = 2.5, x'_5 = 4.3301, x'_6 = 5$$

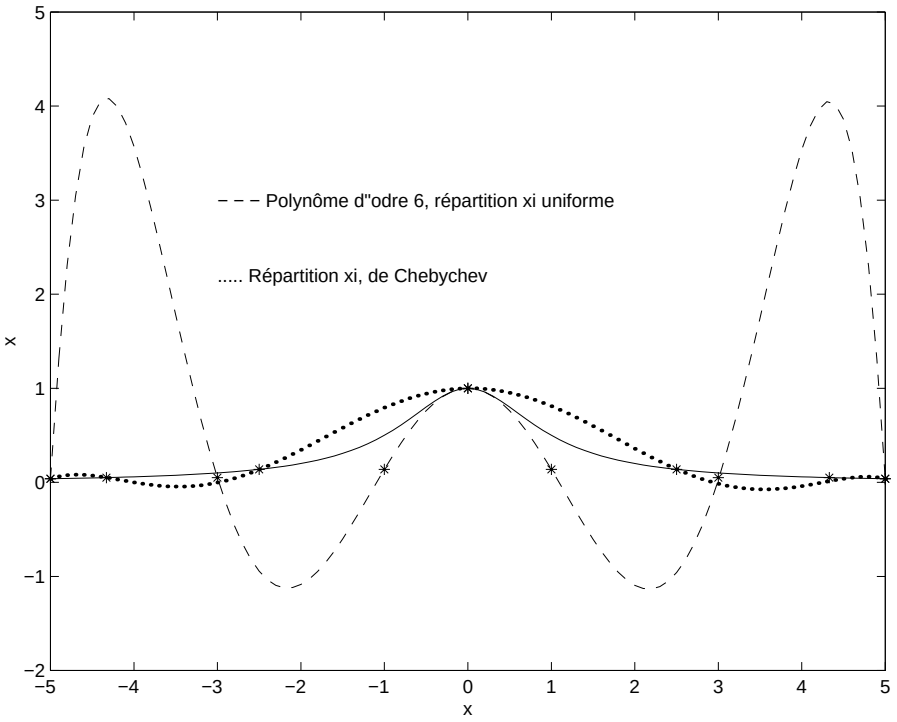

FIG. 5: Fonction de Runge, répartition uniforme et de Chebychev des x_i

3 Interpolation par des Splines

Lorsqu'on réalise une interpolation, on espère généralement que l'interpolation marche encore en dehors des points pris en compte pour calculer le polynôme. Or comme cela a été remarqué précédemment, l'interpolation polynomiale diverge rapidement en dehors des points d'interpolation. Ceci est aussi vrai sur les bords du domaine comme c'est le cas pour la fonction de Runge où des oscillations apparaissent lorsque le nombre de points est supérieur à 10. Une meilleure répartition des points (répartition de Chebychev) permet de réduire l'erreur mais des oscillations subsistent. De plus, l'utilisation de polynômes de degré élevé est à éviter puisqu'on introduit rapidement des instabilités numériques.

On leur préfère alors une interpolation polynomiale par morceaux appelée spline. La spline peut changer de forme aux points (x_i, y_j) appelés noeuds.

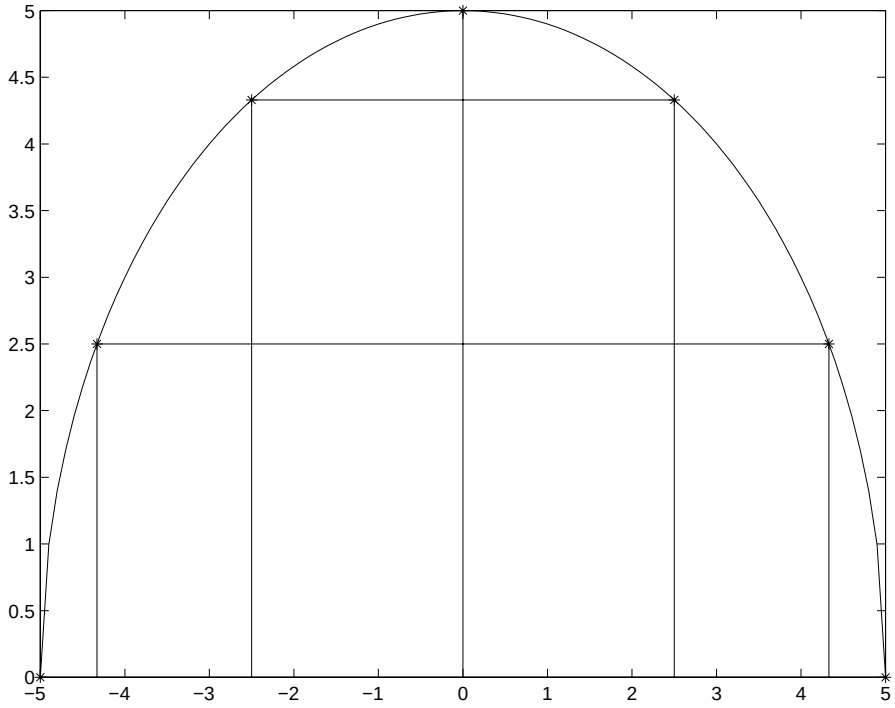

FIG. 6: Répartition de Chebychev des x_i

Définition 1 *Étant donnés* $(n+1)$ *points* (x_i, y_i) *, la spline* $S(x)$ *est définie par*

$$S(x) = \begin{cases} S_0(x), & x \in [x_0, x_1] \\ S_1(x), & x \in [x_1, x_2] \\ \\ S_{n-1}(x), & x \in [x_{n-1}, x_n] \end{cases} \tag{41}$$

Lorsque les polynômes $S_i(x)$ *sont de degré 1, on parle de spline linéaire ; quand ils sont de degré 2, on parle de spline quadratique. S'ils sont de degré 3, on parle de spline cubique.*

3.1 Splines linéaires

Étant donnés $(n+1)$ points (x_i, y_i) , on cherche une spline linéaire $S(x)$ de la forme

$$S(x) = \begin{cases} S_0(x) = a_0 x + b_0, & x \in [x_0, x_1] \\ S_1(x) = a_1 x + b_1, & x \in [x_1, x_2] \\ \\ S_{n-1}(x) = a_{n-1} x + b_{n-1}, & x \in [x_{n-1}, x_n] \end{cases} \tag{42}$$

vérifiant la condition d'interpolation

$$S(x_i) = y_i \tag{43}$$

La première question à laquelle if faut répondre est : la spline $S(x)$ est-elle unique ?

La détermination de $S(x)$ nécessite le calcul de $2n$ coefficients a_i et b_i, $(i = 0, 1, ..., n-1)$. Pour cela on dispose de $2n$ équations :

– $(n+1)$ équations d'interpolation $(i = 0, 1, ..., n)$

$$S_i(x_i) = y_i \tag{44}$$

$$y_i = a_i x_i + b_i \tag{45}$$

– $(n-1)$ équations de continuité de la spline $(i = 0, 1, ..., n-2)$

$$S_i(x_{i+1}) = S_{i+1}(x_{i+1}) \qquad (i = 0, 1, ..., n-2) \tag{46}$$

$$a_i x_{i+1} + b_i = y_{i+1} \tag{47}$$

On en déduit donc de manière unique

$$a_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \ (i = 0, 1, ..., n-1) \tag{48}$$

$$b_i = y_i - a_i x_i, \ (i = 0, 1, ..., n-1) \tag{49}$$

3.2 Splines quadratiques

Nous avons vu précédemment que la spline linéaire ne permettait d'assurer que la continuité de la fonction $S(x)$. Si on veut que $S(x)$ soit continuement dérivable, il faut donc choisir des polynômes $S_i(x)$ de degré au moins égal à 2.

Définition 2 *Supposons donnés $(n+1)$ points définis par les couples (x_i, y_i), $(i = 0, 1, ..., n)$. L'interpolation par des splines quadratiques équivaut à l'approximation sur chaque intervalle $[x_i, x_{i+1}]$ par un polynôme du second degré. On impose aux différents points la continuité des valeurs des polynômes et de leurs dérivées.*

$$S(x) = \begin{cases} S_0(x) = a_0 x^2 + b_0 x + c_0, & x \in [x_0, x_1] \\ S_1(x) = a_1 x^2 + b_1 x + c_1, & x \in [x_1, x_2] \\ \\ S_{n-1}(x) = a_{n-1} x^2 + b_{n-1} x + c_{n-1}, & x \in [x_{n-1}, x_n] \end{cases} \tag{50}$$

Exemple 20 *La fonction*

$$f(x) = \begin{cases} x^2 & x \leq 0 \\ -x^2 & 0 \leq x \leq 1 \\ 1 - 2x & x \geq 1 \end{cases}$$

représente une spline quadratique. En effet :

– elle est définie sur $]-\infty, +\infty[$

– elle est continue sur $]-\infty, +\infty[$

– sa dérivée est continue sur $]-\infty, +\infty[$

  * $f'(0^-) = f'(0^+) = 0$
  * $f'(1^-) = f'(1^+) = -2$

La spline $S(x)$ est-elle unique ? La détermination de $S(x)$ nécessite le calcul des $3n$ coefficients a_i et b_i, $(i = 0, 1, ..., n-1)$. Pour cela on dispose seulement de $(3n-1)$ équations :

– $(n+1)$ équations d'interpolation $(i = 0, 1, ..., n)$

$$S_i(x_i) = y_i \tag{51}$$

$$y_i = a_i x_i^2 + b_i x + c_i \tag{52}$$

– $(n-1)$ équations de continuité de la spline

$$S_i(x_{i+1}) = S_{i+1}(x_{i+1}) \qquad (i = 0, 1, ..., n-2) \tag{53}$$

– $(n-1)$ équations de continuité de sa dérivée

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}) \qquad (i = 0, 1, ..., n-2) \tag{54}$$

Pour pouvoir résoudre le problème, il est donc nécessaire de fixer une condition supplémentaire. Pour cela définissons la variable auxiliaire $z_i = S'(x_i), (i = 0, 1, ..., n)$ et récrivons $S_i(x)$ sous la forme

$$S_i(x) = a_i(x - x_i)^2 + b_i(x - x_i) + c_i \tag{55}$$

Des conditions d'interpolation, il apparaît clairement que, sous cette forme, les c_i sont déterminés de manière unique

$$c_i = S_i(x_i) = y_i \qquad (i = 0, 1, ..., n-2) \tag{56}$$

De plus, par définition

$$S_i'(x_i) = b_i = z_i \tag{57}$$

ce qui détermine les b_i. En écrivant maintenant les conditions de continuité de la dérivée, on obtient

$$2a_i(x_{i+1} - x_i) + b_i = 2a_{i+1}(x_{i+1} - x_{i+1}) + b_{i+1} \qquad (i = 0, 1, ..., n-2) \tag{58}$$

En utilisant 57,on obtient les a_i

$$a_i = \frac{z_{i+1} - z_i}{2\,(x_{i+1} - x_i)} \tag{59}$$

$S_i(x)$ se met donc sous la forme

$$S_i(x) = \frac{z_{i+1} - z_i}{2\,(x_{i+1} - x_i)}(x - x_i)^2 + z_i(x - x_i) + y_i \tag{60}$$

Il ne reste plus qu'à calculer les z_i. Pour cela on écrit les équations de continuité de la spline au point x_{i+1}

$$\begin{aligned} S_i(x_{i+1}) &= \frac{z_{i+1} - z_i}{2\,(x_{i+1} - x_i)}(x_{i+1} - x_i)^2 + z_i(x_{i+1} - x_i) + y_i \\ &= \frac{z_{i+1} - z_i}{2}(x_{i+1} - x_i) + z_i(x_{i+1} - x_i) + y_i \\ &= y_{i+1} \end{aligned} \tag{61}$$

On en déduit donc que

$$z_{i+1} = 2\frac{y_{i+1} - y_i}{x_{i+1} - x_i} - z_i \qquad (i = 0, 1, ..., n-1) \tag{62}$$

Et on voit apparaître naturellement la condition supplémentaire qui sera ici une condition d'initialisation de la récurrence sur les z_i. On doit donc choisir

$$z_0 = S_0'(x_0) \tag{63}$$

Remarque 13 *On peut donner n'importe quelle valeur à z_0, mais généralement on choisit $z_0 = 0$. Notons qu'au lieu de fixer z_0, on aurait pu fixer z_n et faire une récurrence descendante.*

Les figures 7 et 8 montrent les résultats obtenus pour la fonction de Runge respectivement aux points $[-5, -3, -1, 1, 3, 5]$ et $[-5, -3, 0, -1, 1, 3, 5]$. On remarque que le fait d'ajouter le point 0 améliore les résultats au voisinage de 0 mais dégrade les résultats sur l'intervalle $[2, 5]$. On observe en effet des oscillations de forte amplitude.

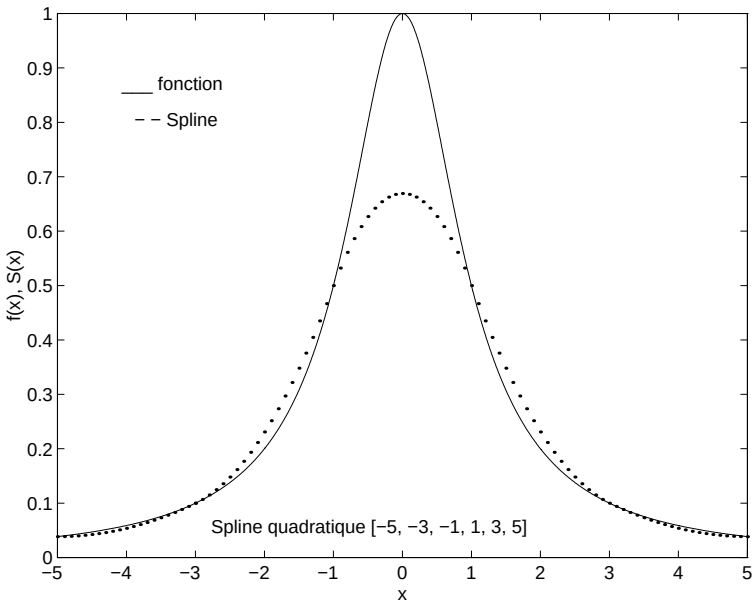

FIG. 7: Spline quadratique aux points $[-5, -3, -1, 1, 3, 5]$

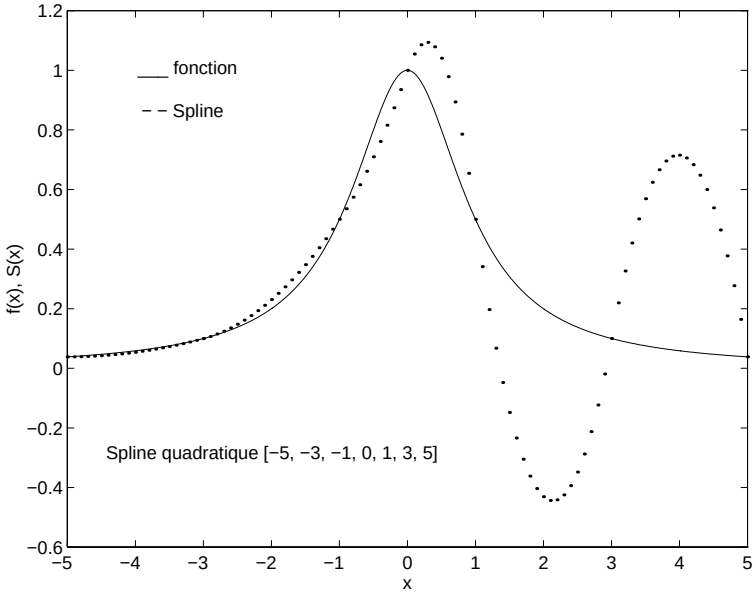

FIG. 8: Spline quadratique aux points $[-5, -3, -1, 0, 1, 3, 5]$

3.3 Splines cubiques

L'interpolation par des splines cubiques (polynômes de degré 3) entraîne la continuité de la spline, de sa dérivée et de la dérivée seconde. Définissons $y_i'' = S''(x_i)$, et écrivons $S_i(x)$ sous la forme

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \tag{64}$$

Ayant $4n$ coefficients à déterminer, nous ne disposons cette fois-ci que de $(4n - 1)$ équations ; les $(3n - 1)$ conditions identiques à celles de la spline quadratique et les $(n - 1)$ conditions supplémentaires de continuité de la dérivée seconde. Il faut donc fixer deux conditions supplémentaires. On impose généralement les valeurs des dérivées secondes aux points x_0 et x_n.

$$y_0'' = S_0(x_0) \tag{65}$$

et

$$y_n'' = S_{n-1}(x_n) \tag{66}$$

Les conditions d'interpolation aux points x_i donnent

$$d_i = y_i \tag{67}$$

Les conditions de continuité de la dérivée et de la dérivée seconde entraînent

$$a_i = \frac{1}{6h_i}(y_{i+1}'' - y_i'') \tag{68}$$

$$b_i = \frac{1}{2}y_i'' \tag{69}$$

$$c_i = \frac{1}{h_i}(y_{i+1} - y_i) - \frac{1}{6}h_i(y_{i+1}'' + 2y_i'') \tag{70}$$

avec

$$h_i = x_{i+1} - x_i \tag{71}$$

Il suffit donc maintenant de calculer les y_i'', $(i = 0, 1, ..., n - 2)$.

L'écriture de la continuité de la spline en utilisant les égalités précédentes donne le système d'équations à $(n - 2)$ inconnues permettant d'obtenir les y_i'', $(i = 1, 1, ..., n - 2)$

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & 0 & . & . & . & 0 & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & . & . & . & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & 0 & . & . & 0 \\ 0 & 0 & h_3 & . & . & . & . & . \\ . & & & . & . & . & . & . \\ . & & & & . & . & h_{n-3} & 0 \\ 0 & 0 & . & . & . & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \\ 0 & 0 & 0 & . & . & . & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix}$$

$$\begin{bmatrix} y''(1) \\ y''(2) \\ y''(3) \\ . \\ . \\ . \\ y''(n-2) \\ y''(n-1) \end{bmatrix} = \begin{bmatrix} -\frac{6}{h_0}(y_1 - y_0) + \frac{6}{h_1}(y_2 - y_1) - h_0 y_0'' \\ -\frac{6}{h_1}(y_2 - y_1) + \frac{6}{h_2}(y_3 - y_2) \\ -\frac{6}{h_2}(y_3 - y_2) + \frac{6}{h_3}(y_4 - y_3) \\ . \\ . \\ . \\ -\frac{6}{h_{n-3}}(y_{n-2} - y_{n-3}) + \frac{6}{h_{n-2}}(y_{n-1} - y_{n-2}) \\ -\frac{6}{h_{n-2}}(y_{n-1} - y_{n-2}) + \frac{6}{h_{n-1}}(y_n - y_{n-1}) - h_{n-1} y_n'' \end{bmatrix}$$

La figure 9 montre un exemple de spline cubique obtenue pour la fonction de Runge aux points d'interpolation

$$x_0 = -5, x_1 = -3, x_2 = -1, x_3 = 0, x_4 = 1, x_5 = 3, x_6 = 5$$

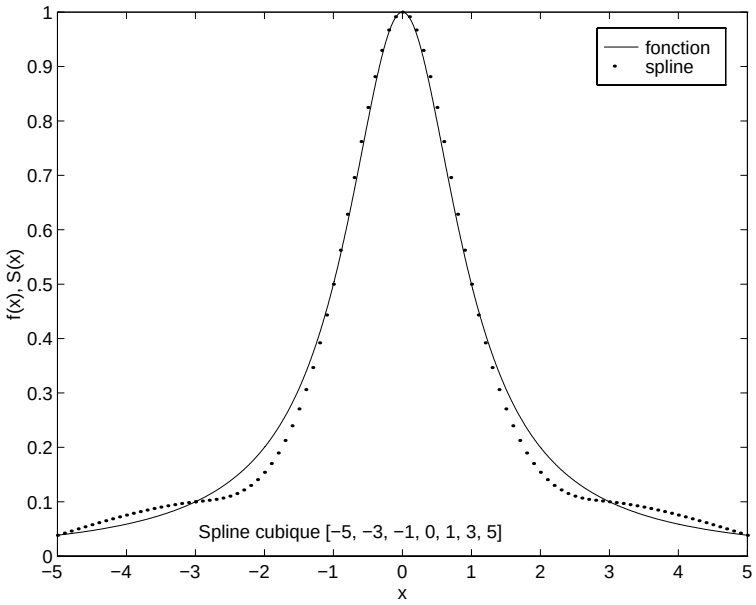

FIG. 9: Spline cubique aux points $[-5, -3, -1, 0, 1, 3, 5]$

On remarquera que dans ce cas les oscillations dont très réduites par rapport à l'interpolation polynomiale.

