Abdelhalim Yahiaoui
Maître de conférences en Hydraulique
Université de Bouira, Algérie

Méthodes Numériques en Ingénierie

Méthodes Numériques en Ingénierie

Abdelhalim Yahiaoui
Maître de conférences en Hydraulique
Université de Bouira, Algérie

Table des matières

Avant-propos

Les méthodes numériques est une discipline à l'interface des mathématiques et de l'informatique. Elle s'intéresse tant aux fondements qu'à la mise en pratique des méthodes permettant de résoudre, par des calculs numériques, des problèmes d'analyse mathématique et de la physique.

Le développement de l'analyse numérique est étroitement lié au développement des ordinateurs et la programmation informatique qui ont grandement facilitée l'utilisation des méthodes numériques dans de nombreux domaines scientifiques, techniques et économique, avec souvent des effets importants.

Ce livre se fonde sur une expérience d'enseignement des méthodes numériques et la programmation MATLAB à l'Université de Béchar, puis à l'université de Bouira il traite les méthodes numériques allant de leurs fondements à leurs implémentations dans MATLAB. Il couvre essentiellement les points suivants :

- les fondements du calcul numérique ;
- les méthodes qui concernent la résolution numériques des systèmes d'équations linéaires et non linéaires ;
- l'interpolation polynomiale et les splines et les méthodes d'approximation des fonctions ;
- l'intégration numérique ;
- les méthodes numériques utilisées dans la résolution des équations différentielles ordinaires et aux dérivées partielles.

L'approche pédagogique de ce livre repose sur une compréhension profonde des méthodes numériques, cela signifie que les exemples choisis cherchent avant tout à illustrer différents aspects des méthodes et à souligner leurs avantages et leurs inconvénients en introduisant à la fin de chaque chapitre une série d'exercices résolus et à résoudre pour tester les méthodes utilisées en mesure de valider leurs résultats.

La préparation de ce livre est débutée vers le mois de mai 2017 après avoir terminé un premier livre sur MATLAB (Yahiaoui, 2017). J'ai trouvé fort utile de préparer ce livre sur les méthodes numériques et leurs implémentations sous MATLAB. En présentant un livre qui traite les méthodes numériques en ingénierie de façon plus avancé. Le livre est organisé suivant sept chapitres et deux annexes complémentaires.

Le premier chapitre, est une introduction aux fondements du calcul numérique, il traite la position d'un problème numérique, son conditionnement, sa stabilité et sa convergence.

Le deuxième chapitre, traite la résolution numérique des systèmes d'équations linéaires ou deux catégories de méthodes peuvent être utilisées directes comme les méthodes

d'élimination et de décomposition et indirectes, comme les méthodes itératives (Gauss-Seidel, Jacobi, SOR,…).

Le troisième chapitre, traite les méthodes numériques utilisées dans la résolution des équations non linéaires et les systèmes d'équations non linéaires.

Le quatrième chapitre, traite le sujet d'interpolation polynomiale et l'approximation des fonctions allant des méthodes d'interpolation classique de Lagrange et des différences divisées de Newton aux méthodes d'interpolation par les splines cubiques.

Le cinquième chapitre, traite le calcul numérique des intégrales simples, impropres et multiples par plusieurs méthodes basées essentiellement sur les méthodes de quadratures.

Le sixième chapitre, traite le sujet de la résolution numérique des équations différentielles ordinaires comme les problèmes à valeurs initiales (problème de Cauchy) et les problèmes aux limites (problème de Dirichlet ou de Neumann).

Le septième et dernier chapitre, traite la résolution numérique des équations différentielles aux dérivées partielles par les méthodes basées essentiellement sur le concept des différences finies.

Le livre se termine avec deux annexes, le premier sur l'algèbre des matrices et le deuxième sur l'essentiel du MATLAB. Ces annexes traitent les points essentiels et forts utiles pour les chapitres cités.

Ce livre est destiné à tous ce qui intéresse des méthodes numériques et leur utilisation que ce soit dans le domaine de l'ingénierie, des sciences et de technologie.

Remerciement,

Je tiens à remercier le Docteur Lefkir Abdelouahab de l'Ecole Nationale Supérieure des Travaux Publics de Kouba à Alger, le Docteur Ait-Yala Abdelmadjid de la Faculté des Sciences et Sciences Appliquées de l'Université de Bouira et le Docteur Boudref Mohamed Ahmed de la même faculté, pour leurs expertises de ce livre et pour leurs suggestions afin de présenter un travail que je souhaite soit très utile pour les étudiants, les chercheurs et les praticiens qui cherchent à résoudre les problèmes mathématiques par les méthodes numériques.

Bouira (Algérie), le 3 novembre 2021

Chapitre 1.

Les fondements du calcul numérique

Les méthodes numériques sont des algorithmes utilisées dans de nombreux problèmes de sciences physiques biologiques, technologiques, etc. Elle intervient dans le développement de codes de calcul (météorologie, physique des particules...), mais aussi dans les problèmes de simulations (hydraulique, aéronautique...) ou d'expérimentations mathématiques. Elle entretient des liens étroits avec la programmation informatique. Si sa partie théorique relève plus des concepts mathématiques, sa mise en pratique aboutit généralement à l'implémentation d'algorithmes sur ordinateur par la programmation avec un langage informatique comme le MATLAB par exemple. Ses méthodes se fondent à la fois sur la recherche de solutions exactes ou du calcul symbolique, sur des solutions approchées qui résultent le plus souvent de processus de discrétisation comme dans le traitement des équations différentielles.

1. Erreurs et précision

Les erreurs commises dans le calcul numérique sont de trois types :

Erreurs arrondies, elles sont imposées par le calculateur. La représentation d'un nombre en mémoire de l'ordinateur étant finie, tout nombre réel n'est connu qu'avec une précision donnée de de n chiffres significatifs. Pour un nombre quelconque compris entre 0 et 1, l'ordinateur écrira par exemple :

$$x = 0.a_1 a_2 a_3 \ldots a_n$$

Lors de la manipulation Lors de la manipulation de ces nombres, la machine devra choisir entre la troncature ou l'arrondi à la décimale la plus proche. Ces erreurs peuvent induire des problèmes de précision (Jedrzejewski, 2005).

Erreurs de troncatures, elles sont liées à la précision de l'algorithme utilisé. Elles peuvent être contrôlées par l'algorithme lui-même. Si une fonction est approchée par son développement de Taylor, l'erreur de troncature sera obtenue par une évaluation du reste du développement. Son contrôle sera obtenu par une majoration de ce reste. Au voisinage d'un point d'abscisse a, si une fonction f admet un développement de Taylor de la forme :

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2!}f''(a) + \cdots + \frac{(x-a)^{n-1}}{(n-1)!}f^{(n-1)}(a) + \int_a^x \frac{(x-t)^{n-1}}{(n-1)!}f^{(n)}(t)dt$$

L'erreur de troncature est exprimée et majorée par une constante M :

$$\left| \int_a^x \frac{(x-t)^{n-1}}{(n-1)!}f^{(n)}(t)dt \right| \leq M\frac{|x-a|^n}{n!}$$

Erreurs de méthode, elles sont produites lorsqu'une expression est mal équilibrée et mélange des valeurs dont la différence est importante. C'est un problème de calibration

numérique qui est sensible aux erreurs d'arrondi. Dans les processus récurrents ou itératifs, les erreurs s'ajoutent, ce qui a pour effet d'amplifier l'erreur globale et de diminuer la précision du calcul. La propagation des erreurs dans diverses parties du calcul a pour conséquence d'ajouter de l'imprécision là où elle n'était pas nécessairement attendue. Dans les calculs itératifs, l'erreur se propage d'une étape à l'autre. Par exemple, dans le calcul numérique des termes de la suite définie par la relation de récurrence :

$$u_{n+1} = \frac{1}{n^2} + a u_n$$

Si u_r la valeur exacte, l'erreur $e_n = |u_n - u_r|$ alors e_{n+1} est exprimée pour n assez grand par :

$$e_{n+1} \approx a e_n$$

L'erreur dans ce cas évolue exponentiellement. A l'itération n, l'erreur est multipliée par a^n d'une itération à l'autre, l'erreur se propage et petit et peut conduire à l'expression de l'algorithme.

2. Problèmes bien posés

Soit le problème suivant explicité par la fonction dont le but est de trouver x, tel que :

$$F(x,\ \theta) = 0$$

θ, est un ensemble de données dont dépend de la solution et F est un opérateur mathématique fonctionnelle entre x et θ. Selon la nature du problème, les variables x et θ peuvent être des nombres réels, des vecteurs ou des fonctions. Un problème est dit *direct* si F et θ sont les données et x l'inconnue, un problème dit *inverse* si F et x sont les données et θ l'inconnue, un problème est dit d'*identification* si x et θ sont données et F est inconnue.

Un problème est *bien posé* si la solution x existe, est unique et dépend continûment des données. Par contre un problème qui ne possède pas cette propriété est dit *mal posé* ou *instable*. Il faut alors le régulariser, c'est-à-dire le transformer convenablement en un problème bien posé (Morozov, 1984) avant d'envisager sa résolution numérique. Il n'est en effet pas raisonnable d'espérer qu'une méthode numérique traite les pathologies intrinsèques d'un problème mal posé (Quarteroni *et al.*, 2007).

La dépendance continue par rapport aux données signifie que de petites perturbations sur les données θ induisent de "petites" modifications de la solution x. Plus précisément, soient $\delta\theta$ une perturbation admissible des données et δx la modification induite sur la solution telles que.

$$F(x+\delta x,\ \theta+\delta\theta) = 0 \ \text{ avec, } \ \|\delta\theta\| \le \eta \ \Rightarrow \ \|\delta x\| \le \kappa_0 \|\delta\theta\|$$

C'est-à-dire, la propriété selon laquelle de petites perturbations sur les données entrainent des perturbations du même ordre sur la solution. Le *conditionnement relatif* du problème $F(x,\ \theta) = 0$ est définie par :

$$\kappa(\theta) = \sup\left\{ \frac{\|\delta x\|/\|x\|}{\|\delta\theta\|/\|\theta\|},\ \delta\theta \neq 0 \right\}$$

Quand $\theta = 0$ ou, il est nécessaire d'introduire le *conditionnement absolu*, définie par :

$$\kappa_a(\theta) = \sup\left\{\frac{\|\delta x\|}{\|\delta\theta\|},\ \delta\theta \neq 0\right\}$$

Le problème $F(x,\ \theta) = 0$ est dit *mal conditionné* si $\kappa(\theta)$ est grand pour toute donnée admissible θ (C'est-à-dire, qu'une petite perturbation dans θ conduit à un grand changement dans le problème $F(x,\ \theta) = 0$). Le sens précis de petit et grand change en fonction du problème). Le fait qu'un problème soit bien conditionné est une propriété indépendante de la méthode numérique choisie pour le résoudre. Il est possible de développer des méthodes stables ou instables pour résoudre des problèmes bien conditionnés. La notion de stabilité d'un algorithme ou d'une méthode numérique est analogue à celle utilisée pour le problème $F(x,\ \theta) = 0$.

Si le problème $F(x,\ \theta) = 0$ admet une unique solution, alors il existe une application *f*, appelée résolvante, de l'ensemble des données sur celui des solutions, telle que :

$$x = f(\theta),\ \text{c'est-à-dire}\ F(f(\theta),\ \theta) = 0$$

Selon cette définition,

$$F(x + \delta x,\ \theta + \delta\theta) = 0\ \text{implique}\ x + \delta x = f(\theta + \delta\theta)$$

Si *f* est différentiable en θ et suivant le développement de Taylor,

$$f(\theta + \delta\theta) - f(\theta) = f'(\theta)\delta\theta + O(\|\delta\theta\|)\quad \text{pour}\ \ \delta\theta \to 0$$

En négligeant l'infiniment petit d'ordre le plus grand par rapport à $\|\delta\theta\|$, on déduit :

$$\kappa(\theta) \approx \|f'(\theta)\|\frac{\|\theta\|}{\|f(\theta)\|}\ \ \text{et}\ \kappa_a(\theta) \approx \|f'(\theta)\|$$

Ces estimations ont un grand intérêt pratique dans l'analyse des problèmes de la forme $F(f(\theta),\ \theta) = 0$, comme le montre l'exemple suivant (Quarteroni *et al.*, 2007).

Soit le système d'équations linéaire **Ax** = **b** dont l'inconnu est le vecteur **x** de dimensions $n \times 1$, **b** est un vecteur connu de même dimension et **A** la matrice carrée de dimension $n \times n$. Supposons que **A** est inversible (c'est-à-dire, son déterminant est différent de 0) dans ce cas *x* est la solution inconnue **x** et θ est les valeurs de la matrice **A** et du vecteur **b**, c'est-à-dire, $\theta = \{a_{ij},\ b_j,\ 1 \le i,\ j \le n\}$.

Supposons à présent une perturbation seulement au second membre **b**. Alors, $\theta \equiv \mathbf{b}$ et $\mathbf{x} \equiv f(\mathbf{b}) = \mathbf{A}^{-1}\mathbf{b}$ de sorte que, $f'(\mathbf{b}) = \mathbf{A}^{-1}$ ce qui entraîne :

$$\kappa(\theta) \simeq \frac{\|\mathbf{A}^{-1}\|\|\mathbf{b}\|}{\|\mathbf{A}^{-1}\mathbf{b}\|} = \frac{\|\mathbf{A}^{-1}\|\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \le \|\mathbf{A}^{-1}\|\|\mathbf{A}\| = \kappa(\mathbf{A})$$

$\kappa(\mathbf{A})$, est le conditionnement de la matrice **A** (voir chapitre 2 section 2.5), par conséquent, si le conditionnement de **A** est "petit", la résolution du système linéaire **Ax=b** est un problème stable par rapport aux perturbations du **b**.

3. Convergence et stabilité

Les méthodes numériques utilisées pour résoudre un problème approché conduisent à un résultat qui est toujours entaché d'erreur. Cette erreur doit être suffisamment petite pour que la solution numérique converge vers la solution réelle, dans ce cas la méthode est dite convergente. Si un raisonnement mathématique permet de montrer qu'une méthode diverge elle ne pourra en aucun cas être utilisée. En revanche, si la méthode est théoriquement convergente il se peut qu'en pratique elle est divergente. La vitesse de convergence est un facteur important de la qualité des méthodes numériques, si la vitesse de convergence est élevée, l'algorithme converge rapidement et le temps de calcul est moindre. Ces préoccupations de rapidité de convergence ont conduit à diversifier les modes de convergence et à chercher des processus optimaux.

La stabilité garantit que les erreurs ne s'amplifient pas au cours du déroulement de l'algorithme et que la méthode reste stable, à côté de cette stabilité numérique, il y a aussi la stabilité des solutions qui intervient dans les problèmes d'équations et qui est bien mise en évidence par les techniques perturbatrices. Lorsqu'un problème (P) admet une solution, il est intéressant d'envisager le problème perturbé, noté (P_ε), où ε est un petit paramètre et de se demander si les solutions du système perturbé sont voisines de la solution du système non perturbé.

Soit u une fonction à valeurs réelles définie sur un intervalle [a, b] et une subdivision $a=t_0<t_1<t_2<\cdots<t_n=b$ où, $h_i=t_{i+1}-t_i$, soit h la plus grande valeur des pas de la subdivision, c'est-à-dire,

$$h=\sup_i\left(h_i\right)$$

On suppose que la fonction u est dotée d'une réalisation numérique comme un processus ou schéma de discrétisation qui s'exprime sous la forme :

$$u_{i+1}=\psi\left(h_1,\ h_2,\ \ldots,\ h_i,\ u_1,\ u_2,\ \ldots,\ u_i\right)$$

- L'*erreur de consistance* relative à la fonction $u(t)$ est exprimée par :

$$e_i=u\left(t_i\right)-u_i$$

- L'*erreur globale* est exprimée par :

$$e=\sup_{0\le i\le n}\left|u\left(t_i\right)-u_i\right|$$

- Si $e\to 0$ quand $h\to 0$, la méthode est dite *convergente.*

- Si $\sum_{i=0}^{n}\left|e_i\right|\to 0$ quand $h\to 0$, la méthode est dite *consistante.*

- Si $\lim_{n\to\infty}\dfrac{\left|e_{n+1}\right|}{\left|e_n\right|^p}$ existe, la méthode est d'ordre p,. L'erreur de consistance est de l'ordre $O\left(h^p\right)$.

La méthode est stable si pour toutes suites voisines u_{i+1} et v_{i+1} vérifiant :

$$u_{i+1}=\psi\left(h_1,\ h_2,\ \ldots,\ h_i,\ u_1,\ u_2,\ \ldots,\ u_i\right)$$

$$v_{i+1} = \psi(h_1,\ h_2,\ \ldots,\ h_i,\ v_1,\ v_2,\ \ldots,\ v_i) + \varepsilon_i$$

Il existe une constante *M* dite de stabilité satisfaisant l'inégalité :

$$\sup_{0\le i\le n} |v_i - u_i| \le M \sum_{i=0}^{n} |e_i|$$

4. Exercices résolus

Exercice 1

L'approximation de la valeur π avec 7 chiffres après la virgule est 3.1415926. Soit 22/7 qui peut être considéré comme une approximation de π, calculer dans ce cas l'erreur absolue et l'erreur relative.

La valeur exacte π est $x = 3.1415926$ et la valeur approximative de π est $\tilde{x} = 22/7 = 3.1428571$ alors l'erreur absolue est :

$$E_a = |x - \tilde{x}| = |3.1415926 - 3.1428571| = |-0.0012645| = 0.0012645$$

et l'erreur relative est :

$$E_r = \left|\frac{x - \tilde{x}}{x}\right| = \left|\frac{3.1415926 - 3.1428571}{3.1415926}\right| = \frac{0.0012645}{3.1415926} = 0.000402502 = 0.402502\%$$

Exercice 2

En utilisant le développement de Taylor, calculer le nombre *e* avec 6 chiffres après la virgule.

Le développement en série de Taylor de la fonction e^x est exprimé par :

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots + \frac{x^n}{n!} + \cdots$$

alors,

$$e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots + \frac{1}{n!} + \cdots$$

or,

$$\frac{1}{2!} = 0.5,\quad \frac{1}{3!} = 0.166667,\quad \frac{1}{4!} = 0.0411667,\ \ldots,\ \frac{1}{9!} = 0.0000027,\quad \frac{1}{10!} = 0.00000027$$

soit,

$$e \approx 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots + \frac{1}{9!} + \cdots = 2.71828$$

Exercice 3

Calculer le conditionnement des expressions suivantes :

$$x - a^\theta = 0,\ a > 0 \text{ et } x = 1 + \theta$$

De l'équation $x - a^\theta = 0,\ \Rightarrow\ x = a^\theta = f(\theta)$, d'après la formule du conditionnement soit,

$$\kappa(\theta) \approx |f'(\theta)| \frac{|\theta|}{|f(\theta)|} = |a^\theta \log a| \frac{|\theta|}{|a^\theta|} = |\theta||\log a|$$

De l'équation $x = 1 + \theta = f(\theta)$, alors

$$\kappa(\theta)\approx|f'(\theta)|\frac{|\theta|}{|f(\theta)|}=\frac{|\theta|}{|1+\theta|}$$

Exercice 4

Etudier si le problème suivant st bien posé et calculer son conditionnement $\left(\text{en norme } \|\,.\,\|_\infty\right)$ en fonction de la donnée θ : trouver *x* et *y* tels que,

$$\begin{cases} x+\theta y=1, \\ \theta x+y=0. \end{cases}$$

Du système,

$$\begin{cases} x+\theta y=1, \\ \theta x+y=0. \end{cases} \Rightarrow \begin{pmatrix} x \\ y \end{pmatrix}=\begin{pmatrix} 1 & \theta \\ \theta & 1 \end{pmatrix}^{-1}\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

soit,

$$\mathbf{A}^{-1}=\begin{pmatrix} a & b \\ c & d \end{pmatrix}=\begin{pmatrix} 1 & \theta \\ \theta & 1 \end{pmatrix}^{-1} \Rightarrow \begin{pmatrix} 1 & \theta \\ \theta & 1 \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix}=\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

c'est-à-dire,

$$\begin{cases} a+\theta c=1 \\ b+\theta d=0 \\ \theta a+c=0 \\ \theta b+d=1 \end{cases} \Rightarrow\ a=d=1/\left(1-\theta^2\right),\ b=c=-\theta/\left(1-\theta^2\right)$$

donc,

$$\mathbf{A}^{-1}=\begin{pmatrix} 1/\left(1-\theta^2\right) & -\theta/\left(1-\theta^2\right) \\ -\theta/\left(1-\theta^2\right) & 1/\left(1-\theta^2\right) \end{pmatrix}=\frac{1}{\left(1-\theta^2\right)}\begin{pmatrix} 1 & -\theta \\ -\theta & 1 \end{pmatrix}$$

Le conditionnement de la matrice est peut-être exprimé par :

$$\kappa_\infty(\mathbf{A})=\left\|\mathbf{A}^{-1}\right\|_\infty\|\mathbf{A}\|_\infty$$

or,

$$\|\mathbf{A}\|_\infty=1+|\theta| \text{ et } \left\|\mathbf{A}^{-1}\right\|_\infty=\frac{1+|\theta|}{|1-\theta^2|}$$

Soit,

$$\kappa_\infty(\mathbf{A})=\frac{\left(1+|\theta|\right)^2}{|1-\theta^2|}=\frac{\left(1+|\theta|\right)\left(1+|\theta|\right)}{|1-\theta||1+\theta|}\approx\frac{\left(1+|\theta|\right)\left(1+|\theta|\right)}{|1-\theta|\left(1+|\theta|\right)}=\frac{1+|\theta|}{|1-\theta|}$$

La solution du système est exprimée par :

$$x=\frac{1}{1-\theta^2},\ y=-\frac{\theta}{1-\theta^2}$$

Exercice 5

Etudier le conditionnement de la formule $x=-\theta\pm\sqrt{\theta^2+\vartheta}$ donnant la solution d'une équation du second degré $x^2+2\theta x-\vartheta$ par rapport aux perturbations des paramètres θ et ϑ séparément.

De l'équation $x=-\theta\pm\sqrt{\theta^2+\vartheta}\equiv f(\theta,\ \vartheta)$ alors, le conditionnement par rapport aux perturbations des paramètres θ et ϑ séparément sont exprimés par :

$$\kappa(\theta)\approx\left|\frac{\partial f(\theta,\ \vartheta)}{\partial\theta}\right|\frac{|\theta|}{|f(\theta,\ \vartheta)|},\ \kappa(\vartheta)\approx\left|\frac{\partial f(\theta,\ \vartheta)}{\partial\vartheta}\right|\frac{|\vartheta|}{|f(\theta,\ \vartheta)|}$$

or,

$$\frac{\partial f(\theta,\ \vartheta)}{\partial\theta}=-1\pm\frac{\theta}{\sqrt{\theta^2+\vartheta}},\ \frac{\partial f(\theta,\ \vartheta)}{\partial\vartheta}=\pm\frac{1}{2\sqrt{\theta^2+\vartheta}}$$

soit,

$$\kappa(\theta)\approx\left|-1\pm\frac{\theta}{\sqrt{\theta^2+\vartheta}}\right|\frac{|\theta|}{\left|-\theta\pm\sqrt{\theta^2+\vartheta}\right|}=\left|\frac{-\sqrt{\theta^2+\vartheta}\pm\theta}{\sqrt{\theta^2+\vartheta}}\right|\frac{|\theta|}{\left|-\theta\pm\sqrt{\theta^2+\vartheta}\right|}\approx\frac{|\theta|}{\left|\sqrt{\theta^2+\vartheta}\right|}$$

$$\kappa(\vartheta)\approx\left|\pm\frac{1}{2\sqrt{\theta^2+\vartheta}}\right|\frac{|\vartheta|}{|x|}=\frac{|\vartheta|}{2|x|\sqrt{\theta^2+\vartheta}}$$

Exercice 6

Calculer le nombre de conditionnement des fonctions $y=\sqrt{x}$ et $y=x^3$.

Le nombre de conditionnement d'une fonction $y=f(x)$ est exprimé par :

$$\kappa(y)=\left|\frac{xf'(x)}{f(x)}\right|$$

Pour $y=\sqrt{x}$,

$$\kappa(y)=\left|\frac{x\left(\frac{1}{2}x^{-\frac{1}{2}}\right)}{\sqrt{x}}\right|=\frac{1}{2}\left|\frac{x}{\sqrt{x}\cdot\sqrt{x}}\right|=\frac{1}{2}$$

Puisque $\kappa\left(\sqrt{x}\right)<1$, la fonction $y=\sqrt{x}$ est bien conditionnée.

Pour $y=x^3$,

$$\kappa(y)=\left|\frac{x\left(3x^2\right)}{x^3}\right|=3\left|\frac{x^3}{x^3}\right|=3$$

Puisque $\kappa\left(x^3\right)>1$, la fonction $y=x^3$ est mal conditionnée.

Exercice 7

Etudier le conditionnement de la fonction $f(x)=1/(1-x)^2$ à $x=1.01$

Le nombre de conditionnement d'une fonction $f(x)$ est exprimé par :

$$\kappa\left(f\left(x\right)\right)=\left|\frac{xf'\left(x\right)}{f\left(x\right)}\right|=\left|\frac{x\left(\frac{-2}{\left(1-x\right)^3}\right)}{\frac{1}{\left(1-x\right)^2}}\right|$$

Pour $x=1.01$

$$\kappa\left(f\left(x=1.01\right)\right)=\left|\frac{x\left(\frac{-2}{\left(1-x\right)^3}\right)}{\frac{1}{\left(1-x\right)^2}}\right|=202$$

La fonction $f\left(x\right)$ pour $x=1.01$ est très mal conditionnée. La fonction est singulière a $x=1$ alors au voisinage de cette valeur il y a des changements brusques dans la valeur de la fonction qui rendent la fonction très mal conditionnée.

Chapitre 2.

Systèmes d'équations linéaires

1. Motivation

La résolution des systèmes d'équations linéaires est peut-être le problème numérique que l'on rencontre le plus fréquemment en ingénierie. Des questions aussi diverses que l'étude des réseaux électriques, hydrauliques en régime permanent, la résistance des matériaux, les modèles économétriques conduisent à des systèmes d'équations linéaires. Les problèmes aux limites, associés à des équations différentielles ou à des équations aux dérivées partielles, sont souvent discrétisés ce qui engendre encore des systèmes d'équations linéaires. Ce domaine s'appuie sur l'ensemble des connaissances de l'analyse matricielle en algèbre linéaire.

Les algorithmes de solution pour ces types de problèmes peuvent être directs ou itératifs. Dans les méthodes directes, la solution est obtenue en un nombre fixe d'étapes avec arrondissement de l'erreur, alors que les méthodes itératives sont basées sur l'amélioration successive de la solution dès la supposition initiale.

Un système d'équations linéaires s'écrit :

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

Sa résolution consiste à déterminer les inconnues $x_1, x_2, x_3, \ldots, x_n$ pour a_{ij} et b_i $i, j = 1, 2, \ldots, n$.

Sous forme matricielle le système d'équations linéaires s'écrit :

$$\mathbf{A}\mathbf{x} = \mathbf{b} \Leftrightarrow \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

avec,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \text{ et } \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

A titre d'exemple (Kharab et Guenther, 2019), soit le circuit électrique (Fig. 2.1). La première loi de Kirchhoff considère que la somme des tensions autour d'une boucle fermée est nulle. La seconde loi de Kirchhoff considère que la chute de tension aux bornes d'une résistance égale au produit du courant et de la résistance. Alors, d'après cette loi, soit le système linéaire :

$$R_6 i_1 + R_1 (i_1 - i_2) + R_2 (i_1 - i_3) = V_1 \quad ; \quad \text{Boucle } 1$$
$$R_3 i_2 + R_4 (i_2 - i_3) + R_1 (i_2 - i_1) = V_2 \quad ; \quad \text{Boucle } 2$$
$$R_5 i_3 + R_4 (i_3 - i_2) + R_2 (i_3 - i_1) = V_3 \quad ; \quad \text{Boucle } 3$$

qui se simplifiée sous la forme matricielle suivante :

$$\begin{pmatrix} R_1 + R_2 + R_6 & -R_1 & -R_2 \\ -R_1 & R_1 + R_3 + R_4 & -R_4 \\ -R_2 & -R_4 & R_2 + R_4 + R_5 \end{pmatrix} \begin{pmatrix} i_1 \\ i_2 \\ i_3 \end{pmatrix} = \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}$$

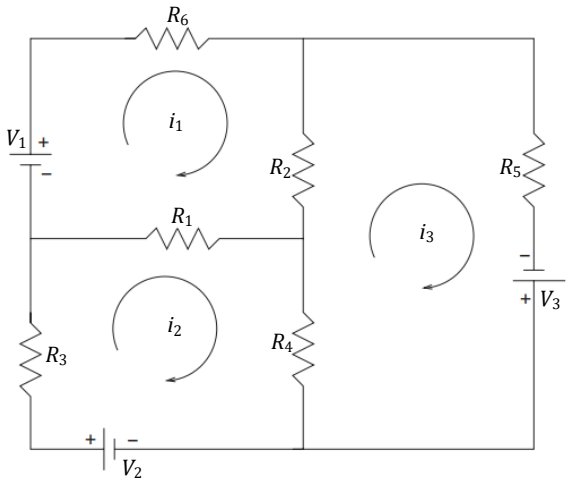


Fig. 2.1 : Circuit électrique.

Il est clair que le déterminant de la matrice est non nul, donc le système est de Cramer et peuvent être résolue par la règle de Cramer ou méthode des déterminants vu dans les cours d'algèbre linéaire.

Le nombre d'opérations à effectuer pour résoudre un système linéaire à l'aide de la règle de Cramer dépend de la méthode utilisée pour calculer le déterminant qui demandera d'avoir recours à un nombre de calculs de déterminants égal à la taille du système.

Les formules de Cramer sont parfaitement inefficaces. Le problème vient du calcul du déterminant. Un système d'équations linéaire de n équations et n inconnues, implique n calculs de déterminants de taille $(n-1)\times(n-1)$, suivis de n multiplications et $(n-1)$ additions. Pour chacun des déterminants de taille $(n-1)\times(n-1)$, il faudra recommencer avec $(n-1)$ déterminants de taille $(n-2)\times(n-2)$. Au total, le calcul d'un déterminant $n\times n$ prendra de l'ordre *n.n*! opérations. Il se trouve que le bon moyen algorithmique de calculer un déterminant est la méthode du pivot de Gauss.

2. Méthodes directes

2.1. Méthode d'élimination naïve de Gauss

Cette méthode est connue par les mathématiciens chinois, elle est référencée dans le livre chinois Jiuzhang suanshu (Les Neuf Chapitres sur l'art mathématique), dont elle constitue le huitième chapitre, sous le titre «Fang cheng» (la disposition rectangulaire). La méthode est présentée au moyen de dix-huit exercices. Dans son commentaire daté de 263, Liu Hui en attribue la paternité à Chang Ts'ang, chancelier de l'empereur de Chine au IIe siècle avant notre ère (Fig. 2.2). En Europe, cette méthode a été découverte et présentée sous forme moderne au XIXe siècle. En 1810, Carl Friedrich Gauss présente sa méthode des moindres carrés dans un livre étudiant le

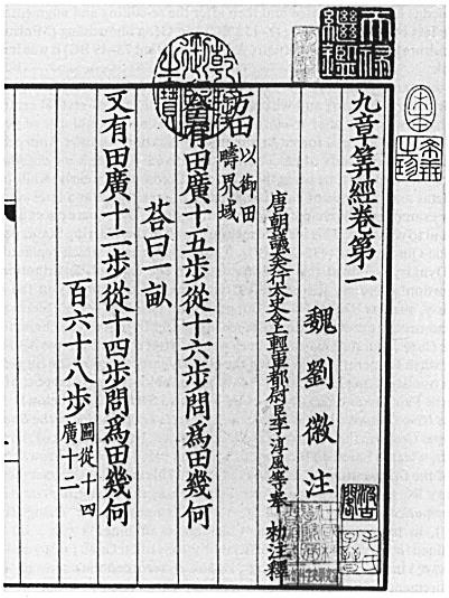


Fig. 2.2 : Les Neuf Chapitres sur l'art mathématique.

mouvement de l'astéroïde Pallas. Dans l'article 13 de ce livre, il décrit une méthode générale de résolution de système d'équations linéaires qui constitue l'essentiel de la méthode du pivot. En 1888, Wilhelm Jordan publie un livre de géodésie précisant comment utiliser cette méthode et adoptant une notation un peu différente. C'est grâce à ce dernier livre que cette méthode se diffusa dans tout l'Occident. Elle est aujourd'hui connue sous le nom d'élimination de Gauss-Jordan ou méthode du pivot de Gauss.

Considérons le système d'équations linéaire de dimension $n \times n$:

$$\mathbf{A}\,\mathbf{x} = \mathbf{b} \;\Leftrightarrow\; \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

A <u>*la première étape*</u>, on note par,

$$\mathbf{A}^{(1)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{pmatrix} \equiv \mathbf{A} \quad \text{et par} \quad \mathbf{b}^{(1)} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{pmatrix} \equiv \mathbf{b}$$

La méthode d'élimination de Gauss consiste à la réduction du système d'équations **Ax** = **b** a un système d'équations équivalent **Ux** = **b**, ou **U** est une matrice triangulaire supérieure et par conséquent la résolution facile par substitution.

Supposons que $a_{11}^{(1)} \neq 0$ soit,

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}$$

Multipliant la ligne inférieur *i* (c'est-à-dire, la première ligne de la matrice) par m_{i1} et faire soustraire de chaque facteur de x_i des lignes successives suivantes, c'est-à-dire,

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)} \equiv a_{ij}^{(1)} - a_{1j}^{(1)}\frac{a_{i1}^{(1)}}{a_{11}^{(1)}} \quad \text{et} \quad b_j^{(2)} = b_j^{(1)} - m_{i1}b_1^{(1)} \equiv b_j^{(1)} - \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}b_1^{(1)}, \;\; \left(j = 2,\ 3,\ \ldots,\ n\right)$$

Cette méthodologie à permet d'avoir de transformer le système précédent en:

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix}$$

En <u>*deuxième étape*</u>, le système d'équations linéaires est de dimension $\left(n-1\right)\times\left(n-1\right)$ c'est-à-dire :

$$\mathbf{A}^{(2)}\,\mathbf{x} = \mathbf{b}^{(2)} \;\Leftrightarrow\; \begin{pmatrix} a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_2^{(2)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix}$$

Si $a_{22}^{(2)} \neq 0$ et suivant la même méthodologie, le nouveau système d'équations de dimension $(n-2)\times(n-2)$ est :

$$\mathbf{A}^{(3)}\mathbf{x} = \mathbf{b}^{(3)} \quad \Leftrightarrow \quad \begin{pmatrix} a_{33}^{(3)} & a_{34}^{(3)} & \cdots & a_{3n}^{(3)} \\ a_{43}^{(3)} & a_{44}^{(3)} & \cdots & a_{4n}^{(3)} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n3}^{(3)} & a_{n4}^{(3)} & \cdots & a_{nn}^{(3)} \end{pmatrix} \begin{pmatrix} x_3 \\ x_4 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_3^{(3)} \\ b_4^{(3)} \\ \vdots \\ b_n^{(3)} \end{pmatrix}$$

A la <u>*k-ième étape*</u>, et si $a_{kk}^{(k)} \neq 0$,

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)} \equiv a_{ij}^{(k)} - a_{kj}^{(k)}\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \quad \text{et} \quad b_j^{(k+1)} = b_j^{(k)} - m_{ik}b_k^{(k)} \equiv b_j^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}b_k^{(k)}, \quad \left(i,\ j = k+1,\ \ldots,\ n\right)$$

Alors à l'étape *k,* le système d'équations linéaires devient sous la forme suivante :

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} & a_{1\ k+1}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2k}^{(2)} & a_{2\ k+1}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{kk}^{(k)} & a_{k\ k+1}^{(k)} & \cdots & a_{kn}^{(k)} \\ 0 & 0 & \cdots & 0 & a_{k+1\ k+1}^{(k+1)} & \cdots & a_{k+1\ n}^{(k+1)} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & a_{n\ k+1}^{(k+1)} & \cdots & a_{nn}^{(k+1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \\ x_{k+1} \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_k^{(k)} \\ b_{k+1}^{(k+1)} \\ \vdots \\ b_n^{(k+1)} \end{pmatrix}$$

et ainsi de suite…

A la <u>(*n*-1)*-ième et la dernière étape*</u>, le système d'équations linéaires devient :

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} & a_{1\ k+1}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2k}^{(2)} & a_{2\ k+1}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{kk}^{(k)} & a_{k\ k+1}^{(k)} & \cdots & a_{kn}^{(k)} \\ 0 & 0 & \cdots & 0 & a_{k+1\ k+1}^{(k+1)} & \cdots & \vdots \\ \vdots & \vdots & & \vdots & \vdots & a_{n-1\ n-1}^{(n-1)} & a_{n-1\ n}^{(n-1)} \\ 0 & 0 & \cdots & 0 & \cdots & 0 & a_{nn}^{(n)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \\ x_{k+1} \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_k^{(k)} \\ b_{k+1}^{(k+1)} \\ \vdots \\ b_n^{(n)} \end{pmatrix}$$

en termes d'équations, s'écrit :

$$\left.\begin{aligned} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1k}^{(1)}x_k + \cdots a_{1n}^{(1)}x_n &= b_1^{(1)} \\ a_{22}^{(2)}x_2 + \cdots + a_{2k}^{(2)}x_k + \cdots a_{2n}^{(2)}x_n &= b_2^{(2)} \\ \cdots \qquad\qquad \cdots \qquad\qquad &= \cdots \\ a_{n-1\ n-1}^{(n-1)}x_{n-1} + a_{n-1\ n}^{(n-1)}x_n &= b_{n-1}^{(n-1)} \\ a_{nn}^{(n-1)}x_n &= b_n^{(n)} \end{aligned}\right\}$$

La dernière équation du système permet d'avoir la valeur de la solution x_n c'est-à-dire :

$$x_n = \frac{b_n^{(n)}}{a_{nn}^{(n-1)}}$$

par conséquent,

$$x_{n-1} = \frac{b_{n-1}^{(n-1)} - a_{n-1\ n}^{(n-1)} x_n}{a_{n-1\ n-1}^{(n-1)}}$$

$$x_i = \frac{b_i^{(i)} - \left(a_{ii}^{(i)} x_{i+1} + \cdots + a_{in}^{(i)} x_n\right)}{a_{ii}^{(i)}} \equiv \frac{b_i^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j}{a_{ii}^{(i)}},\ i = n-2,\ n-3,\ \ldots,\ 1$$

L'algorithme d'élimination de Gauss tel qu'il est marche très bien si à chaque étape *k* le pivot $a_{kk}^{(k)} \neq 0$. Dans le cas contraire l'algorithme s'échoue. Pour résoudre de tel problème, il faut faire des permutations entre les lignes de la matrice. De telle méthodologie sera décrite dans la prochaine section.

Pour une étape *k*, le nombre d'opérations en termes de multiplications et soustractions pour calculer $a_{ij}^{(k)}$ et $b_j^{(k)}$ sont :

$$\left(n-k\right)^2 + \left(n-k\right) \text{ et } \left(n-k\right)$$

Alors le nombre d'opérations total est calculé pour $\left(n-1\right)$ étapes est :

$$N_a = \sum_{k=1}^{n-1}\left(n-k\right)^2 + \sum_{k=1}^{n-1}\left(n-k\right) \text{ pour tous les } a_{ij}^{(k)},\ \forall\ k = 1,\ 2,\ 3,\ \ldots,\ n-1$$

et,

$$N_b = \sum_{k=1}^{n-1}\left(n-k\right) \text{ pour tous les } b_j^{(k)},\ \forall\ k = 1,\ 2,\ 3,\ \ldots,\ n-1$$

or,

$$\sum_{k=1}^{n-1}\left(n-k\right)^2 = \sum_{k=1}^{n-1}\left(n^2 - 2kn + k^2\right) = n^2\sum_{k=1}^{n-1}1 - 2n\sum_{k=1}^{n-1}k + \sum_{k=1}^{n-1}k^2$$

$$= n^2\left(n-1\right) - 2n\frac{\left(n-1\right)n}{2} + \frac{n\left(n+1\right)\left(2n+1\right)}{6} - n^2 = \frac{2n^3 - 3n^2 + n}{6}$$

et,

$$\sum_{k=1}^{n-1}\left(n-k\right) = n\sum_{k=1}^{n-1}1 - \sum_{k=1}^{n-1}k = n\left(n-1\right) - \frac{\left(n-1\right)n}{2} = \frac{n^2 - n}{2}$$

alors,

$$N_a = \frac{n^3 - n}{3} \quad \text{et} \quad N_b = \frac{n^2 - n}{2}$$

Le nombre d'opérations pour déterminer les solutions b_i, $i = 1,\ 2,\ 3,\ \ldots,\ n$ est :

$$N_x = \frac{n\left(n+1\right)}{2}$$

Alors, le nombre d'opérations pour déterminer une solution du système d'équations linéaires avec la méthode d'élimination naïve de Gauss est :

$$N_a + N_b + N_x = \frac{n^3}{3} + n^2 - \frac{n}{3} = n^3\left(\frac{1}{3} + \frac{1}{n} - \frac{1}{3n^2}\right)$$

Qui est de troisième ordre $O\left(n^3\right)$ c'est-à-dire, si n augmente le nombre d'opérations est proche de $n^3/3$.

Soit la fonction **`NaiveGaussElimination`** sert à déterminer une solution du système d'équations linéaires avec la méthode d'élimination naïve de Gauss et donne la matrice **A** et le vecteur **b** après élimination.

```
function NaiveGaussElimination(A, b)
disp('Résolution du système linéaire [A]x=[b]')
n=numel(b);
x=zeros(n,1);
for k=1:n-1
   for i=k+1:n
       m=A(i,k)/A(k,k);
      for j=k+1:n
         A(i,j)=A(i,j)-m*A(k,j);
      end
    A(i,k)=m;
   b(i)=b(i)-m*b(k);
   end
end
    x(n)=b(n)/A(n,n);
for i=n-1:-1:1
    S=b(i);
  for j=i+1:n
      S=S-A(i,j)*x(j);
  end
    x(i)=S/A(i,i);
end
 for i=1:n
    for j=1:n
       if (j<i) A(i,j)=0; end
    end
 end
disp('La matrice A triangulaire après élimination :')
A
disp('Le vecteur b après élimination :')
b
disp('La solution du système est:')
x
end
```

Soit à titre d'exemple le système d'équations :

$$\mathbf{Ax} = \mathbf{b} \quad \Leftrightarrow \quad \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 1 & 5 \\ -1 & 1 & -5 & 3 \\ 3 & 1 & 7 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 10 \\ 31 \\ -2 \\ 18 \end{pmatrix}$$

Introduite dans MATLAB comme :

```
>> A=[1 1 1 1 ; 2 3 1 5 ; -1 1 -5 3 ; 3 1 7 -2];
>> b=[10;31;-2;18];
```

L'application de la fonction **NaiveGaussElimination** conduit au résultat :

```
>> NaiveGaussElimination(A,b)
Résolution du système linéaire [A]x=[b]
La matrice A triangulaire après élimination :
A =
     1     1     1     1
     0     1    -1     3
     0     0    -2    -2
     0     0     0    -1
Le vecteur b après élimination :
b =
    10
    11
   -14
    -4
La solution du système est:
x =
     1
     2
     3
     4
```

Soit l'exemple du système d'équations :

$$\mathbf{A}\mathbf{x} = \mathbf{b} \;\Leftrightarrow\; \begin{pmatrix} 1 & 1 & 7 & 8 \\ -3 & -3 & -1 & 6 \\ 0 & 2 & 2 & 7 \\ 5 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \\ 2 \\ 1 \end{pmatrix}$$

Introduite dans MATLAB comme :

```
>> A=[1 1 7 8 ; -3 -3 -1 6 ; 0 2 2 7 ; 5 1 1 0];
>> b=[4;1;2;1];
```

L'application de la fonction **NaiveGaussElimination** conduit au résultat :

```
>> NaiveGaussElimination(A,b)
Résolution du système linéaire [A]x=[b]
La matrice A triangularisée après élimination :
A =
     1     1     7     8
     0     0    20    30
     0     0  -Inf  -Inf
     0     0     0   NaN
Le vecteur b après élimination :
b =
     4
    13
  -Inf
   NaN
La solution du système est:
x =
   NaN
   NaN
   NaN
   NaN
```

L'exécution de la fonction n'a pas permet de donner une solution du système d'équations, en plus elle montre que la deuxième étape d'élimination du programme est arrêtée, c'est-à-dire le pivot $a_{22}^{(2)} = 0$.

2.2. Méthode d'élimination de Gauss avec pivot

Le dernier exemple traité dans le paragraphe précédent contient un pivot nul ($a_{22}^{(2)} = 0$), dans ce cas le procédé d'élimination de Gauss précédent même pour le cas ou $a_{kk}^{(k)} \approx 0$ peut conduire à des résultats désastreux en arithmétique flottante.

Soit, $\mathbf{A}_k = \left[a_{ik}^{(k)} \right]$ la matrice obtenue à l'étape k ($\mathbf{A}_1 \equiv \mathbf{A}$), le pivot partiel choisis vérifiant :

$$\left| a_{ik}^{(k)} \right| = \max_{k \le p \le n} \left| a_{pk}^{(k)} \right|$$

Ou pivot total vérifiant :

$$\left| a_{ij}^{(k)} \right| = \max_{\substack{k \le p \le n \\ k \le q \le n}} \left| a_{pq}^{(k)} \right|$$

Dans le cas du pivot total, il faut également effectuer des permutations des colonnes. En langage matricielle, ceci se traduit par la multiplication à droite par une matrice de permutation. Ainsi, la matrice triangulaire supérieure $\mathbf{U}$ obtenue correspond à :

$$\mathbf{U} = \mathbf{E}_{n-1}\, \mathbf{P}_{n-1}\, \mathbf{E}_{n-2}\, \mathbf{P}_{n-2}\, \ldots\, \mathbf{E}_2\, \mathbf{P}_2\, \mathbf{E}_1\, \mathbf{P}_1\, \mathbf{A}\, \mathbf{\Pi}_1\, \mathbf{\Pi}_2\, \ldots\, \mathbf{\Pi}_{n-1}$$

$\mathbf{P}_i$ et $\mathbf{\Pi}_i$ sont les matrices de permutation respectivement des lignes et des colonnes et les $\mathbf{E}_i$ les matrices Gaussiennes.

Le choix du pivot total est toujours plus sûr, l'expérience montre que le choix du pivot partiel est suffisamment robuste dans la plupart des cas.

La fonction **`Pivot2Gauss`** sert à déterminer une solution du système d'équations linéaires avec la méthode d'élimination de Gauss avec pivot partiel.

```
function x=Pivot2Gauss(A,b)
 n=length(b);
  x=zeros(n,1);
 for i=1:n
   d(i)=i;
    smax=0;
     for j=1:n
      smax=max(smax,abs(A(i,j)));
     end
  c(i)=smax;
 end
 for k=1:n-1
    rmax=0;
    for i=k:n
        R=abs(A(d(i),k))/c(d(i));
      if (R>rmax)
          j=i;
```

```
            rmax=R;
         end
      end
     dk=d(j);
        d(j)=d(k);
           d(k)=dk;
      for i=k+1:n
        m=A(d(i),k)/A(dk,k);
         for j=k+1:n
           A(d(i),j)=A(d(i),j)-m*A(dk,j);
         end
        A(d(i),k)=m;
      end
  end
% ----- Substitution -----
  for k=1:n-1
    for i=k+1:n
      b(d(i))=b(d(i))-b(d(k))*A(d(i),k);
    end
  end
x(n)=b(d(n))/A(d(n),n);
  for i=n-1:-1:1
   S=b(d(i));
      for j=i+1:n
        S=S-A(d(i),j)*x(j);
      end
   x(i)=S/A(d(i),i);
  end
  for i=1:n
    M(i,:)=A(d(i),:);
  end
  for i=1:n
     for j=1:n
      if (j<i) M(i,j)=0; end
     end
  end
end
```

L'utilisation de l'exemple précédent avec cette fonction, c'est-à-dire :

```
>> A=[1 1 7 8 ; -3 -3 -1 6 ; 0 2 2 7 ; 5 1 1 0];
>> b=[4;1;2;1];
>> x=Pivot2Gauss(A,b)
```

Permet d'avoir la solution du système,

```
x =

    0.1658
   -0.1237
    0.2947
    0.2368
```

2.3. Méthode de décomposition LU

Cette méthode est appelée aussi le schéma de Khaletski (Démidovitch et Maron, 1979). Le système d'équations linéaires s'écrit sous la forme :

$$\mathbf{A}\mathbf{x}=\mathbf{b} \;\Leftrightarrow\; \mathbf{L}\mathbf{U}\mathbf{x}=\mathbf{b}$$

où,

$$\mathbf{L}=\begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix} \text{ et } \mathbf{U}=\begin{pmatrix} 1 & u_{12} & \cdots & u_{1n} \\ 0 & 1 & \cdots & u_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

L, la matrice triangulaire inférieur et **U**, la matrice triangulaire supérieure à diagonale unité. Les éléments l_{ij} et u_{ij} se définissent d'après les formules :

$$\begin{cases} l_{i1}=a_{i1}, \;\; (i=1,\; 2,\; \ldots,\; n) \\ l_{ij}=a_{ij}-\sum_{k=1}^{j-1} l_{ik}u_{kj}, \;\; (i\geq j>1) \end{cases} \text{ et } \begin{cases} u_{1j}=\dfrac{a_{1j}}{l_{11}}, \;\; (j=1,\; 2,\; \ldots,\; n) \\ u_{ij}=\dfrac{1}{l_{ii}}\left(a_{ij}-\sum_{k=1}^{i-1} l_{ik}u_{kj}\right), \;\; (1<i<j) \end{cases}$$

Le vecteur solution **x** peut être calculée d'après la chaîne d'équations :

$$\mathbf{L}\mathbf{y}=\mathbf{b}, \;\; \mathbf{U}\mathbf{x}=\mathbf{y}$$

Puisque les deux matrices **L** et **U** étant triangulaires, alors la résolution du système d'équations est sans peine :

$$\begin{cases} y_1=\dfrac{b_1}{l_{11}}, \\ y_i=\dfrac{1}{l_{ii}}\left(b_i-\sum_{k=1}^{i-1} l_{ik}y_k\right), \;\; (i>1) \end{cases} \text{ et } \begin{cases} x_n=y_n, \\ x_i=y_i-\sum_{k=i+1}^{n} u_{ik}x_k, \;\; (i=n-1,\; n-2,\; \ldots,\; 1) \end{cases}$$

Si la matrice **A** est symétrique, c'est-à-dire $a_{ij}=a_{ji}$ alors,

$$u_{ij}=\frac{l_{ji}}{l_{ii}}, \;\; (i<j)$$

Soit la fonction **DecompositionLU** qui sert à la décomposition de la matrice **A** en deux matrice **L** et **U** et détermine le vecteur **y** puis le vecteur **x** solution du système $\mathbf{A}\mathbf{x}=\mathbf{b}$.

```
function [x , L, U, y]= DecompositionLU(A,b)
% Résolution du système Ax=b avec décomposition LU
n=numel(b);
y=zeros(n,1); x=zeros(n,1);
% Construction des matrices L et U
  for i=1:n
    U(i,i)=1;
  end
L(1,1)=A(1,1)/U(1,1);
  for j=2:n
     L(j,1)=A(j,1)/U(1,1); U(1,j)=A(1,j)/L(1,1);
  end
  for i=2:n-1
```

```
    S=0;
   for k=1:i-1
     S=S+U(k,i)*L(i,k);
   end
    L(i,i)=(A(i,i)-S)/U(i,i);
    for j=i+1:n
        S=0;
         for k=1:i-1
           S=S+U(k,i)*L(j,k);
         end
     L(j,i)=(A(j,i)-S)/U(i,i);
     S=0;
      for k=1:i-1
        S=S+U(k,j)*L(i,k);
      end
     U(i,j)=(A(i,j)-S)/L(i,i);
    end
  end
 S=0;
  for k=1:n-1
   S=S+U(k,n)*L(n,k);
  end
 L(n,n)=(A(n,n)-S)/U(n,n);
% Détermination du vecteur y solution de Ly=b
y(1)=b(1)/L(1,1);
  for i=2:n
    S=b(i);
    for j=1:i-1
      S=S-L(i,j)*y(j);
    end
   y(i)=S/L(i,i);
  end
% Détermination du vecteur x solution du système Ux=b et Ax=b.
x(n)=y(n)/U(n,n);
 for i=n-1:-1:1
    S=y(i);
     for j=i+1:n
       S=S-U(i,j)*x(j);
     end
  x(i)=S/U(i,i);
 end
end
```

A titre d'exemple, soit le système :

$$\begin{pmatrix} -1 & 2 & -1 & 0 & -4 \\ 1 & 2 & 0 & 3 & 0 \\ 0 & -3 & 1 & 1 & 2 \\ 1 & 0 & 2 & -1 & 3 \\ 2 & -2 & 2 & -2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 4 \\ 0 \\ 3 \end{pmatrix}$$

Les données de la matrice **A** et le vecteur **b** sont introduites ainsi,

```
>> A=[-1 2 -1 0 -4;1 2 0 3 0;0 -3 1 1 2;1 0 2 -1 3;2 -2 2 -2 1];
>> b=[1;2;4;0;3];
```

alors la matrice **L** est,

```
>> [~, L, ~, ~]= DecompositionLU(A,b)
L =
   -1.0000         0         0         0         0
    1.0000    4.0000         0         0         0
         0   -3.0000    0.2500         0         0
    1.0000    2.0000    1.5000  -22.0000         0
    2.0000    2.0000    0.5000  -10.0000   -6.1818
```

La matrice **U** est,

```
>> [~, ~, U, ~]= DecompositionLU(A,b)
U =
    1.0000   -2.0000    1.0000         0    4.0000
         0    1.0000   -0.2500    0.7500   -1.0000
         0         0    1.0000   13.0000   -4.0000
         0         0         0    1.0000   -0.3182
         0         0         0         0    1.0000
```

La solution **y** est,

```
>> [~, ~, ~, y]= DecompositionLU(A,b)
y =
   -1.0000
    0.7500
   25.0000
    1.7273
   -1.3382
```

La solution **x** du système est,

```
>> x= DecompositionLU(A,b)
x =
   -0.1397
   -0.8824
    2.7279
    1.3015
   -1.3382
```

2.4. Méthode de décomposition QR

La matrice **A** du système d'équations linéaires $\mathbf{A}\mathbf{x} = \mathbf{b}$ est décomposé sous la forme :

$$\mathbf{A} = \mathbf{Q}\mathbf{R}$$

où **Q**, une matrice orthogonale (c'est-à-dire, $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_n$) et **R**, une matrice triangulaire supérieure. Ce type de décomposition est souvent utilisé pour le calcul de la solution des systèmes linéaires non carrés, notamment pour déterminer le pseudo-inverse d'une matrice.

Le vecteur solution **x** peut être calculé d'après la chaîne d'équations :

$$\mathbf{A}\mathbf{x} = \mathbf{b} \;\Leftrightarrow\; \mathbf{Q}\mathbf{R}\mathbf{x} = \mathbf{b}$$

considérant,

$$\mathbf{R}\mathbf{x} = \mathbf{y}$$

alors,

$$\mathbf{Q}\mathbf{y} = \mathbf{b} \;\Rightarrow\; \mathbf{Q}^T\mathbf{Q}\mathbf{y} = \mathbf{Q}^T\mathbf{b}$$

or,

$$\mathbf{Q}^T\mathbf{Q}=\mathbf{I}_n$$

alors,

$$\mathbf{y}=\mathbf{Q}^T\mathbf{b}$$

par conséquent,

$$\mathbf{x}=\mathbf{R}^{-1}\mathbf{y}$$

c'est-à-dire,

$$\begin{cases} x_n=\dfrac{y_n}{r_{nn}}, \\ x_i=\dfrac{1}{r_{ii}}\left(y_i-\displaystyle\sum_{k=i+1}^{n} r_{ik}x_k\right), \ \left(i=n-1,\ n-2,\ \ldots,\ 1\right) \end{cases}$$

La décomposition de la matrice **A** en produit **QR** peut se faire par la méthode de Gram-Schmidt, de Gram-Schmidt modifiée, de Householder ou de Givens (Stewart, 1998, Higham, 2002). Chaque méthode a des avantages et des inconvénients, la décomposition QR n'étant pas unique, les différentes méthodes produiront des résultats différents. Soit les fonctions **GramSchmidt** et **ModifiedGramSchmidt** pour la décomposition de la matrice **A** en **Q** de décomposition QR de Gram-Schmidt et Gram-Schmidt modifiée :

```
function [Q,R]=GramSchmidt(A)
[n,p] = size(A);
Q = zeros(n,p);
R = zeros(p,p);
for k = 1:p
 Q(:,k) = A(:,k);
    if k ~= 1
      R(1:k-1,k) = Q(:,k-1)'*Q(:,k);
      Q(:,k) = Q(:,k)-Q(:,1:k-1)*R(1:k-1,k);
    end
     R(k,k) = norm(Q(:,k));
    Q(:,k) = Q(:,k)/R(k,k);
end
end
```

```
function [Q,R] = ModifiedGramSchmidt(A)
[n,p] = size(A);
  Q = zeros(n,p);
    R = zeros(p,p);
   for k = 1:p
        Q(:,k) = A(:,k);
      for i = 1:k-1
           R(i,k) = Q(:,i)'*Q(:,k);
           Q(:,k) = Q(:,k) - R(i,k)*Q(:,i);
      end
        R(k,k) = norm(Q(:,k))';
        Q(:,k) = Q(:,k)/R(k,k);
   end
end
```

Pour la matrice **A**, avec ;

$$\mathbf{A}=\begin{pmatrix}3 & 5 & 2\\ 1 & 9 & 7\\ 9 & 0 & 1\end{pmatrix}$$

Les fonction **`GramSchmidt`** et **`ModifiedGramSchmidt`** donnent les résultats suivantes :

```
>> [Q,R]=GramSchmidt(A)

Q =

    0.3145    0.4216   -0.6166
    0.1048    0.8751    0.0844
    0.9435   -0.2378   -0.7827

R =

    9.5394    2.5159    6.7309
         0    9.9835    6.7309
         0         0    4.7910
```

```
>> [Q,R]=ModifiedGramSchmidt(A)

Q =

    0.3145    0.4216   -0.8505
    0.1048    0.8751    0.4725
    0.9435   -0.2378    0.2310

R =

    9.5394    2.5159    2.3062
         0    9.9835    6.7309
         0         0    1.8375
```

2.5. Conditionnement

Soit l'exemple :

$$\begin{pmatrix}1.00 & 2.00\\ 0.49 & 0.99\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}=\begin{pmatrix}3.00\\ 1.47\end{pmatrix}$$

dont la solution est $\mathbf{x}=\begin{pmatrix}3,\ 0\end{pmatrix}^T$. Si la composante 0.49 est remplacé par 0.48 c'est-à-dire une légère différence de 0.01, le système dans ce cas à la solution $\mathbf{x}=\begin{pmatrix}1,\ 1\end{pmatrix}^T$. Alors, la difficulté de résolution d'un système linéaire en plus qu'elle tient effet de sa taille, elle tient aussi aux perturbations des coefficients de la matrice **A** ou du vecteur **b**.

En plus du système $\mathbf{A}\mathbf{x}=\mathbf{b}$, soit le système perturbé dont le vecteur **b** a subi une petite modification $\Delta\mathbf{b}$, ce qui entrainé une petite variation de la solution, c'est-à-dire :

$$\mathbf{A}\left(\mathbf{x}+\Delta\mathbf{x}\right)=\mathbf{b}+\Delta\mathbf{b}$$

d'où,

$$\mathbf{A}\mathbf{x}+\mathbf{A}\Delta\mathbf{x}=\mathbf{b}+\Delta\mathbf{b}\ \Rightarrow\ \mathbf{A}\Delta\mathbf{x}=\Delta\mathbf{b}\ \Rightarrow\ \Delta\mathbf{x}=\mathbf{A}^{-1}\Delta\mathbf{b}$$

alors,

$$\left\|\Delta\mathbf{x}\right\|\le\left\|\mathbf{A}^{-1}\right\|\left\|\Delta\mathbf{b}\right\|$$

Sachant que,

$$\mathbf{b}=\mathbf{A}\mathbf{x}\ \Rightarrow\ \left\|\mathbf{b}\right\|\le\left\|\mathbf{A}\right\|\left\|\mathbf{x}\right\|$$

soit,

$$\left\|\Delta\mathbf{x}\right\|\left\|\mathbf{b}\right\|\le\left\|\mathbf{A}^{-1}\right\|\left\|\Delta\mathbf{b}\right\|\left\|\mathbf{A}\right\|\left\|\mathbf{x}\right\|\ \Rightarrow\ \frac{\left\|\Delta\mathbf{x}\right\|}{\left\|\mathbf{x}\right\|}\le\left\|\mathbf{A}^{-1}\right\|\cdot\left\|\mathbf{A}\right\|\frac{\left\|\Delta\mathbf{b}\right\|}{\left\|\mathbf{b}\right\|}\ \text{ ou }\ \frac{\left\|\Delta\mathbf{x}\right\|}{\left\|\mathbf{x}\right\|}\le\kappa\left(\mathbf{A}\right)\frac{\left\|\Delta\mathbf{b}\right\|}{\left\|\mathbf{b}\right\|}$$

Le nombre $\kappa\left(\mathbf{A}\right)$, est appelée le conditionnement de la matrice **A**, qui mesure la sensibilité de l'erreur relative sur la solution aux variations du vecteur **b**. Ce nombre peut être retrouvé si la perturbation est au niveau de la matrice **A**, c'est-à-dire :

$$\left(\mathbf{A}+\Delta\mathbf{A}\right)\left(\mathbf{x}+\Delta\mathbf{x}\right)=\mathbf{b}$$

alors,

$$\mathbf{A}\mathbf{x}+\mathbf{A}\Delta\mathbf{x}+\Delta\mathbf{A}\left(\mathbf{x}+\Delta\mathbf{x}\right)=\mathbf{b}\ \Rightarrow\ \mathbf{A}\Delta\mathbf{x}+\Delta\mathbf{A}\left(\mathbf{x}+\Delta\mathbf{x}\right)=\mathbf{0}\ \Rightarrow\ \mathbf{A}\Delta\mathbf{x}=-\Delta\mathbf{A}\left(\mathbf{x}+\Delta\mathbf{x}\right)$$

où,

$$\Delta\mathbf{x}=-\mathbf{A}^{-1}\Delta\mathbf{A}\left(\mathbf{x}+\Delta\mathbf{x}\right)\ \Rightarrow\ \left\|\Delta\mathbf{x}\right\|\le\left\|\mathbf{A}^{-1}\right\|\left\|\Delta\mathbf{A}\right\|\left\|\left(\mathbf{x}+\Delta\mathbf{x}\right)\right\|\ \Rightarrow\ \frac{\left\|\Delta\mathbf{x}\right\|}{\left\|\left(\mathbf{x}+\Delta\mathbf{x}\right)\right\|}\le\left\|\mathbf{A}^{-1}\right\|\left\|\mathbf{A}\right\|\frac{\left\|\Delta\mathbf{A}\right\|}{\left\|\mathbf{A}\right\|}$$

donc,

$$\frac{\left\|\Delta\mathbf{x}\right\|}{\left\|\left(\mathbf{x}+\Delta\mathbf{x}\right)\right\|}\le\kappa\left(\mathbf{A}\right)\frac{\left\|\Delta\mathbf{A}\right\|}{\left\|\mathbf{A}\right\|}$$

Remarquons que,

$$\mathbf{A}^{-1}\mathbf{A}=\mathbf{I}\ \Rightarrow\ \left\|\mathbf{A}^{-1}\right\|\cdot\left\|\mathbf{A}\right\|\ge 1\ \Leftrightarrow\ \kappa\left(\mathbf{A}\right)\ge 1$$

Le conditionnement donne une indication de la précision des résultats de l'inversion de matrice et par conséquent de la solution du système d'équations linéaires. Un système d'équations linéaires est bien conditionné si $\kappa\left(\mathbf{A}\right)$ est proche de 1. Si par contre $\kappa\left(\mathbf{A}\right)$ est très grand le système est mal conditionné et dans ce cas il faut prendre de grandes précautions sur l'erreur arrondie et le nombre de chiffres après la virgule dans les coefficients du système.

