UNIVERSITE SIDI MOHAMED BEN ABDELLAH
FACULTE DES SCIENCES ET TECHNIQUES
Département des Mathématiques

Licence Sciences et Techniques (LST)

MATHEMATIQUES ET APPLICATIONS

MEMOIRE DE FIN D'ETUDES

Pour l'obtention du Diplôme de Licence Sciences et Techniques

Titre

Réalisation d'un Logiciel de Calcul en Analyse Numérique

Présenté par :

- **Bouchannafa Karim**

Encadré par :

- **Pr M. Masrar (FST)**

Soutenu Le 19 Juin 2015 devant le jury composé de:

- **Pr M. Masrar (FST)**
- **Pr A. Tajmouati (FSDM Fès)**
- **Pr A. Oudghiri (FST)**

Stage effectué à

FST de Fès, Département mathématiques

Année Universitaire 2014 / 2015

Remerciement

Nous tenons par le présent travail à témoigner notre reconnaissance envers mon encadrant Pr. M. MASRAR pour toute son aide et sa responsabilité ainsi que pour ses conseils.

Je ne peux oublier d'exprimer ma reconnaissance à mes formateurs à l'FSDM et l'FST de Fès notamment ceux des départements mathématiques Pr. A. TAJMOUATI et Pr. A. OUDGHIRI pour les efforts qu'ils ont consentis pour que ma formation se déroule dans les meilleures conditions.

Puis, nous voulons transmettre nos remerciements à toutes les personnes qui nous ont assistées durant l'accomplissement de notre travail de recherche.

Enfin mes remerciements s'adressent également à toutes personnes ayant contribués directement ou indirectement à la mise en forme de ce travail, qu'elles trouvent ici l'expression de ma profonde reconnaissance.

Dédicaces

Je dédie ce mémoire

A mes chers parents ma mère et mon père

Pour leur patience, leur amour, leur soutien et leurs encouragements.

A mes frères.

A mes amies et mes camarades.

A toute personnes qui m'ont encouragé ou aidé au long de mes étude.

Nous dédions aussi ce travail à tous nos professeurs qui nous ont enseigné et à tous ceux qui nous sont chers.

TABLE DES MATIERES

Introduction générale

Dans un monde où les équations ne peuvent avoir des solutions analytiques par le calcul direct, intervient l'analyse et le calcul numérique qui permettent ajustement de palier cette lacune et de proposer une solution approchée au problème.

L'objectif de ce rapport est de donner une base de calcul avec un bref rappel de la théorie mais en insistant plus sur l'aspect pratique que le formalisme théorique. Afin de ne pas surcharger le document de théorie et d'interminables démonstrations, j'ai préféré la procédure suivante : poser le problème, rappeler les principes fondamentaux de la méthode numérique, et aborder directement des applications concrètes. Je n'ai néanmoins pas négligé l'aspect algorithmique qui est la base même d'une programmation exacte et réussie. C'est la raison pour laquelle j'ai proposé, avant chaque code en langage C du problème en explication, l'algorithme qui permet justement d'écrire le programme de façon lisible et compréhensible. Le code C est bien documenté avec des commentaires sur les lignes jugées essentielles au traitement d'instructions ou de boucles permettant l'exécution du programme. A travers ce document, le lecteur peut se rendre compte du côté applicatif qui est mis en exergue avec des problèmes classiques souvent rencontrés en électronique, en électromagnétisme, ou en physique de façon générale.

Finalement, ce document a pour vocation et objectif de donner au chercheur un code C qui a été testé et dont l'algorithme peut être adapté à d'autres problèmes en effectuant de simples modifications.

CHAPITRE 1 : DERIVATION ET INTERPOLATION POLYNOMIALE.

I. Introduction.

La dérivation et l'interpolation polynômiale sont très étroitement reliés puisqu'ils tendent à répondre à diverses facettes d'un même problème. Ce problème est le suivant :

A partir d'une fonction $f(x)$ connue seulement en $(n+1)$ points de la forme $(x_i, f(x_i))$ pour $i = 0,1,\dots,n$, peut-on construire une approximation de $f(x)$ pour une valeur de x différent de x_i ? Les points x_i sont appelés points de collocation ou points d'interpolation et peuvent provenir de données expérimentales. La Figure 1 résume la situation.

II. Dérivation.

Si f est une fonction dérivable sur [a, b], la dérivée en $c\epsilon]a,b[$ est définie par :

$$f'(c) = \lim_{h\to 0} \frac{\Delta f(c)}{h} \quad où\ \Delta f(c) = f(c+h) - f(c)$$

La dérivation numérique nous permet de trouver une estimation de la dérivée ou de la pente d'une fonction, en utilisant seulement un ensemble discret de point.

II.1 Dérivée première.

Soit f une fonction connue seulement par sa valeur en $(n+1)$ points donnés $x_i, i = 0,\dots,n$ distincts.

Les formules de différences classiques peuvent être trouvées en utilisant la formule de Taylor.

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(\eta) \qquad x \le \eta \le x+h$$

- Formule progressive : $h = x_{i+1} - x_i$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{h}{2} f''(\eta)\,,\ \ x_i \le \eta \le x_{i+1}$$

L'erreur est $\frac{h}{2} f''(\eta)$ donc en $\theta(h)$.

- Formule régressive : $h = x_i - x_{i-1}$

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} - \frac{h}{2} f''(\eta)\,,\ x_{i-1} \le \eta \le x_i$$

La formule de différence centrale de la dérivée en x_i peut être trouvée en utilisant la formule de Taylor d'ordre 3 avec $h = x_{i+1} - x_i = x_i - x_{i-1}$

$$f(x_{i+1}) = f(x_i) + hf'(x_i) + \frac{h^2}{2} f''(x_i) + \frac{h^3}{3!} f'''(\eta_1)\ ,\quad x_i \le \eta_1 \le x_{i+1}$$

$$f(x_{i-1}) = f(x_i) - hf'(x_i) + \frac{h^2}{2} f''(x_i) - \frac{h^3}{3!} f'''(\eta_2)\ ,\quad x_{i-1} \le \eta_2 \le x_i$$

Si on suppose que f''' est continue sur $[x_{i-1}, x_{i+1}]$ on peut écrire la formule suivante :

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{h^2}{6} f'''(\eta)\,,\ x_{i-1} \le \eta \le x_{i+1}$$

L'erreur est $\frac{h^2}{6} f'''(\eta)$ donc en $\theta(h^2)$.

On peut interpoler les données par un polynôme au lieu d'utiliser la droite, nous obtenons alors les formules de différence qui utilisent plus de deux points. On suppose que le pas h est constant.

Formule de différence progressive utilisant 3 points :

Appliquons le développement de Taylor au point x_{i+1} et x_{i+2} :

$$(*)\quad f(x_{i+2}) = f(x_i) + 2hf'(x_i) + \frac{4h^2}{2} f''(x_i) + \frac{8h^3}{6} f'''(\eta_1)\ ,\quad x_{i+1} \le \eta_1 \le x_{i+2}$$

$$(**)\quad f(x_{i+1}) = f(x_i) + hf'(x_i) + \frac{h^2}{2} f''(x_i) + \frac{h^3}{6} f'''(\eta_2)\ ,\quad x_i \le \eta_2 \le x_{i+1}$$

$$(*) - 4(**) \quad \Rightarrow \quad f'(x_i) \approx \frac{-f(x_{i+2}) + 4f(x_{i+1}) - 3f(x_i)}{x_{i+2} - x_i}$$

Formule de différence régressive utilisant 3 points :

$$f'(x_i) \approx \frac{3f(x_i) - 4f(x_{i+1}) + f(x_{i-2})}{x_i - x_{i-2}}$$

II.2 Dérivées d'ordre supérieur.

Les formules de dérivées d'ordre supérieur, peuvent être trouvées à partir des dérivées d'ordre du polynôme de Lagrange ou en utilisant les formules de Taylor.

Par exemple, étant donné 3 points x_{i-1}, x_i, x_{i+1} équidistants, la formule de la dérivée seconde est donnée par :

$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2}$$

L'erreur est en $\theta(h^2)$.

Pour obtenir les formules de la 3ème et la 4ème dérivée, on prend une combinaison linéaire de développement de Taylor, pour $f(x+2h), f(x+h), f(x-h)\ et\ f(x-2h)$.

III. Interpolation polynômiale.

Nous abordons dans ce paragraphe un nouveau type de problème, faisant intervenir la notion d'approximation d'une fonction.

Cette notion a déjà été rencontrée dans les cours d'analyse.

III.1 Méthode d'interpolation de Lagrange.

Soit une fonction f (connue explicitement)

- On choisit $(n+1)$ points $x_0, x_1, \dots, x_n$.
- On évaluable par un calcul coûteux.

Principe :

- Représenter f par une fonction simple, facile à évaluer.

Problème :

- Il existe une infinité de solution !

Le problème d'interpolation consiste donc à déterminer l'unique polynôme de degré n passant par $(n+1)$ points donnés d'interpolation $(x_i, f(x_i))$ pour $i = 0,1,\dots,n$.

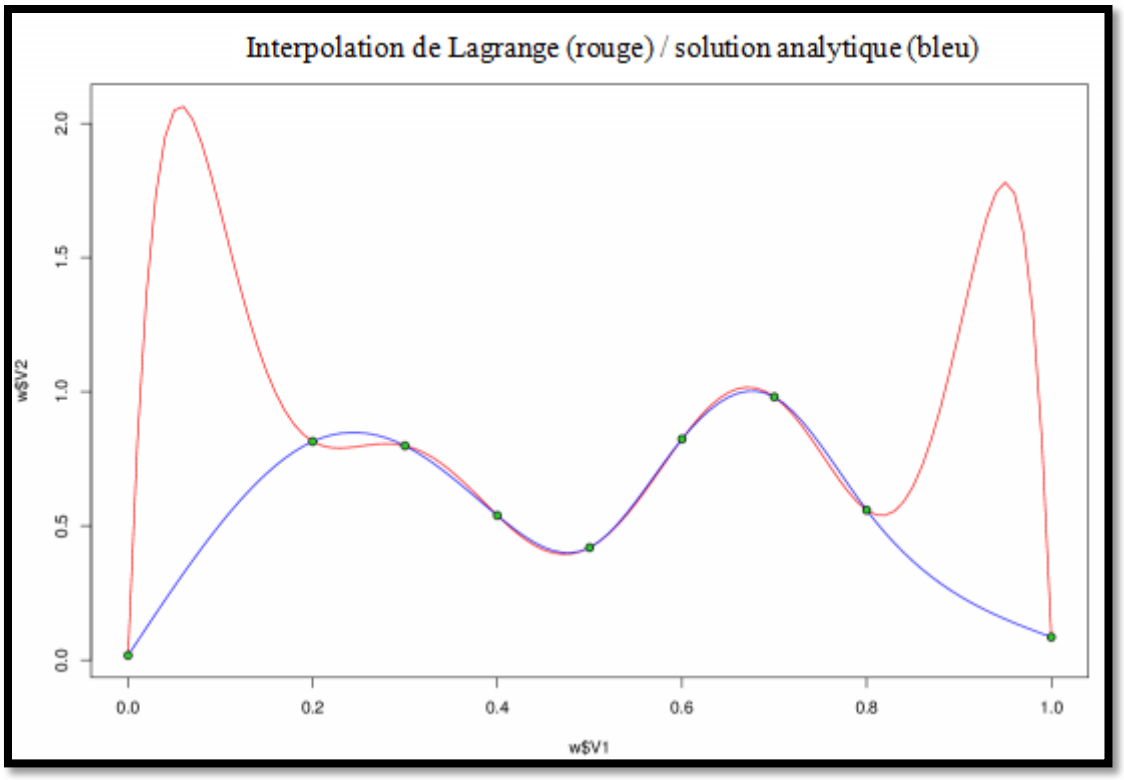


Figure 1 : Méthodes d'interpolation de Lagrange.

- On calcul $y_0 = f(x_0)$, $y_1 = f(x_1)$, … , $y_n = f(x_n)$.
- On cherche un polynôme de degré n tel que $P_n(x_i) = y_i$, $i = 0, \dots, n$ on introduits les coefficients d'interpolation de Lagrange.

$$L_k(x) = \frac{(x-x_0)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)} = \prod_{\substack{j=0 \\ j\neq k}}^{n} \frac{(x-x_j)}{(x_k-x_j)}$$

avec L_k est un polynôme de degré n,

$$L_k(x_i) = \begin{cases} 0 & si\ \ i \neq k \\ 1 & si\ \ i = k \end{cases}$$

Donc, $$P(x) = y_0 L_0(x) + y_1 L_1(x) + \cdots + y_n L_n(x)$$

$$= \sum_{k=0}^{n} y_k L_k(x)$$

est un polynôme de degré n qui vérifie bien $P(x_i) = y_i$.

Algorithme de la méthode de Lagrange

Fonction $b = Lagrange(x[i], y[j], a)$

Pour $i = 1$ **jusqu'**à n

Pour $j = 1$ **jusqu'**à $n, j \neq i$ **;**

$\mathbf{L} \leftarrow \mathbf{L} * \frac{a - x(i)}{x(i) - x(j)}$

fait

som ← som + L * y(i)

fin

fin.

III.2 Méthode d'interpolation de Newton.

Lorsqu'on écrit l'expression générale d'un polynôme, on pense immédiatement à la forme standard, qui est la plus utilisée. Il en existe cependant d'autres qui sont plus appropriées au cas de l'interpolation, par exemple :

$$P_n(x) = a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0)(x - x_1)(x - x_2) \ldots (x - x_{n-1})$$

$$= \sum_{j=0}^{n} a_j N_j(x) \;\; où \;\; N_j(x) = \prod_{i=0}^{j-1} (x - x_i), j \geq 1, avec \; N_0(x) = 1$$

L'aspect intéressant de cette formule apparaît lorsqu'on essaie de déterminer les $(n+1)$ coefficients a_i de telle sorte que $P_n(x)$ passe par les $(n+1)$ points de collocation $(x_i, f(x_i))$ pour $i = 0,1, \ldots, n$. On doit donc s'assurer que :

$$P_n(x_i) = f(x_i) \quad pour \quad i = 0,1, \ldots, n$$

En posant

$$a_n = \sum_{k=0}^{n} \frac{f(x_k)}{(x_k - x_0)(x_k - x_1) \ldots (x_k - x_n)}$$

Les coefficients a_n sont calculés en utilisant les (x_i, y_i) directement et aussi avec l'opérateur progressive Δ.