Remarque 14 *Quand on choisit*

$$y_0'' = y_n'' = 0 \tag{72}$$

la spline cubique est dite naturelle.

Théorème 11 (Optimalité de la Spline cubique naturelle) *Soit f une fonction C^2 sur l'intervalle $[a, b]$, et soit $(n+1)$ points*

$$a = x_0, x_1, ..., x_n = b \tag{73}$$

et

$$f(a) = y_0, y_1, ..., y_n = f(b) \tag{74}$$

alors la spline cubique naturelle $S(x)$ qui réalise l'interpolation de f aux points (x_i, y_i) vérifie

$$\int_a^b \left[S''(x)\right]^2 \, dx \leq \int_a^b \left[f''(x)\right]^2 \, dx \tag{75}$$

En d'autres termes, la courbure moyenne de S est inférieure à celle de f.

Chapitre 5 Programmation linéaire

1 Introduction

On parle de programmation linéaire lorsque le critère à optimiser (à minimiser ou à maximiser) est une fonction linéaire des variables. Le terme de programmation vient du terme prévision ou planification, au sens de la programmation d'un voyage ou d'une production.

On rencontre ce type de problème dans différents domaines mais surtout en économie et maintenant en automatique.

Soit $x = [x_1, x_2, ..., x_n]^T$ le vecteur de variables d'optimisation. On cherche la solution optimale x_{opt} minimisant ou maximisant un critère $f(x)$ sous un ensemble de contraintes d'égalités

$$\begin{aligned} x_{opt} &= Arg \max_x f(x) \\ H(x) &\geq 0 \\ x &\geq 0 \end{aligned} \tag{1}$$

– Le critière étant linéaire, la fonction f peut se mettre sous la forme

$$f(x) = f_0 + \sum_{i=1}^{n} f_i x_i \tag{2}$$

– De même la contrainte $H(x)$ étant elle aussi linéaire alors

$$H(x) = \left[\sum_{i=1}^{n} a_{i1} x_i + b_1, \sum_{i=1}^{n} a_{i2} x_i + b_2, ..., \sum_{i=1}^{n} a_{im} x_i + b_m \right]^T \tag{3}$$

– Un problème de minimisation peut être transformé en maximisation par changement de signe

2 Exemple plan

Nous présentons d'abord un exemple de programmation linéaire à deux variables car celui-ci permet une représentation simple de chacune des contraintes dans le plan délimité par les deux variables x_1 et x_2. Dans ce cas, chacune des contraintes correspond à une droite qui subdivise le plan en deux parties : une partie ou la contrainte est positive et l'autre ou la contrainte et négative.

Pour une valeur donnée de $f(x) = F_0$, le critère est aussi une droite. On obtient donc une famille de droites.

Considérons l'exemple d'une entreprise fabriquant deux produits p_1 et p_2 en quantité x_1 et x_2. Le produit p_1 est vendu trois fois plus cher que le produit p_2, mais le volume de p_1 est 4 fois celui de p_2. Le volume de la surface de stockage de l'entreprise est limité à 1. Quelles quantités x_1 et x_2 de produit faudra-t-il produire pour maximiser l'actif de l'entreprise ?

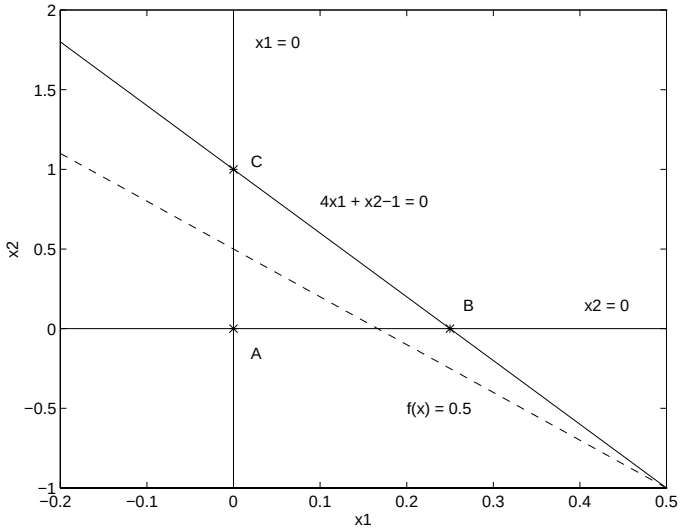

FIG. 1: Problème plan

Le problème d'optimisation qu'il faut résoudre est donc :

$$\begin{aligned} \max_{x_1,x_2} f &= 3x_1 + x_2 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \\ 4x_1 + x_2 &\leq 1 \end{aligned}$$

La solution de ce problème est très facile à obtenir graphiquement comme le montre la figure 1. Les contraintes forment un triangle de sommets (A, B, C). La solution est donc forcément à l'intérieur de ce triangle ou sur les sommets. Dans ce cas, la solution est bien obtenue au point C. En effet si on trace les droites $f(x) = cste$, on obtient un résaux de droites parallèles à la droite $(f(x) = 0.5)$ qui apparaît sur la figure 1. Ces droites sont telles que :

- $f(x) < 0$, la droite passe à gauche du point A, les contraintes 1 et 2 ne sont pas vérifiées.
- $f(x) = 0$, la droite passe par A, les trois contraintes sont vérifiées, c'est une solution possible $(x_1 = x_2 = 0)$.
- $f(x) = \frac{3}{4}$, la droite passe par B, les trois contraintes sont vérifiées, c'est une solution possible $(x_1 = \frac{1}{4}, x_2 = 0)$.
- $f(x) = 1$, la droite passe par C, les trois contraintes sont vérifiées, c'est une solution possible $(x_1 = 0, x_2 = 1)$.
- $f(x) > 1$, la droite passe à droite des points B et C, la contrainte 3 n'est plus vérifiée.

La solution est donc bien au point C.

Cette figure appelle aussi quelques remarques qui, en fait, sont générales aux problèmes de programmation linéaire.

Remarque 15 – *Le domaine de faisabilité est un polygone dans le cas $n = 2$.*

- *Dans le cas $n = 2$, on remarque que le polygone est convexe. Dans le cas $n > 2$, on parle de polyèdre convexe.*
- *La solution est un des sommets du polygone. Ceci reste vrai dans les problèmes de dimension supérieure à 2 à quelques exceptions près où la droite $f = f_0$ (f_0 constante) est parallèle à un des cotés du polygône. Dans ce cas, on a une infinité de solutions.*

3 Méthode du Simplexe

La solution d'un problème de programmation linéaire, étant sur un des sommets du polyèdre, on utilise pour sa résolution une méthode qui teste successivement les différents sommets. C'est la méthode du *Simplex*.

Le principe est relativement simple. On se déplace sur le polyèdre de sommet en sommet en assurant la croissance du critère. Définissons pour cela :

$$y_j = \sum_{i=1}^{n} a_{ij} x_i + b_j \tag{4}$$

On peut donc adopter une notation du problème sous forme de système linéaire

$$\begin{array}{cccccccc} & x_1 & \dots & x_q & \dots & x_n & 1 \\ y_1 = & a_{11} & \dots & a_{1q} & \dots & a_{1n} & b_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_i = & a_{i1} & \dots & a_{iq} & \dots & a_{in} & b_i \\ \vdots & & & & & & \vdots \\ y_p & a_{p1} & \dots & a_{pq} & \dots & a_{pn} & b_p \\ \vdots & \vdots & & \vdots & & \vdots & \vdots \\ y_m = & a_{m1} & \dots & a_{mq} & \dots & a_{mn} & b_m \\ f = & f_1 & \dots & f_q & \dots & f_n & f_0 \end{array} \tag{5}$$

Le but de la méthode est d'inverser les rôles de x_q et de y_q. On dit alors qu'on choisit a_{qp} comme pivot, puis on exprime x_q en fonction de y_p et des x_i avec $(i \neq q)$ puis on remplace x_q par l'expression obtenue dans les différentes équations de y_j $(j \neq i)$. Les pivots successifs a_{pq} sont choisis selon les deux règles suivantes :

Règle 1 La colonne q du pivot doit être choisie de telle sorte que l'élément f_q soit positif ou nul. Si plusieurs colonnes vérifient cette propriété, alors on peut choisir la première colonne ou la colonne présentant la plus grande valeur f_q.

Règle 2 Le pivot a_{pq} doit être négatif. La ligne est alors choisie de telle sorte que la quantité $|b_p/a_{pq}|$ soit la plus faible.

Lorsque l'ensemble des éléments f_i sont tous négatifs ou nuls, alors l'algorithme s'arrête et la valeur du critère vaut alors f_0^*.

$$f(x) = f_0^* + \sum_{i=1}^{n} f_i^* x_i \tag{6}$$

avec $f_i^* \leq 0$

Le choix du pivot a_{pq} permet d'exprimer x_q en fonction des x_i $(i \neq q)$ et de y_p

$$x_q = \frac{1}{a_{pq}} \left[-b_p + y_p - \sum_{i=1, i\neq q}^{n} a_{pi} x_i \right] \tag{7}$$

En remplaçant x_q par l'équation 7 dans les expressions de y_i $(i \neq p)$ et de y_p, on obtient

$$y_i = \sum_{k=1, k\neq q}^{n} a_{i,k} x_k + a_{iq} \left[\frac{1}{a_{pq}} \left[-b_p + y_p - \sum_{i=1, i\neq q}^{n} a_{pi} x_i \right] \right] \tag{8}$$

$$= -\frac{a_{iq}}{a_{pq}} b_p + \frac{a_{iq}}{a_{pq}} y_p + \sum_{k=1, k\neq q}^{n} \left(a_{i,k} - -\frac{a_{iq}}{a_{pq}} a_{pi} \right) x_k \qquad (i \neq p) \tag{9}$$

et

$$
\begin{aligned}
f &= f_0 + \sum_{i=1}^{n} f_i x_i && (10)\\
&= f_0 + \sum_{i=1, i\neq q}^{n} f_i x_i + f_q \frac{1}{a_{pq}} \left[-b_p + y_p - \sum_{j=1, i\neq q}^{n} a_{pi} x_i \right] && (11)\\
&= \left(f_0 - f_q \frac{b_p}{a_{pq}} \right) + \frac{f_q}{a_{pq}} y_p - \sum_{i=1, i\neq q}^{n} \left(f_i - f_q \frac{a_{pi}}{a_{pq}} \right) x_i \qquad (i \neq q) && (12)
\end{aligned}
$$

On obtient finalement le système ci-dessous ou x_q et y_p ont été permutés.

$$
\begin{array}{lcccccc}
 & x_1 & \dots & y_p & \dots & x_n & 1 \\
y_1 = & a'_{11} & \dots & a'_{1q} & \dots & a'_{1n} & b'_1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
y_i = & a'_{i1} & \dots & a'_{iq} & \dots & a'_{in} & b'_i \\
\vdots & & & & & & \vdots \\
x_q = & a'_{p1} & \dots & a'_{pq} & \dots & a'_{pn} & b'_p \\
\vdots & \vdots & & \vdots & & \vdots & \vdots \\
y_m = & a'_{m1} & \dots & a'_{mq} & \dots & a'_{mn} & b'_m \\
f = & f_1 & \dots & f'_q & \dots & f'_n & f'_0
\end{array}
\qquad (13)
$$

avec

$$
\begin{aligned}
& a'_{pq} = \tfrac{1}{a_{pq}} \\
& a'_{pk} = -\tfrac{a_{pk}}{a_{pq}} \ (k \neq q), \qquad b'_p = -\tfrac{b_p}{a_{pq}} \\
& a'_{iq} = \tfrac{a_{iq}}{a_{pq}} (i \neq p) \\
& a'_{ik} = a_{ik} + a_{iq} a'_{pk} \qquad (i \neq p, k \neq q) \\
& b'_i = b_i + a_{iq} b'_p \qquad (i \neq p) \\
& f^*_0 = f_0 - \tfrac{f_q b_p}{a_{pq}}
\end{aligned}
$$

$$
\begin{aligned}
& f^*_i = f_i - f_q \tfrac{a_{pi}}{a_{pq}} \qquad (i \neq q) \\
& f^*_q = \tfrac{f_q}{a_{pq}}
\end{aligned}
$$

Remarque 16 *Le système précédent suppose que l'origine est une solution initiale possible du problème. Ceci n'est pas toujours le cas. Il faudra alors déterminer une solution initiale soit à la main, soit en résolvant un sous problème de programmation linéaire.*

Théorème 12 *La solution du problème 1 est obtenue en un nombre fini de pivots si*

- *les éléments b_i sont strictement positifs pendant toutes les étapes de l'algorithme*
- *il est possible de choisir des pivots vérifiant les deux règles précédentes.*

Théorème 13 *Le problème de programmation linéaire admet une solution unique si les conditions du théorème précédent sont vérifiées et si de plus le critère à l'arrêt de l'algorithme du simplexe vérifie $f^*_i < 0$.*

Application à l'exemple précédent

$$
\begin{array}{lccc}
 & x_1 & x_2 & 1 \\
y_1 = & \text{-4} & \text{-1} & 1 \\
f = & 3 & 1 & 0
\end{array}
\qquad (14)
$$

En suivant les règles précédemment énoncées, on choisit comme pivot (-4), on obtient alors

$$\begin{array}{rccc} & y_1 & x_2 & 1 \\ x_1 = & \frac{-1}{4} & -\frac{1}{4} & \frac{1}{4} \\ f = & -\frac{3}{4} & \frac{1}{4} & \frac{3}{4} \end{array} \tag{15}$$

On remarque que cette opération nous fait passer de l'origine (point A) au deuxième sommet du triangle (point B). En effet en faisant $y_1 = x_2 = 0$,on trouve bien $x_1 = \frac{1}{4}$, ce qui correspond bien aux coordonnées du point B $(\frac{1}{4}, 0)$

En répétant l'opération, on voit maintenant que le seul pivot possible est $\left(-\frac{1}{4}\right)$ car l'élément f_i correspondant est le seul à être positif. On obtient alors

$$\begin{array}{rccc} & y_1 & x_1 & 1 \\ x_2 = & -1 & -4 & 1 \\ f = & -1 & -1 & 1 \end{array} \tag{16}$$

Ce qui correspond au point C $(0, 1)$. L'algorithme s'arrête alors, car f_1 et f_2 sont négatifs et on obtient la solution optimale :

$$x_1 = 0 \qquad x_2 = 1 \qquad f = 1 \tag{17}$$

Chapitre 6 Techniques d'optimisation

1 Introduction

On désigne sous le nom d'optimisation, la minimisation ou la maximisation d'une fonction de coût (ou de profit). Un problème de maximisation peut être ramené sans difficulté à un problème de minimisation. En effet, résoudre le problème :

$$x \in \Omega,\ x_{\max} = Arg \max_x f(x) \tag{1}$$

revient à résoudre le problème de minimisation suivant :

$$x_{\min} = Arg \min_x(-f(x)) \tag{2}$$

Dans toute la suite nous ne parlerons que de minimisation.
f est une fonction scalaire dépendant d'une variable scalaire ou vectorielle x.
On distingue différents problèmes d'optimisation résolus par différentes fonctions de la boite à outils.

1. problème sans contrainte scalaire
$$x \in \mathbb{R} \qquad \min_x f(x) \tag{3}$$
2. problème sans contrainte, vectoriel
$$x \in \mathbb{R}^n \qquad \min_x f(x) \tag{4}$$
3. problème avec contraintes, vectoriel
$$x \in (\Omega \subset \mathbb{R}^n), \qquad \min_x f(x) \qquad G(x) \leq 0 \tag{5}$$
4. problème minmax
$$\min_x \{\max F(x)\} \qquad G(x) \leq 0 \tag{6}$$
5. moindres carrés non linéaires
$$x \in \mathbb{R}^n \quad \min_x f^T(x) f(x) \tag{7}$$
6. la programmation linéaire
$$x \in (\Omega \subset \mathbb{R}^n) \qquad \min_x f^T x \qquad A.x \leq b \tag{8}$$
7. la programmation quadratique
$$x \in (\Omega \subset \mathbb{R}^n) \qquad \min_x \left(\frac{1}{2} X^T H X - c^T X\right) \qquad A.x \leq b \tag{9}$$

Nous ne considérerons ici que quelques méthodes traitant de l'optimisation non-linéaire sans contrainte, puis nous donnerons quelques éléments sur la façon d'intégrer les contraintes.

2 Méthode du Simplexe

Cette méthode est une extension de la méthode du simplexe dans le cas des problèmes de programmation linéaire. Elle est due à Nelder et Mead. On considère encore une fois un polyèdre à $(n+1)$ sommets (simplexe) et à chaque itération on essaye d'obtenir un nouveau simplexe permettant de se rapprocher du minimum de la fonction.

Remarque 17 *Dans le cas d'un problème plan* $(n = 2)$*, le simplex est un triangle. Dans le cas où* $n = 3$*, le simplexe est un tétraèdre dans l'espace à 3 dimensions.*

Soit s_i $(i = 0, ..., n)$ les coordonnées des sommets du simplex et $f(s_i)$ la valeur du critère en chacun de ses points. Supposons que les s_i sont ordonnés de telle sorte que

$$f(s_0) \leq f(s_1) \leq ... \leq f(s_n) \tag{10}$$

On peut donc dire que s_0 est le meilleur point et s_n est le pire point. Sachant que le but est de s'éloigner du point s_n, on prend comme direction de recherche la droite joignant s_n au centre c de la face du polyèdre opposée à s_n. Les étapes de l'algorithmes sont les suivantes :

1. On essaye le point p_0 symétrique à s_n par rapport à c. Ce point est donné par

$$p = c + (c - s_n) = 2c - s_n \tag{11}$$

2. Si $f(p_0) < f(s_0)$, on essaye d'aller plus loin dans cette direction en prenant par exemple

$$p_1 = c + \gamma(c - s_n) \qquad \gamma > 1 \tag{12}$$

   deux cas peuvent alors se présenter :

   - Si $f(p_1) \leq f(p_0)$, on remplace le point s_n par p_1 (expansion du simplexe)
   - Sinon on remplace s_n par p_0. Ce cas représente la reflexion du simplexe

3. Si $f(p_0) > f(s_0)$, on essaie les points p_2 et p_3 situés de part et d'autre de c à mi–distance entre c et p_0 et s_n et c. Ces points sont donc donnés par

$$p_2 = c + \frac{1}{2}(c - s_n) \tag{13}$$

$$p_3 = c - \frac{1}{2}(c - s_n) \tag{14}$$

   Deux cas peuvent encore se présenter :

   - Si $f(p_2) < f(s_{n-1})$ ou $f(p_3) < f(s_{n-1})$ alors on remplace s_n par p_2 ou p_3. On parle de contraction du simplexe.
   - Sinon, on sait que s_0 est proche du minimum. On rétrécit alors le simplexe en gardant s_0 et en remplaçant tous les autres points s_i par les milieux des segments (s_0, s_i).

4. On réordonne les différents points obtenus et on réitère les étapes précédentes. On s'arrête lorsque la dimension du simplexe obtenu est suffisamment petite.