Malheureusement, le calcul de $\left\|\mathbf{A}^{-1}\right\|$ est coûteux à cause du calcul de la matrice inverse $\mathbf{A}^{-1}$ dont nombre d'opérations de l'ordre de n^3, et vue les définitions de la norme, il est possible d'avoir un calcul de $\kappa\left(\mathbf{A}\right)$ avec l'analyse spectrale en algèbre ou la norme d'une matrice **A** est exprimée par :

$$\left\|\mathbf{A}\right\|_2\approx\max_i\left|\lambda_i\right|\ \text{ et }\ \left\|\mathbf{A}^{-1}\right\|_2\approx\max_i\left|\frac{1}{\lambda_i}\right|=\frac{1}{\min_i\left|\lambda_i\right|}$$

soit,

$$\kappa\left(\mathbf{A}\right)=\kappa_2\left(\mathbf{A}\right)\approx\frac{\max_i\left|\lambda_i\right|}{\min_i\left|\lambda_i\right|}$$

λ_i, $i=1,\ 2,\ \ldots,\ n$ sont les valeurs propres de la matrice **A**.

Avec MATLAB, cette quantité est calculée avec la commande **`cond(A,p)`**, ou **`p`** est un paramètre qui peut être **`1`**, **`2`**, **`'fro'`**, ou **`inf`** caractérisant la norme choisie.

soit l'exemple :

```
>> A=[3 2 -9; -9 5 2; 6 7 3]
A =
    3     2    -9
   -9     5     2
    6     7     3
>> cond(A,1)
ans =
   3.0784
>> cond(A,2)
ans =
```

```
   1.4713
>> cond(A,'fro')
ans =
   3.1521
>> cond(A,inf)
ans =
   3.0850
```

2.6. MATLAB et la résolution du système Ax = b

La résolution du système d'équations linéaires $\mathbf{Ax} = \mathbf{b}$ avec MATLAB, fait avec la commande **** avec l'instruction **`x = A\b`** qui fournit le vecteur solution **x**. Cependant il est fort utile de savoir que Matlab utilise l'algorithme approprié suivant la structure de la matrice **A**.

- Si **A** est creuse et a une structure bande, alors des algorithmes spécifiques à ces structures sont utilisés comme l'algorithme de Thomas.

- Si **A** est une matrice triangulaire supérieure ou inférieure (ou bien une permutation d'une matrice triangulaire), alors le système est résolu par un algorithme de remontée (matrices triangulaires supérieures), ou par un algorithme de descente (matrices triangulaires inférieures). Le test de "triangularité" est effectué pour les matrices pleines en vérifiant les éléments nuls et pour les matrices creuses en inspectant la structure de la matrice.

- Si **A** est symétrique et a des éléments diagonaux réels positifs (ce qui n'implique pas que **A** est définie positive), une factorisation de Cholesky est tentée (**`chol`**). Si **A** est creuse, un algorithme de réordonnement est d'abord appliqué.

- Si aucun des critères précédents n'est vérifié, alors une factorisation en matrices triangulaires est calculée par élimination de Gauss avec pivot partiel (**`lu`**).

3. Méthodes itératives

Les méthodes directes sont très efficaces : elles donnent la solution exacte (aux erreurs d'arrondi près) du système d'équations linéaires. Elles ont l'inconvénient de nécessiter une assez grande place mémoire car elles nécessitent le stockage de toute la matrice en mémoire vive. Si la matrice est pleine, c'est-à-dire, si la plupart des coefficients de la matrice sont non nuls et qu'elle est trop grosse pour la mémoire vive de l'ordinateur, il ne reste plus qu'à gérer l'échange de données entre mémoire disque et mémoire vive pour pouvoir résoudre le système. Cependant, si la matrice du système est "creuse", c'est-à-dire, qu'un grand nombre des coefficients de la matrice du système sont nuls ; de plus la matrice a souvent une structure "bande", c'est-à-dire les éléments non nuls de la matrice sont localisés sur certaines diagonales.

La méthode de décomposition LU "conserve le profil a besoin de la place mémoire pour stocker la structure bande. Lorsqu'on a affaire à de très gros systèmes, on cherche à utiliser des méthodes nécessitant le moins de mémoire possible. On a intérêt dans ce cas à utiliser des méthodes itératives. Ces méthodes ne font appel qu'à des produits matrice-vecteur, et ne nécessitent donc pas le stockage du profil de la matrice mais uniquement des termes non nuls. Par exemple, si on a seulement 5 diagonales non nulles dans la matrice du système à résoudre système de *n* équations et *n* inconnues, la place mémoire nécessaire pour un produit matrice

vecteur est 6*n*. Ainsi pour les gros systèmes, il est souvent avantageux d'utiliser des méthodes itératives qui ne donnent pas toujours la solution exacte du système en un nombre fini d'itérations, mais qui donnent une solution approchée à coût moindre qu'une méthode directe car elles ne font appel qu'à des produits matrice vecteur.

L'idée principale des méthodes itératives pour résoudre un système d'équations linéaires $\mathbf{Ax}=\mathbf{b}$ est de transformer la matrice **A** sous la forme de :

$$\mathbf{A}=\mathbf{Q}-\mathbf{P}$$

Où **Q** une matrice non singulière c'est-à-dire $\mathbf{Q}^{-1}$ existe et facilement calculable, alors,

$$\mathbf{Ax}=\mathbf{b} \;\Rightarrow\; \mathbf{Qx}-\mathbf{Px}=\mathbf{b} \;\Rightarrow\; \mathbf{x}=\mathbf{Q}^{-1}\mathbf{Px}+\mathbf{Q}^{-1}\mathbf{b}$$

Soit le schéma itératif :

$$\mathbf{x}^{(k+1)}=\mathbf{Q}^{-1}\mathbf{P}\mathbf{x}^{(k)}+\mathbf{Q}^{-1}\mathbf{b}, \quad k=0,\ 2,\ 3,\ \ldots,$$

$\mathbf{Q}^{-1}\mathbf{P}$, la matrice d'itération de la méthode, elle est notée par **M**.

Le processus itératif obtenu peut être exprimé d'une autre façon, soit,

$$\mathbf{r}^{(k)}=\mathbf{b}-\mathbf{A}\mathbf{x}^{(k)}\equiv\mathbf{b}-\left(\mathbf{Q}-\mathbf{P}\right)\mathbf{x}^{(k)}$$

Où, $\mathbf{r}^{(k)}$ le résidu à l'itération *k*. Alors,

$$\mathbf{x}^{(k+1)}=\mathbf{Q}^{-1}\mathbf{P}\mathbf{x}^{(k)}+\mathbf{Q}^{-1}\left(\mathbf{r}^{(k)}+\left(\mathbf{Q}-\mathbf{P}\right)\mathbf{x}^{(k)}\right)$$

$$\mathbf{x}^{(k+1)}=\mathbf{Q}^{-1}\mathbf{P}\mathbf{x}^{(k)}+\mathbf{Q}^{-1}\mathbf{r}^{(k)}+\mathbf{Q}^{-1}\mathbf{Q}\mathbf{x}^{(k)}-\mathbf{Q}^{-1}\mathbf{P}\mathbf{x}^{(k)}$$

$$\mathbf{x}^{(k+1)}=\mathbf{x}^{(k)}+\mathbf{Q}^{-1}\mathbf{r}^{(k)}$$

Pour l'étude de la convergence du processus itérative, soit $\mathbf{x}_a$ le vecteur solution de l'équation, c'est-à-dire :

$$\mathbf{x}_a=\mathbf{M}\mathbf{x}_a+\mathbf{Q}^{-1}\mathbf{b}$$

alors,

$$\mathbf{x}^{(k+1)}-\mathbf{x}_a=\mathbf{M}\left(\mathbf{x}^{(k)}-\mathbf{x}_a\right)$$

Soit, $\mathbf{e}^{(k)}=\mathbf{x}^{(k)}-\mathbf{x}_a$ l'erreur à l'itération *k*, alors,

$$\mathbf{e}^{(k+1)}=\mathbf{M}\mathbf{e}^{(k)}$$

en termes de norme,

$$\left\|\mathbf{e}^{(k+1)}\right\|=\left\|\mathbf{M}\mathbf{e}^{(k)}\right\|\le\left\|\mathbf{M}\right\|\ \left\|\mathbf{e}^{(k)}\right\|$$

A la limite maximale, la norme $\left\|\mathbf{e}^{(k)}\right\|$ constitue une progression géométrique de raison $\left\|\mathbf{M}\right\|$ qui converge vers 0 si et seulement si $\left\|\mathbf{M}\right\|<1$.

Soit $\rho(\mathbf{M})$ le *rayon spectral* de la matrice **M**, par définition il est exprimé par le plus grand module des valeurs propres de la matrice **M** c'est-à-dire :

$$\rho(\mathbf{M})=\max_i\left|\lambda_i\right|$$

On peut montrer (Quarteroni *et al*., 2010) que $\rho(\mathbf{M})\le\left\|\mathbf{M}\right\|$ et déduire aussi que, le processus itératif converge aussi si et seulement si $\rho(\mathbf{M})<1$.

La condition de convergence $\|\mathbf{M}\| < 1$ ou bien $\rho(\mathbf{M}) < 1$ est une condition suffisante et également nécessaire. Dans la pratique le rayon spectral est difficile à calculer à cause du calcul des valeurs propres. C'est pourquoi on utilise d'autres conditions suffisantes de convergence (Quarteroni *et al.*, 2010).

En théorie, il faudrait effectuer un nombre infini d'itérations pour obtenir la solution exacte d'un système linéaire avec une méthode itérative. En pratique, ce n'est ni nécessaire, ni raisonnable même si effectivement le nombre d'itérations pour obtenir la solution avec la précision machine peut être très élevé pour de grands systèmes. En effet, ce n'est en général pas d'une solution exacte dont on a besoin, mais plutôt d'une valeur $\mathbf{x}^{(k)}$ qui approche la solution exacte avec une erreur inférieure à une tolérance ε fixée. Mais comme l'erreur est elle-même inconnue (puisqu'elle dépend de la solution exacte), on a besoin d'un estimateur d'erreur *a posteriori* qui donne une estimation de l'erreur à partir de quantités calculées au cours de la résolution (Quarteroni *et al.*, 2010).

Pour évaluer le nombre d'itérations minimal $k_{\min}$ nécessaires pour arriver à la convergence du processus suivant la tolérance ε Soit, l'inégalité trouvée précédemment,

$$\mathbf{e}^{(k+1)} = \mathbf{M}\mathbf{e}^{(k)}$$

alors,

$$\mathbf{e}^{(k)} = \mathbf{M}\mathbf{e}^{(k-1)} = \mathbf{M}^2\,\mathbf{e}^{(k-2)} = \mathbf{M}^3\,\mathbf{e}^{(k-3)} = \cdots = \mathbf{M}^k\,\mathbf{e}^{(0)}$$

par conséquent,

$$\left\|\mathbf{e}^{(k)}\right\| = \left\|\mathbf{M}^k\,\mathbf{e}^{(0)}\right\| \;\Rightarrow\; \left\|\mathbf{e}^{(k)}\right\| \le \left\|\mathbf{M}^k\right\| \cdot \left\|\mathbf{e}^{(0)}\right\|$$

$\left\|\mathbf{M}^k\right\|$ est le facteur de convergence à l'itération k. Typiquement, les itérations se poursuivent jusqu'à ce que :

$$\left\|\mathbf{e}^{(k)}\right\| \le \varepsilon \left\|\mathbf{e}^{(0)}\right\| \text{ avec } \varepsilon < 1$$

Le calcul de la quantité $\left\|\mathbf{M}^k\right\|$ est trop coûteux car il requiert l'évaluation de $\mathbf{M}^k$. Dans ce cas il est préférable d'utiliser en général d'estimer le taux de convergence moyen à l'itération k :

$$R_k(\mathbf{M}) = -\frac{1}{k}\log\left\|\mathbf{M}^k\right\|$$

$$R_k(\mathbf{M}) \;\to\; R(\mathbf{M}) = -\log\rho(\mathbf{M}) \text{ quand } k \;\to\; \infty$$

$R(\mathbf{M})$, est le taux de convergence asymptotique (Quarteroni *et al.*, 2010). L'itération minimale estimée de convergence est peut-être obtenue à partir de :

$$\left\|\mathbf{M}^k\right\| \equiv \varepsilon \text{ et } R_{k_{\min}}(\mathbf{M}) = -\frac{1}{k_{\min}}\log\varepsilon \approx R(\mathbf{M})$$

alors,

$$-\frac{1}{k_{\min}}\log\varepsilon \approx -\log\rho(\mathbf{M}) \;\Rightarrow\; k_{\min} \approx \frac{\log\varepsilon}{\log\rho(\mathbf{M})}$$

Dans cette formule, $k_{\min}$ nécessite le calcul des valeurs propre de la matrice **M**. Ce calcul est très coûteux surtout si la dimension de la matrice **M** à une dimension très grande. Une autre méthode pour déterminer l'itération de convergence, de :

$$\left\|\mathbf{e}^{(k+1)}\right\| \le \|\mathbf{M}\| \cdot \left\|\mathbf{e}^{(k)}\right\|$$

or,

$$\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}_a = \mathbf{x}^{(k)} + \mathbf{x}^{(k+1)} - \mathbf{x}^{(k+1)} - \mathbf{x}_a = \left(\mathbf{x}^{(k)} - \mathbf{x}^{(k+1)}\right) + \left(\mathbf{x}^{(k+1)} - \mathbf{x}_a\right) \equiv \left(\mathbf{x}^{(k)} - \mathbf{x}^{(k+1)}\right) + \mathbf{e}^{(k+1)}$$

soit,

$$\left\|\mathbf{e}^{(k+1)}\right\| \le \|\mathbf{M}\| \cdot \left\|\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k+1)}\right) + \mathbf{e}^{(k+1)}\right\| \le \|\mathbf{M}\| \cdot \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| + \|\mathbf{M}\| \cdot \left\|\mathbf{e}^{(k+1)}\right\|$$

Sachant que,

$$\mathbf{x}^{(k+1)} = \mathbf{M}\mathbf{x}^{(k)} + \mathbf{Q}^{-1}\mathbf{b} \quad \text{et} \quad \mathbf{x}^{(k)} = \mathbf{M}\mathbf{x}^{(k-1)} + \mathbf{Q}^{-1}\mathbf{b}$$

alors,

$$\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} = \mathbf{M}\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right) = \mathbf{M}^2\left(\mathbf{x}^{(k-1)} - \mathbf{x}^{(k-2)}\right) = \cdots = \mathbf{M}^k\left(\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\right)$$

donc,

$$\left\|\mathbf{e}^{(k+1)}\right\| \le \|\mathbf{M}\|^{k+1}\left\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\right\| + \|\mathbf{M}\| \cdot \left\|\mathbf{e}^{(k+1)}\right\|$$

alors,

$$\left\|\mathbf{e}^{(k+1)}\right\| \le \frac{\|\mathbf{M}\|^{k+1}}{1 - \|\mathbf{M}\|}\left\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\right\| \equiv \varepsilon_{k+1}$$

Ce qui permet d'avoir pour une tolérance ε_{k+1} fixée, l'itération correspondante à la convergence,

$$k = \frac{\ln\left(\dfrac{1 - \|\mathbf{M}\|}{\left\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\right\|}\varepsilon_{k+1}\right)}{\ln\left(\|\mathbf{M}\|\right)} - 1$$

3.1. Méthode itérative de Jacobi

La matrice **A** s'écrit sous la forme,

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$$

avec,

$$\mathbf{D} = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix}, \mathbf{L} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn-1} & 0 \end{pmatrix} \text{ et } \mathbf{U} = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & a_{n-1n} \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

alors,

$$\mathbf{A}\mathbf{x} = \mathbf{b} \;\Leftrightarrow\; \mathbf{D}\mathbf{x} + \mathbf{L}\mathbf{x} + \mathbf{U}\mathbf{x} = \mathbf{b} \;\Rightarrow\; \mathbf{D}\mathbf{x} = -\left(\mathbf{L} + \mathbf{U}\right)\mathbf{x} + \mathbf{b}$$

Soit alors le processus itératif :

$$\mathbf{D}\mathbf{x}^{(k+1)} = -\left(\mathbf{L} + \mathbf{U}\right)\mathbf{x}^{(k)} + \mathbf{b} \;\Rightarrow\; \mathbf{x}^{(k+1)} = -\mathbf{D}^{-1}\left(\mathbf{L} + \mathbf{U}\right)\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}$$

La matrice inverse de la matrice diagonale **D** est obtenue facilement est vaut :

$$\mathbf{D}^{-1}=\begin{pmatrix} 1/a_{11} & 0 & \cdots & 0 \\ 0 & 1/a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1/a_{nn} \end{pmatrix}$$

Qui peut être exprimé en termes de composantes par :

$$x_i^{(k+1)}=\frac{1}{a_{ii}}\left(b_i-\sum_{\substack{j=1\\ j\neq i}}^{n}a_{ij}x_j^{(k)}\right) \quad \text{pour } i=1,\ 2,\ 3,\ \ldots,\ n$$

La matrice d'itération de Jacobi est exprimée par :

$$\mathbf{M}_{\mathrm{J}}=\mathbf{D}^{-1}\left(\mathbf{L}+\mathbf{U}\right)=\begin{pmatrix} 0 & a_{12}/a_{11} & \cdots & a_{1n}/a_{11} \\ a_{21}/a_{22} & 0 & \cdots & a_{2n}/a_{22} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1}/a_{nn} & a_{n2}/a_{nn} & \cdots & 0 \end{pmatrix}$$

D'après la condition suffisante de convergence :

$$\left\|\mathbf{M}_{\mathrm{J}}\right\|<1 \ \Rightarrow\ \sum_{\substack{j=1\\ j\neq i}}^{n}\frac{|a_{ij}|}{|a_{ii}|}<1 \quad \text{pour } i=1,\ 2,\ 3,\ \ldots,\ n$$

C'est-à-dire la matrice $\mathbf{M}_{\mathrm{J}}$ est strictement dominante car la condition

$$|a_{ii}|>\sum_{\substack{j=1\\ j\neq i}}^{n}|a_{ij}|,\ \forall i=1,\ 2,\ 3,\ \ldots,\ n$$

Soit la fonction **Jacobi** qui sert à résoudre le système d'équations linéaires par le processus itérative de Jacobi.

```
function [x,i] = Jacobi(A,b,x0)
% Résolution du système d'équations Ax=b
% x0 est le vecteur initial
D=diag(diag(A)); % Formulation de la matrice diagonale D
L=tril(A-D); % Formulation de la matrice triangulaire inférieure L
U=triu(A-D); % Formulation de la matrice triangulaire supérieur U
MJ=inv(D)*(L+U); % Matrice d'itération de Jacobi
  if norm(MJ,inf)>=1
     disp('Problème de Convergence')
  end
 i=0;
x=-MJ*x0+inv(D)*b;
 while norm(x - x0) > 1e-6
   x0=x;
     i=i+1;
   x=-MJ*x0+inv(D)*b;
 end
end
```

A titre d'exemple soit le système d'équations linéaires :

$$\begin{pmatrix} 4 & 1 & -1 \\ -2 & 5 & 0 \\ 2 & 1 & 6 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}=\begin{pmatrix} 1 \\ -7 \\ 13 \end{pmatrix} \quad \text{et} \quad \mathbf{x}^{(0)}=\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \text{ comme vecteur initial}$$

L'introduction des données et l'application de la fonction **Jacobi**,

```
>> A = [4 1 -1;-2 5 0;2 1 6]; b = [1;-7;13]; x0 = [0;1;1];
>> [x,i]=Jacobi(A,b,x0)
x =
    1.0000
   -1.0000
    2.0000
i =
    18
```

Ce qui montre que le processus itératif de Jacobi converge vers la solution $x_1 = 1,\ x_2 = -1$ et $x_3 = 2$ après 18 itérations.

3.2. Méthode itérative de Gauss–Seidel

Dans le cas de cette méthode le système d'équations s'écrit :

$$\mathbf{A}\mathbf{x} = \mathbf{b} \ \Leftrightarrow\ \mathbf{D}\mathbf{x} + \mathbf{L}\mathbf{x} + \mathbf{U}\mathbf{x} = \mathbf{b} \ \Rightarrow\ (\mathbf{D}+\mathbf{L})\mathbf{x} = -\mathbf{U}\mathbf{x} + \mathbf{b}$$

Soit alors le processus itératif :

$$(\mathbf{D}+\mathbf{L})\mathbf{x}^{(k+1)} = -\mathbf{U}\mathbf{x}^{(k)} + \mathbf{b} \ \Rightarrow\ \mathbf{x}^{(k+1)} = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}\mathbf{x}^{(k)} + (\mathbf{D}+\mathbf{L})^{-1}\mathbf{b}$$

Qui peut être exprimé en termes de composantes par :

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} \right) \text{ pour } i = 1,\ 2,\ 3,\ \ldots,\ n$$

La matrice d'itération de Gauss–Seidel est :

$$\mathbf{M}_S = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}$$

La fonction **Seidel** qui sert à résoudre le système d'équations linéaires par le processus itérative de Seidel.

```
function [x,i]=Seidel(A,b,x0)
% Résolution du système d'équations Ax=b
% x0 est le vecteur initial
D=diag(diag(A)); % Formulation de la matrice diagonale D
L=tril(A-D); % Formulation de la matrice triangulaire inférieure L
U=triu(A-D); % Formulation de la matrice triangulaire supérieur U
MS=inv(D+L)*U; % Matrice d'itération de Gauss-Seidel
if norm(MS,inf)>=1
     disp('Problème de Convergence')
  end
 i=0;
x=-MS*x0+inv(D+L)*b;
 while norm(x - X0) > 1e-6
   x0=x;
     i=i+1;
   x=-MS*x0+inv(D+L)*b;
 end
end
```

L'introduction des données et l'application de la fonction **Seidel** de l'exemple précédent.

```
>> A = [4 1 -1;-2 5 0;2 1 6]; b = [1;-7;13]; x0 = [0;1;1];
>> [x,i]=Seidel(A,b,x0)
```

```
x =
    1.0000
   -1.0000
    2.0000
i =
    10
```

Ce qui montre que le processus itératif de Gauss–Seidel converge vers la solution $x_1 = 1,\ x_2 = -1$ et $x_3 = 2$ après 10 itérations.

3.3. Méthode itérative de surrelaxation successive

Une généralisation des méthodes itératives Jacobi et Gauss–Seidel est la méthode de surrelaxation successive (JOR pour Jacobi et SOR pour Seidel). L'idée de la méthode itérative de surrelaxation successive est d'utiliser la méthode itérative pour calculer un itéré intermédiaire $\tilde{\mathbf{x}}^{(k+1)}$ qu'on "relaxe" ensuite pour améliorer la vitesse de convergence suivant l'équation exprimée en termes de composantes.

$$x_i^{(k+1)} = \omega \tilde{x}_i^{(k+1)} + (1-\omega)\, x_i^{(k)} \quad \text{pour } i = 1,\ 2,\ 3,\ \ldots,\ n$$

Où, $\omega > 0$ un facteur constant dit de relaxation.
Pour la méthode de Jacobi,

$$\tilde{x}_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right)$$

Pour la méthode de Gauss–Seidel,

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

La matrice d'itération dans la méthode JOR est :

$$\mathbf{M}_{\text{JOR}} = \omega \mathbf{D}^{-1} (\mathbf{L} + \mathbf{U}) + (1-\omega)\mathbf{I}_n$$

Pour le cas de la méthode SOR, soit,

$$\omega \mathbf{A}\mathbf{x} = \omega \mathbf{b} \quad \Leftrightarrow \quad \omega \mathbf{D}\mathbf{x} + \omega \mathbf{L}\mathbf{x} + \omega \mathbf{U}\mathbf{x} = \omega \mathbf{b}$$

soit,

$$\mathbf{D}\mathbf{x} + \omega \mathbf{D}\mathbf{x} + \omega \mathbf{L}\mathbf{x} + \omega \mathbf{U}\mathbf{x} = \mathbf{D}\mathbf{x} + \omega \mathbf{b}$$

alors,

$$\mathbf{D}\mathbf{x} + \omega \mathbf{L}\mathbf{x} = \mathbf{D}\mathbf{x} - \omega \mathbf{D}\mathbf{x} - \omega \mathbf{U}\mathbf{x} + \omega \mathbf{b}$$

ou,

$$(\mathbf{D} + \omega \mathbf{L})\mathbf{x} = -\big((\omega - 1)\mathbf{D} + \omega \mathbf{U}\big)\mathbf{x} + \omega \mathbf{b}$$

Soit alors le processus itératif :

$$(\mathbf{D} + \omega \mathbf{L})\mathbf{x}^{(k+1)} = -\big((\omega - 1)\mathbf{D} + \omega \mathbf{U}\big)\mathbf{x}^{(k)} + \omega \mathbf{b}$$

ou bien,

$$\mathbf{x}^{(k+1)} = -(\mathbf{D} + \omega \mathbf{L})^{-1} \big((\omega - 1)\mathbf{D} + \omega \mathbf{U}\big)\mathbf{x}^{(k)} + \omega (\mathbf{D} + \omega \mathbf{L})^{-1} \mathbf{b}$$

Alors, la matrice d'itération dans ce cas est :

$$\mathbf{M}_{SOR} = -\left(\mathbf{D}+\omega\mathbf{L}\right)^{-1}\left((\omega-1)\mathbf{D}+\omega\mathbf{U}\right)$$

Le choix du facteur de relaxation ω n'est pas trivial et dépend des coefficients de la matrice. Pour une matrice définie positive, on peut démontrer (Quarteroni *et al.*, 2007) que l'algorithme est convergent pour tout $0<\omega<2$. Toutefois, on veut une convergence aussi rapide que possible. Il est clair que pour un facteur de relaxation $\omega=1$ centrale de l'intervalle]0, 2[, on tombe sur la méthode de Gauss-Seidel.

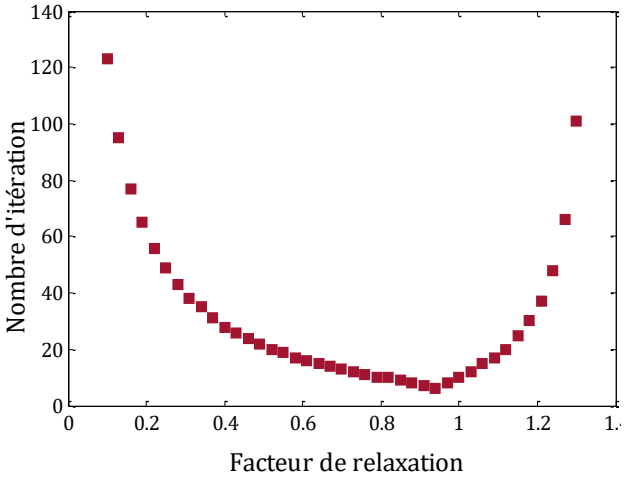


Fig. 2.3 : Variation du nombre d'itération en fonction de ω.

Soit la fonction **SOR** sert à résoudre le système d'équations linéaires par le processus itérative de surrelaxation successives.

```
function [X,iter] = SOR(A,b,X0,Omega)
% Résolution du système d'équations Ax=b
% X0 est le vecteur initial
D=diag(diag(A)); L=tril(A-D); U=triu(A-D);
M=(inv(D+Omega*L))*((Omega-1.)*D+Omega*U);%Matrice d'itération de Seidel
 if norm(M,inf)>=1
    disp('Problème de Convergence')
 end
  iter =0;
  X=-M*X0+(Omega)*inv(D+(Omega*L))*b;
 while norm(X - X0) > 1e-6
  X0=X; iter = iter +1;
  X=-M*X0+(Omega)*inv(D+(Omega*L))*b;
 end
end
```

Pour l'exemple précédent et pour $\omega=0.9$, l'application de la fonction **SOR** conduit donc à :

```
>> [x, iteration]=SOR(A,b,x0,0.9)
x =
    1.0000
   -1.0000
    2.0000
iteration =
    7
```

C'est-à-dire, pour $\omega=0.9$, le processus converge dès la 7ième itération. Dans cette exemple le choix du facteur de relaxation ω de l'intervalle [0.1, 1.33] conduit à un rayon spectral de la matrice d'itération de Seidel inférieur à 1 $\left(\rho\left(\mathbf{M}_{SOR}\right)<1\right)$, par conséquent le processus itératif converge.

Le choix de ω joue un rôle sur la vitesse de convergence du processus itératif. Dans l'exemple traité (Fig. 2.3), le nombre d'itérations minimal pour atteinte la convergence,

correspond à des valeurs de ω proche de 1 (processus itératif de Gauss–Seidel). Hackbusch (2016) et Young (1971) ont démontré que le rayon spectral de la matrice d'itération que ce soit $\rho\left(\mathbf{M}_{\text{JOR}}\right)$ ou $\rho\left(\mathbf{M}_{\text{SOR}}\right)$ est minimal pour de $\omega \cong 1$ (Figs. 2.4 et 2.5).

Dans la pratique, il est recommandé de choisir ω proche de 1 pour atteinte rapidement la solution cherchée avec moins de calculs.

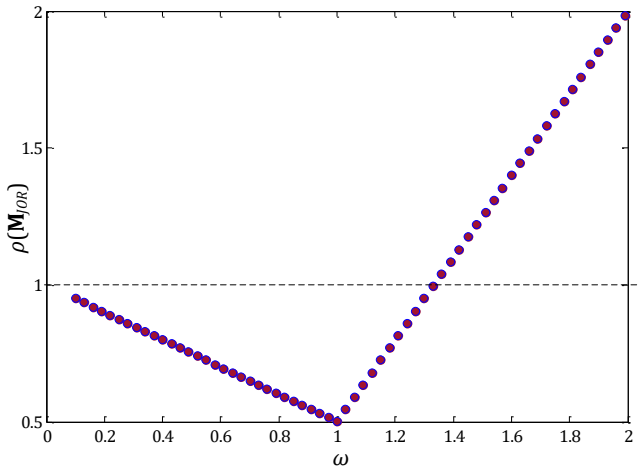


Fig. 2.4 : Variation de $\rho(\mathbf{M}_{\text{JOR}})$ en fonction de ω.

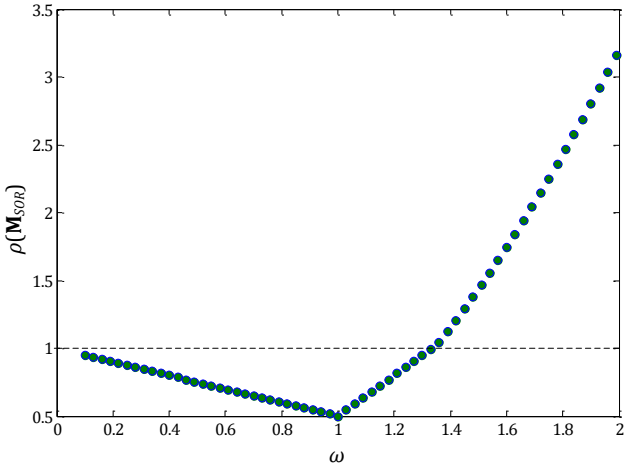


Fig. 2.5 : Variation de $\rho(\mathbf{M}_{\text{SOR}})$ en fonction de ω.

3.4. Amélioration de la convergence des méthodes de surrelaxation successive

Pour les différentes méthodes itératives vues précédemment la matrice $\mathbf{Q}$ est exprimée par $\omega\mathbf{D}$ pour la méthode JOR et par $\omega\left(\mathbf{D}+\omega\mathbf{L}\right)$ pour la méthode SOR. La matrice $\mathbf{Q}$ de Jacobi et de Gauss–Seidel est obtenue pour $\omega = 1$.

Pour améliorer la convergence, soit la généralisation de la méthode itérative JOR ou SOR par la *méthode de Richardson stationnaire* :

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha\mathbf{Q}^{-1}\mathbf{r}^{(k)}$$

Ou, α est constant. La matrice d'itération de Richardson est :

$$\mathbf{R}_{\alpha} = \mathbf{I}_n - \alpha\mathbf{Q}^{-1}\mathbf{A}$$

Si $\mathbf{Q} = \mathbf{I}_n$, la méthode est dite *non pré-conditionnée*. Les itérations de Jacobi et Gauss–Seidel peuvent être vues comme des méthodes de Richardson stationnaires avec $\alpha = 1$.

La méthode de Richardson stationnaire converge tout simplement si et seulement si :

$$\alpha < \frac{2\text{Re}\left(\lambda_i\right)}{\left|\lambda_i\right|^2} \quad \forall i = 1,\ 2,\ 3,\ \ldots,\ n$$

$\text{Re}\left(\lambda_i\right)$ est la partie réelle de la valeur propre λ_i de la matrice $\mathbf{Q}^{-1}\mathbf{A}$.

Si les matrices $\mathbf{Q}$ et $\mathbf{A}$ sont symétriques et définies positives, la méthode de Richardson stationnaire converge pour tout $\mathbf{x}^{(0)}$ si et seulement si, $0 < \alpha < 2/\lambda_{\max}$, où $\lambda_{\max} > 0$ est la valeur propre maximale de $\mathbf{Q}^{-1}\mathbf{A}$. De plus, le rayon spectral de la matrice d'itération $\mathbf{R}_{\alpha} = \mathbf{I}_n - \alpha\mathbf{Q}^{-1}\mathbf{A}$ est minimum si $\alpha = \alpha_{opt}$ est exprimé par (Quarteroni *et al.*, 2007) :

$$\alpha_{opt} = \frac{2}{\lambda_{\min} + \lambda_{\max}}$$

$\lambda_{\min}$ étant la valeur propre minimal de $\mathbf{Q}^{-1}\mathbf{A}$. Le rayon spectral optimal de la matrice d'itération $\mathbf{R}_\alpha$ est :

$$\rho_{opt} = \frac{\lambda_{\min} - \lambda_{\max}}{\lambda_{\min} + \lambda_{\max}}$$

En résumé

- Résoudre un système linéaire avec une méthode itérative consiste à construire, en partant d'une donnée initiale $\mathbf{x}^{(0)}$, une suite de vecteurs $\mathbf{x}^{(k)}$ convergeant vers la solution exacte quand $k \to \infty$.
- Une méthode itérative converge pour toute donnée initiale $\mathbf{x}^{(0)}$ si et seulement si le rayon spectral de la matrice d'itération est strictement plus petit que 1 ($\rho(\mathbf{M}) < 1$).
- Une condition suffisante de convergence des méthodes itératives de Jacobi et de Gauss-Seidel, est que la matrice d'itération soit à diagonale strictement dominante par ligne (ou symétrique définie positive dans le cas de Gauss-Seidel).
- Dans la méthode de Richardson, la convergence est accélérée à l'aide d'un paramètre et (éventuellement) d'un pré-conditionneur bien choisi.
- Deux critères d'arrêt possible pour les méthodes itératives : l'un basé sur le résidu, l'autre sur l'incrément. Le premier est pertinent quand le système est bien conditionné, le second quand le rayon spectral de la matrice d'itération n'est pas trop proche de 1.

4. Méthodes directes ou itératives ?

Deux types de méthodes peuvent être utilisées pour résoudre les systèmes d'équations linéaires. Pour les systèmes linéaires de petite taille, le choix n'a pas beaucoup d'importance car toutes les méthodes feront l'affaire. En revanche, pour les grands systèmes linéaires, le choix dépendra principalement des propriétés de la matrice (telles que la symétrie, la définie positivité, la structure creuse, le conditionnement), mais également des ressources informatiques disponibles (accès mémoire, processeurs rapides, etc.).

Les méthodes directes (surtout quand elles sont implémentées de manière sophistiquée, comme pour la commande **** de MATLAB) sont plus efficaces que les méthodes itératives quand ces dernières ne sont pas utilisées avec des pré-conditionneurs performants. Cependant, elles sont plus sensibles au conditionnement de la matrice et peuvent nécessiter une mémoire importante. Il est également utile de souligner que les méthodes directes ont explicitement besoin des coefficients de la matrice, contrairement aux méthodes itératives. Pour ces dernières, il est seulement nécessaire de pouvoir calculer le produit matrice-vecteur pour des vecteurs arbitraires. Cette propriété est particulièrement intéressante dans les problèmes où la matrice n'est pas construite explicitement.

5. Exercices résolus

Exercice 1

Pour décomposer l'expression $\left(4x^3+4x^2+x-1\right)\Big/x^2\left(x+1\right)^2$ en somme de fractions partielles comme :

$$\frac{4x^3+4x^2+x-1}{x^2\left(x+1\right)^2}=\frac{A_1}{x}+\frac{A_2}{x^2}+\frac{A_3}{x+1}+\frac{A_4}{\left(x+1\right)^2}$$

Déterminer le système d'équations dont les coefficients A_i, pour $i=1,\ 2,\ 3$ et 4 sont les inconnus et utiliser la fonction Matlab **`Pivot2Gauss`** pour résoudre ce système.

De,

$$\frac{A_1}{x}+\frac{A_2}{x^2}+\frac{A_3}{x+1}+\frac{A_4}{\left(x+1\right)^2}=\frac{A_1x\left(x+1\right)^2+A_2\left(x+1\right)^2+A_3x^2\left(x+1\right)+A_4x^2}{x^2\left(x+1\right)^2}$$

$$=\frac{A_1\left(x^3+2x^2+x\right)+A_2\left(x^2+2x+1\right)+A_3\left(x^3+x^2\right)+A_4x^2}{x^2\left(x+1\right)^2}$$

$$=\frac{\left(A_1+A_3\right)x^3+\left(2A_1+A_2+A_4\right)x^2+\left(A_1+2A_2\right)x+A_2}{x^2\left(x+1\right)^2}$$

soit,

$$\begin{cases}A_1+A_3=4,\\ 2A_1+A_2+A_4=4,\\ A_1+2A_2=1,\\ A_2=-1.\end{cases}\Leftrightarrow\begin{pmatrix}1&0&1&0\\2&1&0&1\\1&2&0&0\\0&1&0&0\end{pmatrix}\begin{pmatrix}A_1\\A_2\\A_3\\A_4\end{pmatrix}=\begin{pmatrix}4\\4\\1\\-1\end{pmatrix}$$

La solution du système avec la fonction **`Pivot2Gauss`** donne :

$$A_1=3,\ \ A_2=-1,\ \ A_3=1\ \text{ et }\ A_4=-1$$

alors,

$$\frac{4x^3+4x^2+x-1}{x^2\left(x+1\right)^2}=\frac{3}{x}-\frac{1}{x^2}+\frac{1}{x+1}-\frac{1}{\left(x+1\right)^2}$$

Exercice 2

Soit le système d'équations tridiagonale de 8 inconnues :

$$\begin{pmatrix}-2&1&0&\cdots&0\\1&-2&1&\ddots&\vdots\\0&\ddots&\ddots&\ddots&0\\\vdots&\ddots&1&-2&1\\0&\cdots&0&1&-2\end{pmatrix}\begin{pmatrix}x_1\\x_2\\\vdots\\x_7\\x_8\end{pmatrix}=\mathbf{b}$$

Utiliser la fonction MATLAB **`DecompositionLU`** pour résoudre le système pour :

(a) $\mathbf{b}={}^t\begin{bmatrix}1&1&\cdots&1\end{bmatrix}$

(b) $\mathbf{b}={}^t\begin{bmatrix}10&0&10&\cdots&0&10\end{bmatrix}$

Soit l'introduction de la matrice **A** dans Matlab,

```
>> A=[-2 1 0 0 0 0 0 0
1 -2 1 0 0 0 0 0
0 1 -2 1 0 0 0 0
0 0 1 -2 1 0 0 0
0 0 0 1 -2 1 0 0
0 0 0 0 1 -2 1 0
0 0 0 0 0 1 -2 1
0 0 0 0 0 0 1 -2]

A =

    -2     1     0     0     0     0     0     0
     1    -2     1     0     0     0     0     0
     0     1    -2     1     0     0     0     0
     0     0     1    -2     1     0     0     0
     0     0     0     1    -2     1     0     0
     0     0     0     0     1    -2     1     0
     0     0     0     0     0     1    -2     1
     0     0     0     0     0     0     1    -2
```

(a) Cas où $\mathbf{b} = {}^t\begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}$, alors,

```
>> b=[1 1 1 1 1 1 1 1]

b =

     1     1     1     1     1     1     1     1
```

Les deux matrices **L** et **U** sont :

```
>> [~, L, U, ~]=DecompositionLU(A,b)

L =

  Columns 1 through 5

   -2.0000         0         0         0         0
    1.0000   -1.5000         0         0         0
         0    1.0000   -1.3333         0         0
         0         0    1.0000   -1.2500         0
         0         0         0    1.0000   -1.2000
         0         0         0         0    1.0000
         0         0         0         0         0
         0         0         0         0         0

  Columns 6 through 8

         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
   -1.1667         0         0
    1.0000   -1.1429         0
         0    1.0000   -1.1250
```

```
U =
  Columns 1 through 5
    1.0000   -0.5000         0         0         0
         0    1.0000   -0.6667         0         0
         0         0    1.0000   -0.7500         0
         0         0         0    1.0000   -0.8000
         0         0         0         0    1.0000
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
  Columns 6 through 8
         0         0         0
         0         0         0
         0         0         0
         0         0         0
   -0.8333         0         0
    1.0000   -0.8571         0
         0    1.0000   -0.8750
         0         0    1.0000
```

La solution du système dans ce cas est obtenue à partir de :

```
>> x=DecompositionLU(A,b)
x =
   -4.0000
   -7.0000
   -9.0000
  -10.0000
  -10.0000
   -9.0000
   -7.0000
   -4.0000
```

(b) Cas où $\mathbf{b} = {}^{t}\left[10 \;\; 0 \;\; 10 \;\; \cdots \;\; 10 \;\; 0\right]$,

```
>> b=[10 0 10 0 10 0 10 0]
b =
     10     0     10     0     10     0     10     0
```

La solution du système est obtenue à partir de :

```
>> [~, ~, ~, x]=DecompositionLU(A,b)
x =
  -22.2222
  -34.4444
  -46.6667
  -48.8889
  -51.1111
  -43.3333
  -35.5556
  -17.7778
```

Exercice 3

Résoudre les deux systèmes d'équations linéaires suivantes :

$$\begin{cases} 4x_1 + x_2 + x_3 = 4 \\ x_1 + 4x_2 - 2x_3 = 4 \\ 3x_1 + 2x_2 - 4x_3 = 6 \end{cases} \quad \text{et} \quad \begin{cases} x_1 + x_2 - x_3 = 2 \\ 2x_1 + 3x_2 + 5x_3 = -3 \\ 3x_1 + 2x_2 - 3x_3 = 6 \end{cases}$$

(a) Par la méthode d'élimination de Gauss.

(b) Par la méthode de décomposition **LU** avec $u_{11} = u_{22} = u_{33} = 1$.

Pour le premier système,

(a) Les deux membres du système s'écrit comme :

$$(\mathbf{A} \mid \mathbf{b}) = \left(\begin{array}{ccc|c} 4 & 1 & 1 & 4 \\ 1 & 4 & -2 & 4 \\ 3 & 2 & -4 & 6 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 4 & 1 & 1 & 4 \\ 0 & 15/4 & -9/4 & 3 \\ 0 & 5/4 & -19/4 & 3 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 4 & 1 & 1 & 4 \\ 0 & 15/4 & -9/4 & 3 \\ 0 & 0 & -4 & 2 \end{array}\right)$$

Par substitution arrière,

$$x_3 = -1/2, \quad x_2 = 1/2 \quad \text{et} \quad x_1 = 1$$

(b) $\mathbf{A} = \mathbf{LU}$, avec $u_{11} = u_{22} = u_{33} = 1$ soit,

$$\begin{pmatrix} 4 & 1 & 1 \\ 1 & 4 & -2 \\ 3 & 2 & -4 \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

ce qui conduit à,

$$\mathbf{L} = \begin{pmatrix} 4 & 0 & 0 \\ 1 & 15/4 & 0 \\ 3 & 5/4 & -4 \end{pmatrix}, \quad \mathbf{U} = \begin{pmatrix} 1 & 1/4 & 1/4 \\ 0 & 1 & -3/5 \\ 0 & 0 & 1 \end{pmatrix}$$

On a l'équation,

$$\mathbf{Ly} = \mathbf{b} \Leftrightarrow \begin{pmatrix} 4 & 0 & 0 \\ 1 & 15/4 & 0 \\ 3 & 5/4 & -4 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \\ 6 \end{pmatrix} \Rightarrow \begin{cases} 4y_1 = 4 \\ y_1 + 15y_2/4 = 4 \\ 3y_1 + 5y_2/4 - 4y_3 = 6 \end{cases} \Rightarrow \begin{cases} y_1 = 1 \\ y_2 = 4/5 \\ y_3 = -1/2 \end{cases}$$

et,

$$\mathbf{Ux} = \mathbf{y} \Leftrightarrow \begin{pmatrix} 1 & 1/4 & 1/4 \\ 0 & 1 & -3/5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 4/5 \\ -1/2 \end{pmatrix} \Rightarrow \begin{cases} x_1 + x_2/4 + x_3/4 = 1 \\ x_2 - 3x_3/5 = 4/5 \\ x_3 = -1/2 \end{cases} \Rightarrow \begin{cases} x_3 = -1/2 \\ x_2 = 1/2 \\ x_1 = 1 \end{cases}$$

alors,

$$x_1 = 1, \quad x_2 = 1/2 \quad \text{et} \quad x_3 = -1/2$$

Pour le second système,

(a) Les deux membres du système s'écrit comme :

$$(\mathbf{A} \mid \mathbf{b}) = \left(\begin{array}{ccc|c} 1 & 1 & -1 & 2 \\ 2 & 3 & 5 & -3 \\ 3 & 2 & -3 & 6 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 3 & 2 & -3 & 6 \\ 2 & 3 & 5 & -3 \\ 1 & 1 & -1 & 2 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 3 & 2 & -3 & 6 \\ 0 & 5/3 & 7 & -7 \\ 0 & 1/3 & 0 & 0 \end{array}\right)$$

$$\rightarrow \left(\begin{array}{ccc|c} 3 & 2 & -3 & 6 \\ 0 & 5/3 & 7 & -7 \\ 0 & 0 & -7/5 & 7/5 \end{array}\right)$$

Par substitution arrière,

$$x_3 = -1, \;\; x_2 = 0 \;\; \text{et} \;\; x_1 = 1$$

(b) $\mathbf{A} = \mathbf{LU}$, avec $u_{11} = u_{22} = u_{33} = 1$ soit,

$$\begin{pmatrix} 1 & 1 & -1 \\ 2 & 3 & 5 \\ 3 & 2 & -3 \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

ce qui conduit à,

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & -1 & 7 \end{pmatrix}, \;\; \mathbf{U} = \begin{pmatrix} 1 & 1 & -1 \\ 0 & 1 & 7 \\ 0 & 0 & 1 \end{pmatrix}$$

On a l'équation,

$$\mathbf{Ly} = \mathbf{b} \Leftrightarrow \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & -1 & 7 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 2 \\ -3 \\ 6 \end{pmatrix} \Rightarrow \begin{cases} y_1 = 2 \\ 2y_1 + y_2 = -3 \\ 3y_1 - y_2 + 7y_3 = 6 \end{cases} \Rightarrow \begin{cases} y_1 = 2 \\ y_2 = -7 \\ y_3 = -1 \end{cases}$$

et,

$$\mathbf{Ux} = \mathbf{y} \Leftrightarrow \begin{pmatrix} 1 & 1 & -1 \\ 0 & 1 & 7 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ -7 \\ -1 \end{pmatrix} \Rightarrow \begin{cases} x_1 + x_2 - x_3 = 2 \\ x_2 + 7x_3 = -7 \\ x_3 = -1 \end{cases} \Rightarrow \begin{cases} x_3 = -1 \\ x_2 = 0 \\ x_1 = 1 \end{cases}$$

alors,

$$x_1 = 1, \;\; x_2 = 0 \;\; \text{et} \;\; x_3 = -1$$

Exercice 4

Soit le système d'équations linéaires **Ax** = **b**, avec :

$$\mathbf{A} = \begin{pmatrix} 2 & 3 & 0 & 0 \\ 2 & 4 & 1 & 0 \\ 0 & 2 & 6 & A \\ 0 & 0 & 4 & B \end{pmatrix} \;\; \text{et} \;\; \mathbf{b} = \begin{pmatrix} 1 \\ 2 \\ 4 \\ C \end{pmatrix}$$

Etudier selon *A*, *B* et *C* l'existence de la solution du système.

D'après la méthode d'élimination de Gauss, soit la transformation

$$\left(\begin{array}{cccc|c} 2 & 3 & 0 & 0 & 1 \\ 2 & 4 & 1 & 0 & 2 \\ 0 & 2 & 6 & A & 4 \\ 0 & 0 & 4 & B & C \end{array}\right) \sim \left(\begin{array}{cccc|c} 2 & 3 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 2 & 6 & A & 4 \\ 0 & 0 & 4 & B & C \end{array}\right) \sim \left(\begin{array}{cccc|c} 2 & 3 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 4 & A & 4 \\ 0 & 0 & 4 & B & C \end{array}\right) \sim \left(\begin{array}{cccc|c} 2 & 3 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 4 & A & 2 \\ 0 & 0 & 0 & B-A & C-2 \end{array}\right)$$

alors,

$$\begin{cases} 2x_1+3x_2=1 \\ x_2+x_3=1 \\ 4x_3+Ax_4=2 \\ (B-A)x_4=C-2 \end{cases} \Rightarrow \begin{cases} 2x_1+3x_2=1 \\ x_2+x_3=1 \\ 4x_3+Ax_4=2 \\ (B-A)x_4=C-2 \end{cases} \Rightarrow \begin{cases} x_1=1/2-(3/2)\left(1/2+(C-2)A/\left(4(B-A)\right)\right) \\ x_2=1/2+(C-2)A/\left(4(B-A)\right) \\ x_3=2-Ax_4=1/2-(C-2)A/\left(4(B-A)\right) \\ x_4=(C-2)/(B-A) \end{cases}$$

donc,

$$x_1=\frac{8A-2B-3AC}{8(B-A)},\ x_2=\frac{2B-4A+AC}{4(B-A)},\ x_3=\frac{2B-AC}{4(B-A)} \text{ et } x_4=\frac{C-2}{B-A}$$

Qui est une unique solution pour $A \neq B$.

En particulier, si $C=2$,

$$x_1=-\frac{1}{4},\ x_2=\frac{1}{2},\ x_3=\frac{1}{2} \text{ et } x_4=0$$

Exercice 5

Considérons le système $\mathbf{A}\mathbf{x}=\mathbf{b}$ avec,

$$\mathbf{A}=\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

Montrer que la méthode de Gauss-Seidel est convergente.

La matrice d'itération de la méthode itérative de Gauss-Seidel est :

$$\mathbf{M}=-(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}$$

avec,

$$\mathbf{D}=\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix},\ \mathbf{L}=\begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} \text{ et } \mathbf{U}=\begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

soit,

$$\mathbf{D}+\mathbf{L}=\begin{pmatrix} 2 & 0 & 0 \\ -1 & 2 & 0 \\ 0 & -1 & 2 \end{pmatrix}$$

La matrice inverse $(\mathbf{D}+\mathbf{L})^{-1}$ est de la forme :

$$(\mathbf{D}+\mathbf{L})^{-1}=\begin{pmatrix} 1/2 & 0 & 0 \\ \alpha & 1/2 & 0 \\ \beta & \gamma & 1/2 \end{pmatrix}$$

α, β et γ peuvent être déterminés à partir de :

$$(\mathbf{D}+\mathbf{L})\cdot(\mathbf{D}+\mathbf{L})^{-1}=\mathbf{I} \Leftrightarrow \begin{pmatrix} 2 & 0 & 0 \\ -1 & 2 & 0 \\ 0 & -1 & 2 \end{pmatrix}\cdot\begin{pmatrix} 1/2 & 0 & 0 \\ \alpha & 1/2 & 0 \\ \beta & \gamma & 1/2 \end{pmatrix}=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Par consequent,

$$-\frac{1}{2}+2\alpha=0 \Rightarrow \alpha=\frac{1}{4},\quad -\alpha+2\beta=0 \Rightarrow 2\beta=\alpha=\frac{1}{4} \Rightarrow \beta=\frac{1}{8}$$

$$-\frac{1}{2}+2\gamma=0 \Rightarrow 2\gamma=\frac{1}{2} \Rightarrow \gamma=\frac{1}{4}$$

alors,

$$(\mathbf{D}+\mathbf{L})^{-1}=\begin{pmatrix} 1/2 & 0 & 0 \\ 1/4 & 1/2 & 0 \\ 1/8 & 1/4 & 1/2 \end{pmatrix}$$

Finalement,

$$\mathbf{M}=-(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}=-\begin{pmatrix} 1/2 & 0 & 0 \\ 1/4 & 1/2 & 0 \\ 1/8 & 1/4 & 1/2 \end{pmatrix}\begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}=\begin{pmatrix} 0 & 1/2 & 0 \\ 0 & 1/4 & 1/2 \\ 0 & 1/8 & 1/4 \end{pmatrix}$$

La norme de la matrice **M** est :

$$\|\mathbf{M}\|_\infty=\max\left(\sum_{j=1}^{3}|a_{ij}|\right)=\max(0+1/2+0,\ 0+1/4+1/2,\ 0+1/8+1/4)=\max(1/2,\ 3/4,\ 3/8)=3/4$$

Puisque $\|\mathbf{M}\|_\infty<1$ donc, la méthode itérative de Gauss Seidel est convergente.

Exercice 6

Considérons le réseau hydraulique (Quarteroni *et al.*, 2010) composé de 10 conduites, représenté sur la figure ci-dessous (Fig. 2.6), alimenté par un réservoir d'eau à pression constante $p_0=10$ bar.

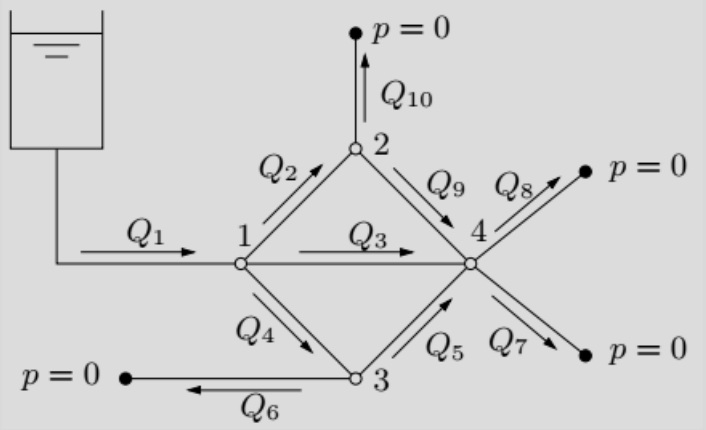


Fig. 2.6 : Réseau de conduites en hydraulique.

Dans ce problème, on convient de prendre la pression atmosphérique comme valeur de référence pour les pressions. Pour la *j*-ème conduite, on a la relation suivante entre le débit Q_j (en $m^3.s^{-1}$) et le saut de pression Δp_j entre l'entrée et la sortie

$$Q_j = \frac{\Delta p_j}{R_j L_j}$$

Où R est la résistance hydraulique par unité de longueur (en bar.s.m^{-4}) et L est la longueur (en m) de la conduite. On suppose que l'eau s'écoule par les sorties (indiquées par un point noir) où règne la pression atmosphérique, fixée à 0 bar (conformément à notre convention).

Le problème consiste à déterminer les valeurs de la pression en chaque nœud intérieur 1, 2, 3, 4. Pour cela, on complète les débits Q_j pour j = 1, 2, 3, 4 en écrivant que la somme des débits algébriques en un nœud j doit être nulle (une valeur négative indiquerait la présence d'une fuite).

En notant $\mathbf{p} = (p_1,\ p_2,\ p_3,\ p_4)^T$ le vecteur des pressions aux nœuds intérieurs, on obtient un système 4 × 4 de la forme $\mathbf{Ap} = \mathbf{b}$.

On indique dans le tableau suivant les caractéristiques des différentes conduites

Conduite	R	L	Conduite	R	L	Conduite	R	L
1	0.2500	20	2	2.0000	10	3	1.0204	14
4	2.0000	10	5	2.0000	10	6	7.8125	8
7	7.8125	8	8	7.8125	8	9	2.0000	10
10	7.8125	8						

La matrice **A** et le vecteur **b** sont donnés par (en ne conservant que les 4 premiers chiffres significatifs)

$$\mathbf{A} = \begin{pmatrix} -0.370 & 0.050 & 0.050 & 0.070 \\ 0.050 & -0.116 & 0 & 0.050 \\ 0.050 & 0 & -0.116 & 0.050 \\ 0.070 & 0.050 & 0.050 & -0.202 \end{pmatrix},\ \mathbf{b} = \begin{pmatrix} -2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

En utilisant la fonction **DecompositionLU** pour déterminer une solution du système. Soit l'introduction des données de la matrice **A** et le vecteur **b**,

```
>> A=[-0.370 0.050 0.050 0.070;0.050 -0.116 0 0.050;0.050 0 -0.116 0.050;
0.070 0.050 0.050 -0.202]
A =
   -0.3700    0.0500    0.0500    0.0700
    0.0500   -0.1160         0    0.0500
    0.0500         0   -0.1160    0.0500
    0.0700    0.0500    0.0500   -0.2020
>> b=[-2;0;0;0];
```

L'exécution de la fonction **DecompositionLU** permet d'avoir la pression dans les nœuds 1, 2, 3 et 4,

```
>> p=(DecompositionLU(A,b))'
p =
    8.1172    5.9893    5.9893    5.7779
```

Donc,

$$p_1 = 8.1172,\ p_2 = 5.9893,\ p_3 = 5.9893 \ \text{ et } \ p_4 = 5.779 \ \text{ bar}$$

Par conséquent les débits en $m^3.s^{-1}$, en chaque tronçon sont :

$$Q_1 = \frac{p_0 - p_1}{R_1 L_1} = \frac{10 - 8.1172}{0.2500 \times 20} = 0.37656 \qquad Q_2 = \frac{p_1 - p_2}{R_2 L_2} = \frac{8.1172 - 5.9893}{2.0000 \times 10} = 0.10640$$

$$Q_3 = \frac{p_1 - p_4}{R_3 L_3} = \frac{8.1172 - 5.7779}{1.0204 \times 14} = 0.16375 \qquad Q_4 = \frac{p_1 - p_3}{R_4 L_4} = \frac{8.1172 - 5.9893}{2.0000 \times 10} = 0.10640$$

$$Q_5 = \frac{p_3 - p_4}{R_5 L_5} = \frac{5.9893 - 5.7779}{2.0000 \times 10} = 0.01057 \qquad Q_6 = \frac{p_3 - p}{R_6 L_6} = \frac{5.9893 - 0.0000}{7.8125 \times 8} = 0.09583$$

$$Q_7 = \frac{p_4 - p}{R_7 L_7} = \frac{5.9893 - 0.0000}{7.8125 \times 8} = 0.09583 \qquad Q_8 = \frac{p_4 - p}{R_8 L_8} = \frac{5.9893 - 0.0000}{7.8125 \times 8} = 0.09583$$

$$Q_9 = \frac{p_2 - p_4}{R_9 L_9} = \frac{5.9893 - 5.7779}{2.0000 \times 10} = 0.01057 \qquad Q_{10} = \frac{p_2 - p}{R_{10} L_{10}} = \frac{5.9893 - 0.0000}{7.8125 \times 8} = 0.09583$$

D'après l'équation de continuité au nœuds ou principe de conservation de la masse,

Nœud 1 : $Q_1 - Q_2 - Q_3 - Q_4 = 0.37656 - 2 \times 0.10640 - 0.16375 = 0.00001 \approx 0$

Nœud 2 : $Q_2 - Q_9 - Q_{10} = 0.10640 - 0.01057 - 0.09583 = 0$

Nœud 3 : $Q_4 - Q_6 - Q_5 = 0.10640 - 0.09583 - 0.01057 = 0$

Nœud 4 : $Q_3 + Q_5 - Q_7 - Q_8 + Q_9 = 0.16375 + 0.01057 - 2 \times 0.09583 + 0.01057 = -0.00677 \approx 0$

la somme algébrique des débits dans chaque nœud est pratiquement nulle.