On a : $a_0 = y_0 = \Delta_0^0$

$$a_1 = \frac{y_1 - y_0}{h} = \frac{\Delta_0^1}{h}$$

On généralise, on trouve : $a_n = \frac{\Delta_0^n}{n! h^n}$

avec l'opérateur progressif est calculé par la formule

$$\Delta_0^n = \sum_{k=0}^{n} (-1)^k \binom{n}{k} y_{n-k}$$

Où $\binom{n}{k}$ désigne la combinaison de k parmi n calculé par :

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

Donc la formule générale de Neuton progressive écrit sous la forme :

$$P_n(x) = y_0 + \frac{\Delta_0^1}{h}(x - x_0) + \cdots + \frac{\Delta_0^n}{h}(x - x_0)(x - x_1) \ldots (x - x_{n-1})$$

IV. Conclusion.

L'interpolation polynomiale donc est une technique d'interpoler un ensemble de données ou d'une fonction par un polynôme. En d'autres termes, étant donné un ensemble de points (obtenu, par exemple, à la suite d'une expérience), on cherche un polynôme qui passe par tous ces points, en appliquant la méthode de Lagrange ou de Newton.

CHAPITRE 2 : INTEGRATION NUMERIQUE DES FONCTIONS.

I. Introduction.

Dans ce chapitre, nous introduisons des *formules de quadratiques*, qui consistent à approcher la valeur de l'intégrale par une somme pondérée finie de valeurs de la fonction f en des points choisis ; c'est le principe de l'intégrale de Riemann.

Si f une fonction continue sur $[a,b]$, l'intégrale de f sur $[a,b]$ est définie par :

$$\int_b^a f(x)dx = \lim_{h\to 0} R(h) \quad o\text{ù } R(h) = \sum_{k=1}^{n} f(a+kh)h.$$

$R(h)$ est la somme de Riemann avec $h = \frac{b-a}{n}$.

L'intégrale est pratiquement l'air représenté sous la courbe de la fonction, donc le calcul numérique de l'intégral réduit sous l'approximation de l'air en utilisant plusieurs méthodes (rectangle, trapèzes, Simpson …).

II. Méthodes de calcul numérique d'intégrale.

II.1 Principe.

Plutôt que d'augmenter le degré du polynôme d'interpolation, on peut obtenir une formule d'intégration en découpant l'intervalle d'intégration en sous-intervalles et en appliquant des formules simples sur chacun des sous-intervalles.

II.2 Méthode des rectangles.

La méthode des rectangles vient pour poser le changement qui remplace chaque arc de la courbe de la fonction f par des segments horizontaux sur un petit intervalle $[x_n, x_{n+1}]$ en d'autres mots, revient à une approximation de f par une fonction en escalier, avec n « marches » de longueur h.

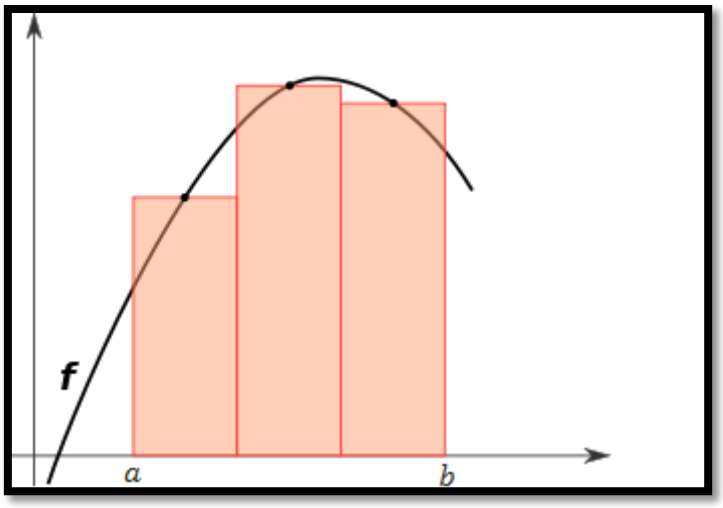


Figure 2 : Méthode des rectangles.

On peut faire varier la taille des marches. Toujours avec n marches, en prenant des x_i pour i entre 0 et n, avec l'intervalle de subdivision est $[a, b]$, et en calculant la fonction au milieu des rectangles, l'intégrale vaut :

$$I(f) = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx = \sum_{i=0}^{n-1} (x_{i+1} - x_i) f\left(x_i + \frac{h}{2}\right) \quad ou \quad h = \frac{b-a}{n} \quad et\ x_i = a + ih$$

Algorithme de la Méthode des Rectangles

Données : $f(x), a, b$
L'utilisateur défini la précision souhaitée et donc N
On définit un Tableau $x[\,]$ de dimension N
Calcul : $h = (b-a)/N$
$Somme = 0.0$
Pour $i = 1$ à N
{
$x[i] = a + i * h$
$Somme = Somme + h * f(x[i] + h/2)$
}
afficher Somme
fin.

II.3 Méthode des trapèzes.

on utilise une fonction continue affine par morceaux approchant la fonction à intégrer et égale à celle-ci sur les points de la subdivision en n sous-intervalles égaux de l'intervalle d'intégration $[a, b]$ pour obtenir une approximation de la valeur de son intégrale sur $[a, b]$.

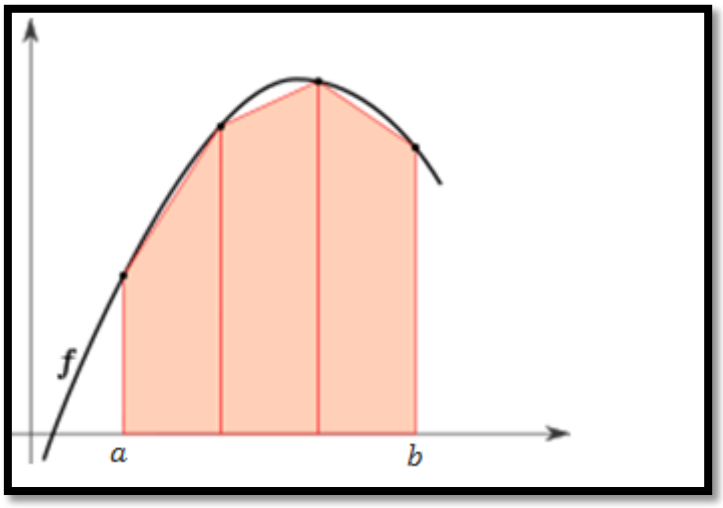


Figure 3 : Méthode des trapèzes.

La valeur approximative de l'intégrale correspond à l'aire sous la courbe de la fonction f.

Posons $h = \frac{b-a}{n}$, $x_i = a + ih$

alors

$$I(f) = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx = \sum_{i=0}^{n-1} \frac{h}{2}(f(x_i) + f(x_{i+1}))$$

II.4 Méthode de Simpson.

En se basant sur les autres méthodes, la méthode de Simpson de calcul numérique d'intégrale se repose sur le faite d'interpolé la fonction en 3 points qui résulte un polynôme de degré 2, qui est une parabole approché de la fonction sur le morceau $[x_i, x_{i+1}]$ pour avoir un polynôme approcher de la fonction sur l'intervalle $[a, b]$.

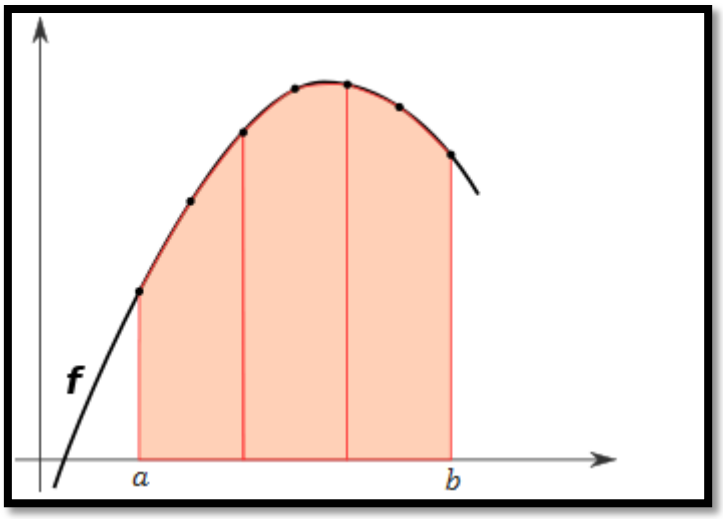


Figure 4 : Méthode de Simpson.

On obtient alors une valeur approchée de I avec la formule suivante :

$$I(f) = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx = \sum_{i=0}^{n} \frac{h}{6}\left[f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1})\right]$$

$$où \quad h = \frac{b-a}{n} \quad et \quad x_i = a + ih$$

III. Conclusion.

Les études simples de ces trois méthodes donnent une mauvaise approximation de la valeur exacte de l'intégrale, donc pour améliorer la précision on va utiliser une stratégie de travail qui consiste à décomposer ces méthodes comme ce qu'on a fait pour diminuer l'erreur commise.

CHAPITRE 3 : RESOLUTION NUMERIQUE DES EQUATIONS.

I. L'équation f(x)=0.

I.1 Introduction.

Nous nous intéressons dans ce chapitre à l'approximation des zéros (ou racines dans le cas d'un polynôme) d'une fonction réelle d'un variable réelle, c'est-à-dire, étant donnés un intervalle $I \subseteq \mathbb{R}$ et une application f de I dans $\mathbb{R}$, la résolution approchée du problème : trouver $\alpha \epsilon \mathbb{R}$ (ou plus généralement $\mathbb{C}$) tel que

$$f(\alpha) = 0.$$

Ce problème intervient notamment dans l'étude générale de fonctions d'une variable réelle, qu'elle soit motivée ou non par des applications, pour lesquelles des solutions exactes de ce type d'équation ne sont pas connues.

Toutes les méthodes que nous allons présenter solutions sont itératives et consistent donc en la construction d'une suite de réels $(x_k)_{k\epsilon\mathbb{N}}$ qui, on l'espère, sera telle que

$$\lim_{k\to+\infty} x_k = \alpha.$$

I.2 Méthode de Dichotomie.

I.2.1 Principe.

Considérons une fonction f continue sur un intervalle $[a, b]$. On suppose que f admet une et une seule racine α dans $]a, b[$ et que $f(a)\, f(b) < 0$. On note $c = \frac{a+b}{2}$ le milieu de l'intervalle.

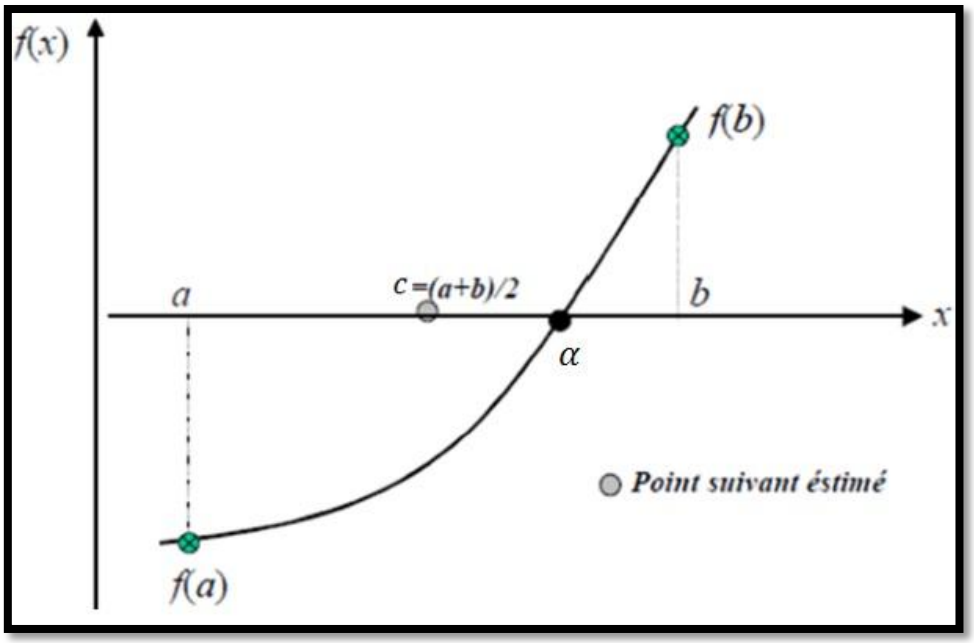


Figure 5 : Méthode de dichotomie.

1. Si $f(c) = 0$, c'est la racine de f et le problème est résolu.
2. Si $f(c) \neq 0$, nous regardons le signe de $f(a)\,f(c)$.
    1. Si $f(a)\,f(c) < 0$, alors $\alpha \in]a, c[$
    2. Si $f(a)\,f(c) > 0$, alors $\alpha \in]c, b[$

On recommence le processus en prenant l'intervalle $[a, c]$ au lieu de $[a, b]$ dans le premier cas, et l'intervalle $[c, b]$ au lieu de $[a, b]$ dans le second cas. De cette manière, on construit par récurrence sur n trois suites (a_n), (b_n) et (c_n) telles que $a_0 = a$, $b_0 = b$ et telles que pour tout $n \geq 0$,

1. $c_n = \frac{a_n + b_n}{2}$
2. Si $f(c_n)\,f(b_n) < 0$ alors $a_{n+1} = c_n$ et $b_{n+1} = b_n$
3. Si $f(c_n).f(a_n) < 0$ alors $a_{n+1} = a_n$ et $b_{n+1} = c_n$

I.2.2 Etude de la convergence.

Soit f une fonction continue sur $[a, b]$, vérifiant $f(a)\,f(b) < 0$ et soit $\alpha \in [a, b]$ l'unique solution de l'équation $f(x) = 0$. Si l'algorithme de dichotomie arrive jusqu'à l'étape n alors on a l'estimation :

$$e_n = |c_n - \alpha| \leq \frac{b_n - a_n}{2} = \frac{b - a}{2^{n+1}}.$$

ce qui entraîne $\lim_{n\to\infty} e_n = 0.$

Par conséquent, la suite (c_n) converge vers α c'est aussi vrai si $(c_n) = \alpha$.