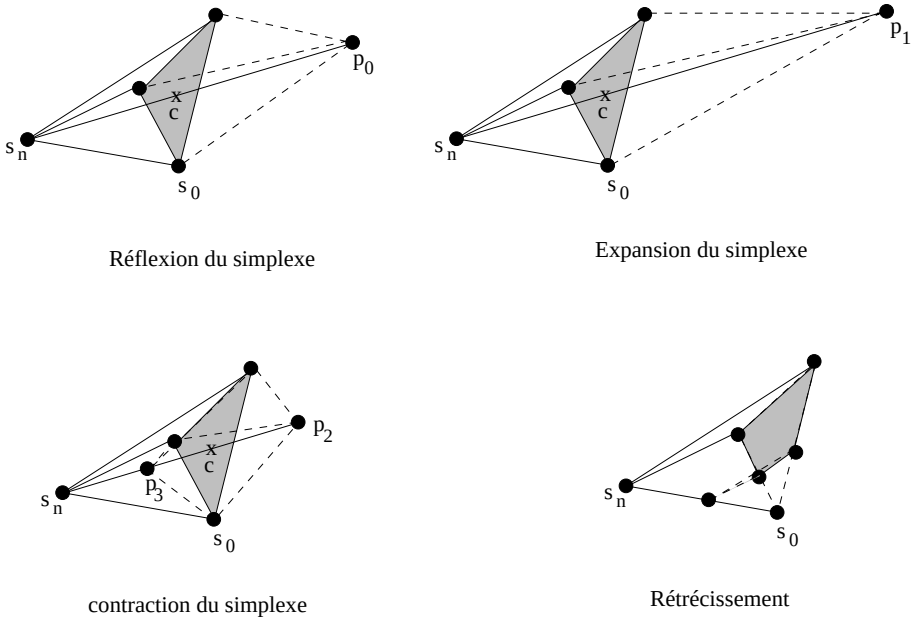

FIG. 1: Différentes évolutions possibles du simplexe

3 Méthodes utilisant une approximation locale de la fonction

3.1 Méthodes du gradient

Ces méthodes sont des méthodes dites du premier ordre car elle utilisent comme direction de recherche le gradient g de la fonction $f(x)$. Le gradient est défini par

$$g(x) = [g_1, g_2, ..., g_n]^T \tag{15}$$

avec

$$g_i = \frac{\partial f}{\partial x_i} \qquad (i = 1, ..., n) \tag{16}$$

L'algorithme prend alors la forme

$$x^{(k+1)} = x^{(k)} - \lambda g\left(x^{(k)}\right) \tag{17}$$

où λ est un nombre positif qui est soit constant soit variable au cours de l'algorithme. On peut choisir λ selon plusieurs règles

- λ constant faible (l'algorithme est lent)
- λ constant grand (l'algorithme est plus rapide, mais il peut être instable)
- λ variable, alors il est usuel de choisir λ par la méthode de la plus profonde descente. En effet on choisit à chaque fois λ de telle sorte que le gradient en $x^{(k+1)}$ soit orthogonal au gradient en $x^{(k)}$. C'est à dire

$$< g\left(x^{(k)}\right) g\left(x^{(k+1)}\right) >= 0 \tag{18}$$

Remarque 18 *De deux choses l'une : soit le gradient $g(x)$ est connu sous forme analytique, soit il devra être estimé par une méthode de différentiation numérique (cf chapitre 2)*

3.2 Méthodes de type Newton

3.2.1 Cas scalaire

Afin d'illustrer la méthode, nous allons d'abord considérer le cas monovariable ou x est un scalaire. Supposons que $x^{(k)}$ soit connu, alors au voisinage de ce point, le développement limité de $f(x)$ au $2^{ème}$ ordre donne

$$f(x) = f\left(x^{(k)}\right) + \left(x - x^{(k)}\right) f'\left(x^{(k)}\right) + \frac{1}{2}\left(x - x^{(k)}\right)^2 f''\left(x^{(k)}\right) + O\left(\left(x - x^{(k)}\right)^3\right) \tag{19}$$

Nous avons donc une approximation polynomiale de degré 2 de $f(x)$.

$$f_a(x) = f\left(x^{(k)}\right) + \left(x - x^{(k)}\right) f'\left(x^{(k)}\right) + \frac{1}{2}\left(x - x^{(k)}\right)^2 f''\left(x^{(k)}\right) \tag{20}$$

Sachant que l'extremum est obtenu pour une dérivée nulle, on choisit alors $x^{(k+1)}$ de telle sorte que

$$(f_a)'\left(x^{(k+1)}\right) = f'\left(x^{(k)}\right) + \left(x^{(k+1)} - x^{(k)}\right) f''\left(x^{(k)}\right) \tag{21}$$

ce qui donne l'algorithme de Newton

$$x^{(k+1)} = x^{(k)} - \frac{f'\left(x^{(k)}\right)}{f''\left(x^{(k)}\right)} \tag{22}$$

Remarque 19 *Par rapport à l'algrithme de Newton vu dans le chapitre sur la résolution des équations non-linéaires, cette formule est analogue sauf qu'au lieu dêtre appliquée à f, elle est appliquée à f'.*

Remarque 20 *On remarquera que dans la formule 22, la dérivée seconde de f devra être calculée à chaque pas de l'algorithme. Il existe une deuxième méthode de Newton qui s'affranchit de ce calcul. On écrit*

$$f_a(x) = f\left(x^{(0)}\right) + \left(x - x^{(0)}\right) f'\left(x^{(0)}\right) + \frac{1}{2}\left(x - x^{(0)}\right)^2 f''\left(x^{(0)}\right) \tag{23}$$

et donc

$$(f_a)'\left(x^{(k+1)}\right) = \left(x^{(k+1)} - x^{(0)}\right) f''\left(x^{(0)}\right) \tag{24}$$

d'où

$$x^{(k+1)} = x^{(k)} - \frac{f'\left(x^{(k)}\right)}{f''\left(x^{(0)}\right)} \tag{25}$$

3.2.2 Cas multivariable

La formule 20 dans le cas multivariable s'écrit

$$f_a(x) = f\left(x^{(k)}\right) + g\left(x^{(k)}\right)^T \left(x - x^{(k)}\right) + \frac{1}{2}\left(x - x^{(k)}\right)^T H\left(x^{(k)}\right)\left(x - x^{(k)}\right) \tag{26}$$

où H est le Hessien de la fonction défini par

$$H = [H_{ij}]_{(i,j=1,\ldots,n)} \quad H_{ij} = \frac{\partial f}{\partial x_i \partial x_j} \tag{27}$$

La formule 22 se généralise donc à

$$x^{(k+1)} = x^{(k)} - H\left(x^{(k)}\right)^{-1} g\left(x^{(k)}\right) \tag{28}$$

Remarque 21 *Comme dans la méthode du gradient, l'équation 28 prend souvent la forme*

$$x^{(k+1)} = x^{(k)} + \lambda^{(k)} H\left(x^{(k)}\right)^{-1} g\left(x^{(k)}\right) \tag{29}$$

où λ_k est, par exemple, calculé de telle sorte à minimiser le critère dans la direction $d^{(k)} = -H\left(x^{(k)}\right)^{-1} g\left(x^{(k)}\right)$

Le paramètre $\lambda^{(k)}$ est calculé par interpolation quadratique ou cubique de la fonction.

Remarque 22 *Il faut remarquer que l'équation 28 nécessite le calcul de l'inverse de la matrice H à chaque itération. Ceci peut être évité en remplaçant H^{-1} par une suite récurrente qui converge vers H^{-1}. Différentes méthodes existent, les plus connues sont la méthode de BFGS (Broyden, Fletcher, Goldfarb et Shanno) et la méthode DFP (Davidson, Fletcher et Powell)*

$$q^{(k)} = g\left(x^{(k+1)}\right) - g\left(x^{(k)}\right) \tag{30}$$

$$s^{(k)} = x^{(k+1)} - x^{(k)} \tag{31}$$

$$\left(H^{-1}\right)^{(k+1)} = \left(H^{-1}\right)^{(k)} + \frac{\left(s^{(k)}\right)^T s^{(k)}}{\left(s^{(k)}\right)^T q^{(k)}} - \frac{\left(H^{-1}\right)^{(k)} q^{(k)} \left(q^{(k)}\right)^T \left(H^{-1}\right)^{(k)}}{\left(q^{(k)}\right)^T \left(H^{-1}\right)^{(k)} q^{(k)}} \tag{32}$$

La matrice $\left(H^{-1}\right)$ peut être initialisée avec $\left(H^{-1}\right)^{(0)} = I$

Chapitre 7 Les moindres carrés linéaires

1 Introduction

La méthode des moindres carrées est très utilisée dans les sciences expérimentales et plus particulièrement dans les problèmes d'estimation et d'identification. Comme son nom l'indique, cette méthode consiste à minimiser la norme quadratique (norme euclidienne) d'une fonction appelée fonction d'erreur. Le problème à résoudre peut s'énoncer de la manière suivante :

Supposons donnés N points d'un relevé expérimental $(x_i, y_i), (i = 1, ..., N)$. x_i est par exemple l'instant de prélèvement d'une substance dans une réaction chimique et y_i est la concentration de cette substance.

Supposons maintenant que l'on recherche une relation linéaire entre x et y de la forme

$$y = ax + b \tag{1}$$

Il est clair que pour $N = 2$ les inconnues a et b sont déterminés de manière unique. Il est clair aussi que la droite ne peut passer par l'ensemble des points (figure 1). En fait ce qu'on recherche ici est la droite la plus probable, donc minimisant une certaine erreur.

Au point (x_i, y_i), la distance entre la droite et le point est

$$e_i = |(ax_i + b) - y_i| \tag{2}$$

On peut alors envisager de déterminer a et b en résolvant différents problèmes d'optimisation

- minimiser la plus grande valeur de e_i (problème de min-max)

$$\min_{a,b} \max_{0 \leq i \leq (n-1)} e_i \tag{3}$$

- minimiser la somme des erreurs (programmation linéaire)

$$\min_{a,b} \sum_{i=0}^{N-1} e_i \tag{4}$$

- minimiser la somme des carrés des erreurs (droite des moindres carrés)

$$\min_{a,b} \sum_{i=0}^{N-1} e_i^2 = \min_{a,b} e^T e \tag{5}$$

$$= \min_{a,b} \|e\|_2^2 \tag{6}$$

avec $e = [e_0, e_1, ..., e_{N-1}]^T$

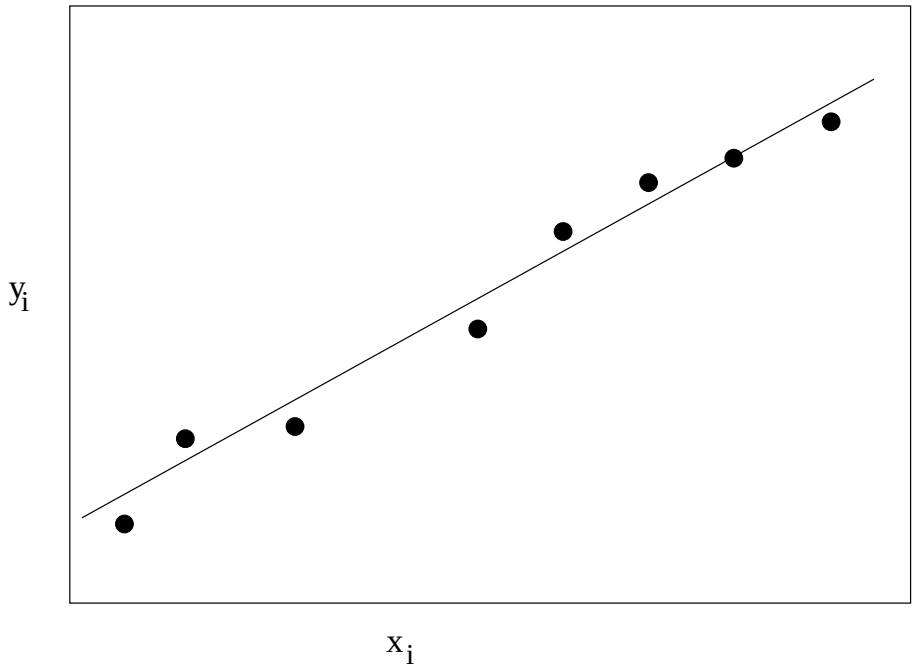

FIG. 1: Droite des moindres carrés

On peut se poser la question de l'intérêt de la norme 2. Il y en a deux : d'une part les calculs sont faciles et d'autre part dans le cas d'une distribution normale des erreurs, l'estimateur des moindres carrés est aussi l'estimateur du maximum de vraisemblance..

2 Méthode de calcul des paramètres

2.1 Calcul direct

Soit la fonction $\phi(a,b)$ la fonction à minimiser

$$\phi(a,b) = \sum_{i=0}^{N-1} e_i^2 \tag{7}$$

$$= \sum_{i=0}^{N-1} (ax_i + b - y_i)^2 \tag{8}$$

La condition nécessairement satisfaite au minimum est que les dérivées partielles de ϕ par rapport à a et b sont nulles

$$\frac{\partial \phi}{\partial a} = 2\sum_{i=0}^{N-1} (ax_i + b - y_i)\, x_i = 0 \tag{9}$$

$$\frac{\partial \phi}{\partial b} = 2\sum_{i=0}^{N-1} (ax_i + b - y_i) = 0 \tag{10}$$

Les deux conditions précédentes se mettent sous forme d'un système d'équations à deux inconnues

$$\begin{cases} \left(\sum_{i=0}^{N-1} x_i^2\right) a + \left(\sum_{i=0}^{N-1} x_i\right) b = \left(\sum_{i=0}^{N-1} x_i y_i\right) \\ \left(\sum_{i=0}^{N-1} x_i\right) a + Nb = \left(\sum_{i=0}^{N-1} y_i\right) \end{cases} \tag{11}$$

Les solutions du système sont

$$\left\{ \begin{array}{l} a = \frac{1}{D}\left[N\left(\sum_{i=0}^{N-1} x_i y_i\right) - \left(\sum_{i=0}^{N-1} x_i\right)\left(\sum_{i=0}^{N-1} y_i\right)\right] \\ b = \frac{1}{D}\left[\left(\sum_{i=0}^{N-1} x_i^2\right)\left(\sum_{i=0}^{N-1} y_i\right) - ()\left(\sum_{i=0}^{N-1} x_i y_i\right)\right] \end{array} \right. \tag{12}$$

avec

$$D = N\left(\sum_{i=0}^{N-1} x_i^2\right) - \left(\sum_{i=0}^{N-1} x_i\right)^2 \tag{13}$$

On vérifie que cet extremum correspond bien à un minimum en calculant les dérivées secondes qui sont positives

$$\frac{\partial^2 \phi}{\partial a^2} = 2\sum_{i=0}^{N-1} x_i^2 > 0 \tag{14}$$

$$\frac{\partial^2 \phi}{\partial b^2} = 2N > 0 \tag{15}$$

2.2 Calcul vectoriel

Il est possible d'effectuer le calcul précédent sous forme vectoriel. D'après 5,

$$\phi(a,b) = e^T e \tag{16}$$

de plus

$$e = Cz - d \tag{17}$$

avec

$$C = \left[\begin{array}{cc} x_0 & 1 \\ x_1 & 1 \\ . & 1 \\ . & 1 \\ x_{n-1} & 1 \end{array}\right], z = \left[\begin{array}{c} a \\ b \end{array}\right], d = \left[\begin{array}{c} y_0 \\ y_1 \\ . \\ . \\ y_{n-1} \end{array}\right] \tag{18}$$

d'où

$$\phi(a,b) = z^T C^T C z - 2(C^T d)^T z + d^T d \tag{19}$$

on en déduit donc les conditions du minimum

$$\frac{\partial \phi}{\partial z} = 2\left(C^T C\right) z - 2(C^T d) = 0 \tag{20}$$

$$\frac{\partial^2 \phi}{\partial z^2} = 2\left(C^T C\right) > 0 \tag{21}$$

L'équation 20 est appelée équation normale, nous la verrons plus en détail en traitement statistique du signal. La solution est donc obtenue sous la forme

$$z = \left(C^T C\right)^{-1}\left(C^T d\right) \tag{22}$$

Remarque 23 *L'équation 22 est très générale. Elle reste valable dans le cas de n paramètres* $z = [z_0, z_2, ..., z_{n-1}]^T$ $(n \leq N)$ *avec*

$$C : (N \times n) \qquad d : (N \times 1) \tag{23}$$

3 Extension de la méthode

En réalité la méthode des moindres carrés précédemment développée reste applicable du moment que la fonction d'approximation recherchée dépend de manière linéaire des inconnues. C'est le cas de toute combinaison linéaire de fonctions non-linéaires préalablement choisies.

Exemple 21 *Pour la fonction*

$$y = a \ln x + b \cos x + ce^x$$

les matrices du calcul vectoriel prennent la forme suivante

$$C = \begin{bmatrix} \ln x_0 & \cos x_0 & e^{x_0} \\ \ln x_1 & \cos x_1 & e^{x_1} \\ . & . & . \\ . & . & . \\ \ln x_{n-1} & \cos x_{n-1} & e^{x_{n-1}} \end{bmatrix}, z = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, d = \begin{bmatrix} y_0 \\ y_1 \\ . \\ . \\ y_{n-1} \end{bmatrix}$$

Chapitre 8 Les moindres carrés non-linéaires

1 Introduction

Le problème à résoudre dans ce cas est encore une fois une recherche de paramètres en minimisant une fonction d'erreur quadratique. La seule différence est que la dépendance de la fonction vis-à-vis des paramètres n'est plus forcément linéaire.

Exemple 22 *Les relevés du trafic routier à partir de capteurs (boucles magnétiques) laissent penser que la dépendance entre la vitesse moyenne v sur le tronçon et le taux d'occupation o de la boucle magnétique est de la forme*

$$v = v_f \exp\left(-\frac{1}{a}\left(\frac{o}{o_{cr}}\right)^a\right) \tag{1}$$

où v_f est la vitesse libre sur le tronçon (vitesse en trafic très fluide), o_{cr} est le taux d'occupation critique (point de basculement entre des conditions de trafic fluide et de trafic congestionné) et a est un paramètre généralement compris entre 1 et 2. Cette relation est appelée diagramme fondamental du tronçon (figure 1).

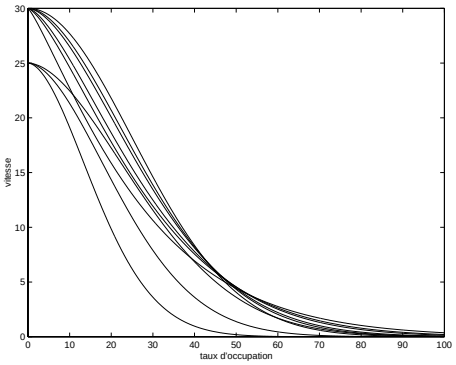

FIG. 1: Exemple de relation entre la vitesse et le taux d'occupation

Exemple 23 *La calibrage d'un diagramme fondamental correspond à trouver un jeu de paramètres (v_f, o_{cr}, a) à partir d'un ensemble de mesures (v_i, o_i) $(i = 0, 1, ..., N-1)$ en minimisant par exemple l'erreur quadratique moyenne*

$$e = \sum_{i=0}^{N-1}\left[v_f \exp\left(-\frac{1}{a}\left(\frac{o_i}{o_{cr}}\right)^a\right) - v_i\right]^2 \tag{2}$$

On remarquera que dans ce cas les observations concernent toujours la même fonction en des points différents. Il peut bien sûr s'agir de l'observation (mesure) de plusieurs fonctions faisant intervenir 1 ou plusieurs paramètres à déterminer.

2 Position du problème

Le problème des moindres carrés non-linéaires peut être posé de la manière suivante :

Étant données N observations y_i $(i = 0, 2, ..., N-1)$ et n $(n \leq N)$ paramètres inconnus $(z_0, z_2, ..., z_{n-1})$ intervenant dans des fonctions non-linéaires f_i, $(i = 0, 2, ..., N)$, on définit alors le problème des moindres carrés non-linéaires comme le problème de minimisation ci-dessous

$$z_{opt} = \arg\min_z e^T e \tag{3}$$

avec

$$z = [z_0, z_2, ..., z_{n-1}]^T \tag{4}$$

et

$$e(z) = \begin{bmatrix} f_0(z_0, z_2, ..., z_{n-1}) - y_0 \\ f_1(z_0, z_2, ..., z_{n-1}) - y_1 \\ : \\ f_{N-1}(z_0, z_2, ..., z_{n-1}) - y_{N-1} \end{bmatrix} \tag{5}$$

Remarque 24 *Comme nous l'avons précédemment signalé, les fonctions f_i ne sont pas forcément différentes.*

3 Résolution, méthode de Gauss-Newton

Cette méthode est basée sur une approximation linéaire locale du système non-linéaire. Le nom de Gauss-Newton vient du fait qu'on utilise le principe de Gauss pour la résolution des systèmes linéaires et que le problème coïncide avec la méthode de Newton de résolution des systèmes non-linéaires pour $n = N$, (autant d'équations que d'inconnues).