Exercice 7

Analyser la convergence des méthodes de Jacobi et Gauss-Seidel pour la résolution d'un système linéaire associé à la matrice

$$\mathbf{A} = \begin{pmatrix} \alpha & 0 & 1 \\ 0 & \alpha & 0 \\ 1 & 0 & \alpha \end{pmatrix}, \qquad \alpha \in \mathbb{R}$$

La matrice **A** est symétrique et s'écrit comme :

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U} = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \alpha \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

La matrice d'itération de Jacobi est :

$$\mathbf{M}_{\mathrm{J}} = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}) = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \alpha \end{pmatrix}^{-1} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1/\alpha & 0 & 0 \\ 0 & 1/\alpha & 0 \\ 0 & 0 & 1/\alpha \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1/\alpha \\ 0 & 0 & 0 \\ 1/\alpha & 0 & 0 \end{pmatrix}$$

alors,

$$\left\| \mathbf{M}_{\mathrm{J}} \right\|_{\infty} = \frac{1}{|\alpha|}$$

La matrice d'itération de Gauss-Seidel est :

$$\mathbf{M}_S = (\mathbf{D}+\mathbf{L})^{-1}\mathbf{U} = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 1 & 0 & \alpha \end{pmatrix}^{-1} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1/\alpha & 0 & 0 \\ 0 & 1/\alpha & 0 \\ -1/\alpha^2 & 0 & 1/\alpha \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1/\alpha \\ 0 & 0 & 0 \\ 0 & 0 & -1/\alpha \end{pmatrix}$$

alors,

$$\|\mathbf{M}_S\|_\infty = \frac{1}{|\alpha|}$$

Pour que les deux méthodes convergent, il faut que,

$$\frac{1}{|\alpha|} < 1 \;\Rightarrow \alpha < -1 \;\text{ et }\; \alpha > 1 \text{ c'est-à-dire, } \alpha \in]-\infty, -1[\cup]1, +\infty[$$

Exercice 8

Soit le système d'équations linéaire :

$$\begin{pmatrix} 4 & 0 & 1 \\ 1 & 4 & 1 \\ 1 & 0 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ -10 \end{pmatrix}$$

(a) Déterminer la solution du système en utilisant la méthode d'élimination de Gauss.
(b) Montrer si c'est possible de résoudre le système par la méthode itérative de Jacobi.
(c) Calculer les deux premières itérations avec la méthode itérative de Gauss-Seidel, pour une première itération le vecteur $(1, 2, -3)^T$.
(d) En utilisant les fonctions **`Jacobi`** et **`Seidel`** déterminer la solution du système et commenter sur les deux processus.

(a) Pour déterminer une solution du système par la méthode de d'élimination de Gauss, soit l'écriture suivante :

$$\left(\begin{array}{ccc|c} 4 & 0 & 1 & 5 \\ 1 & 4 & 1 & 3 \\ 1 & 0 & 4 & -10 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 0 & 1/4 & 5/4 \\ 1 & 4 & 1 & 3 \\ 1 & 0 & 4 & -10 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 0 & 1/4 & 5/4 \\ 0 & 4 & 3/4 & 7/4 \\ 0 & 0 & 15/4 & -45/4 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 4 & 0 & 1 & 5 \\ 0 & 4 & 3/4 & 7/4 \\ 0 & 0 & 15/4 & -45/4 \end{array}\right)$$

alors,

$$\begin{pmatrix} 4 & 0 & 1 \\ 1 & 4 & 1 \\ 1 & 0 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ -10 \end{pmatrix} \Rightarrow \begin{pmatrix} 4 & 0 & 1 \\ 0 & 4 & 3/4 \\ 0 & 0 & 15/4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 5 \\ 7/4 \\ -45/4 \end{pmatrix} \Rightarrow \begin{cases} 4x + z = 5 \\ 4y + \frac{3}{4}z = \frac{7}{4} \\ \frac{15}{4}z = -\frac{45}{4} \end{cases}$$

ce qui donne,

$$\frac{15}{4}z = -\frac{45}{4} \;\Rightarrow\; 15z = -45 \;\Rightarrow\; z = -\frac{45}{15} = -3$$

$$4y + \frac{3}{4}z = \frac{7}{4} \;\Rightarrow\; 4y = \frac{7}{4} - \frac{3}{4}z = \frac{16}{4} = 4 \;\Rightarrow\; y = 1$$

$$4x + z = 5 \;\Rightarrow\; 4x = 5 - z = 5 + 3 = 8 \;\Rightarrow\; x = 2$$

La solution du système est donc,

$$x=2,\ \ y=1\ \ \text{et}\ \ z=-3$$

(b) La matrice d'itération de Jacobi est exprimée par :

$$\mathbf{M}_{\mathrm{J}}=\mathbf{D}^{-1}\left(\mathbf{L}+\mathbf{U}\right)=\begin{pmatrix}4&0&0\\0&4&0\\0&0&4\end{pmatrix}^{-1}\begin{pmatrix}0&0&1\\1&0&1\\1&0&0\end{pmatrix}=\begin{pmatrix}1/4&0&0\\0&1/4&0\\0&0&1/4\end{pmatrix}\begin{pmatrix}0&0&1\\1&0&1\\1&0&0\end{pmatrix}=\begin{pmatrix}0&0&1/4\\1/4&0&1/4\\1/4&0&0\end{pmatrix}$$

or, $\left\|\mathbf{M}_{\mathrm{J}}\right\|_{\infty}=1/2<1$, donc la méthode d'itération de Jacobi est convergente.

(c) Le processus itérative de Gauss-Seidel peut être fondé comme suit,

$$\begin{cases}4x+z=5\\x+4y+z=3\\x+4z=-10\end{cases}\Rightarrow\begin{cases}x=\dfrac{5}{4}-\dfrac{1}{4}z\\y=\dfrac{3}{4}-\dfrac{1}{4}x-\dfrac{1}{4}z\\z=-\dfrac{10}{4}-\dfrac{1}{4}x\end{cases}$$

soit,

$$\begin{cases}x_{k+1}=\dfrac{5}{4}-\dfrac{1}{4}z_k\\y_{k+1}=\dfrac{3}{4}-\dfrac{1}{4}x_{k+1}-\dfrac{1}{4}z_k\\z_{k+1}=-\dfrac{10}{4}-\dfrac{1}{4}x_{k+1}\end{cases}$$

Pour $x_0=1,\ \ y_0=2\ \ \text{et}\ \ z_0=-3$,

$$\begin{cases}x_1=\dfrac{5}{4}-\dfrac{1}{4}z_0=\dfrac{5}{4}+\dfrac{3}{4}=\dfrac{8}{4}=2\\y_1=\dfrac{3}{4}-\dfrac{1}{4}x_1-\dfrac{1}{4}z_0=\dfrac{3}{4}-\dfrac{2}{4}+\dfrac{3}{4}=\dfrac{4}{4}=1\\z_1=-\dfrac{10}{4}-\dfrac{1}{4}x_1=-\dfrac{10}{4}-\dfrac{2}{4}=-\dfrac{12}{4}=-3\end{cases}$$

On remarque bien que le processus itératif de Gauss-Seidel converge vers la solution dès la première itération pour une première itération $(1, 2, -3)^T$.

(d) L'utilisation des fonctions **`Jacobi`** et **`Seidel`** après avoir entré la matrice **`A`**, le vecteur **`b`** et la première itération **`x0`**, conduit à :

```
>> A=[4 0 1;1 4 1; 1 0 4];
>> b=[5;3;-10];
>> x0=[1;2;-3];
>> [x, nombreIterations] = Jacobi(A,b,x0)
x =
    2.0000
    1.0000
   -3.0000
nombreIterations =
    11
```

```
>> [x, nombreIterations] = Siedel(A,b,x0)
x =
     2
     1
    -3
nombreIterations =
     1
```

Pour la même itération, la méthode de Jacobi converge dès la 11ième itération et la méthode de Gauss-Seidel converge dès la 1ière itération ce qui montre bien la rapidité de la méthode Gauss-Seidel par rapport à celle de Jacobi.

Exercice 9

Soit le système linéaire

$$\begin{cases} x_1 + 4x_2 = -15 \\ 5x_1 + x_2 = 1 \end{cases}$$

En gardant cette disposition du système, montrer que la méthode de Jacobi est divergente. Changer la disposition des équations dans le système et appliquer la méthode de Jacobi avec $\mathbf{x} = \left(1.01,\ -4.01\right)^T$ comme première itération et observer la convergence de la méthode dans ce cas.

Pour cette disposition, la matrice de Jacobi est donnée par :

$$\mathbf{M}_J = \mathbf{D}^{-1}\left(\mathbf{L}+\mathbf{U}\right) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 4 \\ 5 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 4 \\ 5 & 0 \end{pmatrix}$$

$\left\|\mathbf{M}_J\right\|_\infty = 5 > 1$, par conséquent la méthode de Jacobi diverge.

En changeant la disposition des équations, c'est-à-dire le système d'équations devient :

$$\begin{cases} 5x_1 + x_2 = 1 \\ x_1 + 4x_2 = -15 \end{cases}$$

La matrice d'itération de Jacobi devient :

$$\mathbf{M}_J = \mathbf{D}^{-1}\left(\mathbf{L}+\mathbf{U}\right) = \begin{pmatrix} 5 & 0 \\ 0 & 4 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1/5 & 0 \\ 0 & 1/4 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1/5 \\ 1/4 & 0 \end{pmatrix}$$

$\left\|\mathbf{M}_J\right\|_\infty = 1/4 < 1$, alors la méthode de Jacobi dans ce cas converge vers la solution.

Soit,

$$\begin{cases} 5x_1 + x_2 = 1 \\ x_1 + 4x_2 = -15 \end{cases} \Rightarrow \begin{cases} x_1 = \dfrac{1}{5} - \dfrac{1}{5}x_2 \\ x_2 = -\dfrac{15}{4} - \dfrac{1}{4}x_1 \end{cases} \Rightarrow \begin{cases} x_1^{(k+1)} = \dfrac{1}{5}\left(1 - x_2^{(k)}\right) \\ x_2^{(k+1)} = -\dfrac{1}{4}\left(15 + x_1^{(k)}\right) \end{cases}$$

Pour, $\mathbf{x}^{(0)} = \left(1.01,\ -4.01\right)^T$,

$$\begin{cases} x_1^{(1)} = \dfrac{1}{5}\left(1 + 4.01\right) = 1.00 \\ x_2^{(1)} = -\dfrac{1}{4}\left(15 + 1.01\right) = -4.00 \end{cases} \Rightarrow \begin{cases} x_1^{(2)} = \dfrac{1}{5}\left(1 + 4.0025\right) = 1.00 \\ x_2^{(2)} = -\dfrac{1}{4}\left(15 + 1.002\right) = -4.00 \end{cases}$$

Alors, la solution du système est $\mathbf{x} = \begin{pmatrix} 1, & -4 \end{pmatrix}^T$

6. Exercices supplémentaires

Exercice 1

En utilisant la méthode d'élimination naïve de Gauss manuellement ou avec la fonction MATLAB **`NaiveGaussElimination`**, trouver une solution du système tridiagonal suivant :

$$\begin{pmatrix} -4 & 1 & 0 & \cdots & 0 \\ 1 & -4 & 1 & \ddots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -4 & 1 \\ 0 & \cdots & 0 & 1 & -4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_9 \\ x_{10} \end{pmatrix} = \begin{pmatrix} 0.5 \\ 2 \\ \vdots \\ 2 \\ 0.5 \end{pmatrix}$$

Exercice 2

Les composantes de la matrice **A** de Hilbert d'ordre *n* sont définies par :

$$a_{ij} = \frac{1}{i + j - 1}$$

(a) Déterminer la matrice **A** pour $n = 5$.

(b) Si $\mathbf{b} = \begin{bmatrix} 1, & 0, & 0, & 0, & 0 \end{bmatrix}^T$, déterminer une solution du système $\mathbf{Ax} = \mathbf{b}$ en utilisant la méthode d'élimination de Gauss.

Exercice 3

La réaction chimique de l'Ethane (C_2H_6) avec l'oxygène (O_2) conduit à la formation de l'oxyde carbonique (CO_2) et l'eau (H_2O). Équilibrer cette équation chimique.

Exercice 4

On considère le système linéaire $\mathbf{Ax} = \mathbf{b}$,

$$\begin{pmatrix} 2 & -2 & 0 \\ \varepsilon - 2 & 2 & 0 \\ 0 & -1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ \varepsilon \\ 2 \end{pmatrix}$$

En utilisant la méthode d'élimination de Gauss, montrer que la solution du système est $\mathbf{x} = \begin{pmatrix} 1, & 1, & 1 \end{pmatrix}^T$ et n'est pas affectée ε.

Expliquer que ce passe si $\varepsilon \to 0$.

Exercice 5

Soit le système d'équation :

$$\begin{pmatrix} 15 & 6 & 8 & 11 \\ 6 & 6 & 5 & 3 \\ 8 & 5 & 7 & 6 \\ 11 & 3 & 6 & 9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 40 \\ 20 \\ 26 \\ 29 \end{pmatrix}$$

Résoudre le système avec la méthode de pivot partiel (utiliser la fonction **`Pivot2Gauss`**).

Exercice 6

Donner une condition suffisante sur β pour que les méthodes de Jacobi et de Gauss-Seidel convergent toutes les deux quand on les applique à un système associé à la matrice :

$$\mathbf{A} = \begin{pmatrix} -10 & 2 \\ \beta & 5 \end{pmatrix}$$

Exercice 7

Soit le système d'équations linéaires :

$$\begin{pmatrix} 4 & 1 \\ 2 & 5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

(a) Résoudre le système par la méthode de Jacobi en considérant comme première itération $x = 3,\ \ y = 11$.

(b) Résoudre le système par la méthode de Gauss-Seidel en considérant comme première itération $x = 3,\ \ y = 11$.

(c) Expliquer la différence de convergence entre les deux méthodes.

Exercice 8

Soit le système d'équations **Ax=b** :

$$\mathbf{A} = \begin{pmatrix} 1 & k \\ 2k & 1 \end{pmatrix},\ \ k \in \mathbb{R}$$

(a) Déterminer les valeurs de *k* pour que la matrice **A** à diagonale dominante stricte.

(b) Résoudre pour *k* = 0.25, le système avec la méthode de Jacobi.

Exercice 9

Utiliser la function **`Seidel`** avec $\mathbf{x}^{(0)} = \mathbf{0}$ pour résoudre le système linéaire :

$$\begin{pmatrix} 2 & -5 & 2 & 5 & 30 & -8 \\ -8 & 36 & -1 & 0 & 1 & 8 \\ 11 & -4 & 25 & 1 & -4 & 4 \\ 1 & 0 & -3 & 19 & 2 & 1 \\ 42 & -2 & -9 & 0 & 3 & 0 \\ -3 & 7 & -7 & 4 & 5 & -32 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 27 \\ 36 \\ 42 \\ 18 \\ 54 \\ 60 \end{pmatrix}$$

Si la matrice n'est pas à diagonal dominante, essayer de permuter ente les lignes de la matrice afin d'obtenir la matrice à diagonale dominante et résoudre le système avec la fonction **`Seidel`** avec $\mathbf{x}^{(0)} = \mathbf{0}$.

Répéter le même procédé de calcul en utilisant avec la fonction **`Jacobi`**.

Exercice 10

Calculer les rayons spectraux des méthodes de Jacobi et de Gauss-Seidel des matrices :

$$\begin{pmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{pmatrix} \text{ et } \begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix}$$

Et étudier la convergence des deux méthodes

Exercice 11

On donne les deux systèmes linéaires :

$$\begin{pmatrix} 10 & 1 \\ 2 & 10 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}=\begin{pmatrix} 11 \\ 12 \end{pmatrix} \text{ et } \begin{pmatrix} 1 & 10 \\ 10 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}=\begin{pmatrix} 11 \\ 12 \end{pmatrix}$$

a) Trouver les solutions exactes.

b) Résoudre chaque système par la méthode itérative de Jacobi, à partir du vecteur initial $\mathbf{x}^{(0)}=(0,\ 0)^T$, jusqu'au vecteur $\mathbf{x}^{(4)}$ inclus.

c) Expliquer qualitativement le comportement différent des deux suites de vecteurs.

d) Résoudre les deux systèmes de la question par la méthode itérative de Gauss-Seidel, en partant du vecteur nul et en poussant le calcul jusqu'à $\mathbf{x}^{(3)}$ inclus.

Exercice 12

La solution du système **Ax** = **b** s'interprète géométriquement comme le point d'intersection des deux droites

$$(D_1) \quad a_{11}x+a_{12}y=b_1$$
$$(D_2) \quad a_{21}x+a_{22}y=b_2$$

On suppose que $a_{11}\neq 0$ et $a_{22}\neq 0$.

1. Calculer les matrices d'itération des méthodes de Jacobi et de Gauss-Seidel associées à ce système.

2. Calculer les rayons spectraux de ces matrices. Que remarque-t-on ?

3. Calculer les rayons spectraux des matrices des méthodes de Jacobi et de Gauss-Seidel associées au système obtenu en permutant les deux équations ci-dessus.

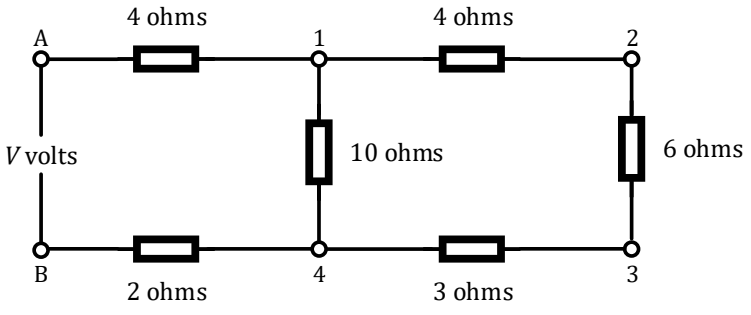
4. Interpréter géométriquement les méthodes de Jacobi et de Gauss-Seidel appliquées à **Ax=b**.

Pour cette dernière, on notera que le vecteur $\mathbf{x}^{(k+1)}$ est solution du système triangulaire

$$\begin{cases} a_{11}x_{k+1}+a_{12}y_k=b_1 \\ a_{21}x_{k+1}+a_{22}y_{k+1}=b_2 \end{cases}$$

Exercice 13

L'application des lois d'Ohm et de Kirchhoff aux circuit électrique permet d'avoir le système d'équations linéaires suivant :



$$\begin{aligned} 3v_1 - 5v_2 + 2v_3 &= 0 \\ -2v_1 + 5v_2 + 2v_4 &= 5V \\ v_2 - 3v_3 + 2v_4 &= 0 \\ 3v_1 + 10v_3 - 28v_4 &= 0 \end{aligned}$$

V est la différence de potentielle entre A et B. Calculer les potentiels en chaque nœud si $V = 40.80$ volts.

Exercice 14

Soit le système **Ax** = **b**, avec :

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & -1/4 & -1/4 \\ 0 & 1 & -1/4 & -1/4 \\ -1/4 & -1/4 & 1 & 0 \\ -1/4 & -1/4 & 0 & 1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \quad \text{et} \quad \mathbf{b} = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

1. Calculer les matrices d'itérations des méthodes de Jacobi $\mathbf{M}_J$ et de Gauss-Seidel $\mathbf{M}_S$ et de la relaxation (SOR) $\mathbf{M}_\omega$ associées à **A**.

2. Donner les expressions des processus itératives de résolution des méthodes de Jacobi et de Gauss-Seidel quand le point initial est l'origine.

3. Calculer les rayons spectraux des matrices $\mathbf{M}_J$ et $\mathbf{M}_S$.

4. a) Montrer que si λ est valeur propre de $\mathbf{M}_\omega$ alors $\lambda = 1 - \omega$ ou bien λ est racine de l'équation :

$$\lambda^2 - \left(2(1-\omega) + \frac{\omega^2}{4}\right)\lambda + (1-\omega)^2 = 0$$

b) Calculer $\rho(\mathbf{M}_\omega)$ en distinguant les cas où les racines de l'équation précédente sont réelles ou non.

c) Trouver la valeur de ω qui rend $\rho(\mathbf{M}_\omega)$ minimum.

Exercice 15

La méthode des moindres carrées est souvent utilisée pour déterminer une relation polynomiale pour représenter les données expérimentale. Si le polynôme de troisième degré $y = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ est utilisé, les coefficients du polynôme sont déterminés en résolvant le système formé par les équations :

$$\begin{aligned} a_o n \quad &+ a_1 \sum_{i=1}^{n} x_i + a_2 \sum_{i=1}^{n} x_i^2 + a_3 \sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} y_i \\ a_o \sum_{i=1}^{n} x_i &+ a_1 \sum_{i=1}^{n} x_i^2 + a_2 \sum_{i=1}^{n} x_i^3 + a_3 \sum_{i=1}^{n} x_i^4 = \sum_{i=1}^{n} y_i x_i \\ a_o \sum_{i=1}^{n} x_i^2 &+ a_1 \sum_{i=1}^{n} x_i^3 + a_2 \sum_{i=1}^{n} x_i^4 + a_3 \sum_{i=1}^{n} x_i^5 = \sum_{i=1}^{n} y_i x_i^2 \\ a_o \sum_{i=1}^{n} x_i^3 &+ a_1 \sum_{i=1}^{n} x_i^4 + a_2 \sum_{i=1}^{n} x_i^5 + a_3 \sum_{i=1}^{n} x_i^6 = \sum_{i=1}^{n} y_i x_i^3 \end{aligned}$$

Déterminer les coefficients du polynôme pour les données suivantes :

X	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5
Y	4.3	6.2	10.1	13.5	19.8	22.6	24.7	29.2

Exercice 16

Considérons la conduction thermique dans un petit fil transportant un courant électrique qui produit de la chaleur à un taux constant. L'équation décrivant la température $T(x)$ le long du fil $(0 \le x \le 1 \text{ cm})$ est :

$$D\frac{d^2T}{dx^2} = -S$$

Avec, les conditions frontières $T(0) = T(1) = 0°\text{C}$, le coefficient de thermo-diffusion $D = 0.01 \text{ cm}^2/\text{s}$ et le terme de source normalisée S = 1°C/s.

Si on discrétise le domaine en 20 sous-intervalles égaux, en utilisant $x_j = j/20$ pour j = 0 à 20, l'équation peut être approchée par :

$$D\frac{T_{j-1} - 2T_j + T_{j+1}}{h^2} = -S$$

où, T_j est la température à $x = x_j$ et h = 0.05 est le pas de discrétisation. L'équation discrétisée en tenant compte les conditions aux limites à x_0 et x_{20} à système d'équation linéaires **Ay = b** de 19 inconnues de températures de y_1 à y_{19} :

$$\mathbf{A} = \begin{pmatrix} -2 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & -2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0 & 0 & \cdots & 0 & 1 & -2 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & -2 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ \vdots \\ y_{18} \\ y_{19} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} -0.25 \\ -0.25 \\ -0.25 \\ \vdots \\ \vdots \\ -0.25 \\ -0.25 \end{pmatrix}$$

Résoudre avec la méthode itérative de Jacobi et de Gauss-Seidel avec comme première itération **y** = **0**.

Chapitre 3.

Résolution des équations non linéaires

1. Motivation

La résolution d'une équation numérique $f(x)=0$ ou un système d'équations numériques $\mathbf{f}(\mathbf{x})=\mathbf{0}$ sont parmi les problèmes mathématiques. Soit le cas de l'équation polynomiale de second degré :

$$ax^2+bx+c=0$$

Qui possède comme solution analytique quand $b^2-4ac\geq 0$:

$$x_1=\frac{-b-\sqrt{b^2-4ac}}{2a} \quad \text{et} \quad x_2=\frac{-b+\sqrt{b^2-4ac}}{2a}$$

Les deux solutions x_1 et x_2 sont appelées aussi les racines ou les zéros de la fonction $f(x)=ax^2+bx+c$.

Examinons le cas de l'équation $x-e^{-x}=0$, le tracer des fonctions $y=x$ et $y=e^{-x}$ (Fig. 3.1) montre qu'il y a une intersection entre les deux graphes dont l'abscisse est x_0 solution de l'équation $x=e^{-x}$ qui peut être déterminé graphiquement.

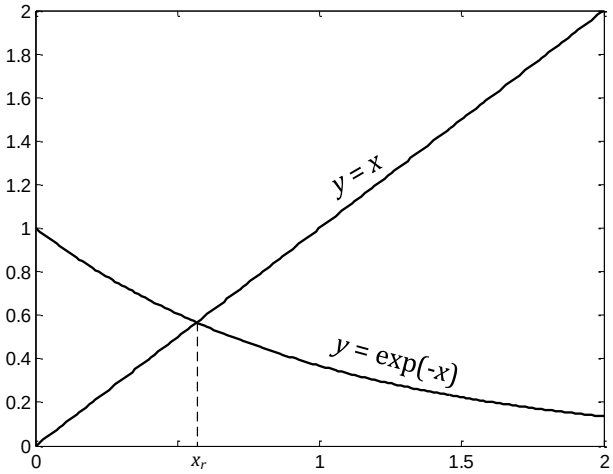


Fig. 3.1 : Résolution graphique de l'équation $x-e^{-x}=0$.

La résolution analytique de telle équation est impossible, ce qui nous ramène à utiliser des techniques numériques afin de déterminer une solution approchée de l'équation.

Plusieurs méthodes numériques sont utilisées pour résoudre de tels problèmes à savoir la méthode de bissection, de point fixe, de Newton-Raphson et autres…

2. Méthode de la bissection

La méthode de la bissection (Fig. 3.2) ou de dichotomie consiste à répéter des partages d'un intervalle en deux parties puis à sélectionner le sous-intervalle dans lequel existe la racine de la fonction $f(x)$.

Soit deux nombres réels a et b et une fonction réelle $f(x)$ continue sur l'intervalle [a, b] telle que $f(a).f(b)<0$ c'est-à-dire $f(a)$ et $f(b)$ soient de signes opposés. D'après le théorème des valeurs intermédiaires, $f(x)$ a au moins un zéro dans l'intervalle [a, b]. La méthode de bissection consiste à diviser l'intervalle en deux en calculant $m=(a+b)/2$. Si

$f(m) \cong 0$, alors *m* est la solution de l'équation $f(x) = 0$, sinon, il y a deux possibilités, soit $f(a).f(m) < 0$ ou bien $f(m).f(b) < 0$. Dans le cas ou $f(a).f(m) < 0$ *b* prend la valeur de *m* de même si $f(m).f(b) < 0$ *a* prend la valeur de *m*, le processus est répété jusqu'à l'arrivée à $f(m) \cong 0$.

L'algorithme de bissection est alors appliqué au sous-intervalle dans lequel le changement de signe se produit, ce qui signifie que l'algorithme de bissection est récursif. L'erreur absolue de la méthode de bissection est au plus de :

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

Après *n* étapes car l'erreur est diminuée de moitié à chaque étape. Ainsi, la méthode de bissection converge linéairement, ce qui est très lent par comparaison à d'autres méthodes numériques. Sous l'hypothèse que le signe de $f(m)$ soit déterminable.

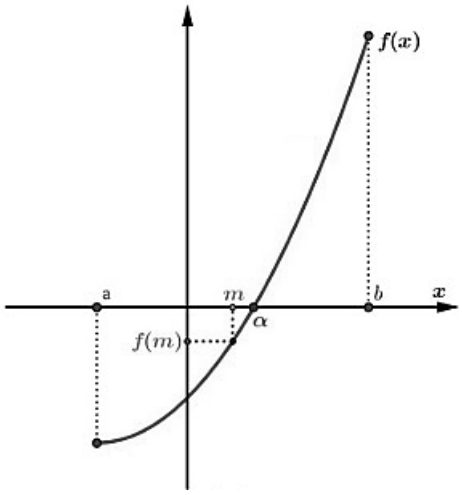


Fig. 3.2 : Méthode de bissection.

L'exemple suivant, consiste à chercher la racine de la fonction $f(x) = (5-x)e^x - 3$ dans l'intervalle [4, 6] par la méthode de bissection.

Pour répondre à cette question, vérifions les valeurs de cette fonction dans les bornes de l'intervalle [4, 6] :

$$f(4) = (5-4)e^4 - 3 = 51.5982 \quad , \quad f(6) = (5-6)e^6 - 3 = -406.4288$$

Puisque $f(4).f(6) < 0$ alors, d'après le théorème des valeurs intermédiaires, il existe une racine de la fonction $f(x) = (5-x)e^x - 3$ dans l'intervalle I_1 = [4, 6] :

$m = (4+6)/2 = 5$	$f(5) = -3$	I_2 = [4, 5]
$m = (4+5)/2 = 4.5$	$f(4.5) = 42.086$	I_3 = [4.5, 5]
$m = (4.5+5)/2 = 4.75$	$f(4.75) = 25.8961$	I_4 = [4.75, 5]
$m = (4.75+5)/2 = 4.8750$	$f(4.8750) = 13.3718$	I_5 = [4.8750, 5]
$m = (4.8750+5)/2 = 4.9375$	$f(4.9375) = 5.7138$	I_6 = [4.9375, 5]
$m = (4.9375+5)/2 = 4.9688$	$f(4.9688) = 1.4952$	I_7 = [4.9688, 5]
$m = (4.9688+5)/2 = 4.9844$	$f(4.9844) = -0.7206$	I_8 = [4.9688, 4.9844]
$m = (4.9688+4.9844)/2 = 4.9766$	$f(4.9766) = 0.3925$	I_9 = [4.9766, 4.9844]
$m = (4.9766+4.9844)/2 = 4.9805$	$f(4.9805) = -0.1618$	I_{10} = [4.9766, 4.9805]
$m = (4.9766+4.9805)/2 = 4.9786$	$f(4.9786) = 0.1159$	I_{11} = [4.9786, 4.9805]
$m = (4.9786+4.9805)/2 = 4.9795$	$f(4.9795) = -0.0264$	I_{12} = [4.9786, 4.9795]
$m = (4.9786+4.9795)/2 = 4.9791$	$f(4.9791) = 0.0448$	I_{13} = [4.9791, 4.9795]
$m = (4.9791+4.9795)/2 = 4.9793$	$f(4.9793) = 0.0092$	I_{14} = [4.9793, 4.9795]
$m = (4.9793+4.9795)/2 = 4.9794$	$f(4.9794) = -0.0052$	I_{15} = [4.9793, 4.9794]
$m = (4.9793+4.9794)/2 = 4.97940$	$f(4.97940) = 0.0021$	I_{16} = [4.9740, 4.9794]

Le calcul à 0.001 près, montre bien que la racine de la fonction $f(x)=(5-x)e^{x}-3$ est au voisinage de 4.974.

La fonction **Bissection** suivante permet d'avoir après son exécution la vraie valeur de la racine cherchée.

```
function [x,k] = Bissection(f,a,b,varargin)
% f, nom de la fonction tel que f(x)=0
% [a, b], intervalle de la solution de f(x)=0 (a < b)
% x, solution de l'équation f(x)=0
% k, nombre d'intervalles subdivisés
if nargin<3,error('à la limite 3 arguments sont nécessaires'),end
if ~(b>a),error('Attention, il faut que a < b'),end
eps=0.000001; % Définition de la précision souhaitée
for k=1:1000
   x=(a+b)/2;
    if abs(f(x,varargin{:}))<=eps
       break
    elseif f(a,varargin{:})*f(x,varargin{:})<0
      b=x;
    else
      a=x;
    end
end
end
```

L'exécution de cette fonction pour chercher la racine de la fonction $f(x)=(5-x)e^{x}-3$ dans l'intervalle [4, 6] conduit à :

```
>> [x,k] = Bissection(@(x)(5-x).*exp(x)-3,4,6)
x =
    4.9794
k =
    27
```

L'exécution de la fonction **Bissection**, montre que la racine approchée 4.9794 de la fonction $f(x)=(5-x)e^{x}-3$ dans l'intervalle [4, 6] est atteinte après 27 subdivisions pour une erreur de l'ordre 10^{-6}.

3. Méthode du point fixe

Le principe de cette méthode (Fig. 3.3) consiste à transformer l'équation $f(x)=0$ en une équation équivalente à $x=g(x)$ et construire le processus itératif :

$$x_{k+1}=g(x_k)$$

La première itération x_1 est proposée et soit proche de la solution probable de l'équation $f(x)=0$. Le processus converge vers la solution de l'équation ou arrêté jusqu'à la satisfaction du critère de convergence dite aussi l'erreur de l'estimation :

$$\varepsilon_a=\left|\frac{x_{i+1}-x_i}{x_i}\right|$$

Le choix de la fonction $g(x)$ est motivé par les exigences du théorème du point fixe. En effet, elle doit être contractante dans le voisinage de la solution. Le processus du point fixe est :

$$x_{k+1} = g(x_k)$$

Si x_r est la solution de l'équation cela veut dire que :

$$x_r = g(x_r)$$

Soit alors l'erreur commise à l'itération k+1 :

$$e_{k+1} = |x_r - x_{k+1}| = |g(x_r) - g(x_k)|$$

D'après le théorème moyenne de la dérivée, si $g(x)$ une fonction continue dans l'intervalle [α, β] alors il existe ξ de cet intervalle de façon que :

$$g'(\xi) = \frac{g(\beta) - g(\alpha)}{\beta - \alpha}$$

Par conséquent,

$$g(\beta) - g(\alpha) = g'(\xi)(\beta - \alpha)$$

Effectuant le changement de variables $\beta = x_r$ et $\alpha = x_k$, soit alors :

$$x_r - x_{k+1} = g'(\xi)(x_r - x_k) = g'(\xi)$$

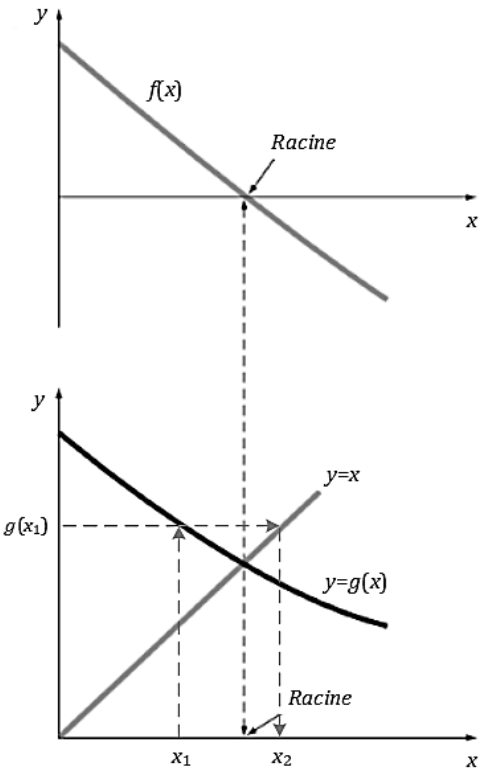


Fig. 3.3 : Méthode du point fixe.

En valeur absolue,

$$\frac{e_{k+1}}{e_k} = |g'(\xi)|, \ \forall \ x_k \le \xi \le x_r \ (\text{ou}, \ x_r \le \xi \le x_k)$$

De cette équation, il est clair que ce rapport diminue, c'est-à-dire, l'erreur décroit avec l'itération si est seulement si :

$$|g'(x)| < 1$$

Ce qui conduit donc à la convergence du processus du point fixe (Fig. 3.4), dans le cas contraire, le processus diverge (Fig. 3.5).

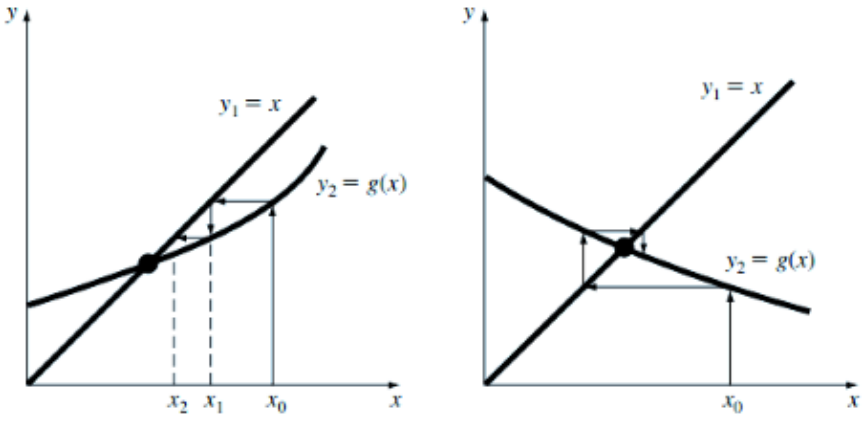


Fig. 3.4 : Processus convergent du point fixe.

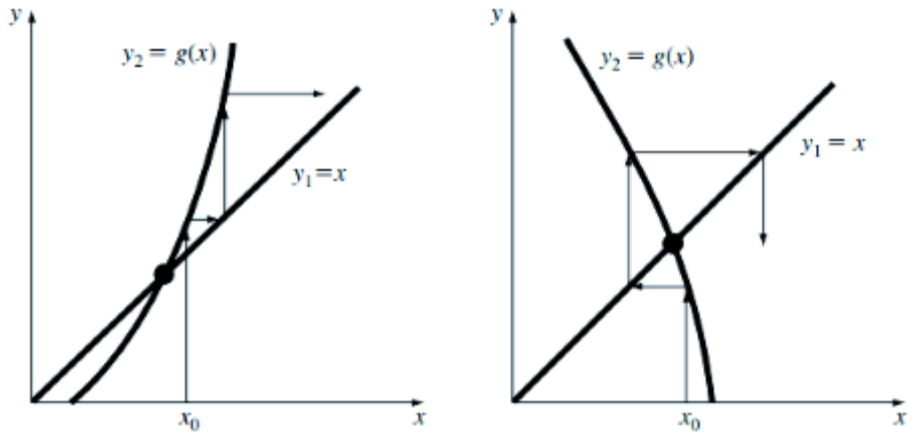


Fig. 3.5 : Processus divergent du point fixe.

Une fois le critère de convergence est vérifié, alors il est possible de déterminer l'itération à partir du quelle commence la convergence du processus pour une tolérance ε.

Soit $\mathrm{M} = \max\left|g'(x)\right|,\ \forall\ a \le x \le b$, donc :

$$\left|g'(x)\right| \le \mathrm{M} < 1\ \ \forall\ a \le x \le b \ \ \Rightarrow\ \ e_{k+1} \le \mathrm{M}e_k \le \mathrm{M}^2 e_{k-1} \le \mathrm{M}^3 e_{k-2} \le \cdots \le \mathrm{M}^{1+\eta} e_{k-\eta}$$

Pour $\eta = k$, soit,

$$e_{k+1} \le \mathrm{M}^{k+1} e_0 \ \Rightarrow\ e_k \le \mathrm{M}^k e_0 \ \text{ avec } \ e_0 = \left|x_r - x_0\right|$$

or,

$$e_0 = \left|x_r - x_0\right| = \left|x_r - x_1 + x_1 - x_0\right| \le \left|x_r - x_1\right| + \left|x_1 - x_0\right| = e_1 + \left|x_1 - x_0\right|$$

alors,

$$e_0 \le \mathrm{M}e_0 + \left|x_1 - x_0\right| \ \Rightarrow\ e_0 - \mathrm{M}e_0 \le \left|x_1 - x_0\right| \ \Rightarrow\ e_0 \le \frac{\left|x_1 - x_0\right|}{1 - \mathrm{M}}$$

par conséquent,

$$e_k \le \mathrm{M}^k \frac{\left|x_1 - x_0\right|}{1 - \mathrm{M}}$$

Pour une tolérance ε, c'est-à-dire $e_k \le \varepsilon$, alors :

$$\mathrm{M}^k \frac{\left|x_1 - x_0\right|}{1 - \mathrm{M}} \le \varepsilon \ \Rightarrow\ \mathrm{M}^k \le \frac{\varepsilon(1 - \mathrm{M})}{\left|x_1 - x_0\right|} \ \Rightarrow\ k \ln \mathrm{M} \le \ln \frac{\varepsilon(1 - \mathrm{M})}{\left|x_1 - x_0\right|}$$

Or, $0 < \mathrm{M} < 1$, ce qui implique que :

$$\ln \mathrm{M} < 0 \ \text{ et } \ \ln \frac{\varepsilon(1 - \mathrm{M})}{\left|x_1 - x_0\right|} < 0 \ \Rightarrow\ k \ge \frac{1}{\ln \mathrm{M}} \ln \frac{\varepsilon(1 - \mathrm{M})}{\left|x_1 - x_0\right|}$$

C'est l'itération minimale à partir du quelle commence la convergence du processus itératif.

Soit l'exemple précédent de l'équation $(5 - x)e^x - 3 = 0$ qui peut être transformée comme :

$$(5 - x)e^x - 3 = 0 \ \Rightarrow\ x = \ln 3 - \ln(5 - x) \ \Rightarrow\ \left|g_1'(x)\right| = \frac{1}{\left|5 - x\right|}$$

$$(5 - x)e^x - 3 = 0 \ \Rightarrow\ x = 5 - \frac{3}{e^x} \ \Rightarrow\ \left|g_2'(x)\right| = \frac{3}{e^x}$$

Sachant que la solution cherchée est positive et pour le premier cas, il faut que $5-x>0 \Rightarrow 0<x<5$ dans cette intervalle $\left|g_1'(x)\right|>1$ par conséquent le processus itératif $x_{k+1}=g_1'(x_k)$ est divergent.

Pour le deuxième cas où, $\ln 3<x\le 5 \Rightarrow \left|g_2'(x)\right|<1$, le processus itératif $x_{k+1}=g_2(x_k)$ converge. Soit le script MATLAB qui permet de déterminer la solution par la méthode du point fixe pour une première itération $x_1 = 1$.

```
g=@(x) 5-(3./exp(x));% Définition de g(x)
xin = 1.5; % Définition de la valeur initiale
X(1)=xin;
xfin=g(xin); % Définition de la précision souhaitée
k=0;
X(k+1)=xfin;
% Définition de la précision souhaitée
eps = 0.0001;
while abs(xfin-xin)> eps
    k=k+1;
    xin = xfin; xfin = g(xin); X(k+1)=xfin;
end
X=X';
disp ('Convergence atteinte dès l itération:'),k
disp ('La solution de l équation est :'),X(k)
```

Tab. 3.1 : Résultats.

k	x_k
0	10
1	4.9998638
2	4.9797834
3	4.9793733
4	4.9793649
5	4.9793647
6	4.9793647

L'exécution de ce script permet d'avoir (Tab. 3.1) :

```
Convergence atteinte dès l itération:
k =
     4
La solution de l équation est :
ans =
    4.9794
```

C'est-à-dire le processus converge vers la solution 4.9794 dès la quatrième itération.

Dans l'intervalle [1.5, 5] :

$$\mathrm{M}=\max_{1.5\le x\le 5}\left|g_2'(x)\right|=\frac{3}{e^{1.5}}=0.6694$$

Pour $x_0=1.5$, et $\varepsilon=0.0001$, l'itération minimale certaine à partir du quelle commence la convergence est :

$$k_{\min}=\frac{1}{\ln \mathrm{M}}\ln\frac{\varepsilon(1-\mathrm{M})}{\left|x_1-x_0\right|}=28.2960\approx 28$$

Alors que le processus converge dès la quatrième itération pour $\varepsilon=0.0001$. $k_{\min}$ ici est la valeur la plus certaine de la convergence du processus itératif.

Extension vers la résolution d'un système d'équations non linéaires

Cette méthode consiste à transformer le système d'équations non-linéaires comme suit :

$$\mathbf{f}(\mathbf{x})=\mathbf{0} \Rightarrow \mathbf{x}=\mathbf{g}(\mathbf{x})$$

Qui s'écrit analytiquement :

$$x_1 = g_1\left(x_1,\ x_2,\ x_3,\ \ldots,\ x_n\right)$$
$$x_2 = g_2\left(x_1,\ x_2,\ x_3,\ \ldots,\ x_n\right)$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$x_n = g_n\left(x_1,\ x_2,\ x_3,\ \ldots,\ x_n\right)$$

Il faut noter qu'il n'est pas obligatoire d'extraire la première variable de la première équation, mais nous avons une multitude de combinaison possible. Le choix du schéma obtenu est régi par la condition de convergence. En on passe au schéma de récurrence suivant :

$$\mathbf{x}_{k+1} = \mathbf{g}\left(\mathbf{x}_k\right)$$

c'est-à-dire,

$$x_1^{(k+1)} = g_1\left(x_1^{(k+1)},\ x_2^{(k+1)},\ x_3^{(k+1)},\ \ldots,\ x_n^{(k+1)}\right)$$
$$x_2^{(k+1)} = g_2\left(x_1^{(k+1)},\ x_2^{(k+1)},\ x_3^{(k+1)},\ \ldots,\ x_n^{(k+1)}\right)$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$x_n^{(k+1)} = g_n\left(x_1^{(k+1)},\ x_2^{(k+1)},\ x_3^{(k+1)},\ \ldots,\ x_n^{(k+1)}\right)$$

Choisir un vecteur de départ $\mathbf{x}_1 = \left(x_1^{(1)},\ x_2^{(1)},\ x_3^{(1)},\ \ldots,\ x_n^{(1)}\right)$ puis déterminer la succession des vecteurs $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$, $\mathbf{x}_5$, ... jusqu'à la convergence du processus, c'est-à-dire pour que le critère suivant soit vérifié :

$$\left\|\mathbf{x}_{s+1} - \mathbf{x}_s\right\| \le \varepsilon$$

Pour obtenir la convergence d'un schéma de point fixe, la condition de convergence suivante doit être vérifiée :

$$\left|\frac{\partial g_1}{\partial x_1}\right| + \left|\frac{\partial g_2}{\partial x_1}\right| + \left|\frac{\partial g_3}{\partial x_1}\right| + \ldots + \left|\frac{\partial g_n}{\partial x_1}\right| < 1,$$
$$\left|\frac{\partial g_1}{\partial x_2}\right| + \left|\frac{\partial g_2}{\partial x_2}\right| + \left|\frac{\partial g_3}{\partial x_2}\right| + \ldots + \left|\frac{\partial g_n}{\partial x_2}\right| < 1,$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$\left|\frac{\partial g_1}{\partial x_n}\right| + \left|\frac{\partial g_2}{\partial x_n}\right| + \left|\frac{\partial g_3}{\partial x_n}\right| + \ldots + \left|\frac{\partial g_n}{\partial x_n}\right| < 1,$$

Soit l'exemple d'un système d'équations non linéaires à deux inconnus :

$$\begin{cases} x^2 + y^2 - 4 = 0, \\ \exp(x) + y - 1 = 0. \end{cases} \text{ avec } \begin{pmatrix} x_1 = -2 \\ y_1 = 1 \end{pmatrix}$$

Il est clair que dans ce système, $f_1\left(\mathbf{x}\right) = x^2 + y^2 - 4$, $f_2\left(\mathbf{x}\right) = \exp\left(x\right) + y - 1$ et $\mathbf{x} = \left(x,\ y\right)$.

alors,

$$\begin{cases} x^2 + y^2 - 4 = 0, \\ \exp(x) + y - 1 = 0. \end{cases} \Rightarrow \begin{cases} x = \left(4 - y^2\right)/x, \\ y = 1 - \exp(x). \end{cases}$$

Ici $g_1(x,\ y)=\left(4-y^2\right)/x$, $g_2(x,\ y)=1-\exp(x)$, par conséquent le processus de point fixe peut être explicité comme suit :

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix}=\begin{pmatrix} \left(4-y_k^2\right)/x_k \\ 1-\exp\left(x_k\right) \end{pmatrix} \text{ avec, } \begin{pmatrix} x_1=-2 \\ y_1=1 \end{pmatrix}$$

Soit le script MATLAB qui permet la résolution de ce système en basant sur ce processus de récurrence :

```
g1=@(x,y)(4-y^2)/x; g2=@(x,y)1-exp(x);
x(1)=-2; y(1)=1;
eps=0.0001;
for k=1:100
 x(k+1)=g1(x(k),y(k)); y(k+1)=g2(x(k),y(k));
 if sqrt((x(k+1)-x(k))^2+(y(k+1)-y(k))^2) <= eps
  break
 end
end
```

L'utilisation de ce script pour différentes valeurs de ε, conduit aux résultats suivants dans le tableau ci-dessous, la valeur de k correspondant la valeur de l'itération de la convergence du processus.

Tab. 3.2 : Résultats.

ε	0.01	0.001	0.0001	0.00001
K	23	34	46	58
x_k	-1.811535	-1.816772	-1.816309	-1.816259
y_k	0.838310	0.837267	0.837359	0.837369
$f_1(x_k, y_k)$	-0.015576	0.001675	0.000147	0.0000157
$f_2(x_k, y_k)$	0.001713	-0.000184	-0.000016	-0.0000017

4. Méthode de Newton-Raphson

La méthode de résolution des équations numériques a été initiée par Newton vers 1669 sur des exemples numériques mais la formulation était fastidieuse. Dix ans plus tard, Joseph Raphson met en évidence une formule de récurrence. Un siècle plus tard, Mouraille et Lagrange étudient la convergence des approximations successives en fonction des conditions initiales par une approche géométrique. Cinquante ans plus tard, Fourier et Cauchy s'occupe de la rapidité de la convergence. La méthode consiste à introduire une suite $\left(x_k\right)$ d'approximations successives de l'équation $f\left(x\right)=0$.

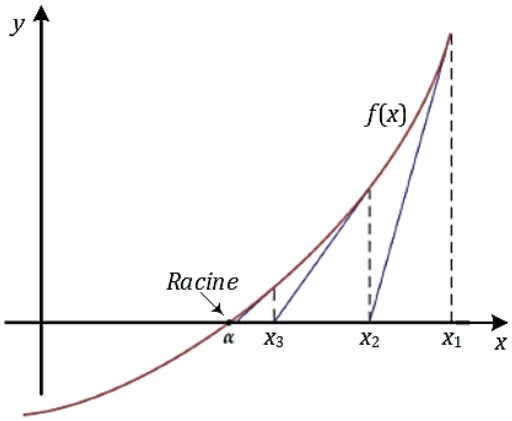


Fig. 3.6 : Méthode de Newton-Raphson.

Du point dont l'abscisse x_1 la droite tangente (Fig. 3.6) à pour équation :

$$y = f'(x_1)(x - x_1) + f(x_1)$$

Cette tangente coupe l'axe des abscisses en x_2, en termes d'équation :

$$0 = f'(x_1)(x_2 - x_1) + f(x_1) \;\Rightarrow\; x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

À son tour, la valeur de x_2 permettra de considérer un deuxième point $\left(x_2,\ f(x_2)\right)$. À nouveau, l'abscisse x_3 point d'intersection de la deuxième tangente avec l'axe des x sera considérée comme une troisième, c'est-à-dire :

$$0 = f'(x_2)(x_3 - x_2) + f(x_2) \;\Rightarrow\; x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

En poursuivant ce procédé itérativement, on obtiendra, sous certaines conditions, une séquence de valeurs $x_1,\ x_2,\ x_3,\ \ldots$ reliés par :

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Qui vont se rapprocher de plus en plus de la solution de l'équation $f(x) = 0$.

Lorsque la suite converge, elle converge de façon quadratique c'est-à-dire que le nombre de chiffres significatifs double à chaque itération. Si l'on s'en tient à une précision inférieure à 10^{-15}, la suite doit alors converger en moins de dix itérations. On pourra mettre une condition d'arrêt. Il faudra de prendre un x_1 plus proche de la racine α pour que le processus converge rapidement vers la solution.

Dans le cas où la première itération x_1 est très loin de la solution, il y a une possibilité de divergence du processus de Newton-Raphson (Fig. 3.7).

Le processus converge vers la solution x_r de l'équation ou arrêté jusqu'à la satisfaction du critère de convergence telle que :

$$\left|x_{i+1} - x_i\right| \le \varepsilon$$

Par l'analogie avec la méthode du point fixe, l'équation :

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \;\Rightarrow\; g(x) = x - \frac{f(x)}{f'(x)}$$

La dérivée de la fonction $g(x)$ est :

$$g'(x) = 1 - \frac{f'(x)f'(x) - f(x)f''(x)}{\left(f'(x)\right)^2} = \frac{f(x)f''(x)}{\left(f'(x)\right)^2}$$

Quand $x = x_r$, $f(x_r) = 0 \;\Rightarrow\; g'(x_r) = 0$ ce qui signifie que x_r est un optimum de la fonction $g(x)$.

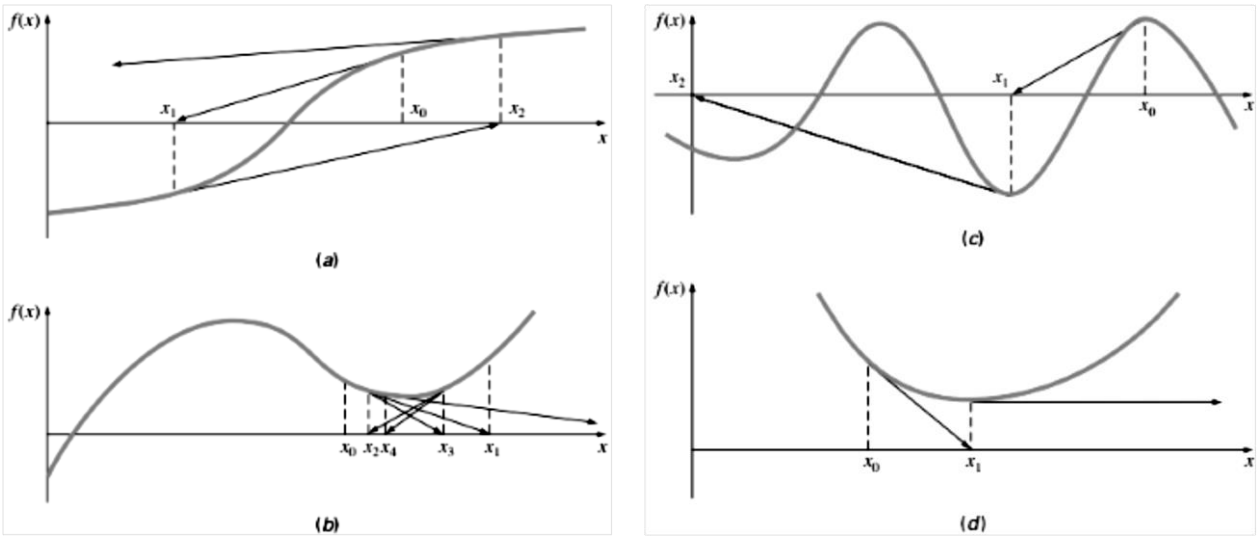


Fig. 3.7 : Possibilités de divergence du processus de Newton-Raphson.

Par définition l'erreur commise à l'itération k+1, $e_{k+1} = \left|x_{k+1} - x_r\right| = \left|g\left(x_k\right) - g\left(x_r\right)\right|$. Le développement de Taylor d'ordre 2 de la fonction $g\left(x_k\right)$ au voisinage de x_r est exprimé par :

$$g\left(x_k\right) = g\left(x_r\right) + \left(x_k - x_r\right) g'\left(x_r\right) + \frac{1}{2}\left(x_k - x_r\right)^2 g''\left(\xi\right), \text{ avec } x_k \le \xi \le x_r$$

or,

$$g'\left(x_r\right) = 0 \text{ et } e_k^2 = \left(x_k - x_r\right)^2$$

soit,

$$g\left(x_k\right) = g\left(x_r\right) + \frac{1}{2} e_k^2 g''\left(\xi\right) \Rightarrow g\left(x_k\right) - g\left(x_r\right) = \frac{1}{2} e_k^2 g''\left(\xi\right)$$

alors,

$$e_{k+1} = \left|x_{k+1} - x_r\right| = \left|g\left(x_k\right) - g\left(x_r\right)\right| = \frac{1}{2} e_k^2 \left|g''\left(\xi\right)\right| \le \frac{1}{2} \max_{\xi} \left|g''\left(\xi\right)\right| e_k^2$$

soit,

$$\mathrm{M} = \frac{1}{2}\left\|g''\right\|_\infty$$

alors,

$$e_{k+1} \le \mathrm{M} e_k^2$$

Cette inégalité montre bien que la convergence vers la solution x_r est quadratique. Le processus de Newton-Raphson converge rapidement vers la solution si le choix de la première itération soit plus proche de la solution, par exemple la solution obtenue par la méthode de bissection peut être utilisée comme une première itération du processus, car la méthode de bissection est moins rapide.

Le processus itératif de Newton-Raphson de l'équation précédente, s'écrit :

$$x_{k+1} = x_k - \frac{\left(5 - x_k\right)\exp\left(x_k\right) - 3}{\left(4 - x_k\right)\exp\left(x_k\right)}$$

Sachant que la première itération est x_0 = 6, soit le script MATLAB suivant qui génère le processus afin de résoudre l'équation :

```
f2x=sym('(5-x)*exp(x)-3');
df2x=diff(f2x); % Calcul de la dérivée
f2x=matlabFunction(f2x);
df2x=matlabFunction(df2x);
xintial= 6; X(1)=xintial;
xfinale=xintial-(f2x(xintial)/df2x(xintial));
k=0;
X(k+1)=xfinale;
eps=0.000001
while abs(xfinale-xintial)> eps
 k=k+1;
 xintial=xfinale;
 xfinale=xintial-(f2x(xintial)/df2x(xintial));
 X(k+1)=xfinale;
end
X=X';
disp ('Convergence atteinte dès l itération:'),k
disp ('La solution de l équation est :'),X(k)
```

Tab. 2.3 : Résultats.

k	x_k
0	6
1	5.4962819
2	5.1563808
3	5.0061981
4	4.9800730
5	4.9793652
6	4.9793647
7	4.9793647

L'exécution de ce script permet d'avoir (Tab. 3.3) :

```
Convergence atteinte dès l itération:
k =
     5
La solution de l équation est :
ans =
    4.9794
```

C'est-à-dire le processus converge vers la solution 4.9794 dès la sixième itération.

Extension vers la résolution d'un système d'équations non linéaires

La méthode de Newton-Raphson est un algorithme efficace pour trouver numériquement une approximation précise d'un zéro (ou racine) d'une fonction $f(x)=0$ d'une variable x. Cette méthode doit son nom aux mathématiciens anglais Isaac Newton (1643-1727) et Joseph Raphson (peut-être 1648-1715), qui furent les premiers à la décrire pour la recherche des zéros d'une équation polynomiale. Thomas Simpson (1710-1761) a élargi considérablement le domaine d'application de l'algorithme en montrant, grâce à la notion de dérivée, comment on pouvait l'utiliser pour calculer un zéro d'une équation non linéaire, pouvant ne pas être un polynôme, et d'un système formé de telles systèmes d'équations $\mathbf{f}(\mathbf{x})=\mathbf{0}$.

Le développement d'ordre deux en série de Taylor de la fonction $f_i(\mathbf{x})$ au voisinage du vecteur $\mathbf{x}$ est :

$$f_i(\mathbf{x}+\Delta\mathbf{x}) = f_i(\mathbf{x}) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\Delta x_j + O(\Delta x^2), \quad i = 1,\ 2,\ 3,\ \ldots,\ n$$

Sachant que $O(\Delta x^2) \approx 0$ quand $\Delta x \approx 0$, alors ce développement peut être explicité par le système :

$$f_1(\mathbf{x}+\Delta\mathbf{x}) = f_1(\mathbf{x}) + \sum_{j=1}^{n} \frac{\partial f_1}{\partial x_j}\Delta x_j$$

$$f_2(\mathbf{x}+\Delta\mathbf{x}) = f_2(\mathbf{x}) + \sum_{j=1}^{n} \frac{\partial f_2}{\partial x_j}\Delta x_j$$

$$\cdots\cdots\cdots\cdots\cdots\cdots$$

$$\cdots\cdots\cdots\cdots\cdots\cdots$$

$$f_n(\mathbf{x}+\Delta\mathbf{x}) = f_n(\mathbf{x}) + \sum_{j=1}^{n} \frac{\partial f_n}{\partial x_j}\Delta x_j$$

ou bien,

$$\mathbf{f}(\mathbf{x}+\Delta\mathbf{x}) = \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\Delta\mathbf{x}$$

avec, $\mathbf{J}(\mathbf{x})$ est la matrice jacobienne, dont les éléments sont :

$$J_{ij} = \frac{\partial f_i}{\partial x_j}$$

Si $\mathbf{x}+\Delta\mathbf{x}$ est la solution exacte de l'équation c'est-à-dire $\mathbf{f}(\mathbf{x}+\Delta\mathbf{x}) = \mathbf{0}$, alors $\mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\Delta\mathbf{x} = \mathbf{0}$ ce qui conduit :

$$\mathbf{J}(\mathbf{x})\Delta\mathbf{x} = -\mathbf{f}(\mathbf{x}) \;\Rightarrow\; \Delta\mathbf{x} = -\mathbf{J}^{-1}(\mathbf{x})\mathbf{f}(\mathbf{x})$$

soit le processus,

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{J}^{-1}\left(\mathbf{x}^{(k)}\right)\mathbf{f}\left(\mathbf{x}^{(k)}\right)$$

$\mathbf{J}^{-1}\left(\mathbf{x}^{(k)}\right)$ est la matrice jacobienne inverse.

Pratiquement ce processus consiste donc à déterminer numériquement une solution de l'équation vectorielle $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ suivant les étapes suivantes :

i. Choisir un vecteur de départ $\mathbf{x}^{(it)}$ qui constitue une première itération,
ii. Calculer le vecteur fonction $\mathbf{f}\left(\mathbf{x}^{(it)}\right)$ et la matrice jacobienne $\mathbf{J}\left(\mathbf{x}^{(it)}\right)$,
iii. Inverser la matrice jacobienne c'est-à-dire $\mathbf{J}^{-1}\left(\mathbf{x}^{(it)}\right)$,
iv. Effectuer la multiplication matricielle $\mathbf{J}^{-1}\left(\mathbf{x}^{(it)}\right)\mathbf{f}\left(\mathbf{x}^{(it)}\right)$,
v. Calculer la différence vectorielle $\mathbf{x}^{(it)} - \mathbf{J}^{-1}\left(\mathbf{x}^{(it)}\right)\mathbf{f}\left(\mathbf{x}^{(it)}\right)$ pour obtenir le vecteur $\mathbf{x}^{(it+1)}$,
vi. Voir si $\left\|\mathbf{x}^{(it+1)} - \mathbf{x}^{(it)}\right\| \le \varepsilon$, ou ε est une tolérance qui traduit le critère de convergence.

Si le critère de convergence est vrai, alors le processus converge vers la solution $\mathbf{x}^{(it+1)}$ sinon on refait le processus avec $\mathbf{x}^{(it+1)}$ pour déterminer $\mathbf{x}^{(it+2)}$ et ainsi de suite jusqu'à la satisfaction du critère de convergence.