I.2.3 Test d'arrêt.

Pour que la valeur (c_n) de la suite à la nième itération soit une valeur approchée de α à $\varepsilon>0$ près, il suffit que n vérifie :

$$\frac{b-a}{2^{n+1}} \leq \varepsilon$$

Alors on : $$|c_n - \alpha| \leq \frac{b-a}{2^{n+1}} \leq \varepsilon$$

Ce qui permet de calculer à l'avance le nombre maximal $n_0 \in \mathbb{N}$ d'itérations assurant la précision.

$$\frac{b-a}{2^{n+1}} \leq \varepsilon \quad \Leftrightarrow \quad \frac{b-a}{\varepsilon} \leq 2^{n+1}$$

Algorithme de la méthode de Dichotomie

Initialisation

$\boldsymbol{a(0) = a,}$

$\boldsymbol{b(0) = b, et}$

$$\boldsymbol{c(0) = [a(0) + b(0)]/2}$$

Boucle

Pour $\boldsymbol{k \geq 0}$ et tant que = $\boldsymbol{|b(k) - a(k)| > \varepsilon}$

- **Si** $\boldsymbol{f(x(k))f(a(k)) < 0}$
  - $\boldsymbol{a(k+1) = a(k), b(k+1) = x(k)}$
- **Si** $\boldsymbol{f(x(k))f(b(k)) < 0}$
  - $\boldsymbol{a(k+1) = x(k), b(k+1) = b(k)}$
- $\boldsymbol{x(k+1) = a(k+1) + b(k+1)}$

afficher $\boldsymbol{x(k)}$

fin.

I.3 Méthode de Point fixe.

I.3.1 Principe.

Le principe de cette méthode consiste à transformer l'équation $f(x)=0$ en une équation équivalente $g(x) = x$ où g est une fonction auxiliaire ''bien'' choisie. Le point α est alors point fixe de g. Approcher les zéros de f revient à approcher les points fixes de g. Le choix de la fonction g est motivé par les exigences du théorème de point fixe. En effet, elle doit être contrainte dans un voisinage de α, ce qui revient à vérifier que $|g'(x)| < 1$ sur ce voisinage. Dans ce cas on construit une suite $(x_n)_{n\epsilon\mathbb{N}}$ définie par :

$$\begin{cases} x_0 \text{ dans un voisinage I de } \alpha \\ \forall\, n \geq 0, x_{n+1} = g(x_n) \end{cases}$$

Il ne reste plus qu'à appliquer localement le théorème de point fixe pour démontrer que :

$$\alpha = \lim_{n\to+\infty} x_n$$

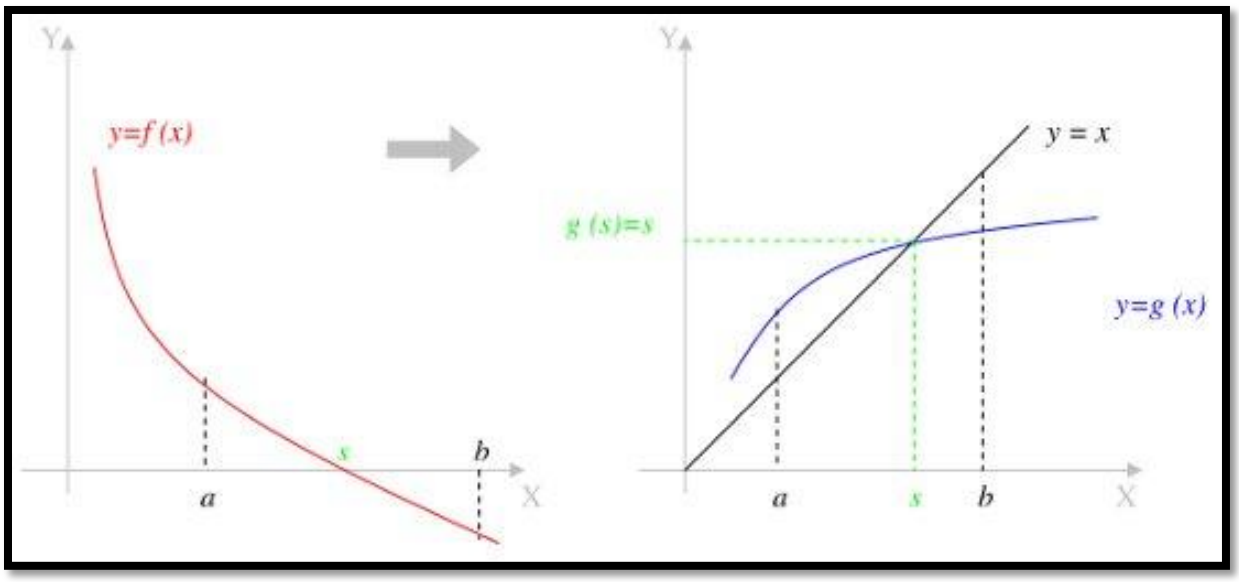


Figure 6 : Méthode de point fixe

I.3.2 Etude de la convergence.

On définit l'erreur à la nième itération par : $e_n = x_n - \alpha$

Pour avoir la convergence, il faut que $e_n \to 0$ quand n tend vers l'infinie.

Soit α un point fixe de g.

D'après le développement de Taylor-Lagrange de g en α :

$$g(x_n) = g(\alpha) + \sum_{K=1}^{m-1} \frac{(x_n - \alpha)^k}{k\,!}\ g^{(k)}(\alpha) + \frac{(x_n - \alpha)^m}{m!} g^{(m)}(c).$$

$$g(x_n) - g(\alpha) = x_{n+1} - \alpha = \sum_{K=1}^{m-1} \frac{(x_n - \alpha)^k}{k\,!}\ g^{(k)}(\alpha) + \frac{(x_n - \alpha)^m}{m!} g^{(m)}(c).$$

Or $e_n = x_n - \alpha$

$$\Rightarrow \quad e_{n+1} = \sum_{K=1}^{m-1} \frac{e_n^k}{k\,!}\ g^{(k)}(\alpha) + \frac{e_n^m}{m!} g^{(m)}(c).$$

Si $g'(\alpha) \neq 0$ et si on néglige les termes supérieurs ou égaux à 2, c.-à-d. :

$g''(\alpha) = \ \ldots\ = g^{(m-1)}(\alpha) = 0 \ \ et \ \ g'(\alpha) \neq 0$ (1)

On obtient :

$$e_{n+1} \approx g'(\alpha) e_n \qquad \Rightarrow \qquad \frac{e_{n+1}}{e_n} \approx g'(\alpha)$$

Finalement, $\dfrac{|e_{n+1}|}{|e_n|} \approx |g'(\alpha)|$

La suite (x_n) est convergente et à convergence linéaire.

Théorème de point fixe.

Soit $\boldsymbol{g} : \mathbb{R} \to \mathbb{R}$ de classe $\mathcal{C}^{\boldsymbol{n}}$. On suppose que $\boldsymbol{g}$ admet un unique point fixe $\alpha \in [a, b]$ vérifiant :

1) $\forall \boldsymbol{x} \in [\boldsymbol{a}, \boldsymbol{b}]$, $\boldsymbol{g}(\boldsymbol{x}) \in [\boldsymbol{a}, \boldsymbol{b}]$.
2) $\boldsymbol{g}$ est une contraction stricte dans $[\boldsymbol{a}, \boldsymbol{b}]$.

i.e. $\exists \boldsymbol{k} \in]\boldsymbol{0}, \boldsymbol{1}[\ , \forall (\boldsymbol{x}, \boldsymbol{y}) \in [\boldsymbol{a}, \boldsymbol{b}] \quad |\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{y})| \leq \boldsymbol{k}|\boldsymbol{x} - \boldsymbol{y}|$

Il existe alors un voisinage $\boldsymbol{V_\alpha}$dans $\boldsymbol{I}$ tel que la suite $(\boldsymbol{x_n})_{\boldsymbol{n} \in \mathbb{N}}$ définie par :

$$\begin{cases} x_0 \in V_\alpha \\ \forall n \in \mathbb{N}, x_{n+1} = g(x_n) \end{cases}$$

Converge vers $\boldsymbol{\alpha}$. De plus on a l'estimation de l'erreur :

$$\boldsymbol{e_n = |x_n - \alpha| \leq \frac{k^n}{1-k}|x_1 - x_0| < \varepsilon}$$

Définition :

Soit $g : I = [a, b] \rightarrow [a, b]$ de classe $\mathcal{C}^n$. On suppose que g admet un point fixe $\alpha \epsilon [a, b]$ tel que :

$$\begin{cases} x_0 \ donné \\ x_{n+1} = g(x_n) \end{cases}$$

- Si $|g'(x)| < 1 \Rightarrow$ le point fixe α est attractif et la suite (x_n) converge.
- Si $|g'(x)| > 1 \Rightarrow$ le point fixe α est répulsif et la suite (x_n) ne converge pas.

➢ *Algorithme de la méthode de Dichotomie*

•Données : $\boldsymbol{g(x)}$, $\boldsymbol{\varepsilon}$**(précision voulue),** $\boldsymbol{x(0)}$
Etape 1 : Calcul de $\boldsymbol{x(k+1) = g(x(k))}$ pour $\boldsymbol{k \geq 0}$
Etape 2 : Si $\boldsymbol{|x(k+1) - x(k)| < \varepsilon}$
Ecrire « la solution est $\boldsymbol{x(k+1)}$ **» et Sortir du programme**
Sinon
$\boldsymbol{k = k+1}$ **et reprendre l'étape à 1**

I.4 Méthode de la Sécante.

I.4.1 Principe.

On suppose que f est continue et 2 fois dérivables sur [a, b], avec $f(a)f(b) < 0$ et f' et f'' ont un signe constant sur [a, b].
Soient $A(a, f(a))$ et $B(b, f(b))$ avec c l'abscisse du point d'intersection de la corde AB et de l'axe ox est une valeur approché de la racine α.

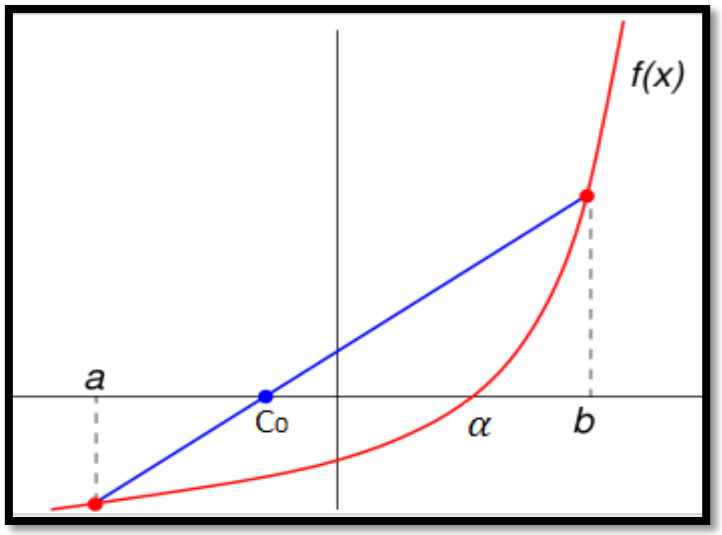


Figure 7 : Méthode de la Sécante.

L'équation de la droite passant par les points A et B :

$$y = \frac{f(b) - f(a)}{b - a}(x - a) + f(a)$$

Au point $x = c_0$, on a : $0 = \frac{f(b)-f(a)}{b-a}(c_0 - a) + f(a)$

$$\Rightarrow \quad c_0 = \frac{af(b)-bf(a)}{f(b)-f(a)}$$

Par conséquent, si $f'f'' < 0$ la valeur approchée obtenue est une valeur approchée par excès et si $f'f'' > 0$ la valeur approchée obtenue est une valeur approchée par défaut.

Itération de la méthode :

La méthode de la corde donne pour $a < \alpha < c_0$ si $f'f'' < 0$ respectivement $c_0 < \alpha < b$ si $f'f'' > 0$.

Il est possible d'appliquer à nouveau la méthode de la corde sur l'intervalle $[a, c_0]$

(respectivement $[c_0, b]$), nous construisons ainsi une suite de valeur approchées (c_n) par excès (respectivement par défaut) de α.

La suite (c_n) est définie par :

$$\begin{cases} c_0 = b \\ \forall n \epsilon \mathbb{N}, \quad c_{n+1} = \dfrac{af(c_n) - c_n f(a)}{f(c_n) - f(a)} \end{cases}$$

Respectivement

$$\begin{cases} c_0 = a \\ \forall n \epsilon \mathbb{N}, \quad c_{n+1} = \dfrac{c_n f(b) - bf(c_n)}{f(b) - f(c_n)} \end{cases}$$

L'avantage de cette méthode est qu'elle ne nécessite pas le calcul de la dérivée *f'*. Mais l'inconvénient est que la convergence n'est plus quadratique.

I.4.2 Etude de la convergence.

Théorème.

Soit $\boldsymbol{f} : [\boldsymbol{a}, \boldsymbol{b}] \to \mathbb{R}$ de classe $\mathcal{C}^{\boldsymbol{2}}$telle que $\boldsymbol{f}'$ et $\boldsymbol{f}''$soient strictement positives sur $[\boldsymbol{a}, \boldsymbol{b}]$.

On suppose que :

$$\boldsymbol{f(a) < 0 \, , f(b) > 0}$$

et on appelle $\boldsymbol{\alpha}$ l'unique solution de l'équation $\boldsymbol{f(x) = 0}$. Alors :

1. La suite $(\boldsymbol{c_n})$ telle que :

$$\begin{cases} \boldsymbol{c_0 = b} \\ \boldsymbol{\forall n \epsilon \mathbb{N}, \quad c_{n+1} = \dfrac{c_n f(b) - b f(c_n)}{f(b) - f(c_n)}} \end{cases}$$

est bien définie.