Notons d'abord que $e^T e$ s'écrit aussi :

$$\phi(z) = e^T e = \sum_{i=0}^{N-1} [f_i(z_0, z_2, ..., z_{n-1}) - y_i]^2 \tag{6}$$

Comme nous l'avons déjà vu dans le cas des moindres carrés linéaires, la condition nécessaire du minimum est l'annulation des dérivées partielles de l'erreur, c'est-à-dire :

$$\frac{\partial \phi(z)}{\partial z_j} = 2 \sum_{i=0}^{N-1} [f_i(z_0, z_2, ..., z_{n-1}) - y_i] \frac{\partial f_i(z_0, z_2, ..., z_{n-1})}{\partial z_j} = 0 \qquad (j = 0, 1, .., n-1) \tag{7}$$

Remarque 25 *L'équation 7 peut s'écrire sous forme matricielle en introduisant le gradient de ϕ ,$\bigtriangledown\phi$ et le jacobien de f , Φ_f :*

$$\bigtriangledown\phi = \begin{bmatrix} \frac{\partial \phi(z)}{\partial z_0} & \frac{\partial \phi(z)}{\partial z_1} & ... & \frac{\partial \phi(z)}{\partial z_{n-1}} \end{bmatrix}^T \tag{8}$$

$$\Phi_f(z) = \begin{bmatrix} \frac{\partial f_0(z)}{\partial z_0} & \frac{\partial f_0(z)}{\partial z_1} & ... & \frac{\partial f_0(z)}{\partial z_{n-1}} \\ \frac{\partial f_1(z)}{\partial z_0} & & ... & \frac{\partial f_1(z)}{\partial z_{n-1}} \\ : & : & & : \\ \frac{\partial f_N - 1(z)}{\partial z_0} & & ... & \frac{\partial f_{N-1}(z)}{\partial z_{n-1}} \end{bmatrix} \tag{9}$$

L'équation 7 s'écrit alors

$$\bigtriangledown\phi(z) = 2.\Phi_f^T(z).e(z) \tag{10}$$

On aboutit donc à la résolution d'un système d'équations qui n'est plus linéaire mais non-linéaire. Cette résolution étant difficile, on procède à la linéarisation du système. Supposons pour cela qu'une solution initiale approximative $z^{(0)}$ du problème soit connue :

$$z^{(0)} = \left[z_0^{(0)}, z_2^{(0)}, ..., z_{n-1}^{(0)}\right]^T \tag{11}$$

Considérons alors le point $z^{(1)}$ au voisinage de $z^{(0)}$ défini par

$$z^{(1)} = z^{(0)} + \xi^{(0)} \tag{12}$$

$$= \left[z_0^{(0)} + \xi_0^{(0)}, z_2^{(0)} + \xi_1^{(0)}, ..., z_{n-1}^{(0)} + \xi_{n-1}^{(0)}\right]^T \tag{13}$$

Le développement en série de f_i au premier ordre permet d'écrire

$$f_i\left(z^{(1)}\right) \approx f_i\left(z^{(0)}\right) + \sum_{j=0}^{n-1} \frac{\partial f_i(z^{(0)})}{\partial z_j} \xi_j^{(0)} \tag{14}$$

On peut donc écrire aussi que

$$\phi(z) \approx \sum_{i=0}^{N-1} \left[f_i\left(z^{(0)}\right) + \sum_{j=0}^{n-1} \frac{\partial f_i(z^{(0)})}{\partial z_j} \xi_j^{(0)} - y_i \right]^2 \tag{15}$$

On constate qu'on obtient alors un problème des moindres carrés linéaires de la forme de l'équation 22 ou la matrice C est remplacée par la matrice $C^{(0)}$ et le vecteur d est remplacé par le vecteur $d^{(0)}$ donnés ci-dessous

$$C^{(0)} = \Phi_f\left(z^{(0)}\right) = \begin{bmatrix} \frac{\partial f_0(z^{(0)})}{\partial z_0} & \frac{\partial f_0(z^{(0)})}{\partial z_1} & \cdots & \frac{\partial f_0(z^{(0)})}{\partial z_{n-1}} \\ \frac{\partial f_1(z^{(0)})}{\partial z_0} & & \cdots & \frac{\partial f_1(z^{(0)})}{\partial z_{n-1}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_N - 1(z^{(0)})}{\partial z_0} & & \cdots & \frac{\partial f_{N-1}(z^{(0)})}{\partial z_{n-1}} \end{bmatrix}, d^{(0)} = \begin{bmatrix} f_0\left(z^{(0)}\right) - y_0 \\ f_1\left(z^{(0)}\right) - y_1 \\ \vdots \\ f_{N-1}\left(z^{(0)}\right) - y_{N-1} \end{bmatrix} \tag{16}$$

L'élément de correction est donc donné par (cf équation 22)

$$\xi^{(0)} = \left(C^{(0)T} C^{(0)}\right)^{-1} (C^{(0)T} d^{(0)}) \tag{17}$$

On itère alors le processus avec le nouveau point

$$z^{(1)} = z^{(0)} + \xi^{(0)} \tag{18}$$

lequel, on espère, sera plus proche de la solution que $z^{(1)}$, et ainsi de suite.

Comme toute méthode de Newton, la convergence de la suite $z^{(k)}$ vers la solution n'est pas assurée. Elle dépend fortement du point initial choisi. Ce problème a déjà été soulevé dans le cas de la résolution des équations non-linéaires (chapitre 5). Il faut surveiller la décroissance du critère, c'est à dire être assuré que :

$$\phi(z^{(k)}) < \phi(z^{(k-1)}) \tag{19}$$

Le théorème suivant donne une condition suffisante pour que la propriété de décroissance soit satisfaite.

Théorème 14 *Le terme $\xi^{(k)}$ déterminé par l'équation*

$$\xi^{(k)} = \left(C^{(k)T} C^{(k)}\right)^{-1} (C^{(k)T} d^{(k)}) \tag{20}$$

fait décroître le critère si

$$\bigtriangledown \phi(z^{(k)}) \neq 0 \tag{21}$$

Chapitre 9 Intégration numérique

1 Introduction

L'objectif de l'intégration numérique est de calculer l'intégrale $I(a,b)$ d'une fonction $f(x)$ sur un certain intervalle $[a,b]$ (figure 1).

$$I(a,b) = \int_a^b f(x)\ \mathrm{d}x \tag{1}$$

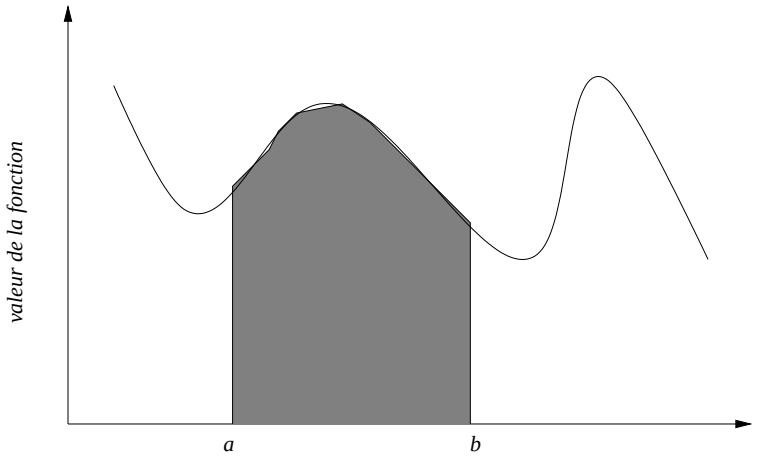

FIG. 1: L'intégrale d'une fonction sur $[a,b]$ équivaut à la surface pleine

L'expression analytique de $f(x)$ peut être connue dans certains cas comme elle peut être inconnue. On supposera ici que $f(x)$ est connue sur $(n+1)$ points dans l'intervalle $[a,b]$ (n sous-intervalles $[x_i, x_{i+1}]$)

$$a = x_0 < x_1 < x_2 < .... < x_n = b \tag{2}$$

2 Méthode des rectangles

Il existe plusieurs manières d'appliquer la méthode des rectangles. Considérons d'abord le cas où on prend sur chaque intervalle $[x_i, x_{i+1}]$ la plus faible valeur de la fonction. On aura alors

$$m_i = \inf\{f(x), x_i \leq x \leq x_{i+1}\} \tag{3}$$

et une estimation de l'intégrale est

$$L(f) = \sum_{i=0}^{n-1} m_i\,(x_{i+1} - x_i) \tag{4}$$

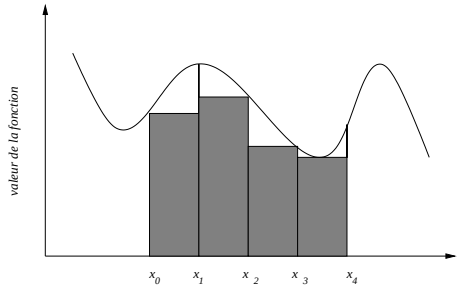

FIG. 2: $L(f)$ Intégrale de f en prenant les minima

Dans le cas où on prend le maximum de la fonction sur l'intervalle $[x_i, x_{i+1}]$,on obtient

$$M_i = \sup\{f(x), x_i \leq x \leq x_{i+1}\} \tag{5}$$

et

$$U(f) = \sum_{i=0}^{n-1} M_i \left(x_{i+1} - x_i\right) \tag{6}$$

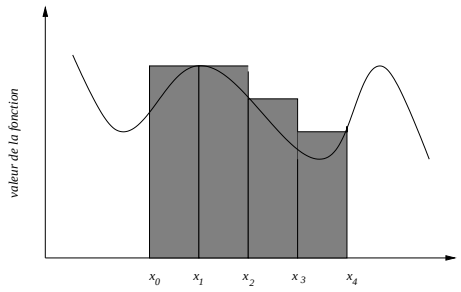

FIG. 3: $U(f)$ Intégrale de f en prenant les maxima

L'intégrale de la fonction est donc comprise entre ces deux limites

$$L(f) \leq \int_a^b f(x)dx \leq U(f) \tag{7}$$

Une troisième méthode pour obtenir une estimée de l'intégrale serait de prendre la valeur moyenne

$$M(f) = \frac{L(f) + U(f)}{2} \tag{8}$$

Exemple 24 $f(x) = x^2,\ [a,b] = [0,2],\ x_i = \{0, \frac{1}{4}, \frac{1}{2}, 1, \frac{5}{4}, \frac{3}{2}, 2\}$

On a

$$\int_0^2 f(x)dx = \left[\frac{x^3}{3}\right]_0^2 = \frac{8}{3} \tag{9}$$

et

$$L(f) = 0 + \left(\frac{1}{4}\right)^2 + \left(\frac{1}{2}\right)^2 + 1 + \left(\frac{5}{4}\right)^2 + \left(\frac{3}{2}\right)^2 = \frac{41}{8} \tag{10}$$

$$U(f) = \left(\frac{1}{4}\right)^2 + \left(\frac{1}{2}\right)^2 + 1 + \left(\frac{5}{4}\right)^2 + \left(\frac{3}{2}\right)^2 + 2^2 = \frac{73}{8} \tag{11}$$

$$M(f) = \frac{57}{4} \tag{12}$$

La différence est $M(f)$ avec $\int_0^2 f(x)$ est de $\frac{139}{12}$.Bien sûr cette différence décroît lorsque le nombre de points augmente (le pas d'intégration diminue). Ceci est illustré par le théorème suivant.

Théorème 15 (Integration au sens de Reimann) *Soit f une fonction continue sur l'intervalle $[a,b]$. Soit n le nombre de subdivisions de l'intervalle $[a,b]$ alors*

$$\lim_{n\to+\infty} L(f) = \int_a^b f(x)dx = \lim_{n\to+\infty} U(f) \tag{13}$$

Exemple 25 *D'après le théorème précédent, il est nécessaire que la fonction soit continue. Un exemple est fourni par la fonction de Dirichelet définie par*

$$d(x) = \begin{cases} 0, & x \quad rationnel \\ 1, & x \quad irrationnel \end{cases} \tag{14}$$

On a pour cette fonction $L = 0$ et $U = b - a$.

La question qu'on peut maintenant se poser est : si on suppose les points régulièrement espacés d'un pas $h = \frac{b-a}{n}$, quel est le nombre n de points nécessaires pour atteindre une précision donnée ε ?

La valeur de l'intégrale étant encadrée par $L(f)$ et $U(f)$, on peut prendre pour ε

$$\varepsilon = \frac{1}{2}(U - L) \tag{15}$$

Exemple 26 *Calculer n pour que l'erreur sur $\int_0^\pi e^{\cos x}dx$ soit inférieure à 0.5×10^{-3}*

La fonction $e^{\cos x}$ étant décroissante sur $[0,\pi]$,

$$L(f) = h\sum_{i=0}^{n-1} f(x_{i+1}) \tag{16}$$

$$U(f) = h\sum_{i=0}^{n-1} f(x_i) \tag{17}$$

avec $h = \frac{\pi}{n}$

Donc

$$\varepsilon = \frac{1}{2}(U - L) = \frac{1}{2}h\left(\sum_{i=0}^{n-1} f(x_i) - \sum_{i=0}^{n-1} f(x_{i+1})\right) \tag{18}$$

$$= \frac{1}{2}\frac{\pi}{n}\left(e^1 - e^{-1}\right) < 0.5 \times 10^{-3} \tag{19}$$

On en déduit

$$n \geq 7385 \tag{20}$$

On voit donc que n est élevé.

3 Méthode des trapèzes

La méthode des trapèzes est basée sur le fait que sur chaque sous intervalle $[x_i, x_{i+1}]$, on prend le rectangle ayant pour ordonnée y_i, valeur médiane entre $f(x_i)$ et $f(x_{i+1})$, c'est-à-dire :

$$y_i = \frac{1}{2}\left[f(x_i) + f(x_{i+1})\right] \tag{21}$$

L'intégrale calculée par cette méthode s'exprime donc sous la forme

$$T(f) = \frac{1}{2}\sum_{i=0}^{n-1}(x_{i+1} - x_i)\left[f(x_i) + f(x_{i+1})\right] \tag{22}$$

Remarque 26 *Si la fonction f est monotone, alors*

$$T(f) = \frac{L(f) + U(f)}{2} \tag{23}$$

Si on suppose que les points sont régulièrement espacés avec un pas $h = \frac{b-a}{n}$, alors la formule 22 s'écrit aussi

$$T(f) = h \left\{ \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2} \left[f(x_0) + f(x_n) \right] \right\} \tag{24}$$

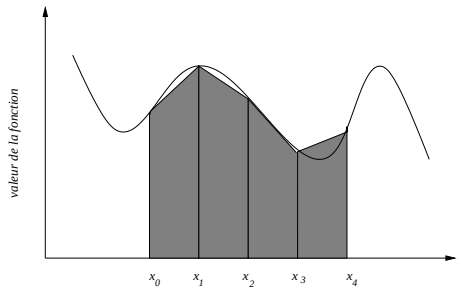

FIG. 4: $T(f)$ Intégrale de f par la méthode des trapèzes

On montre que la différence entre $T(f)$ et $\int_a^b f(x)dx$ est un infiniment petit d'ordre 2 en h. Ce résultat est donné par le théorème suivant

Théorème 16 *Soit f une fonction C^2 sur l'intervalle $[a, b]$. Il existe alors ξ dans l'intervalle $[a, b]$ tel que*

$$\int_a^b f(x)dx - T(f) = \frac{-1}{12}(b-a)h^2 f''(\xi) \tag{25}$$

$$= O(h^2) \tag{26}$$

La méthode des trapèzes permet donc une meilleure précision que la méthode de rectangles ($O(h)$ pour la méthode des rectangles). De plus à précision fixée, la méthode des trapèzes nécessite moins de points comme on le voit dans l'exemple suivant qui reprend la fonction $\int_0^\pi e^{\cos x} dx$

Exemple 27

$$f(x) = e^{\cos x} \qquad f'(x) = -e^{\cos x} \sin x \qquad f''(x) = e^{\cos x} \sin^2 x - e^{\cos x} \cos x$$

On remarque que sur $[0, \pi]$

$$\left| f''(x) \right| < e$$

On aura donc

$$\left| \frac{-1}{12}(b-a)h^2 f''(\xi) \right| < \frac{1}{12}\pi \left(\frac{\pi}{n} \right)^2 e < 0.5 \times 10^{-3}$$

On en déduit

$$n \geq 119$$

ce qui est bien plus faible que 7385.

La méthode des trapèzes peut être interprétée comme une interpolation linéaire de la fonction sur chaque intervalle $[x_i, x_{i+1}]$. En effet le polynôme d'interpolation $p_i(x)$ est donné par

$$p_i(x) = \frac{x - x_{i+1}}{x_i - x_{i+1}} f(x_i) + \frac{x - x_i}{x_{i+1} - x_i} f(x_{i+1}) \tag{27}$$

et le calcul l'intgrale de $p_i(x)$ sur $[x_i, x_{i+1}]$ donne

$$\int_{x_i}^{x_{i+1}} p_i(x)\mathrm{d}x = (x_{i+1} - x_i)\,\frac{[f(x_i) + f(x_{i+1})]}{2} \tag{28}$$

qui n'est rien d'autre que la formule précédemment établie.

Dès lors, on peut bien sûr considérer des interpolations d'ordre plus élevé faisant intervenir plus de points. C'est le cas notamment de la méthode de Simpson qui travaille sur 3 points.

4 Méthode de Simpson

Comme nous l'avons dit précédemment, la méthode de Simpson utilise une interpolation sur 3 points x_i, $\frac{x_i+x_{i+1}}{2}$, x_{i+1}. La formule de Simpson est donnée par

$$\int_{x_i}^{x_{i+1}} p_i(x)\mathrm{d}x = \frac{x_{i+1} - x_i}{6}\left[f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1})\right]$$

La méthode de Simpson fait mieux que la méthode des trapèzes. Elle pondère davantage le point milieux $\frac{x_i+x_{i+1}}{2}$. Ceci est confirmé par le théorème ci-dessous.

Théorème 17 *Soit f une fonction C^4 sur l'intervalle $[a, b]$ subdivisé en n sous intervalles de longueurs $h = \frac{b-a}{n}$, avec n pair. Alors, il existe ξ dans l'intervalle $[a, b]$ tel que*

$$\int_a^b f(x)dx = \frac{h}{3}\left\{[f(a) + f(b)] + 4\sum_{i=1}^{n/2} f(a + (2i-1)h) + 2\sum_{i=1}^{n/2-1} f(a + 2ih)\right\} - \frac{b-a}{180}h^4 f^{(4)}(\xi)$$

On constate donc qu'on obtient un infiniment petit en h d'ordre 4.

5 Intégration par interpolation

Dans ce cas, on remplace simplement la fonction $f(x)$ par le polynôme d'interpolation $P(x)$ sur les $(n + 1)$ points (cf polynôme de Lagrange)

$$P(x) = \sum_{i=0}^{n} f(x_i) l_i(x) \tag{29}$$

On peut alors espérer que si $P(x)$ est peu différent de $f(x)$, il en sera de même de leurs intégrales. Cette méthode assure évidement que l'erreur est nulle si $f(x)$ est une fonction polynomiale de degré n, et ce quelque soit le choix des x_i.

L'intégrale du polynôme $P(x)$ peut s'écrire en utilisant l'équation 29 sous la forme :

$$\int_a^b P(x)\mathrm{d}x = \sum_{i=0}^{n} f(x_i) \int_a^b l_i(x)\mathrm{d}x \tag{30}$$

$$= \sum_{i=0}^{n} f(x_i) I_{l_i} \tag{31}$$

On remarquera que les quantités I_{l_i} ne dépendent pas de la fonction. Elles dépendent seulement du choix des points x_i.

Partant de la remarque que les différentes méthodes affectent des pondérations différentes à chaque point x_i, nous avons donc au total $2(n+1)$ degrés de liberté : $(n+1)$ dans le choix des x_i et $(n+1)$ dans le choix des pondérations. On pourrait donc pouvoir assurer l'annulation de l'erreur pour tous les polynômes de degré $(2n+1)$.