Soit l'exemple suivant d'un système d'équations non linéaires :

$$\begin{cases} x_1^2 + x_2^2 - 1 = 0, \\ 2x_1 + x_2 - 1 = 0. \end{cases}$$

Dans ce système $f_1(x_1,\ x_2) = x_1^2 + x_2^2 - 1$ et $f_2(x_1,\ x_2) = 2x_1 + x_2 - 1$, par conséquent la matrice jacobienne :

$$\mathbf{J}(x_1,\ x_2) = \begin{pmatrix} 2x_1 & 2x_2 \\ 2 & 1 \end{pmatrix}$$

Le processus itératif de Newton–Raphson de ce système est explicité comme suit :

$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{pmatrix} = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \end{pmatrix} - \begin{pmatrix} 2x_1^{(k)} & 2x_2^{(k)} \\ 2 & 1 \end{pmatrix}^{-1} \begin{pmatrix} \left(x_1^{(k)}\right)^2 + \left(x_2^{(k)}\right)^2 - 1 \\ 2x_1^{(k)} + x_2^{(k)} - 1 \end{pmatrix}$$

Le script suivant permet de déterminer une solution de ce système pour un vecteur de départ $\mathbf{x}^{(1)} = \left(x_1^{(1)} = 1,\ x_2^{(1)} = 2\right)^T$.

```
% Définition de l'itération de départ
x(1)=1; x(2)=2; x=x';
for it=1:10000
 % Calcul du vecteur fonction
  f(1)=x(1)^2+x(2)^2-1; f(2)=2*x(1)+x(2)-1;
 % Calcul des composantes de la matrice jacobienne
   J(1,1)=2*x(1); J(1,2)=2*x(2); J(2,1)=2; J(2,2)=1;
 % Calcul du nouveau vecteur xPlus a l'itération it+1
   xPlus=x-inv(J)*f';
   x1(it)=x(1); x2(it)=x(2);
 % Utilisation d'un critère de convergence
   if abs(xPlus-x)<=0.00000001
      break
   else
      x=xPlus;
   end
end
```

Ce processus converge dès la sixième itération vers la solution $x_1 = x_1^{(6)} = 0,\ x_2 = x_2^{(6)} = 1$.

Remarquons que le choix du vecteur de départ est très important dans la convergence rapide du processus, si ce vecteur est loin de la solution exacte du système, le processus peut arriver à un niveau d'itération ou la matrice jacobienne devient instable ou singulière. Ces problèmes peuvent être réglé suivant la disposition des équations du système et sur le choix du vecteur de départ.

5. Méthode de la sécante

La méthode de la sécante est une méthode dérivée de celle de Newton-Raphson, elle est utilisée dans le cas où le calcul de la dérivée de la fonction $f(x)$ est délicat. La dérivée $f'(x_k)$ est donc remplacé dans le processus de Newton-Raphson par :

$$\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

Le processus itératif de la méthode de la sécante est exprimé par la relation de récurrence :

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k)$$

L'initialisation nécessite deux points x_0 et x_1, proches, si possible, de la solution recherchée.

Pour démontrer la formule précédente, soit x_0, x_1, la droite passant (Fig. 3.8) par les points de coordonnées $(x_0,\ f(x_0))$ et $(x_1,\ f(x_1))$ à pour équation :

$$y - f(x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_1)$$

Soit x_2 l'abscisse du point d'ordonnée $y = 0$ de cette droite :

$$0 - f(x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_2 - x_1)$$

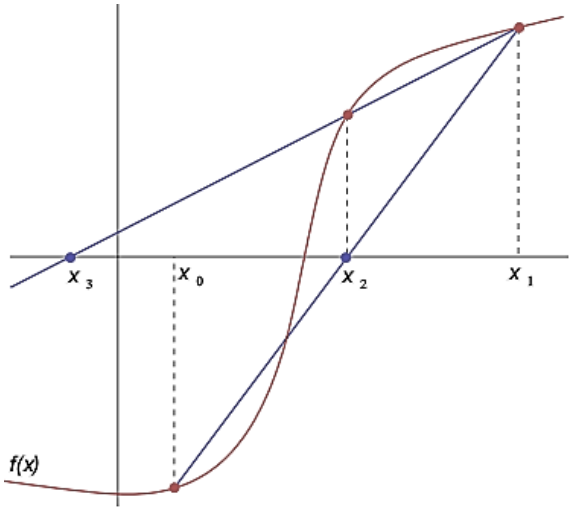


Fig. 3.8 : Méthode de la sécante.

soit,

$$x_2 = x_1 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_1)$$

de même,

$$x_3 = x_2 - \frac{x_2 - x_1}{f(x_2) - f(x_1)} f(x_2)$$

et ainsi de suite jusqu'à la convergence du processus.

La méthode de la sécante jouit d'une convergence d'ordre φ (nombre d'or). En d'autres termes,

$$|e_{k+1}| \le \mathrm{M} |e_k|^{\varphi} \quad \text{avec} \quad \varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$$

La preuve est extrêmement technique (Atkinson, 1989), de la formule de la méthode de la sécante soit :

$$x_r - x_{k+1} = x_r - x_k + \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k)$$

Le terme à droite de l'expression précédente peut être s'écrit en fonction des différences divisées de première et second ordre $f[x_{k-1},\ x_k]$ et $f[x_{k-1},\ x_k,\ x_r]$, c'est-à-dire :

$$-(x_r - x_k)(x_r - x_{k-1}) \frac{f[x_{k-1},\ x_k,\ x_r]}{f[x_{k-1},\ x_k]}$$

avec,

$$f[x_{k-1},\ x_k] = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \quad \text{et} \quad f[x_{k-1},\ x_k,\ x_r] = \frac{f[x_k,\ x_r] - f[x_{k-1},\ x_k]}{x_r - x_k}$$

D'après le théorème de la valeur moyenne, il existe ξ_k entre x_{k-1} et x_k et κ_k entre x_{k-1}, x_k et x_r de façon que :

$$f[x_{k-1},\ x_k]=f'(\xi_k)\ \text{ et }\ f[x_{k-1},\ x_k,\ x_r]=\frac{1}{2}f''(\kappa_k)$$

par conséquent,

$$x_r-x_{k+1}=-(x_r-x_k)(x_r-x_{k-1})\frac{f''(\kappa_k)}{2f'(\xi_k)}\ \Leftrightarrow\ e_{k+1}=-e_ke_{k-1}\frac{f''(\kappa_k)}{2f''(\xi_k)}$$

En valeur absolue,

$$|e_{k+1}|=\frac{1}{2}\left|\frac{f''(\kappa_k)}{f'(\xi_k)}e_ke_{k-1}\right|\approx \mathrm{M}|e_k||e_{k-1}|\ \Rightarrow\ \mathrm{M}|e_{k+1}|\approx \mathrm{M}|e_k|\mathrm{M}|e_{k-1}|\ \Rightarrow\ \log\mathrm{M}|e_{k+1}|\approx\log\mathrm{M}|e_k|+\log\mathrm{M}|e_{k-1}|$$

$$\text{Si }\ u_k=\log\mathrm{M}|e_k|\ \text{ alors, }\ u_{k+1}\approx u_k+u_{k-1}$$

Ceci est une relation de récurrence qui ressemble à la suite de Fibonacci. On peut montrer que le terme général de cette suite est exprimé par la formule de Binet (Weisstein, 2003) :

$$u_k=\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^k-\left(\frac{1-\sqrt{5}}{2}\right)^k\right]\equiv\frac{1}{\sqrt{5}}\left(\varphi^k-\varphi'^k\right)$$

par conséquent,

$$\log\mathrm{M}|e_k|\approx\frac{1}{\sqrt{5}}\varphi^k-\frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^k\approx\frac{1}{\sqrt{5}}\varphi^k$$

donc,

$$\log\mathrm{M}|e_{k+1}|\approx\frac{1}{\sqrt{5}}\varphi^{k+1}=\frac{1}{\sqrt{5}}\varphi^k\varphi\approx\varphi\log\mathrm{M}|e_k|=\log\left(\mathrm{M}|e_k|\right)^\varphi\ \Rightarrow\ \mathrm{M}|e_{k+1}|\approx\left(\mathrm{M}|e_k|\right)^\varphi$$

ou,

$$|e_{k+1}|\approx\mathrm{M}^{\varphi-1}|e_k|^\varphi\ \text{ ou bien }\ |e_{k+1}|\le\left(\frac{1}{2}\frac{\max\limits_x|f''(x)|}{\min\limits_x|f'(x)|}\right)^{\varphi-1}|e_k|^\varphi=\mathrm{M}|e_k|^\varphi$$

Le processus itératif de la méthode de la sécante de l'équation $f(x)=(5-x)e^x-3=0$, s'écrit :

$$x_{k+1}=x_k-\frac{(x_k-x_{k-1})\left[(5-x_k)e^{x_k}-3\right]}{(5-x_k)e^{x_k}-(5-x_{k-1})e^{x_{k-1}}}$$

Sachant que les deux premières itérations sont x_0 = 6.5 et x_1 = 6. Soit la fonction **Secante** qui génère le processus de la méthode de la sécante :

```
function [x,k] = Secante(f,xint0,xint1,varargin)
% f, nom de la fonction tel que f(x)=0
% xint0,xint1, les deux valeurs initiales
% x, solution de l'équation f(x)=0
% k, nombre d'itérations jusqu'à la convergence
if nargin<3,error('à la limite 3 arguments sont nécessaires'),end
eps=0.000001; % Définition de la précision souhaitée
k=0;
X(k+1)=xint1; x=xint1-((xint1-xint0)/(f(xint1)-f(xint0)))*f(xint1);
X(k+2)=x;
```

```
while abs(x-xint1)>eps
 k=k+1; xint0=xint1; xint1=x;
 x=xint1-((xint1-xint0)/(f(xint1)-f(xint0)))*f(xint1); X(k)=x;
end
k=k-1;
end
```

Pour l'équation $f(x)=(5-x)e^{x}-3=0$ avec les deux itérations initiales $x_0 = 6.5$ et $x_1 = 6$:

```
[x,k] = Secante(@(x)(5-x).*exp(x)-3,6.5,6)
x =
    4.9794
k =
     7
```

Le processus converge vers la solution 4.9794 dès la 7ième itération pour une tolérance de 10^{-6}.

6. Méthode Regula-Falsi

La méthode Regula-Falsi dite aussi de la fausse position est un algorithme de recherche d'un zéro d'une fonction, qui combine les possibilités de la méthode de la bissection et de la méthode de la sécante.

Comme dans le cas de la bissection, si deux points a et b tels que $f(a).f(b)<0$, la solution de l'équation $f(x)=0$ existe entre a et b et x_1 est calculé par la méthode de la sécante.

$$x_1 = \frac{af(b)-bf(a)}{f(b)-f(a)}$$

Il y a deux possibilités soit, $f(a).f(x_1)<0$ ou $f(x_1).f(b)<0$.

Dans le cas où $f(a).f(x_1)<0$, la solution de l'équation $f(x)=0$ existe entre a et x_1 et x_2 est calculé comme suit :

$$x_2 = \frac{af(x_1)-x_1f(a)}{f(x_1)-f(a)}$$

Sinon $f(x_1).f(b)<0$, la solution de l'équation $f(x)=0$ existe entre x_1 et b, x_2 est calculé comme suit :

$$x_2 = \frac{x_1f(b)-bf(x_1)}{f(b)-f(x_1)}$$

Le processus est répété jusqu'à la convergence. La vitesse de convergence de la méthode Regula-Falsi n'est pas super linéaire comme dans le cas de la sécante. Pour le comprendre, il suffit de voir que dans le cas d'une fonction dont la convexité ne change pas au voisinage de la racine, l'itéré obtenu a toujours le même signe. Dès lors, en supposant sans perte de généralité, que $f(x_2)$ à le même signe que $f(x_1)$, x_0

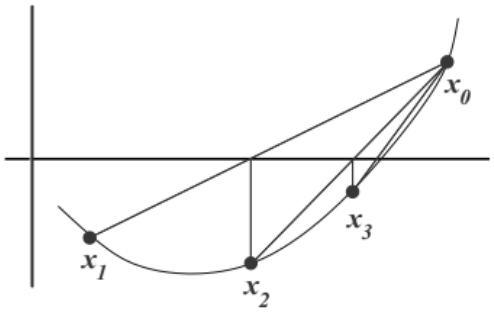


Fig. 3.9 : Convexité dans la méthode Regula-Falsi.

ne sera jamais remplacé dans le courant de l'algorithme. L'erreur suivra, asymptotiquement la formule :

$$|e_{k+1}| \approx \mathrm{M}|e_k||e_{k-1}| = \mathrm{M}|e_k||e_0|$$

Puisque, dans le cas de la Regula-Falsi, le point x_{k-1} ne change en réalité pas et reste x_0 durant tout l'algorithme. L'équation, $|e_{k+1}| \approx \mathrm{M}|e_0||e_k|$ exprime la convergence linéaire de la méthode (Fig. 3.9).

La fonction **RegulaFalsi** utilise le processus de Regula-Falsi pour chercher la solution de l'équation $f(x)=0$ dans l'intervalle [*a*, *b*].

```
function [x,k] = RegulaFalsi(f,a,b,varargin)
% f, nom de la fonction tel que f(x)=0
% [a, b], intervalle de la solution de f(x)=0 (a < b)
% x, solution de l'équation f(x)=0
% k, nombre d'opérations jusqu'à la convergence
if nargin<3,error('à la limite 3 arguments sont nécessaires'),end
if ~(b>a),error('Attention, il faut que a < b'),end
eps=0.000001; % Définition de la précision souhaitée
k=1; x=(a*f(b)-b*f(a))/(f(b)-f(a));
 while abs(f(x))> eps
  if f(a)*f(x)<0, b=x; else a=x; end
  k=k+1; x=(a*f(b)-b*f(a))/(f(b)-f(a));
 end
end
```

A titre d'exemple l'application de cette fonction à l'équation $f(x)=(5-x)e^x-3=0$ dans l'intervalle [4, 6] conduit à :

```
>> [x,k] = RegulaFalsi(@(x) (5-x).*exp(x)-3,4,6)
x =
    4.9794
k =
    44
```

Dans cet exemple, la méthode de Regula-Falsi converge vers la solution 4.9794 après 50 opérations pour une tolérance imposée de 10^{-6}.

7. MATLAB et la résolution des équations non linéaires

La fonction **fzero** de MATLAB qui est basée sur la méthode de Van Wijngaarden-Dekker-Brent (Forsythe *et al.*, 1977 et Brent, 1973) (voir chapitre 4 section 4.1) qui combine la méthode de la bissection ou la méthode de la sécante avec l'interpolation quadratique inverse pour l'amélioration de l'itération. Cette méthode est très utile pour trouver le zéro d'une fonction $f(x)$ sans calculer les dérivées de la fonction $f(x)$. L'application de la fonction **fzero**, nécessite deux arguments : la fonction $f(x)$ et la première itération x_0 qui doit être proche de la solution. Soit l'exemple de la fonction précédente.

```
>> f=@(x) (5-x).*exp(x)-3;
>> Racine1=fzero(f,6)
Racine1 =
```

```
      4.9794
>> Racine2=fzero(f,1)
Racine2 =
   -0.6294
```

Remarquons ici que la fonction $f(x)=(5-x)e^{x}-3$ possède deux racines, 4.9794 et -0.6294 qui sont obtenues suivant le choix de la première itération 6 et 1. Avec le script du processus de Newton-Raphson et la fonction **Secante**, la racine -0.6294 est obtenue suivant le choix de la première itération **xintial = xint0 = 1**, les résultats d'exécution :

Suivant la méthode de Newton

```
Convergence atteinte dès l itération:
k =
     5
La solution de l équation est :
ans =
   -0.6294
```

Suivant la méthode de la sécante

```
>> f=@(x)(5-x).*exp(x)-3
>> [x,k] = Secante(f,1,6)
x =
   -0.6294
k =
     7
```

8. Exercices résolus

Exercice 1

Le but cet exercice est l'établissement d'un processus qui sert à déterminer la racine carrée d'un nombre entier positif λ en utilisant les processus du point fixe et celle de Newton-Raphson. Pour cela soit l'équation :

$$x^2=\lambda,\ \ x,\ \ \lambda\geq 0$$

Qui possède la solution $x=\sqrt{\lambda}$.

On a l'équation : $x^2=\lambda$ avec $\lambda\geq 0$, peut être transformée ainsi :

$$x^2=\lambda \ \Rightarrow\ x^2+x^2=x^2+\lambda \ \Rightarrow\ 2x^2=x^2+\lambda \ \Rightarrow\ x=\frac{1}{2}\left(x+\frac{\lambda}{x}\right)$$

Soit le processus itératif :

$$x_{k+1}=\frac{1}{2}\left(x_k+\frac{\lambda}{x_k}\right),\ \ x_1=\lambda$$

De l'équation $x^2=\lambda \ \Rightarrow\ x^2-\lambda=f(x)=0 \ \Rightarrow\ f'(x)=2x$. Soit le processus de Newton-Raphson :

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)} \ \Rightarrow\ x_{k+1}=x_k-\frac{x_k^2-\lambda}{2x_k} \ \Rightarrow\ x_{k+1}=\frac{1}{2}\left(x_k+\frac{\lambda}{x_k}\right)$$

C'est le même processus itératif qui sert finalement à déterminer $\sqrt{\lambda}$.

Soit l'exemple $\lambda=2$, alors,

$$x_{k+1}=\frac{1}{2}\left(x_k+\frac{2}{x_k}\right),\ \text{ et }\ x_1=2$$

soit,

$$x_2 = \frac{1}{2}\left(2+\frac{2}{2}\right) = \frac{3}{2} = 1.5, \quad x_3 = \frac{1}{2}\left(1.5+\frac{2}{1.5}\right) = 1.4167$$

$$x_4 = \frac{1}{2}\left(1.4167+\frac{2}{1.4167}\right) = 1.4142, \quad x_5 = \frac{1}{2}\left(1.4142+\frac{2}{1.4142}\right) = 1.4142$$

Le processus converge rapidement dès la quatrième itération vers la solution $\sqrt{2} \approx 1.4142$.

Exercice 2

Le coefficient λ de perte de charge (ou de frottement) en écoulement turbulent dans une conduite de diamètre *D* et de rugosité *ε*, est exprimé en fonction du nombre de Reynolds Re et de la rugosité relative ε/D par l'équation de Colebrook :

$$\frac{1}{\sqrt{\lambda}} = -2\log_{10}\left(\frac{\varepsilon}{3.7D}+\frac{2.51}{\mathrm{Re}\sqrt{\lambda}}\right)$$

Sachant que $\varepsilon/D = 0.03$ et $\mathrm{Re} = 10000$

1. Déterminer la valeur du coefficient λ en utilisant la méthode de point fixe.
2. Retrouver la valeur de λ calculé précédemment en utilisant la méthode de Newton-Raphson.
3. Vérifier les résultats obtenus par les deux méthodes avec celle obtenu par l'utilisation de la fonction **`fzero`** de MATLAB.

1. De la formule de Colebrook, soit le processus itératif du point fixe suivant :

$$\frac{1}{\sqrt{\lambda_{k+1}}} = -2\log_{10}\left(\frac{\varepsilon}{3.7D}+\frac{2.51}{\mathrm{Re}\sqrt{\lambda_k}}\right), \qquad \lambda_1 = 1$$

Soit l'affectation suivante :

$$x = \frac{1}{\sqrt{\lambda}} \;\Rightarrow\; x_{k+1} = -2\log_{10}\left(\frac{\varepsilon}{3.7D}+\frac{2.51}{\mathrm{Re}}x_k\right), \quad x_1 = 1$$

par conséquent,

$$x_2 = -2\log_{10}\left(\frac{\varepsilon}{3.7D}+\frac{2.51}{\mathrm{Re}}x_1\right) = -2\log_{10}\left(\frac{0.03}{3.7}+\frac{2.51}{10000}\times 1\right) = 4.1557$$

$$x_3 = -2\log_{10}\left(\frac{\varepsilon}{3.7D}+\frac{2.51}{\mathrm{Re}}x_2\right) = -2\log_{10}\left(\frac{0.03}{3.7}+\frac{2.51}{10000}\times 4.1557\right) = 4.0770$$

$$x_4 = -2\log_{10}\left(\frac{\varepsilon}{3.7D}+\frac{2.51}{\mathrm{Re}}x_3\right) = -2\log_{10}\left(\frac{0.03}{3.7}+\frac{2.51}{10000}\times 4.0770\right) = 4.0789$$

$$x_5 = -2\log_{10}\left(\frac{\varepsilon}{3.7D}+\frac{2.51}{\mathrm{Re}}x_4\right) = -2\log_{10}\left(\frac{0.03}{3.7}+\frac{2.51}{10000}\times 4.0789\right) = 4.0789$$

D'après ces calculs, le processus de la méthode du point fixe converge dès la 4ième itération vers la valeur $x = 4.0789$, alors le coefficient de perte de charge est $\lambda = 1/x^2 = 0.0601$.

2. La formule de Colebrook peut être s'écrit sous la forme suivante :

$$\frac{1}{\sqrt{\lambda}} = -2\log_{10}\left(\frac{\varepsilon}{3.7D} + \frac{2.51}{\mathrm{Re}\sqrt{\lambda}}\right) \;\Rightarrow\; \frac{1}{\sqrt{\lambda}} + 2\log_{10}\left(\frac{\varepsilon}{3.7D} + \frac{2.51}{\mathrm{Re}\sqrt{\lambda}}\right) = 0$$

De même, on pose $x = 1/\sqrt{\lambda}$ ce qui conduit à une équation de la forme $f(x) = 0$ avec :

$$f(x) = x + 2\log_{10}\left(\frac{\varepsilon}{3.7D} + \frac{2.51}{\mathrm{Re}}x\right)$$

Le processus de Newton–Raphson pour cette équation est :

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

or,
$$f'(x) = 1 + \frac{2}{\ln(10)}\frac{2.51}{\left(\frac{\varepsilon}{3.7D}\mathrm{Re} + 2.51x\right)}$$

alors le processus itératif de Newton–Raphson s'écrit :

$$x_{k+1} = x_k - \frac{x_k + 2\log_{10}\left(\frac{\varepsilon}{3.7D} + \frac{2.51}{\mathrm{Re}}x_k\right)}{1 + \frac{2}{\ln(10)}\frac{2.51}{\left(\frac{\varepsilon}{3.7D}\mathrm{Re} + 2.51x_k\right)}}$$

Pour une première itération $x_1 = 1$, soit donc :

$$x_2 = x_1 - \frac{x_1 + 2\log_{10}\left(\frac{\varepsilon}{3.7D} + \frac{2.51}{\mathrm{Re}}x_1\right)}{1 + \frac{2}{\ln(10)}\frac{2.51}{\left(\frac{\varepsilon}{3.7D}\mathrm{Re} + 2.51x_1\right)}} = 1 - \frac{1 + 2\log_{10}\left(\frac{0.03}{3.7} + \frac{2.51}{10^4}\times 1\right)}{1 + \frac{2}{\ln(10)}\frac{2.51}{\left(\frac{0.03}{3.7}10^4 + 2.51\times 1\right)}} = 4.0755$$

$$x_3 = x_2 - \frac{x_2 + 2\log_{10}\left(\frac{\varepsilon}{3.7D} + \frac{2.51}{\mathrm{Re}}x_2\right)}{1 + \frac{2}{\ln(10)}\frac{2.51}{\left(\frac{\varepsilon}{3.7D}\mathrm{Re} + 2.51x_2\right)}} = 4.0755 - \frac{4.0755 + 2\log_{10}\left(\frac{0.03}{3.7} + \frac{2.51}{10^4}\times 4.0755\right)}{1 + \frac{2}{\ln(10)}\frac{2.51}{\left(\frac{0.03}{3.7}10^4 + 2.51\times 4.0755\right)}} = 4.0789$$

$$x_4 = x_3 - \frac{x_3 + 2\log_{10}\left(\frac{\varepsilon}{3.7D} + \frac{2.51}{\mathrm{Re}}x_3\right)}{1 + \frac{2}{\ln(10)}\frac{2.51}{\left(\frac{\varepsilon}{3.7D}\mathrm{Re} + 2.51x_3\right)}} = 4.0789 - \frac{4.0789 + 2\log_{10}\left(\frac{0.03}{3.7} + \frac{2.51}{10^4}\times 4.0789\right)}{1 + \frac{2}{\ln(10)}\frac{2.51}{\left(\frac{0.03}{3.7}10^4 + 2.51\times 4.0789\right)}} = 4.0789$$

Le processus itératif de Newton-Raphson converge dès la 3ième itération vers la même valeur à celle du processus du point fixe $x = 4.0789$, alors $\lambda = 0.0601$.

3. Avec MATLAB, la fonction **`fzero`** peut être utilisée pour déterminer une solution de l'équation $f(x) = 0$ comme suit :

```
>> f2X=@(X) X.+2*log10((0.03/3.7)+(2.51/10000)*X);
>> x=fzero(f2X,1)
```

```
x =
      4.0789
>> Lamda=1./(x.^2)
Lamda =
    0.0601
```

Dans le diagramme de Moody (Fig. 3.10) et pour la rugosité relative $\varepsilon/D=0.03$ un nombre de Reynolds égale à 10000 le coefficient est de λ est à 0.061 qui est pratiquement la même à celle obtenue par les deux méthodes numériques.

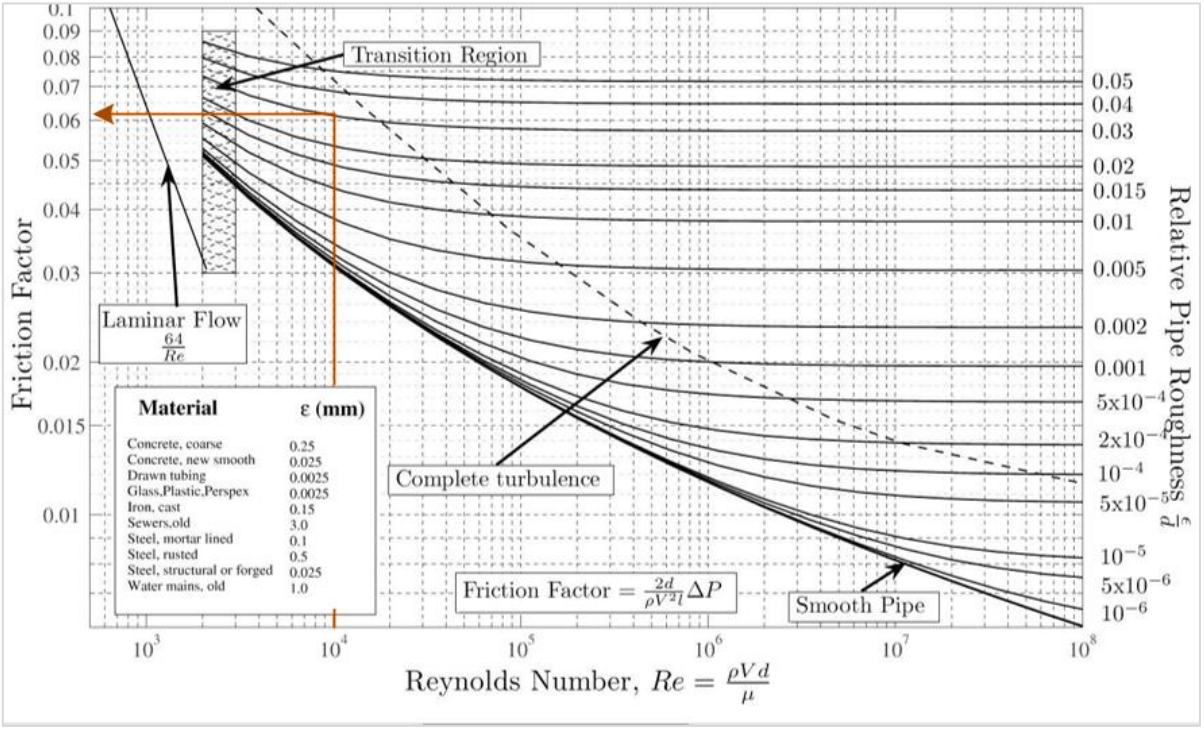


Fig. 3.10 : Diagramme de Moody.

Exercice 3

Soit le système d'équations non linéaires suivant :

$$\begin{cases} y^2 - x^2 = 1, \\ y + x = 4. \end{cases}$$

1. Montrer que ce système peut être se transformer sous la forme de $\sqrt{x^2+1} = -x+4$ et donner dans ce cas le schéma de résolution suivant le processus de Newton-Raphson de cette équation.

2. Soit $x_1 = 2$ la première itération, déterminer dans ce cas la solution de l'équation $\sqrt{x^2+1} = -x+4$ et par conséquent la solution du système d'équation.

3. Ecrire un script MATLAB qui permet de résoudre numériquement ce problème.

1. A partir de la deuxième équation du système $y = -x+4$ et par substitution dans la première équation du système :

$$y + x = 4 \;\Rightarrow\; y = -x+4$$

$$y^2 - x^2 = 1 \;\Rightarrow\; (-x+4)^2 - x^2 = 1 \;\Rightarrow\; (-x+4)^2 = x^2+1 \;\Rightarrow\; \sqrt{x^2+1} = -x+4$$

De l'équation, $\sqrt{x^2+1}=-x+4 \Rightarrow \sqrt{x^2+1}+x-4=0\equiv f(x)$. Le processus de Newton-Raphson de l'équation $f(x)=0$ est :

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)} \Rightarrow x_{k+1}=x_k-\frac{\sqrt{x_k^2+1}+x_k-4}{1+x_k/\sqrt{x_k^2+1}}$$

2. Pour une première itération $x_1=2$ soit,

$$x_2=x_1-\frac{\sqrt{x_1^2+1}+x_1-4}{1+x_1/\sqrt{x_1^2+1}}=2-\frac{\sqrt{2^2+1}+2-4}{1+2/\sqrt{2^2+1}}=1.8754$$

$$x_3=x_2-\frac{\sqrt{x_2^2+1}+x_2-4}{1+x_2/\sqrt{x_2^2+1}}=1.8754-\frac{\sqrt{1.8754^2+1}+1.8754-4}{1+1.8754/\sqrt{1.8754^2+1}}=1.8750$$

$$x_4=x_3-\frac{\sqrt{x_3^2+1}+x_3-4}{1+x_3/\sqrt{x_3^2+1}}=1.8750-\frac{\sqrt{1.8750^2+1}+1.8750-4}{1+1.8750/\sqrt{1.8750^2+1}}=1.8750$$

Le processus converge dès la troisième itération vers la solution $x=1.8750$, par conséquent

$$y=-1.8750+4=2.1250$$

Donc la solution du système d'équations est : $(x,\ y)=(1.875,\ 2.125)$.

3. Soit le script MATLAB qui permet une détermination d'une solution du système d'équations non linéaires :

```
% Définition de la fonction f(x) comme une fonction symbolique
f2x=sym('sqrt(x^2+1)+x-4'); % Détermination de la fonction dérivée f'(x)
fprime2x=diff(f2x);
% Conversion des fonctions f(x) et f'(x) vers des fonctions exécutables
f2x=matlabFunction(f2x); fprime2x=matlabFunction(fprime2x);
xinit=2; % Définition de la première itération
X(1)=xinit; Y(1)=-X(1)+4;
for k=1:100
  % Utilisation du processus de Newton-Raphson
  xPlus=xinit-(f2x(xinit)/fprime2x(xinit));
  X(k+1)=xPlus; Y(k+1)=-X(k+1)+4;
  % Emploi d'un critère de convergence
  if abs(xPlus-xinit)==0
      break
  else
     xinit=xPlus;
  end
end
disp ('Convergence atteinte dès l itération:'),k
disp ('La solution x :'), X(k)
disp ('La solution y :'), Y(k)
```

L'exécution du script conduit à :

```
Convergence atteinte dès l itération:
k =
```

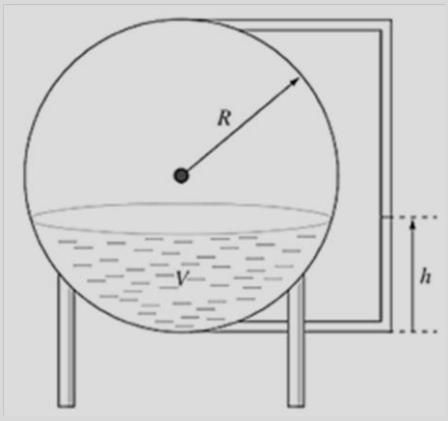
```
     4
La solution x :
ans =
    1.8750
La solution y :
ans =
    2.1250
```

Exercice 4

Le volume d'eau dans un réservoir de forme sphérique de rayon R est exprimé par la formule :

$$V = \frac{\pi}{3} h^2 (3R - h)$$

ou, h est la hauteur d'eau dans le réservoir. Si R = 3 m et le volume d'eau dans le réservoir est V = 30 m^3, utiliser une des méthodes des approximations successives pour déterminer la hauteur h dans le réservoir et évaluer l'erreur commisse en termes de volume au moins pour les trois premières itérations.



De l'équation,

$$V = \frac{\pi}{3} h^2 (3R - h) \Rightarrow \frac{3}{\pi} V - h^2 (3R - h) \equiv f(h) = 0$$

alors,

$$f'(h) = 3h(h - 2R)$$

Soit le processus de Newton-Raphson :

$$h_{i+1} = h_i - \frac{f(h_i)}{f'(h_i)} \Leftrightarrow h_{i+1} = h_i - \frac{\frac{3}{\pi} V - (3R - h_i) h_i^2}{3(h_i - 2R) h_i}$$

Sachant que V = 30 m^3 et R = 3 m, alors :

$$h_{i+1} = h_i - \frac{\frac{90}{\pi} - (9 - h_i) h_i^2}{3(h_i - 6) h_i}$$

Pour déterminer le premier itéré, la moitié du volume de la sphère est :

$$V' = \frac{2\pi}{3} R^3 = \frac{2\pi}{3} 3^3 = 56.5\,\text{m}^3, \text{ c'est-à-dire, } V' > V \Rightarrow h < 3\,\text{m}$$

Soit $h_1 = 1$, alors,

$$h_2 = 1 - \frac{\frac{90}{\pi} - (9 - 1) 1^2}{3(1 - 6) 1} = 2.37653\ \text{m} \Rightarrow V_2 = 39.17\ \text{m}^3 \Rightarrow \Delta V_2 = 9.17\ \text{m}^3$$

$$h_3 = 2.37653 - \frac{\frac{90}{\pi} - (9 - 2.37653) 2.37653^2}{3(2.37653 - 6) 2.37653} = 2.03741\ \text{m} \Rightarrow V_3 = 30.27\ \text{m}^3 \Rightarrow \Delta V_3 = 0.27\ \text{m}^3$$

$$h_4 = 2.03741 - \frac{\frac{90}{\pi} - (9 - 2.03741)2.03741^2}{3(2.03741 - 6)2.03741} = 2.02692 \text{ m} \Rightarrow V_4 = 30.00 \text{ m}^3 \Rightarrow \Delta V_4 = 0.00 \text{ m}^3$$

$$h_5 = 2.02692 - \frac{\frac{90}{\pi} - (9 - 2.02692)2.02692^2}{3(2.02692 - 6)2.02692} = 2.02691 \text{ m} \Rightarrow V_5 = 30 \text{ m}^3 \Rightarrow \Delta V_5 = 0.00 \text{ m}^3$$

$$h_6 = 2.02691 - \frac{\frac{90}{\pi} - (9 - 2.02691)2.02691^2}{3(2.02691 - 6)2.02691} = 2.02691 \text{ m} \Rightarrow V_6 = 30 \text{ m}^3 \Rightarrow \Delta V_6 = 0.00 \text{ m}^3$$

Le processus converge vers la profondeur $h \approx 2$ m pour qui correspond au volume V = 30 m³.

Exercice 5

Soit l'équation non linéaire :

$$f(x) = 3x^2\left(\frac{9-x^2}{x}\right)^{1/3} - \frac{9-x^2}{x} - 4 = 0$$

En utilisant le processus de Newton-Raphson, observez pour quelle solution converge le processus si la première itération est 1.2, -2, -1.2 et 2.

La fonction $f(x)$ est définie et continue quel que soit $x \neq 0$, sa dérivée est explicitée ainsi par :

$$f'(x) = 6x\left(\frac{9-x^2}{x}\right)^{1/3} + 3\left(3-x^2\right)\left(\frac{9-x^2}{x}\right)^{-2/3} - \frac{3\left(3-x^2\right)}{x^2}$$

Soit le script MATLAB suivant qui permet de déterminer une solution numérique de l'équation $f(x)$ pour les initialisations 1.2, -2, -1.2 et 2

```
% Préparation de la fonction f(x) en forme symbolique
f2x=sym('3*(x^2)*(((9-x^2)/x)^(1/3))-((9-x^2)/x)-4');
df2x=diff(f2x); % Calcul symbolique de la dérivée f'(x)
% Conversion des fonction f(x) et f'(x) en fonction MATLAB
f2x=matlabFunction(f2x); df2x=matlabFunction(df2x);
xintial= 1.2; X(1)=xintial; % xinitial est la première itération
xfinale=xintial-(f2x(xintial)/df2x(xintial)); k=1; X(k+1)=xfinale;
while abs(xfinale-xintial)>0.00000001
  k=k+1; xintial=xfinale;
  xfinale=xintial-(f2x(xintial)/df2x(xintial)); X(k+1)=xfinale;
end
X=X';
disp ('Convergence atteinte à :');k
disp ('Solution de f(x)=0 est');X(k)
disp ('f(Solution)= ');f2x(X(k))
```

Les résultats obtenus après l'exécution de ce script pour une première itération différentes $x_1 = 2,\ 1.2,\ -2,\ -1.2$ et 2.

```
xintial= 1.2
Convergence atteinte à :
k =
```

```
xintial= -2
Convergence atteinte à :
k =
```

```
     4
Solution de f(x)=0 est
ans =
    1.3364
f(Solution)=
ans =
  -1.4690e-11

xintial= -1.2
Convergence atteinte à :
k =
     6
Solution de f(x)=0 est
ans =
  -0.8288 + 0.5627i
f(Solution)=
ans =
  -1.0658e-14 + 5.3291e-15i
```

```
     7
Solution de f(x)=0 est
ans =
  -0.8288 + 0.5627i
f(Solution)=
ans =
   1.7764e-15 + 1.9718e-13i

xintial= 2
Convergence atteinte à :
k =
     5
Solution de f(x)=0 est
ans =
    1.3364
f(Solution)=
ans =
  -5.1381e-12
```

D'après ces résultats, si $x_1 = 1.2$ et 2, le processus itératif de Newton-Raphson converge vers la solution 1.3364 et si $x_1 = -1.2$ et -2 le processus itératif de Newton-Raphson converge vers la solution complexe -0.8288+0.5627*i*.

Exercice 6

Dans un écoulement permanent à surface libre le long d'un canal de section trapézoïdale véhicule un débit $Q = 20\ \text{m}^3.\text{s}^{-1}$. A l'état critique :

$$\frac{Q^2 B}{g A_c^3} = 1$$

Où, $g = 9.81\ \text{m/s}^2$ l'accélération de la pesanteur,

B, la largeur du canal (m), est exprimée ici par : $B = 3 + y$

A_c, surface de la section transversale du canal (m²), $A_c = 3y + y^2/2$.

Déterminer en utilisant la programmation sous MATLAB, la hauteur critique y par les trois méthodes : bissection, point fixe et Newton-Raphson.

Un écoulement à surface libre en régime critique :

$$\frac{Q^2 B}{g A_c^3} = 1 \ \Rightarrow \ A_c^3 = \frac{Q^2}{g} B$$

soit,

$$\left(3y + \frac{y^2}{2}\right)^3 = \frac{Q^2}{g}(3 + y)$$

ou,

$$\left(3y + \frac{y^2}{2}\right)^3 - \frac{Q^2}{g}(3 + y) \equiv f(y) = 0$$

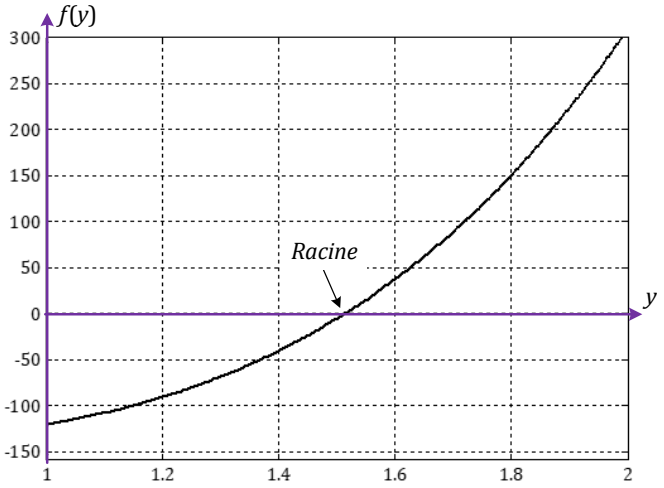


Fig. 3.11 : Tracé de la fonction $f(y)$.

La profondeur y de l'écoulement est positif, soit la séquence de commandes qui sert à tracer la fonction $f(y)$ dans l'intervalle [0, 2] (Fig. 3.11) :

```
>> f=sym('((3*y+0.5*y*y)^3)-(20*20/9.81)*(3+y)');
>> grid on
>> ezplot(f,[0,2])
```

D'après le tracé de la fonction $f(y)$ (Fig. 3.11), il y a une intersection entre la courbe de la et l'axe des abscisses dans l'intervalle [1.4, 1.6]. Soit le script de la méthode de bissection qui permet de déterminer l'abscisse de ce point d'intersection.

```
syms y
f=sym('((3*y+0.5*y*y)^3)-(20*20/9.81)*(3+y)')
f=matlabFunction(f);
a=1.4; b=1.6;
eps=0.0001;
for k=1:100
   y=(a+b)/2;
  if abs(f(y))<=eps
    break
  elseif f(a)*f(y)<0
    b=y;
  else
    a=y;
  end
end
disp ('Nombre de réduction de l intervalle :'),k
disp ('La profondeur critique est :'),y
```

L'exécution de ce script permet d'avoir le résultat :

```
Nombre de réduction de l intervalle :
k =
    17
La profondeur critique est :
y =
    1.5141
```

Avec la méthode du point fixe l'équation $f(y)=0$ est transformée en $y=g(y)$ comme :

$$\left(3y+\frac{y^2}{2}\right)^3-\frac{Q^2}{g}(3+y)=0 \Rightarrow y^3\left(\frac{6+y}{2}\right)^3=\frac{Q^2}{g}(3+y) \Rightarrow y=\frac{2}{6+y}\left[\frac{Q^2}{g}(3+y)\right]^{1/3}\equiv g(y)$$

La dérivée de la fonction $g(y)$ est :

$$g'(y)=-\frac{2}{3}\left[\frac{Q^2}{g}(3+y)\right]^{1/3}\frac{3+2y}{(3+y)(6+y)^2}$$

La séquence des commandes suivantes consiste à l'analyse de la variation de la fonction $|g'(y)|$ dans l'intervalle [0, 2] (Fig. 3.12) :

```
>> dg=@(y)-
((2/3)*(((20^2/9.81)*(3+y)).^(1/3)).*(3+2*y))./((3+y).*((6+y).^2));
>> y=0:0.01:2;
```

```
>> Dy=abs(dg(y));
>> plot(y,Dy)
```

$|g'(y)|<1,\ \forall\ 0\le y\le 2$, donc le processus itératif du point fixe suivant est convergent :

$$y_{k+1}=\frac{2}{6+y_k}\left[\frac{Q^2}{g}(3+y_k)\right]^{1/3}$$

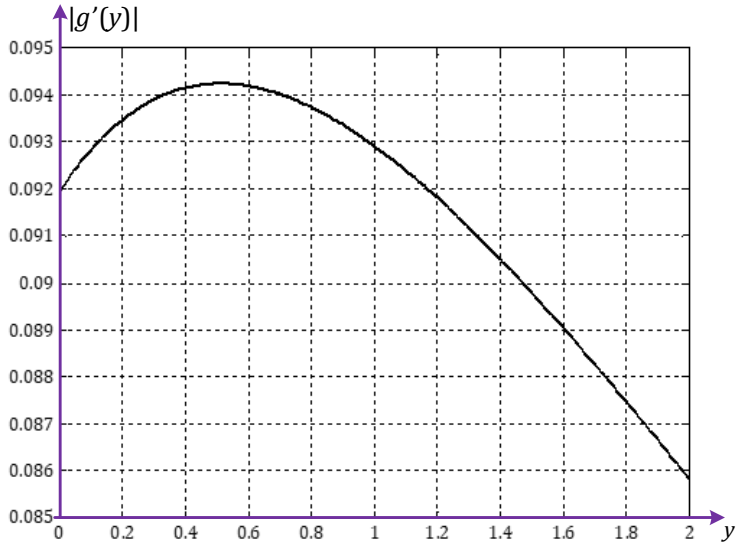


Fig. 3.12 : Variation de $|g'(y)|$ dans l'intervalle de solution.

Soit alors, le script :

```
g=@(y) 2*(((20*20/9.81)*(3+y)).^(1/3))./(6+y);
yintial= 0;
y=g(yintial);
iteration=1;
while abs(y-yintial)>0.00001
 iteration=iteration+1; yintial=y; y=g(yintial);
end
disp ('Convergence atteinte à partir de'),iteration
disp ('La profondeur critique est :'),y
```

Qui permet d'avoir comme résultat :

```
Convergence atteinte à partir de
iteration =
     6
La profondeur critique est :
y =
    1.5141
```

Avec la méthode de Newton-Raphson, la première des choses et de déterminer la dérivée de la fonction $f(y)$, c'est-à-dire :

$$f'(y)=3(3+y)\left(3y+\frac{y^2}{2}\right)^2-\frac{Q^2}{g}$$

Soit le processus itératif :

$$y_{k+1} = y_k - \frac{\left(3y_k + \frac{y_k^2}{2}\right)^3 - \frac{Q^2}{g}(3+y_k)}{3(3+y_k)\left(3y_k + \frac{y_k^2}{2}\right)^2 - \frac{Q^2}{g}}$$

Soit alors, le script :

```
f2y=sym('(3*y+0.5*(y^2))^3-(20*20/9.81)*(3+y)');
df2y=diff(f2y); f2y=matlabFunction(f2y); df2y=matlabFunction(df2y);
yintial=1; y=yintial-(f2y(yintial)/df2y(yintial)); iteration=1;
while abs(y-yintial)>0.00001
  iteration = iteration+1; yintial = y;
  y = yintial-(f2y(yintial)/df2y(yintial));
end
disp ('Convergence atteinte à partir de');iteration
disp ('La profondeur critique est :');y
```

Qui permet d'avoir comme résultat :

```
Convergence atteinte à partir de
iteration =
     6
La profondeur critique est :
y =
    1.5141
```

Exercice 7

Construire un script MATLAB afin de résoudre par la méthode de Newton–Raphson le système d'équations non linéaires suivant :

$$\begin{aligned} \sin x + y^2 + \ln z - 7 &= 0 \\ 2x + 2^y - z^3 + 1 &= 0 \\ x + y + z - 5 &= 0 \end{aligned}$$ avec comme première itération $\begin{pmatrix} x_1 = 1 \\ y_1 = 1 \\ z_1 = 1 \end{pmatrix}$

Dans ce système,

$f_1(x,\ y,\ z) = \sin x + y^2 + \ln z - 7$, $f_2(x,\ y,\ z) = 2x + 2^y - z^3 + 1$ et $f_3(x,\ y,\ z) = x + y + z - 5$

Ce qui permet de former le vecteur fonction et la matrice jacobienne :

$$\mathbf{f}(x,\ y,\ z) = \begin{pmatrix} \sin x + y^2 + \ln z - 7 \\ 2x + 2^y - z^3 + 1 \\ x + y + z - 5 \end{pmatrix},\ \mathbf{J}(x,\ y,\ z) = \begin{pmatrix} \cos x & 2y & \frac{1}{z} \\ 2 & (\ln 2)2^y & -3z^2 \\ 1 & 1 & 1 \end{pmatrix}$$

Soit le processus itératif suivant le schéma de Newton–Raphson

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \\ z_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} - \begin{pmatrix} \cos x_k & 2y_k & \dfrac{1}{z_k} \\ 2 & (\ln 2)2^{y_k} & -3z_k^2 \\ 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} \sin x_k + y_k^2 + \ln z_k - 7 \\ 2x_k + 2^{y_k} - z_k^3 + 1 \\ x_k + y_k + z_k - 5 \end{pmatrix}$$

Le script suivant montre que ce processus converge rapidement vers la solution (Fig. 3.13) en tenant compte du critère de convergence.

```
% Définition de l'itération de départ
x(1)=1; x(2)=1; x(3)=1; x=x';
for k=1:1000
% Calcul du vecteur fonction
 f(1)=sin(x(1))+x(2)^2+log(x(3))-7;
 f(2)=2*x(1)+2^(x(2))-x(3)^3+1;
 f(3)=x(1)+x(2)+x(3)-5;
% Composantes de la matrice Jacobienne
J(1,1)=cos(x(1)); J(1,2)=2*x(2);
   J(1,3)=1/x(3);
J(2,1)=2; J(2,2)=(2^(x(2)))*log(2);
   J(2,3)=-3*(x(3)^2);
J(3,1)=1; J(3,2)=1;J(3,3)=1;
% Calcul du nouveau vecteur xPlus
 xPlus=x-inv(J)*f';
 X(k)=x(1); Y(k)=x(2); Z(k)=x(3);
% Critère de convergence
  if abs(xPlus-x)<=0.000001
     break
  else
     x=xPlus;
  end
end
j=1:numel(X);
plot(j,X,'-og',j,Y,'-*r',j,Z,'-+b');
```

k	x_k	y_k	z_k
1	1	1	1
2	-0.6895	3.3819	2.3076
3	0.4600	2.5879	1.9520
4	0.6424	2.3992	1.9584
5	0.6463	2.3925	1.9612
6	0.6463	2.3925	1.9612

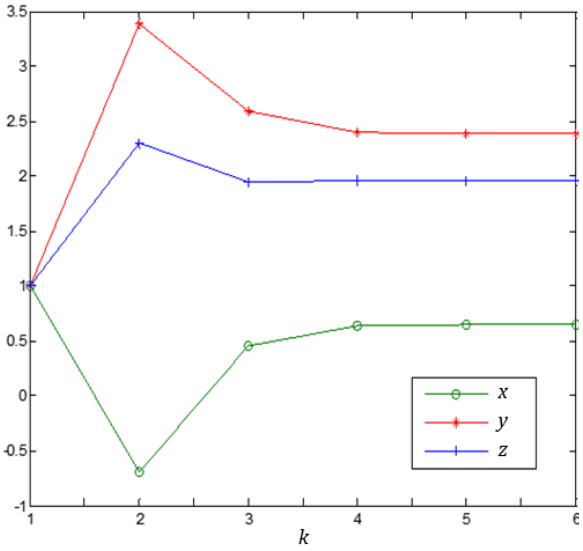


Fig. 3.13 : Résultats.

Exercice 8

Résoudre le système d'équations

$$x^2 + xy^3 = 9 \qquad 3x^2y - y^3 = 4$$

En utilisant la méthode de Newton–Raphson. Utiliser $(x_1,\ y_1) = (1.2,\ 2.5)$ comme première itération.

Dans cette exercice, $f_1(x,\ y) = x^2 + xy^3 - 9$ et $f_2(x,\ y) = 3x^2y - y^3 - 4$. La matrice jacobienne est:

$$\mathbf{J}(x,\ y) = \begin{pmatrix} 2x + y^3 & 3xy^2 \\ 6xy & 3x^2 - 3y^2 \end{pmatrix}$$

Soit le processus de Newton–Raphson pour la résolution du système d'équations :

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - \begin{pmatrix} 2x_k + y_k^3 & 3x_k y_k^2 \\ 6x_k y_k & 3x_k^2 - 3y_k^2 \end{pmatrix}^{-1} \begin{pmatrix} x_k^2 + x_k y_k^3 - 9 \\ 3x_k^2 y_k - y_k^3 - 4 \end{pmatrix}$$

Soit le script MATLAB pour déterminer la solution suivant ce procédé :

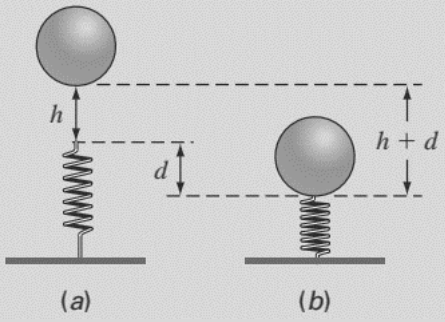
```
x(1)=1.2;
x(2)=2.5; % x(1) et x(2) sont les composantes initiales
   x=x';
for k=1:1000
% Vecteur fonction
 f(1)=x(1)^2+x(1)*(x(2)^3)-9;
 f(2)=3*(x(1)^2)*x(2)-x(2)^3-4;
% Matrice Jacobienne
J(1,1)=2*x(1)+x(2)^3;J(1,2)=3*x(1)*(x(2)^2);
J(2,1)=6*x(1)*x(2);J(2,2)=3*(x(1)^2-x(2)^2);
% Calcul du vecteur xPlus
   xPlus=x-inv(J)*f';
% Donc,
  X(k)=x(1); Y(k)=x(2);
% Utilisant le critère de convergence
 if abs(xPlus-x)==0
     break
 else
      x=xPlus;
 end
end
j=1:numel(X);
plot(j,X,'-ob',j,Y,'-*r');
```

k	x_k	y_k
1	1.2	2.5
2	1.2557672	1.957991
3	1.3228345	1.772769
4	1.3361912	1.7543637
5	1.3363554	1.7542352
6	1.3363554	1.7542352
7	1.3363554	1.7542352

Exercice 9

Les systèmes mécaniques réels peuvent impliquer la déviation de ressorts non linéaires. Dans la figure 3.14, un bloc de masse m est relâché à une distance h au-dessus d'un ressort non linéaire. La force de résistance F du ressort est donnée par :

$$F = -\left(k_1 d + k_2 d^{3/2}\right)$$



La conservation de l'énergie peut être utilisée pour montrer que,

$$\frac{2k_2 d^{5/2}}{5} + \frac{1}{2}k_1 d^2 - mgd - mgh = 0$$

Résoudre pour d et calculer la force de résistance F.

Sachant que : $k_1 = 40000\ \text{g/s}^2$, $k_2 = 40\ \text{g/}\left(\text{s}^2\text{m}^{1/2}\right)$, $m = 95\ \text{g}$, $g = 9.81\ \text{m/s}^2$ et $h = 0.43\ \text{m}$.

Considérant la fonction :

$$f(d) = \frac{2k_2 d^{5/2}}{5} + \frac{1}{2}k_1 d^2 - mgd - mgh$$

dont sa dérivée :

$$f'(d) = k_2 d^{3/2} + k_1 d - mg$$

soit la processus de Newton-Raphson :

$$d_{i+1} = d_i - \frac{f(d_i)}{f'(d_i)} \Rightarrow d_{i+1} = d_i - \frac{\frac{2k_2 d_i^{5/2}}{5} + \frac{1}{2}k_1 d_i^2 - mgd_i - mgh}{k_2 d_i^{3/2} + k_1 d_i - mg}$$

Qui peut être simplifié :

$$d_{i+1} = \frac{\frac{3k_2 d_i^{5/2}}{5} + \frac{1}{2}k_1 d_i^2 + mgh}{k_2 d_i^{3/2} + k_1 d_i - mg}$$

Le script MATLAB suivant permet de calculer *d* et par conséquent la force de résistance *F* :

```
k=0;
dintial= 0.1;
dfinal=(24*(dintial^(5/2))+20000*dintial*dintial+95*9.81*0.43);
dfinal=dfinal/(40*(dintial^(3/2))+40000*dintial-95*9.81);
d(k+1)=dfinal;
while abs(dfinal-dintial)>0.00000001
 k=k+1;
  dintial=dfinal;
   dfinal=(24*(dintial^(5/2))+20000*dintial*dintial+95*9.81*0.43);
  dfinal=dfinal/(40*(dintial^(3/2))+40000*dintial-95*9.81);
 d(k)=dfinal;
end
disp ('Convergence a ');k
disp ('La valeur de d en metres est ');d(k)
F=(40000*d(k)+40*(d(k)^(3/2)))/1000;
disp ('La force de resistance en Newton est ');F
```

Les résultats obtenus après l'exécution du script :

```
Convergence a
k =
     4
La valeur de d en metres est
ans =
    0.1667
La force de resistance en Newton est
F =
    6.6717
```

Exercice 10

Utiliser la méthode de Newton–Raphson pour trouver la valeur minimale de la fonction :

$$f(x,\ y) = x^4 + xy + (1+y)^2$$

C'est un problème d'optimisation sans contraintes. Pour calculer la valeur minimale d'une fonction à plusieurs variable $f(x,\ y)$, il faut résoudre l'équation vectorielle :

$$\nabla f = \mathbf{0}$$

Où, ∇ est l'opérateur gradient, par rapport à une base canonique $(\mathbf{e}_x,\ \mathbf{e}_y)$,

$$\nabla f = \frac{\partial f}{\partial x}\mathbf{e}_x + \frac{\partial f}{\partial y}\mathbf{e}_y$$

par conséquent,

$$\nabla f = \mathbf{0} \;\Rightarrow\; \frac{\partial f}{\partial x}\mathbf{e}_x + \frac{\partial f}{\partial y}\mathbf{e}_y = \mathbf{0} \;\Rightarrow\; \frac{\partial f}{\partial x} = 0 \text{ et } \frac{\partial f}{\partial y} = 0$$

c'est-à-dire,

$$4x^3 + y = 0 \qquad x + 2y + 2 = 0$$

de l'équation,

$$4x^3 + y = 0 \;\Rightarrow\; y = -4x^3, \text{ alors } x - 8x^3 + 2 = 0$$

Le processus de Newton–Raphson appliqué à cette équation est :

$$x_{k+1} = x_k - \frac{x_k - 8x_k^3 + 2}{\left(x - 8x^3 + 2\right)'\Big|_{x=x_k}} \;\Leftrightarrow\; x_{k+1} = x_k - \frac{x_k - 8x_k^3 + 2}{1 - 24x_k^2}$$

Pour un choix de $x_1 = 0.5$ comme première itération, soit :

$$x_2 = x_1 - \frac{x_1 - 8x_1^3 + 2}{1 - 24x_1^2} = 0.5 - \frac{0.5 - 8\times(0.5)^3 + 2}{1 - 24\times(0.5)^2} = 0.8000$$

$$x_3 = x_2 - \frac{x_2 - 8x_2^3 + 2}{1 - 24x_2^2} = 0.8 - \frac{0.8 - 8\times(0.8)^3 + 2}{1 - 24\times(0.8)^2} = 0.7097$$

$$x_4 = x_3 - \frac{x_3 - 8x_3^3 + 2}{1 - 24x_3^2} = 0.7097 - \frac{0.7097 - 8\times(0.7097)^3 + 2}{1 - 24\times(0.7097)^2} = 0.6962$$

$$x_5 = x_4 - \frac{x_4 - 8x_4^3 + 2}{1 - 24x_4^2} = 0.6962 - \frac{0.6962 - 8\times(0.6962)^3 + 2}{1 - 24\times(0.6962)^2} = 0.6959$$

$$x_6 = x_5 - \frac{x_5 - 8x_5^3 + 2}{1 - 24x_5^2} = 0.6959 - \frac{0.6959 - 8\times(0.6959)^3 + 2}{1 - 24\times(0.6959)^2} = 0.6959$$

Remarquons que le processus converge vers la solution $x = x_6 = 0.6959$ par conséquent $y = -1.3479$.

Pour $(x,\ y) = (0.6959,\ -1.3479)$, $f(0.6959,\ -1.3479) = -0.5824$, qui est la valeur minimale de la fonction $f(x,\ y) = x^4 + xy + (1+y)^2$

Si on utilise l'outil symbolique de MATLAB, le tracé de la fonction $f(x,\ y)$ se fait comme suit (Fig. 3.14) :

```
>> z=sym('x^4+x*y+(1+y)^2')
>> ezsurf(z)
```

Le tracé de la fonction $f(x,\ y)$ montre bien l'existence de son minimum qui concorde mieux avec le résultat numérique.

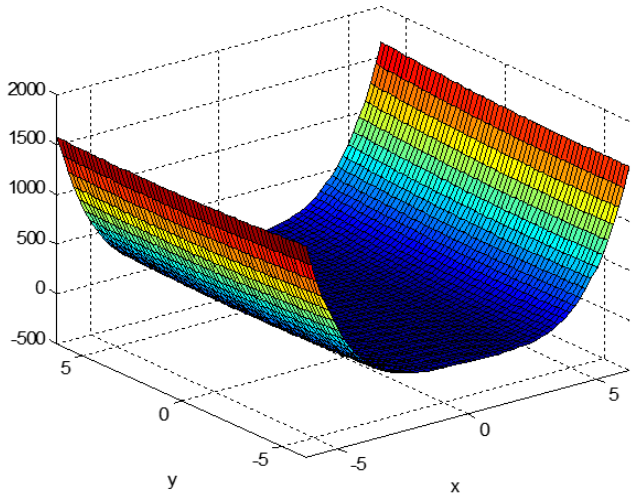


Fig. 3.14 : Tracé de la fonction $f(x,y)=x^4+xy+(1+y)^2$.

Exercice 11

Soit le système d'équations non linéaires :

$$\begin{cases} \sin(xy) - x^2 - 5x - y = 0 \\ \cos(x+y) + y^2 - x - 6y + 2 = 0 \end{cases}$$

Déterminer la solution du système par la méthode du point fixe, en prend comme première itération $(0,\ 0)$.

Le système d'équations peut être s'écrit :

$$\begin{cases} x = \left(\sin(xy) - x^2 - y\right)/5 \\ y = \left(\cos(x+y) + y^2 - x + 2\right)/6 \end{cases}$$

Soit le processus itératif :

$$\begin{cases} x_{i+1} = \left(\sin(x_i y_i) - x_i^2 - y_i\right)/5 \\ y_{i+1} = \left(\cos(x_i + y_i) + y_i^2 - x_i + 2\right)/6 \end{cases} \text{ avec, } \begin{cases} x_0 = 0 \\ y_0 = 0 \end{cases}$$

Alors,

$$\begin{cases} x_1 = \left(\sin(x_0 y_0) - x_0^2 - y_0\right)/5 = 0 \\ y_1 = \left(\cos(x_0 + y_0) + y_0^2 - x_0 + 2\right)/6 = 0.5 \end{cases}$$

De même,

$$\begin{cases} x_2 = \left(\sin(x_1 y_1) - x_1^2 - y_1\right)/5 = -0.1 \\ y_2 = \left(\cos(x_1 + y_1) + y_1^2 - x_1 + 2\right)/6 = 0.521264 \end{cases}$$

De la même manière :

$$x_3 = -0.116673 \quad y_3 = 0.547381$$
$$x_4 = -0.124963 \quad y_4 = 0.554162$$
$$x_5 = -0.127794 \quad y_5 = 0.556893$$
$$x_6 = -0.128866 \quad y_6 = 0.557878$$
$$x_7 = -0.129263 \quad y_7 = 0.558245$$
$$x_8 = -0.129410 \quad y_8 = 0.558382$$
$$x_9 = -0.129465 \quad y_9 = 0.558432$$
$$x_{10} = -0.129486 \quad y_{10} = 0.558451$$
$$x_{11} = -0.129486 \quad y_{11} = 0.558451$$

Alors, la solution approchée du système est :

$$x = -0.129486 \quad y = 0.558451$$

Exercice 12

Soit le système d'équations non linéaires suivant :

$$\begin{cases} \sin x - x^2 y^2 - x^4 = 0, \\ x^2 - x + y^2 = 0. \end{cases}$$

1. Déterminer la matrice jacobienne du système et construire le processus itératif de Newton-Raphson.
2. Montrer que ce système peut être se transformer sous la forme de $\sin x = x^3$ et donner dans ce cas le schéma de résolution suivant la méthode de Newton-Raphson de cette équation.
3. Soit $x_1 = -1.20$ la première itération, déterminer dans ce cas la solution de l'équation $\sin x = x^3$ et par conséquent la solution du système d'équation.