2. La suite $(\boldsymbol{c_n})$ est croissante et converge vers α.
3. La méthode de la sécante est au moins d'ordre 1 :

$$\boldsymbol{\lim_{n \to +\infty} \frac{|c_{n+1} - \alpha|}{|c_n - \alpha|} = \left| 1 + (\alpha - b) \frac{f'(\alpha)}{f(b)} \right|}$$

Preuve.

On pose $c_{n+1} = g(c_n)$ où $g(x) = \frac{xf(b) - bf(x)}{f(b) - f(x)}$.

La fonction f est strictement convexe : sa courbe est en dessous de tout segment reliant deux points de cette courbe. Donc f admet son unique zéro α dans l'intervalle $[a,b]$. Comme $f(a)<0$ et $f(b)>0$, le réel c_0 est l'abscisse de l'intersection de la droite passant par $(a,f(a))$ et $(b,f(b))$ et vérifie $f(c_1)<0$; de même, $f(c_2)<0$ et par récurrence on a

$$f(c_n)<0 \quad , \qquad \forall n \in N.$$

On vérifie que la suite (c_n) est croissante majorée par b, donc convergente. Comme $c_{n+1}=g(c_n)$ et que g est continue, la limite est l'unique point fixe de g. De plus,

$$\left|\frac{c_{n+1}-\alpha}{c_n-\alpha}\right| = \left|\frac{g(c_n)-g(\alpha)}{c_n-\alpha}\right| \rightarrow \left|g'(\alpha)\right| = \left|1+(\alpha-b)\frac{f'(\alpha)}{f(b)}\right|$$

La dernière égalité est obtenue en dérivant g au point α.

Algorithme de la méthode de la Sécante.

Initialisation: $a_0 = a\,, b_0 = b$ ***avec*** $f(a)f(b) < 0$
Itérations: Pour $k = 0,1,2,\ldots$

$$c_k = \frac{a_k f(b_k) - b_k f(a_k)}{f(b_k) - f(a_k)}$$

Si $f(a_k)f(c_k) < 0$
alors $a_{k+1} = a_k$ ***et*** $b_{k+1} = c_k$
sinon $a_{k+1} = c_k$ ***et*** $b_{k+1} = b_k$
fin du si
fin du pour

I.5 Méthode de Newton.

I.5.1 Principe.

On suppose que f est continue et deux fois dérivables sur $[a,b]$.
Soient $A(a,f(a))$, $B(b,f(b))$ avec t_0 l'abscisse du point d'intersection de la tangente en A (respectivement en B) au graphe de la fonction f et de l'axe ox est une valeur approchée de la racine α.

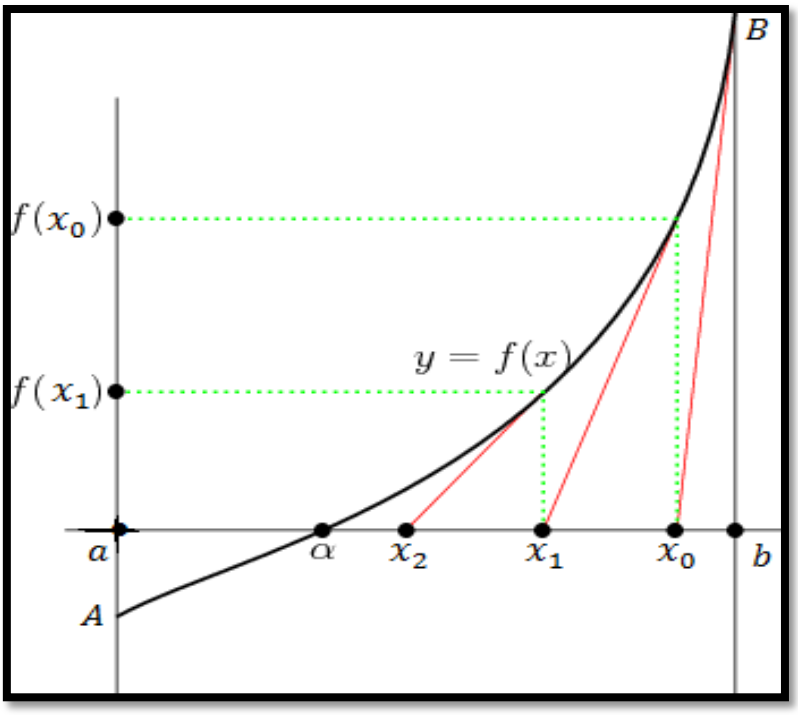


Figure 8 : Méthode de Newton.

Ecrivons l'équation de la tangente en B (respectivement en A) :

$$y = f'(b)(x - b) + f(b)$$

Respectivement $\quad y = f'(a)(x - a) + f(a)$

Pour $x = x_0$, on a :

$$0 = f'(b)(x_0 - b) + f(b)$$

$$\Rightarrow \quad x_0 = b - \frac{f(b)}{f'(b)} \quad \text{respectivement} \quad x_0 = a - \frac{f(a)}{f'(a)}.$$

Par conséquent, la méthode de Newton s'applique à l'extrémité où f et f'' sont de même signe.

$$\begin{cases} soit\ A\ si\ f(a)f''(a) > 0\ et\ dans\ ce\ cas\ x_0 < \alpha \\ soit\ B\ si\ f(b)f''(b) > 0\ et\ dans\ ce\ cas\ x_0 > \alpha \end{cases}$$

Comme précédemment il est possible d'appliquer à nouveau la méthode sur l'intervalle $[x_0, b]$ (respectivement $[a, x_0]$), nous construisons ainsi une suite (x_n) de valeur approchée de α par défaut (respectivement par excès).

La suite (x_n) est définie par :

$$\begin{cases} x_0 = b \\ \forall n \in \mathbb{N}, x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)} \end{cases}$$

Respectivement

$$\begin{cases} x_0 = a \\ \forall n \in \mathbb{N}, x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)} \end{cases}$$

I.5.2 Etude de la convergence.

Soit f est une fonction définie au voisinage de α et deux fois continument différentiable. On suppose que α se trouve être un zéro de f qu'on essaie d'approcher par la méthode de Newton. On fait l'hypothèse que α est un zéro d'ordre 1, autrement dit que $f'(\alpha)$ est non nul. *La formule de Taylor-Lagrange* s'écrit :

$$0 = f(\alpha) = f(x) + (\alpha - x)f'(x) + (a - x)^2 \frac{f''(c)}{2} \qquad \text{où} \quad c\epsilon]x, \alpha[$$

Partant de l'approximation x, la méthode de Newton fournit au bout d'une itération :

$$F(x) - \alpha = x - \frac{f(x)}{f'(x)} = \frac{f''(c)}{2f'(x)}(x - a)^2$$

Pour un intervalle compact I contenant x et α et inclus dans le domaine de définition de f, on pose :

$$m = \min_{x \in I} |f'(x)| \quad \text{ainsi que} \quad M = \min_{x \in I} |f''(x)|$$

Alors, pour tout $x\epsilon I$:

$$|F(x) - \alpha| \leq \frac{M}{2m}|x - a|^2$$

Par récurrence immédiate, il vient : $|x - \alpha| \leq \left(\frac{M}{2m}\right)^{2^n - 1} |x_0 - \alpha|^{2^n}$

La convergence de (x_n) vers α est donc quadratique.

Théorème de convergence globale.

Soit f est de classe $\mathcal{C}^2([\boldsymbol{a}, \boldsymbol{b}])$ vérifient :

1) $\boldsymbol{f(a)f(b) < 0}$
2) $\forall \boldsymbol{x\epsilon}[\boldsymbol{a}, \boldsymbol{b}]$, $\boldsymbol{f'(x) \neq 0}$ (strict monotone)
3) $\forall \boldsymbol{x\epsilon}[\boldsymbol{a}, \boldsymbol{b}]$, $\boldsymbol{f''(x) \neq 0}$ (concavité de f dans le même sens)

Alors en choisissant $\boldsymbol{x_0 \epsilon [a,b]}$ tel que $\boldsymbol{f(x_0)\, f''(x_0) > 0}$

Les itérations de Newton convergent vers l'unique solution α de $f(x) = 0$ dans $[a, b]$.

Algorithme de la méthode de Newton

Données : $\boldsymbol{N, x, , f(x_i)}$ ***et*** $\boldsymbol{f'(x_i)}$
Boucle
Pour $\boldsymbol{i = 1, N}$
On calculera :

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Si $|x_{i+1} - x_i| < \varepsilon$ ***alors*** x_{i+1} ***est une racine***
Sinon on continue les itérations (c-à-d. i = i+1)

II. Les équations différentielles ordinaires.

II.1 Introduction.

Une équation différentielle est une équation qui dépend d'une variable t et d'une fonction y(t) et qui contient des dérivées de (t) . Elle s'écrit :

$$F\left(t, y(t), y^{(1)}(t), \dots\dots, y^{(m)}(t)\right) = 0 \quad \text{où} \quad y^{(m)}(t) \equiv \frac{d^m y}{dt^m} \quad (1)$$

L'ordre de cette équation est déterminé par sa dérivée d'ordre le plus élevé. Donc l'équation (1) est d'ordre m.

La résolution numérique des équations différentielles est probablement le domaine de l'analyse numérique où les applications sont les plus nombreuses. Mais elles sont utilisées pour construire des modèles mathématiques de phénomènes physiques et biologiques, par exemple pour l'étude de la radioactivité ou la mécanique céleste. Par conséquent, les équations différentielles représentent un vaste champ d'étude, aussi bien en mathématiques pures qu'en mathématiques appliquées.
En analyse, il existe des procédés de résolution numérique pour les équations différentielles.

II.2 Généralités sur le problème de Cauchy.

II.2.1 Objectif.

On s'intéresse à la résolution numérique du problème de Cauchy qui consiste en la recherche d'une fonction $y(t)$ de classe C^1 définie dans $[t_0, t_0 + T]$ à valeurs réelles vérifiant :

$$\begin{cases} y'(t) = f\big(t, y(t)\big) \quad \forall t \epsilon [t_0, t_0 + T] \\ y(0) = 0 \ \ donné \ (condition\ de\ Cauchy) \end{cases}$$

Théorème d'existence et d'unicité :

On suppose que la fonction : $[\boldsymbol{t_0, t_0 + T}] \times \mathbb{R} \rightarrow \mathbb{R}$, continue par rapport à l'ensemble des deux variables et ***Lipchitzienne*** en $\boldsymbol{y}$ uniformément par rapport à $\boldsymbol{t}$.

i.e. $\exists \boldsymbol{k} > 0 \ \boldsymbol{tel\ que}\ \big|\boldsymbol{f}\big(\boldsymbol{t}, \boldsymbol{y_1}(\boldsymbol{t})\big) - \boldsymbol{f}(\boldsymbol{t}, \boldsymbol{y_2}(\boldsymbol{t}))\big| \leq \boldsymbol{k}|\boldsymbol{y_1}(\boldsymbol{t}) - \boldsymbol{y_2}(\boldsymbol{t})|$

Alors le problème de Cauchy admet une solution unique.

- Le principe de la méthode numérique consiste à subdiviser l'intervalle $[t_0, t_0 + T]$ en n intervalles de longueur $h = \frac{T}{n}$, alors pour chaque nœud t_i tel que $0 \leq i \leq n$. On cherche la valeur y_i qui approche $y(t_i)$.

  C'est-à-dire : $y(i) \simeq y(t_i)$

II.3 Méthode d'Euler.

II.3.1 Méthode d'Euler explicite.

La méthode d'Euler, autrement connue sous le nom de méthode aux différences finies, consiste à utiliser un développement de Taylor à l'ordre 1 pour la résolution de l'équation différentielle.

Soit le problème de Cauchy :

$$\begin{cases} y'(t) = f\big(t, y(t)\big) \quad \forall t \epsilon [t_0, t_0 + T] \\ y(0) = 0 \end{cases}$$

On subdivise l'intervalle $[t_0, t_0 + T]$ en n sous intervalles de même longueur h et appelons t_i les points de subdivision. Nous avons donc :

$t_i = t_{i-1} + h = t_0 + ih$ avec $h = {}^{T}/_{n} = pas\ de\ discrétisations\ et\ i = 0, \dots, n$

Le développement de Taylor de la fonction $y(t)$ jusqu'à l'ordre n autour du point t_n :

$$y(t_{n+1}) = y(t_n + h) = y(t_n) + hy'(t_n) + \cdots + \frac{h^n}{n!}y^{(n)}(t_n) + \theta(h^{n+1})$$

En tronquant le développement à l'ordre 1 : $y(t+h) = y(t) + hy'(t) + \theta(h^2)$

On trouve donc l'approximation suivante : $y(t+h) \approx y(t) + hf\big(t, y(t)\big)$.

Le problème de Cauchy discrétisé s'écrit :
$$\begin{cases} y_{i+1} = y_i + hf(t_i, y_i) \\ y(t_0) = y_0 \\ t_i = t_{i-1} + h = t_0 + ih \end{cases} \quad (1)$$

D'où (1) s'appelle méthode d'Euler explicite.

On peut aisément en donner une interprétation graphique

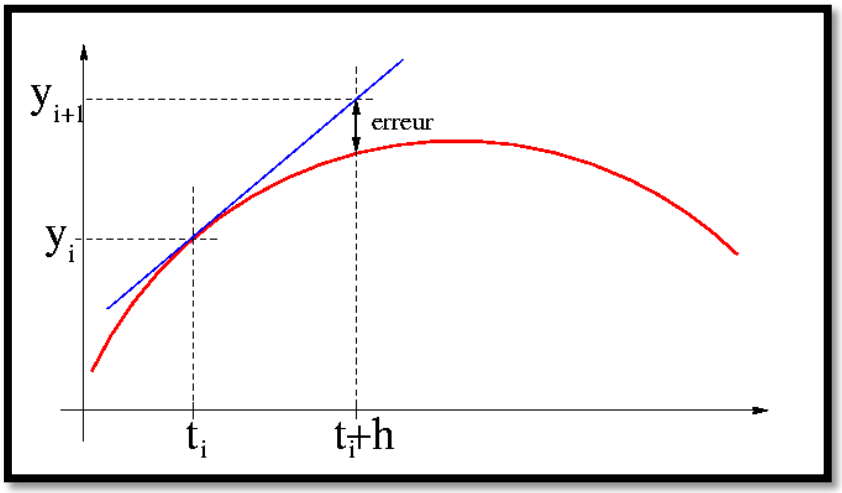


Figure 9 : Méthode d'Euler.