Supposons qu'il existe un polynôme de degré $(n+1)$ satisfaisant la propriété :

$$\int_a^b x^k q(x)\mathrm{d}x = 0 \qquad (k = 0, 1, ..., n) \tag{32}$$

ce qui revient à dire que le polynôme $q(x)$ est orthogonal à tous les polynôme de degré inférieur ou égal à n. Supposons de plus que $q(x)$ est choisi de telle sorte que :

$$q(x_i) = 0 \qquad (i = 0, 1, ..., n) \tag{33}$$

Supposons maintenant que la fonction à intégrer soit un polynôme de degré $(2n+1)$. La division euclidienne nous assure que

$$\exists p(x), r(x)/ \quad \deg(p(x)) = \deg(r(x)) = n \qquad f(x) = p(x)q(x) + r(x) \tag{34}$$

On peur donc écrire :

$$\int_a^b f(x)\mathrm{d}x = \int_a^b p(x)q(x)\mathrm{d}x + \int_a^b r(x)\mathrm{d}(x) \tag{35}$$

$$= \int_a^b r(x)\mathrm{d}x \tag{36}$$

$r(x)$ étant un polynôme de degré n, on peut donc utiliser l'équation 31. On obtient alors

$$\int_a^b f(x)\mathrm{d}x = \sum_{i=0}^{n} r(x_i) I_{l_i} \tag{37}$$

$$= \sum_{i=0}^{n} \left[f(x_i) - p(x_i)q(x_i)\right] I_{l_i} \tag{38}$$

x_i étant racine de $q(x)$, on obtient finalement

$$\int_a^b f(x)\mathrm{d}x = \sum_{i=0}^{n} f(x_i) I_{l_i} \tag{39}$$

Ce qui veut donc dire que l'erreur commise est nulle pour les fonctions polynomiales de degré $(2n+1)$. Il reste donc à construire les polynômes $q(x)$. Ceci est réalisé par les polynômes de Legendre et la méthode s'appelle méthode de Gauss-Legendre.

Définition 3 *On appelle polynôme de Legendre de degré k, le polynôme $g(x)$ de degré k défini par*

$$g_k(x) = \frac{d^k}{dx^k}(x^2 - 1)^k \qquad k \in N \tag{40}$$

Les premiers éléments de cette suite sont

$$g_0(x) = 1 \qquad g_1(x) = 2x \qquad g_2(x) = 12x^2 - 4 \tag{41}$$

Ces polynômes sont orthogonaux, c'est à dire que le produit scalaire de $g_k(x)$ avec $g_i(x)$ est nul pour i différent de k.

$$\int_{-1}^{1} g_k(x) g_i(x)\mathrm{d}x = 0 \tag{42}$$

Théorème 18 *Pour tout $k \geq 1$, le polynôme $g_k(x)$ admet n racines distinctes sur l'intervalle ouvert* $]-1, 1[$.

Établissons maintenant la formule de Gauss Legendre pour une intégration sur l'intervalle $[-1, 1]$. Pour cela considérons $g_{n+1}(x)$, le $(n+1)^{ème}$ polynôme de Legendre. On sait qu'il admet $(n+1)$ racines distinctes ξ_i $(i = 0, 1, ..., n)$. D'après l'équation 39, on peut écrire :

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx \sum_{i=0}^{n} f(\xi_i) I_{l_i} \tag{43}$$

On démontre que les I_{l_i} sont simplement solution du système d'équations linéaires

$$\begin{bmatrix} g_0(\xi_0) & g_0(\xi_0) & .... & g_0(\xi_n) \\ g_1(\xi_0) & g_1(\xi_1) & ... & g_1(\xi_n) \\ : & : & : & : \\ g_n(\xi_0) & g_{n-1}(\xi_1) & ... & g_n(\xi_n) \end{bmatrix} \begin{bmatrix} I_{l_0} \\ I_{l_1} \\ : \\ I_{l_n} \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ : \\ 0 \end{bmatrix} \tag{44}$$

Exemple 28 *Supposons que nous ayons 2 points ($n = 1$). Les racines du polynôme $g_2(t)$ sont :*

$$\xi_0 = -\tfrac{\sqrt{3}}{3} \qquad \xi_1 = \tfrac{\sqrt{3}}{3} \tag{45}$$

Le système 44 s'écrit dans ce cas

$$\begin{bmatrix} 1 & 1 \\ -2\frac{\sqrt{3}}{3} & 2\frac{\sqrt{3}}{3} \end{bmatrix} \begin{bmatrix} I_{l_0} \\ I_{l_1} \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \tag{46}$$

dont la résolution donne

$$I_{l_0} = 1 \qquad I_{l_1} = 1 \tag{47}$$

On a donc la formule

$$\int_{-1}^{1} f(x)dx \approx f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right) \tag{48}$$

On vérifie maintenant que cette formule est exacte pour les polynômes de degré $(2n+1)$ c'est-à-dire 3. *Pour cela, on prend*

$$p(t) = ax^3 + bx^2 + cx + d \tag{49}$$

Son intégrale sur $[-1, 1]$ obtenue par intégration directe vaut

$$\int_{-1}^{1} p(x)dx = \frac{2}{3}b + 2d \tag{50}$$

Le calcul par la méthode de Gauss-Legendre (équation 48) donne le même résultat :

$$\int_{-1}^{1} p(x)dx \approx p\left(-\frac{\sqrt{3}}{3}\right) + p\left(\frac{\sqrt{3}}{3}\right) \tag{51}$$

$$= -\frac{\sqrt{3}}{9}a + \frac{1}{3}b - \frac{\sqrt{3}}{3}c + d + \frac{\sqrt{3}}{9}a + \frac{1}{3}b + \frac{\sqrt{3}}{3}c + d$$

$$= \frac{2}{3}b + 2d \tag{52}$$

Chapitre 10 Résolution des équations différentielles

1 Introduction

On appelle équations différentielles ordinaires, une équation ou un système d'équations différentielles dont les fonctions et leurs dérivées successives ne dépendent que d'une variable, le temps par exemple. On oppose le terme ordinaire à équations différentielles aux dérivées partielles.

On appelle ordre de l'équation différentielle le plus fort degré de dérivation apparaissant dans l'équation.

Une équation différentielle est dite linéaire si on peut l'écrire comme une combinaison linéaire (à coefficients constants) de dérivées successives des inconnues.

Une équation différentielle est dite homogène si tous les termes font intervenir des dérivées des inconnues

ED	ordinaire	linéaire	homogène
$\frac{\mathrm{d}x}{\mathrm{d}t} - x = e^t$	oui	oui	non
$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + 9x = 0$	oui	oui	oui
$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + 9\frac{\mathrm{d}x}{\mathrm{d}t}x = 0$	oui	non	oui

Dans certains cas, il est possible de trouver des solutions analytiques aux équations différentielles comme le montrent les exemples ci-dessous :

ED	solution
$\frac{\mathrm{d}x}{\mathrm{d}t} - x = e^t$	$x(t) = te^t + ce^t$
$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + 9x = 0$	$x(t) = c_1 \sin 3t + c_2 \cos 3t$
$\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{1}{2x} = 0$	$x(t) = \sqrt{c - t}$

Les constantes sont alors déterminées en considérant les conditions initiales et/ou les conditions finales (conditions aux limites).

Malheureusement dans la majorité des cas, la solution analytique est impossible à obtenir et, même lorsqu'on peut la calculer, on évite parfois de le faire à cause des temps de calculs gigantesques. Que peuvent alors apporter les méthodes numériques qui nous donneront les valeurs de la solution $x(t)$ à des instants discrets t_i? Et quelle est la précision de la solution obtenue ?

Avant d'aborder l'exposé des méthodes, on peut noter que toute équation différentielle ordinaire peut se mettre sous la forme d'une équation différentielle vectorielle ordinaire du premier ordre :

$$\frac{\mathrm{d}X}{\mathrm{d}t} = f(t, X) \tag{1}$$

Il suffit de rassembler dans le vecteur X les dérivées successives de $x(t)$.

Exemple 29 – *l'équation différentielle linéaire du $2^{ème}$ ordre* $\ddot{x} + 9x = 0$ *peut s'écrire sous la forme*

$$\frac{dX}{dt} = \begin{bmatrix} 0 & 1 \\ -9 & 0 \end{bmatrix} X$$

avec $X^T = \begin{bmatrix} x & \frac{dx}{dt} \end{bmatrix}$

– *l'équation différentielle non-linéaire*

$$\frac{d^3x}{dt^3} + 10te^{x^2(t)} - \left(\frac{dx}{dt}\right)^2(t) + \ln x(t) = 20$$

se met sous la forme

$$\frac{dX}{dt} = \begin{bmatrix} x_2 \\ x_3 \\ 20 - 10te^{x_1^2(t)} + x_2^2(t) - \ln x_1(t) \end{bmatrix} X$$

avec $X^T = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} = \begin{bmatrix} x & \frac{dx}{dt} & \frac{d^2x}{dt^2} \end{bmatrix}^T$

Il faut noter de plus que les problèmes de physique font intervenir généralement des conditions aux limites sur la fonction et ses dérivées. Ici nous nous intéresserons qu'aux méthodes traitant les problèmes définis avec conditions initiales. Nous donnerons par la suite les modifications à apporter pour traiter ceux incluant des conditions aux limites. Nous nous intéresserons donc à la résolution de l'équation différentielle vectorielle

$$\frac{\mathrm{d}X}{\mathrm{d}t} = f(t, X) \qquad X(t_0) = X_0 \tag{2}$$

2 Équations différentielles avec conditions initiales

2.1 Méthode d'Euler

Considérons l'équation différentielle précédente (2) sur l'intervalle $[t_0, t_1]$. On subdivise cet intervalle en N sous-intervalles de même longueur. On définit alors le pas h par

$$h = \frac{t_1 - t_0}{N} \tag{3}$$

Le développement en série de Taylor à l'ordre 2 de $X(t+h)$ donne

$$\begin{aligned} X(t+h) &= X(t) + h\dot{X}(t) + O(h^2) \\ &\approx X(t) + hf(t, X(t)) \end{aligned} \tag{4}$$

En itérant, on obtient la solution $X(t_i)$ aux points $(t_i = t_0 + h \times i)$ par la formule suivante.

$$y(t_0) = X_0 \tag{5}$$

$$X(t_{i+1}) = X(t_i) + hf(t_i, X(t_i)) \tag{6}$$

La méthode d'Euler présente l'avantage de ne nécessiter que le calcul de la fonction f et ce une fois par pas d'intégration. On remarquera que la précision est de l'ordre $O(h^2)$ pour ce premier pas, ce qui est bien ; par contre les erreurs sont cumulées et la solution obtenue s'écarte au fur et à mesure de la solution exacte (figure 1). En t_1, la précision n'est plus que du 1^{er} ordre $\left(n \times O(h^2) = O(h)\right)$.

Exemple 30 *Considérons l'équation différentielle :*

$$\frac{dx}{dt} + \frac{1}{2x} = 0 \qquad x(0) = 10$$

Elle admet comme solution analytique

$$x(t) = \sqrt{100 - t}$$

Cette solution apparaît en trait plein sur la figure 1. Les solutions obtenues par la méthode d'Euler apparaissent sur la même figure (signe o) pour $(h = 10)$ *et (signe *) pour* $(h = 5)$.

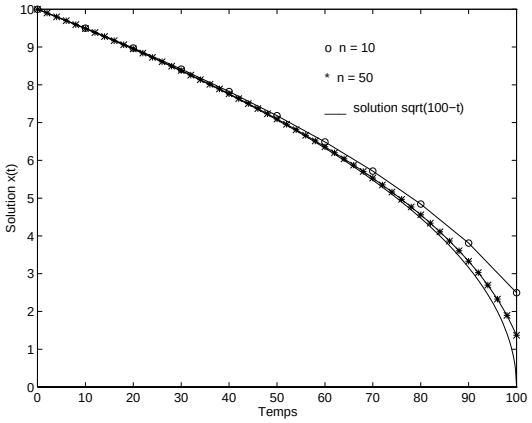

FIG. 1: Méthode d'Euler, influence du nombre de points

2.2 Méthode de Taylor d'ordre plus élevé

Ces méthodes sont simplement basées sur un développement en série de Taylor d'un ordre plus élevé que pour la méthode d'Euler. A titre d'exemple un développement à l'ordre 4 donne

$$X(t+h) = X(t) + hX^{(1)}(t) + \frac{h^2}{2!}X^{(2)}(t) + \frac{h^3}{3!}X^{(3)}(t) + \frac{h^4}{4!}X^{(4)}(t) + O(h^5) \tag{7}$$

$X^{(1)}(t)$ est donné par l'équation différentielle alors que les termes $X^{(2)}(t)$, $X^{(3)}(t)$ et $X^{(4)}(t)$ sont obtenus par dérivations successives de f.

Les figures 2 et 3 montrent la différence entre la solution obtenue par la méthode d'Euler (signe o) et la solution obtenue par un développement à l'ordre 3 (signe *) pour l'équation différentielle

$$\frac{\mathrm{d}x}{\mathrm{d}t} = x + e^t \qquad x(0) = 2, h = 0.25$$

dont la solution exacte est

$$x(t) = te^t + 2e^t$$

Pour $h = 5$, la méthode de Taylor est plus précise que la méthode d'Euler mais l'écart avec la solution exacte n'est pas nul. Par contre pour $h = 0.5$, on remarque que cette erreur est quasiment nulle aux points d'intégration.

Bien que plus précises que la méthode d'Euler $(O(h^4))$, ces méthodes nécessitent le calcul des dérivées de f ce qui peut être coûteux en temps de calcul voire impossible à obtenir. La méthode de Runge-Kutta permet d'éviter le calcul de ces dérivées.

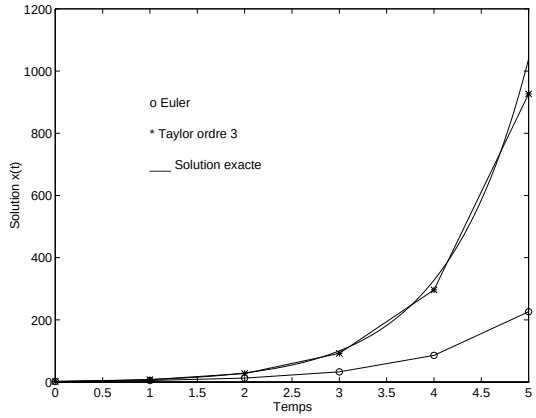

FIG. 2: Intégration par les méthodes d'Euler et de Taylor, $h = 1$

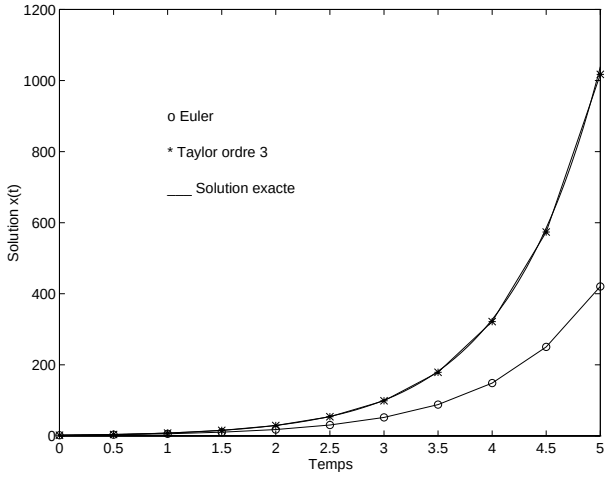

FIG. 3: Intégration par les méthodes d'Euler et de Taylor, $h = 0.5$

2.3 Méthode de Runge-Kutta

Il existe plusieurs variantes de cette méthode (ordre 3, 4, 5,...). Nous n'en étudierons qu'une seule : la méthode de Runge-Kutta d'ordre 4 qui calcule la valeur de la fonction en quatre points intermédiaires de la manière suivante :

$$k_1 = h.f(t_i, X(t_i)) \tag{8}$$

$$k_2 = h.f\left(t_i + \frac{h}{2}, X(t_i) + \frac{k_1}{2}\right) \tag{9}$$

$$k_3 = h.f\left(t_i + \frac{h}{2}, X(t_i) + \frac{k_2}{2}\right) \tag{10}$$

$$k_4 = h.f\left(t_i + h, X(t_i) + k_3\right) \tag{11}$$

$$X(t_{i+1}) = X(t_i) + \frac{(k_1 + 2.k_2 + 2.k_3 + k_4)}{6} \tag{12}$$

Il est à noter qu'une méthode de Runge-Kutta d'ordre m quelconque égale en précision une méthode de Taylor du même ordre.

Remarque 27 *Les formules pour la méthode de Rung-Kutta d'ordre 2 sont données par*

$$k_1 = h.f(t_i, X(t_i)) \tag{13}$$

$$k_2 = h.f\left(t_i + h, X(t_i) + k_1\right) \tag{14}$$

et

$$X(t_{i+1}) = X(t_i) + \frac{(k_1 + k_2)}{2} \tag{15}$$

Les méthodes de Runge-Kutta sont les méthodes préférées des ingénieurs, mais on ne peut pas dire pour autant quelles sont les meilleures. En effet, à chaque problème, une méthode optimale de résolution. Les méthodes de Runge-Kutta échouent notamment lorque le système présente à la fois des dynamiques rapides et lentes.

Exemple 31 *L'équation différentielle*

$$\begin{aligned} x(0) &= 0.1 \\ \frac{dx}{dt} &= \frac{x}{2}\left(1 - \frac{x}{2}\right) \end{aligned}$$

On vérifie facilement que l'équation 16 est solution de l'équation différentielle

$$x(t) = \frac{0.2e^{\cdot\frac{t}{2}}}{2 + 0.1(e^{\frac{t}{2}} - 1)} \tag{16}$$

1. La figure 4 montre les différences entre la solution obtenue respectivement par les méthodes d'Euler, de Taylor d'ordre 3 et de Runge-Kutta d'ordre 4.

Il est évident qu'on ne peut pas directement comparer la méthode de Runge-Kutta aux précédentes puisqu'elles n'utilisent pas le même nombre de points.

2.4 Les autres méthodes

Ces méthodes sont basées sur l'interpolation polynomiale d'ordre 1, 2, 3 ou 4. Ces méthodes sont aussi appelées méthodes à pas multiples par opposition aux méthodes précédentes qui sont des méthodes à un pas.

Les méthodes d'Adams-Bashforth sont bien adaptées aux systèmes à faibles non-linéarités mais dont les paramètres ne varient pas beaucoup dans le temps.