1. Ce système est formé par deux équations et deux inconnus *x* et *y*. Alors le vecteur fonction est formé par les deux fonctions $f_1(x,\ y) = \sin x - x^2 y^2 - x^4$ et $f_2(x,\ y) = x^2 - x + y^2$, par conséquent la matrice jacobienne est donnée par :

$$\mathbf{J}(x,\ y) = \begin{pmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} \end{pmatrix} = \begin{pmatrix} \cos x - 2xy^2 - 4x^3 & -2x^2 y \\ 2x - 1 & 2y \end{pmatrix}$$

Alors le processus itératif de Newton–Raphson pour résoudre ce système est :

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - \begin{pmatrix} \cos x_k - 2x_k y_k^2 - 4x_k^3 & -2x_k^2 y_k \\ 2x_k - 1 & 2y_k \end{pmatrix}^{-1} \begin{pmatrix} \sin x_k - x_k^2 y_k^2 - x_k^4 \\ x_k^2 - x_k + y_k^2 \end{pmatrix}$$

2. A partir de la deuxième équation du système $x^2 - x + y^2 = 0 \Rightarrow y^2 = -x^2 + x$, par substitution dans la première équation du système :

$\sin x - x^2 y^2 - x^4 = 0 \Rightarrow \sin x - x^2\left(-x^2 + x\right) - x^4 = 0 \Rightarrow \sin x + x^4 - x^3 - x^4 = 0 \Rightarrow \sin x - x^3 = 0$

alors, $\sin x = x^3$ où, $f\left(x\right) = \sin x - x^3$.

Le processus de Newton–Raphson pour l'équation $f\left(x\right) = 0$ est :

$$x_{k+1} = x_k - \frac{f\left(x_k\right)}{f'\left(x_k\right)} \Rightarrow x_{k+1} = x_k - \frac{\sin x_k - x_k^3}{\cos x_k - 3x_k^2}$$

3. Pour une première itération $x_1 = -1.20$ soit,

$$x_2 = x_1 - \frac{\sin x_1 - x_1^3}{\cos x_1 - 3x_1^2} = \left(-1.20\right) - \frac{\sin\left(-1.20\right) - \left(-1.20\right)^3}{\cos\left(-1.20\right) - 3\times\left(-1.20\right)^2} = -0.9989$$

$$x_3 = x_2 - \frac{\sin x_2 - x_2^3}{\cos x_2 - 3x_2^2} = \left(-0.9989\right) - \frac{\sin\left(-0.9989\right) - \left(-0.9989\right)^3}{\cos\left(-0.9989\right) - 3\times\left(-0.9989\right)^2} = -0.9353$$

$$x_4 = x_3 - \frac{\sin x_3 - x_3^3}{\cos x_3 - 3x_3^2} = \left(-0.9353\right) - \frac{\sin\left(-0.9353\right) - \left(-0.9353\right)^3}{\cos\left(-0.9353\right) - 3\times\left(-0.9353\right)^2} = -0.9287$$

$$x_5 = x_4 - \frac{\sin x_4 - x_4^3}{\cos x_4 - 3x_4^2} = \left(-0.9287\right) - \frac{\sin\left(-0.9287\right) - \left(-0.9287\right)^3}{\cos\left(-0.9287\right) - 3\times\left(-0.9287\right)^2} = -0.9286$$

$$x_6 = x_5 - \frac{\sin x_5 - x_5^3}{\cos x_5 - 3x_5^2} = \left(-0.9286\right) - \frac{\sin\left(-0.9286\right) - \left(-0.9286\right)^3}{\cos\left(-0.9286\right) - 3\times\left(-0.9286\right)^2} = -0.9286$$

Le processus converge dès la cinquième itération vers la solution $x = -0.9286$.
par conséquent,

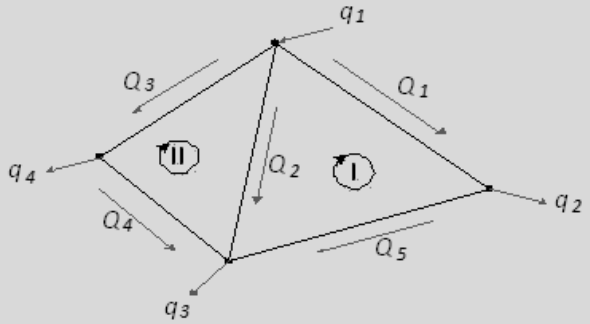
$$y^2 = -\left(-0.9286\right)^2 + \left(-0.9286\right) = -1.7909 \Rightarrow y = 1.3382\ i$$

Exercice 13

Soit un réseau maillé académique formé par cinq tronçons t_1, t_2, t_3, t_4 et t_5 de diamètres D_1, D_2, D_3, D_4 et D_5 et de longueurs l_1, l_2, l_3, l_4 et l_5 (voir tableau et figure ci-dessous).

i	l_i (m)	D_i (mm)	***CHW***
1	9	60	130
2	13	40	130
3	5	40	130
4	5	40	130
5	9	60	130

q_1=18 *l/s*
q_2=6 *l/s*
q_3=8 *l/s*
q_4=4 *l/s*



1. Etablir le modèle mathématique de ce réseau, sachant que la perte de charge dans une conduite transitée par un débit Q_i est exprimée par la formule de Hazen–Williams.
2. En utilisant la méthode de Newton–Raphson calculer les débits Q_1, Q_2, Q_3, Q_4 et Q_5 dans chaque tronçon.

1. L'utilisation du principe de la conservation de la masse conduit à établir quatre équations de continuité suivante :

$$\text{Nœud 1 : } q_1 = Q_1 + Q_2 + Q_3$$

$$\text{Nœud 2 : } Q_1 = q_2 + Q_5$$

$$\text{Nœud 3 : } Q_2 + Q_4 + Q_5 = q_3$$

$$\text{Nœud 4 : } Q_3 = Q_4 + q_4$$

Le principe de la conservation de la quantité de mouvement permet d'établir deux équations aux mailles :

$$\text{Maille I : } \Delta h_1 - \Delta h_2 + \Delta h_5 = 0$$

$$\text{Maille II : } \Delta h_2 - \Delta h_3 - \Delta h_4 = 0$$

A partir de ces équations, soit le système d'équations non linéaire à résoudre :

$$\begin{cases} f_1(Q_1,\ Q_2,\ Q_3,\ Q_4,\ Q_5) = Q_1 + Q_2 + Q_3 - q_1 = 0, \\ f_2(Q_1,\ Q_2,\ Q_3,\ Q_4,\ Q_5) = Q_1 - Q_5 - q_2 = 0, \\ f_3(Q_1,\ Q_2,\ Q_3,\ Q_4,\ Q_5) = Q_2 + Q_4 + Q_5 - q_3 = 0, \\ f_4(Q_1,\ Q_2,\ Q_3,\ Q_4,\ Q_5) = R_1 Q_1^{\alpha} - R_2 Q_2^{\alpha} + R_5 Q_5^{\alpha} = 0, \\ f_5(Q_1,\ Q_2,\ Q_3,\ Q_4,\ Q_5) = R_2 Q_2^{\alpha} - R_3 Q_3^{\alpha} - R_4 Q_4^{\alpha} = 0. \end{cases}$$

où,

$$R_i = \left(\frac{3.592}{CHW}\right)^{1.852} \frac{L_i}{D_i^{4.87}} \quad \text{et} \quad \alpha = 1.852$$

2. Le modèle mathématique ainsi formé est un système d'équations non linéaires formé par cinq équations et cinq inconnus Q_1, Q_2, Q_3, Q_4 et Q_5 à déterminer. Le schéma de résolution numérique de Newton–Raphson est :

$$\mathbf{Q}^{(k+1)} = \mathbf{Q}^{(k)} - \mathbf{J}^{-1}\left(\mathbf{Q}^{(k)}\right)\mathbf{f}\left(\mathbf{Q}^{(k)}\right)$$

avec,

$$\mathbf{Q}^{(k+1)} = \begin{pmatrix} Q_1^{(k+1)} \\ Q_2^{(k+1)} \\ Q_3^{(k+1)} \\ Q_4^{(k+1)} \\ Q_5^{(k+1)} \end{pmatrix},\ \mathbf{Q}^{(k)} = \begin{pmatrix} Q_1^{(k)} \\ Q_2^{(k)} \\ Q_3^{(k)} \\ Q_4^{(k)} \\ Q_5^{(k)} \end{pmatrix},\ \mathbf{f}\left(\mathbf{Q}^{(k)}\right) = \begin{pmatrix} Q_1^{(k)} + Q_2^{(k)} + Q_3^{(k)} - q_1 \\ Q_1^{(k)} - Q_5^{(k)} - q_2 \\ Q_2^{(k)} + Q_4^{(k)} + Q_5^{(k)} - q_3 \\ R_1\left(Q_1^{(k)}\right)^{\alpha} - R_2\left(Q_2^{(k)}\right)^{\alpha} + R_5\left(Q_5^{(k)}\right)^{\alpha} \\ R_2\left(Q_2^{(k)}\right)^{\alpha} - R_3\left(Q_3^{(k)}\right)^{\alpha} - R_4\left(Q_4^{(k)}\right)^{\alpha} \end{pmatrix}$$

et

$$\mathbf{J}\left(\mathbf{Q}^{(k)}\right)=\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 1 & 1 \\ \alpha R_1\left(Q_1^{(k)}\right)^{\alpha-1} & -\alpha R_2\left(Q_2^{(k)}\right)^{\alpha-1} & 0 & 0 & \alpha R_5\left(Q_5^{(k)}\right)^{\alpha-1} \\ & \alpha R_2\left(Q_2^{(k)}\right)^{\alpha-1} & -\alpha R_3\left(Q_3^{(k)}\right)^{\alpha-1} & -\alpha R_4\left(Q_4^{(k)}\right)^{\alpha-1} & 0 \end{pmatrix}$$

Soit le script suivant qui permet de résoudre ce problème pour une distribution initiale des débits en chaque tronçon qui respecte les équations de continuité aux nœuds.

```
longueur=[9 13 5 5 9]; % en m Diametre=[60 40 40 40 60]; % en mm CHW=130;
q=[18 6 8 4]; q=q/1000; % en m3/s
 for i=1:numel(longueur)
  R(i)=((3.592/CHW)^1.852)*longueur(i)/((Diametre(i)/1000)^4.87);
 end
% Définition de l'itération de départ
Qi=[9 3 6 2 3]; Qi=Qi/1000; % en m3/s
Qi=Qi';
for k=1:1000
% Calcul du vecteur fonction
   f(1)=Qi(1)+Qi(2)+Qi(3)-q(1); f(2)=Qi(1)-Qi(5)-q(2);
  f(3)=Qi(2)+Qi(4)+Qi(5)-q(3);
 f(4)=R(1)*(Qi(1)^(1.852))-R(2)*(Qi(2)^(1.852))+R(5)*(Qi(5)^(1.852));
f(5)=R(2)*(Qi(2)^(1.852))-R(3)*(Qi(3)^(1.852))-R(4)*(Qi(4)^(1.852));
% Calcul des composantes de la matrice Jaconienne
 J(1,1)=1;  J(1,2)=1;  J(1,3)=1; J(1,4)=0; J(1,5)=0;
 J(2,1)=1;  J(2,2)=0;  J(2,3)=0; J(2,4)=0; J(2,5)=-1;
 J(3,1)=0;  J(3,2)=1;  J(3,3)=0; J(3,4)=1; J(3,5)=1;
 J(4,1)=1.852*R(1)*(Qi(1)^(0.852)); J(4,2)=-1.852*R(2)*(Qi(2)^(0.852));
 J(4,3)=0; J(4,4)=0; J(4,5)=1.852*R(5)*(Qi(5)^(0.852));
 J(5,1)=0; J(5,2)=1.852*R(2)*(Qi(2)^(0.852));
 J(5,3)=-1.852*R(3)*(Qi(3)^(0.852)); J(5,4)=-1.852*R(4)*(Qi(4)^(0.852));
 J(5,5)=0;
% Calcul du nouveau vecteur Qf par Newton-Raphson
  Qf=Qi-inv(J)*f';
   Q1(k)=1000*Qi(1); Q2(k)=1000*Qi(2); Q3(k)=1000*Qi(3);
     Q4(k)=1000*Qi(4); Q5(k)=1000*Qi(5);
% Utilisation d'un critère de convergence
  if abs(Qf-Qi)<=0.0000001
     break
  else
     Qi=Qf;
  end
end
% Vérification
    f1=Qf(1)+Qf(2)+Qf(3)-q(1); f2=Qf(1)-Qf(5)-q(2);
```

```
  f3=Qf(2)+Qf(4)+Qf(5)-q(3);
 f4=R(1)*(Qf(1)^(1.852))-R(2)*(Qf(2)^(1.852))+R(5)*(Qf(5)^(1.852));
f5=R(2)*(Qf(2)^(1.852))-R(3)*(Qf(3)^(1.852))-R(4)*(Qf(4)^(1.852));
```

L'exécution de ce script permet d'avoir les résultats dans la fenêtre "Workspace" de MATLAB. Soit les débits en $l.s^{-1}$ après la résolution du système d'équation par la méthode de Newton–Raphson qui parcourt chaque conduite :

$$Q_1 = 9.9,\ Q_2 = 3.1,\ Q_3 = 5.0,\ Q_4 = 1.0 \text{ et } Q_5 = 3.1$$

Pour l'équation au nœud 4 est bien vérifiée avec ces résultats :

$$Q_3 = Q_4 + q_4 \quad \Leftrightarrow \quad 5.0 = 1.0 + 4$$

9. Exercices supplémentaires

Exercice 1

Déterminer une solution par la méthode de bissection à une tolérance de 10^{-5}, les équations non linéaires suivantes.

(a) $e^x - 3x^2 = 0$ (b) $x^3 = x^2 + x + 1$ (c) $e^x = \dfrac{1}{x^2+1}$ (d) $x = 1 + 0.3\cos(x)$

Exercice 2

Déterminer une racine par la méthode du point fixe de la fonction :

$$f(x) = \sin\left(\sqrt{x}\right) - x$$

(a) Utiliser comme itération initiale $x_0 = 0.5$ et pour un seuil de convergence $\varepsilon_a \le 0.01\%$

(b) Vérifier que la convergence du processus utilisé est linéaire.

Exercice 3

Montrer que la fonction :

$$f(x) = \cos(x) - xe^x$$

admet une racine unique dans l'intervalle $[0, \pi/2]$. Expliciter l'algorithme de Newton-Raphson sur cet exemple.

Exercice 4

Utiliser la méthode du point fixe et de Newton-Raphson pour déterminer une racine de la fonction $f(x) = -0.9x^2 + 1.7x + 2.5$, pour une première itération $x_0 = 5$ et pour un seuil de convergence $\varepsilon_a \le 0.01\%$.

Exercice 5

Déterminer la racine réelle la plus élevée de la fonction $f(x) = x^3 - 6x^2 + 11x - 6.1$:

(a) Graphiquement.

(b) Par la méthode de Newton-Raphson pour $x_0 = 3.5$.

(c) Par la méthode de la sécante avec $x_0 = 2.5$ et $x_1 = 3.5$.

(d) Par la méthode de la sécante modifiée avec $x_0 = 3.5,\ \delta = 0.01$.

(e) Déterminer tous les racines avec MATLAB.

Exercice 6

Déterminer la plus petite racine positive de la fonction $f(x)=7\sin(x)e^{x}-1$:

(a) Graphiquement.

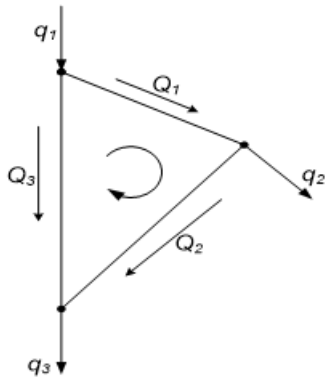
(b) Avec la méthode de Newton-Raphson ($x_0=0.3$).

(c) Avec la méthode de la sécante ($x_0=0.5$ et $x_1=0.3$).

(d) Avec la méthode de la sécante modifiée ($x_0=0.3,\ \delta=0.01$).

Exercice 7

Soit un réseau de conduites réalisé dans le laboratoire formé par trois conduites T_1, T_2 et T_3 de diamètres D_1, D_2 et D_3 et longueurs L_1, L_2 et L_3 (voir figure ci-contre).



1. Etablir les équations propres à ce réseau (nœuds et maille).
2. Etablir le processus itératif de Newton–Raphson qui permet de calculer les débits Q_1, Q_2 et Q_3 dans chaque tronçon.

Exercice 8

Utiliser la méthode de Newton-Raphson pour la racine unique de l'équation :

$$x+e^{-Bx^2}\cos(x)=0,\ B>0$$

Utiliser le processus de Newton-Raphson pour $x_0=0$ et $B=1,\ 5,\ 10,\ 25,\ 50$ et expliquer ce que vous remarquez à propos de la convergence du processus et comparer les résultats quand $B_0\approx\infty$.

Exercice 9

Soit le système d'équations non linéaires :

$$\begin{cases} x_0^2-2x_0-2x_1^2+x_1=0 \\ x_0^2+x_0-2x_1^2-1.5x_1-0.05=0 \end{cases}$$

Déterminer une solution du système en implémentant un script MATLAB utilisant la méthode du point fixe.

Exercice 10

La chaleur spécifique à pression constante nulle c_p en KJ/(kg.K) de l'air humide est liée à la température T (en °K) par la relation polynomiale :

$$c_p=0.99403+1.671\times10^{-4}T+9.7215\times10^{-8}T^2-9.5838\times10^{-11}T^3+1.9520\times10^{-14}T^4$$

Ecrire un script MATLAB permettant de :

(a) Tracer c_p en fonction de la variation de la température T entre 0 et 1200 °K.

(b) Déterminer la température correspond à $c_p=1.1$ kJ/(kg.K).

Exercice 11

Dans un processus de génie chimique, la vapeur d'eau (H_2O) est chauffée à des températures suffisamment élevées pour qu'une partie importante de l'eau se dissocie ou se sépare pour former de l'oxygène (O_2) et de l'hydrogène (H_2) :

$$2H_2O \rightleftarrows 2H_2 + O_2$$

Si l'on suppose que c'est la seule réaction impliquée, la fraction molaire x de H_2O qui se dissocie peut-être représentée par

$$K = \frac{x}{1-x}\sqrt{\frac{2p_t}{2+x}}$$

où K est la constante d'équilibre de la réaction et p_t est la pression totale du mélange. Si $p_t = 3$ atm et $K = 0.05$, déterminer la valeur de x.

Exercice 12

L'équation d'état de *Redlich-Kwong* est exprimée par :

$$p = \frac{RT}{v-b} - \frac{a}{v(v+b)\sqrt{T}}$$

avec R, constante universelle du gaz (= 0.518 kJ/(kg.K)), T, température absolue (en K) et v, le volume d'un kg du gaz (en m^3/kg).
a et b sont deux paramètres peuvent être calculés par :

$$a = 0.427\frac{R^2T_c^{2.5}}{p_c}, \quad b = 0.0866\frac{RT_c}{p_c}$$

ou, $p_c = 4600$ kPa et $T_c = 191$ K

Déterminer la quantité de méthane qui peut être détenu dans un réservoir de 3 m^3 à une température de -40 °C et à une pression de 65000 kPa.

Exercice 13

L'équation de Manning s'écrit pour un canal rectangulaire à surface libre,

$$Q = \frac{\sqrt{I}(BH)^{5/3}}{n(B+2H)^{2/3}}$$

avec, Q le débit, I la pente du canal, H la profondeur et n le coefficient de rugosité de Manning.

Développer une méthode du point fixe pour calculer la profondeur H pour Q = 5 $m^3.s^{-1}$ I = 0.0002, B = 20 m et $n = 0.03$.

Exercice 14

Soit l'équation, $x^2 = 2^x$

Monter avec l'outil symbolique de MATLAB que cette équation possède trois racines et utiliser le processus itératif de Newton-Raphson pour déterminer la racine existant dans l'intervalle [-1,0].

Chapitre 4.

Interpolation numérique et approximation

La théorie de l'interpolation est considérée comme la fondation du développement des méthodes numériques du calcul différentiel, intégrale, la théorie des approximations et la résolution numérique des équations différentielles.

1. Motivation

Soit un ensemble de données (entrée-sortie) $\{(x_i,\ y_i),\ i=0,\ 1,\ \ldots,\ n\}$ avec $x_i \in [a,\ b]$, $n+1$ valeurs distinctes ou nœuds. La question qui se pose, quelle est la valeur de y pour une donnée $x \neq x_i$?

Pour répondre à cette question, il y a pas mal de méthodes en mathématiques ; l'interpolation numérique et la régression.

2. Quelques notions essentielles sur l'algèbre des polynômes

Dans l'ensemble des nombres réelles, un polynôme de degré *n* est exprimé par :

$$P_n(x)=\sum_{k=0}^{n} a_k x^k = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

- La suite, $a_0,\ a_1,\ \ldots,\ a_{n-1}$ et a_n constitue les coefficients du polynôme $P_n(x)$.
- L'ensemble des polynômes constitue un espace vectoriel dans $\mathbb{R}$ ou $P_n(x)$ est un vecteur de coordonnées $(a_0,\ a_1,\ \ldots,\ a_{n-1},\ a_n)$ dans le repère de base $(1,\ x,\ \ldots,\ x^{n-1},\ x^n)$.
- Un polynôme de degré *n*, à *n* racines dans l'ensemble des nombres complexe.
- Si $r_1,\ r_2,\ \ldots,\ r_{n-1}$ et r_n les racines du polynôme $P_n(x)$, alors,

$$P_n(x)=\prod_{j=1}^{n}(x-r_j)=(x-r_1)(x-r_2)\ldots(x-r_{n-1})(x-r_n)$$

3. Interpolation de Vandermonde

Etant donnée l'ensemble $\{(x_i,\ y_i),\ i=0,\ 1,\ \ldots,\ n\}$ avec $x_i \in [a,\ b]$, $n+1$ valeurs distinctes, cherchons un polynôme de degré *n* sachant que :

$$P_n(x_i)=y_i,\ \ \forall i \in \{0,\ 1,\ 2,\ \ldots,\ n\}$$

c'est-à-dire,

$$y_i = a_n x_i^n + a_{n-1}x_i^{n-1} + \cdots + a_2 x_i^2 + a_1 x_i + a_0,\ \ \forall i \in \{0,\ 1,\ 2,\ \ldots,\ n\}$$

L'équation $P_n(x_i)=y_i$, conduit à un système d'équations linéaires dont les inconnus sont les coefficients $a_n,\ a_{n-1},\ a_{n-2},\ \ldots,\ a_1,\ a_0$:

$$\begin{pmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \cdots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \cdots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \cdots & x_n & 1 \end{pmatrix} \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

La matrice carrée du système d'équations est appelée la matrice de Vandermonde. Son déterminant est explicité par :

$$\prod_{i,\ j=1,\ i\neq j}^{n+1} \left(x_i - x_j\right) \neq 0$$

Pour construire $P_n(x)$, il suffit de résoudre ce système et obtenir les valeurs des coefficients $a_n,\ a_{n-1},\ a_{n-2},\ \ldots,\ a_1,\ a_0$. Soit l'exemple :

x_i	0	1	2
y_i	1	3	7

Le polynôme qu'il faut chercher ici est de degré 2 c'est-à-dire :

$$P_2(x) = ax^2 + bx + c$$

par conséquent,

$$\left.\begin{array}{lll} P_2(x_1) = y_1 & \Rightarrow & ax_1^2 + bx_1 + c = y_1 \ \Rightarrow\ c = 1 \\ P_2(x_2) = y_2 & \Rightarrow & ax_2^2 + bx_2 + c = y_2 \ \Rightarrow\ a + b + c = 3 \\ P_2(x_3) = y_3 & \Rightarrow & ax_3^2 + bx_3 + c = y_3 \ \Rightarrow\ 4a + 2b + c = 7 \end{array}\right\} \Rightarrow \begin{cases} a = 1, \\ b = 1, \\ c = 1. \end{cases}$$

alors,

$$P_2(x) = x^2 + x + 1$$

MATLAB construit la matrice de Vandermonde à partir d'un vecteur des nœuds **x** par la fonction **vander(x)** et calcul son inverse et par conséquent la résolution du système d'équations. Pour l'exemple précédent :

```
>> x=[0 1 2]; y=[1 3 7];
>> A=vander(x)
A =
     0     0     1
     1     1     1
     4     2     1
>> Coefficients=inv(A)*y'
Coefficients =
     1
     1
     1
```

La technique d'interpolation de Vandermonde nécessite la résolution d'un système d'équations linéaires et par conséquent l'inversion de la matrice, et par conséquent l'obtention d'une solution approchée très précise plus un temps de calcul. Le système obtenu suite à l'interpolation de Vandermonde peut être mal conditionné et par conséquent les coefficients obtenues suite à la résolution de du système linéaires sont loin de la réalité voulue. Pour cette raison d'autres interpolations seront utilisées.

4. Interpolation de Lagrange

Soit deux points $\{(x_i,\ y_i),\ i=1,\ 2\}$ et cherchons un polynôme d'interpolation qui sert à déterminer y a partir de $x_1 < x < x_2$ sous la forme :

$$L_1(x) \text{ et } L_2(x)$$

qui sont les coefficients de pondération de Lagrange $L_1(x)y_1$ est l'équation de la droite qui passe par les points de coordonnées $(x_1,\ y_1)$ et $(x_2,\ 0)$ et $L_2(x)y_2$ est l'équation de la droite qui passe par les points de coordonnées $(x_1,\ 0)$ et $(x_2,\ y_2)$ (Fig. 4.1).

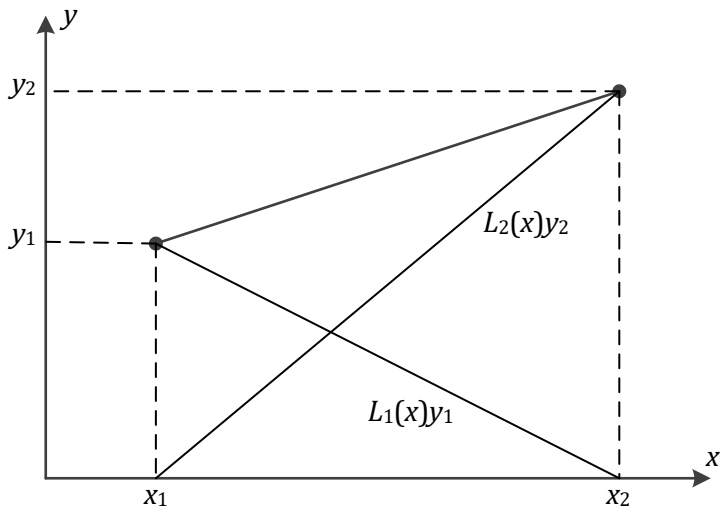


Fig. 4.1 : Interpolation de Lagrange.

c'est-à-dire,

$$L_1(x)y_1 = \frac{x-x_2}{x_1-x_2}y_1 \quad \Rightarrow \quad L_1(x) = \frac{x-x_2}{x_1-x_2}$$

$$L_2(x)y_2 = \frac{x-x_1}{x_2-x_1}y_2 \quad \Rightarrow \quad L_2(x) = \frac{x-x_1}{x_2-x_1}$$

par conséquent,

$$y = \frac{x-x_2}{x_1-x_2}y_1 + \frac{x-x_1}{x_2-x_1}y_2$$

C'est la formule d'interpolation linéaire de Lagrange. Cette interpolation est la méthode la plus simple pour estimer la valeur prise par une fonction continue entre deux points. Cette technique était d'un emploi systématique lorsque l'on ne disposait que de tables numériques pour le calcul avec les fonctions transcendantes : les tables comportaient d'ailleurs à cet effet en marge les ''différences tabulaires'', auxiliaire de calcul servant à l'interpolation linéaire.

Suivant la même stratégie, la formule d'interpolation de Lagrange passant par les points $\{(x_i,\ y_i),\ i=0,\ 1,\ 2\}$ $(x_i \text{ distincts})$ est exprimé par :

$$y = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2$$

Alors, étant donnée l'ensemble $\{(x_i,\ y_i),\ i=0,\ 1,\ \ldots,\ n\}$ avec $x_i \in [a,\ b]$, $n+1$ valeurs distinctes, le polynôme d'interpolation de Lagrange est exprimé par la formule :

$$P_n(x)=\sum_{j=0}^{n} y_j L_j(x) \quad \text{avec} \quad L_j(x)=\prod_{k=0,\ k\neq j}^{n}\frac{(x-x_k)}{(x_j-x_k)}$$

Les coefficients de pondération de Lagrange $L_j(x)$ peuvent être exprimés comme suit :

$$L_j(x)=\prod_{k=0,\ k\neq j}^{n}\frac{(x-x_k)}{(x_j-x_k)}=\frac{\prod_{k=0,\ k\neq j}^{n}(x-x_k)}{\prod_{k=0,\ k\neq j}^{n}(x_j-x_k)}=\frac{\prod_{k=0}^{n}(x-x_k)}{(x-x_j)\prod_{k=0,\ k\neq j}^{n}(x_j-x_k)}$$

soit,

$$\psi(x)=\prod_{k=0}^{n}(x-x_k)=(x-x_0)(x-x_1)(x-x_2)\cdots(x-x_j)\cdots(x-x_n)$$

alors,

$$\ln\psi(x)=\sum_{k=0}^{n}\ln(x-x_k) \ \Rightarrow\ \frac{d}{dx}\ln\psi(x)=\sum_{k=0}^{n}\frac{d}{dx}\ln(x-x_k) \ \Leftrightarrow\ \frac{\psi'(x)}{\psi(x)}=\sum_{k=0}^{n}\frac{1}{x-x_k}$$

où,

$$\psi'(x)=\sum_{k=0}^{n}\frac{\psi(x)}{x-x_k}$$

alors,

$$\psi'(x_j)=\sum_{k=0}^{n}\frac{\psi(x_j)}{x_j-x_k}=(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)=\prod_{k=0,\ k\neq j}^{n}(x_j-x_k)$$

finalement,

$$L_j(x)=\frac{\psi(x)}{(x-x_j)\,\psi'(x_j)}$$

Il est clair que,

$$L_j(x_i)\equiv\delta_{ij}$$

δ_{ij} est le symbole de Kronecker ($\delta_{ij}=1$ si $i=j$, $\delta_{ij}=0$ sinon)

Notons que la forme des coefficients de Lagrange est invariante par rapport à une substitution linéaire entière $x=\alpha t+\beta$ (ou α et β sont des constantes et $\alpha\neq 0$). En effet,

$$x_j=\alpha t_j+\beta \quad (j=0,\ 1,\ 2,\ \ldots,\ n)$$

alors,

$$\forall\ k=0,\ 1,\ 2,\ \ldots,\ n,\ \ x-x_k=\alpha(t-t_k) \ \text{ et } \ x_j-x_k=\alpha(t_j-t_k)$$

donc,

$$L_j(x)=\frac{\prod_{k=0,\ k\neq j}^{n}(x-x_k)}{\prod_{k=0,\ k\neq j}^{n}(x_j-x_k)}=\frac{\alpha^n\prod_{k=0,\ k\neq j}^{n}(t-t_k)}{\alpha^n\prod_{k=0,\ k\neq j}^{n}(t_j-t_k)}=\frac{\prod_{k=0}^{n}(t-t_k)}{(t-t_j)\prod_{k=0,\ k\neq j}^{n}(t_j-t_k)}=\frac{\psi(t)}{(t-t_j)\,\psi'(t_j)}\equiv L_j(t)$$

Dans le cas des nœuds uniformes dont le pas de discrétisation *h*, c'est-à-dire :

$$x = x_0 + th \quad \text{et } t_j = j$$

alors,

$$\psi(t) = t(t-1)(t-2)\cdots(t-n)$$

aussi,

$$\psi'(t_j) = (t_j - t_0)(t_j - t_1)\cdots(t_j - t_{j-1})(t_j - t_{j+1})\cdots(t_j - t_n)$$

c'est-à-dire,

$$\psi'(t_j) = j(j-1)\cdots 3\cdot 2\cdot 1\cdot(-1)\cdot(-2)\cdot(-3)\cdots(j-n) \equiv j!(-1)^{n-j}(n-j)!$$

alors,

$$\psi'(t_j) = (-1)^{n-j}\, j!(n-j)!$$

soit,

$$L_j(t) = \frac{(-1)^{n-j}\psi(t)}{(t-j)\,j!(n-j)!} = \frac{1}{n!}\psi(t)\frac{(-1)^{n-j}C_n^j}{t-j} \quad \text{avec,} \quad C_n^j = \frac{n!}{j!(n-j)!}$$

par conséquent le polynôme d'interpolation de Lagrange est déduit ainsi par :

$$P_n(x) = \frac{1}{n!}\frac{\psi(x)}{h^n}\sum_{j=0}^{n}(-1)^{n-j}\frac{C_n^j}{x-x_j}y_j$$

$\psi(x)$ est un polynôme de degré $n+1$, il est de la forme :

$$\psi(x) = x^{n+1} + \sum_{\eta=0}^{n}\alpha_\eta x^\eta$$

La (n+1)-ième dérivée de $\psi(x)$ est :

$$\psi^{(n+1)}(x) = (n+1)!$$

Soit la fonction **`Interp2Lagrange`**qui permet l'estimation de **`Y`** par interpolation à partir de **`X`** donnée, sachant que le polynôme d'interpolation passe par les points dont les abscisses et les ordonnées sont données respectivement par les vecteurs **`x`** et **`y`**.

```
function Y = Interp2Lagrange(x,y,X)
%Input : x = [x0 x1 ... xN], y = [y0 y1 ... yN], X = Valeur donnée
%Output: Y = Valeur interpolée correspond à X
N = length(x)-1; % Le degré du polynôme
  if length(y)~=N+1
   error('les dimensions de x et de y sont différentes ')
  end
l = 0;
for m = 1:N + 1
   P = 1;
 for k = 1:N + 1
     if k ~= m, P = conv(P,[1 -x(k)])/(x(m)-x(k)), end
 end
  l = l + y(m)*P; % Coefficients du polynôme de Lagrange
end
Y = polyval(l,X);
```

L'application de cette fonction se fait comme suit :

```
>> x=0:3; y=[1 7 35 103];
```

```
>> X=1.5;
>> Y= Interp2Lagrange(x,y,X)
Y =
   17.1250
```

4.1. Méthode de Van Wijngaarden-Dekker-Brent

L'interpolation de Lagrange est utilisée comme un outil pour améliorer de la solution des équations non linéaires $f(x)=0$ sans calculer les dérivées de celle-ci. C'est la méthode de Van Wijngaarden-Dekker-Brent (Forsythe *et al.*, 1977 et Brent, 1973) qui est un algorithme combinant la méthode de la bissection ou la méthode de la sécante avec l'interpolation de Lagrange inverse de second degré.

On construit trois suites de points a_k, b_k et c_k avec $f(a_k).f(b_k)<0$ et,

$$c_k=\frac{a_k+b_k}{2}, \text{ avec la méthode de la bissection}$$

$$c_k=a_k-\frac{a_k-b_k}{f(a_k)-f(b_k)}f(a_k), \text{ avec la méthode de la sécante}$$

A chaque itération, on évalue l'interpolation de Lagrange inverse, c'est-à-dire :

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

Pour $y=0$ soit,

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

Qui s'écrit encore sous la forme :

$$x_k=b_k+\frac{P_k}{Q_k}$$

Les quantités P_k et Q_k sont déterminées par les relations :

$$P_k=(R_k-T_k)S_kT_k(c_k-b_k)-(1-R_k)S_k(b_k-a_k)$$

et

$$Q_k=(T_k-1)(R_k-1)(S_k-1)$$

avec

$$R_k=\frac{f(b_k)}{f(c_k)},\quad S_k=\frac{f(b_k)}{f(a_k)} \text{ et } T_k=\frac{f(a_k)}{f(c_k)}$$

L'expression P_k/Q_k est un résidu qui diminue à chaque itération *k*, et par conséquent la convergence vers la solution de l'équation $f(x)=0$.

Soit les deux fonctions **`BrentBissection`** et **`BrentSecante`** utilisent l'interpolation inverse quadratique de Lagrange dans la résolution de l'équation non linéaire $f(x)=0$ ou sa solution existe dans l'intervalle [*a*, *b*].

```
function [x,k] = BrentBissection(f,a,b,varargin)
% f, nom de la fonction tel que f(x)=0
% a, b), intervalle de la solution de f(x)=0 (a < b)
% x, solution de l'équation f(x)=0
% k, nombre d'intervalles subdivisés
if nargin<3,error('à la limite 3 arguments sont nécessaires'),end
if ~(b>a),error('Attention, il faut que a < b'),end
eps=0.000001; % Définition de la précision souhaitée
for k=1:1000
  c=(a+b)/2;
    R=f(b,varargin{:})/f(c,varargin{:});
      S=f(b,varargin{:})/f(a,varargin{:});
        T=f(a,varargin{:})/f(c,varargin{:});
    P=(T*(R-T)*(c-b)-(1-R)*(b-a))*S;
   Q=(T-1)*(R-1)*(S-1);  x(k)=b+(P/Q);
   if abs(f(x(k),varargin{:})) <= eps
       break
   end
    if f(a,varargin{:})*f(c,varargin{:})<0
      b=c;
    else
      a=c;
    end
end
 x=x(k);
end
```

```
function [x,k] = BrentSecante(f,a,b,varargin)
% Les Entrées :
% f, nom de la fonction tel que f(x)=0
% (a, b), intervalle de la solution de f(x)=0 (a < b)
% Les Sorties :
% x, solution de l'équation f(x)=0
% k, nombre d'itérations jusqu'à la convergence
if nargin<3,error('à la limite 3 arguments sont nécessaires'),end
if ~(b>a),error('Attention, il faut que a < b'),end
eps=0.000001; % Définition de la précision souhaitée
for k=1:1000
 c=(a*f(b)-b*f(a))/(f(b)-f(a));
  R=f(b)/f(c);
   S=f(b)/f(a);
    T=f(a)/f(c);
  P=(T*(R-T)*(c-b)-(1-R)*(b-a))*S;
 Q=(T-1)*(R-1)*(S-1);
x(k)=b+(P/Q);
   if abs(f(x(k))) <= eps
     break
   end
 if f(a)*f(c)<0
  b=c;
 else
  a=c;
 end
end
x=x(k);
end
```

A titre d'exemple, soit à chercher la racine de la fonction $f(x)=(5-x)e^{x}-3$ dans l'intervalle [4, 6], c'est-à-dire :

```
>> f=@(x) (5-x).*exp(x)-3;
>> [x,k] = BrentBissection(f,4,6)
x =
    4.9794
k =
    10
```

```
>> f=@(x) (5-x).*exp(x)-3;
>> [x,k] = BrentSecante(f,4,6)
x =
    4.9794
k =
    23
```

Remarquons bien qu'il y a une nette amélioration dans le nombre d'opérations effectués entre la méthode de la bissection (resp. Regula-Falsi) vue dans le chapitre précédent section 2 (resp. 6) et la méthode de de Van Wijngaarden-Dekker-Brent.

5. MATLAB et les polynômes

Soit le polynôme $P_n(x)$ de degré *n* :

$$P_n(x)=a_n x^n+a_{n-1}x^{n-1}+\cdots+a_2x^2+a_1x+a_0$$

Dans MATLAB le polynôme $P_n(x)$ est défini par ces coefficients a_n, a_{n-1}, a_{n-2}, ⋯, a_1, a_0 sous forme d'un tableau comme suit :

```
>> Pn=[an an-1 an-2 . . . a1 a0]
```

La fonction **roots(Pn)** permet de déterminer numériquement tous les racines du polynôme, soit par exemple :

$$P_3(x)=3x^3+2x^2+x+1$$

```
>> P3=[3 2 1 1]; % composantes ou coefficients du polynôme
>> roots(P3) % fonction de détermination des racines du polynôme
ans =
  -0.7839 + 0.0000i
   0.0586 + 0.6495i
   0.0586 - 0.6495i
```

Il est possible de construire un polynôme à partir de leurs racines à l'aide de la fonction **poly** qui renvoie le vecteur des coefficients du polynôme. Soit l'exemple :

```
>> r=[-2 2 1 3]
r =
    -2     2     1     3
>> P4=poly(r)
P4 =
    1    -4    -1    16   -12
```

C'est-à-dire, ce polynôme est :

$$P_4(x)=x^4-4x^3-x^2+16x-12$$

Pour calculer le polynôme à une valeur **X** (**X** peut être un vecteur ou tableau), il suffit d'utiliser la fonction **polyval** avec comme arguments, le vecteur des coefficients et la valeur de **X**, soit l'exemple :

```
>> X = [1.5 2.5 ; 4 5]
X =
    1.5000    2.5000
    4.0000    5.0000
>> P2X = polyval(P4,X)
P2X =
    1.3125   -1.6875
   36.0000  168.0000
```

Soit $Q_m(x)$ un polynôme de degré *m*, c'est-à-dire :

$$Q_m(x) = b_m x^m + b_{m-1} x^{m-1} + \cdots + b_2 x^2 + b_1 x + b_0$$

Le produit $P_n(x).Q_m(x)$ est un polynôme de degré *n*+*m*. Avec MATLAB ce produit s'effectue avec la fonction **conv** après avoir entré les deux vecteurs coefficients de chaque polynôme :

```
>> Qn=[an an-1 an-2 . . . a1 a0];
>> Qm=[bm bm-1 qm-2 . . . b1 b0];
>> PQnPlusm=conv(Pn,Qm)
```

Il est possible de diviser un polynôme par un autre, mais certaine division possède un reste. Si $D(x)$ est le résultat de la division de $P_n(x)/Q_m(x)$ et $R(x)$ le reste de la division alors :

$$P_n(x) = Q_m(x).D(x) + R(x)$$

La fonction **deconv** de permet de fournir les résultats de la division **D** et le reste **R** comme suit :

```
>> [D,R]=deconv(Pn,Qm)
```

La division s'effectue surtout quand $n \geq m$, dans le cas contraire, MATLAB vous donne **D=0** et par conséquent **R=Pn.**

Il est possible de dériver et intégrer un polynôme $P_n(x)$. La dérivation (resp. l'intégration) d'un polynôme de degré *n* est aussi un polynôme de degré inférieur *n*-1 (resp. supérieur *n*+1) se fait avec la fonction **polyder** (resp. **polyint**), soit :

```
>> Pn = [an an-1 an-2 . . . a1 a0];
>> diff2Pn = polyder(Pn)
diff2Pn =
    n*an  (n-1)*an-1  (n-2)*an-2  .  .  .  a1
>> int2Pn = polyint(Pn,k)
int2Pn =
    an/(n+1)  an-1/n  an-2/(n-1)  .  .  .  a1/2 a0 k
```

6. Interpolation de Newton, méthode des différences divisées

Le polynôme d'interpolation de Lagrange présente certaine élégance dans sa forme. Une fois établit, ce polynôme n'est pas flexible, c'est-à-dire l'ajout d'un point $\left(x_{n+1},\ y_{n+1}\right)$ par exemple nécessite la reconstitution du polynôme. Soit $f\left(x\right)$ une fonction infiniment dérivable sur un intervalle I = [a, b]. Etant donnée un polynôme d'interpolation $P_k\left(x\right)$ des $k+1$ points $\left\{\left(x_i,\ y_i=f\left(x_i\right)\right),\ i=0,\ 1,\ \ldots,\ k\right\}$. Le polynôme d'interpolation $P_{k+1}\left(x\right)$ des $k+2$ points $\left\{\left(x_i,\ y_i=f\left(x_i\right)\right),\ i=0,\ 1,\ \ldots,\ k,\ k+1\right\}$ peut être exprimé, par la relation récurrente :

$$P_{k+1}\left(x\right)=P_k\left(x\right)+C\left(x\right)$$

Le terme de correction $C\left(x\right)$ est un polynôme de degré $k+1$.

$$P_{k+1}\left(x_i\right)=P_k\left(x_i\right)+C\left(x_i\right)\ \Leftrightarrow\ y_i=y_i+C\left(x_i\right)\ \Rightarrow\ C\left(x_i\right)=0,\ \forall\ i=0,\ 1,\ \ldots,\ k$$

et,

$$P_{n+1}\left(x_{k+1}\right)=y_{k+1}\equiv f\left(x_{k+1}\right)$$

par conséquent, le polynôme $C\left(x\right)$ peut être considéré comme :

$$C\left(x\right)=a_{k+1}\prod_{j=0}^{k}\left(x-x_j\right)$$

soit finalement,

$$P_{k+1}\left(x\right)=P_k\left(x\right)+a_{k+1}\prod_{j=0}^{k}\left(x-x_j\right)$$

Il est trivial que si $k=0$ (polynôme constant), $P_0\left(x\right)=a_0\ \Rightarrow\ a_0=y_0$.

D'après la formule précédente, le polynôme $P_1\left(x\right)$ est :

$$P_1\left(x\right)=P_0\left(x\right)+a_1\prod_{j=0}^{0}\left(x-x_j\right)=a_0+a_1\left(x-x_0\right)$$

$$P_1\left(x_1\right)=y_1=a_0+a_1\left(x_1-x_0\right)\ \Rightarrow\ a_1=\frac{y_1-a_0}{x_1-x_0}=\frac{y_1-y_0}{x_1-x_0}=\frac{f\left(x_1\right)-f\left(x_0\right)}{x_1-x_0}$$

De même le polynôme $P_2\left(x\right)$ est obtenu par :

$$P_2\left(x\right)=P_1\left(x\right)+a_2\left(x-x_0\right)\left(x-x_1\right)=a_0+a_1\left(x-x_0\right)+a_2\left(x-x_0\right)\left(x-x_1\right)$$

$$P_2\left(x_2\right)=y_2\ \Leftrightarrow\ y_2=a_0+a_1\left(x_2-x_0\right)+a_2\left(x_2-x_0\right)\left(x_2-x_1\right)\ \Rightarrow\ a_2=\frac{y_2-a_0-a_1\left(x_2-x_0\right)}{\left(x_2-x_0\right)\left(x_2-x_1\right)}$$

ou bien,

$$a_2=\frac{\dfrac{y_2-y_1}{x_2-x_1}-\dfrac{y_1-y_0}{x_1-x_0}}{x_2-x_0}=\frac{\dfrac{f\left(x_2\right)-f\left(x_1\right)}{x_2-x_1}-\dfrac{f\left(x_1\right)-f\left(x_0\right)}{x_1-x_0}}{x_2-x_0}$$

notons par :

$$f[x_i]=f\left(x_i\right),\ \ f[x_0,\ x_1]=\frac{f[x_1]-f[x_0]}{x_1-x_0}\ \text{ et }\ f[x_0,\ x_1,\ x_2]=\frac{f[x_2,\ x_1]-f[x_1,\ x_0]}{x_2-x_0}$$

alors,

$$a_0=f[x_0],\ a_1=f[x_1,\ x_2],\ a_2=f[x_0,\ x_1,\ x_n],\ \ldots,\ a_n=f[x_0,\ x_1,\ x_2,\ \ldots,\ x_n]$$

par conséquent,

$$P_0(x) = f[x_0], \quad P_1(x) = f[x_0] + f[x_0,\ x_1](x_1 - x_0)$$

$$P_2(x) = f[x_0] + f[x_0,\ x_1](x_1 - x_0) + f[x_0,\ x_1,\ x_2](x - x_1)(x - x_0)$$

$$\cdots\cdots\cdots\cdots$$

$$P_n(x) = f[x_0] + f[x_0,\ x_1](x - x_0) + f[x_0,\ x_1,\ x_2](x - x_0)(x - x_1) + \cdots$$
$$+ f[x_0,\ x_1,\ x_2,\ \ldots,\ x_{n-1},\ x_n](x - x_0)\cdots(x - x_{n-1})$$

Qui s'écrit aussi sous la forme compacte :

$$P_n(x) = \sum_{k=0}^{n}\left\{ f[x_0,\ x_1,\ x_2,\ \ldots,\ x_k]\prod_{j=1}^{k}(x - x_{j-1})\right\}$$

Les quantités, $f[x_0]$, $f[x_0,\ x_1]$, $f[x_0,\ x_1,\ x_2]$, $\cdots$ et $f[x_0,\ x_1,\ x_2,\ \ldots,\ x_n]$ sont appelées les différences divisées de la fonctions $f(x)$ au valeurs $(x_0,\ x_1,\ x_2,\ \ldots,\ x_n)$.

Tab. 4.1 : Différences divisées.

x_i	$f[x_i]$	$f[x_{i-1},\ x_i]$	$f[x_{i-2},\ x_{i-1},\ x_i]$	$f[x_{i-3},\ x_{i-2},\ x_{i-1},\ x_i]$
x_0	$f[x_0] \equiv a_0$			
		$f[x_0,\ x_1] \equiv a_1$		
x_1	$f[x_1]$		$f[x_0,\ x_1,\ x_2] \equiv a_2$	
		$f[x_1,\ x_2]$		$f[x_0,\ x_1,\ x_2,\ x_3] \equiv a_3$
x_2	$f[x_2]$		$f[x_1,\ x_2,\ x_3]$	
		$f[x_2,\ x_3]$		
x_3	$f[x_3]$			

Les expressions généralisées des différences divisées sont :

$$f[x_i] = f(x_i), \quad f[x_i,\ x_j] = \frac{f[x_i] - f[x_j]}{x_i - x_j} = f[x_j,\ x_i]$$

$$f[x_i,\ x_j,\ x_k] = \frac{f[x_i,\ x_j] - f[x_j,\ x_k]}{x_i - x_k} = \frac{f[x_j,\ x_i] - f[x_k,\ x_j]}{x_i - x_k} = f[x_k,\ x_j,\ x_i]$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$f[x_0,\ x_1,\ \ldots,\ x_{n-1},\ x_n] = \frac{f[x_1,\ x_2,\ \ldots,\ x_{n-1},\ x_n] - f[x_0,\ x_1,\ \ldots,\ x_{n-2},\ x_{n-1}]}{x_n - x_0}$$

Vu l'unicité du polynôme d'interpolation, la comparaison avec l'expression du polynôme d'interpolation de Lagrange (Phillips et Taylor, 1996), $f[x_0,\ x_1,\ \ldots,\ x_{n-1},\ x_n]$ est peut-être exprimée par :

$$f[x_0,\ x_1,\ \ldots,\ x_{n-1},\ x_n] = \sum_{i=0}^{n}\frac{f(x_i)}{\prod_{\substack{j=0\\ j\neq i}}^{n}(x_i - x_j)}$$

A titre d'exemple,

$$f[x_0,\ x_1,\ x_2] = \frac{f(x_0)}{(x_0 - x_1)(x_0 - x_2)} + \frac{f(x_1)}{(x_1 - x_0)(x_1 - x_2)} + \frac{f(x_2)}{(x_2 - x_0)(x_2 - x_1)}$$

Le tableau ci-dessus (Tab. 4.1) montre comment se fait le calcul pratique des différences divisées et comment les coefficients $a_0,\ a_1,\ a_2,\ \ldots,\ a_n$ peuvent être choisis.

Soit l'exemple,

x_i	1	4	6	5
y_i	0	1.386294	1.791759	1.609438

le polynôme d'interpolation est :

$$P_3(x) = a_0 + a_1(x-1) + a_2(x-4)(x-1) + a_3(x-6)(x-4)(x-1)$$

Les différences divisées de sont :

$$f[x_0] = 0,\quad f[x_0,\ x_1] = \frac{1.386294 - 0}{4-1} = 0.4620981$$

$$f[x_1,\ x_2] = \frac{1.791759 - 1.386294}{6-4} = 0.2027326,\ f[x_2,\ x_3] = \frac{1.609438 - 1.791759}{5-6} = 0.1823216$$

$$f[x_0,\ x_1,\ x_2] = \frac{0.2027326 - 0.4620981}{6-1} = -0.05187311$$

$$f[x_1,\ x_2,\ x_3] = \frac{0.1823216 - 0.2027326}{5-4} = -0.02041100$$

$$f[x_0,\ x_1,\ x_2] = \frac{-0.02041100 - (-0.05187311)}{5-1} = 0.007865529$$

Soit le tableau des différences divisées (Tab. 4.2).

Tab. 4.2 : Différences divisées.

x_i	$f[x_i]$	$f[x_{i-1},\ x_i]$	$f[x_{i-2},\ x_{i-1},\ x_i]$	$f[x_{i-3},\ x_{i-2},\ x_{i-1},\ x_i]$
1	0.000000			
4	1.386294	0.4620981		
6	1.791759	0.2027326	-0.05187311	
5	1.609438	0.1823216	-0.02041100	0.007865529

L'expression du polynôme d'interpolation est donc :

$$P_3(x) = 0 + 0.4620980\,(x-1) - 0.0518731\,(x-4)(x-1) + 0.0078654\,(x-6)(x-4)(x-1)$$

Par exemple, l'interpolation de $P_3(2) = 0.6287674$.

La fonction **`InterpNewtonForwardDd`**, permet de construire le tableau des différences divisées avant à partir des vecteurs donnée **`x`** et **`y`** et d'estimer le vecteur **`Y`** par interpolation pour un vecteur donnée **`X`**.

```
function [Y,TableDividedDifferences] = InterpNewtonForwardDd(x,y,X)
% Entrées du modèle d'interpolation: x = [x0 ... xN] et y = [y0 ... yN]
% X = Vecteur donné, Y = Vecteur interpolé correspond à X
% TableDividedDifferences: Tableau des différences divisées avant
n = length(x);
if numel(y)~=n
```

```
 error('les dimensions de x et de y sont différentes ')
end
for indice=1:numel(X)
a(1) = y(1);
for k = 1 : n - 1
   d(k, 1) = (y(k+1) - y(k))/(x(k+1) - x(k));
end
for j = 2 : n - 1
   for k = 1 : n - j
      d(k, j) = (d(k+1, j - 1) - d(k, j - 1))/(x(k+j) - x(k));
   end
end
for j = 2 : n
  a(j) = d(1, j-1);
end
  Df(1) = 1; c(1) = a(1);
for j = 2 : n
 Df(j)=(X(indice) - x(j-1)) .* Df(j-1); c(j,indice) = a(j) .* Df(j);
end
end
d(numel(y),1:(numel(y)-1))=0;
TableDividedDifferences=[y' d];
Y=sum(c);
end
```

Soit les données du modèle d'interpolation :

```
x=[1 4 6 5]; y=[0 1.386294 1.791759 1.609438]; X=[1.5 3 5.3];
```

L'exécution de la fonction conduit à :

```
>> [Yinterpole,TableauDd]=InterpNewtonForwardDd(x,y,X)
Yinterpole =
    0.3401    1.0751    1.6663
TableauDd =
         0    0.4621   -0.0519    0.0079
    1.3863    0.2027   -0.0204         0
    1.7918    0.1823         0         0
    1.6094         0         0         0
```

L'expression précédente du polynôme $P_n(x)$, elle peut être écrite sous la forme compacte suivante :

$$P_n(x)=\sum_{k=0}^{n}\left\{f[x_0,\ x_1,\ x_2,\ \ldots,\ x_k]\prod_{j=1}^{k}\left(x-x_{j-1}\right)\right\}$$

Qui prend en considération le parcours avant des couples (x_i, f_i), $i=0,\ 1,\ 2,\ \ldots,\ n$ elle est appelée la *formule d'interpolation de Newton en différences divisées avant*. Si par contre le parcours prise en considération des couples est arrière, c'est-à-dire (x_i, f_i), $i=n,\ n-1,\ n-2,\ \ldots,\ 0$, alors le polynôme $P_n(x)$ peut être exprimé par :

$$P_n(x)=f[x_n]+f[x_n,\ x_{n-1}](x-x_n)+f[x_n,\ x_{n-1},\ x_{n-1}](x-x_n)(x-x_{n-1})+\cdots$$
$$+f[x_n,\ x_{n-1},\ x_{n-2},\ \ldots,\ x_1,\ x_0](x-x_n)(x-x_{n-1})\cdots(x-x_1)$$

et s'écrit sous la forme compacte,

$$P_n(x)=\sum_{k=0}^{n}\left\{f[x_{n-k},\ x_{n-k+1},\ x_{n-k+2},\ \ldots,\ x_n]\prod_{j=1}^{k}\left(x-x_{n-j+1}\right)\right\}$$

C'est la *formule d'interpolation de Newton en différences divisées arrières*.

Dans le cas où les abscisses $\left(x_i,\ i=0,\ 1,\ \ldots,\ n\right)$ sont équidistants (nœuds uniformes), c'est-à-dire :

$$x_i=x_{i-1}+h=x_0+ih,\ \ \forall\ i=1,\ 2,\ \ldots,\ n\ \text{ et }\ h=\text{Cte}$$

Dans la formule d'interpolation de Newton en différences divisées avant, le terme :

$$f[x_0,\ x_1]=\frac{f(x_1)-f(x_0)}{x_1-x_0}\equiv\frac{f_1-f_0}{x_1-x_0}=\frac{\Delta f_0}{h}\ \ \Rightarrow\ \ \Delta f_0=hf[x_0,\ x_1]$$

alors,

$$\Delta^2 f_0=\Delta f_1-\Delta f_0=hf[x_1,\ x_2]-hf[x_0,\ x_1]=h\left\{f[x_1,\ x_2]-f[x_0,\ x_1]\right\}$$

$$\Delta^2 f_0=h\cdot 2h\frac{f[x_1,\ x_2]-f[x_0,\ x_1]}{2h}=2h^2\,\frac{f[x_1,\ x_2]-f[x_0,\ x_1]}{x_2-x_0}=2h^2 f[x_0,\ x_1,\ x_2]$$

Suivant le même raisonnement (Whittaker et Robinson, 1944),

$$\Delta^k f_i=k!\cdot h^k f[x_i,\ x_{i+1},\ x_{i+2},\ \ldots,\ x_{i+k}]\ \ \Rightarrow\ \ f[x_i,\ x_{i+1},\ x_{i+2},\ \ldots,\ x_{i+k}]=\frac{1}{k!}\frac{\Delta^k f_i}{h^k}$$

Pour i = 0,

$$f[x_0,\ x_1,\ x_2,\ \ldots,\ x_k]=\frac{1}{k!}\frac{\Delta^k f_0}{h^k}$$

La formule d'interpolation de Newton en différences divisées avant devient dans ce cas :

$$P_n(x)=\sum_{k=0}^{n}\left\{\frac{1}{k!}\frac{\Delta^k f_0}{h^k}\prod_{j=1}^{k}\left(x-x_{j-1}\right)\right\}$$

Soit le changement de variable $x=x_0+sh\ \ \Rightarrow\ \ x-x_{j-1}=\left(x_0+sh\right)-\left(x_0+\left(j-1\right)h\right)=\left(s-j+1\right)$.

par conséquent,

$$P_n(x)=\sum_{k=0}^{n}\left\{\frac{1}{k!}\frac{\Delta^k f_0}{h^k}\prod_{j=1}^{k}(s-j+1)h\right\}=\sum_{k=0}^{n}\left\{\frac{1}{k!}\frac{\Delta^k f_0}{h^k}h^k\prod_{j=1}^{k}(s-j+1)\right\}$$

ou,

$$P_n(x)=\sum_{k=0}^{n}\frac{s(s-1)(s-2)\cdots(s-k+1)}{k!}\Delta^k f_0$$

or,

$$\frac{s(s-1)(s-2)\cdots(s-k+1)}{k!}\equiv\binom{s}{k}$$

alors,

$$P_n(x)=\sum_{k=0}^{n}\binom{s}{k}\Delta^k f_0\ \text{ notée par }\ p_n^f(s)$$

$p_n^f(s)$ est la formule d'interpolation de *Gregory-Newton progressive* (Whittaker et Robinson, 1944).

De même, dans la formule d'interpolation de Newton en différences divisées arrières, le terme :

$$f[x_n,\ x_{n-1}]=\frac{f(x_n)-f(x_{n-1})}{x_n-x_{n-1}}\equiv\frac{f_n-f_{n-1}}{x_n-x_{n-1}}=\frac{\nabla f_n}{h}\ \Rightarrow\ \nabla f_n=hf[x_n,\ x_{n-1}]$$

alors,

$$\nabla^2 f_n=\nabla f_n-\nabla f_{n-1}=hf[x_n,\ x_{n-1}]-hf[x_{n-1},\ x_{n-2}]=h\{f[x_n,\ x_{n-1}]-f[x_{n-1},\ x_{n-2}]\}$$

$$\nabla^2 f_0=h\cdot 2h\frac{f[x_n,\ x_{n-1}]-f[x_{n-1},\ x_{n-2}]}{2h}=2h^2\frac{f[x_n,\ x_{n-1}]-f[x_{n-1},\ x_{n-2}]}{x_n-x_{n-2}}=2h^2 f[x_n,\ x_{n-1},\ x_{n-2}]$$

Suivant le même raisonnement (Whittaker et Robinson, 1944),

$$\nabla^k f_i=k!\cdot h^k f[x_{i+k},\ x_{i+k-1},\ x_{i+k-2},\ \ldots,\ x_i]\ \Rightarrow\ f[x_{i+k},\ x_{i+k-1},\ x_{i+k-2},\ \ldots,\ x_i]=\frac{1}{k!}\frac{\nabla^k f_i}{h^k}$$

Pour $i = 0$ et $k = n$,

$$f[x_n,\ x_{n-1},\ x_{n-2},\ \ldots,\ x_0]=\frac{1}{k!}\frac{\nabla^k f_n}{h^k}$$

La formule d'interpolation de Newton en différences divisées arrières devient :

$$P_n(x)=\sum_{k=0}^{n}\left\{\frac{1}{k!}\frac{\nabla^k f_n}{h^k}\prod_{j=1}^{k}(x-x_{n-j+1})\right\}$$

soit le changement de variable $x=x_n+sh$, alors

$$x-x_{n-j+1}=(x_0+nh+sh)-(x_0+(n-j+1)h)=(s+j-1)h$$

par conséquent,

$$P_n(x)=\sum_{k=0}^{n}\left\{\frac{1}{k!}\frac{\nabla^k f_n}{h^k}\prod_{j=1}^{k}(s+j-1)h\right\}=\sum_{k=0}^{n}\left\{\frac{1}{k!}\frac{\nabla^k f_n}{h^k}h^k\prod_{j=1}^{k}(s+j-1)\right\}$$

ou,

$$P_n(x)=\sum_{k=0}^{n}\frac{s(s+1)(s+2)\cdots(s+k-1)}{k!}\nabla^k f_n$$

on pose $s=-v$, alors,

$$\frac{-v(-v+1)(-v+2)\cdots(-v+k-1)}{k!}=(-1)^k\frac{v(v-1)(v-2)\cdots(v-k+1)}{k!}\equiv(-1)^k\binom{v}{k}$$

soit,

$$P_n(x)=\sum_{k=0}^{n}(-1)^k\binom{-s}{k}\nabla^k f_n\ \text{ notée par }\ p_n^b(s)$$

$p_n^b(s)$ est la formule d'interpolation de *Gregory-Newton régressive* (Zwillinger, 2003).