Connaissant y_n ; on calcul y_{n+1} comme étant l'ordonnée du point d'intersection de la droite $t = t_{n+1}$ avec la droite passant par le point (t_n, y_n) ayant pour pente $f(t_n, y_n)$ (i.e.) pente de la tangente en (t_n, y_n) à la courbe solution.

Algorithme de la Méthode d'Euler explicite

Données : $\boldsymbol{f(t, y(t)), a, b}$

Initialisation $\boldsymbol{t_0}$, $\boldsymbol{y_0}$

Le nombre d'itération maximale N
On définit un Tableau $t[\,]$ et $y[\,]$ de dimension N
Calcul : $h = (b-a)/N$
Pour $i = 0$ à N
{
$x[i] = a + i * h$
$y[i+1] = y[i] + h * f(t[i], y[i])$
}
afficher $y[i]$
fin.

II.3.2 Méthode d'Euler implicite.

Au lieu d'exprimer y_{i+1} à partir de y_i grâce au théorème de Taylor, on peut procéder autrement et exprimer y_i à partir de y_{i+1} selon la formule :

$$\begin{cases} y_{i+1} = y_i + hf(t_{i+1}, y_{i+1}) \quad \Rightarrow y_i = y_{i+1} - hf(t_{i+1}, y_{i+1}) \\ y(t_0) = y_0 \\ t_i = t_{i-1} + h = t_0 + ih \end{cases}$$

Le développement de Taylor de la fonction $y(t)$ jusqu'à l'ordre n autour du point t_n :

$$y(t_i) = y(t_{i+1} - h) = y(t_{i+1}) + hy'(t_{i+1}) + \cdots + \frac{h^n}{n!}y^{(n)}(t_{i+1}) + \theta(h^{n+1})$$

En tronquant le développement à l'ordre 1 : $y(t-h) = y(t) - hy'(t) + \theta(h^2)$

On trouve donc l'approximation suivante : $y(t-h) \approx y(t) - hf\big(t, y(t)\big)$.

Remarque :

L'erreur de troncature est définie entre la solution exacte $y(t_i)$ et l'approximation numérique obtenue y_i elle est évaluée par la relation :

$$\varepsilon = |y_i - y(t_i)|$$

II.4 Méthodes de Runge-Kutta.

L'idée fondamentale des méthodes de Runge-Kutta est d'intégrer l'équation $y'(t) = f(t, y)$ entre t_n et t_{n+1} et de calculer :

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

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

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

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

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

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

Ce de méthodes son d'ordre 2 par rapport à h.

Si on utilise la méthode de point milieu on trouve :

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

Si maintenant on approche $y_{n+\frac{1}{2}}$ par $y_{n+\frac{1}{2}} = y_n + \frac{1}{2}f(t_n, y_n)$ on trouve la méthode d'Euler modifiée :

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

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

$$y_{n+1} = y_n + \frac{h}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

Où les k_i son calculés comme suit :

$$k_1 = f(t_n, y_n) \quad k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \quad k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right)$$

$$k_4 = f(t_{n+1}, y_n + hk_3).$$

CHAPITRE 4 : RESOLUTION NUMERIQUE DES SYSTEMES *LINEAIRES*

I. Introduction.

Les systèmes d'équations algébriques jouent un rôle très important en ingénierie. On peut classer ces systèmes en deux grandes familles : les systèmes linéaires et les systèmes non linéaires.

Dans ce chapitre, on se propose d'étudier quelques méthodes permettant de résoudre les systèmes linéaires de n équations à n inconnues de la forme :

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

Nous noterons ces systèmes sous la forme matricielle $Ax = b$ (1.2)

Où A désigne la matrice des éléments $(aij), 1 \leq i, j \leq n$; b étant le vecteur des composantes $(b_1, \ldots, b_n)$ et x le vecteur inconnue des composantes $(x_1, \ldots\ldots, x_n)$.

La matrice A est dite régulière (ou non singulière) si $det\, A \neq 0$; on a existence et unicité de la solution x si et seulement si, la matrice associée au système linéaire est régulière.

Théoriquement, si A est non singulière, la solution est donnée par la *formule de Cramer* :

$x_i = \frac{\Delta_i}{\Delta}$, $i = 1, \ldots, n$ *où* $\Delta_i = \begin{vmatrix} & & i & & \\ a_{11} & \ldots & b_1 & \ldots & a_{1n} \\ \vdots & & \vdots & & \vdots \\ a_{n1} & \ldots & b_n & \ldots & a_{nn} \end{vmatrix}$ et $\Delta = det\, A$ (Δ_i est le déterminant dont la colonne i est remplacée par la colonne du second membre b).

Dans la pratique nous aurons à résoudre des systèmes d'ordres $n = 50,100, \ldots$.

Il est donc impossible de résoudre de tels systèmes par les formules de Cramer. Il faut donc développer des algorithmes alternatifs avec un coût raisonnable.

Dans les sections suivantes plusieurs méthodes sont analysées.

II. Résolution des systèmes triangulaire.

Nous pouvons remarquer tout de suite que si la matrice A du système (1.2) est triangulaire, sa résolution numérique est immédiate.

On suppose que la matrice A est régulière, donc aucun des $a_{ii}\ i = 1, \dots, n$ n'est nul.

Suivant ces cas, le système à résoudre est dit *système triangulaire supérieure ou inférieure.*

Si A est triangulaire supérieure on a :

$x_n = \frac{b_n}{a_n}$, et pour $i = n-1, \dots, 2, 1$

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

Cet algorithme est appelé *méthode de remontée*.

Si A est triangulaire inférieure on a :

$x_1 = \frac{b_1}{a_{11}}$, et pour $i = 2, 3, \dots, n$

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

Cet algorithme est appelé *méthode de descente*.

III. Méthodes directes.

Nous appelons méthode directe de résolution du système (1.2), une méthode qui aboutit à la solution du problème au bout d'un nombre fini d'opérations arithmétiques, fonction de la dimension du système. On peut alors en déduire le temps de calcul nécessaire à la résolution (qui peut être très long si n est grand). Les méthodes directe s'opposent sur ce point aux méthodes dites itératives, qui peuvent converger en quelques itérations, ou en un très nombre d'itérations ou même diverger, selon le cas.

III.1 Méthode de Gauss.

La méthode de Gauss transforme le système (2.1) $Ax = b$ en un système triangulaire équivalent à l'aide d'un algorithme d'élimination convenable. La résolution numérique du système triangulaire ne posera alors aucun problème.

Construction de la matrice de transformation P_1(matrice carré d'ordre n) :

$$P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ -\frac{a_{21}}{a_{11}} & 1 & 0 & & & \vdots \\ \vdots & & \ddots & \ddots & & \\ -\frac{a_{i1}}{a_{11}} & 0 & \cdots & 1 & \ddots & \vdots \\ \vdots & & & & \ddots & 0 \\ -\frac{a_{n1}}{a_{11}} & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$ avec le premier pivot d'élimination $a_{11} \neq 0$.

- Propriétés de la matrice P_1 :

  - les a_{ij} dans P_1 sont les a_{ij} dans la matrice A.

  - P_1 est triangulaire inférieure avec terme diagonaux égaux à 1.

  - $det(P_1) = 1$.

En multipliant le système (2.1) par la matrice P_1, nous obtenons alors un système de la forme :

$$A^{(1)}x = b^{(1)} \quad où \; A^{(1)} = \left(a_{ij}^{(1)}\right) \; et \; b^{(1)} = (b_j^{(1)})$$

Avec $$\begin{cases} a_{1j}^{(1)} = a_{1j} \;, \quad j = 1, \dots, n \\ a_{i1}^{(1)} = 0 \quad , \quad i = 2, \dots, n \\ a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j} \quad , \quad i,j=2,\dots,n \\ b_1^{(1)} = b_1 \\ b_i^{(1)} = b_i - \frac{a_{i1}}{a_{11}} b_i \quad , \; i = 2, \dots, n \end{cases}$$

Plus précisément on a :

$$A^{(1)} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ 0 & a_{22}^{(1)} & \cdots & a_{2j}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & a_{i2}^{(1)} & \cdots & a_{ij}^{(1)} & \cdots & a_{in}^{(1)} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nj}^{(1)} & \cdots & a_{nn}^{(1)} \end{pmatrix} \quad et \quad b^{(1)} = \begin{pmatrix} b_1 \\ b_2^{(1)} \\ \vdots \\ b_i^{(1)} \\ \vdots \\ b_n^{(1)} \end{pmatrix}$$

On peut alors remarquer que les composantes du vecteur x n'interviennent pas dans cette élimination, et que les opérations effectuées sur les lignes de la matrice A et sur les composantes de b sont les mêmes.

Plus généralement, la méthode de Gauss consiste donc à construire une suite de systèmes équivalents à (2.1) de la forme $A^{(k-1)}x = b^{(k-1)}$ (2.2)

Où la matrice $A^{(k-1)}$ du système finale est triangulaire.

Construction de la matrice de la $k-ième$ transformation $\boldsymbol{P_k}$:

$$\boldsymbol{P_k} = \begin{bmatrix} 1 & 0 & & \cdots & & 0 \\ & 1 & & & & \\ 0 & & \ddots & \ddots & & \\ & & & 1 & & \vdots \\ \vdots & & & -\dfrac{a_{k+1,k}^{(k-1)}}{a_{kk}^{(k-1)}} & \ddots & \\ & & & \vdots & & 0 \\ 0 & & 0 & \dfrac{a_{nk}^{(k-1)}}{a_{kk}^{(k-1)}} & & 1 \end{bmatrix}$$ avec le $k-ième$ pivot d'élimination $a_{kk}^{(k-1)} \neq 0$.

- Les propriétés de P_k :

  - les termes $a_{ij}^{(k-1)}$ dans P_k sont les termes $a_{ij}^{(k-1)}$ de la matrice $A^{(k-1)}$, avec $A^{(k-1)} = \boldsymbol{P_{k-1}P_{k-2}} \ldots \boldsymbol{P_1}A$.

  -$\boldsymbol{P_k}$ est triangulaire inférieure avec ses termes diagonaux sont égaux à 1.

  -$det(\boldsymbol{P_k}) = 1$.

En multipliant le système (2.2) par la matrice P_1, nous obtenons alors un système de la forme :

$$A^{(k)}x = b^{(k)} \quad où \ A^{(k)} = \left(a_{ij}^{(k)}\right) \ \ et \ \ b^{(k)} = (b_j^{(k)})$$

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

À la *nième* étape, la matrice $A^{(n)}$ est triangulaire supérieure et le système $A^{(n)}x = b^{(n)}$ est résolu par les méthodes de descente où de remontée.

Il peut arriver que lors de l'élimination, dans l'ordre naturel des lignes, un pivot soit nul. En outre, pour des raisons de stabilité, on a intérêt à chaque étape de l'élimination à choisir le pivot tel que $\left|a_{kk}^{(k-1)}\right|$ soit le plus grand possible.

Pour cela deux stratégies sont possibles.

III.1.1 Stratégie de pivot partiel.

Le pivot est l'un des éléments $a_{ik}^{(k)}$

$k \leq i \leq n$ vérifiant :

$$\left|a_{ik}^{(k)}\right| = \max_{k \leq l \leq n} \left|a_{ik}^{(k)}\right| \text{ (c'est le plus grand en module de la colonne } k\text{).}$$

III.1.2 Stratégie de pivot total.

Le pivot est l'un des éléments $a_{ij}^{(k)}$

$k \leq i,j \leq n$ vérifiant :

$$\left|a_{ij}^{(k)}\right| = \max_{k \leq p,q \leq n} \left|a_{pq}^{(k)}\right|.$$

Si le pivot choisi n'est pas dans la colonne k ; il faut effectuer un échange de colonne (en plus d'un échange de lignes).

Algorithme de la Méthode de Gauss pivot partiel

début
  pour k **allant de** 1 **à** $n-1$ **faire**
    ref $\leftarrow 0$
      Pour i **allant de** $k+1$ **à** n **faire**
        ref $\leftarrow |A(i,k)|$
        ligne $\leftarrow i$
      pour j **allant de** $k+1$ **à** n **faire**
        temp $\leftarrow A(k,j)$
        $A(k,j) \leftarrow A(ligne,j)$
        $A(ligne,j) \leftarrow$ **temp**
    temp $\leftarrow b(k)$
    $b(k) \leftarrow b(ligne)$
    $b(ligne) \leftarrow$ **temp**
    pour i **allant de** $k+2$ **à** n **faire**
      $p = A(i,k)/A(k,k)$
      pour j **allant de** $k+1$ **à** n **faire**
        $A(i,j) \leftarrow A(i,j) - p * A(k,j)$
      $b(i) \leftarrow b(i) - p * b(k)$
  Pour i **allant de** n **à** 1 **faire**
    $Som \leftarrow 0$
    Pour j **allant de** $i+1$ **à** n **faire**
      $Som \leftarrow som + A(i,j) * x(j)$
    $x(i) \leftarrow (b(i) - som)/A(i,i)$
fin.

Remarque.

C'est deux stratégies donnent des méthodes numériquement stable. La méthode de Gauss sans stratégie de pivot ne donne pas nécessairement un algorithme stable. De plus si l'un des pivots est nul, l'algorithme n'aboutit pas.

III.2 Décomposition LU.

Nous allons donner maintenant une explication matricielle de la méthode de Gauss.

Supposons que l'on puisse effectuer l'élimination sans permutation des lignes et des colonnes.

Considérons les matrices

$$P_k = \begin{bmatrix} 1 & 0 & \cdots & & 0 \\ 0 & \ddots & & & \vdots \\ \vdots & & 1 & \ddots & \\ & & -p_{k+1,k} & \ddots & \\ & & \vdots & & 0 \\ 0 & \cdots & -p_{nk} & & 1 \end{bmatrix} \quad pour \;\; k = 1, \ldots, n-1$$

$$avec \;\; p_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} \;\; pour \; i = k+1, \ldots, n$$

Les formules récurrentes peuvent s'écrire de façon matricielle comme suit :

$$\tilde{A}^{(k)} = \boldsymbol{P}_k \tilde{A}^{(k-1)} \; avec \; \tilde{A}^{(k)} = \left(A^{(k)} \middle| b^{(k)}\right)$$

Plus précis, $\tilde{A}^{(k)}$ composée de la matrice $A^{(k)}$ et $b^{(k)}$.