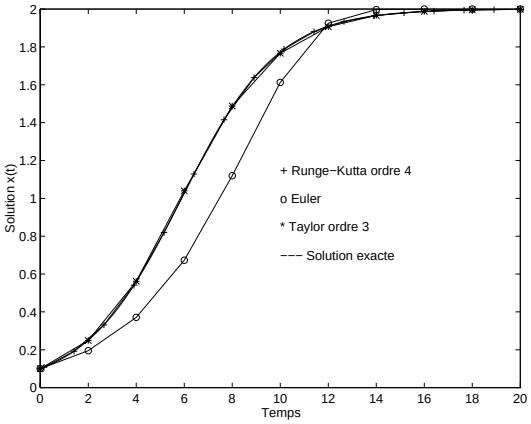

FIG. 4: Intégration par les méthodes d'Euler, de Taylor et de Runge-Kutta

2.4.1 Adams-Bashforth à deux pas

$$X(t_{i+1}) = X(t_i) + h\frac{[3f(t_i,X(t_i))-f(t_{i-1},X(t_{i-1}))]}{2} \qquad n \geq 1 \tag{17}$$

L'erreur est du type $O(h^2)$

2.4.2 Adams-Bashforth à trois pas

$$X(t_{i+1}) = X(t_i) + h\frac{[23f(t_i,X(t_i))-16f(t_{i-1},X(t_{i-1}))+5f(t_{i-2},X(t_{i-2}))]}{12} \qquad n \geq 2 \tag{18}$$

L'erreur est du type $O(h^3)$

2.4.3 Adams-Bashforth à quatre pas

$$X(t_{i+1}) = X(t_i) + h\frac{\left[\begin{array}{c} 55f(t_i,X(t_i)) - 59f(t_{i-1},X(t_{i-1}))+ \\ 37f(t_{i-2},X(t_{i-2})) - 9f(t_{i-3},X(t_{i-3})) \end{array}\right]}{24} \qquad n \geq 3 \tag{19}$$

L'erreur est du type $O(h^4)$. Examinons comment cette formule et d'ailleurs les précédentes aussi, sont obtenues. On considère pour cela quatre points d'abscisses $(t_{i-3}, t_{i-2}, t_{i-1}, t_i)$. Le polynôme d'interpolation de Lagrange de degré 3 pour f est donné par

$$P(x) = \sum_{k=0}^{3} f(t_{i-k}, X(t_{i-k}))l_{i,k}(t) \tag{20}$$

avec

$$l_{i,0}(t) = \frac{t-t_{i-1}}{t_i-t_{i-1}}\frac{t-t_{i-2}}{t_i-t_{i-2}}\frac{t-t_{i-3}}{t_i-t_{i-3}} \qquad l_{i,1}(t) = \frac{t-t_i}{t_{i-1}-t_i}\frac{t-t_{i-2}}{t_{i-1}-t_{i-2}}\frac{t-t_{i-3}}{t_{i-1}-t_{i-3}} \tag{21}$$

$$l_{i,2}(t) = \frac{t-t_i}{t_{i-2}-t_i}\frac{t-t_{i-1}}{t_{i-2}-t_{i-1}}\frac{t-t_{i-3}}{t_{i-2}-t_{i-3}} \qquad l_{i,3}(t) = \frac{t-t_i}{t_{i-3}-t_i}\frac{t-t_{i-1}}{t_{i-3}-t_{i-1}}\frac{t-t_{i-2}}{t_{i-3}-t_{i-2}} \tag{22}$$

On peut donc remplacer l'équation différentielle par l'intégrale de $P(x)$:

$$X(t_{i+1}) = X(t_i) + \int_{t_i}^{t_{i+1}} P(x)\mathrm{d}t \tag{23}$$

$$= X(t_i) + \sum_{k=0}^{3} f(t_{i-k}, X(t_{i-k})) \int_{t_i}^{t_{i+1}} l_{i,k}(t)\mathrm{d}t \tag{24}$$

On remarque que les termes $\int_{t_i}^{t_{i+1}} l_{i,k}(t)\mathrm{d}t$ sont indépendants de la fonction f. Ils peuvent donc être calculés une fois pour toutes.

En supposant un pas de discrétisation constant $h = t_i - t_{i-1}$, on obtient

$$\int_{t_i}^{t_{i+1}} l_{i,0}(t)\mathrm{d}t = \frac{55}{24}h \qquad \int_{t_i}^{t_{i+1}} l_{i,1}(t)\mathrm{d}t = -\frac{59}{24}h \tag{25}$$

$$\int_{t_i}^{t_{i+1}} l_{i,2}(t)\mathrm{d}t = \frac{37}{24}h \qquad \int_{t_i}^{t_{i+1}} l_{i,3}(t)\mathrm{d}t = -\frac{9}{24}h \tag{26}$$

2.5 Méthodes d'Adams-Moulton

Ces méthodes sont implicites car le calcul de $X(t_{i+1})$ nécessite la connaissance de $f(t_{i+1}, X(t_{i+1}))$. Elle utilise un polynôme d'interpolation de degré 4 aux points $(t_{i-3}, t_{i-2}, t_{i-1}, t_i, t_{i+1})$. La formule est

$$X(t_{i+1}) = X(t_i) + h\frac{\left[\begin{array}{c} 251f(t_{i+1}, X(t_{i+1})) + 646f(t_i, X(t_i)) - 264f(t_{i-1}, X(t_{i-1})) + \\ 106f(t_{i-2}, X(t_{i-2})) - 19f(t_{i-3}, X(t_{i-3})) \end{array}\right]}{720}$$

3 Équations différentielles avec conditions aux limites

Les problèmes des conditions aux limites apparaissent souvent dans la pratique. Prenons l'exemple d'un missile balistique dont l'objectif est situé à une distance $x_c(t_f)$. Le problème qui se pose alors est : comment choisir la vitesse initiale (module et orientation) pour que le missile qui est régit par la relation fondamentale de la dynamique entre le point de lancement et la cible atteigne effectivement la cible située en x_c. Plusieurs solutions sont possibles.

3.1 Méthodes des tirs

Les méthodes de tirs sont basées sur les quatre étapes itératives suivantes :

- On choisit une condition initiale $X(t_0)$. Par exemple la vitesse initiale dans le cas du missile.
- On résout alors le problème aux conditions initiales par l'une des méthodes vues précédemment.
- On calcule la valeur de la condition aux limites.
- On modifie alors la condition initiale selon la valeur finale atteinte et la valeur finale souhaitée et on itère les étapes précédentes.

Le problème revient donc à trouver comment modifier la condition initiale pour approcher de proche en proche la valeur finale souhaitée.

Dans le cas du tir du missile, il est facile d'obtenir la solution analytique du problème. Dans le cas général le problème peut être formulé comme la résolution d'un système d'équations non linéaires

$$f = p(c) \tag{27}$$

où f est la valeur finale, p est la solution du problème aux conditions initiales et c est la condition initiale recherchée.

On pourra donc utiliser une méthode de résolution de type Newton.

3.2 Méthode des différences finies

Dans ce cas, on discrétise l'espace d'étude en $(n+1)$ points, puis on approche les dérivées $X^{(k)}$ en utilisant les formules aux différences finies utilisant la valeur de la fonction aux points voisins. A

titre d'exemple, si on suppose un pas constant h ($t_i - t_{i-1} = h$), on peut adopter les approximations suivantes :

$$x_i^{(1)} \approx \frac{x_i - x_{i-1}}{h} \tag{28}$$

$$x_i^{(2)} \approx \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2} \tag{29}$$

Il suffit de réécrire alors l'équation différentielle aux points de discrétisation, en remplaçant les dérivées successives par leurs approximations quand cela est possible. Une fois qu'on aura ajouté les conditions initiales, on aboutit alors à la résolution d'un système linéaire ou non linéaire. If faut noter que comme les approximations sont locales, le système obtenu, même s'il est de taille élevée, est généralement creux. Il est pratiquement tri-diagonal dans le cas d'une équation différentielle d'ordre deux où seuls les indices $(i-1)$, i et $(i+1)$ sont liés.

Chapitre 11 Équations aux dérivées partielles

1 Introduction

Nous limiterons ici notre exposé au cas des équations aux dérivées partielles (EPD) **linéaires** à coefficients **constants** du deuxième ordre, donc de la forme

$$a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial x \partial y} + c\frac{\partial^2 u}{\partial y^2} + d\frac{\partial u}{\partial x} + e\frac{\partial u}{\partial y} + fu = h \tag{1}$$

Afin d'alléger les écritures, les dérivées partielles, et l'EDP seront écrites sous la forme

$$au_{xx} + bu_{xy} + cu_{yy} + du_x + eu_y + fu = h \tag{2}$$

Exemple 32 – *En électrostatique, u est le potentiel et l'équation de Laplace s'écrit :*

$$u_{xx} + u_{yy} = 0 \tag{3}$$

– *Pour l'équation de la chaleur, u est la température et l'on a*

$$u_{xx} = K^2\frac{\partial u}{\partial t} \tag{4}$$

– *L'équation des cordes vibrantes où u étant le déplacement s'écrit*

$$\frac{\partial^2 u}{\partial t^2} = c^2 u_{xx} \tag{5}$$

On utilise généralement la terminologie suivante

- On dit que l'équation est elliptique (ex : équation de Laplace) si $b^2 - 4ac < 0$
- On dit que l'équation est parabolique (ex : équation de la chaleur) si $b^2 - 4ac = 0$
- On dit que l'équation est hyperbolique (ex : équation des cordes) si $b^2 - 4ac > 0$

La résolution d'une EDP n'a de sens que si on impose un certain nombre de conditions aux limites que la solution doit respecter. Ces conditions peuvent être des conditions initiales (le temps par exemple) pour certaines variables et des conditions aux limites sur une région pour d'autres (exemple : champs électrique tangentiel). Les trois cas types sont les suivants :

- On impose la valeur de la solution sur la frontière de la région $u = u_0$. On parle alors de conditions de Dirichlet (exemple potentiel sur une électrode).
- On impose une condition de flux de la solution à la frontière de la région de la forme $\frac{\partial u}{\partial n} = \phi_0$, n étant la normale à la frontière. On parle alors de condition de Neumann.
- Il arrive aussi qu'on impose une condition de la forme $\frac{\partial u}{\partial n} + \alpha u = \beta$. On parle alors de conditions de Cauchy.

2 Méthodes de résolution

2.1 Méthode des différences finies

La méthode est identique à celle utilisée dans le cas des équations différentielles ordinaires avec conditions aux limites. On remplace l'EDP initiale par une équation aux différences valable en des points de dicrétisation de l'espace.

Supposons le cas ou u est une fonction de deux variables x et y :

$$u = u(x, y) \tag{6}$$

On réalise un maillage (grille) du plan en définissant les points

$$x_i = x_{i-1} + h_x = x_0 + i.h_x \tag{7}$$
$$y_j = y_{j-1} + h_y = y_0 + j.h_y \tag{8}$$

Dans ce cas on parle de maillage rectangulaire (figure 1). D'autres type de maillages sont possibles et peuvent être extrêmement utiles comme les maillages triangulaires ou hexagonaux.

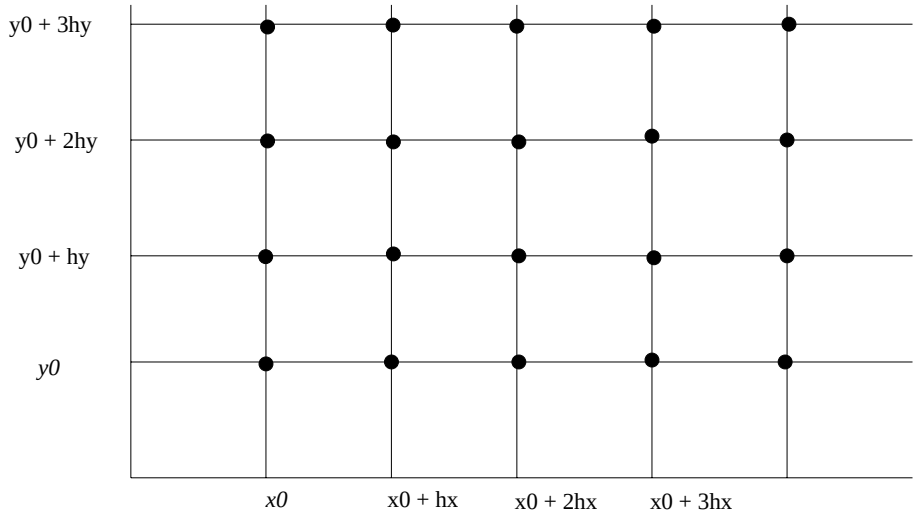

FIG. 1: Maillage rectangulaire

Une fois cette discrétisation de l'espace réalisée, on remplace les dérivées partielles de u aux points (x_i, y_j) par :

$$u_x(x_i, y_j) \approx \frac{u(x_{i+1}, y_j) - u(x_i, y_j)}{h_x} \qquad u_{xx}(x_i, y_j) \approx \frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{2h_x^2} \tag{9}$$
$$u_y(x_i, y_j) \approx \frac{u(x_i, y_{j+1}) - u(x_i, y_j)}{h_y} \qquad u_{yy}(x_i, y_j) \approx \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{2h_y^2} \tag{10}$$

Remarque 28 *D'autres approximations sont possibles. Pour calculer u_x (resp. u_y), on peut prendre les deux voisins $(i+1)$ et $(i-1)$ (resp $j+1$ et $j-1$) :*

$$u_x(x_i, y_j) \approx \frac{u(x_{i+1}, y_j) - u(x_{i-1}, y_j)}{h_x} \qquad u_y(x_i, y_j) \approx \frac{u(x_i, y_{j+1}) - u(x_i, y_{j-1})}{h_y} \tag{11}$$

On réécrit alors l'EDP aux points de discrétisation, en remplaçant les dérivées successives par leurs approximations quand cela est possible. Par exemple avec les formules 11, on ne pourra pas écrire $u_x(x_0, y_0)$ puisqu'on aurait besoin de x_{-1}. Une fois qu'on aura ajouté les conditions aux limites, on aboutit à un système d'équations linéaires qui est soit implicite soit explicite.

- Le système est explicite si pour le calcul de $u(x_i, y_i)$, on n'a besoin que des indices déjà calculés. On a alors une équation de récurrence.
- Le système est sous forme implicite si le calcul de $u(x_i, y_i)$, nécessite la connaissance de u aux indices supérieurs.

Exemple 33 *Soit à résoudre l'EDP similaire à l'équation de la chaleur*

$$u_{xx} = u_t \tag{12}$$

avec les conditions suivantes

$$0 \leq x \leq 1 \qquad t \geq 0 \qquad u(0,t) = u(1,t) = 0 \qquad u(x,0) = f(x)$$

Les formules aux différences vues précédemment restent valables en posant $t = y$.

La dicrétisation de l'espace se fait en posant :

$$\begin{aligned} x_0 &= 0 \qquad & x_i &= i.h_x \\ x_0 &= 0 \qquad & t_j &= j.h_t \end{aligned}$$

En appliquant l'équation aux différences l'équation 12 devient :

$$\frac{u(x_{i+1}, t_j) - 2u(x_i, t_j) + u(x_{i-1}, t_j)}{2h_x^2} = \frac{u(x_i, t_{j+1}) - u(x_i, t_j)}{h_t}$$

Ce qui se met finalement sous la forme

$$u(x_i, t_{j+1}) = \frac{h_t}{2h_x^2} u(x_{i-1}, t_j) + \left(1 - \frac{h_t}{h_x^2}\right) u(x_i, t_j) + \frac{h_t}{2h_x^2} u(x_{i+1}, t_j) \tag{13}$$

Cette équation est sous forme explicite sur le temps. Il suffit de l'initialiser avec

$$\begin{aligned} u(x_1, 0) &= u(0,0) = 0 \\ u(x_0, 0) &= f(x_1) \\ u(x_2, 0) &= f'(x_2) \end{aligned}$$

Exemple 34 *Soit à résoudre l'EDP elliptique (équation de Laplace)*

$$u_{xx} + u_{yy} = 0 \tag{14}$$

avec les conditions suivantes

$$0 \leq x \leq 1 \qquad 0 \leq y \leq 1 \tag{15}$$

L'application des formules aux différences permet d'obtenir si on suppose $h_x = h_y$ pour alléger les écritures

$$u(x_i, y_j) = \frac{1}{4}\left[u(x_{i-1}, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j-1}) + u(x_i, y_{j+1})\right] \tag{16}$$

L'ajout des conditions aux limites permet d'obtenir un système linéaire de forme tridiagonale facile à résoudre avec les méthodes du chapitre 4.

2.2 Méthode des éléments finis

Il s'agit simplement ici de signaler que ces méthodes sont très utilisées dans la construction des structures mécaniques. Elles sont dans un premier temps étudiées en simulation par des méthodes qui utilisent les éléments finis pour l'intégration des EDP.

A la différence des méthodes aux différences finies qui recherchent des valeurs de la solution aux points de discrétisation, la méthode des éléments finis s'attache à rechercher des paramètres de fonctions analytiques permettant d'approcher la solution sur un certain nombre de domaines élémentaires.

Deuxième partie

Travaux dirigés

TD1 : Résolution des systèmes linéaires

Exercice 1

L'objet de cet exercice est de mettre en évidence les problèmes de précision numérique qui peuvent survenir lors de la résolution de systèmes d'équations linéaires.

Soit à résoudre le système d'équations à deux inconnues

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

1. Résoudre littéralement ce système. Que vaut la solution quand ε tend vers 0 ?
2. On suppose que l'on travaille sur un calculateur où les nombres sont représentés en virgule flottante avec 8 chiffres significatifs et que $\varepsilon = 10^{-9}$. Résoudre le système en appliquant la méthode de Gauss (sans pivot partiel). Conclure.
3. Résoudre cette fois-ci le système en appliquant la méthode de Gauss avec pivot partiel. Conclusion.

Exercice2

Soit à résoudre le système d'équations linéaires d'ordre 4 suivant :

$$\begin{array}{ccc} & A.X & = & B \\ \begin{bmatrix} 1 & 2 & 4 & -1 \\ 3 & 1 & 0 & 1 \\ -2 & 1 & 2 & -1 \\ 1 & 1 & -1 & 4 \end{bmatrix} & \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} & = & \begin{bmatrix} 1 \\ -2 \\ 1 \\ 1 \end{bmatrix} \end{array}$$

1. Résoudre le système en appliquant la méthode de Gauss avec pivot partiel.
2. Montrer que le fait de permuter la première et la deuxième ligne de A est équivalent à multiplier à gauche la matrice A par la matrice

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Généraliser au cas de la permutation des lignes i et j.

3. Montrer que soustraire α fois la $1^{ère}$ ligne d'une matrice à la $2^{ème}$ ligne revient à multiplier la matrice par la matrice suivante

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -\alpha & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Généraliser au cas où on veut soustraire α fois la $i^{ème}$ ligne de la $j^{ème}$ ligne.

4. Que réalise la multiplication de la matrice A par la matrice

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -\alpha & 1 & 0 & 0 \\ -\beta & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5. Que vaut la matrice inverse de cette matrice ?
6. En réappliquant la méthode du pivot de Gauss sous forme matricielle, montrer que la matrice A précédente peut s'écrire sous la forme

$$P.A = L.U$$

où P est la matrice des permutations des lignes et des colonnes, L est une matrice triangulaire inférieure et U est la matrice triangulaire supérieure obtenue dans la question 1 par la méthode de Gauss à pivot partiel.

7. Quel est le déterminant de la matrice A ? On rappelle que la permutation des lignes ne change pas la valeur du déterminant.
8. Montrer que le produit $P.P$ donne la matrice identité.
9. Montrer que la résolution du système précédent peut se faire par la résolution par la méthode de substitution des systèmes suivants

$$\begin{aligned} L.Y &= P.B \\ UX &= Y \end{aligned}$$

10. Calculer simplement l'inverse de la matrice U puis de la matrice A.

TD2 : Résolution d'équations nonlinéaires

Exercice 1
On désire trouver un zéro de la fonction en utilisant la méthode de dichotomie

$$f(x) = x\sin(x) - 1$$

1. Calculer $f(0)$ et $f(2)$. Montrer que l'intervalle $[0, 2]$ peut être choisi comme intervalle initial pour cette recherche.
2. Quel est le nombre maximal d'itérations nécessaires pour atteindre une précision sur la racine de 10^{-3}.
3. Appliquer l'algorithme et calculer la valeur approchée de la racine et de la fonction.

Exercice 2 Algorithmes de Regula Falsi (fausse position) et Newton
Soit la fonction $f(x) = e^{-2x} - \cos(x) - 3$

1. Vérifier que le zéro de cette fonction est situé dans l'intervalle $[-1, 0]$.
2. Trouver la valeur de ce zéro par la méthode de Regula Falsi. On prendra comme points initiaux les bornes de l'intervalle précédent.
3. Calculer la valeur de ce même zéro par la méthode de Newton avec comme point initial le point $x_0 = 0$.

Exercice 3 Algorithme de Newton pour les systèmes
On désire résoudre le système d'équations non-linéaires à deux variables x_1 et x_2

$$\begin{cases} x_2 - x_1^2 = 0 \\ (x_1 - 1)^2 + (x_2 - 6)^2 = 25 \end{cases}$$

1. Trouver graphiquement une solution approchée du problème.
2. Résoudre le système par la méthode de Newton exposée en cours.

TD3 : Interpolation polynomiale

Exercice 1 Interpolation de Lagrange

1. Calculer le polynôme d'interpolation de Lagrange interpolant la fonction $\sin(x)$ aux points $x_1 = 0, x_2 = \pi/5, x_3 = 3\pi/5, x_4 = 4\pi/5$.

Exercice 2 Interpolation de Newton

2. Donner de même les coefficients du polynôme d'interpolation de Newton.
3. Proposer un algorithme permettant de calculer la valeur du polynôme d'interpolation de Newton en un point d'abscisse x quelconque.