7. Interpolation inverse

Etant donnée l'ensemble de points de coordonnées $\{(x_i,\ y_i),\ i=0,\ 1,\ \ldots,\ n\}$ avec x_i, $n+1$ valeurs distinctes. Il est possible de construire avec la méthode d'interpolation de Lagrange ou de différences divisées de Newton un polynôme d'interpolation inverse et d'estimer x à partir de y comme suit :

$$P_n(y)=\sum_{j=0}^{n}x_j L_j(y)\ \text{ avec }\ L_j(y)=\prod_{k=0,\ k\neq j}^{n}\frac{(y-y_k)}{(y_j-y_k)}$$

$$P_n(y)=f^{-1}[y_0]+f^{-1}[y_0,\ y_1](y-y_0)+f^{-1}[y_0,\ y_1,\ y_2](y-y_0)(y-y_1)+\cdots$$
$$+f^{-1}[y_0,\ y_1,\ y_2,\ \ldots,\ y_n](y-y_0)\cdots(y-y_{n-1})$$

8. Existence et unicité du polynôme d'interpolation

Trois méthodes d'interpolation polynomiale ont été traitée (Vandermonde, Lagrange et différences divisées de Newton) afin donc de déterminer le polynôme $P_n(x)$ de degré *n* qui passe par $n+1$ points de coordonnées $\{(x_i,\ y_i),\ i=0,\ 1,\ \ldots,\ n\}$, ceci permet de justifier l'existence du polynôme d'interpolation.

Soit $P_n(x)$ et $Q_n(x)$ deux polynômes d'interpolation de l'ensemble des points $\{(x_i,\ y_i),\ i=0,\ 1,\ \ldots,\ n\}$ c'est-à-dire :

$$\forall\ i=0,\ 1,\ \ldots,\ n,\ P_n(x_i)=y_i\ \text{ et }\ Q_n(x_i)=y_i\ \Rightarrow\ P_n(x)\equiv Q_n(x)$$

Ce qui permet de dire que le polynôme d'interpolation est unique et quel que soit la méthodologie utilisée.

9. Erreur d'interpolation

Avant de donner une estimation de l'erreur, soit le théorème suivant :

$$f(\alpha_i)=0,\ \forall\ i=0,\ 1,\ \ldots,\ n,\ n+1\ \Rightarrow\ \exists\ \xi_j\ \text{ tel que }\ f'(\xi_j)=0,\ \forall\ j=0,\ 1,\ \ldots,\ n$$

Pour démontrer ce théorème, il suffit d'appliquer le théorème de Rolle entre deux zéros consécutifs de $f(x)$. Par conséquent, si en plus la fonction $f(x)$ est $n+1$ fois dérivable il existe au moins ζ tel que $f^{(n+1)}(\zeta)=0$.

Soit $x_0,\ x_1,\ \ldots,\ x_n$ des valeurs réelles distinctes (appelées aussi nœuds) et $y_i=f(x_i),\ \forall\ i=0,\ 1,\ \ldots,\ n$, le polynôme d'interpolation de $P_n(x)$ est exprimé par la *méthode de Lagrange* :

$$P_n(x)=\sum_{j=0}^{n}f(x_j)L_j(x)$$

Pour un réel $x\neq x_i,\ \forall\ i=0,\ 1,\ \ldots,\ n$, l'erreur d'interpolation est :

$$E(x)=f(x)-P_n(x)=f(x)-\sum_{j=0}^{n}f(x_j)L_j(x)$$

il est clair,

$$\text{si } x=x_i \text{ alors, } E(x_i)=f(x_i)-P_n(x_i)=0$$

considérant,

$$E(x)=a(x-x_0)(x-x_1)\cdots(x-x_n)\equiv a\psi(x)$$

et la fonction,

$$G(x)=E(x)-\frac{\psi(x)}{\psi(t)}E(t)$$

$$G(x_i)=E(x_i)-\frac{\psi(x_i)}{\psi(t)}E(t)=0-0=0,\ \forall\ i=0,\ 1,\ \ldots,\ n$$

$G(x)$ est $n+1$ fois dérivable car $E(x)$ est $n+1$ fois dérivable, par conséquent et suivant le théorème précédent il existe au moins ξ de l'intervalle $I=[\inf x,\ \sup x]$ tel que, $G^{(n+1)}(\xi)=0$.

avec,

$$\inf x=\min(x_0,\ x_1,\ \ldots,\ x_n)\ \text{ et }\ \sup x=\max(x_0,\ x_1,\ \ldots,\ x_n)$$

or,

$$G^{(n+1)}(x)=E^{(n+1)}(x)-\frac{\psi^{(n+1)}(x)}{\psi(t)}E(t)=E^{(n+1)}(x)-\frac{(n+1)!}{\psi(t)}E(t)$$

alors,

$$E^{(n+1)}(\xi)-\frac{(n+1)!}{\psi(t)}E(t)=0\ \Leftrightarrow\ f^{(n+1)}(\xi)-\frac{(n+1)!}{\psi(t)}E(t)=0$$

soit,

$$E(t)=\frac{\psi(t)}{(n+1)!}f^{(n+1)}(\xi)$$

Voyons les choses d'une autre façon. Soit t un réel différent des nœuds $x_0,\ x_1,\ \ldots,\ x_n$, le polynôme d'interpolation par la méthode des différences divisées est :

$$P_{n+1}(x)=f[x_0]+f[x_0,\ x_1](x-x_0)+f[x_0,\ x_1,\ x_2](x-x_0)(x-x_1)+\cdots$$
$$+f[x_0,\ x_1,\ x_2,\ \ldots,\ x_n,\ t](x-x_0)\cdots(x-x_n)$$

c'est-à-dire,

$$P_{n+1}(x)=P_n(x)+f[x_0,\ x_1,\ x_2,\ \ldots,\ x_n,\ t]\psi(x)$$

$P_n(x)$, est le polynôme d'interpolation des nœuds $x_0,\ x_1,\ \ldots,\ x_n$ et $P_{n+1}(t)=f(t)$, alors :

$$f(t)=P_n(t)+f[x_0,\ x_1,\ x_2,\ \ldots,\ x_n,\ t]\psi(t)$$

soit,

$$f(t)-P_n(t)=f[x_0,\ x_1,\ x_2,\ \ldots,\ x_n,\ t]\psi(t)\equiv E(t)$$

Par comparaison à la formule précédente de l'erreur,

$$f[x_0,\ x_1,\ x_2,\ \ldots,\ x_n,\ t]=\frac{f^{(n+1)}(\xi)}{(n+1)!}$$

Dans le cas d'une interpolation linéaire entre les deux points $(x_1,\ f(x_1))$ et $(x_2,\ f(x_2))$, l'erreur maximale est explicitée par :

$$E_{\max}=\frac{1}{2}\left(\frac{x_1-x_2}{2}\right)^2\|f''\|_\infty=\frac{1}{8}(x_1-x_2)^2\|f''\|_\infty$$

Si l'interpolation polynomiale est de type de Hermite, l'erreur d'interpolation est exprimée par :

$$f(x)-H_{2n+1}(x)=\frac{(\psi(x))^2}{(2n+2)!}f^{(2n+2)}(\zeta_x)$$

9.1. Distribution uniforme des nœuds

L'intervalle I = [a, b] est subdivisé uniformément (Fig. 4.2) en n parties avec un pas de discrétisation h, de façon que :

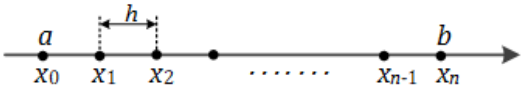


Fig. 4.2 : Discrétisation à pas constant.

$$x_0 = a,\ \ x_n = b \ \text{ et } \ h = \frac{b-a}{n}$$

par conséquent,

$$x_0 = a,\ \ x_1 = a+h,\ \ x_2 = x_1 + h = a+h+h = a+2h,$$

$$x_3 = x_2 + h = a+2h+h = a+3h, \ldots,\ \ x_k = a+kh, \ldots,\ x_{n-1} = a+(n-1)h,\ \ x_n = b$$

soit,

$$|\psi(x)| = \left|\prod_{k=0}^{n}(x-x_k)\right| = \prod_{k=0}^{n}|x-x_k|$$

sachant que,

$$|(x-x_0)(x-x_1)| = |(x-a)(x-a-h)| \le \left(\frac{h}{2}\right)^2$$

et,

$$|x-x_2| \le 2h,\ \ |x-x_3| \le 3h,\ \ldots,\ |x-x_k| \le kh,\ \ldots,\ |x-x_n| \le nh$$

alors,

$$|\psi(x)| \le \left(\frac{h}{2}\right)^2 \prod_{k=2}^{n}|x-x_k| \le \left(\frac{h}{2}\right)^2 \prod_{k=2}^{n}(kh) = \frac{h}{4}\prod_{k=1}^{n}(kh) = \frac{h}{4}n!h^n = \frac{n!}{4}h^{n+1}$$

Puisque l'erreur d'interpolation est :

$$E(x) = \frac{\psi(x)}{(n+1)!}f^{(n+1)}(\zeta)$$

alors,

$$|E(x)| \le \frac{n!}{4(n+1)!}h^{n+1}\left\|f^{(n+1)}\right\|_\infty = \frac{h^{n+1}}{4(n+1)}\left\|f^{(n+1)}\right\|_\infty = \frac{1}{4(n+1)}\left(\frac{b-a}{n}\right)^{n+1}\left\|f^{(n+1)}\right\|_\infty$$

ou,

$$|E(x)| \le \frac{1}{4(n+1)}\left(\frac{b-a}{n}\right)^{n+1}\left\|f^{(n+1)}\right\|_\infty$$

9.2. Nœuds de Tchebychev

Le polynôme de Tchebychev de première espèce $T_n(x)$ de degré n est exprimé pour x de l'intervalle [-1, 1] (Weisstein, 2003 et Zwillinger, 2003) par :

$$T_n(x) = \cos(n\arccos x) = 2^{n-1}\prod_{j=1}^{n}\left(x - \cos\left(\frac{2j-1}{n}\frac{\pi}{2}\right)\right) \ \Rightarrow\ T_{n-1}(x) = 2^{n-2}\prod_{j=1}^{n-1}\left(x - \cos\left(\frac{2j-1}{n-1}\frac{\pi}{2}\right)\right)$$

$x_j = \cos\left(\pi(2j-1)/2(n-1)\right)$, pour $j = 1,\ 2,\ \ldots,\ n-1$ sont les $n-1$ racines du polynôme de Tchebychev $T_{n-1}(x)$, en plus $\forall j = 1,\ 2,\ \ldots,\ n-1,\ x_j \ne \pm 1$ les nœuds extrêmes de l'intervalle [-1, 1]. Soit $x_0 = +1$ et $x_n = -1$, donc :

$$T_{n-1}(x) = \frac{2^{n-2}}{(x+1)(x-1)}\left(\prod_{j=1}^{n-1}(x-x_j)\right)(x+1)(x-1) = \frac{2^{n-2}}{(x+1)(x-1)}\prod_{j=0}^{n}(x-x_j) \equiv \frac{2^{n-2}}{(x+1)(x-1)}\psi(x)$$

ou bien,

$$\psi(x) = 2^{2-n}(x+1)(x-1)T_{n-1}(x)$$

pour $-1 \le x \le 1$,

$$|\psi(x)| \le 2^{2-n}$$

alors,

$$\left|E(x)\right| \le \frac{\left\|f^{(n+1)}\right\|_\infty}{2^{n-2}(n+1)!}$$

En déduit donc les nœuds dits de Tchebychev de type I,

$$x_j = \begin{cases} \cos\left(\dfrac{j}{n}\pi\right) \text{ si } j=0, \text{ et } n \\ \cos\left(\dfrac{2j-1}{n-1}\dfrac{\pi}{2}\right), \ \forall\ j=1,\ 2,\ \ldots,\ n-1 \end{cases}$$

Le polynôme de Tchebychev de deuxième espèce $U_n(x)$ de degré *n* est exprimé pour *x* de l'intervalle [-1, 1] (Weisstein, 2003 et Zwillinger, 2003) par :

$$U_n(x) = \frac{\sin\left((n+1)\arccos x\right)}{\sqrt{1-x^2}} = 2^n \prod_{j=1}^{n}\left(x - \cos\left(\frac{j}{n+1}\pi\right)\right)$$

par conséquent $U_{n-1}(x)$ est exprimé par :

$$U_{n-1}(x) = \frac{\sin\left(n\arccos x\right)}{\sqrt{1-x^2}} = 2^{n-1} \prod_{j=1}^{n-1}\left(x - \cos\left(\frac{j}{n}\pi\right)\right)$$

$x_j = \cos(\pi j/n)$, pour $j=1,\ 2,\ \ldots,\ n-1$, les $n-1$ racines du polynôme de Tchebychev $U_{n-1}(x)$ en plus, $j=0,\ x_0=1$ et $j=n,\ x_n=-1$ les nœuds extrêmes de l'intervalle [-1, 1], alors :

$$U_{n-1}(x) = \frac{2^{n-1}}{(x+1)(x-1)} \prod_{j=0}^{n}\left(x - \cos\left(\frac{j}{n}\pi\right)\right) \equiv \frac{2^{n-1}}{(x+1)(x-1)}\psi(x)$$

ou bien,

$$\psi(x) = 2^{1-n}(x+1)(x-1)U_{n-1}(x)$$

pour $-1 \le x \le 1$,

$$\left|\psi(x)\right| \le 2^{1-n} \ \Rightarrow\ \left|E(x)\right| \le \frac{\left\|f^{(n+1)}\right\|_\infty}{2^{n-1}(n+1)!}$$

En déduit donc les nœuds dits de Tchebychev de type II :

$$x_j = \cos\left(\frac{j}{n}\pi\right), \ \forall\ j=0,\ 1,\ \ldots,\ n$$

Considérant selon les trois cas précédents la quantité $\omega(n)$ qui peut être déduite de l'inégalité :

$$\frac{\left|E(x)\right|}{\left\|f^{(n+1)}\right\|_\infty} \le \omega(n)$$

c'est-à-dire, quand $-1 \le x \le 1$,

$$\omega_u(n) = \frac{1}{4(n+1)}\left(\frac{2}{n}\right)^{n+1} \text{ si les nœuds sont uniformes}$$

$$\omega_{TI}(n)=\frac{1}{2^{n-2}(n+1)!} \quad \text{si les nœuds sont de type I de Tchebychev}$$

$$\omega_{TII}(n)=\frac{1}{2^{n-1}(n+1)!} \quad \text{si les nœuds sont de type II de Tchebychev}$$

Remarquons que,

$$\omega_{TII}(n)\le\omega_{TI}(n)\le\omega_{u}(n)$$

Ce qui permet de dire que les nœuds de Tchebychev permettent d'avoir un polynôme d'interpolation meilleur si les nœuds sont uniformes.

Dans le cas où $a\le x\le b$, les expressions analytiques des nœuds de Tchebychev sont obtenues par une transformation affine et sont exprimés par :

$$x_j=\begin{cases}\dfrac{b-a}{2}\cos\left(\dfrac{j}{n}\pi\right)+\dfrac{a+b}{2} & \text{si } j=0,\ \text{et } n\\ \dfrac{b-a}{2}\cos\left(\dfrac{2j-1}{n-1}\dfrac{\pi}{2}\right)+\dfrac{a+b}{2} & \text{si } j=1,\ 2,\ \ldots,\ n-1\end{cases} \qquad \text{pour le type I}$$

$$x_j=\frac{b-a}{2}\cos\left(\frac{j}{n}\pi\right)+\frac{a+b}{2} \quad \forall\ j=0,\ 1,\ 2,\ \ldots,\ n \qquad \text{pour le type II}$$

9.3. Phénomène de Runge

Le phénomène de Runge (Demailly, 2006) se manifeste dans le contexte de l'interpolation polynomiale. Avec certaines fonctions, ou l'augmentation du nombre n de points d'interpolation ne constitue pas nécessairement une bonne stratégie d'approximation. Pour étudier cette question, le mathématicien allemand Carl Runge découvrit, en 1901, un résultat contraire à l'intuition : il existe des configurations où l'écart maximal entre la fonction et son interpolation augmente indéfiniment avec n.

La fonction $f(x)$ de Runge est donnée par :

$$f(x)=\frac{1}{1+25x^2}$$

Considérant le cas des nœuds x_0, x_1, …, x_n distribués uniformément. Le polynôme d'interpolation de Lagrange dans ce cas est exprimé par :

$$P_n(x)=\sum_{j=0}^{n}f(x_j)L_j(x) \quad \text{avec} \quad L_j(x)=\prod_{k=0,\ k\neq j}^{n}\frac{(x-x_k)}{(x_j-x_k)}$$

Soit le script MATLAB suivant qui sert à tracer le graph du polynôme d'interpolation.

```
f=@(x)1./(1+25*x.*x);
a = -2 ; b = -a; % Définition de l'intervalle de travaille
X=a:0.01:b; fx=f(X);
plot(X,fx,'-b')
hold on
n=4; % Choix du degré du polynôme d'interpolation
h=(a-b)/n; % Noeuds Uniformes
xu=a:h:b; yu=f(xu); plot(xu,yu,'or',xu,yu,'*b');
```

```
for i=1:numel(X)
    YU(i)=Interp2Newton(xu,yu,X(i));
end
plot(X,YU,'-r')
```

Pour n = 4, 6, 10 et 15 ce script permet d'avoir les graphes (Fig. 4.3).

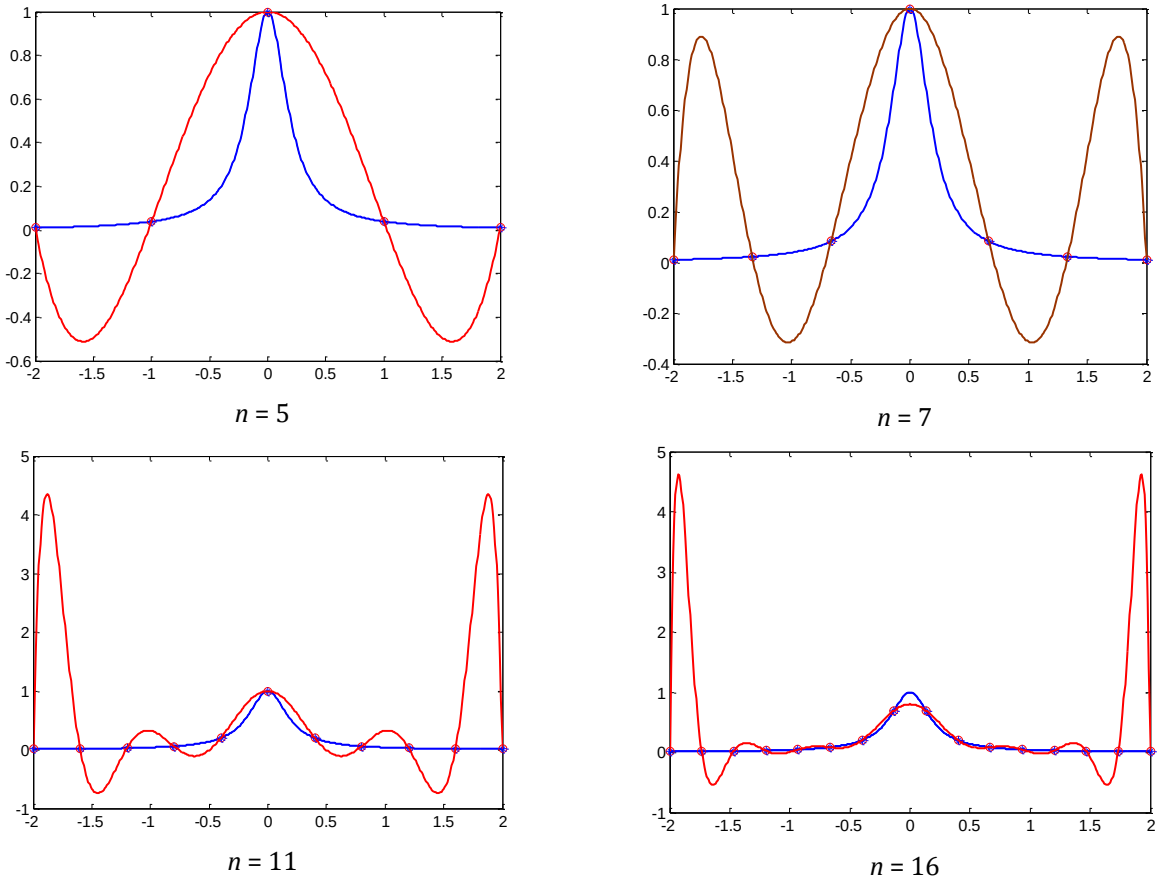


Fig. 4.3 : Interpolations avec des nœuds uniformes.

Lorsque les nœuds sont uniformes, l'interpolation lagrangienne n'est pas la meilleure (Fig. 4.3).

Considérant les nœuds de Tchebychev type I, soit le script MATLAB :

```
f=@(x)1./(1+25*x.*x);
a=-1.5; b=-a; % Définition de l'intervalle de travail
X=a:0.01:b; fx=f(X);
plot(X,fx,'-b');
n=26; % Choix du degré du polynôme d'interpolation
% Noeuds de Tchebychev Type 1
for j=1:(n+1)
 if (j==1) | (j==n+1)
  xShebychev1(j)=((b-a)/2)*cos((j-1)*pi/n)+0.5*(a+b);
 else
  xShebychev1(j)=((b-a)/2)*cos(((2*j-3)*pi)/(2*n-2))+0.5*(a+b);
 end
  yShebychev1(j)=f(xShebychev1(j));
end
```

```
for i=1:numel(X)
YSH1(i)=Interp2Newton(xShebychev1,yShebychev1,X(i));
end
plot(X,YSH1,'-r')
```

L'exécution du script pour différentes valeurs de *n* permet d'avoir les résultats (Fig. 4.4 & 4.5). L'utilisation des nœuds de Tchebychev peut minimiser l'oscillation des polynômes interpolateurs au lieu des nœuds uniformes surtout pour des degrés élevés (plus que 26).

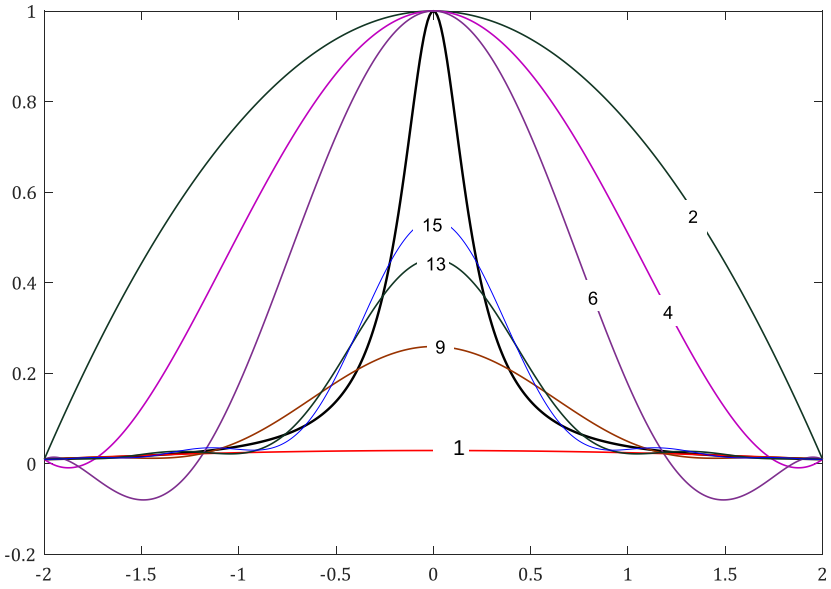


Fig. 4.4 : Interpolations avec nœuds de Tchebychev pour *n* = 1, 2, 4, 6, 9, 13 et 15.

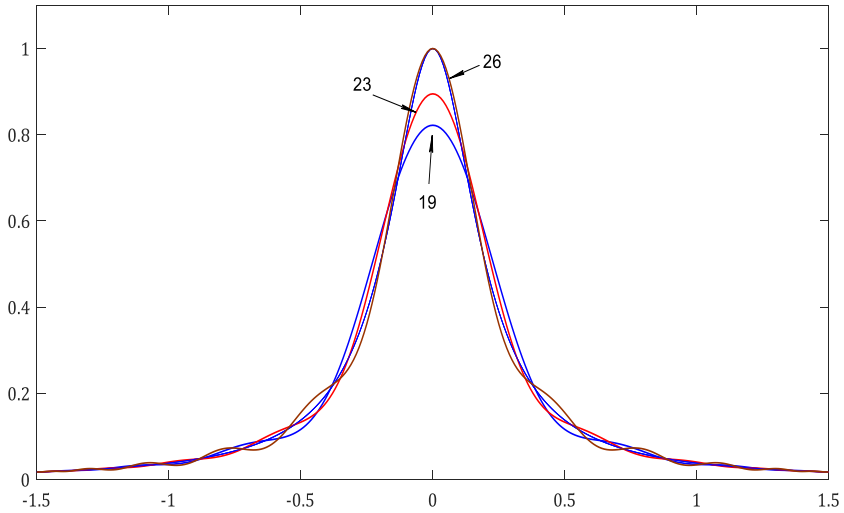


Fig. 4.5 : Interpolations avec nœuds de Tchebychev pour *n*=19, 23 et 26.

10. Interpolation d'Hermite

L'interpolation d'Hermite, est une extension de l'interpolation de Lagrange, qui consiste pour une fonction donnée $f(x)$ et un nombre fini de nœuds x_0, x_1, …, x_n de construire un polynôme à la fois *interpolateur* et *osculateur* c'est-à-dire,

$$P_n(x_i) = f(x_i) \quad \text{et} \quad P_n'(x_i) = f'(x_i) \quad \forall i = 0,\ 1,\ \ldots,\ n$$

Cette méthode d'interpolation permet de manipuler des polynômes ayant des propriétés proches de celles de la fonction $f(x)$. Soit le théorème :

$x_0,\ x_1,\ \ldots,\ x_n$ $n+1$ nœuds distincts et $f(x)$ une fonction continue et dérivable. Le polynôme d'interpolation de Hermite est de degré $2n+1$ est donné par :

$$H_{2n+1}(x)=\sum_{j=0}^{n} f(x_j)H_{n,j}(x)+\sum_{j=0}^{n} f'(x_j)\hat{H}_{n,j}(x)$$

où,

$$H_{n,j}(x)=\left[1-2(x-x_j)L'_{n,j}(x)\right]L^2_{n,j}(x) \quad \text{et} \quad \hat{H}_{n,j}(x)=(x-x_j)L^2_{n,j}(x)$$

avec,

$$L_{n,j}(x)=\prod_{\substack{k=0\\k\neq j}}^{n}\frac{x-x_k}{x_j-x_k}$$

Le polynôme $H_{2n+1}(x)$ interpole la fonction $f(x)$ dans les nœuds $x_0,\ x_1,\ \ldots,\ x_n$, c'est-à-dire :

$$H_{2n+1}(x_j)=f(x_j) \ \Rightarrow\ H_{n,j}(x_k)=\delta_{jk} \quad \text{et} \quad \hat{H}_{n,j}(x_k)=0,\ \forall\ j\ \&\ k$$

$$H'_{2n+1}(x_j)=f'(x_j) \ \Rightarrow\ H'_{n,j}(x_k)=0\ \forall\ j\ \&\ k \quad \text{et} \quad \hat{H}'_{n,j}(x_k)=\delta_{jk}$$

Puisque le degré du polynôme $H_{2n+1}(x)$ est $2n+1$, soit à titre de considération :

$$H_{n,j}(x)=(\alpha_j x+\beta_j)L^2_{n,j}(x) \quad \text{et} \quad \hat{H}_{n,j}(x)=(\hat{\alpha}_j x+\hat{\beta}_j)L^2_{n,j}(x)$$

par conséquent,

$$H'_{n,j}(x)=\left[2xL_{n,j}(x)L'_{n,j}(x)+L^2_{n,j}(x)\right]\alpha_j+2L_{n,j}(x)L'_{n,j}(x)\beta_j$$

$$\hat{H}'_{n,j}(x)=\left[2xL_{n,j}(x)L'_{n,j}(x)+L^2_{n,j}(x)\right]\hat{\alpha}_j+2L_{n,j}(x)L'_{n,j}(x)\hat{\beta}_j$$

$L_{n,j}(x)$ est le facteur de Lagrange,

$$L_{n,j}(x_k)=\delta_{jk}$$

alors,

$$H_{n,j}(x_j)=1 \ \Rightarrow\ (\alpha_j x_j+\beta_j)L^2_{n,j}(x_j)=1 \ \text{ ou } \ \alpha_j x_j+\beta_j=1$$

$$\hat{H}_{n,j}(x_j)=0 \ \Rightarrow\ (\hat{\alpha}_j x_j+\hat{\beta}_j)L^2_{n,j}(x_j)=0 \ \text{ ou } \ \hat{\alpha}_j x_j+\hat{\beta}_j=0$$

et

$$H'_{n,j}(x_j)=0 \ \Rightarrow\ \left[2x_jL_{n,j}(x_j)L'_{n,j}(x_j)+L^2_{n,j}(x_j)\right]\alpha_j+2L_{n,j}(x_j)L'_{n,j}(x_j)\beta_j=0$$

ou bien,

$$\left[2x_jL'_{n,j}(x_j)+1\right]\alpha_j+2L'_{n,j}(x_j)\beta_j=0$$

de même,

$$\hat{H}'_{n,j}(x_j)=1 \ \Rightarrow\ \left[2x_jL_{n,j}(x_j)L'_{n,j}(x_j)+L^2_{n,j}(x_j)\right]\hat{\alpha}_j+2L_{n,j}(x_j)L'_{n,j}(x_j)\hat{\beta}_j=1$$

ou bien,

$$\left[2x_jL'_{n,j}(x_j)+1\right]\hat{\alpha}_j+2L'_{n,j}(x_j)\hat{\beta}_j=1$$

soit finalement les deux systèmes :

$$\begin{cases}\alpha_j x_j+\beta_j=1,\\ \left(2x_jL'_{n,j}\left(x_j\right)+1\right)\alpha_j+2L'_{n,j}\left(x_j\right)\beta_j=0.\end{cases}\quad\text{et}\quad\begin{cases}\hat{\alpha}_j x_j+\hat{\beta}_j=0,\\ \left(2x_jL'_{n,j}\left(x_j\right)+1\right)\hat{\alpha}_j+2L'_{n,j}\left(x_j\right)\hat{\beta}_j=1.\end{cases}$$

la résolution des systèmes conduit à :

$$\alpha_j=-2L'_{n,j}\left(x_j\right),\ \beta_j=1+2x_jL'_{n,j}\left(x_j\right),\ \hat{\alpha}_j=1\ \text{ et }\ \hat{\beta}_j=-x_j$$

alors,

$$H_{n,j}\left(x\right)=\left[1-2\left(x-x_j\right)L'_{n,j}\left(x\right)\right]L^2_{n,j}\left(x\right)\ \text{ et }\ \hat{H}_{n,j}\left(x\right)=\left(x-x_j\right)L^2_{n,j}\left(x\right)$$

A titre d'exemple, déterminant le polynôme d'interpolation de Hermite pour les données :

j	x_j	$f(x_j)$	$f'(x_j)$
0	1.3	0.62	-0.52
1	1.6	0.46	-0.57
2	1.9	0.28	-0.58

Détermination des polynômes de Lagrange et leurs dérivées :

$$L_{2,0}\left(x\right)=\frac{\left(x-x_1\right)\left(x-x_2\right)}{\left(x_0-x_1\right)\left(x_0-x_2\right)}=\frac{\left(x-1.6\right)\left(x-1.9\right)}{\left(1.3-1.6\right)\left(1.3-1.9\right)}=\frac{50}{9}x^2-\frac{175}{9}x+\frac{152}{9},\ L'_{2,0}\left(x\right)=\frac{100}{9}x-\frac{175}{9}$$

$$L_{2,1}\left(x\right)=\frac{\left(x-x_0\right)\left(x-x_2\right)}{\left(x_1-x_0\right)\left(x_1-x_2\right)}=\frac{\left(x-1.3\right)\left(x-1.9\right)}{\left(1.6-1.3\right)\left(1.6-1.9\right)}=-\frac{100}{9}x^2+\frac{320}{9}x-\frac{274}{9},\ L'_{2,1}\left(x\right)=-\frac{200}{9}x+\frac{320}{9}$$

$$L_{2,2}\left(x\right)=\frac{\left(x-x_0\right)\left(x-x_1\right)}{\left(x_2-x_0\right)\left(x_2-x_1\right)}=\frac{\left(x-1.3\right)\left(x-1.6\right)}{\left(1.9-1.3\right)\left(1.9-1.6\right)}=\frac{50}{9}x^2-\frac{145}{9}x+\frac{104}{9},\ L'_{2,2}\left(x\right)=\frac{100}{9}x-\frac{145}{9}$$

alors,

$$H_{2,0}\left(x\right)=\left[1-2\left(x-1.3\right)\left(-5\right)\right]\left(\frac{50}{9}x^2-\frac{175}{9}x+\frac{152}{9}\right)^2=\left(10x-12\right)\left(\frac{50}{9}x^2-\frac{175}{9}x+\frac{152}{9}\right)^2$$

$$H_{2,1}\left(x\right)=1.\left(-\frac{100}{9}x^2+\frac{320}{9}x-\frac{274}{9}\right)^2,\ H_{2,2}\left(x\right)=10\left(2-x\right)\left(\frac{50}{9}x^2-\frac{145}{9}x+\frac{104}{9}\right)^2$$

$$\hat{H}_{2,0}\left(x\right)=\left(x-1.3\right)\left(\frac{50}{9}x^2-\frac{175}{9}x+\frac{152}{9}\right)^2,\ \hat{H}_{2,1}\left(x\right)=\left(x-1.6\right)\left(-\frac{100}{9}x^2+\frac{320}{9}x-\frac{274}{9}\right)^2$$

$$\hat{H}_{2,2}\left(x\right)=\left(x-1.9\right)\left(\frac{50}{9}x^2-\frac{145}{9}x+\frac{104}{9}\right)^2$$

Le polynôme d'interpolation d'Hermite est :

$$H_5\left(x\right)=0.62H_{2,0}\left(x\right)+0.46H_{2,1}\left(x\right)+0.28H_{2,2}\left(x\right)-0.52\hat{H}_{2,0}\left(x\right)-0.57\hat{H}_{2,1}\left(x\right)-0.58\hat{H}_{2,2}\left(x\right)$$

L'estimation pour x = 1.5 est :

$$H_5\left(1.5\right)=0.62\left(\frac{4}{27}\right)+0.46\left(\frac{64}{81}\right)+0.28\left(\frac{5}{81}\right)-0.52\left(\frac{4}{405}\right)-0.57\left(-\frac{32}{405}\right)-0.58\left(-\frac{2}{405}\right)=0.52$$

Dans le cas où la fonction $f'(x)$ n'est pas connue et $f(x)$ est donnée que par $y_j = f(x_j)$ c'est-à-dire que par l'ensemble des couples $(x_0,\ y_0)$, $(x_1,\ y_1)$, …, $(x_n,\ y_n)$, alors la dérivée $f'(x)$ peut être approchée par différences finies,

$$\text{avant } \frac{y_{j+1}-y_j}{x_{j+1}-x_j} \text{ et arrière } \frac{y_j-y_{j-1}}{x_j-x_{j-1}}$$

Par conséquent l'expression du polynôme d'interpolation de Hermite est peut-être approchée par :

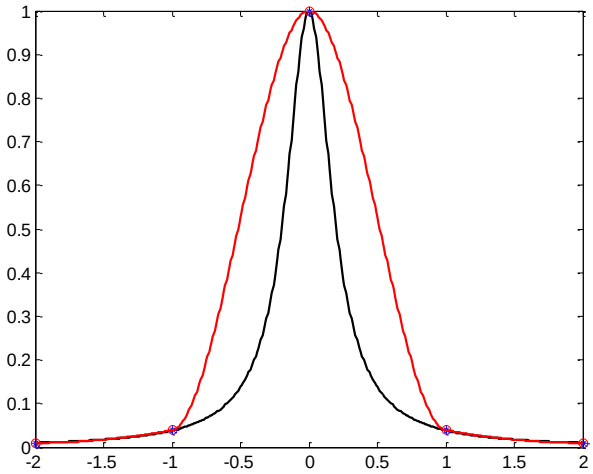
$$H_{2n+1}(x) \approx \sum_{j=0}^{n} y_j H_{n,j}(x) + \sum_{j=0}^{n-1} \frac{y_{j+1}-y_j}{x_{j+1}-x_j} \hat{H}_{n,j}(x) + \frac{y_n - y_{n-1}}{x_n - x_{n-1}} \hat{H}_{n,n}(x)$$

Si les nœuds sont uniformes une bonne approximation de $H_{2n+1}(x)$ qui est exprimé par :

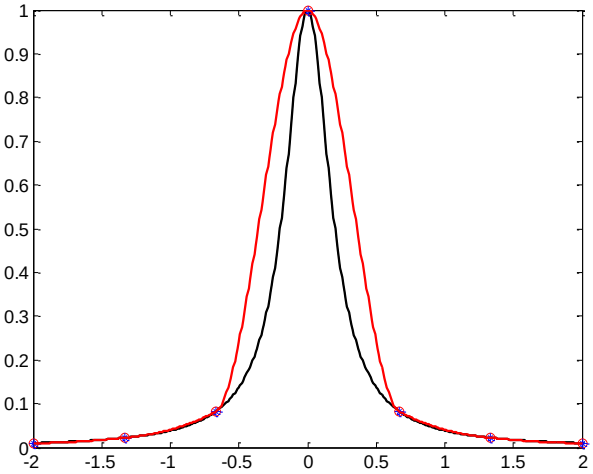
$$H_{2n+1}(x) \approx \sum_{j=0}^{n} y_j H_{n,j}(x) + \frac{y_1 - y_0}{\Delta x} \hat{H}_{n,0}(x) + \frac{1}{2\Delta x} \sum_{j=1}^{n-1} (y_{j+1} - y_{j-1}) \hat{H}_{n,j}(x) + \frac{y_n - y_{n-1}}{\Delta x} \hat{H}_{n,n}(x)$$

Avec MATLAB, il est possible de réaliser ce type d'interpolation avec la fonction **pchip**, le script suivant traite le phénomène de Runge pour différents nombres de nœuds uniformes (Fig. 4.6) :

```
f=@(x)1./(1+25*x.*x);
a = -1; b = -a; % Définition de l'intervalle de travaille
X=a:0.01:b; fx=f(X);
plot(X,fx,'-k') % Graphe de la fonction de Runge
hold on
Nombre2Noeuds=5; % Choix du nombre de nœuds
n = Nombre2Noeuds-1; h=(b-a)/n;
xu=a:h:b; yu=f(xu);
plot(xu,yu,'or',xu,yu,'*b') % Points d'interpolation
for i=1:numel(X)
  YbyHermite(i)=pchip(xu,yu,X(i));
end
plot(X,YU,'-r')
```



5 nœuds



7 nœuds

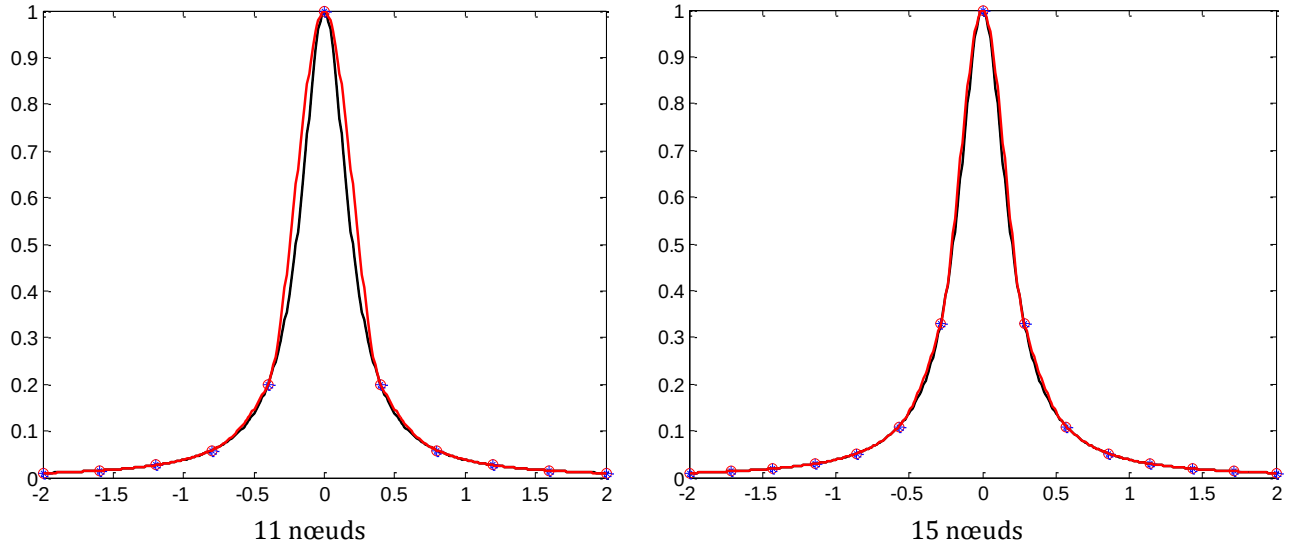


Fig. 4.6 : Interpolation de l'Hermite pour les nœuds uniformes de nombres 5, 7, 11 et 15.

11. Splines cubiques

Lorsqu'on cherche à interpoler sur un segment [a, b] une fonction $f(x)$ par un polynôme $P(x)$ même en augmentant le nombre de points d'interpolation, l'approximation globale peut être mauvaise (phénomène de Runge). En outre, le calcul du polynôme $P(x)$ devient compliqué pouvant donner lieu à des erreurs.

Une autre idée consiste à utiliser plusieurs polynômes de bas degré sur chaque intervalle [x_i, x_{i+1}] de [a, b], et de recoller ces polynômes pour définir une fonction $S(x)$ sur l'intervalle tout entier de façon que la fonction $S(x)$ possède un minimum de régularité (continuité, dérivabilité...).

Par définition, soient $a = x_0 < x_1 < x_2 \ldots < x_n = b$ n+1 nœuds, $y_0 = f(x_0)$, $y_1 = f(x_1)$, …, $y_n = f(x_n)$ leurs ordonnées ou images. On appelle spline cubique associée à la famille $(x_j,\ f(x_j))$ toute fonction $S(x)$ polynômiale de degré au plus trois sur chaque sous intervalle [x_j, x_{j+1}] de [a, b] telle que $S(x_j) = y_j = f(x_j)$,$\forall\ j = 0,\ 1,\ \ldots,\ n$. Une telle fonction $S(x)$ interpolant $f(x)$ satisfait les conditions suivantes :

(a) $S(x)$ est polynôme de degré 3, noté $S_j(x)$ pour $x_j \le x \le x_{j+1}$, $\forall\ j = 0,\ 1,\ \ldots,\ n-1$;

(b) $S_j(x_j) = f(x_j) = y_j$ et $S_j(x_{j+1}) = f(x_{j+1}) = y_{j+1}$, $\forall\ j = 0,\ 1,\ \ldots,\ n-1$;

(c) $S_j(x_{j+1}) = S_{j+1}(x_{j+1})$, $\forall\ j = 0,\ 1,\ \ldots,\ n-1$;

(d) $S'_j(x_{j+1}) = S'_{j+1}(x_{j+1})$, $\forall\ j = 0,\ 1,\ \ldots,\ n-2$;

(e) $S''_j(x_{j+1}) = S''_{j+1}(x_{j+1})$, $\forall\ j = 0,\ 1,\ \ldots,\ n-2$;

(f) Conditions concernent les extrémités de l'intervalle [x_0, x_n].

Pour déterminer la nature de la fonction spline $S(x)$, soit $S_j(x)$ le polynôme d'interpolation définie sur le sous intervalle [x_j, x_{j+1}] :

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \ \forall\ j = 0,\ 1,\ \ldots,\ n-1$$

d'après la condition (b),

$$S_j(x_j) = f(x_j) = y_j \ \Rightarrow\ a_j = y_j, \ \forall\ j = 0,\ 1,\ \ldots,\ n-1$$

de la condition (c),

$$S_j(x_{j+1}) = S_{j+1}(x_{j+1}) = y_{j+1} \ \Rightarrow\ a_j + b_j(x_{j+1} - x_j) + c_j(x_{j+1} - x_j)^2 + d_j(x_{j+1} - x_j)^3 = y_{j+1}$$

soit, $h_j = x_{j+1} - x_j$ et puisque $a_j = y_j$, alors l'équation précédente devient :

$$b_j + c_j h_j + d_j h_j^2 = \frac{y_{j+1} - y_j}{h_j}, \ \forall\ j = 0,\ 1,\ \ldots,\ n-1 \qquad (4.1)$$

D'après la condition (d),

$$S'_j(x_{j+1}) = S'_{j+1}(x_{j+1}), \ \forall\ j = 0,\ 1,\ \ldots,\ n-2$$

c'est-à-dire,

$$b_j + 2c_j(x_{j+1} - x_j) + 3d_j(x_{j+1} - x_j)^2 = b_{j+1} + 2c_{j+1}(x_{j+1} - x_{j+1}) + 3d_{j+1}(x_{j+1} - x_{j+1})^2$$

ou bien,

$$b_j + 2c_j h_j + 3d_j h_j^2 = b_{j+1}, \ \forall\ j = 0,\ 1,\ \ldots,\ n-2 \qquad (4.2)$$

D'après la condition (e),

$$S''_j(x_{j+1}) = S''_{j+1}(x_{j+1}), \ \forall\ j = 0,\ 1,\ \ldots,\ n-2$$

c'est-à-dire,

$$2c_j + 6d_j(x_{j+1} - x_j) = 2c_{j+1} + 6d_{j+1}(x_{j+1} - x_{j+1})$$

ou bien,

$$c_j + 3d_j h_j = c_{j+1}, \ \forall\ j = 0,\ 1,\ \ldots,\ n-2 \qquad (4.3)$$

Pour $j = 0,\ 1,\ \ldots,\ n-2$, la résolution de l'équation (4.3) pour d_j est :

$$d_j = \frac{c_{j+1} - c_j}{3h_j} \qquad (4.4)$$

remplaçant d_j dans l'équation (4.2),

$$b_{j+1} - b_j = h_j(c_j + c_{j+1})$$

de même dans l'équation (4.1),

$$b_j + \frac{1}{3}(2c_j + c_{j+1})h_j = \frac{y_{j+1} - y_j}{h_j}$$

soit,

$$b_j = \frac{y_{j+1} - y_j}{h_j} - \frac{1}{3}(2c_j + c_{j+1})h_j \qquad (4.5)$$

b_{j+1} est déduit de b_j comme :

$$b_{j+1} = \frac{y_{j+2} - y_{j+1}}{h_{j+1}} - \frac{1}{3}(2c_{j+1} + c_{j+2})\,h_{j+1}$$

or,

$$b_{j+1} - b_j = h_j(c_j + c_{j+1}) \ \Rightarrow\ \Delta y_{j+1} - \frac{1}{3}(2c_{j+1} + c_{j+2})h_{j+1} - \Delta y_j + \frac{1}{3}(2c_j + c_{j+1})h_j = h_j(c_j + c_{j+1})$$

avec,

$$\Delta y_j = \frac{y_{j+1} - y_j}{h_j} \text{ et } \Delta y_{j+1} = \frac{y_{j+2} - y_{j+1}}{h_{j+1}}$$

finalement,

$$h_j c_j + 2\left(h_j + h_{j+1}\right)c_{j+1} + h_{j+1}c_{j+2} = 3\left(\Delta y_{j+1} - \Delta y_j\right)$$

qui est un système de $n-1$ équations linéaires avec $n+1$ inconnus c_j, $j=0, 1, \ldots, n$ à déterminés est peut-être explicité sous forme :

$$\begin{cases} h_0 c_0 + 2(h_0 + h_1)c_1 + h_1 c_2 = 3(\Delta y_1 - \Delta y_0), \\ h_1 c_1 + 2(h_1 + h_2)c_2 + h_2 c_3 = 3(\Delta y_2 - \Delta y_1), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ h_k c_k + 2(h_k + h_{k+1})c_{k+1} + h_{k+1} c_{k+2} = 3(\Delta y_{k+1} - \Delta y_k), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ h_{n-2} c_{n-2} + 2(h_{n-2} + h_{n-1})c_{n-1} + h_{n-1} c_n = 3(\Delta y_{n-1} - \Delta y_{n-2}). \end{cases}$$

ou sous forme matricielle,

$$\begin{pmatrix} - & - & - & - & - & - \\ h_0 & 2(h_0+h_1) & h_1 & 0 & \cdots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \ddots & \vdots \\ \vdots & \ddots & & \ddots & \ddots & 0 \\ 0 & & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ - & - & - & - & - & - \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} - \\ 3(\Delta y_1 - \Delta y_0) \\ 3(\Delta y_2 - \Delta y_1) \\ \vdots \\ 3(\Delta y_{n-1} - \Delta y_{n-2}) \\ - \end{pmatrix}$$

Pour compléter le système d'équations, Il faut deux équations qui peuvent être obtenues suivant les conditions aux extrémités.

11.1. Conditions aux extrémités

Plusieurs conditions aux extrémités peuvent être définies selon la nature du problème étudiée. Parmi ces conditions soit :

1. $S''(x_0) = S''(x_n) = 0$ (*extrémités naturelles* ou *libres "Natural Spline"*) ;
2. $S'(x_0) = f'(x_0)$ et $S'(x_n) = f'(x_n)$ (*extrémités serrées* ou *tendues "Clamped Spline"*) ;
3. $S'(x_0) = S'(x_n)$ et $S''(x_0) = S''(x_n)$ (*extrémités périodiques "Periodic Spline"*) ;
4. $S_0'''(x_1) = S_1'''(x_1)$ et $S_{n-2}'''(x_{n-1}) = S_{n-1}'''(x_{n-1})$ (*"Not-a-Knot Spline"*).

11.1.1. Spline naturelle

Dans ce cas,

$$S''(x_0) = 0 \text{ et } S''(x_n) = 0$$

c'est-à-dire,

$$2c_0 + 6d_0(x_0 - x_0) = 0 \text{ et } 2c_{n-1} + 6d_{n-1}(x_n - x_{n-1}) = 0 \Leftrightarrow c_0 = 0 \text{ et } c_{n-1} + 3d_{n-1}h_{n-1} = 0$$

de l'équation (4.4),

$$d_{n-1} = \frac{c_n - c_{n-1}}{3h_{n-1}}$$

alors,

$$c_{n-1}+3d_{n-1}h_{n-1}=0 \;\Leftrightarrow\; c_{n-1}+3\frac{c_n-c_{n-1}}{3h_{n-1}}h_{n-1}=0 \;\Rightarrow\; c_n=0$$

finalement soit le système d'équations du spline naturelle :

$$\begin{cases} c_0=0, \\ h_0c_0+2(h_0+h_1)c_1+h_1c_2=3(\Delta y_1-\Delta y_0), \\ h_1c_1+2(h_1+h_2)c_2+h_2c_3=3(\Delta y_2-\Delta y_1), \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ h_{n-2}c_{n-2}+2(h_{n-2}+h_{n-1})c_{n-1}+h_{n-1}c_n=3(\Delta y_{n-1}-\Delta y_{n-2}), \\ c_n=0. \end{cases}$$

sous forme matricielle,

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 & \cdots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \ddots & \vdots \\ \vdots & \ddots & & \ddots & \ddots & 0 \\ 0 & & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} 0 \\ 3(\Delta y_1-\Delta y_0) \\ 3(\Delta y_2-\Delta y_1) \\ \vdots \\ 3(\Delta y_{n-1}-\Delta y_{n-2}) \\ 0 \end{pmatrix}$$

Soit la fonction **NaturalSpline** sert à l'estimation d'un vecteur **Y** par spline cubique naturelle à partir d'un vecteur donné **X** dont les valeurs des composantes sont comprises dans l'intervalle [*a*, *b*].

```
function Y = NaturalSpline(x,y,X)
% x et y sont les vecteurs de données de même dimension numel(x)>=3
% Formulation des pas h, dy et B de chaque sous intervalle
 for i=1:numel(x)-1
    h(i)=x(i+1)-x(i); dy(i)=(y(i+1)-y(i))/(x(i+1)-x(i));
 end
 for i=2:numel(x)-1
    B(i)=3*(dy(i)-dy(i-1));
 end
% Formulation des conditions
a(1,1)=1; a(numel(x),numel(x))=1; B(1)=0; B(numel(x))=0;
% Formulation de la matrice a(i,j), i=2:numel(x)-1, j=1:numel(x)
 for j=1:numel(x)
  for i=2:numel(x)-1
    if i==j
        a(i,i-1)=h(i-1); a(i,i)=2*h(i-1)+2*h(i); a(i,i+1)=h(i);
    else
      a(i,j)=0;
    end
  end
 end
c=inv(a)*B';
for j=1:numel(x)-1
   b(j)=dy(j)-(1/3)*(2*c(j)+c(j+1))*h(j); d(j)=(c(j+1)-c(j))/(3*h(j));
```

```
end
 for j=1:numel(X)
  for i=1:numel(x)-1
     if X(j)<x(1) | X(j)>x(numel(x))
      Y(j)=9999;
      else
      if X(j)>=x(i) & X(j)<=x(i+1)
        dX=X(j)-x(i); Y(j)=y(i)+b(i)*dX+c(i)*(dX^2)+d(i)*(dX^3);
        break
      end
     end
  end
 end
end
```

11.1.2. Spline avec extrémités serrées ou tendues "Clamped Spline"

Dans ce cas,

$$S'(x_0)=f'(x_0) \text{ et } S'(x_n)=f'(x_n)$$

où,

$$S'(x_0)=S'_0(x_0)=b_0+2c_0(x_0-x_0)+3d_0(x_0-x_0)^2=f'(x_0) \;\Rightarrow\; b_0=f'(x_0)$$

et,

$$S'(x_n)=S'_{n-1}(x_n)=b_{n-1}+2c_{n-1}(x_n-x_{n-1})+3d_{n-1}(x_n-x_{n-1})^2=f'(x_n)$$

ou bien,

$$b_{n-1}+2c_{n-1}h_{n-1}+3d_{n-1}h_{n-1}^2=f'(x_n)$$

d'après (4.5) et (4.4),

$$b_0=\frac{y_1-y_0}{h_0}-\frac{1}{3}(2c_0+c_1)h_0,\;\; b_{n-1}=\frac{y_n-y_{n-1}}{h_{n-1}}-\frac{1}{3}(2c_{n-1}+c_n)h_{n-1} \text{ et } d_{n-1}=\frac{c_n-c_{n-1}}{3h_{n-1}}$$

soit,

$$\frac{y_1-y_0}{h_0}-\frac{1}{3}(2c_0+c_1)h_0=f'(x_0) \;\Rightarrow\; 2h_0c_0+h_0c_1=3\Delta y_0-3f'(x_0)\equiv 3\left(f[x_0,x_1]-f'(x_0)\right)$$

$$h_{n-1}c_{n-1}+2h_{n-1}c_n=3f'(x_{n-1})-3\Delta y_{n-1}\equiv 3\left(f'(x_{n-1})-f[x_{n-1},x_n]\right)$$

Soit la forme matricielle du système d'équations de spline avec extrémités serrées ou tendues "*Clamped Spline*" :

$$\begin{pmatrix} 2h_0 & h_0 & 0 & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 & \cdots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \ddots & \vdots \\ \vdots & \ddots & & \ddots & \ddots & 0 \\ 0 & & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} 3(\Delta y_0-f'(x_0)) \\ 3(\Delta y_1-\Delta y_0) \\ 3(\Delta y_2-\Delta y_1) \\ \vdots \\ 3(\Delta y_{n-1}-\Delta y_{n-2}) \\ 3(f'(x_{n-1})-\Delta y_{n-1}) \end{pmatrix}$$

Soit la fonction **`ClampedSpline`** qui sert à l'estimation d'un vecteur **`Y`** par spline cubique avec extrémités serrées ou tendues "*Clamped Spline*" à partir d'un vecteur donné **`X`** dont les valeurs des composantes sont comprises dans l'intervalle [*a*, *b*].

```
function Y = ClampedSpline(x,y,dfmin,dfmax,X)
% x et y sont les vecteurs de données de même dimensions m>=3
% dfmin=f'(x0) et dfmax=f'(xn-1)
```

```
m=numel(x);
% Formulation des pas h, dy et B de chaque sous intervalle
for i=1:numel(x)-1
  h(i)=x(i+1)-x(i); dy(i)=(y(i+1)-y(i))/(x(i+1)-x(i));
end
 for i=2:numel(x)-1
    B(i)=3*(dy(i)-dy(i-1));
 end
B=B';
% Formulation des conditions
a(1,1)=2*h(1);a(1,2)=h(1); B(1)=3*(dy(1)-dfmin);
a(m,m-1)=h(m-1); a(m,m)=2*h(m-1);B(m)=3*(dfmax-dy(m-1));
% Formulation de la matrice a(i,j), i=2:numel(x)-1, j=1:numel(x)
 for j=1:numel(x)
    for i=2:numel(x)-1
       if i==j
           a(i,i-1)=h(i-1); a(i,i)=2*h(i-1)+2*h(i); a(i,i+1)=h(i);
       else
         a(i,j)=0;
       end
    end
 end
c=inv(a)*B;
 for j=1:numel(x)-1
    b(j)=dy(j)-(1/3)*(2*c(j)+c(j+1))*h(j); d(j)=(c(j+1)-c(j))/(3*h(j));
 end
 for j=1:numel(X)
  for i=1:numel(x)-1
       if X(j)<x(1) | X(j)>x(numel(x))
        Y(j)=9999;
        else
        if X(j)>=x(i) & X(j)<=x(i+1)
          DX=X(j)-x(i);
          Y(j)=y(i)+b(i)*DX+c(i)*(DX^2)+d(i)*(DX^3);
          break
        end
       end
  end
 end
end
```

11.1.3. Spline périodique

Dans ce cas,

$$S'(x_0)=S'(x_n) \quad \text{et} \quad S''(x_0)=S''(x_n)$$

alors,

$$S'(x_0)=S'(x_n) \;\Leftrightarrow\; S'_0(x_0)=S'_{n-1}(x_n) \;\Rightarrow\; b_0=b_{n-1}+2c_{n-1}(x_n-x_{n-1})+3d_{n-1}(x_n-x_{n-1})^2$$

c'est-à-dire,

$$b_{n-1}+2c_{n-1}h_{n-1}+3d_{n-1}h_{n-1}^2=b_0$$

et,

$$S''(x_0)=S''(x_n) \;\Leftrightarrow\; S''_0(x_0)=S''_{n-1}(x_n) \;\Rightarrow\; 2c_0=2c_{n-1}+6d_{n-1}(x_n-x_{n-1})$$

c'est-à-dire,

$$c_0=c_{n-1}+3d_{n-1}h_{n-1}$$

d'après (4.5) et (4.4),

$$b_0 = \frac{y_1 - y_0}{h_0} - \frac{1}{3}(2c_0 + c_1)h_0,\ \ b_{n-1} = \frac{y_n - y_{n-1}}{h_{n-1}} - \frac{1}{3}(2c_{n-1} + c_n)h_{n-1} \text{ et } d_{n-1} = \frac{c_n - c_{n-1}}{3h_{n-1}}$$

soit,

$$\frac{y_n - y_{n-1}}{h_{n-1}} - \frac{1}{3}(2c_{n-1} + c_n)h_{n-1} + 2c_{n-1}h_{n-1} + 3\left(\frac{c_n - c_{n-1}}{3h_{n-1}}\right)h_{n-1}^2 = \frac{y_1 - y_0}{h_0} - \frac{1}{3}(2c_0 + c_1)h_0$$

c'est-à-dire,

$$(2c_0 + c_1)h_0 + (c_{n-1} + 2c_n)h_{n-1} = 3\frac{y_1 - y_0}{h_0} - 3\frac{y_n - y_{n-1}}{h_{n-1}}$$

ou,

$$2h_0c_0 + h_0c_1 + h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3(\Delta y_0 - \Delta y_{n-1})$$

et,

$$c_0 = c_{n-1} + 3\left(\frac{c_n - c_{n-1}}{3h_{n-1}}\right)h_{n-1} \ \Rightarrow\ c_0 - c_n = 0$$

Soit la forme matricielle du système d'équations de spline périodique :

$$\begin{pmatrix} 2h_0 & h_0 & 0 & \cdots & h_{n-1} & 2h_{n-1} \\ h_0 & 2(h_0+h_1) & h_1 & 0 & \cdots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \ddots & \vdots \\ \vdots & \ddots & & \ddots & \ddots & 0 \\ 0 & & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 1 & \cdots & \cdots & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} 3(\Delta y_0 - \Delta y_{n-1}) \\ 3(\Delta y_1 - \Delta y_0) \\ 3(\Delta y_2 - \Delta y_1) \\ \vdots \\ 3(\Delta y_{n-1} - \Delta y_{n-2}) \\ 0 \end{pmatrix}$$

Soit la fonction **PeriodicSpline** qui sert à l'estimation d'un vecteur **Y** par spline périodique à partir d'un vecteur donné **X** dont les valeurs des composantes sont comprises dans l'intervalle [*a*, *b*].

```
function Y = PeriodicSpline(x,y,X)
% x et y sont les vecteurs de données de même dimension m>=3
% Formulation des pas h, dy et B de chaque sous intervalle
 m=numel(x);
 for i=1:numel(x)-1
    h(i)=x(i+1)-x(i); dy(i)=(y(i+1)-y(i))/(x(i+1)-x(i));
 end
 for i=2:numel(x)-1
    B(i)=3*(dy(i)-dy(i-1));
 end
B=B';
% Formulation des conditions
a(1,1)=2*h(1);a(1,2)=h(1);a(1,m-1)=h(m-1);a(1,m)=2*h(m-1);
B(1)=3*(dy(1)-dy(m-1)); a(m,1)=1; a(m,m)=-1; B(m)=0;
% Formulation de la matrice a(i,j), i=2:numel(x)-1, j=1:numel(x)
for j=1:numel(x)
  for i=2:numel(x)-1
    if i==j
     a(i,i-1)=h(i-1); a(i,i)=2*h(i-1)+2*h(i); a(i,i+1)=h(i);
    else
     a(i,j)=0;
    end
    end
```

```
 end
c=inv(a)*B;
for j=1:numel(x)-1
  b(j)=dy(j)-(1/3)*(2*c(j)+c(j+1))*h(j); d(j)=(c(j+1)-c(j))/(3*h(j));
end
for j=1:numel(X)
 for i=1:numel(x)-1
      if X(j)<x(1) | X(j)>x(numel(x))
       Y(j)=9999;
      else
       if X(j)>=x(i) & X(j)<=x(i+1)
          DX=X(j)-x(i); Y(j)=y(i)+b(i)*DX+c(i)*(DX^2)+d(i)*(DX^3);
         break
       end
      end
 end
end
end
```

11.1.4. Spline de type "*Not-a-Knot*"

Dans ce cas,

$$S_0'''(x_1)=S_1'''(x_1) \quad \text{et} \quad S_{n-2}'''(x_{n-1})=S_{n-1}'''(x_{n-1})$$

alors,

$$S_0'''(x_1)=S_1'''(x_1) \Rightarrow d_0=d_1 \quad \text{et} \quad S_{n-2}'''(x_{n-1})=S_{n-1}'''(x_{n-1}) \Rightarrow d_{n-2}=d_{n-1}$$

d'après (4.4),

$$d_0=d_1 \Rightarrow \frac{c_1-c_0}{3h_0}=\frac{c_2-c_1}{3h_1} \Rightarrow h_1c_0-(h_0+h_1)c_1+h_0c_2=0$$

et,

$$d_{n-2}=d_{n-1} \Rightarrow \frac{c_{n-1}-c_{n-2}}{3h_{n-2}}=\frac{c_n-c_{n-1}}{3h_{n-1}} \Rightarrow h_{n-1}c_{n-2}-(h_{n-2}+h_{n-1})c_1+h_{n-2}c_n=0$$

soit la forme matricielle du système d'équations de spline *"Not-a-Knot"* :

$$\begin{bmatrix} h_1 & -(h_0+h_1) & h_0 & 0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 & \cdots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \ddots & \vdots \\ \vdots & \ddots & & \ddots & \ddots & 0 \\ 0 & & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & 0 & h_{n-1} & -(h_{n-2}+h_{n-1}) & h_{n-2} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ 3(\Delta y_1-\Delta y_0) \\ 3(\Delta y_2-\Delta y_1) \\ \vdots \\ 3(\Delta y_{n-1}-\Delta y_{n-2}) \\ 0 \end{bmatrix}$$

Si $n = 2$, le système d'équations est singulier et s'écrit :

$$\begin{pmatrix} h_1 & -(h_0+h_1) & h_0 \\ h_0 & 2(h_0+h_1) & h_1 \\ h_1 & -(h_0+h_1) & h_0 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 3(\Delta y_1-\Delta y_0) \\ 0 \end{pmatrix}$$

c_0 et c_1 peuvent être exprimés en fonction de c_2 comme suit :

$$c_0 = \frac{3(\Delta y_1 - \Delta y_0)}{h_0 + 2h_1} - \frac{3h_0 + h_1}{h_0 + 2h_1} c_2 \quad \text{et} \quad c_1 = \frac{3(\Delta y_1 - \Delta y_0) h_1}{h_0 + 2h_1} - \frac{h_1 - h_0}{h_0 + 2h_1} c_2$$

On prend $c_2 = 0$ alors,

$$c_0 = \frac{3(\Delta y_1 - \Delta y_0)}{h_0 + 2h_1} \quad \text{et} \quad c_1 = \frac{3(\Delta y_1 - \Delta y_0) h_1}{h_0 + 2h_1}$$

Pour chaque type de spline, une fois c_0, c_1, …, c_n sont déterminés, b_0, b_1, …, b_{n-1} et d_0, d_1, …, d_{n-1} sont déterminées respectivement par les équations (4.4) et (4.5).