Posons $\quad U = \boldsymbol{P}_{n-1} \ldots \boldsymbol{P}_k \ldots \boldsymbol{P}_1 A = A^{(n-1)}$

$$L = [\boldsymbol{P}_{n-1} \ldots \boldsymbol{P}_k \ldots \boldsymbol{P}_1]^{-1} = \boldsymbol{P}_1^{-1} \ldots \boldsymbol{P}_k^{-1} \ldots \boldsymbol{P}_{n-1}^{-1}$$

U est une matrice triangulaire supérieure et L est une matrice triangulaire inférieure à diagonale unité. Donc nous avons écrit A sous la forme $A = LU$.

Où $\quad L = \begin{pmatrix} 1 & 0 & & \cdots & & 0 \\ p_{21} & \ddots & & & & \\ \vdots & & 1 & \ddots & & \vdots \\ p_{i1} & & p_{ik} & \ddots & & \\ \vdots & & \vdots & & & 0 \\ p_{n1} & \cdots & p_{nk} & \cdots & p_{n,n-1} & 1 \end{pmatrix} \; et \; U = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ 0 & a_{22}^{(1)} & & \cdots & & a_{2n}^{(1)} \\ & & & & & \vdots \\ \vdots & & \ddots & \ddots & & a_{in} \\ & & & & & \vdots \\ 0 & & \cdots & & 0 & a_{nn}^{(n-1)} \end{pmatrix}$

Nous sommes amenés à résoudre successivement les deux systèmes triangulaires

$$\begin{cases} Ly = b \\ Ux = y \end{cases}$$

III.3 Méthode de Cholesky.

C'est une méthode qui traite le système : $Ax = b$ avec A symétrique définie positive. La méthode de Cholesky consiste à poser $A = LL^t$où L est une matrice triangulaire inférieure ainsi on :

$Ax = b \Leftrightarrow LL^t x = b$, on pose $y = L^t x$ et on obtient le système à matrice triangulaire inférieure :

$Ly = b$ puis on résout $L^t x = y$.

Algorithme de la décomposition Cholesky :

Détermination des termes de la matrice L. Par identification :

$$A = LL^t \ \ où \ \ A = (a_{ij})$$

Calcul de la 1er colonne de L (1er ligne de L^t) :

$$a_{11} = \ell_{11}^2 \ \Rightarrow \ell_{11} = \sqrt{a_{11}} \quad , \quad a_{11} > 0$$

$$a_{i1} = \ell_{i1}\ell_{11} \ \Rightarrow \ \ell_{i1} = \frac{a_{i1}}{\ell_{11}} \quad , \ i = 2, \dots, n$$

Calcul de la 2ème colonne de L (2ème ligne de L^t) :

$$a_{22} = \ell_{21}^2 + \ell_{22}^2 \ \Rightarrow \ \ell_{22} = \sqrt{a_{22} - \ell_{21}^2}$$

$$a_{i2} = \ell_{i1}\ell_{21} + \ell_{i2}\ell_{22} \ \Rightarrow \ \ell_{i2} = \frac{a_{i2} - \ell_{i1}\ell_{21}}{\ell_{22}} \ , \qquad i = 3, \dots, n$$

Calcul de la $kième$ colonne de L ($kième$ lignes de L^t) :

$$\begin{cases} \ell_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{i-1} \ell_{kj}} \\ \begin{cases} \ell_{ik} = \dfrac{a_{ik} - \sum_{j=1}^{k-1} \ell_{ij}\,\ell_{kj}}{\ell_{ii}} \\ i = k+1, \dots, n \end{cases} \\ k = 2, \dots, n \end{cases}$$

Remarque :

Pour les petits systèmes le gain est peu sensible. Par contre les grands systèmes il est intéressant d'appliquer la méthode de Cholesky que la méthode de Gauss. $(avec\ A\ \ S.D.P.)$

IV. Méthodes itératives.

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

$$x = \lim_{k \to +\infty} x^{(k)} \quad (1)$$

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

Une stratégie est de considérer la relation de récurrence linéaire :

$$x^{(k+1)} = Bx^{(k)} + C \quad (2)$$

Où B est la matrice d'itération de la méthode itérative $(dépendant\ de\ A)$ et C est un vecteur $(dépendant\ de\ b)$, tel que : $x = Bx + C$. Etant $x = A^{-1}b$, on obtient $C = (I - B)A^{-1}b$, la méthode itérative (2) est complètement définie par la matrice G.

En définissant l'erreur au pas k comme $e^{(k)} = x - x^{(k)}$.

On obtient la relation de récurrence $e^{(k)} = Be^{(k-1)}$, et donc , $e^{(k)} = B^{(k)}e^{(0)}$,

$k = 0, 1, \ldots$

On démontre que

$$\lim_{k \to \infty} e^{(k)} = 0 \quad \Leftrightarrow \quad \rho(B) < 1.$$

Où $\rho(B)\ le\ rayon\ spectral$ de la matrice B, défini comme $\rho(B) = \max|\lambda_i(B)|$ et $\lambda_i(B)$ sont les valeurs propres de B.

VI.1 Méthode de Jacobi.

On remarque que si les éléments diagonaux de A sont non nuls, le système linéaire $Ax = b$ est équivalent à :

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

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

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

Il s'écrit aussi

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

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

$$A = M - N \; avec \; M = D \;\; et \;\; N = E + F$$

D : diagonale de A.

E : triangulaire inférieure avec des 0 sur la diagonale.

F : triangulaire supérieure avec des 0 sur la diagonale.

La matrice d'itération de la méthode de Jacobi associée à A est donnée :

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

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

VI.2 Méthode de Gauss-Seidel.

Pour cette méthode on a :

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

Il s'écrit aussi

$$x^{(k+1)} = (D-E)^{-1}Fx^{(k)} + (D-E)^{-1}b\,.$$

Dans ce cas, le $splitting$ de A est

$$A = M - N \quad avec \quad M = D - E \;\; et \;\; N = F\,.$$

et la matrice d'itération de la méthode de Gauss-Seidel associée à A est

$$B_G = (D-E)^{-1}F$$

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

Théorème. (Convergence des méthodes itérative)

Pour qu'une méthode itérative soit convergente il faut et il suffit que $\rho(B) < 1$.

Preuve. Voir [10] P : 48.

CHAPITRE 5 : *METHODE DE RESOLUTION NUMERIQUE DES EDP*

I. Principe.

Le principe de toutes les méthodes de résolution numérique des équations aux dérivées partielles est d'obtenir des valeurs numériques discrètes (c'est-à-dire en nombre fini) qui **''approchent''** (en un sens convenable à préciser) la solution exacte. Dans ce procédé il faut bien être conscient de deux points fondamentaux : premièrement, on ne calcule pas des solutions exactes mais approchées ; deuxièmement, on discrétise le problème en représentant des fonctions par un nombre fini de valeurs, c.-à-d. que l'on passe du **''continu''** au **''discrète''**.

Il existe de nombreuses méthodes d'approximation numérique des solutions d'équations aux dérivées partielles. Nous présentons maintenant une des plus anciennes et des plus simples, appelée **méthode des différences finies**. Pour simplifier la présentation, nous n'abordons pour l'instant que les principes pratiques de cette méthode, c.-à-d. la construction de ce qu'on appelle des **schémas numériques.**

Terminaisons cette introduction en disant que la méthode des différences finies est une méthode de simulation numérique qui est encore utilisée pour certaines applications, comme la propagation des ondes ou la mécanique des fluides compressible etc.…, on lui préfère souvent la méthode des éléments finies. Néanmoins, de nombreux concepts en différences finies se retrouvent dans toutes les autres méthodes numériques.

II. Discrétisation du domaine.

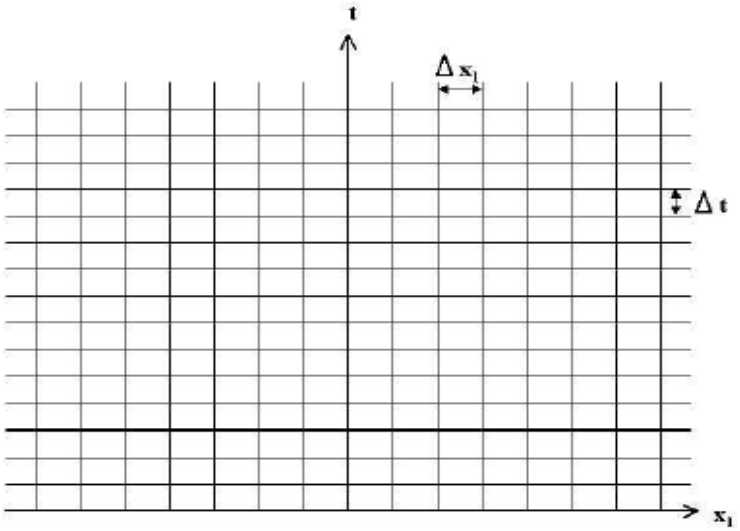


Figure 10 : Maillage en différences finies.

Pour discrétiser le continu, on introduit un pas d'espace $\Delta x > 0$ et un pas de temps $\Delta t > 0$ qui seront les plus petites échelles représentées par la méthode numérique. On définit un maillage ou des coordonnées discrètes de l'espace et du temps (Figure 2.1)

$$(x_i, t_n) = (i\Delta x, n\Delta t) \quad pour\ n \geq 0\ et\ i \in Z$$

On note u_i^n la valeur d'une solution discrète approchée au point (x_i, t_n), et $u(x,t)$ la solution exacte (inconnue). Le principe de la méthode des différences finies est de remplacer les dérivées partielles par des différences finies en utilisant des formules de Taylor dans laquelle on néglige les restes.

III. Approximation des dérivées par des différences finies.

III.1 Schémas d'ordre 1.

Soit $u(x)$ une fonction de l'espace. Par définition de la dérivée, on a :

$$\frac{\partial u}{\partial x} = \lim_{h \to 0} \frac{u(x+h) - u(x)}{h}$$

Si h est petit, un développement de Taylor de $u(x)$ au voisinage de x donne :

$$u(x+h) = u(x) + h\frac{\partial u}{\partial x}(x) + \cdots + \frac{h^n}{n!}\frac{\partial^n u}{\partial x^n} + \theta(h^{n+1})$$

En tronquant la série en premier ordre en h, on obtient :

$$\frac{u(x+h)-u(x)}{h} = \frac{\partial u}{\partial x}(x) + \theta(h)$$

L'approximation de la dérivée $\frac{\partial u}{\partial x}(x)$ est alors d'ordre 1 indiquant que l'erreur de troncature $\theta(h)$ tend vers 0 comme la puissance première de h.

On note $\left(\frac{\partial u}{\partial x}\right)_{x=x_i} = \left(\frac{\partial u}{\partial x}\right)_i = u_i^{'}$. Cette notation s'utilise d'une façon équivalente pour toutes les dérivées d'ordre successif de la grandeur u.

Le schéma aux différences finies d'ordre 1 présenté au dessus s'écrit :

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1}-u_i}{h} + \theta(h)$$

Ce schéma est dit ‘‘décentré avant’’.

Il est possible de construire un autre schéma d'ordre 1, appelé ‘‘décentré en arrière’’ :

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_i-u_{i-1}}{h} + \theta(h) \;\; avec\; h = \Delta x$$

III.2 Schéma d'ordre supérieur.

Des schémas aux différences finies d'ordre supérieur peuvent être construits en manipulant de développement de Taylor au voisinage de x_i. On écrit :

$$u_{i+1} = u(x_i+\Delta x) = u_i + \Delta x\left(\frac{\partial u}{\partial x}\right)_i + \frac{\Delta x^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i + \theta(\Delta x^3)$$

$$u_{i-1} = u(x_i-\Delta x) = u_i - \Delta x\left(\frac{\partial u}{\partial x}\right)_i + \frac{\Delta x^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i - \theta(\Delta x^3)$$

La soustraction de ces deux relations donne : $u_{i+1} - u_{i-1} = 2\Delta x\left(\frac{\partial u}{\partial x}\right)_i + 2\theta(\Delta x^3)$

Ce qui permet d'obtenir le schéma d'ordre 2 dit ‘‘centré’’ pour approximer la dérivée première de u :

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + \theta(\Delta x^2)$$

Pour obtenir des ordres supérieurs, il faut utiliser plusieurs nœuds voisins de x_i. Par exemple, un schéma aux différences finies d'ordre 3 pour la dérivée première s'écrit :

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{-u_{i+2} + 6u_{i+1} - 3u_i - 2u_{i-1}}{6\Delta x} + \theta(\Delta x^3)$$

III.3 Dérivée d'ordre supérieur.

Le principe est identique et repose sur les développements de Taylor au voisinage de x_i. Par exemple pour construire un schéma d'approximation de la dérivée seconde de u, on écrit :

$$u_{i+1} = u(x_i + \Delta x) = u_i + \Delta x\left(\frac{\partial u}{\partial x}\right)_i + \frac{\Delta x^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{\Delta x^3}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \theta(\Delta x^4)$$

$$u_{i-1} = u(x_i - \Delta x) = u_i - \Delta x\left(\frac{\partial u}{\partial x}\right)_i + \frac{\Delta x^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{\Delta x^3}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \theta(\Delta x^4)$$

En faisant la somme de ces deux égalités, on abouti à :

$$u_{i+1} + u_{i-1} = 2u_i + \Delta x^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i + 2\theta(\Delta x^4)$$

Ce qui permet d'obtenir le schéma d'ordre 2 dit ‘‘centré’’ pour approximer la dérivée seconde de u :

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} - 2\theta(\Delta x^2)$$

Il reste aussi une formulation ‘‘avant’’ et ‘‘arrière’’ pour la dérivée seconde, toute deux d'ordre 1 :

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{u_{i+2} - 2u_{i+1} + u_{i-1}}{\Delta x^2} + \theta(\Delta x) \qquad \left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{u_i - 2u_{i-1} + u_{i-2}}{\Delta x^2} + \theta(\Delta x)$$

Il est également possible de construire, par le même procédé, des schémas aux différences finies d'ordre supérieur pour les dérivées deuxième, troisième, etc.…

IV. Exemples de discrétisations de l'équation de la chaleur en dimension.

Nous considérons le problème de l'équation de la chaleur dans le domaine borné Ω=]0,1[

$$\begin{cases} \dfrac{\partial u}{\partial t} = \nu \dfrac{\partial^2 u}{\partial x^2} \quad ; \quad x\epsilon]0,1[\ , \ t\epsilon]0,T[\\ u(x,0) = u^0(x) \quad ; \quad x\epsilon]0,1[\\ u(0,t) = \alpha \quad ; \quad t\epsilon]0,T] \\ u(1,t) = \beta \quad ; \quad t\epsilon]0,T] \end{cases} \qquad (4.1)$$

Les conditions aux limites de (4.1) peuvent être de plusieurs types, mais leur choix n'intervient pas dans la définition des schémas. Ici, nous utilisons des conditions aux limites de Dirichlet.