TD4 : Interpolation spline

Exercice 1 **Splines quadratiques**

Soit la fonction f définie par morceaux :

$$f(x) = \begin{cases} x^2, & x \leq 0 \\ -x^2, & 0 \leq x \leq 1 \\ 1 - 2x, & x \geq 1 \end{cases}$$

1. Monter que la fonction f vérifie les propriétés d'une fonction spline quadratique.

Exercice 2 **Splines cubiques**

Existe-t-il des nombres réels a, b et c tels que la fonction f définie par morceaux par

$$f(x) = \begin{cases} -x, & x \leq -1 \\ ax^3 + bx^2 + cx + d, & -1 \leq x \leq 1 \\ x, & x \geq 1 \end{cases}$$

soit une spline cubique.

Exercice 3 **Calcul d'une splines quadratique**

Trouver la spline quadratique qui passe par les points d'interpolation $(x_0 = -1, y_0 = -1)$, $(x_1 = 0, y_1 = 2)$, $(x_2 = 1, y_2 = -1)$ et $(x_3 = 2, y_3 = 0)$.

Exercice 4 **Calcul d'une splines cubiques**

Nous avons eu l'occasion pendant le cours de traiter la fonction de Runge donnée par

$$f(x) = \frac{1}{1 + x^2}$$

On définit sur l'intervalle $[-5, 5]$ six points équidistants $x_0 = -5$, $x_1 = -3$, $x_2 = -1$, $x_3 = 1$, $x_4 = 3$, $x_5 = 5$.

Calculer la fonction spline cubique pour la subdivision précédente.

TD5 : Méthode des moindres carrés et intégration

Exercice 1 **Méthode des moindres carrés**

On considère les données suivantes issues d'un relevé expérimental :

x_i	-2	-1	0	1	2
y_i	0.5	0.5	2	3.5	3.5

On désire modéliser la relation entre x et y par la fonction linéaire suivante (droite)

$$y(x) = \alpha + \beta x$$

1. Trouver les valeurs de α et β rendant minimale la quantité

$$\sum_i [y(x_i) - y_i]^2$$

2. Calculer la valeur de l'erreur.
3. Examiner le cas d'une approximation par un polynôme du deuxième degré pour la même fonction erreur.

Exercice 2 **Méthode des moindre carrés**

On considère les données suivantes issues d'un relevé expérimental :

x_i	1	2	3	4
y_i	7	11	17	27

On désire modéliser la relation entre x et y sous la forme

$$y = ae^{bx}$$

Calculer a et b au sens des moindres carrés.

Exercice 3 **Intégration méthode de Simpson**

1. Donner une valeur approchée de $\int_1^3 \frac{1}{x} dx$ en utilisant la méthode de Simpson avec $h = 0.01$.
2. Donner un majorant de l'erreur.

TD6 : Programmation linéaire

Exercice 1 **Programmation linéaire**

On considère le problème de programmation linéaire à deux variables x_1 et x_2 défini par

$$\begin{aligned} & \min_{x_1,x_2} (x_2 - x_1) \\ 0 \leq\ & x_1 \\ 0 \leq\ & x_2 \\ -x_1 + 2x_2 \leq\ & 2 \\ x_1 + x_2 \leq\ & 4 \\ x_1 \leq\ & 3 \end{aligned}$$

1. Le problème étant seulement à deux variables, il est simple à résoudre sous forme géométrique (problème plan). Effectuer cette résolution géométrique.
2. Appliquer maintenant la méthode du simplex au problème.

Exercice 2 **Programmation linéaire**

Trouver par la méthode du simplex les quantités x_1, x_2 et x_3 qui minimisent la quantité

$$f(x_1, x_2, x_3) = 2x_1 + 4x_2 + 3x_3$$

sous les contraintes

$$\begin{aligned} x_1 \geq\ & 0 \\ x_2 \geq\ & 0 \\ x_3 \geq\ & 0 \\ x_1 - x_2 - x_3 \leq\ & 1 \\ 2x_1 + x_2 \leq\ & 1 \end{aligned}$$

TD7 : Équations différentielles

Exercice 1 **Résolution des EDO par la méthode de Taylor**

Résoudre l'équation différentielle

$$\frac{dx}{dt} = t + x + x^2$$

en utilisant la méthode de Taylor d'ordre 5 avec $h = 1/500$.

Exercice 2 **Méthode de Runge-Kutta**

Montrer que la formule de Runge-Kutta d'ordre 4 se réduit à une forme simple dans le cas de l'équation différentielle de la forme

$$\frac{dx}{dt} = f(t)$$

A quelle autre méthode pourrait-on la comparer dans ce cas ?

Exercice 3

La vitesse d'une goutte de pluie est régie par l'équation différentielle

$$\frac{dv}{dt} = 32 - \frac{c}{m}v^2$$

où c est la résistance de l'air et m est la masse de la goutte. On démontre que la vitesse $v(t)$ atteint une valeur limite. Vérifier ce fait en résolvant l'équation différentielle précédente.

Troisième partie

Travaux pratiques

TP0 : Initiation à Matlab, à faire avant le début des TP

1 Introduction

L'objectif de cette séance de travaux pratiques est de vous initier au logiciel Matlab. Le logiciel est constitué d'un noyau écrit en langage C dont seuls les exécutables sont fournis et d'un certain nombre de boites à outils qui couvrent les différents domaines des sciences de l'ingénieur. Ces boites à outils appelées Toolbox sont les suivantes :

- Identification Toolbox
- Signal Processing Toolbox
- Control Toolbox
- Image Processing Toolbox
- Optimization Toolbox
- et d'autres beaucoup plus spécifiques à différents domaines de la recherche

Ces Toolbox constituent en fait des contributions de différentes universités et de l'industrie. Les fonctions fournies peuvent être modifiées par l'utilisateur par changement du source. Le langage est le langage Matlab très proche du langage C.

2 Présentation générale

2.1 Installation/initialisation

Lancer le logiciel puis taper la commande path. Vous verrez alors s'afficher la liste des répertoires disponibles dans l'environnement de travail. Pour obtenir l'aide sur une commande, il suffit de lancer : help nom de la commande. Lancer "help path", puis ajouter le répertoire sur lequel vous voulez travailler.

2.2 Commande générales

Les commandes générales permettent d'avoir des informations sur l'environnement de travail. Ces commandes sont (extrait du fichier help de Matlab) :

Managing commands and functions.
help On-line documentation.
what Directory listing of M-, MAT- and MEX-files. type List M-file.
path Control Matlab's search path.
Managing variables and the workspace.
who List current variables.

load Retrieve variables from disk.
save Save workspace variables to disk.
clear Clear variables and functions from memory.
size Size of matrix. length Length of vector.
disp Display matrix or text.
Working with files and the operating system.
cd Change current working directory.
dir Directory listing.
delete Delete file.
! Execute operating system command.
clc Clear command window.
home Send cursor home.
Echo Commands inside script files.
Type List mfile contents
Quitting from Matlab.
quit Terminate Matlab.
Familiarisez-vous à ces différentes fonctions avec l'aide de l'enseignant.

2.3 Représentation des données

A la différence d'un langage de programmation, Matlab crée une variable lors de son affectation. Ainsi la commande :

A =[1 2 5 ; 3 4 7]

définit une matrice 2 x 3. On peut le vérifier par la fonction **size**

[m,n]=**size**(A)

Une matrice est définie par le contenu entre crochets. Un espace sépare les éléments d'une même ligne entre eux et le point virgule définit une nouvelle ligne ; la virgule définit des blocks d'une même ligne. Si on ne met pas un point virgule à la fin d'une commande le résultat est affiché.

Le logiciel ne travaille qu'avec des nombres réels, la troncature est réalisée automatiquement lors de l'affichage suivant la précision désirée. En fait la seule représentation des données est la forme matricielle : un nombre est seulement une matrice de dimension 1.

2.4 Création de fonctions

Il y a deux manières différentes pour travailler avec Matlab. La première est d'utiliser la fenêtre de commande et d'appeler les fonctions (de Matlab ou celles que vous aurez créées) les unes après les autres comme précédemment. La seconde est de créer un file.m qui contient la liste des commandes que vous voulez exécuter : des fonctions de Matlab ou créées par vous. Ces fichiers file.m peuvent recevoir des arguments et en renvoyer. Il faudra alors spécifier en en-tête de la fonction la ligne suivante :

function [ou1, ou2, ...] = file(in1, in2, in3, ...)

Si cette ligne figure en en-tête de la fonction alors les variables définies dans celle-ci seront locales et les variables définies dans l'environnement de travail ne seront plus accessibles à la fonction. Elle seront accessibles si elles figurent en argument d'entrée ou si elles sont définies en global.

Écrire un programme recevant en entrée les coefficients (a, b, c) d'un polynôme du second degré. Ce programme retourne les racines si elles existent ou donne un message indiquant que le discriminant est négatif.

La fonction **disp** vous sera utile.

3 Fonctions mathématiques élémentaires

La plupart des fonctions mathématiques sont applicables aux grandeurs scalaires, vectorielles ou matricielles. Dans les deux derniers cas, la valeur de la fonction est calculée pour chacun des éléments.

liste des fonctions : sin, sinh, *a*sin, *a*sinh, *cos, cosh, acos, acosh,tanh,...,*exp, log, log10, sqrt, abs, angle, conj, real, imag,....

4 Algèbre générale

4.1 Opération sur une matrice

1. Créer une matrice A de dimension 5×5. Vous pouvez choisir les valeurs des éléments ou utiliser la fonction *rand*.
2. Calculer la trace de cette matrice "*trace*"
3. Calculer le déterminant de la matrice "*det*"
4. Calculer les valeurs et vecteurs propres "*eig*"
5. Calculer l'inverse de la matrice "*inv*". Vérifier le résultat.
6. Calculer la transposée de cette matrice (fonction ')
7. Calculer la norme 1, 2, infinie de cette matrice. (Fonction *norm* et ses différentes options). Vérifier les résultats en utilisant les définitions.
8. Que réalise la fonction *sqrtm*?

4.2 Extraction d'éléments et de blocs

1. Extraire la première colonne de A et la ranger dans le vecteur B. Utilisation de " : "
2. Calculer la somme des éléments de B. Fonction *sum*. Faire la même opération sur la matrice A. Conclusion.

4.3 Opérations entre deux ou plusieurs matrices

1. Créer une matrice identité de dimension 6. Fonction *eye*
2. Créer une matrice unité (tous les éléments égaux à 1). Fonction *ones*
3. Créer un vecteur V de dimension 3. Fonction *rand*. Créer une matrice diagonale de dimension 3×3 ayant les éléments de V sur la diagonale principale. Créer finalement une matrice diagonale de dimension 5×5 ayant les éléments de V sur la deuxième diagonale supérieure. Fonction *diag*.
4. Créer deux matrices A et B de dimensions 3×5 et 5×4.
5. Calculer les produit A*B et B*A. Conclure
6. Calculer les opérations A.*A et A./A. Conclure
7. Créer une matrice A de dimension 5×5. Calculer l'opération A/A. Conclure.

5 Fonctions graphiques

Les fonctions graphiques de Matlab sont rassemblées sous plusieurs variantes de la fonction *plot*. Le tracé des courbes se fait sur des fenêtres graphiques. Une fenêtre graphique est créée automatiquement lors du premier tracé. On peut aussi créer ou rappeler une fenêtre graphique en utilisant les menus ou la fonction $figure(numéro\ de\ figure)$.

Tracé d'un signal
Nous allons réaliser le tracé de plusieurs signaux sinusoïdaux

1. Définition de la base de temps t=(0 :0.01 :1) ;
2. Définir maintenant trois signaux sinusoïdaux $s1$, $s2$, $s3$ de fréquences : F1 = 5Hz, F2 = 15Hz, F3 = 30Hz. On peut prendre directement le sinus d'un vecteur, fonction *sin*.
3. Tracer le signal s1 en fonction du temps. Fonction *plot*.
4. Changer la couleur et le type de trait. Option de la fonction *plot*.
5. Tracer de même les signaux s2 et s3.
6. Tracer sur un même graphique les signaux s1 et s2. Fonction *hold*.
7. Donner un titre, un label (étiquette) aux axes. Fonctions $title, xlabel, ylabel$.
8. Changer les axes. Fonction *axis*.
9. Ajouter un quadrillage. Fonction *grid*.
10. Effacer la fenêtre. Fonction *clg*.
11. Faire un tracé tri-dimensionnel des points ayant pour coordonnées $(t, s1, s3)$.
12. Si vous avez le temps, regarder les fonctions *subplot*, *mesh*, ...

6 Éléments de programmation

Rappelons que seul le type matrice existe en langage de programmation de Matlab.

6.1 La rupture de séquence

Les différentes opérations de logique sont : $==, <, >, <=, >=, ou\tilde{}\ =$
Les syntaxes possibles sont les suivantes :

```
if var1 op var2
        instructions
        instructions
end ;
```

ou

```
if var1 op var2
        instructions
else
        instructions
end ;
```

ou encore

```
if var1 op var2
        instructions
elseif var3 op var4
        instructions
else
        instructions
end ;
```

6.2 Les boucles

Il existe deux types de boucles en Matlab. La boucle *for* et la boucle *while*

```
for i=ind1 :pas :ind2
                instructions
                instructions
end ;
while (var1 op var2)
                instructions
                instructions
end ;
```

6.3 Opérations d'entrées-sorties

La fonction *disp* permet d'afficher un message ou une variable dans l'environnement.

La fonction *input* permet au contraire la lecture d'un nombre ou d'un caractère

Les commandes *save* et *load* permettent respectivement de sauvegarder et de charger des données sous forme de fichier binaire (fichier.mat).

1. Sauvegarder votre environnement sous un fichier (.mat). *save*
2. Effacer l'environnement. *clear*
3. Recharger le fichier. *load*. Refaire les différentes opérations sur quelques variables seulement.
4. Écrire un programme permettant de réordonner les élements d'une matrice A par colonne. La dimension de la matrice est quelconque. Utiliser les fonctions $size, length, ...$

TP1 :Résolution des équations non-linéaires

1 Objectif

L'objectif du TP est de mettre en oeuvre et de comparer différentes méthodes de résolution d'équations non linéaires du type :

$$f(x) = 0$$

La racine recherchée sera désignée par la lettre s.

2 Fonctions à une variable, exposé des méthodes

1. Écrire une fonction sous MATLAB donnant la valeur de la fonction dont on cherche les racines en un point x quelconque. Cette fonction devra accepter des variables vectorielles.

On choisira pour fonction $f(x) = \cos(x).\cosh(x) + 1$

2. Écrire une fonction sous MATLAB donnant la valeur de la dérivée de la fonction dont on cherche les racines en un point x quelconque.

$$function \quad [y] = df(x)$$

3. Programmer les quatre méthodes précédentes

$$\begin{gathered} function \quad [] = dich(a, b, epsilon) \\ function \quad [] = reg\_fal(a, b, epsilon) \\ function \quad [] = \sec ante(a, b, epsilon) \\ function \quad [] = newton(a, epsilon) \end{gathered}$$

Ces différentes fonctions feront appel aux programmes d'évaluation de la fonction et de sa dérivée $f.m$ et $df.m$.

4. Quel est le nombre d'itérations k nécessaires pour chacune de ces méthodes si on désire arrêter l'algorithme au moment où $|f(x^{(k)})| \leq 10^{-9}$?
5. Quelle est la longueur du dernier segment ?

3 Systèmes d'équations non linéaires

Soit à résoudre le système d'équations non-linéaires suivant :
$e^{xy} + x^2 + y - 1.2 = 0$
$x^2 + y^2 + x - 0.55 = 0$

1. Écrire une fonction sous MATLAB donnant la valeur de la fonction dont on cherche les racines en un point $u = [x; y]$ quelconque

$$function \quad [F] = sys(u)$$

u et F sont des vecteurs de dimension 2×1.

2. Écrire une fonction sous MATLAB donnant la valeur du Jacobien sous la forme d'une matrice 2×2.

$$function \quad [J] = Jacobien(u)$$

3. Programmer la méthode de Newton pour ce cas. On prendra comme point initial le vecteur $u^{(0)} = [x^{(0)}; y^{(0)}] = [0.6; 0.5]$;

$$function \quad [] = new(u, epsilon)$$

4. Quel est le nombre d'itérations k nécessaires pour chacune de ces méthodes si on désire arrêter l'algorithme au moment où $||f(x^{(k)})|| \leq 10^{-9}$.

4 Zéros de polynômes

Le polynôme

$$P(x) = x^3 + 4x^2 - 55x + 50$$

admet pour racines les nombres $-1, 10$ et -5.

1. Programmer la méthode exposée en cours. On prendra comme point initial une valeur proche de la racine recherchée. On arrêtera l'algorithme lorsque le critère $|P(p)| < 1e^{-6}$ est satisfait.
2. Proposer une amélioration du programme permettant de calculer toutes les racines.

TP2 : Interpolation polynômiale et splines

1 Interpolation polynômiale

1.1 Interpolation de Lagrange

Soit la fonction $f(x) = \log(x) - \frac{2(x-1)}{x}$.

- Tracer cette fonction sous MATLAB.
- Calculer les valeurs de la fonction aux points $x_0 = 1, x_1 = 2, x_2 = 4, x_3 = 8, x_4 = 10$.
- Calculer le polynôme de Lagrange de degré 4 interpolant cette fonction. La fonction "*poly*" pourrait vous être utile.
- Donner le nombre de divisions effectuées pour le calcul des coefficients.
- Quelle est la valeur approchée de la fonction aux points $x = 2.9$ et $x = 5.25$? Donner la valeur de l'erreur.

1.2 Interpolation de Newton

Soit la fonction de Runge $f(x) = \frac{1}{1+x^2}$.

- Tracer cette fonction sous MATLAB dans l'intervalle $[-5, 5]$.
- Calculer les valeurs de la fonction aux points $x_0 = -5, x_1 = -3, x_2 = -1, x_3 = 1, x_4 = 3, x_5 = 5$
- Calculer le polynôme P de Newton de degré 5 interpolant cette fonction. On programmera l'algorithme présenté en cours.
- Écrire un programme permettant de calculer la valeur du polynôme en un point quelconque.
- Tracer sur un même graphique la fonction f ainsi que le polynôme P. Examiner les effets de bord.
- Comment évoluent ces effets avec l'ordre du polynôme ?
- Reprendre les 5 questions précédentes en prenant comme subdivision de l'intervalle les points de Chebychev. Conclusion.

2 Interpolation par des Splines

2.1 Splines quadratiques

Supposons donnés $(n+1)$ points définis par les couples (x_i, y_i), $(i = 0, 1, ..., n)$. L'interpolation par des splines quadratiques équivaut à l'approximation sur chaque intervalle $[x_i, x_j]$ par un polynôme

du second degré. On impose aux différents point la continuité des valeurs des polynômes et de leurs dérivées. Nous avons montré en cours que pour pouvoir résoudre le problème, il est nécessaire d'imposer la valeur de la dérivée en x_0 (ex. $f'(x_0) = 0$)

1. Écrire un programme général d'interpolation par des splines quadratiques. Le programme reçoit en entrée les couples (x_i, y_i) ainsi que la valeur de la dérivée au point x_0. Le programme retournera une matrice $3 \times n$ contenant les coefficients des polynômes $S_i(x)$.
2. Appliquer aux points $X = [0, 1, 2, 3]; Y = [0, 0.5, 2.0, 1.5]$ ainsi qu'à la fonction de Runge.
3. Examiner l'effet de différentes valeurs de $f'(x_0)$.

2.2 Splines cubiques

1. Écrire un programme général d'interpolation par des splines cubiques. Le programme reçoit en entrée les couples (x_i, y_i) ainsi que la valeur de la dérivée au point x_0. Le programme retournera une matrice $4 \times n$ contenant les coefficients des polynômes $S_i(x)$.
2. Appliquer aux points $X = [0, 1, 2, 3]; Y = [0, 0.5, 2.0, 1.5]$ ainsi qu'à la fonction de Runge. On tracera sur un même graphique les résultats obtenus par interpolation polynômiale, les splines quadratiques et cubiques. Comparer.

TP3 : Techniques d'optimisation et méthodes des moindres carrés

1 Introduction

L'objectif du T.P est de se familiariser avec quelques techniques d'optimisation en utilisant la boite à outils "*Matlab Optimisation*".