Soit la fonction **`NotAKnotSpline`** qui sert à l'estimation d'un vecteur **`Y`** par spline de type "*Not-a-Knot*" à partir d'un vecteur donné **`X`** dont les valeurs des composantes sont comprises dans l'intervalle [*a*, *b*].

```
function Y = NotAKnotSpline(x,y,X)
% x et y sont les vecteurs de données de même dimension m>=3
% Formulation des pas h, dy et B de chaque sous intervalle
 m=numel(x);
 for i=1:numel(x)-1
    h(i)=x(i+1)-x(i);
    dy(i)=(y(i+1)-y(i))/(x(i+1)-x(i));
 end
 for i=2:numel(x)-1
    B(i)=3*(dy(i)-dy(i-1));
 end
B=B';
% Formulation des conditions
 if m==3
    c(1)=B(2)/(h(1)+2*h(2));c(2)=c(1)*h(2);c(3)=0;
 else
   % Formulation de la matrice a(i,j), i=2:m-1, j=1:m
     a(1,1)=h(2); a(1,2)=-h(1)-h(2); a(1,3)=h(1); B(1)=0;
     a(m,m-2)=h(m-1); a(m,m-1)=-h(m-1)-h(m-2); a(m,m)=h(m-2); B(m)=0;
   for j=1:numel(x)
      for i=2:numel(x)-1
       if i==j
          a(i,i-1)=h(i-1); a(i,i)=2*h(i-1)+2*h(i); a(i,i+1)=h(i);
       else
         a(i,j)=0;
       end
      end
   end
  c=inv(a)*B;
 end
 for j=1:numel(x)-1
    b(j)=dy(j)-(1/3)*(2*c(j)+c(j+1))*h(j); d(j)=(c(j+1)-c(j))/(3*h(j));
 end

 for j=1:numel(X)
  for i=1:numel(x)-1
     if X(j)<x(1) | X(j)>x(numel(x))
      Y(j)=9999;
       else
      if X(j)>=x(i) & X(j)<=x(i+1)
        dX=X(j)-x(i); Y(j)=y(i)+b(i)*dX+c(i)*(dX^2)+d(i)*(dX^3);
```

```
            break
        end
    end
  end
 end
end
```

Il est possible d'estimer le vecteur **Y** avec MATLAB à partir d'un vecteur donné **X** par la fonction **spline**, celle-ci utilise une variante de condition de type "*Not-a-Knot*". La syntaxe de cette fonction est :

```
Y = spline(x,y,X)
```

L'exemple suivant (Burden & Faires, 2010) consiste à la détermination du polynôme d'interpolation par spline cubique naturelle passant par les points $(1,\ 2)$, $(2,\ 3)$, et $(3,\ 5)$.

Le polynôme d'interpolation $S(x)$ est exprimé par :

$$S(x)=\begin{cases} S_0(x)=a_0+b_0(x-1)+c_0(x-1)^2+d_0(x-1)^3 & \text{pour}\ \ 1\le x\le 2,\\ S_1(x)=a_1+b_1(x-2)+c_1(x-2)^2+d_1(x-2)^3 & \text{pour}\ \ 2\le x\le 3. \end{cases}$$

soit,

$$S_0(1)=2 \ \Rightarrow\ a_0=2,\ S_0(2)=S_1(2)=3 \ \Rightarrow\ b_0+c_0+d_0=1$$

$$S_1(2)=3 \ \Rightarrow\ a_1=3 \ \text{ et } S_1(3)=5 \ \Rightarrow\ b_1+c_1+d_1=2$$

$$S_0'(2)=S_1'(2) \ \Rightarrow\ b_0+2c_0+3d_0=b_1 \ \text{ et } S_0''(2)=S_1''(2) \ \Rightarrow\ 2c_0+6d_0=2c_1$$

$$S_1''(3)=0 \ \Rightarrow\ 2c_1+6d_1=0 \ \text{ et } S_1''(3)=0 \ \Rightarrow\ 2c_1+6d_1=0$$

Ces équations ont permis d'avoir, $a_0=2$, $a_1=3$ et $c_0=0$, le reste des coefficients sont obtenus du système d'équation :

$$\left.\begin{array}{l} b_0+d_0=1,\\ b_1+c_1+d_1=2,\\ b_0-b_1+3d_0=0,\\ c_1-3d_0=0,\\ c_1+3d_1=0. \end{array}\right\} \ \Rightarrow\ \begin{cases} b_0=3/4,\\ b_1=3/2,\\ c_1=3/4,\\ d_0=1/4,\\ d_1=-1/4. \end{cases}$$

finalement,

$$S(x)=\begin{cases} 2+\dfrac{3}{4}(x-1)+\dfrac{1}{4}(x-1)^3 & \text{pour}\ \ 1\le x\le 2,\\ 3+\dfrac{3}{2}(x-2)+\dfrac{3}{4}(x-2)^2-\dfrac{1}{4}(x-2)^3 & \text{pour}\ \ 2\le x\le 3. \end{cases}$$

11.2. Phénomène de Runge, comparaison

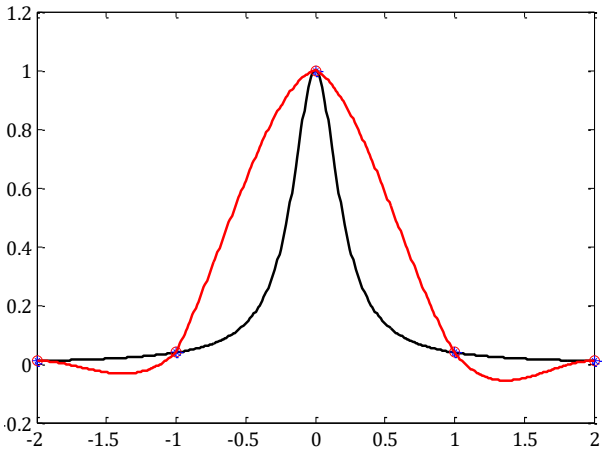
Le phénomène de Runge étudié précédemment, a montré l'interpolation polynomiale simple issue d'une distribution uniforme des nœuds n'est pas une bonne approximation de la fonction de Runge (Fig. 4.4), de même si les nœuds sont de type de Tchebychev ou le nombre de nœuds est inférieure à 17 (Fig. 4.5).

Considérons l'interpolation spline cubique avec extrémités serrées ou tendues sur l'intervalle [-2, 2] de la fonction Runge. Soit les deux scripts permettant de tracer une comparaison entre la courbe d'interpolation obtenue par spline cubique et la fonction de Runge pour un nombre de nœuds **n**, qui peut être des nœuds uniformes ou de Tchebychev :

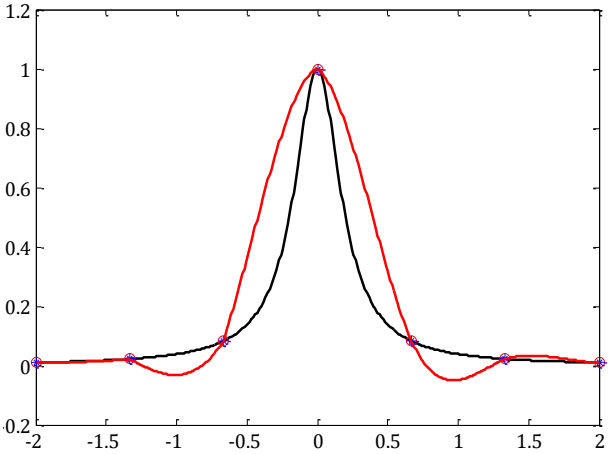
```
% Cas des nœuds uniformes
f=@(x)1./(1+25*x.*x);
df=@(x)-50*x./((1+25*x.*x).^2);
a = -2; b = -a; % L'intervalle [a, b]
X=a:0.01:b;
fx=f(X);
plot(X,fx,'-k') % Tracé de la fonction de Runge
hold on
Nombre2Noeuds=17; % Choix du nombre de nœuds uniformes
n=Nombre2Noeuds-1;
h=(b-a)/n;
xu=a:h:b; yu=f(xu);
plot(xu,yu,'or',xu,yu,'*b') % tracé de la fonction de Runge
for i=1:numel(X)
     YCS(i)=ClampedSpline(xu,yu,df(a),df(b),X(i)); % Clamped Spline
end
plot(X,YCS,'-g')
```

Pour un nombre de nœuds uniformes **n = (5, 7, 11** et **17)** l'interpolation avec spline cubique (Fig. 4.7) permet d'avoir une bonne approximation de la fonction de Runge à celle d'une interpolation polynomiale classique (Fig. 4.3).

Il est clair que l'interpolation par spline cubique est meilleure si les segments $h_j = \Delta x_j$ sont proches de zéro.



5 nœuds



7 nœuds

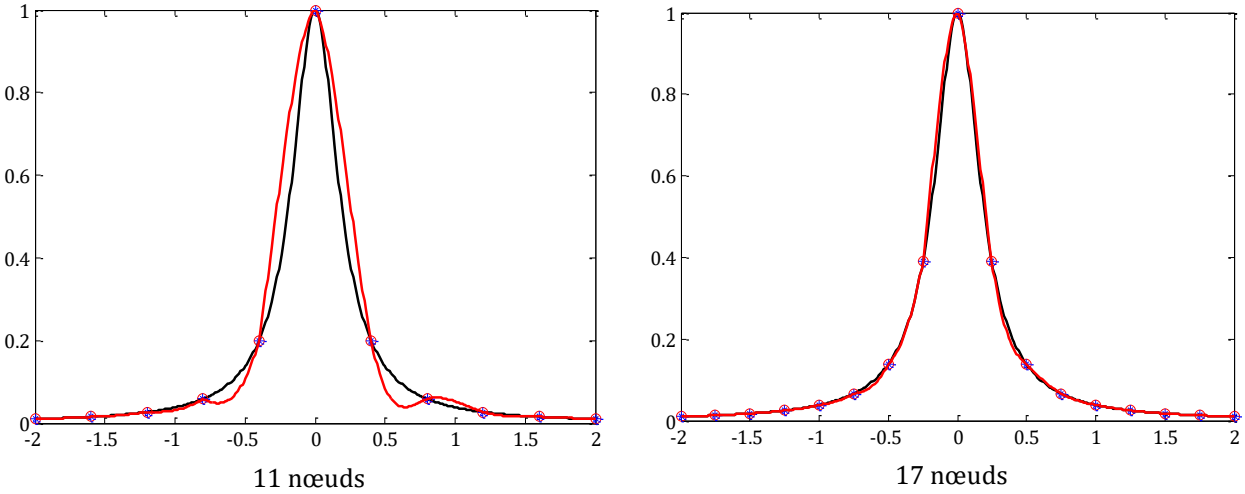


Fig. 4.7 : Spline cubique interpolation avec nœuds uniformes.

12. MATLAB et interpolation

MATLAB permet de faire l'interpolation d'une façon automatique. Par exemple pour les données :

$$(0,\ 1),\ (1,\ 7),\ (2,\ 35),\ \text{et}\ (3,\ 103)$$

Dans la fenêtre (Fig. 4.8) obtenue suite à la commande et dans la boite à outils (Basic Fitting) de menu Tools une boite de dialogue est ouverte, celle-ci, permet de choisir le degré de polynôme d'interpolation et d'afficher son l'équation et calculer la valeur **`Yd`** pour un **`Xd`** donné.

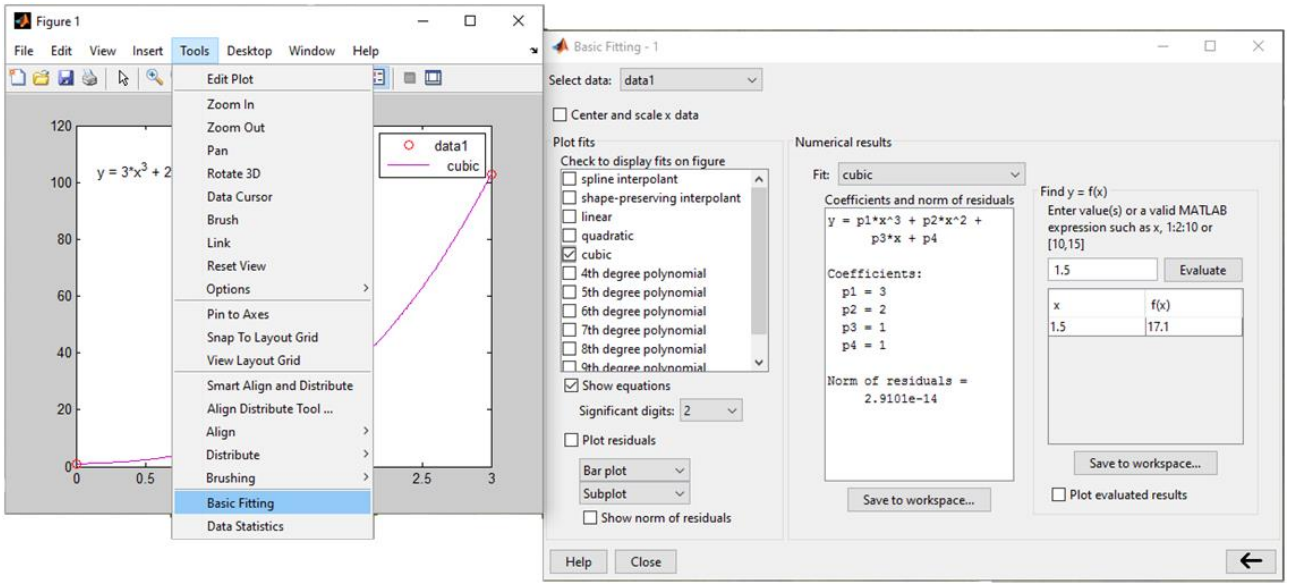


Fig. 4.8: Interpolation polynomiale avec les outils propres de MATLAB.

13. Interpolation multidimensionnelle

L'interpolation polynomiale vue dans les sections précédentes, peut être étendue à une interpolation multidimensionnelle. Soit $f(x_1,\ y_1)$, $f(x_1,\ y_2)$, $f(x_2,\ y_1)$ et $f(x_2,\ y_2)$ quatre points définis dans l'espace cartésien (Fig. 4.9). Pour un point donné dans le plan (2D) $(x_i,\ y_i)$ déterminant par interpolation linéaire $z_i = f(x_i,\ y_i)$.

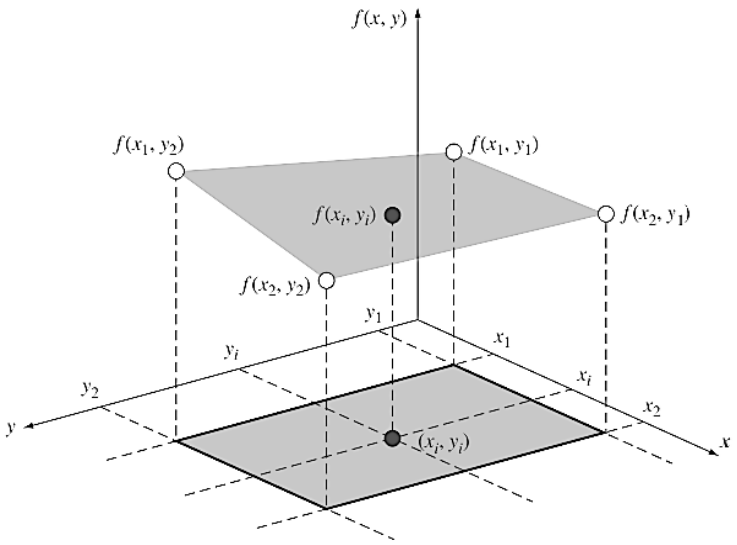


Fig. 4.9 : Interpolation bidimensionnelle linéaire.

Une approche simple pour estimer z_i est basée sur le développement d'une fonction *bilinéaire* (Fig. 4.10), qui consiste dans un premier lieu de considérer y est constant et faire une interpolation linéaire suivant l'axe des abscisses comme suit :

$$f(x_i,\ y_1)=\frac{x_i-x_2}{x_1-x_2}f(x_1,\ y_1)+\frac{x_i-x_1}{x_2-x_1}f(x_2,\ y_1)$$

$$f(x_i,\ y_2)=\frac{x_i-x_2}{x_1-x_2}f(x_1,\ y_2)+\frac{x_i-x_1}{x_2-x_1}f(x_2,\ y_2)$$

Et dans un deuxième lieu de considérer ces deux points et faire une interpolation linéaire inverse suivant l'axe des ordonnées comme suit :

$$f(x_i,\ y_i)=\frac{y_i-y_2}{y_1-y_2}f(x_i,\ y_1)+\frac{y_i-y_1}{y_2-y_1}f(x_i,\ y_2)$$

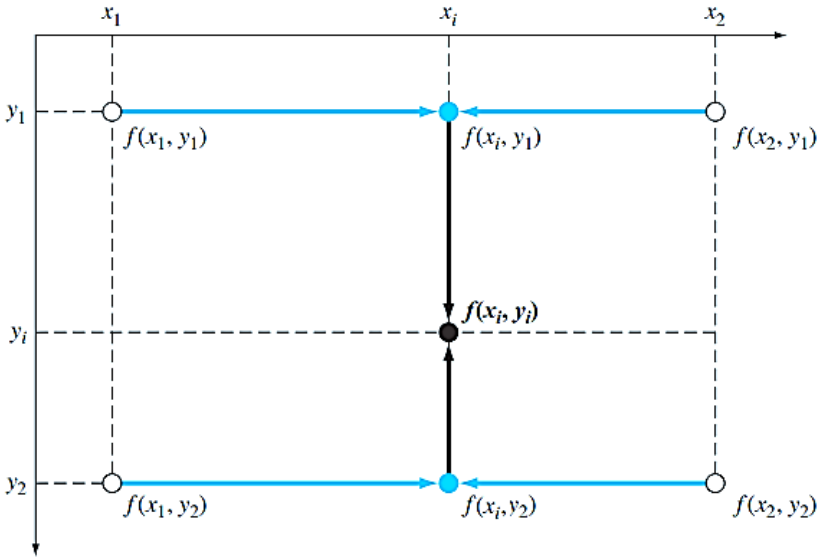


Fig. 4.10 : Procédure d'interpolation bilinéaire.

remplaçant $f(x_i,\ y_1)$ et $f(x_i,\ y_2)$ par leurs expressions, soit finalement :

$$f(x_i,\ y_i)=\frac{y_i-y_2}{y_1-y_2}\frac{x_i-x_2}{x_1-x_2}f(x_1,\ y_1)+\frac{y_i-y_2}{y_1-y_2}\frac{x_i-x_1}{x_2-x_1}f(x_2,\ y_1)$$
$$+\frac{y_i-y_1}{y_2-y_1}\frac{x_i-x_2}{x_1-x_2}f(x_1,\ y_2)+\frac{y_i-y_1}{y_2-y_1}\frac{x_i-x_1}{x_2-x_1}f(x_2,\ y_2)$$

Soit l'exemple suivant (Chapra, 2012) qui consiste à estimer la température dans le point de cordonnées x = 5.25 et y = 4.8 d'une plaque rectangulaire dont les extrémités ont les températures :

$T(2, 1) = 60$, $T(2, 6) = 55$, $T(9, 1) = 57.5$ et $T(9, 6) = 70$.

Par substitution dans l'expression trouvée précédemment, la température cherchée est :

$$T(5.25,\ 4.8)=\left(\frac{4.8-6}{1-6}\right)\left(\frac{5.25-9}{2-9}\right)60+\left(\frac{4.8-6}{1-6}\right)\left(\frac{5.25-2}{9-2}\right)57.5$$
$$+\left(\frac{4.8-1}{6-1}\right)\left(\frac{5.25-9}{2-9}\right)55+\left(\frac{4.8-1}{6-1}\right)\left(\frac{5.25-2}{9-2}\right)70=61.2143$$

Suivant le même principe il est possible de faire une interpolation *multilinéaire*. Dans MATLAB, il y a des fonctions qui traitent ces cas de figures en termes d'interpolation multidimensionnelle que ce soit linéaire ou spline, à titre d'exemple dans une interpolation bidimensionnelle, la fonction **`interp2`** dont la syntaxe est :

```
zi = interp2(x, y, z, xi, yi, 'method')
```

`x` et **`y`** sont les tableaux de même dimension qui contient les coordonnées des points dont les valeurs de **`z`** sont données pour chaque coordonnée.
`zi` est le tableau des résultats calculés pour les tableaux de coordonnées **`xi`** et **`yi`**.
`'method'` est un argument qui spécifier le type d'interpolation par exemple **`linear`** pour une interpolation bilinéaire ou **`spline`** pour une interpolation spline cubique de type "*Not-a-Knot*" utilisée sur un seul axe et nécessite que la dimension de **`x`** et **`y`** est égale à 4 au minimum.

Pour l'exemple précédent :

```
>> x=[2 9];
>> y=[1 6];
>> z=[60 57.5;55 70];
>> interp2(x,y,z,5.25,4.8)
ans =
   61.2143
```

Pour une interpolation tridimensionnelle, il y a aussi la fonction **`interp3`** qui fait la même chose que la fonction précédente, sa syntaxe en termes d'interpolation *tri-linéaire* est :

```
wi = interp3(x, y, z, v, xi, yi, zi)
```

14. Exercices résolus

Certains exercices traités nécessitent de la programmation (script ou fonction) surtout pour les cas des tableaux de grandes dimensions.

Exercice 1

Soit le tableau des données :

x	0	1.8	5	6	8.2	9.2	12
y	26	16.415	5.375	3.5	2.015	2.24	8

Utiliser les méthodes d'interpolation linéaire, polynomiale et spline cubique pour estimer y_0 pour $x_0 = 3.5$ (Utiliser les scripts ci-dessus)

Pour résoudre cet exercice utilisant les fonctions MATLAB suivants :
`Interp2Newton` fonction MATLAB qui utilise l'interpolation polynomiale,
`NaturalSpline` fonction MATLAB qui utilise l'interpolation spline cubique naturelle.

Pour l'interpolation linéaire, soit :

```
>> x=[1.8 5];y=[16.415 5.375];
>> x0=3.5;
>> y0=Interp2Newton(x,y,x0)
y0 =
   10.5500
```

Pour l'interpolation polynomiale, soit :

```
>> x=[0 1.8 5 6 8.2 9.2 12];
>> y=[26 16.415 5.375 3.5 2.015 2.24 8];
>> x0=3.5;
>> y0=Interp2Newton(x,y,x0)
y0 =
   9.6981
```

Pour l'interpolation polynomiale, soit :

```
>> x=[0 1.8 5 6 8.2 9.2 12];
>> y=[26 16.415 5.375 3.5 2.015 2.24 8];
>> x0=3.5;
>> y0=NaturalSpline(x,y,x0)
y0 =
   9.3945
```

Exercice 2

La fonction de Bessel (Watson, 1966) est souvent utilisée dans les différents domaines de la physique. Le tableau ci-dessous illustre les valeurs de la fonction de Bessel de première espèce de degré de liberté 1 évalué à l'ordre 0,

x	1.8	2.0	2.2	2.4	2.6
$J_1(x)$	0.5815	0.5767	0.5560	0.5202	0.4708

Estimer $J_1(2.1)$ avec une interpolation polynomiale de troisième et quatrième degré et comparer les résultats obtenus avec celle de la vraie valeur (en utilisant la fonction **`besselj`** de MATLAB).

Le polynôme d'interpolation de 3ième degré suivant la formule de Lagrange est :

$$P_3(x)=\frac{(x-2.0)(x-2.2)(x-2.4)}{(1.8-2.0)(1.8-2.2)(1.8-2.4)}0.5815+\frac{(x-1.8)(x-2.2)(x-2.4)}{(2.0-1.8)(2.0-2.2)(2.0-2.4)}0.5767+$$
$$\frac{(x-1.8)(x-2.0)(x-2.4)}{(2.2-1.8)(2.2-2.0)(2.2-2.4)}0.5560+\frac{(x-1.8)(x-2.0)(x-2.2)}{(2.4-1.8)(2.4-2.0)(2.4-2.2)}0.5202$$

Le polynôme d'interpolation de 4ième degré suivant la formule de Lagrange est :

$$P_4(x)=\frac{(x-2.0)(x-2.2)(x-2.4)(x-2.6)}{(1.8-2.0)(1.8-2.2)(1.8-2.4)(1.8-2.6)}0.5815+\frac{(x-1.8)(x-2.2)(x-2.4)(x-2.6)}{(2.0-1.8)(2.0-2.2)(2.0-2.4)(2.0-2.6)}0.5767+$$
$$\frac{(x-1.8)(x-2.0)(x-2.4)(x-2.6)}{(2.2-1.8)(2.2-2.0)(2.2-2.4)(2.2-2.6)}0.5560+\frac{(x-1.8)(x-2.0)(x-2.2)(x-2.6)}{(2.4-1.8)(2.4-2.0)(2.4-2.2)(2.4-2.6)}0.5202+$$
$$\frac{(x-1.8)(x-2.0)(x-2.2)(x-2.4)}{(2.6-1.8)(2.6-2.0)(2.6-2.2)(2.6-2.4)}0.4708$$

alors,

$$J_1(2.1)\approx P_3(2.1)=0.5683$$
$$J_1(2.1)\approx P_4(2.1)=0.5683$$

Il parait que l'utilisation de la méthode d'interpolation de Lagrange est fastidieuse. La méthode des différences divisée parait très rapide et très flexible. D'après la méthode des différences divisées de Newton le polynôme d'interpolation du quatrième degré $P_4(x)$ est peut-être exprimé ainsi :

$$P_4(x)=P_3(x)+a_4(x-1.8)(x-2.0)(x-2.2)(x-2.4)$$

$P_3(x)$ est le polynôme d'interpolation du troisième degré, il est exprimé par :

$$P_3(x)=a_0+a_1(x-1.8)+a_2(x-1.8)(x-2.0)+a_3(x-1.8)(x-2.0)(x-2.2)$$

a_0, a_1, a_2, a_3 et a_4 sont les différences divisées, soit le tableau 4.3.

Tab. 4.3 : Différences divisées.

x_i	$f[x_i]$	$f[x_i,x_{i-1}]$	$f[x_i,x_{i-1}]$	$f[x_i,x_{i-1},x_{i-2},x_{i-3}]$	$f[x_i,x_{i-1},x_{i-2},x_{i-3},x_{i-4}]$
1.8	**0.5815**				
2.0	0.5767	**-0.0240**			
2.2	0.5560	-0.1035	**-0.1987**		
2.4	0.5202	-0.1790	-0.1888	**0.0167**	
2.6	0.4708	-0.2470	-0.1700	0.0313	**0.0182**

Du tableau ci-dessus, $a_0=0.5815$, $a_1=-0.0240$, $a_2=-0.1987$, $a_3=0.0167$ et $a_4=0.0182$ alors,

$$J_1(2.1)\approx P_3(2.1)=0.5683$$
$$J_1(2.1)\approx P_4(2.1)=0.5683$$

Avec la fonction **besselj** de MATLAB, on peut avoir le même résultat, soit :

```
>> besselj(1,2.1)
ans =
    0.5683
```

Exercice 3

Soit les données,

x	1	2	2.5	3	4	5
$f(x)$	1	5	7	8	2	1

Tracer les graphs d'interpolation, avec :
(a) Spline cubique naturelle,
(b) Spline cubique type not-a-knot,
(c) Interpolation de l'Hermite.

Pour résoudre cet exercice on utilise les fonctions MATLAB, **NaturalSpline**, **NotAKnotSpline** ou **spline** et **pchip** sur l'intervalle [1, 5]. Soit le script suivant qui résoudre cet exercice,

```
% Les données x et f(x)
x=[1 2 2.5 3 4 5]; f2x=[1 5 7 8 2 1];
% Tracer les points des données
% Balayage de l'intervalle [1, 5]
X=min(x):0.001:max(x);
% (a) Ajustement avec spline cubique naturelle
Ya=NaturalSpline(x,f2x,X);
% (b) Ajustement avec spline cubique type Not-a-Knot
Yb=spline(x,f2x,X);
hold on
plot(X,Yb,'-r'); hold off
% (c) Ajustement avec interpolation de Hermite
Yc=pchip(x,f2x,X);
```

La comparaison entre les trois ajustements est illustrée dans la figure ci-dessous (Fig. 4.11). la comparaison montre que les trois ajustements sont identiques pour les trois premiers points, car elles existent sur une ligne droite, tandis que pour les trois points qui suivent, il y a des différences entre les ajustements car ces points n'appartiennent pas à une ligne droite.

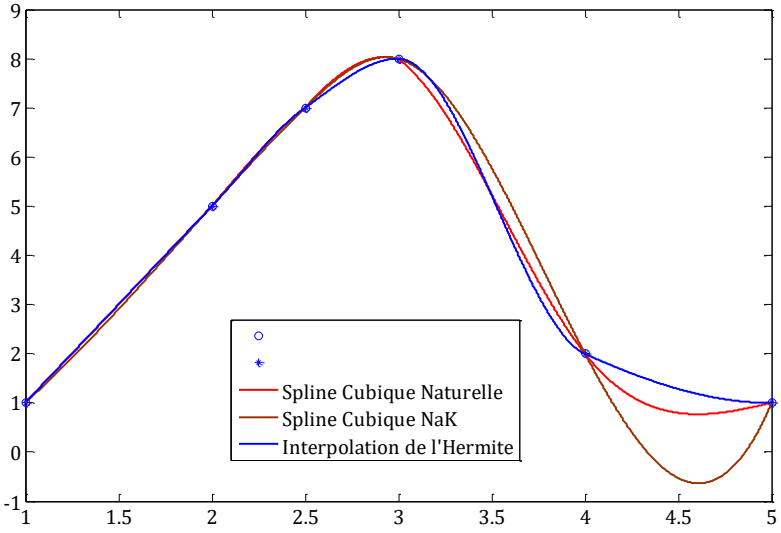


Fig. 4.11 : Comparaison des trois ajustements précédents.

Exercice 4

Déterminer la seconde différence divisée de la fonction $f(x)=1/x$ pour les points dont l'abscisse est *a*, *b* et *c*.

$$f[a,\ b]=\frac{f(b)-f(a)}{b-a}=\frac{1}{b-a}\left(\frac{1}{b}-\frac{1}{a}\right)=\frac{a-b}{(b-a)ab}=-\frac{1}{ab}$$

$$f[a,\ c]=\frac{f(c)-f(a)}{c-a}=\frac{1}{c-b}\left(\frac{1}{c}-\frac{1}{b}\right)=\frac{b-c}{(c-b)bc}=-\frac{1}{bc}$$

$$f[a,\ b,\ c]=\frac{f[a,\ c]-f[a,\ b]}{c-a}=\frac{1}{c-a}\left(\frac{1}{bc}+\frac{1}{ab}\right)=\frac{c-a}{(c-a)abc}=\frac{1}{abc}$$

Exercice 5

La fonction $f(x)=\sin x$, déterminer le polynôme d'interpolation pour les nœuds $x_0 = 0$ $x_1 = \pi/2$ et $x_2 = \pi$ et estimer la valeur de $\sin(\pi/4)$ par interpolation et calculer l'erreur d'interpolation pour cette valeur.

D'après la formule Lagrange, le polynôme d'interpolation passant par ces nœuds est :

$$P_2(x)=\frac{(x-\pi/2)(x-\pi)}{(0-\pi/2)(0-\pi)}\sin(0)+\frac{(x-0)(x-\pi)}{(\pi/2-0)(\pi/2-\pi)}\sin(\pi/2)+\frac{(x-0)(x-\pi/2)}{(\pi-0)(\pi-\pi/2)}\sin(\pi)$$

ou,

$$P_2(x)=-\left(\frac{2}{\pi}\right)^2 x(x-\pi)$$

L'estimation de $\sin(\pi/4)$ par interpolation est :

$$\sin\left(\frac{\pi}{4}\right)\approx P_2\left(\frac{\pi}{4}\right)=-\left(\frac{2}{\pi}\right)^2\frac{\pi}{4}\left(\frac{\pi}{4}-\pi\right)=\frac{3}{4}$$

L'erreur d'interpolation est explicitée par :

$$E(x)=\sin(\pi/4)-P_2(x)=\frac{x(x-\pi/2)(x-\pi)}{3!}f'''(\zeta),\ \ 0\le\zeta\le\pi$$

ou,

$$E(x)=\sin(\pi/4)-P_2(x)=-\frac{x(x-\pi/2)(x-\pi)}{3!}\cos(\zeta),\ \ 0\le\zeta\le\pi$$

soit,

$$\left|E(\pi/4)\right|=\left|\sin(\pi/4)-P_2(\pi/4)\right|=\left|\frac{\pi/4(\pi/4-\pi/2)(\pi/4-\pi)}{3!}\right|\left|\cos(\zeta)\right|=\frac{\pi^3}{2^7}\left|\cos(\zeta)\right|,\ \ 0\le\zeta\le\pi$$

Il est clair que,

$$\left|E(\pi/4)\right|=\left|\sin(\pi/4)-P_2(\pi/4)\right|\le\frac{\pi^3}{2^7}$$

on sait que,

$$\sin(\pi/4)=0.75$$

alors,

$$\left|E(\pi/4)\right|=\left|0.75-3/4\right|=0.0429<\frac{\pi^3}{2^7}$$

Donc l'erreur est bien vérifiée et inférieur à $\pi^3/2^7$.

Exercice 6

Soit le tableau des résultats x et $y(x)$:

x	0.1	0.3	0.5	0.7	0.9	1.1	1.3
y	0.003	0.067	0.148	0.248	0.370	0.518	0.697

a. Calculer $y(0.54)$ par interpolation cubique qui commence à partir de $x = 0.1$.
b. Répétez la partie (a) mais commencez à partir de $x = 0.3$. Cela devrait-il être une meilleure valeur ?
c. Quel est le degré minimum de polynôme interpolant tous les données ?
d. Construire le tableau des différences divisées. En quoi cela diffère-t-il ?

a. Le calcul de $y(0.54)$ par interpolation cubique qui commence à partir de $x = 0.1$, c'est à dire on considère les données :

x	0.1	0.3	0.5	0.7
y	0.003	0.067	0.148	0.248

suivant l'interpolation de Lagrange :

$$y(x)=\frac{(x-0.3)(x-0.5)(x-0.7)}{(0.1-0.3)(0.1-0.5)(0.1-0.7)}0.003+\frac{(x-0.1)(x-0.5)(x-0.7)}{(0.3-0.1)(0.3-0.5)(0.3-0.7)}0.067$$
$$+\frac{(x-0.1)(x-0.3)(x-0.7)}{(0.5-0.1)(0.5-0.3)(0.5-0.7)}0.148+\frac{(x-0.1)(x-0.3)(x-0.5)}{(0.7-0.1)(0.7-0.3)(0.7-0.5)}0.248$$

alors,

$$y(0.54)=\frac{(0.54-0.3)(0.54-0.5)(0.54-0.7)}{(0.1-0.3)(0.1-0.5)(0.1-0.7)}0.003+\frac{(0.54-0.1)(0.54-0.5)(0.54-0.7)}{(0.3-0.1)(0.3-0.5)(0.3-0.7)}0.067$$
$$+\frac{(0.54-0.1)(0.54-0.3)(0.54-0.7)}{(0.5-0.1)(0.5-0.3)(0.5-0.7)}0.148+\frac{(0.54-0.1)(0.54-0.3)(0.54-0.5)}{(0.7-0.1)(0.7-0.3)(0.7-0.5)}0.248=0.1664$$

b. Le calcul de $y(0.54)$ par interpolation cubique qui commence à partir de $x = 0.3$, c'est à dire on considère les données :

x	0.3	0.5	0.7	0.9
y	0.067	0.148	0.248	0.370

suivant l'interpolation de Lagrange :

$$y(x)=\frac{(x-0.5)(x-0.7)(x-0.9)}{(0.3-0.5)(0.3-0.7)(0.3-0.9)}0.067+\frac{(x-0.3)(x-0.7)(x-0.9)}{(0.5-0.3)(0.5-0.7)(0.5-0.9)}0.148$$
$$+\frac{(x-0.3)(x-0.5)(x-0.9)}{(0.7-0.3)(0.7-0.5)(0.7-0.9)}0.248+\frac{(x-0.3)(x-0.5)(x-0.7)}{(0.9-0.3)(0.9-0.5)(0.9-0.7)}0.370$$

alors,

$$y(0.54)=\frac{(0.54-0.5)(0.54-0.7)(0.54-0.9)}{(0.3-0.5)(0.3-0.7)(0.3-0.9)}0.067+\frac{(0.54-0.3)(0.54-0.7)(0.54-0.9)}{(0.5-0.3)(0.5-0.7)(0.5-0.9)}0.148$$
$$+\frac{(0.54-0.3)(0.54-0.5)(0.54-0.9)}{(0.7-0.3)(0.7-0.5)(0.7-0.9)}0.248+\frac{(0.54-0.3)(0.54-0.5)(0.54-0.7)}{(0.9-0.3)(0.9-0.5)(0.9-0.7)}0.370=0.1664$$

Entre les deux méthodes, les deux résultats sont les mêmes.

c. Le degré minimum du polynôme d'interpolation en considérons tous les données est 3.
d. Le tableau des différences divisées est :

Tab. 4.4 : Différences divisées.

i	x_i	$f[x_i]$	$f[x_i, x_{i-1}]$	$f[x_i, x_{i-1}]$	$f[x_i, \ldots, x_{i-3}]$	$f[x_i, \ldots, x_{i-4}]$	$f[x_i, \ldots, x_{i-5}]$	$f[x_i, \ldots, x_{i-6}]$
0	0.1	**0.003**						
1	0.3	0.067	**0.320**					
2	0.5	0.148	0.405	**0.213**				
3	0.7	0.248	0.500	0.238	**0.042**			
4	0.9	0.37	0.610	0.275	0.062	**0.026**		
5	1.1	0.518	0.740	0.325	0.083	0.026	**9.489E-15**	
6	1.3	0.697	0.895	0.388	0.104	0.026	-7.466E-15	**-1.413E-14**

D'après la formule d'interpolation de Newton des différences divisées $y(x)$ est exprimé par :

$$y(x) = 0.003 + 0.320(x-0.1) + 0.213(x-0.1)(x-0.3) + 0.042(x-0.1)(x-0.3)(x-0.5)$$
$$+ 0.026(x-0.1)(x-0.3)(x-0.5)(x-0.7) + 9.489\cdot 10^{-15}(x-0.1)(x-0.3)(x-0.5)(x-0.7)(x-0.9)$$
$$- 1.413\cdot 10^{-14}(x-0.1)(x-0.3)(x-0.5)(x-0.7)(x-0.9)(x-1.1)$$

par conséquent,

$$y(0.54) = 0.1664$$

qui est le même résultat obtenu.

Exercice 7

Soit la fonction $f(x) = \exp(-2x)$, déterminer l'erreur maximale d'interpolation de la fonction $f(x)$ pour les nœuds $x_0 = 0$, $x_1 = 1/9$, $x_2 = 2/9$, $x_3 = 3/9$, …, $x_8 = 8/9$ et $x_9 = 1$.

L'erreur maximale d'approximation de la fonction $f(x)$ par le polynôme d'interpolation $P_n(x)$ pour des nœuds uniformes de l'intervalle $I = [x_0, x_n]$ est exprimée par :

$$E_{\max} = \frac{1}{4(n+1)}\left(\frac{x_n - x_0}{n}\right)^{n+1} \max_{\forall \xi \in I} \left| f^{(n+1)}(\xi) \right|$$

or,

$$f(x) = \exp(-2x)$$

alors,

$$f'(x) = -2\exp(-2x),\ f''(x) = 2\cdot 2\exp(-2x),\ f'''(x) = -2\cdot 2\cdot 2\exp(-2x),\ \ldots,\ f^{(k)}(x) = (-2)^k \exp(-2x)$$

Puisque, $n = 9$, $x_0 = 0$, $x_9 = 1$ et $\max_{\forall \xi \in I} \left| f^{(10)}(\xi) \right| = 2^{10}$

soit :

$$E_{\max} = \frac{1}{40}\left(\frac{2}{9}\right)^{10} = 7.34\cdot 10^{-9}$$

Exercice 8

Soit les données :

x	-1	0	1	2
$f(x)$	2	2	2	26
$f'(x)$	2	0	2	68

Déterminer le polynôme d'interpolation de l'Hermite et calculer $f(0.5)$.

On a 4 nœuds $x_0=-1,\ x_1=0,\ x_2=1, x_3=2$ alors le polynôme d'interpolation de l'Hermite est de degré 7 est explicité par :

$$P(x)=\sum_{i=0}^{3}\left(-2L_i'(x_i)x+1+2x_iL_i'(x_i)\right)L_i^2(x)f(x_i)+\sum_{i=0}^{3}(x-x_i)L_i^2(x)f'(x_i)$$

avec,

$$L_0(x)=\prod_{\substack{k=0\\k\neq 0}}^{3}\frac{x-x_k}{x_0-x_k}=\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}=-\frac{1}{6}(x-0)(x-1)(x-2)$$

$$\Rightarrow L_0'(x_0)=-\frac{11}{6}$$

$$L_1(x)=\prod_{\substack{k=0\\k\neq 0}}^{3}\frac{x-x_k}{x_1-x_k}=\frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}=\frac{1}{2}(x+1)(x-1)(x-2)$$

$$\Rightarrow L_1'(x_1)=-\frac{1}{2}$$

$$L_2(x)=\prod_{\substack{k=0\\k\neq 0}}^{3}\frac{x-x_k}{x_2-x_k}=\frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}=-\frac{1}{2}(x+1)(x-0)(x-2)$$

$$\Rightarrow L_2'(x_2)=\frac{1}{2}$$

$$L_3(x)=\prod_{\substack{k=0\\k\neq 0}}^{3}\frac{x-x_k}{x_3-x_k}=\frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}=\frac{1}{6}(x+1)(x-0)(x-1)$$

$$\Rightarrow L_3'(x_3)=\frac{11}{6}$$

soit,

$$\begin{aligned}P(x)=&\left(-2L_0'(x_0)x+1+2x_0L_0'(x_0)\right)L_0^2(x)f(x_0)+\left(-2L_1'(x_1)x+1+2x_1L_1'(x_1)\right)L_1^2(x)f(x_1)\\&+\left(-2L_2'(x_2)x+1+2x_2L_2'(x_2)\right)L_2^2(x)f(x_2)+\left(-2L_3'(x_3)x+1+2x_3L_1'(x_3)\right)L_3^2(x)f(x_3)\\&+(x-x_0)L_0^2(x)f'(x_0)+(x-x_1)L_1^2(x)f'(x_1)+(x-x_2)L_2^2(x)f'(x_2)+(x-x_3)L_3^2(x)f'(x_3)\end{aligned}$$

Numériquement,

$$\begin{aligned}P(x)=&\left(\frac{11}{3}x+\frac{14}{3}\right)\left(-\frac{1}{6}x(x-1)(x-2)\right)^2\cdot 2+(x+1)\left(\frac{1}{2}(x+1)(x-1)(x-2)\right)^2\cdot 2\\&+(-x+2)\left(-\frac{1}{2}(x+1)x(x-2)\right)^2\cdot 2+\left(-\frac{11}{3}x+\frac{25}{3}\right)\left(\frac{1}{6}x(x+1)(x-1)\right)^2\cdot 26\end{aligned}$$

$$+(x+1)\left(-\frac{1}{6}(x-1)x(x-2)\right)^2\cdot 2+x\left(\frac{1}{2}(x+1)(x-1)(x-2)\right)^2\cdot 0+(x-1)\left(-\frac{1}{2}(x+1)x(x-2)\right)^2\cdot 2$$
$$+(x-2)\left(\frac{1}{6}(x+1)x(x-1)\right)^2\cdot 68$$

$$P(x)=\left(\frac{28}{3}x+\frac{34}{3}\right)\left(-\frac{1}{6}x(x-1)(x-2)\right)^2+2(x+1)\left(\frac{1}{2}(x+1)(x-1)(x-2)\right)^2$$
$$+2\left(-\frac{1}{2}(x+1)x(x-2)\right)^2+\left(-\frac{82}{3}x+\frac{244}{3}\right)\left(\frac{1}{6}x(x+1)(x-1)\right)^2$$
$$P(x)=x^5-x^3+2$$

Alors,

$$P(0.5)=(0.5)^5-(0.5)^3+2=1.90625$$

Exercice 9

Est-ce que les fonctions suivantes sont des splines cubiques :

$$a)\ f(x)=\begin{cases}x^3-2x+3 & \text{si } 0\le x\le 1,\\ 2x^3-3x^2+x+2 & \text{si } 1\le x\le 2.\end{cases},\qquad b)\ f(x)=\begin{cases}5x^3-3x^2+1 & \text{si } 0\le x\le 1,\\ 2x^3+6x^2-9x+4 & \text{si } 1\le x\le 2.\end{cases}$$

$a)$ Pour le cas de la fonction,

$$f(x)=\begin{cases}S_1(x)=x^3-2x+3 & \text{si } 0\le x\le 1,\\ S_2(x)=2x^3-3x^2+x+2 & \text{si } 1\le x\le 2.\end{cases}$$

Dans le nœud intermédiaire $x=1$,

$$f(1)=\begin{cases}S_1(1)=1-2+3=2,\\ S_2(1)=2-3+1+2=2.\end{cases}\Rightarrow\ S_1(1)=S_2(1)$$

Pour la dérivée,

$$f'(x)=\begin{cases}S_1'(x)=3x^2-2 & \text{si } 0\le x\le 1,\\ S_2'(x)=6x^2-6x+1 & \text{si } 1\le x\le 2.\end{cases}\Rightarrow\ f'(1)=\begin{cases}S_1'(1)=3-2=1,\\ S_2'(1)=6-6+1=1.\end{cases}\Rightarrow\ S_1'(1)=S_2'(1)$$

Pour la dérivée seconde,

$$f''(x)=\begin{cases}S_1''(x)=6x & \text{si } 0\le x\le 1,\\ S_2''(x)=12x-6 & \text{si } 1\le x\le 2.\end{cases}\Rightarrow\ f''(1)=\begin{cases}S_1''(1)=6,\\ S_2''(1)=12-6=6.\end{cases}\Rightarrow\ S_1''(1)=S_2''(1)$$

Puisque ces conditions dans le seul nœud intermédiaire $x=1$ sont bien vérifiées donc cette fonction est bien une spline cubique.

$b)$ Pour le cas de la fonction,

$$f(x)=\begin{cases}S_1(x)=5x^3-3x^2+1 & \text{si } 0\le x\le 1,\\ S_2(x)=2x^3+6x^2-9x+4 & \text{si } 1\le x\le 2.\end{cases}$$

Dans le nœud intermédiaire $x=1$,

$$f(1)=\begin{cases}S_1(1)=5-3+1=3,\\ S_2(1)=2+6-9+4=3.\end{cases}\Rightarrow\ S_1(1)=S_2(1)$$

Pour la dérivée,

$$f'(x)=\begin{cases}S_1'(x)=15x^2-6x\ \text{ si }\ 0\le x\le 1,\\ S_2'(x)=6x^2+12x-9\ \text{ si }\ 1\le x\le 2.\end{cases}\Rightarrow\ f'(1)=\begin{cases}S_1'(1)=15-6=9,\\ S_2'(1)=6+12-9=9.\end{cases}\Rightarrow\ S_1'(1)=S_2'(1)$$

Pour la dérivée seconde,

$$f''(x)=\begin{cases}S_1''(x)=30x-6\ \text{ si }\ 0\le x\le 1,\\ S_2''(x)=12x+12\ \text{ si }\ 1\le x\le 2.\end{cases}\Rightarrow f''(1)=\begin{cases}S_1''(1)=30-6=24,\\ S_2''(1)=12+12=24.\end{cases}\Rightarrow\ S_1''(1)=S_2''(1)$$

Puisque ces conditions dans le seul nœud intermédiaire $x=1$ sont bien vérifiées donc cette fonction est bien une spline cubique.

15. Exercices supplémentaires

Exercice 1

On considère la fonction :

$$f(x)=\frac{4}{1-x}$$

Ecrire le polynôme de Lagrange $p(x)$ aux points $x_0=-1$, $x_1=0$, $x_2=2$ et $x_3=3$. Evaluer l'erreur au point $x=1+\sqrt{5}$. Tracer les courbes $f(x)$ et $p(x)$.

Ecrire pour la fonction, le polynôme d'Hermite vérifiant

$$P(0)=f(0)=4$$
$$P(2)=f(2)=-4$$
$$P'(0)=f'(0)=4$$
$$P'(2)=f'(2)=4$$

Tracer les courbes $f(x)$ et $p(x)$. Evaluer l'erreur au point $x=1+\sqrt{5}$.

Exercice 2

On va approcher la fonction :

$$f(x)=\frac{1}{1+x^2},\qquad -5\le x\le 5$$

Par une interpolation avec nœuds uniformes.

1. Tracer *f* sur l'intervalle [-5, 5].
2. Sur le même graphe, tracez les polynômes d'interpolation P_5 et P_{11}.
3. Que constatez-vous, que pouvez-vous en conclure ?

Exercice 3

On interpole sur [-1, 1] la fonction $f(x)=\sin(2\pi x)$ avec 17 nœuds uniformes. On génère ensuite des données perturbées $\tilde{f}(x_i)$ des $f(x_i)$ avec la condition :

$$\max_{i=0,\ \ldots,\ 16} \left|\tilde{f}(x_i) - f(x_i)\right| \le 9.5 \cdot 10^{-4}$$

Tracez et comparez les polynômes d'interpolation de f et $\tilde{f}$. Que peut 'on remarquer ?

Exercice 4

La fonction erreur est une fonction spéciale elle trouve son importance surtout dans le calcul des probabilités et dans la solution du problème de la conduction de la chaleur etc. Elle est définie par l'intégrale :

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du$$

Le tableau ci-contre donne les valeurs de la fonction $\operatorname{erf}(x)$ pour onze valeurs de l'intervalle [0, 1].

x	$\operatorname{erf}(x)$
0.0	0.000000000
0.1	0.112462916
0.2	0.222702589
0.3	0.328626759
0.4	0.428392355
0.5	0.520499878
0.6	0.603856091
0.7	0.677801194
0.8	0.742100965
0.9	0.796908212
1.0	0.842700793

a) Construire l'interpolation spline quadratique de la fonction erreur pour les nœuds $x_0 = 0$, $x_1 = 0.5$ et $x_2 = 1$, tracer le graph du polynôme d'interpolation et comparer avec les données du tableau.

b) Construire l'interpolation spline cubique naturelle de la fonction erreur pour les nœuds $x_0 = 0$, $x_1 = 0.3$, $x_2 = 0.7$ et $x_3 = 1$, tracer le graph du polynôme d'interpolation et comparer avec les données du tableau.

Exercice 5

La fonction $f(x) = \ln x$, déterminer le polynôme d'interpolation pour les nœuds x_0 = 1, x_1 = 2 et x_2 = 4 et estimer la valeur de ln 3 par et calculer l'erreur d'interpolation de cette valeur.

Exercice 6

On définit les polynômes de Tchebychev par la relation de récurrence:

$$(1 - x^2) T_n(x) = -nx T_n(x) + n T_{n-1}(x)$$

Démontrer les relations d'orthogonalité

$$\int_{-1}^{1} T_n(x) T_m(x) \frac{dx}{\sqrt{1-x^2}} = \frac{\pi}{2} \delta_{nm}, \quad n \neq 0$$

et pour n = 0

$$\int_{-1}^{1} T_0^2(x) \frac{dx}{\sqrt{1-x^2}} = \pi$$

Exercice 7

En utilisant l'interpolation spline cubique naturelle et l'interpolation d'Hermite pour les données citées dans le tableau ci-dessous.

x	0	100	200	400	600	800	1000
$f(x)$	0	0.82436	1.0	0.73576	0.40601	0.19915	0.09158

et comparer les résultats avec la fonction :

$$f(x)=\frac{x}{200}e^{1-x/200}$$

Exercice 8

Estimer par l'interpolation de l'Hermite $f(0.5)$ et $f'(1.5)$ en utilisant les données :

x	0	2	3
$f(x)$	2	16	80
$f'(x)$	-2	31	107

Exercice 9

Est-ce que les fonctions suivantes sont des splines cubiques :

$$f(x)=\begin{cases}3x^2+x+1 & \text{si } 0\le x\le 1,\\ 3x^2-5x+1 & \text{si } 1<x\le 2.\end{cases} \quad \text{et} \quad g(x)=\begin{cases}x^3-3x^2+1 & \text{si } 0\le x\le 1,\\ x^3-2 & \text{si } 1\le x\le 2.\end{cases}$$

Exercice 10

Trouver les valeurs de α et β pour que la fonction suivante soit une spline cubique,

$$f(x)=\begin{cases}\alpha x^3+\beta x^2+2x & \text{si } -1\le x\le 0,\\ 3x^3+x^2+2x & \text{si } 0\le x\le 1.\end{cases}$$

Chapitre 5.

Intégration numérique

1. Motivation

Soit $f(x)$ une fonction définie et continue sur l'intervalle $[a, b]$ et $F(x)$ sa primitive $F'(x) = f(x)$, l'intégrale définie de la fonction $f(x)$ dans les limites de a à b peut être calculée d'après la formule de Newton-Leibniz :

$$I(f) = \int_a^b f(x)dx = F(b) - F(a)$$

Mais dans de nombreux cas la primitive $F(x)$ est trop compliquée ou ne peut s'obtenir à l'aide de procédés élémentaires ; il en résulte que le calcul de l'intégrale définie d'après la formule de Newton-Leibniz peut être trop difficile ou même pratiquement impossible, soit les deux exemples :

$$I_1 = \int_{-1}^{1} e^{-x^2} dx \quad \text{et} \quad I_2 = \int_1^2 \frac{\sin x}{x} dx$$

L'intégration numérique, consiste à rechercher la valeur de l'intégrale définie à partir de plusieurs valeurs de la fonction $f(x)$. Le calcul numérique d'une telle intégrale s'appelle aussi quadrature mécanique (Démidovitch et Maron, 1979) d'où le calcul de l'intégrale par les formules de quadrature.

On appelle formule de quadrature une expression linéaire dont l'évaluation fournit une valeur approchée de l'intégrale sur intervalle. Une transformation affine permet de transposer la formule sur un morceau particulier. Les formules de quadrature sont en effet obtenues à l'aide de la substitution de la fonction par une approximation, par une fonction proche dont l'intégrale peut être déterminée algébriquement par la combinaison linéaire :

$$\int_a^b w(x) f(x)dx \approx \sum_{j=1}^{n} \lambda_j f(x_j)$$

$w(x)$, est une fonction de pondération, λ_1, λ_2, …, λ_n sont appelés les coefficients poids.

La réalisation d'une quadrature, consiste à remplacer la fonction $f(x)$ sur l'intervalle $[a, b]$ par une fonction d'interpolation $P(x)$ pour admettre approximativement :

$$\int_a^b w(x) f(x)dx \approx \int_a^b P(x)dx$$

La fonction $P(x)$ doit être telle que le calcul de l'intégrale $\int_a^b P(x)dx$ soit immédiat.

2. Formule de quadrature de Newton-Cotes

Soit $f(x)$ une fonction continue sur l'intervalle $[a,\ b]$, il faille calculer l'intégrale :

$$I(f)=\int_a^b f(x)dx$$

Soit h le pas de discrétisation :

$$h=(b-a)/n$$

C'est-à-dire l'intervalle $[a, b]$ est subdivisé en n+1 nœuds équidistants, avec :

$$x_0=a,\quad x_j=x_0+jh\ \ (j=1,\ 2,\ 3,\ \ldots,\ n-1)\ \text{ et }\ x_n=b$$

D'après la méthode d'interpolation de Lagrange, la fonction $f(x)$ est peut-être approchée par un polynôme de degré n :

$$P_n(x)=\sum_{j=0}^{n}L_j(x)f(x_j)$$

où,

$$P_n(x)=\sum_{j=0}^{n}\frac{\psi(x)}{(x-x_j)\ \psi'(x_j)}f(x_j)$$

avec,

$$\frac{\psi(x)}{(x-x_j)\ \psi'(x_j)}=\frac{(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)}$$

Soit le changement de variable,

$$u=\frac{x-x_0}{h}\ \ \Rightarrow\ \ x=x_0+hu\ \ \text{et}\ \ dx=hdu$$

Donc,

$$x-x_j=x_0+hu-(x_0+jh)=h(u-j),\ \ (\forall\ j=0,\ 2,\ 3,\ \ldots,\ n)$$

Alors,

$$\psi(x)=h^{n+1}u(u-1)(u-2)\cdots(u-n)=h^{n+1}u^{[n+1]}$$

$u^{[n+1]}$, la puissance généralisée (n+1)-ième de u (Démidovitch et Maron, 1979)

De même,

$$\psi'(x_j)=h^n j(j-1)\cdots3\cdot2\cdot1\cdot(-1)\cdot(-2)\cdot(-3)\cdots(j-n)=h^n j!(-1)^{n-j}(n-j)!$$

Alors,

$$L_j(x)=\frac{(-1)^{n-j}}{j!(n-j)!}\frac{u^{[n+1]}}{u-j}$$

Soit,

$$P_n(x)=\sum_{j=0}^{n}\frac{(-1)^{n-j}}{j!(n-j)!}\frac{u^{[n+1]}}{u-j}f(x_j)$$

L'intégrale de la fonction $f(x)$ est peut-être approchée par l'intégrale du son polynôme d'interpolation, ainsi,

$$I(f)=h\int_0^n f(x_0+hu)du\approx\int_a^b\sum_{j=0}^{n}\frac{(-1)^{n-j}}{j!(n-j)!}\frac{u^{[n+1]}}{u-j}f(x_j)dx=\sum_{j=0}^{n}\left(\frac{(-1)^{n-j}h}{j!(n-j)!}\int_0^n\frac{u^{[n+1]}}{u-j}du\right)f(x_j)\equiv\sum_{j=0}^{n}w_j f(x_j)$$

avec,

$$w_j = h\frac{(-1)^{n-j}}{j!(n-j)!}\int_0^n \frac{u^{[n+1]}}{u-j}du = \frac{(b-a)}{n}\frac{(-1)^{n-j}}{j!(n-j)!}\int_0^n \frac{u^{[n+1]}}{u-j}du \equiv (b-a)H_j$$

w_j sont les *facteurs poids* et H_j les *coefficients de Cotes.*

soit,

$$I(f) = \int_a^b f(x)dx \approx (b-a)\sum_{j=0}^{n} H_j f(x_j)$$

Les relations suivantes dites conditions de consistances concernant les coefficients de Cotes sont bien vérifiées,

$$\sum_{j=0}^{n} H_j = 1 \text{ et } H_j = H_{n-j}$$

L'erreur ou le reste qui peut être produit suite au calcul approché de l'intégrale $I(f)$ est exprimé par :

$$R_n(I) = \int_a^b f(x)dx - \int_a^b P_n(x)dx = \int_a^b \left[f(x) - P_n(x)\right]dx$$

or,

$$E(x) = f(x) - P_n(x) = \frac{\psi(x)}{(n+1)!}f^{(n+1)}(\xi), \text{ avec } \xi \in [a,\ b]$$

alors, le reste d'intégration $R(h)$ est :

$$R(I) = \frac{1}{(n+1)!}f^{(n+1)}(\xi)\int_a^b \psi(x)dx$$

or, $x = x_0 + hu \implies dx = hdu$ alors,

$$R(I) = \frac{h^{n+2}}{(n+1)!}f^{(n+1)}(\xi)\int_0^n u^{[n+1]}du = \frac{h^{2\eta+3}}{(2\eta+2)!}f^{(2\eta+2)}(\xi)\int_0^{2\eta+1} u^{[2\eta+2]}du,\ \eta = 0,\ 1,\ 2,\ \ldots$$

car si $n = 2\eta,\ \forall\eta = 1,\ 2,\ \ldots$, l'intégrale,

$$\int_0^{2\eta} u^{[2\eta+1]}du = 0$$

3. Méthode des trapèzes

3.1. Règle du trapèze

L'application de la formule du paragraphe précédent pour *n* =1, conduit donc à,

$$H_j = \frac{(-1)^{1-j}}{j!(1-j)!}\int_0^1 \frac{u^{[2]}}{u-j}du$$

Alors,

$$H_0 = -\int_0^1 \frac{u(u-1)}{u}du = \frac{1}{2}, \qquad H_1 = \int_0^1 \frac{u(u-1)}{u-1}du = \frac{1}{2}$$

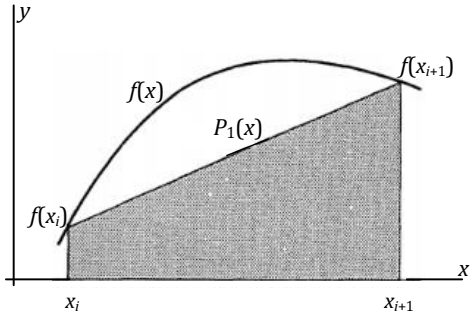


Fig. 5.1 : Règle du trapèze.

d'où,

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{(x_{i+1}-x_i)}{2}\left[f(x_i)+f(x_{i+1})\right]$$

C'est la règle du trapèze pour calculer approximativement l'intégrale définie sur le sous intervalle $[x_i,\ x_{i+1}]$ (Fig. 5.1).