Pour discrétiser le domaine $\Omega \times [0,T]$, on introduit un pas d'espace $\Delta x > 0$ et un pas de temps $\Delta t > 0$, et on définit les nœuds d'un maillage régulier

$$(x_i, t_n) = (i\Delta x, i\Delta t) \quad pour\ n \geq 0, i \in \{0,1 \dots, N\}$$

On note u_i^n la température au point (x_i, t_n).

On peut utiliser deux approches pour discrétiser cette équation de la chaleur. La première dite *explicite* utilise une discrétisation au nœud x_i et à l'itération courante n :

$$\left(\frac{\partial u}{\partial t}\right)_i^n = \boldsymbol{\nu}\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n$$

Et la seconde dite *implicite* utilise une discrétisation au nœud x_i et à l'itération $n+1$:

$$\left(\frac{\partial u}{\partial t}\right)_i^{n+1} = \boldsymbol{\nu}\left(\frac{\partial^2 u}{\partial x^2}\right)_i^{n+1}$$

IV.1 Schéma explicite.

Nous utilisons un schéma avant d'ordre 1 pour évaluer la dérivée temporelle et un schéma centré d'ordre 2 pour la dérivée seconde en espace :

$$\left(\frac{\partial u}{\partial t}\right)_i^n = \frac{u_i^{n+1} - u_i^n}{\Delta t}$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}$$

En posant $\lambda = \nu\frac{\Delta t}{\Delta x^2}$, la température à l'itération $n+1$ est donnée par :

$$u_i^{n+1} = \lambda u_i^n + (1-2\lambda)u_i^n + \lambda u_{i+1}^n \quad i \text{ variant de 1 à } N-1$$

Soit sous forme matricielle :

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix}^{n+1} = \begin{bmatrix} 1-2\lambda & \lambda & 0 & \cdots & 0 \\ \lambda & 1-2\lambda & \lambda & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \lambda & 1-2\lambda & \lambda \\ 0 & 0 & 0 & \lambda & 1-2\lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix}^n + \lambda \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \\ \beta \end{bmatrix}$$

IV.2 Schéma implicite.

Nous utilisons un schéma arrière d'ordre 1 pour évaluer la dérivée temporelle et un schéma centré d'ordre 2 pour la dérivée seconde en espace :

$$\left(\frac{\partial u}{\partial t}\right)_i^{n+1} = \frac{u_i^{n+1} - u_i^n}{\Delta t}$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i^{n+1} = \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2}$$

De même on pose $\lambda = \nu\frac{\Delta t}{\Delta x^2}$, la température à l'itération $n+1$ est donnée par :

$$(1+2\lambda)u_i^{n+1} - \lambda\left(u_{i+1}^{n+1} + u_{i-1}^n\right) = u_i^n \quad i \text{ variant de 1 à } N-1$$

On constate que les inconnues à l'itération $n+1$ sont reliées entre elles par une relation implicite (d'où le nom de la méthode).

Sous forme matricielle :

$$\begin{bmatrix} 1+2\lambda & -\lambda & 0 & \cdots & 0 \\ -\lambda & 1+2\lambda & -\lambda & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & -\lambda & 1+2\lambda & -\lambda \\ 0 & 0 & 0 & -\lambda & 1+2\lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix}^{n+1} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix}^n + \lambda \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \\ \beta \end{bmatrix}$$

A chaque itération le vecteur, le vecteur des inconnues discrètes se détermine par résolution d'un système linéaire. La matrice du système étant tridiagonale, un algorithme de pivot de Gauss est très souvent utilisé.

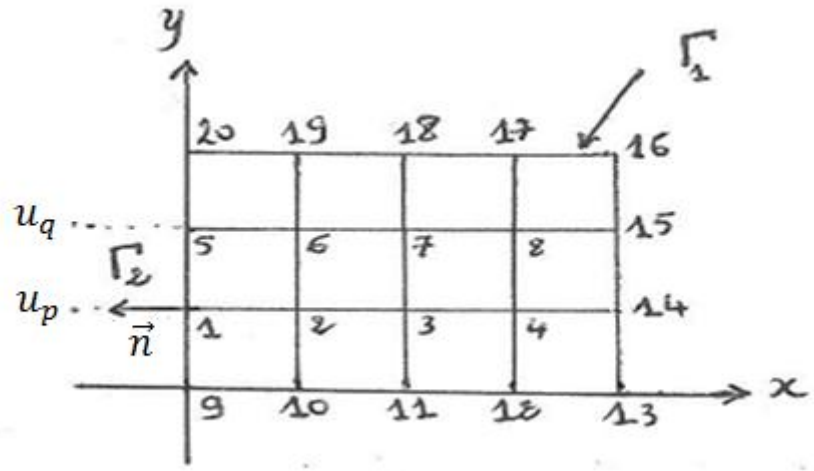
V. Exemples d'application.

V.1 Equation de Laplace.

Problème en différences finies à 5 points

$$(P)\begin{cases} \Delta u = 0 & dans\ \Omega \\ u|_{\Gamma_1} = 1 & sur\ \Gamma_1 \quad (condition\ de\ Direchlet) \\ \left(u + \dfrac{du}{dx}\right)_{\Gamma_2} = 1 & sur\ \Gamma_2 \quad (condition\ de\ Fourier) \end{cases}$$

On considère le maillage suivant :



On a les pas $\Delta x = \Delta y = h = 1$.

Et on a $\Delta u = 0 \quad \Leftrightarrow \quad \frac{d^2u}{dx^2} + \frac{d^2u}{dy^2} = 0$ avec $u(x,y) = u_{ij}$

Nous utilisons un schéma centré d'ordre 2 pour la dérivée seconde :

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{ij} = \frac{u_{i-1,j} - 2u_{ij} + u_{i+1,j}}{\Delta x^2}$$

$$\left(\frac{\partial^2 u}{\partial y^2}\right)_{ij} = \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{\Delta y^2}$$

On obtient un schéma à 5 points associée à le problème (P) ci-dessous :

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij} = 0$$

On note $u_{ij} \rightarrow u_k \quad k = 9, \dots, 20.$

Les inconnues : $u_1, u_2, \dots, u_8$?

Equations discrétisés :

Au point 1 : $u_2 + u_p + u_5 + u_9 - 4u_1 = 0$

Au point 2 : $u_3 + u_1 + u_6 + u_{10} - 4u_2 = 0$

Au point 3 : $u_4 + u_2 + u_7 + u_{11} - 4u_3 = 0$

Au point 4 : $u_{14} + u_3 + u_8 + u_{12} - 4u_4 = 0$

Au point 5 : $u_6 + u_q + u_{20} + u_1 - 4u_5 = 0$

Au point 6 : $u_7 + u_5 + u_9 + u_2 - 4u_6 = 0$

Au point 7 : $u_8 + u_6 + u_{18} + u_3 - 4u_7 = 0$

Au point 8 : $u_{15} + u_7 + u_{17} + u_4 - 4u_8 = 0$

Les conditions de Fourier au point 1 et 5 :

Tout d'abord on a : $\left(u + \frac{du}{dx}\right)_{\Gamma_2} = 1 \quad \rightarrow \quad \left(u_{ij} + \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}\right)_{\Gamma_2} = 1$

- Au point 1 : $u_1 + \frac{u_p - u_2}{2} = 1 \quad \Leftrightarrow \quad u_p = 2 - 2u_1 + u_2$
- Au point 5 : $u_5 + \frac{u_q - u_6}{2} = 1 \quad \Leftrightarrow \quad u_q = 2 - 2u_5 + u_6$

On obtient un système de la forme :

$$\left(\begin{array}{cccc|cccc} -6 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & -6 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 \end{array}\right)\left(\begin{array}{c} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \hline u_5 \\ u_6 \\ u_7 \\ u_8 \end{array}\right) = \left(\begin{array}{c} -3 \\ -1 \\ -1 \\ -2 \\ \hline -3 \\ -1 \\ -1 \\ -2 \end{array}\right)$$

Donc, on a un problème symétrique c.-à-d. symétrie géométrique et symétries des conditions aux limites.

D'où $$\begin{cases} u_1 = u_5 \\ u_2 = u_6 \\ u_3 = u_7 \\ u_4 = u_8 \end{cases}$$

Donc on a :

$$\begin{pmatrix} -6 & 2 & 0 & 0 \\ 1 & -4 & 1 & 0 \\ 0 & 1 & -4 & 1 \\ 0 & 0 & 1 & -4 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} -3 \\ -1 \\ -1 \\ -2 \end{pmatrix}$$

Alors $$\begin{pmatrix} -5 & 2 & 0 & 0 \\ 1 & -3 & 1 & 0 \\ 0 & 1 & -3 & 1 \\ 0 & 0 & 1 & -3 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} -3 \\ -1 \\ -1 \\ -2 \end{pmatrix}$$

En utilisant la méthode de Gauss on obtient :

$$u_1 = u_2 = u_3 = u_4 = 1$$

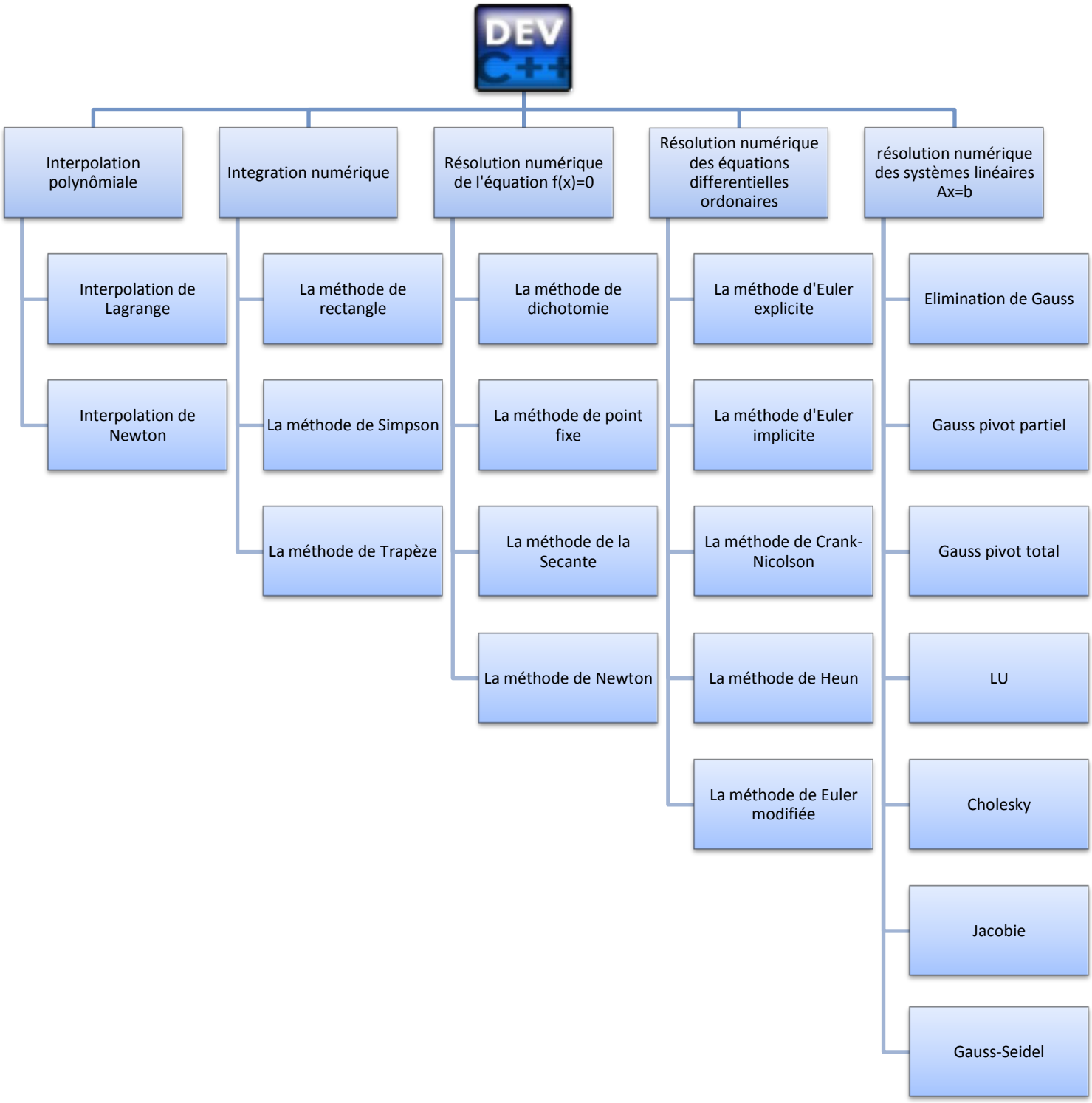
Et on a aussi $$u_5 = u_6 = u_7 = u_8 = 1$$

D'où la symétrie.

CHAPITRE 6 : *REALISATION D'UN LOGICIEL DE CALCUL D'ANALYSE NUMERIQUE.*

I. Présentation du programme.

L'organigramme de logiciel source :



II. Le code source et validation.

III.1 Le code source en Langage C.