On désigne sous le nom d'optimisation, la minimisation ou la maximisation d'une fonction. Un problème de maximisation peut être ramené sans difficulté à un problème de minimisation. En effet, résoudre le problème :

$$\max_x f(x)$$

revient à résoudre le problème de minimisation suivant :

$$\min_x(-f(x))$$

Dans toute la suite nous ne parlerons que de minimisation.

Soit f une fonction scalaire dépendant d'une variable scalaire x ou vectorielle X. On distingue différents problèmes d'optimisation résolus par différentes fonctions de la boite à outils.

1. problème sans contrainte scalaire $\min_x f(x)$ $x = \text{fmin}('f', x_0)$
2. problème sans contrainte vectoriel $\min_X f(X)$ $X = \text{fminu}('f', X_0)$ ou $X = \text{fmins}('f', X_0)$
3. problème avec contraintes vectoriel $\min_x f(X)$ avec $G(X) \leq 0$ $X = \text{constr}('f', X_0)$
4. problème minmax $\min_x \{\max F(X)\}$ avec $G(X) \leq 0$ $X = \text{minmax}('f', X_0)$
5. moindres carrés non linéaires $\min_x \sum F(X). \times F(X)$ $X = \text{leastsq}('f', X_0)$

La boite à outils permet de résoudre de plus quelques problèmes matriciels tels que :

1. la programmation linéaire $\min_X f^T X$ avec $A.X \leq b$ $X = \text{lp}(f, A, b)$
2. la programmation quadratique $\min_X \left(\frac{1}{2} X^T H X - c^T X\right)$ avec $A.x \leq b$ $x = \text{qp}(H, c, A, b)$

1.1 Exercice d'application

Soit à minimiser la fonction $f(x) = e^{-x^2}(x^3 + 4x + 1)$

- Tracer la fonction et déterminer approximativement la position de son minimum
- Entrer la commande suivante : [x,options]=fmin('$e^{-x^2}(x^3+4x+1)$',-3,3). La chaîne de caractères donnant la fonction est évaluée à chaque itération.
- Taper "help FOPTIONS" et examiner le vecteur options précédent. En déduire le nombre d'itérations et la valeur de la fonction en ce minimum.

Il est à noter que la chaîne de caractères précédente définissant la fonction peut être remplacée par une fonction Matlab (.m file) ayant pour argument d'entrée la variable x et comme argument de sortie la valeur de la fonction

ex :

```
function [f]=f1(x)
f=exp(-x^2)*(x^3+4*x+1) ;
```

La fonction doit être sauvegardée sous le nom f1.m. La minimisation se fera alors par l'appel suivant :

```
[x,options]=fmin('f1',-3,3)
```

2 Minimisation d'une fonction à plusieurs variables

2.1 Minimisation sans contrainte

En utilisant la fonction 'fminu', trouver le minimum de la fonction à deux variables $x = [x_1, x_2]$:

$$\min_x f(x) = e^{(-x_1^2 - x_2^2)}(4x_1^2 + 4x_1x_2 + 2x_2 + 1)$$

On choisira comme point solution initial $x_0 = [-1, 1]$

- Quel est le nombre d'itérations ? Donner la position du minimum et la valeur de la fonction en ce point.
- Choisir une autre solution initiale et examiner l'évolution du nombre d'itérations

2.2 Minimisation avec contraintes

Résoudre le problème suivant issu du problème précédent auquel deux contraintes ont été ajoutées :

$$\begin{gathered}\min_x f(x) = e^{(-x_1^2 - x_2^2)}(4x_1^2 + 4x_1x_2 + 2x_2 + 1) \\ 1.5 + x_1x_2 - x_1 - x_2 \leq 0 \\ 2x_1^2 - x_2 - 10 \geq 0\end{gathered}$$

Les contraintes doivent être avant tout converties sous la forme d'un vecteur $g(x) < 0$. On utilisera la fonction "constr" pour résoudre ce problème. La mfile définissant la fonction à minimiser doit non seulement retourner la valeur de la fonction à minimiser mais aussi la valeur des contraintes sous la forme :

```
function [f,g]=f2(x)
f=exp(-x(1)^2-x(2)^2)*(4*x(1)^2+4*x(1)*x(2)+2*x(2)+1 ;
g(1)=1.5+x(1)*x(2)-x(1)-x(2)
g(2)=.....à vous de compléter
```

Répondre aux questions du sous chapitre précédent.

Des contraintes d'égalité peuvent aussi être définies, elles doivent alors être placées au début du vecteur g et leur nombre doit être spécifié dans la composante 13 du vecteur "options".

Résoudre le problème précédent en ajoutant la contrainte supplémentaire

$$2x_1 + x_2 = 0$$

3 Méthode des moindres carrés non-linéaires

Rappelons que cette méthode permet la minimisation des fonctions se présentant sous forme de somme de termes au carré, c'est à dire de la forme :

$$\min_X \sum F(X). \times F(X)$$

Cette méthode est bien adaptée au problème qui consiste à faire correspondre un modèle à un certain nombre de mesures (identification).

3.1 Application 1

La concentration $z(t)$ dans un processus chimique peut être modélisée par la loi suivante :

$$z(t) = a_1 + a_2 e^{b_1 t} + a_3 e^{b_2 t}$$

Le relevé expérimental de la concentration est donné sur le tableau suivant :

t_k	0	0.5	1	1.5	2	3	5	8	10
z_k	3.85	2.95	2.63	2.33	2.24	2.05	1.82	1.8	1.75

En utilisant la fonction 'leastsq', trouver les paramètres a_i et b_i minimisant l'erreur quadratique moyenne. On prendra comme solution initiale : $a_1 = 1.75, a_2 = 1.20, a_3 = 0.8, b_1 = -0.5, b_2 = -2$.

3.2 Application 2

Les grandeurs macroscopiques de l'écoulement du trafic routier sont exprimées en terme de débit, taux d'occupation et de vitesse. Ces grandeurs sont généralement mesurées à l'aide de boucles magnétiques enterrées dans le bitume.

Le fichier "trafic.m" contient le relevé de ces grandeurs pour une station située sur l'autoroute A6a à proximité de la sortie porte d'Orléans.

- Tracer la vitesse en fonction du taux d'occupation

Ce diagramme appelé diagramme fondamental peut être approximé par une fonction de la forme

$$v = v_f \exp(-\tfrac{1}{a}(\tfrac{o}{o_{cr}})^a)$$

ou v est la vitesse, o est le taux d'occupation et v_f est la vitesse libre. o_{cr} est la densité critique et a est un facteur.

- Charger les fichiers vitesse.mat et to.mat
- Trouver les valeur des coefficients précédents.

4 Méthode des moindres carrés linéaires

La méthode des moindres carrées est très utilisée dans les sciences expérimentales et plus particulièrement dans les problèmes d'estimation et d'identification. Comme son nom l'indique, cette méthode consiste à minimiser la norme quadratique (norme euclidienne) d'une fonction appelée fonction d'erreur. Le problème à résoudre peut s'énoncer de la manière suivante :

Soit $x_1, x_2, ..., x_n$, N inconnues dépendant de constantes connues c_{ik} et d_{ik}. Trouver les valeurs des x_i rendant minimale la norme du vecteur d'erreur r donnée par la relation suivante :

$r_i = \sum_{k=1}^{n} c_{ik} x_k - d_i \qquad (i = 1, 2, ..., n)$

La relation précédente s'écrit sous la forme matricielle

$Cx - d = r$

1. Montrer que la norme du vecteur r s'écrit :
   $\|r\|^2 = r^T r = x^T C^T C x - 2(C^T d)^T x + d^T d$
   Dans la suite nous supposerons que la matrice C est de rang plein, la matrice $A = C^T C$ est donc symétrique définie positive.
2. Dériver l'expression précédente et montrer que le minimum est obtenu pour le vecteur x vérifiant
   $A.x - b = 0$
   avec $b = C^T d$
3. L'équation précédente est appelée équation normale, nous la verrons plus en détail en traitement statistique du signal.

La matrice A étant définie positive, le vecteur x est déterminé de manière unique comme solution du système linéaire. Cette matrice étant de plus symétrique nous pouvons utiliser la méthode suivante :

- décomposition de Cholesky de $A = LL^T, L$ est triangulaire inférieure
- résoudre par la méthode de substitution les équations suivantes :

  $Ly - b = 0$ et $L^T x + y = 0$
- calculer le résidu $r = Cx + d$

4.1 Application 1

On considère les données suivantes :

x_i	-2	-1	0	1	2
y_i	0.5	0.5	2	3.5	3.5

On désire modéliser la relation entre x et y par la fonction linéaire suivante (droite)

$$y(x) = \alpha + \beta x$$

1. Trouver les valeurs de α et β rendant minimale la quantité

$$\sum_i [y(x_i) - y_i]^2$$

On commencera par calculer les matrices C, d, A, b et L.

2. Calculer le vecteur d'erreur ainsi que sa norme.

4.2 Application 2

Les relevés expérimentaux d'un processus physique donnent :

x_i	0.2	0.5	1	1.5	2.0	3.0
y_i	0.3	0.5	0.8	1.0	1.2	1.3

1. Tracer les y_i en fonction des x_i.
2. La caractéristique est linéaire au voisinage de l'origine et présente une asymptote horizontale pour les grandes valeurs de x. On décide donc d'adopter le modèle suivant :

$$f(x) = \alpha_1 \frac{x}{1+x} + \alpha_2(1 - e^{-x})$$

Déterminer les valeurs des deux coefficients.

3. Tracer sur un même graphique le modèle et les données expérimentales.

TP4 :Résolution des équations différentielles et initiation à Simulink

1 Méthodes d'Euler et de Runge-Kutta

Les différentes méthodes que nous étudierons tentent de résoudre une équation différentielle de la forme

$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y)$$

On montre en effet aisément que toute équation différentielle peut se mettre sous la forme précédente.

1.1 Méthode d'Euler

1. Programmer l'algorithme d'intégration des équations différentielles par la méthode d'Euler
2. Vérifier que

$$y = \frac{y_0.k.e^{\lambda.x}}{k + y_0(e^{\lambda.x} - 1)}$$

est solution de l'équation différentielle

$$\begin{aligned} y(0) &= y_0 \\ \frac{\mathrm{d}y}{\mathrm{d}x} &= \lambda.y.\left(1 - \frac{y}{k}\right) \end{aligned}$$

3. L'appliquer à la résolution de l'équation différentielle précédente sur l'intervalle $[0, 20]$ avec $y_0 = 0.1, k = 2$ et $\lambda = 0.5$.
4. Tracer sur un même graphique la solution exacte et la solution approchée.
5. Calculer l'erreur commise et étudier son évolution avec h.

1.2 Méthode de Runge-Kutta

1. Répondre aux même questions que précédemment.
2. Comparer les deux méthodes pour des pas de discrétisation identiques.
3. Pour un pas de discrétisation donné pour la méthode de Runge Kutta, pour quelle valeur de h la méthode d'Euler atteint la précision de la méthode de Runge Kutta ?

2 Initiation à Simulink

2.1 Introduction

Simulink représente la couche graphique de MATLAB. Simulink permet de construire très rapidement un environnement de simulation en manipulant des macro-fonctions (systèmes, correcteurs, sources, oscilloscopes, interfaces d'entrée sortie,...) représentées sous forme d'objets contenus dans la fenêtre obtenue en tapant la commande Simulink dans la fenêtre de commande.

Vous pouvez créer une fenêtre de simulation par la commande "file + new + model" que vous sauverez sous nomfichier.m. Il suffira alors de copier ou de déplacer les blocs dont vous avez besoin à partir de la fenêtre Simulink.

2.2 Génération et visualisation de signaux (librairies Sources et Sinks)

Visualiser les différents signaux. Agir sur la fréquence, l'amplitude et les échelles des différents organes de visualisation.

Tester les blocs permettant l'échange de données entre l'environnement de travail (fenêtre de commande de Matlab) et Simulink.

2.3 Simulation de systèmes linéaires discrets

On considère le système linéaire du second ordre en temps continu défini par la fonction de transfert

$$H(p) = \frac{1}{p^2 + 2p + 10}$$

1. Discrétiser la fonction de transfert en utilisant les fonctions $tf2ss, c2d, ss2tf$, pour une période d'échantillonnage de $10ms$.
2. Écrire l'équation aux différences régissant le système discret obtenu.
3. Simuler le système sous Simulink on utilisant comme signal d'entrée un créneau. Visualisez la sortie avec et sans bloqueur d'ordre 0 (faire varier la fréquence du bloqueur).

2.4 Simulation de systèmes linéaire continus

On considère le système continu

$$H(p) = \frac{20}{(a+p)\left(1+\frac{p}{10}\right)\left(1+\frac{p}{40}\right)}$$

1. Calculer la représentation d'état du système.
2. Simuler le système en utilisant des intégrateurs simples, une fois le schéma terminé
3. Construire le système à partir d'intégrateurs simples et le simuler. Vérifier que le système est bien instable pour $a < 0$.
4. Faire varier les différents paramètres de simulation. Quels sont leurs effets sur le résultat ?

On corrige le système ($a = 1$) par un correcteur à avance de phase

$$C(p) = \frac{1+\frac{p}{20}}{1+\frac{p}{100}}$$

1. Construire le système en boucle fermée.
2. Simuler le système avec différents signaux d'entrée. Vérifier que le système bouclé est stable, visualiser sa réponse pour plusieurs valeurs de a.

2.5 Systèmes non-linéaires

1. Reprendre le système bouclé précédent et y inclure des contraintes de saturation du signal de commande (en amplitude et en vitesse de variation). Quel est l'effet de ces non-linéarités ?
2. Construire un signal sinusoïdal modulé en amplitude. La fréquence de la porteuse sera choisie égale à $f_p = 1GHz$ et celle du signal modulant à $f_m = 1MHz$. Le signal modulé prend la forme

$$s(t) = (E_0 + A_p \sin(2\pi f_p t)) \sin(2\pi f_m t)$$

Visualisez le signal.

Quatrième partie

Quelques fonctions de Matlab

Algèbre linéaire

1 Création de matrices et vecteurs

Matrice nulle $n \times m$	$A = zeros(n, m)$	
Matrice identité $n \times n$	$A = eye(n)$	
Matrice de 1 $n \times m$	$A = ones(n, m)$	
Matrice diagonale	$A = diag([a_1, a_2, ..., a_n])$	
Matrice à val aléatoires $n \times m$	$A = rand(n, m)$	distribution uniforme
Matrice à val aléatoires $n \times m$	$A = randn(n, m)$	distribution normale
Matrice de Toeplitz	$A = toeplitz(c)$	c première ligne

2 Manipulations et opérations sur les vecteurs et matrices

Extraction de l'élément i, j	$A(i, j)$
Extraction d'un ligne i	$A(i, :)$
Extraction d'une colonne j	$A(:, j)$
Extraction de bloc	$A([i_1, i_2, ..], [j_1, j_2, ..])$
Extraction de la diagonale	$diag(A)$
Extraction bloc triangulaire supérieur	$triu(A)$
Extraction bloc triangulaire inférieur	$tril(A)$
Transposition	A'
Inversion	$inv(A) \quad 1/A \quad A\hat{} - 1$
Puissance (α réel) de matrice	A^{α}
Puissance α élét. par élét.	$A.^{\alpha}$
Déterminant	$det(A)$
Rang	$rank(A)$
Nombre condition	$cond(A)$
Vecteurs, valeurs propres	$[v, d] = eig(A)$
norme 1,2,inf,fro	$norme(A, 1), ..., norm(A,' fro')$

3 Factorisation de matrices

Factorisation QR	$[Q, R] = qr(X)$
Factorisation LU	$[L, U, P] = lu(X)$
Factorisation de Cholesky	$R = chol(X)$

4 Opérations entre matrices et vecteurs

Addition de matrices	$A + B$
Multiplication de matrices	$A * B$
Multiplication élét. par élét	$A. * B$
Résolution du système $Ax = b$	$x = inv(A) * b$ ou $x = A\backslash b$

5 Les polynômes

Un polynôme est représenté par le vecteur ligne ou colonne de ses coefficients, du degré le plus élevé au plus faible. Le polynôme

$$p(x) = x^2 + 3x - 1$$

sera déclaré sous Matlab par la commande

$$p = [1, 3, -1]$$

Déclaration par les racines	$p = poly(v)$	v, vect. des racines
Polynôme caractéristique d'une matrice A	$p = poly(A)$	
Valeur d'un polynôme en a	$p0 = polyval(p, a)$	
Racines d'un polynôme p	$r = roots(p)$	v, vect. des racines
Produit de deux polynômes	$p = conv(p1, p2)$	

L'addition de deux polynômes pose des problèmes, car cela revient à additionner deux vecteurs de dimensions incompatibles si les polynômes ne sont pas de même degré. Pour résoudre ce problème, il suffit d'écrire une fonction qu'on va appeler '*polyadd*' qui met les polynômes à la même dimension avant de les additionner.

$$
\begin{array}{ll}
function & p = polyadd(p1, p2) \\
if & (size(p1, 2) > 1), p1 = p1'; end; \\
elseif & (size(p2, 2) > 1), p1 = p1'; end; \\
end; & \\
if & (size(p1, 1) >= size(p2, 1)) \\
 & p = p1 + [zeros(size(p1, 1) - size(p2, 1)); p2]; \\
else & \\
 & p = p2 + [zeros(size(p2, 1) - size(p1, 1)); p1]; \\
end; &
\end{array}
$$

Différentiation

Différences finies	$diff(x)$
Gradient, dérivées partielles	$[px, py] = gradient(f, dx, dy)$

Intégration

Intégration d'une fonction f	$Q = quad('f', a, b, tol)$	$y = f(x)$ fonction f.m
Intégration d'une fonction f	$Q = quad8('f', a, b, tol)$	Newton-Cote ordre 8

Interpolation

Interp. linéaire	$YI = interp1(X, Y, XI, 'linear')$	Polynômiale d'ordre 1
Interp. cubique	$YI = interp1(X, Y, XI, 'cubic')$	Polynômiale d'ordre 3
Interp. linéaire	$YI = interp1(X, Y, XI, 'spline')$	Spline
Interp. 2D	$ZI = interp2(X, Y, Z, XI, YI, 'méthod')$	Idem, 2 dimensions
Interp. Spline	$YI = spline(X, Y, XI)$	Splines cubiques

Équations différentielles ordinaires

Runge-kutta ordre 2 et 3	$[t,y] = ode23('f', t0, tf, y0, tol, 1)$	EDO dans $yprime = f(t,y)$
Runge-kutta ordre 4 et 5	$[t,y] = ode45('f', t0, tf, y0, tol, 1)$	EDO dans $yprime = f(t,y)$

Il faut noter aussi l'existence des méthodes d'intégration suivantes

$$RK23, RK45, ADAMS, EULER, LINSIM, SFUNC, DSFUNC, TRIM, LINMOD$$

utilisables pour des systèmes d'équations différentielles décrites sous forme de fonctions Simulink (sfunctions).

Optimisation

Zéro d'une fonction	$x = fsolve('f', x0, options)$	
Minimisation scalaire	$xopt = fmin('f', x1, x2)$	1 variable
Minimisation sans contraintes	$xopt = fmins('f', x0, options)$	simplexe
Minimisation quadratique	$xopt = f\min u('f', x0, options)$	Méthode BFGS
Minimisation avec contraintes	$X = constr('fun', x, options)$	$[f, g] = fun(x)$
Programmation linéaire	$X = lp('f', A, b, VLB, VUB)$	
Moindres carrés	$X = leastsq('fun', X0, OPTIONS,)$	
Moindres carrés ≥ 0	$X = nnls(A, b)$	$x \geq 0$
Min-max	$X = \min i\max('fun', X0, OPTIONS,)$	
Programmation quadratique	$X = qp(H, f, A, b, VLB, VUB)$	

- Il faut noter que la plupart des fonctions ne sont pas disponibles sous matlab. Elles le sont dans la toolbox "optimisation" :

$$fsolve, f\min u, constr, lp, ...$$

- "options" est le vecteur des options qu'on peut obtenir par la commande

$$options = foptions$$

  Faire

$$help \text{ options}$$

  pour voir la signification des différentes composantes. On peut alors changer certaines valeurs.
- g est le vecteur des contraintes
- Certaines fonctions acceptent une expression analytique du gradient de la fonction.

Index