Le code C de l'interpolation de Lagrange.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define n 4       // le nombre de points d'interpolations
float Lagrange(float x[n],float y[n],float a){
float som=0,L;
int i,j;
for(i=0;i<n;i++){
L=1;
for(j=0;j<i;j++)     // calcule des valeurs de Li(x)
L*=(a-x[j])/(x[i]-x[j]);
for(j=i+1;j<n;j++)
L*=(a-x[j])/(x[i]-x[j]);
som+=L*y[i];}       // calcule du polynôme d'interpolation en un point donné
return som;
}
main(){
float x[n],y[n],a;
int i;
printf("saisir les points d'interpolations:\n");
for(i=0;i<n;i++){
printf("x[%d]=",i);
scanf("%f",&x[i]);}
printf("saisir les images d'interpolations:\n");
```

```
for(i=0;i<n;i++){
printf("y[%d]=",i);
scanf("%f",&y[i]);}
printf("saisir le poit point que vous voulez interpoler :");
scanf("%f",&a);
float b;
b=Lagrange(x,y,a);
printf("P(%.0f)=%f\n",a,b);
getch();
}
```

Le code C de la méthode de Rectangle

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>
//calcule d'integrale par la formule du rectangle (composée)
float fx(float x){
    float t;
    t=exp(-pow(x,2));
    return t;}
//fonction qui permet de calculer la surface de l'intégrale
main(){
 int n,a,b;
 float h,s=0;
 printf("saisir n:\n");
 scanf("%d",&n);    //le nombre de subdivision de l'intervalle
 printf("saisir les bornes de l'intervalle:\n");
```

```
 scanf("%d %d",&a,&b);
 h=(float)(b-a)/n;
 int i;
 for(i=0;i<n;i++)  // a=x[0] est fixé car x[i]=x[0]+i*h
 s+=h*fx(a+i*h+h/2);
 printf("la surface de la fontion entre [%d,%d] est %f\n",a,b,s);
getch();
}
```

Le code C de la méthode de Dichotomie

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>
#define N 200 // Le nombre maximale d'itérations pour atteindre la solution approchée
float fct_dich(float x){
return log(x);
}
void Dichotomie(){
float *a,*b,*c,x,y;
a=(float*)malloc(N*sizeof(float)); // Allocation dynamique
b=(float*)malloc(N*sizeof(float));
c=(float*)malloc(N*sizeof(float));
printf("\n\n --------- Dichotomie ---------\n");
printf("\n\n saisir un intervalle dont il existe un seul solution:  ");
scanf("%f %f",&x,&y);
a[0]=x;
```

```
b[0]=y;
c[0]=(a[0]+b[0])/2.0;
int i=0;
do{
if(fct_dich(a[i])*fct_dich(c[i])<0){
a[i+1]=a[i];
b[i+1]=c[i];
}
if(fct_dich(b[i])*fct_dich(c[i])<0){
a[i+1]=c[i];
b[i+1]=b[i];
}
c[i+1]=(a[i+1]+b[i+1])/2.0;
i++;
}while(fabs(b[i]-a[i])>pow(10,-20));
printf("\nle nombre d'iteration pour approcher de la solution est %d\n\n",i-1);
printf("\n\ la solution proche est x=%f\n\n",c[i-1]);
getch();
}
```

Le code C de la méthode de d'Euler explicite

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define h 0.1 // pas h
#define M 10 // nombre de points
float f(float x, float y){ // la fonction f(t,y)
return y+x;
```

```
}
main(){
float *t,*y; // tableau xn et yn
t = (float *)malloc(M*sizeof(float));
y = (float *)malloc(M*sizeof(float));
printf("\n\n--------- Euler explicite ---------\n\n");
y[0]=1; // valeur initiatle de y(0)
for(int i=0;i<=M;i++){
t[i] = t[0]+i*h; // initialisation des abscisses t_n
y[i+1] = y[i] +h*f(t[i],y[i]);
printf("n=%d y[%d]=%f\n",i,i,y[i]);
}
free(t);
free(y);
getch();
}
```

Le code C de la méthode de Gauss pivot partiel

```
#include <stdio.h>
 #include<math.h>
 #include<stdlib.h>
 // fonction de remplissage
 void remplissage(float a[19][19],float b[19],int n){
 int i,j;
 printf("\n * La matrice A:\n\n");
 printf("\n * remplisser La matrice ligne par ligne \n\n");
 for (i=0;i<n;i++){
 printf(" | ");
```

```
for (j=0;j<n;j++)
scanf("%f",&a[i][j]);
}
printf("\n * Le vecteur b:\n\n");
for(i=0;i<n;i++){
printf(" | ");
scanf("%f",&b[i]);
} }
// fonction d'affichage système
void aff_syst(float A[19][19],float b[19],int n){
int i,j;
printf("\n\n");
for (i=0;i<n;i++){
printf(" [");
for (j=0;j<n;j++){
printf(" %.4f ",A[i][j]);
}
printf("] [ %.4f ]",b[i]);
printf("\n");
} }
// Mettre à zéro les éléments qui doivent être des zéros
void zero(float A[19][19],float b[19],int n){
int i,j;float eps=1e-4;
for(i=0;i<n;i++){
for (j=0;j<n;j++) if (fabs(A[i][j])<eps) A[i][j]=0;
if (fabs(b[i])<eps)  b[i]=0;
}
}
```

```
// ------------------------------------------------------------
// Résolution Par Gauss pivot partiel
// ------------------------------------------------------------
void gauss_pivot_partiel(float A[19][19],float b[19],int n){
float x[19],p,s,ref,temp;int i,j,k,ligne;
for(k=0;k<n-1;k++){
// max pour le pivot partiel
ref=0;
for(i=k;i<n;i++)
if(fabs(A[i][k])>ref){
ref=fabs(A[i][k]);ligne=i;
}
// pivotations
for(j=k;j<n;j++){
temp=A[k][j]; A[k][j]=A[ligne][j] ;A[ligne][j]=temp;
}
temp=b[k]; b[k]=b[ligne]; b[ligne]=temp;
if (A[k][k]==0){
printf("\n* Un pivot nul ! => methode de Gauss pivot partiel non applicable");
}
//réduction
for(i=k+1;i<n;i++){
p=A[i][k]/A[k][k];
for (j=k;j<n;j++) A[i][j]=A[i][j]-p*A[k][j];
b[i]=b[i]-p*b[k];
} }
//Résolution
for(i=n-1;i>=0;i--){
```

```
s=0;
for(j=i+1;j<n;j++) s=s+A[i][j]*x[j];
x[i]=(b[i]-s)/A[i][i];}
zero(A,b,n);
printf("\n-------- Gauss avec pivot partiel --------\n");
printf("\n * La matrice reduite :");
aff_syst(A,b,n);
printf("\n * La resolution donne :\n\n");
for (i=0;i<n;i++) printf(" X_%d = %f ;\n",i+1,x[i]);
printf("\n");
}
main(){
float A[19][19],b[19];int n,i,j;
printf("\n ************************************************\n");
printf(" * *\n");
printf(" * Resolution de A x = b *\n");
printf(" * *\n");
printf(" ************************************************\n");
printf(" * *\n");
printf(" * la methode de Gauss pivot partiel *\n");
printf(" ************************************************\n");
printf("\n\n * Nombre d'equation-inconues : \n\n * n = ");
scanf("%d",&n);
remplissage(A,b,n);
printf("\n\n * Le systeme est : ");
aff_syst(A,b,n);
gauss_pivot_partiel(A,b,n);
system("pause"); }
```

III.2 Les exécutions des programmes.

// Le polynôme d'interpolation est : $P(x) = -\frac{1}{6}x^3 + \frac{1}{2}x^2 + \frac{2}{3}x + 1$.

- **Méthode de Lagrange.**

```
saisir les points d'interpolations:
x[0]=-1
x[1]=1
x[2]=0
x[3]=2
saisir les images d'interpolations:
y[0]=1
y[1]=1
y[2]=2
y[3]=3
saisir un point que vous voulez interpoler: 2
P(2)=3.000000
```

// Calcul de la surface de l'intégrale $S = \int_0^1 e^{-x^2} dx$ sur l'intervalle [0,1].

- **Méthode de Rectangle.**

```
saisir n:10
saisir les bornes de l'intervalle:0 1
la surface de la fontion entre [0,1] est S=0.747131
```

// le but est de trouver la solution de l'équation ln(x)=0 sur l'intervalle $[0.5,2]$

- **Méthode de Dichotomie.**

```
saisir un intervalle dont il existe un seul solution:  0.5  2

le nombre d'iteration pour approcher de la solution est 22

la solution proche est x=1.000000
```

// on cherche l'approximation y_{10} de la fonction définie par $\begin{cases} f(t,y(t)) = y(t) + t\,, t\epsilon[0,1] \\ y(0) = 1 \end{cases}$

en subdivisant l'intervalle [0,1] en 10 parties égaux avec un pas $h = 0.1$.

- **Méthode d'Euler explicite.**

```
n = 0          y[0]=1.000000
n = 1          y[1]=1.100000
n = 2          y[2]=1.220000
n = 3          y[3]=1.362000
n = 4          y[4]=1.528200
n = 5          y[5]=1.721020
n = 6          y[6]=1.943122
n = 7          y[7]=2.197434
n = 8          y[8]=2.487178
n = 9          y[9]=2.815895
n = 10          y[10]=3.187485
```

Remarque.

La solution analytique est simple et on peut facilement vérifier notre solution à la solution exacte.

❖ **Méthode de Gauss pivot partiel.**

```
* Nombre d'equation-inconues :

* n = 3

* La matrice A:

| 0.000003 0.213472 0.332147
| 0.215512 0.375623 0.476625
| 0.173257 0.663257 0.625675

* Le vecteur b:

| 0.235262
| 0.127653
| 0.285321


* Le systeme est :

[ 0.0000  0.2135  0.3321 ] [ 0.2353 ]
[ 0.2155  0.3756  0.4766 ] [ 0.1277 ]
[ 0.1733  0.6633  0.6257 ] [ 0.2853 ]
la permutation des lignes avant la transformation

[ 0.2155  0.3756  0.4766 ] [ 0.1277 ]
[ 0.0000  0.2135  0.3321 ] [ 0.2353 ]
[ 0.1733  0.6633  0.6257 ] [ 0.2853 ]

la matrice apres  la transformation 1:

[ 0.2155  0.3756  0.4766 ] [ 0.1277 ]
[ 0.0000  0.2135  0.3321 ] [ 0.2353 ]
[ 0.0000  0.3613  0.2425 ] [ 0.1827 ]
la permutation des lignes avant la transformation

[ 0.2155  0.3756  0.4766 ] [ 0.1277 ]
[ 0.0000  0.3613  0.2425 ] [ 0.1827 ]
[ 0.0000  0.2135  0.3321 ] [ 0.2353 ]

la matrice apres  la transformation 2:

[ 0.2155  0.3756  0.4766 ] [ 0.1277 ]
[ 0.0000  0.3613  0.2425 ] [ 0.1827 ]
[ 0.0000  0.0000  0.1889 ] [ 0.1273 ]

-------- Gauss avec pivot partiel --------

* La matrice reduite :

[ 0.2155  0.3756  0.4766 ] [ 0.1277 ]
[ 0.0000  0.3613  0.2425 ] [ 0.1827 ]
[ 0.0000  0.0000  0.1889 ] [ 0.1273 ]

* La resolution donne :

X_1 = -0.991289 ;
X_2 = 0.053204 ;
X_3 = 0.674122 ;
```

Conclusion Générale

Le travail fourni à travers ce polycopié a tenté de faire un rappel de l'algorithme servant comme base de calcul sans démonstration ni calcul d'erreur. D'autre part, comme je peux le constater, chaque code jugé délicat et utile est commenté et expliqué. Les exemples que j'ai choisis sont le fruit de notre propre expérience et serviront comme support de calcul pour d'autres cas. Néanmoins, il est fortement recommandé de bien comprendre les routines écrites en Langage C avant de plonger dans la programmation quand il s'agira de traiter des cas similaires. En outre, l'étudiant doit fournir des efforts et imaginer d'autres solutions et d'innover dans la programmation. Une fonction en Langage C, par exemple, peut être écrite de plusieurs façons et nous recommandons d'alléger le code et le rendre claire et lisible afin de faciliter son débogage. Comme dernier conseil, il est toujours plus facile de modifier un code C dont les fonctions sont séparées du main (), il est alors vivement conseillé d'écrire d'abord les constantes fixes, les variables globales, les fonctions, ensuite viendra le main() qui doit ne contenir que les appels aux fonctions. C'est la meilleure façon de se rendre compte de ses erreurs sans perturber la structure du programme.

Bibliographie

Les documents employés :

[1] A. Quarteroni, R. Sacco, and F. Saleri.Méthodes numériques. Algorithmes, analyseet applications. Springer, 2007.

[2] DURAND – Solution numérique des équations algébriques, Masson, 1962.

[3] Fortin, M. et pierre, R., Analyse numérique MAT-10427, Québec, Université Laval, 1993.

[4] G. Allaire and S. M. Kaber. Algèbre linéaire numérique. Mathématiques pour le deuxième cycle. Ellipses, 2002.

[5] G.H. HACQUES – Mathématique pour l'informatique (3) , Colin, 1971.

[6] H. BEREZIS, M. SIBONY – Méthodes d'approximation et d'itération pour les opérateurs monotones, Arch. For Rational Mechanics and analysis (Vol. 28 n°1) , 1968 .

[7] H. BEREZIS, M. SIBONY – Equivalence de deux inéquations variationnelles et applications, Arch. For Rational Mechanics and analysis (Vol. 41 n°4) , 1971 .

[8] M. Schatzman, Analyse numérique, Masson.

[9] M. SIBONY – Sur l'approximation d'équations et inéquations aux dérivées partielles non linéaires de type monotone, Math. Analysis and Appl. (Vol. 34 n°3) , 1971.

Vuibert, 1998 P. Depondt Méthodes de calcul numérique, Ed. Hermès, 2001 J.P. Nougier..

[10] M. SIBONY & J.CI.MARDON– ANALYSE NUMERIQUE I, Hermann, 293 rue Lecourbe, 75015 Paris.

[11] P. Viot. Méthodes d'analyse numériques. Cours de DEA de Jussieu, 2003