La règle du trapèze, consiste donc à approcher la fonction $f(x)$ par un polynôme d'interpolation linéaire de premier degré et approcher l'intégrale $I(f)$ par l'aire du trapèze (Fig. 5.1). L'aire compris entre la droite supérieure limitant le trapèze et la courbe de la fonction, c'est donc le reste qui peut être déterminé pour l'intervalle $[x_i,\ x_{i+1}]$ par la formule du reste :

$$R_i = \frac{h^3}{2!}f''(\xi_i)\int_0^1 u^{[2]}du = \frac{h^3}{2}f''(\xi_i)\int_0^1 u(u-1)\,du = -\frac{h^3}{12}f''(\xi_i),\quad x_i \le \xi_i \le x_{i+1}$$

Par conséquent, l'erreur d'intégration sur l'intervalle $[x_i,\ x_{i+1}]$ est exprimé par :

$$E_i^T = \frac{h^3}{12}\left|f''(\xi_i)\right|,\quad x_i \le \xi_i \le x_{i+1}$$

3.2. Formule composée

Dans la pratique, l'intervalle [*a*, *b*] est divisé en *n* parties équidistants $[x_0,\ x_1]$, $[x_1,\ x_2]$, ..., $[x_{n-1},\ x_n]$ et appliquer à chacune d'elles la règle du trapèze (Fig. 5.2).

$$I(f) = \int_a^b f(x)dx = \int_{x_0=a}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \ldots + \int_{x_{n-1}}^{x_n} f(x)dx$$

c'est-à-dire,

$$I(f) \approx \frac{(x_1-x_0)}{2}\left[f(x_0)+f(x_1)\right] + \frac{(x_2-x_1)}{2}\left[f(x_1)+f(x_2)\right] + \ldots + \frac{(x_n-x_{n-1})}{2}\left[f(x_{n-1})+f(x_n)\right]$$

or,

$$x_{I+1} - x_{I+1} = h$$

alors,

$$I(f) \approx h\left[\frac{f(x_0)}{2} + f(x_1) + f(x_2) + \ldots + f(x_{n-1}) + \frac{f(x_n)}{2}\right] = h\left[\frac{f(x_0)+f(x_n)}{2} + \sum_{j=1}^{n-1} f(x_j)\right] \equiv I_t(f,\ h)$$

Remarquons que les facteurs poids sont :

$w_0 = h/2,\quad w_j = h\ (j = 1,\ 2,\ 3,\ \ldots,\ n-1)$ et $w_n = h/2$

Par conséquent les coefficients de Cotes :

$H_0 = 1/2,\quad H_j = 1\ (j = 1,\ 2,\ 3,\ \ldots,\ n-1)$ et $H_n = 1/2$

Qui vérifient bien les conditions :

$$\sum_{j=0}^{n} H_j = 1 \text{ et } H_j = H_{n-j}$$

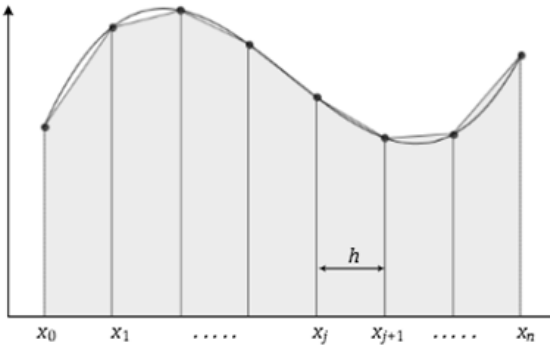


Fig. 5.2 : Méthode des trapèzes.

Soit la fonction **`Trap`** qui permet l'estimation de l'intégrale d'une fonction **`fun`** définie et continue sur un l'intervalle [**`a, b`**] :

```
function I = Trap(func,a,b,n,varargin)
% func = Nom de la fonction à intégrer
% a, b = Bornes d'intégration (a < b)
% n = Nombre de discrétisation (par défaut = 100)
% I = intégrale estimée par la méthode des trapèzes
if nargin<3,error('à la limite 3 arguments sont nécessaires'),end
if ~(b>a),error('Attention, il faut que a < b'),end
if nargin<4|isempty(n),n=100;end
x = a; h = (b-a)/n;
s=func(a,varargin{:});
for i = 1:(n-1)
x = x + h; s = s + 2*func(x,varargin{:});
end
s = s + func(b,varargin{:}); I = (b-a) * s/(2*n);
end
```

A titre d'exemple, soit à calculer l'intégrale :

$$I(f)=\int_1^3 x^2\ln(x)dx$$

alors,

```
>> f=@(x) (x.^2).*log(x);
>> Trap(f,1,3,100)
ans =
    6.9989
```

3.3. Estimation de l'erreur

Soit R_i^T le reste d'intégration par la règle du trapèze de la fonction $f(x)$ sur le sous intervalle $[x_i,\ x_{i+1}]$ de $[a,\ b]$, c'est-à-dire,

$$R_i^T=-\frac{h_i^3}{12}f''(\xi_i)\ \text{ avec }\ x_i\le\xi_i\le x_{i+1}$$

alors, le reste total d'intégration est :

$$R^T=-\frac{1}{12}\sum_{i=0}^{n-1}h_i^3 f''(\xi_i)$$

par conséquent, l'erreur d'intégration est estimée à :

$$E^T(I)=\frac{1}{12}\sum_{i=0}^{n-1}h_i^3\left|f''(\xi_i)\right|\le\frac{1}{12}\sum_{i=0}^{n-1}h_i^3\left\|f''\right\|_\infty^i$$

avec,

$$\left\|f''\right\|_\infty^i=\max_{\forall\ x_i\le x\le x_{i+1}}\left|f''(x)\right|$$

Dans le cas ou *h* est constant (cas des nœuds uniformes),

$$R^T(I)=-\frac{h^3}{12}\sum_{i=0}^{n-1}f''(\xi_i)=-\frac{h^2}{12}(b-a)\left\{\frac{1}{n}\sum_{i=0}^{n-1}f''(\xi_i)\right\}$$

La moyenne arithmétique qui existe dans la formule, permet de dire qu'il existe au moins ξ de $[a,\ b]$, tel que,

$$f''(\xi)\equiv\frac{1}{n}\sum_{i=0}^{n-1}f''(\xi_i)$$

soit alors,

$$R^T(I) = -\frac{h^2}{12}(b-a)f''(\xi)$$

L'erreur d'intégration dans ce cas est peut-être limitée par :

$$E^T(I) \le \frac{h^2}{12}(b-a)\|f''\|_\infty$$

qui est une erreur d'ordre 2.

4. Méthodes de Simpson

Il y'a deux type du règle de Simpson, 1/3 et 3/8 qui correspond respectivement à n = 2 et n = 3, c'est-dire l'intégrale $I(f)$ est approchée par l'intégrale $I(P_2)$ pour la règle de Simpson 1/3 et par l'intégrale $I(P_3)$ pour la règle de Simpson 3/8 ; $P_2(x)$ et $P_3(x)$, sont les polynômes d'interpolation de degré 2 et 3.

4.1. Méthode de Simpson 1/3

4.1.5. Règle de Simpson 1/3 et son reste

Soit x_i, x_{i+1} et x_{i+2} trois nœuds équidistants, la règle de Simpson 1/3 est obtenue de :

$$\int_{x_i}^{x_{i+2}} f(x)dx \approx (x_{i+2}-x_i)\sum_{j=0}^{2} H_j f(x_{i+j}) = (x_{i+2}-x_i)(H_0 f(x_i) + H_1 f(x_{i+1}) + H_2 f(x_{i+2}))$$

Sachant que,

$$H_j = \frac{1}{2}\frac{(-1)^{2-j}}{j!(2-j)!}\int_0^2 \frac{u(u-1)(u-2)}{u-j}du$$

alors,

$$H_0 = \frac{1}{4}\int_0^2 (u-1)(u-2)du = \frac{1}{6},\quad H_1 = -\frac{1}{2}\int_0^2 u(u-2)du = \frac{2}{3}$$

et

$$H_2 = \frac{1}{4}\int_0^2 u(u-1)du = \frac{1}{6}$$

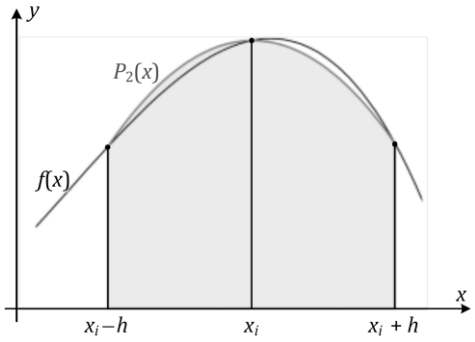


Fig. 5.3 : Règle de Simpson 1/3.

donc,

$$\int_{x_i}^{x_{i+2}} f(x)dx \approx (x_{i+2}-x_i)\sum_{j=0}^{2} H_j f(x_{i+j}) = (x_{i+2}-x_i)\left(\frac{1}{6}f(x_i) + \frac{2}{3}f(x_{i+1}) + \frac{1}{6}f(x_{i+2})\right)$$

or,

$$x_{i+2} = x_{i+1} + h = x_i + 2h$$

alors, la règle de Simpson 1/3 (Fig. 5.3) est :

$$\int_{x_i}^{x_{i+2}} f(x)dx \approx \frac{h}{3}\left(f(x_i) + 4f(x_{i+1}) + f(x_{i+2})\right)$$

Soit l'intervalle $[x_i - h,\ x_i + h]$, le reste d'intégration est exprimé par :

$$R(h) = \int_{x_i-h}^{x_i+h} f(x)dx - \frac{h}{3}\left[f(x_i-h) + 4f(x_i) + f(x_i+h)\right]$$

soit,

$$R'(h)=\left[f(x_i+h)+f(x_i-h)\right]-\frac{1}{3}\left[f(x_i-h)+4f(x_i)+f(x_i+h)\right]-\frac{h}{3}\left[-f'(x_i-h)+f'(x_i+h)\right]$$

$$=\frac{2}{3}\left[f(x_i-h)+f(x_i+h)\right]-\frac{4}{3}f(x_i)-\frac{h}{3}\left[-f'(x_i-h)+f'(x_i+h)\right]$$

$$R''(h)=\frac{2}{3}\left[-f'(x_i-h)+f'(x_i+h)\right]-\frac{1}{3}\left[-f'(x_i-h)+f'(x_i+h)\right]-\frac{h}{3}\left[f''(x_i-h)+f''(x_i+h)\right]$$

$$=\frac{1}{3}\left[-f'(x_i-h)+f'(x_i+h)\right]-\frac{h}{3}\left[f''(x_i-h)+f''(x_i+h)\right]$$

$$R'''(h)=\frac{1}{3}\left[f''(x_i-h)+f''(x_i+h)\right]-\frac{1}{3}\left[f''(x_i-h)+f''(x_i+h)\right]-\frac{h}{3}\left[-f'''(x_i-h)+f'''(x_i+h)\right]$$

$$=-\frac{h}{3}\left[f'''(x_i+h)-f'''(x_i-h)\right]=-\frac{2h^2}{3}\left(\frac{f'''(x_i+h)-f'''(x_i-h)}{2h}\right)$$

$$=-\frac{2h^2}{3}f^{(4)}(\xi),\quad x_i-h\le\xi\le x_i+h$$

En outre, on a :

$$R(0)=0,\ R'(0)=0\ \text{ et }\ R''(0)=0$$

Une intégration de proche en proche de $R'''(h)$ et l'application du théorème de la moyenne donnent :

$$R''(h)=R''(0)+\int_0^h R''(\tau)d\tau=-\frac{2}{3}f^{(4)}(\xi)\int_0^h\tau^2d\tau=-\frac{2}{9}h^3f^{(4)}(\xi),$$

$$R'(h)=R'(0)+\int_0^h R''(\tau)d\tau=-\frac{2}{9}f^{(4)}(\xi)\int_0^h\tau^3d\tau=-\frac{1}{18}h^4f^{(4)}(\xi),$$

$$R(h)=R(0)+\int_0^h R'(\tau)d\tau=-\frac{1}{18}f^{(4)}(\xi)\int_0^h\tau^4d\tau=-\frac{1}{90}h^5f^{(4)}(\xi)$$

Ainsi le reste d'intégration de la règle de Simpson 1/3 sur l'intervalle $\left[x_i-h,\ x_i+h\right]$vaut :

$$R(h)=-\frac{1}{90}h^5f^{(4)}(\xi)\ \text{ avec }\ \xi\in\left[x_i-h,\ x_i+h\right]$$

Par conséquent, l'erreur d'intégration locale est exprimée par :

$$E_i^{S1/3}=\frac{1}{90}h^5\left|f^{(4)}(\xi_i)\right|\ \text{ avec }\ \xi_i\in\left[x_i-h,\ x_i+h\right]$$

4.1.6. Formule de Simpson 1/3 composée et son reste

La règle de Simpson 1/3 nécessite trois nœuds équidistants. Soient n=2m un nombre pair et $f(x_i)$, $i=0,\ 1,\ 2,\ \ldots,\ n$, les valeurs de la fonction $f(x)$ pour les nœuds équidistants $a=x_0,\ x_1,\ \ldots,\ x_n=b$, dont le pas de discrétisation est :

$$h=(b-a)/n\equiv(b-a)/2m$$

L'application de la règle de Simpson 1/3 à chaque intervalle double $\left[x_0,\ x_2\right]$, ...,$\left[x_{2m-2},\ x_{2m}\right]$ d'une longueur de 2h, conduit à :

$$I(f)\approx\frac{h}{3}\left[f(x_0)+4f(x_1)+f(x_2)\right]+\frac{h}{3}\left[f(x_2)+4f(x_3)+f(x_4)\right]+\cdots+\frac{h}{3}\left[f(x_{2m-2})+4f(x_{2m-1})+f(x_{2m})\right]$$

ou,

$$I(f)\approx\frac{h}{3}\left[f(x_0)+f(x_n)+4\sum_{i=0}^{m-1}f(x_{2i+1})+2\sum_{i=1}^{2m-1}f(x_{2i})\right]\equiv I_{S1/3}(f,\ h)$$

Le reste d'intégration est composée de l'addition des restes d'intégration de chaque intervalle $[x_0,\ x_2]$, $[x_2,\ x_4]$, ...,$[x_{2m-2},\ x_{2m}]$ c'est-à-dire :

$$R^{S1/3}=-\frac{h^5}{90}\sum_{j=1}^{m}f^{(4)}(\xi_j)=-\frac{h^4}{180}\frac{(b-a)}{m}\sum_{j=1}^{m}f^{(4)}(\xi_j)$$

Comme $f^{(4)}(x)$ est continue sur l'intervalle [*a*, *b*], il existe ξ de $[a,\ b]$ tel que,

$$f^{(4)}(\xi)=\frac{1}{m}\sum_{j=1}^{m}f^{(4)}(\xi_j)$$

soit,

$$R^{S1/3}=-\frac{(b-a)h^4}{180}f^{(4)}(\xi)$$

par conséquent l'erreur d'intégration est :

$$E^{S1/3}=\frac{(b-a)h^4}{180}\left|f^{(4)}(\xi)\right|$$

qui est une erreur d'ordre 4.

Soit la fonction **`Simpson1by3`** qui permet l'estimation de l'intégrale d'une fonction **`fun`** définie et continue sur un l'intervalle [**`a, b`**].

```
function I = Simpson1by3(func,a,b,n,varargin)
% func = Nom de la fonction à intégrer
% a, b = Bornes d'intégration (a < b)
% n = Nombre de discrétisation (par défaut = 100)
% I = intégrale estimée par la méthode de Simpson 1/3
if nargin<4|isempty(n),n=100;end
if ~(b>a),error('Attention, il faut que a < b'),end
if ~(n>2),error('n est faible...'),end
Reste=mod(n,2);
if ~(Reste==0),error('Attention, il faut que n soit pair'),end
h=(b-a)/n;
  x=a:h:b;
S1=0; S2=0;
 for k=1:(n/2)
      S1=S1+func(x(2*k),varargin{:});
 end
 for k=1:((n/2)-1)
      S2=S2+func(x(2*k+1),varargin{:});
 end
I=(h/3)*(func(x(1),varargin{:})+4*S1+2*S2+func(x(n+1),varargin{:}));
end
```

Pour tester la fonction **`Simpson1by3`**, soit l'intégrale :

$$I(f)=\int_1^3 x^2\ln(x)dx$$

alors,

```
>> f=@(x)(x.^2).*log(x);
>> Simpson1by3(f,1,3,10)
ans =
    6.9986
```

4.2. Méthode de Simpson 3/8

4.2.7. Règle de Simpson 3/8 et son reste

Soit x_i, x_{i+1}, x_{i+2} et x_{i+3} quatre nœuds équidistants, la règle de Simpson 3/8 est obtenue à partir de la méthode de Newton-Cotes :

$$\int_{x_i}^{x_{i+3}} f(x)dx \approx (x_{i+3}-x_i)\sum_{j=0}^{3} H_j f(x_{i+j}) = (x_{i+3}-x_i)\left(H_0 f(x_i)+H_1 f(x_{i+1})+H_2 f(x_{i+2})+H_3 f(x_{i+3})\right)$$

sachant que,

$$H_j = \frac{1}{3}\frac{(-1)^{3-j}}{j!(3-j)!}\int_0^3 \frac{u(u-1)(u-2)(u-3)}{u-j}du$$

alors,

$$H_0 = -\frac{1}{18}\int_0^3 (u-1)(u-2)(u-3)du = \frac{1}{8},\quad H_1 = \frac{1}{6}\int_0^3 u(u-2)(u-3)du = \frac{3}{8}$$

$$H_2 = -\frac{1}{6}\int_0^3 u(u-1)(u-3)du = \frac{3}{8} \quad \text{et} \quad H_3 = \frac{1}{18}\int_0^3 u(u-1)(u-2)du = \frac{1}{8}$$

donc,

$$\int_{x_i}^{x_{i+3}} f(x)dx \approx (x_{i+3}-x_i)\sum_{j=0}^{3} H_j f(x_{i+j}) = \frac{(x_{i+3}-x_i)}{8}\left(f(x_i)+3f(x_{i+1})+3f(x_{i+2})+f(x_{i+3})\right)$$

or,

$$x_{i+3} = x_{i+2}+h = x_{i+1}+2h = x_i+3h$$

finalement,

$$\int_{x_i}^{x_{i+3}} f(x)dx \approx \frac{3}{8}h\left[f(x_i)+3f(x_{i+1})+3f(x_{i+2})+f(x_{i+3})\right]$$

Le reste d'intégration est déterminé par :

$$R_i = \frac{h^5}{4!}f^{(4)}(\xi_i)\int_0^3 u^{[4]}du = -\frac{h^5}{4\times 3\times 2}f^{(4)}(\xi_i)\frac{9}{10} = -\frac{3}{80}h^5 f^{(4)}(\xi_i)$$

Par conséquent, l'erreur d'intégration locale est exprimée par :

$$E_i^{S3/8} = \frac{3}{80}h_i^5\left|f^{(4)}(\xi_i)\right| \quad \text{avec} \quad \xi_i \in [x_i,\ x_i+3h]$$

4.2.8. Formule de Simpson 3/8 composée et son reste

La règle de Simpson 3/8 nécessite quatre nœuds équidistants. Soient $n=3m$ et $f(x_i)$ $(i=0,\ 1,\ 2,\ \ldots,\ n)$ les valeurs de la fonction $f(x)$ pour les nœuds équidistants $a=x_0,\ x_1,\ \ldots,\ x_n=b$, dont le pas de discrétisation est :

$$h=(b-a)/n \equiv (b-a)/3m$$

L'application de la règle de Simpson 3/8 à chaque intervalle $[x_0,\ x_3]$, $[x_3,\ x_6]$, ..., $[x_{3m-3},\ x_{3m}]$ d'une longueur de 3*h*, conduit à :

$$I(f)\approx\frac{3h}{8}\left[f(x_0)+3f(x_1)+3f(x_2)+f(x_3)\right]+\frac{3h}{8}\left[f(x_3)+3f(x_4)+3f(x_5)+f(x_6)\right]+\cdots$$
$$+\frac{3h}{8}\left[f(x_{3m-3})+3f(x_{3m-2})+3f(x_{3m-1})+f(x_{3m})\right]$$

ou,

$$I(f)\approx\frac{3h}{8}\left[f(x_0)+f(x_n)+3\sum_{i=1}^{m}\left[f(x_{3i-2})+f(x_{3i-1})\right]+2\sum_{i=1}^{m-1}f(x_{3i})\right]\equiv I_{S3/8}(f,\ h)$$

Le reste d'intégration est composée de l'addition des restes d'intégration de chaque intervalle $[x_0,\ x_2]$, $[x_2,\ x_4]$, ...,$[x_{2m-2},\ x_{2m}]$, c'est-à-dire :

$$R^{S3/8}=-\frac{3}{80}h^5\sum_{j=1}^{m}f^{(4)}(\xi_j)=-\frac{h^4}{80}\frac{(b-a)}{m}\sum_{j=1}^{m}f^{(4)}(\xi_j)$$

comme $f^{(4)}(x)$ est continue, il existe ξ de [*a*, *b*] tel que

$$f^{(4)}(\xi)=\frac{1}{m}\sum_{j=1}^{m}f^{(4)}(\xi_j)$$

soit,

$$R^{S3/8}=-\frac{(b-a)h^4}{80}f^{(4)}(\xi)$$

sar conséquent l'erreur d'intégration est :

$$E^{S3/8}=\frac{(b-a)h^4}{80}\left|f^{(4)}(\xi)\right|$$

qui est une erreur d'ordre 4.

Soit la fonction **`Simpson3by8`** qui permet l'estimation de l'intégrale d'une fonction **`fun`** définie et continue sur un l'intervalle [**`a, b`**].

```
function I = Simpson3by8(func,a,b,n,varargin)
% func = Nom de la fonction à intégrer
% a, b = Bornes d'intégration (a < b)
% n = Nombre de discrétisation (par défaut = 99)
% I = intégrale estimée par la méthode de Simpson 3/8
if nargin<4|isempty(n),n=99;end
Reste=mod(n,3);
if ~(n>3),error('n est faible...'),end
if ~(Reste==0),error('Il faut que n  est un multiple de 3'),end
h=(b-a)/n; m=n/3;
x=a:h:b; S1=0; S2=0;
for k=1:m
    S1=S1+func(x(3*k-1),varargin{:})+func(x(3*k),varargin{:});
end
for k=1:(m-1)
    S2=S2+func(x(3*k+1),varargin{:});
end
I=func(x(1),varargin{:})+3*S1+2*S2+func(x(n+1),varargin{:});
I=(3*h/8)*I;
end
```

Pour tester la fonction **`Simpson1by3`**, soit l'intégrale :

$$I(f)=\int_1^3 x^2\ln(x)dx$$

alors,

```
>> f=@(x) (x.^2).*log(x);
>> Simpson3by8(f,1,3,9)
ans =
    6.9986
```

Les deux méthodes des Trapèzes et de Simpson sont très utilisées dans le calcul approché des intégrales définies des fonctions continues. Le développement dans l'analyse mathématique numérique à permet d'avoir d'autres méthodes de calcul approché rapides et efficaces à celles des méthodes classiques des trapèzes et de Simpson celles-ci qui sont basées sur la formule de quadrature de Newton-Cotes qui n'est autre qu'une approximation de l'intégrale de la fonction par l'intégrale de son polynôme d'interpolation sur l'intervalle $[a,\ b]$ subdivisé en nœuds équidistants. Mais, pour des questions d'instabilité numérique provenant en particulier du phénomène de Runge, il est cependant préférable de limiter le degré du polynôme d'interpolation, quitte à subdiviser l'intervalle en sous-intervalles.

5. Formules de quadrature de Gauss

Soit $f(t)$ une fonction continue dans l'intervalle $[-1,\ 1]$ et $w(t)=1$, la formule de quadrature :

$$\int_{-1}^{1} f(t)dt=\sum_{i=1}^{n}\lambda_i f(t_i) \qquad (5.1)$$

Cette formule contient les inconnus $\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_n,\ t_1,\ t_2,\ \ldots$ et t_n, c'est-à-dire $2n$ inconnus. Pour que la formule (5.1) soit exacte, la fonction $f(t)$ est approchée par un polynôme qui possède $2n$ coefficients, c'est-à-dire ce polynôme est de degré $N=2n-1$:

$$f(t)\approx\sum_{k=0}^{N}C_k t^k$$

Pour garantir la relation (5.1) soit toujours vérifiée, il faut que,

$$\int_{-1}^{1} t^k dt=\sum_{i=1}^{n}\lambda_i t_i^k \qquad (5.2)$$

En effet,

$$\int_{-1}^{1} f(t)dt=\int_{-1}^{1}\sum_{k=0}^{N}C_k t^k dt=\sum_{k=0}^{N}C_k\int_{-1}^{1} t^k dt=\sum_{k=0}^{N}C_k\sum_{i=1}^{n}\lambda_i t_i^k=\sum_{i=1}^{n}\lambda_i\sum_{k=0}^{N}C_k t_i^k\equiv\sum_{i=1}^{n}\lambda_i f(t_i)$$

à partir de l'équation (5.2),

$$\int_{-1}^{1} t^k dt=\frac{1-(-1)^{k+1}}{k+1}=\sum_{i=1}^{n}\lambda_i t_i^k,\ \ k=0,\ 1,\ 2,\ \ldots,\ 2n-2,\ 2n-1$$

ou bien,

$$\begin{cases} \lambda_1+\lambda_2+\cdots+\lambda_n=2, \\ \lambda_1 t_1+\lambda_2 t_2+\cdots+\lambda_n t_n=0, \\ \lambda_1 t_1^2+\lambda_2 t_2^2+\cdots+\lambda_n t_n^2=\dfrac{2}{3}, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \lambda_1 t_1^{2n-2}+\lambda_2 t_2^{2n-2}+\cdots+\lambda_n t_n^{2n-2}=\dfrac{2}{2n-1}, \\ \lambda_1 t_1^{2n-1}+\lambda_2 t_2^{2n-1}+\cdots+\lambda_n t_n^{2n-1}=0. \end{cases}$$

Le système obtenu est non linéaire de 2*n* équations et 2*n* inconnus, sa résolution par la voie usuelle ou numérique (surtout si *n* est grand) est fastidieuse.

Considérons le cas où $f(t)$ est approché par :

$$f(t)\approx\sum_{k=0}^{N}C_k t^k\equiv t^{n-1}P_n(t)$$

$P_n(t)$, est le polynôme de Legendre de degré *n*. D'après la formule (5.1),

$$\int_{-1}^{1}t^{n-1}P_n(t)dt=\sum_{i=1}^{n}\lambda_i t_i^{n-1}P_n(t_i)$$

Vu la propriété d'orthogonalité des polynômes de Legendre, rend vraies l'égalité,

$$\int_{-1}^{1}t^{n-1}P_n(t)dt=0\ \Rightarrow\ \sum_{i=1}^{n}\lambda_i t_i^{n-1}P_n(t_i)=0\ \Rightarrow\ P_n(t_i)=0,\ \forall\ \lambda_1,\ \lambda_2,\ \ldots,\ \lambda_n$$

L'équation $P_n(t_i)=0$, possède t_1, t_2, …, t_n solutions réels distincts et compris dans l'intervalle [-1, 1]. Par la suite et à partir de *n* premières équations du système précédent, soit :

$$\begin{cases} \lambda_1+\lambda_2+\cdots+\lambda_n=2, \\ \lambda_1 t_1+\lambda_2 t_2+\cdots+\lambda_n t_n=0, \\ \lambda_1 t_1^2+\lambda_2 t_2^2+\cdots+\lambda_n t_n^2=\dfrac{2}{3}, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \lambda_1 t_1^{n-1}+\lambda_2 t_2^{n-1}+\cdots+\lambda_n t_n^{n-1}=\dfrac{1-(-1)^n}{n}. \end{cases}$$

Ce sous-système à un déterminant de Vandermonde et par suite la détermination des λ_1, λ_2, …, λ_n est univoque ou peuvent être déterminés aussi par (Abramowitz et Stegun, 1972) :

$$\lambda_i=\frac{-2}{(n+1)P_n'(t_i)P(t_i)_n}=\frac{2}{(1-t_i^2)\left[P_n'(t_i)\right]^2}$$

La formule (5.1) où les t_1, t_2, …, t_n sont les zéros du polynôme de Legendre $P_n(t)$ et où λ_1, λ_2, …, λ_n obtenues à partir du système de Vandermonde s'appelle *formule de quadrature de Gauss–Legendre*.

Examinons la formule de quadrature de Gauss–Legendre pour le cas $n=3$. D'après la formule de Rodrigues (Abramowitz et Stegun, 1972, Zwillinger, 2003), le polynôme de Legendre est exprimé par :

$$P_n(t)=\frac{1}{2^n n!}\frac{d^n}{dt^n}\left[\left(t^2-1\right)^n\right]$$

pour *n* = 3,

$$P_3(t)=\frac{1}{2}\left(5t^3-3t\right)$$

alors,

$$P_3(t)=0 \;\Rightarrow t_1=-\sqrt{\frac{3}{5}},\; t_2=0 \;\text{ et }\; t_3=\sqrt{\frac{3}{5}}$$

Pour déterminer les coefficients λ_1, λ_2 et λ_3, soit le système d'équations linéaires :

$$\begin{cases}\lambda_1+\lambda_2+\lambda_3=2,\\ -\sqrt{\frac{3}{5}}\lambda_1+\sqrt{\frac{3}{5}}\lambda_3=0,\\ \frac{3}{5}\lambda_1+\frac{3}{5}\lambda_3=\frac{2}{3}.\end{cases}\Rightarrow\begin{cases}\lambda_1=\frac{5}{9},\\ \lambda_2=\frac{8}{9},\\ \lambda_3=\frac{5}{9}.\end{cases}$$

par conséquent,

$$\int_{-1}^{1} f(t)dt=\frac{1}{9}\left[5f\left(-\sqrt{\frac{3}{5}}\right)+8f(0)+5f\left(\sqrt{\frac{3}{5}}\right)\right]$$

C'est la *formule de quadrature de Gauss à trois points.*

Examinons l'utilisation de la formule de quadrature de pour calculer l'intégrale,

$$I(f)=\int_a^b f(x)dx$$

en changeant la variable

$$x=\frac{a+b}{2}+\frac{b-a}{2}t \;\Rightarrow\; dx=\frac{b-a}{2}dt$$

alors,

$$I(f)=\frac{b-a}{2}\int_{-1}^{1} f\left(\frac{a+b}{2}+\frac{b-a}{2}t\right)dt$$

Appliquant à la dernière intégrale la formule de Gauss (5.1), on aura :

$$I(f)=\frac{b-a}{2}\sum_{i=1}^{n}\lambda_i f\left(\frac{a+b}{2}+\frac{b-a}{2}t_i\right)$$

Le reste de la formule de Gauss à *n* points est donnée par l'expression (Abramowitz et Stegun, 1972) :

$$R_n=\frac{(b-a)^{2n+1}(n!)^4}{[(2n)!]^3(2n+1)}f^{(2n)}(\xi)\quad\text{avec}\;\; a\le\xi\le b$$

d'où l'on tire

$$R_2=\frac{1}{135}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi),\quad R_3=\frac{1}{15750}\left(\frac{b-a}{2}\right)^7 f^{(6)}(\xi),\quad R_4=\frac{1}{3472875}\left(\frac{b-a}{2}\right)^9 f^{(8)}(\xi)$$

$$R_5=\frac{1}{1237732650}\left(\frac{b-a}{2}\right)^{11} f^{(11)}(\xi),\quad R_6=\frac{1}{6489844861\,50}\left(\frac{b-a}{2}\right)^{13} f^{(12)}(\xi)$$

Soit la fonction **GaussLegendreQuad** qui permet l'estimation de l'intégrale d'une fonction **fun** définie et continue sur un l'intervalle [**a, b**] par la formule de quadrature de Gauss. Cette fonction possède comme argument le degré **n** du polynôme de Legendre.

```
function I = GaussLegendreQuad(fun,a,b,n,varargin)
% func = Nom de la fonction à intégrer
% legendreP polynôme de Legendre introduit à partir de MATLAB 2014b
% a, b : bornes d'intégration (a < b)
% n : degré du polynôme  de Legendre (par défaut = 8)
% I : intégrale estimée par la formule de Gauss-Legendre
% t : racines du polynôme de Legendre de degré n
if ~(b>a),error('Attention, il faut que a soit inférieur à b'),end
if nargin<4|isempty(n),n=8;end
syms X;
t = double(vpasolve(legendreP(n,X) == 0));
LaMatrice=vander(t)';
for k=1:n
    LeVecteurB(k)=(1-((-1)^k))/k;
end
j=1;
for i=n:-1:1
   B(j)=LeVecteurB(i);
   j=j+1;
end
   Lamda = inv(LaMatrice)*B';
I=0;
for i=1:n
    I=I+Lamda(i)*fun(0.5*(a+b+(b-a)*t(i)),varargin{:});
end
I=0.5*(b-a)*I;
end
```

Pour tester la fonction **GaussLegendreQuad**, soit l'intégrale :

$$I(f)=\int_1^3 x^2\ln(x)dx$$

alors,

```
>> f=@(x) (x.^2).*log(x);
>> GaussLegendreQuad(f,1,3,5)
ans =
    6.9986
```

Autres formules analogues à la formule de quadrature de Gauss existent dans la littérature mathématique (Abramowitz et Stegun, 1972). Elles sont basées sur la propriété d'orthogonalité des polynômes, parmi ces formules soit :

- Formule de quadrature de Gauss – Tchebychev de premier type,

$$\int_{-1}^{1} w(t)f(t)dt \approx \sum_{j=1}^{n} \lambda_j f(t_j) \text{ avec, } w(t)=\frac{1}{\sqrt{1-t^2}},\ t_j=\cos\left(\frac{2j-1}{2n}\pi\right)$$

$$\lambda_j=\frac{\pi}{n},\quad R_n=\frac{\pi}{(2n)!\,2^{2n-1}}f^{(2n)}(\xi),\ -1<\xi<1$$

- Formule de quadrature de Gauss–Tchebychev de second type,

$$\int_{-1}^{1} w(t)f(t)dt \approx \sum_{j=1}^{n} \lambda_j f(t_j) \text{ avec, } w(t)=\sqrt{1-t^2},\ t_j=\cos\left(\frac{j}{n+1}\pi\right)$$

$$\lambda_j=\frac{\pi}{n+1}\sin^2\left(\frac{j}{n+1}\pi\right),\quad R_n=\frac{\pi}{(2n)!\,2^{2n-1}}f^{(2n)}(\xi),\ -1<\xi<1$$

- Formule de quadrature de Gauss – Laguerre,

$\int_{0}^{+\infty} w(t)f(t)dt \approx \sum_{j=1}^{n} \lambda_j f(t_j)$ avec, $w(t)=e^{-t}$, t_j est la *j*-ième racine du polynôme de Laguerre :

$$L_n(t)=\frac{e^t}{n!}\frac{d^n}{dt^n}\left(e^{-t}t^n\right)$$

$$\lambda_j=\frac{(n!)^2 t_j}{(n+1)^2\left[L_{n+1}(t_j)\right]^2},\quad R_n=\frac{(n!)^2}{(2n)!}f^{(2n)}(\xi),\ 0<\xi<\infty$$

- Formule de quadrature de Gauss – Hermite,

$\int_{-\infty}^{+\infty} w(t)f(t)dt \approx \sum_{j=1}^{n} \lambda_j f(t_j)$ avec, $w(t)=e^{-t^2}$, t_j est la *j*-ième racine du polynôme de l'Hermite :

$$H_n(t)=(-1)^n e^{t^2/2}\frac{d^n}{dt^n}\left(e^{-t^2/2}\right)$$

$$\lambda_j=\frac{2^{n-1}n!\sqrt{\pi}}{n^2\left[H_{n-1}(t_j)\right]^2},\quad R_n=\frac{n!\sqrt{\pi}}{2^n(2n)!}f^{(2n)}(\xi),\ -\infty<\xi<\infty$$

Les formules de quadrature ont la structure générale,

$$\int_{a}^{b} f(x)dx=\sum_{j=1}^{n}\lambda_j f(x_j)+R$$

x_1, x_2, …, x_n étant le système des points donné appartenant au segment d'intégration [*a*, *b*]. λ_j certaines constantes connues et *R* le reste. Pour le même nombre d'ordonnées, la précision de différentes formules de quadrature n'est pas la même. Considérons l'exemple :

$$I=\int_{-1}^{1}\sqrt{2+x}dx=\int_{1}^{3}\sqrt{u}du=\frac{2}{3}\left[u\sqrt{u}\right]_1^3=2\sqrt{3}-\frac{2}{3}=2.797435\ldots$$

Comparons la précision des différentes formules, sachant que l'intervalle [-1, 1] est divisé en trois segments égales, c'est-à-dire $x_0=-1$, $x_1=-1/3$, $x_2=1/3$, $x_3=1$.

D'après la formule de Simpson 1/3,

$$I\approx\frac{1}{3}\left[\sqrt{2-1}+4\sqrt{2+0}+\sqrt{2+1}\right]=\frac{1}{3}\times 8.428905=2.809635$$

et d'après la formule de Gauss,

$$I \approx 0.555566 \cdot \left(\sqrt{2-0.774597} + \sqrt{2+0.774597}\right) + 0.888889 \cdot \sqrt{2+0} = 2.797460$$

La formule de Gauss donne un résultat très proche de la valeur réel.

La précision des formules de quadrature est définie surtout par l'ordre du reste,

$$R = O\left(h^m\right)$$

m, est un nombre naturel, par exemple, pour la formule des trapèzes *m* = 2 et celui de la formule de Simpson 1/3, *m* = 4. La précision de la formule de quadrature est considérée d'autant plus élevée que le nombre *m* est plus grand ; dans ce sens la formule de Simpson est plus exacte que celle des trapèzes. La qualité de la formule se manifeste déjà avec un pas de discrétisation *h* suffisamment petit. Mais, il ne s'ensuit nullement que dans des cas concrets une formule plus grossière ne peut donner pour le même pas de discrétisation de meilleurs résultats qu'une formule exacte. Par exemple, pour la fonction (Fig. 5.4) :

$$f(x) = -25x^4 + 45x^2 - 8$$

on a,

$$I = \int_{-1}^{1} f(x)dx = 4$$

pour *h* = 1, la formule des trapèzes donne la valeur exacte

$$I_T = \frac{1}{2}f(-1) + f(0) + \frac{1}{2}f(1) = 6 - 8 + 8 = 4$$

Alors que la formule de Simpson 1/3 pour *h* = 1 n'assure même pas le signe de l'intégrale :

$$I_{S1/3} = \frac{1}{3}\left[f(-1) + 4f(0) + f(1)\right] = \frac{1}{3}\left[12 - 32 + 12\right] = -\frac{8}{3}$$

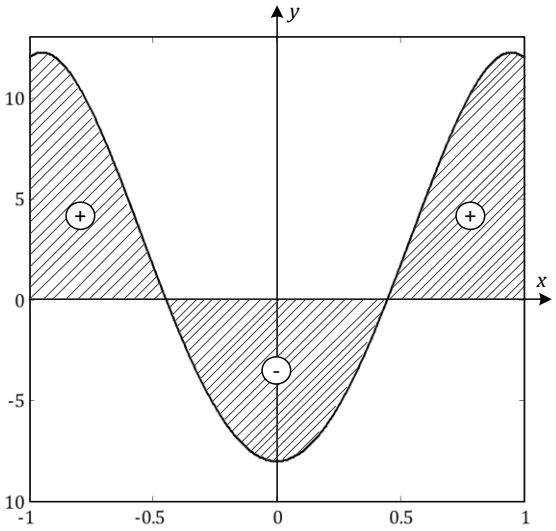


Fig. 5.4 : Précision de la méthode de quadrature.

La précision d'une formule de quadrature pour un nombre de points fixe dépend sensiblement de la répartition de ces points. Si cette répartition est mauvaise, les résultats fournis par une formule de quadrature peuvent être très compromis. Il n'est pas difficile non plus de construire des exemples analogues pour une formule de quadrature quelconque à nombre d'ordonnées arbitraire.

En général, dans le cas d'un grand nombre de zéros de la fonction $f(x)$ sous le signe somme ou d'un grand nombre de ses extrémums, c'est-à-dire d'un grand nombre des zéros de la dérivée $f'(x)$. La précision des formules de quadrature diminue nettement par la suite de grandes valeurs inévitables de sa dérivée $f'(x)$. A cette fin on recommande de partitionner le segment d'intégration principale [*a*, *b*] en segments partiels [α, β] à l'intérieur desquels les fonctions $f(x)$ et $f'(x)$ restent du même signe (si c'est possible), et de calculer l'intégrale par parties en choisissant en général pour chaque segment partiel son propre pas. Dans des cas plus complexes il faut tenir compte également du comportement des dérivées d'ordre

supérieur $f^{(n)}(x)$, $(n \geq 2)$. Pour une orientation générale, il convient de construire au préalable la courbe de la fonction sous le signe somme. Si cette fonction oscille fortement, il convient d'appliquer des procédés de calcul spéciaux. La précision des formules de quadrature peut être également améliorée par des procédés généraux établis à cet effet.

Lorsqu'on cherche la borne d'erreur totale d'une formule de quadrature, on doit également tenir compte de l'erreur de sommation R_1. Supposons que les termes de la somme $f(x_i)$, $(i = 1,\ 2,\ \ldots,\ n)$ sont calculés avec une erreur absolue égale ou inférieure à ε, et les coefficients λ_i de la formule de quadrature sont des constantes positives exactes. On peut alors poser,

$$R_1 \leq \sum_{j=1}^{n} \lambda_j \varepsilon = \varepsilon \sum_{j=1}^{n} \lambda_j$$

La formule de quadrature étant vérifiée pour $f(x) = 1$, il vient

$$\int_a^b dx = b - a = \sum_{j=1}^{n} \lambda_j$$

ce qui entraîne,

$$R_1 \leq (b - a)\varepsilon$$

Par conséquent, si l'on ne tient pas compte de l'erreur d'arrondi du résultat, la borne d'erreur totale de la formule de quadrature est :

$$\overline{R} = (b - a)\varepsilon + |R|$$

où $|R|$ est une erreur de la méthode qui peut être définie par le procédé indiqué dans ce qui précède.

Constatons que si la fonction $f(x)$ sous le signe somme est donnée par le tableau des valeurs $f(x_i)$, $(i = 1,\ 2,\ \ldots,\ n)$, alors en toute rigueur nous sommes dans l'impossibilité d'évaluer la précision de la formule de quadrature. Il en est ainsi parce que par un système fini de points, on peut mener un nombre illimité de courbes (Fig. 5.5) délimitant sur le segment donné [*a*, *b*] des surfaces différentes, c'est-à-dire l'intégrale :

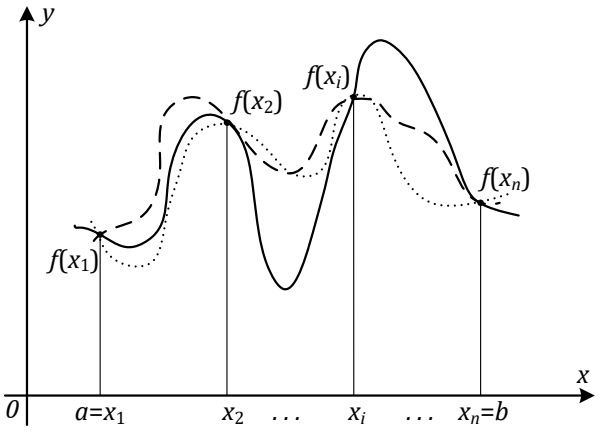


Fig. 5.5 : Possibilité illimité de courbes.

$$I(f) = \int_a^b f(x)dx$$

peut avoir a priori une valeur parfaitement arbitraire. L'application des formules de quadrature n'est alors admissible que le cas où l'on connaît dans une certaine mesure des valeurs intermédiaires non utilisées de la fonction sous le signe somme et ses propriétés générales qui permettent de juger sur l'allure de sa courbe (Démidovitch et Maron, 1979).

6. Formule d'intégration d'Euler-Maclaurin

La formule d'Euler-Maclaurin est une relation entre sommes discrètes et intégrales. Elle fut découverte indépendamment, aux alentours de 1735, par le mathématicien suisse Leonhard Euler pour accélérer le calcul de limites de séries lentement convergentes et par l'Écossais Colin Maclaurin pour calculer des valeurs approchées d'intégrales.

Soit $f(x)$ une fonction infiniment dérivable sur $[x_0, +\infty[$ et l'opérateur de la différence Δ tel que :

$$\Delta f(x) = f(x+h) - f(x)$$

h, une valeur positive fixe. L'opérateur inverse $1/\Delta$ de la fonction $f(x)$ la fonction $F(x)$ qui vérifie l'équation aux différences finies :

$$\Delta F(x) = f(x) \;\Rightarrow\; F(x) = \frac{1}{\Delta} f(x)$$

Si la fonction $f(x)$ est considérée sur un ensemble des points x_0, x_1, x_2, … où $\Delta x_i = x_{i+1} - x_i = h$, $\forall\, i = 0,\ 1,\ 2,\ \ldots$ l'opérateur inverse $F(x_i) = f(x_i)/\Delta$ se construit facilement. Soit la somme finie :

$$S(x_i) = \sum_{\eta=0}^{i-1} f(x_\eta) \text{ pour } i = 1,\ 2,\ \ldots$$

admettant par convention $S(x_0) = 0$, alors :

$$\Delta S(x_i) = S(x_{i+1}) - S(x_i) = f(x_i)$$

or,

$$\Delta F(x_i) = f(x_i)$$

alors,

$$\Delta F(x_i) - \Delta S(x_i) = 0 \;\Rightarrow\; \Delta[F(x_i) - S(x_i)] = 0 \;\Rightarrow\; F(x_i) - S(x_i) = F(x_0) - S(x_0) = F(x_0)$$

d'où,

$$F(x_i) = F(x_0) + S(x_i) \;\Leftrightarrow\; \frac{1}{\Delta} f(x_i) = F(x_0) + S(x_i)$$

c'est-à-dire l'opérateur inverse d'une différence finie est opérateur de sommation finie. $F(x_0)$ étant une grandeur constante arbitraire.

Soit l'opérateur de dérivation :

$$D \equiv \frac{d}{dx}$$

c'est-à-dire,

$$Df(x) = \frac{df(x)}{dx} \;\Rightarrow\; f(x)dx = \frac{1}{D} df(x) \;\Rightarrow\; \frac{1}{D} f(x) = \int_{x_0}^{x} f(x)dx$$

$1/D$ est l'opérateur inverse qui correspond à l'opérateur d'intégration.

Le développement en série de Taylor de la fonction $f(x)$ au voisinage de $x+h$ est :

$$f(x+h) = \sum_{k\geq 0} \frac{h^k}{k!} \frac{d^k f(x)}{dx^k} = f(x) + \sum_{k\geq 1} \frac{h^k}{k!} \frac{d^k f(x)}{dx^k}$$

où,

$$\Delta f(x)=f(x+h)-f(x)=\sum_{k\geq 1}\frac{h^k}{k!}\frac{d^k f(x)}{dx^k} \;\Rightarrow\; \Delta f(x)=\left(\sum_{k\geq 1}\frac{h^k}{k!}D^k\right)f(x)$$

qui s'écrit aussi,

$$\Delta f(x)=\left(\sum_{k\geq 0}\frac{(hD)^k}{k!}-1\right)f(x) \;\Leftrightarrow\; \Delta f(x)=\left(e^{hD}-1\right)f(x)$$

par conséquent,

$$\Delta=e^{hD}-1 \;\Rightarrow\; \frac{1}{\Delta}=\frac{1}{e^{hD}-1} \;\Rightarrow\; hD\frac{1}{\Delta}=\frac{hD}{e^{hD}-1}\equiv\varphi(hD)$$

La fonction $\varphi(hD)$ dite *la fonction génératrice des nombres de Bernoulli* (Weisstein, 2003).

Les nombres de Bernoulli B_n, $n=0,\ 1,\ 2,\ 3,\ \ldots$ ont été introduits par Jacques (James, Jacob) Bernoulli. Plusieurs définitions équivalentes ont été proposées par des mathématiciens afin de définir ces nombres parmi lesquelles le développement en série entière de la fonction $\varphi(x)$:

$$\varphi(x)=\frac{x}{e^x-1}=\sum_{n=0}^{\infty}\frac{B_n}{n!}x^n$$

sachant que,

$$e^x=\sum_{n=0}^{\infty}\frac{x^n}{n!}=1+\sum_{n=1}^{\infty}\frac{x^n}{n!} \;\Rightarrow\; \varphi(x)=\frac{1}{1+\sum_{n=1}^{\infty}\frac{x^n}{(n+1)!}}=B_0+\sum_{n=1}^{\infty}\frac{B_n}{n!}x^n$$

Ce qui donne $\varphi(0)=1=B_0$

D'après le développement en série de Maclaurin de la fonction $\varphi(x)$, B_n est peut-être exprimé par :

$$B_n=\left.\frac{d^n\varphi(x)}{dx^n}\right|_{x=0}\equiv\left.\frac{d^n}{dx^n}\left(\frac{x}{e^x-1}\right)\right|_{x=0}$$

par exemple,

$$B_1=\left.\frac{d}{dx}\left(\frac{x}{e^x-1}\right)\right|_{x=0}=\left.\frac{1}{e^x-1}\right|_{x=0}-\left.\frac{xe^x}{\left(e^x-1\right)^2}\right|_{x=0}=-\frac{1}{2}$$

si l'on tient compte les valeurs de $B_0=1$ et $B_1=1/2$, on a :

$$\phi(x)=\varphi(x)-B_1x=\frac{x}{e^x-1}+\frac{x}{2}=1+\sum_{n=2}^{\infty}\frac{B_n}{n!}x^n$$

il est évident que,

$$\phi(x)=\frac{x\left(e^x+1\right)}{2\left(e^x-1\right)}=\frac{x}{2}\frac{e^{\frac{x}{2}}+e^{-\frac{x}{2}}}{e^{\frac{x}{2}}-e^{-\frac{x}{2}}}=\frac{x}{2}\coth\frac{x}{2}$$

or,

$$\phi(-x)=\frac{-x}{2}\coth\frac{-x}{2}=-\frac{x}{2}\frac{e^{-\frac{x}{2}}+e^{\frac{x}{2}}}{e^{-\frac{x}{2}}-e^{\frac{x}{2}}}=\frac{x}{2}\frac{e^{\frac{x}{2}}+e^{-\frac{x}{2}}}{e^{\frac{x}{2}}-e^{-\frac{x}{2}}}=\phi(x)$$

$\phi(x)$ est une fonction paire, son développement ne contient que les puissances paires de x c'est-à-dire,

$$B_{2k+1}=0,\ \forall\ k=1,\ 2,\ 3,\ \ldots$$

Pour déterminer les autres nombres B_{2n} pour $n=1,\ 2,\ 3,\ \ldots$ utilisons l'identité (Arfken et Weber, 2005) :

$$1=\left(\sum_{m=0}^{\infty}\frac{x^m}{(m+1)!}\right)\left(1-\frac{x}{2}+\sum_{n=1}^{\infty}\frac{B_{2n}}{(2n)!}x^{2n}\right)=1+\sum_{m=1}^{\infty}x^m\left(\frac{1}{(m+1)!}-\frac{1}{2m!}\right)+\sum_{\eta=2}^{\infty}x^{\eta}\sum_{1\le n\le \eta/2}\frac{B_{2n}}{(2n)!\,(\eta-2n+1)!}$$

c'est-à-dire,

$$\sum_{m=1}^{\infty}x^m\left(\frac{1}{(m+1)!}-\frac{1}{2m!}\right)+\sum_{\eta=2}^{\infty}x^{\eta}\sum_{1\le n\le \eta/2}\frac{B_{2n}}{(2n)!\,(\eta-2n+1)!}=0$$

Pour $\eta>0$ le coefficient de x^{η} est nul, alors :

$$\frac{1}{2}(\eta+1)-1=\sum_{1\le n\le \eta/2}B_{2n}C_{\eta+1}^{2n}=\frac{1}{2}(\eta-1)$$

qui est équivalent à :

$$\eta-\frac{1}{2}=\sum_{n=1}^{\eta}B_{2n}C_{2\eta+1}^{2n}\ \text{ ou }\ \eta-1=\sum_{n=1}^{\eta-1}B_{2n}C_{2\eta}^{2n}$$

L'utilisation de l'une des équations permet de donner par exemple les nombres suivants :

$$B_2=\frac{1}{6}\ ;\ B_4=-\frac{1}{30}\ ;\ B_6=\frac{1}{42}\ ;\ B_8=-\frac{1}{30}\ ;\ B_{10}=\frac{5}{66}\ ;\ B_{12}=-\frac{691}{2730}$$

$$B_{14}=\frac{7}{6}\ ;\ B_{16}=-\frac{3617}{510}\ ;\ B_{18}=\frac{43867}{798}\ ;\ B_{20}=-\frac{174611}{330}$$

Les nombres de Bernoulli trouvent une application dans de nombreux domaines, en particulier ils sont utilisés dans la formule importante d'Euler-Maclaurin.

Soit alors,

$$\frac{d}{dx}\left[\frac{1}{\Delta}f(x)\right]=\sum_{k=0}^{\infty}\frac{B_k}{k!}h^{k-1}D^k f(x)$$

L'intégration de cette expression entre les limites de x_0 et x_n conduit à :

$$\frac{1}{\Delta}f(x_n)-\frac{1}{\Delta}f(x_0)=\frac{1}{h}\int_{x_0}^{x_n}f(x)dx+\sum_{k=1}^{\infty}\frac{B_k}{k!}h^{k-1}\left[f^{(k-1)}(x_n)-f^{(k-1)}(x_0)\right]$$

or,

$$\frac{1}{\Delta}f(x_n)=F(x_0)+S(x_n)\ \text{ et }\ \frac{1}{\Delta}f(x_0)=F(x_0)+S(x_0)=F(x_0)$$

soit,

$$\frac{1}{\Delta}f(x_n)-\frac{1}{\Delta}f(x_0)=S(x_i)\equiv\sum_{\eta=0}^{n-1}f(x_\eta)$$

donc,

$$\sum_{\eta=0}^{n-1}f(x_\eta)=\frac{1}{h}\int_{x_0}^{x_n}f(x)dx+\sum_{k=1}^{\infty}\frac{B_k}{k!}h^{k-1}\left[f^{(k-1)}(x_n)-f^{(k-1)}(x_0)\right]$$

vu que,

$$B_1=-\frac{1}{2}\ \text{ et }\ B_{2k+1}=0,\ \forall\ k=1,\ 2,\ 3,\ \ldots$$

soit,

$$\sum_{\eta=0}^{n-1} f(x_\eta) = \frac{1}{h}\int_{x_0}^{x_n} f(x)dx - \frac{1}{2}[f(x_n) - f(x_0)] + \sum_{k=1}^{\infty} \frac{B_{2k}}{(2k)!} h^{2k-1}\left[f^{(2k-1)}(x_n) - f^{(2k-1)}(x_0)\right]$$

finalement,

$$\int_{x_0}^{x_n} f(x)dx = h\left[\frac{1}{2}f(x_0) + \sum_{\eta=1}^{n-1} f(x_\eta) + \frac{1}{2}f(x_n)\right] - \sum_{k=1}^{\infty} \frac{B_{2k}}{(2k)!} h^{2k}\left[f^{(2k-1)}(x_n) - f^{(2k-1)}(x_0)\right]$$

ou bien,

$$\int_{x_0}^{x_n} f(x)dx = h\left[\frac{1}{2}f(x_0) + \sum_{\eta=1}^{n-1} f(x_\eta) + \frac{1}{2}f(x_n)\right] - \sum_{k=1}^{m} \frac{B_{2k}}{(2k)!} h^{2k}\left[f^{(2k-1)}(x_n) - f^{(2k-1)}(x_0)\right] + R_{2m}$$

C'est la formule d'Euler-Maclaurin, R_{2m} est un reste de la formule qui peut être exprimé par :

$$R_{2m} = -\frac{B_{2m+2}}{(2m+2)!} h^{2m+2}\left[f^{(2m+1)}(x_n \pm \theta) - f^{(2m+1)}(x_0 \pm \theta)\right]$$

sachant que,

$$f^{(2m+1)}(x_n \pm \theta) - f^{(2m+1)}(x_0 \pm \theta) = (x_n - x_0)\frac{f^{(2m+1)}(x_n \pm \theta) - f^{(2m+1)}(x_0 \pm \theta)}{x_n - x_0}$$

or,

$$x_n - x_0 = nh$$

alors,

$$f^{(2m+1)}(x_n \pm \theta) - f^{(2m+1)}(x_0 \pm \theta) = nh\frac{f^{(2m+1)}(x_n \pm \theta) - f^{(2m+1)}(x_0 \pm \theta)}{x_n - x_0}$$

D'après le théorème de la valeur moyenne, il existe $\xi \in [x_0, x_n]$ tel que :

$$\frac{f^{(2m+1)}(x_n \pm \theta) - f^{(2m+1)}(x_0 \pm \theta)}{x_n - x_0} = f^{(2m+2)}(\xi)$$

par substitution, l'expression finale du reste est :

$$R_{2m} = -\frac{nB_{2m+2}}{(2m+2)!} h^{2m+3} f^{(2m+2)}(\xi)$$

finalement la formule d'intégration d'Euler-Maclaurin est :

$$\int_{x_0}^{x_n} f(x)dx = \frac{h}{2}\left[f(x_0) + 2\sum_{\eta=1}^{n-1} f(x_\eta) + f(x_n)\right] - \sum_{k=1}^{m} \frac{B_{2k}}{(2k)!} h^{2k}\left[f^{(2k-1)}(x_n) - f^{(2k-1)}(x_0)\right] - h^{2m+3}\frac{nB_{2m+2}}{(2m+2)!} f^{(2m+2)}(\xi)$$

La formule d'Euler-Maclaurin s'emploie pour le calcul approché des intégrales définies, ainsi que la sommation approchée des valeurs des fonctions. Remarquons l'existence dans cette formule le terme :

$$\frac{h}{2}\left[f(x_0) + 2\sum_{\eta=1}^{n-1} f(x_\eta) + f(x_n)\right] \equiv I_t(f,\ h)$$

qui est l'approximation par la méthode des trapèzes de l'intégrale définie :

$$I(f) = \int_{x_0}^{x_n} f(x)dx$$

soit,

$$I(f) - I_t(f,\ h) = -\sum_{k=1}^{m} \frac{B_{2k}}{(2k)!} h^{2k}\left[f^{(2k-1)}(x_n) - f^{(2k-1)}(x_0)\right] - h^{2m+3}\frac{nB_{2m+2}}{(2m+2)!} f^{(2m+2)}(\xi)$$

Soit le changement,

$$\gamma_k = c_k\left[f^{(2k-1)}(x_n) - f^{(2k-1)}(x_0)\right] \quad \text{avec} \quad c_k = -\frac{B_{2k}}{(2k)!}$$

alors,

$$I(f) - I_t(f,\ h) = \sum_{k=1}^{m} \gamma_k h^{2k} + (x_n - x_0) c_{m+1} h^{2m+2} f^{(2m+2)}(\xi)$$

c'est-à-dire,

$$I(f) - I_t(f,\ h) = \gamma_1 h^2 + \gamma_2 h^4 + \gamma_3 h^6 + \cdots + \gamma_m h^{2m} + (x_n - x_0) c_{m+1} h^{2m+2} f^{(2m+2)}(\xi)$$

ou,

$$I(f) = I_t(f,\ h) + \gamma_1 h^2 + \gamma_2 h^4 + \gamma_3 h^6 + \cdots + \gamma_m h^{2m} + O\left(h^{2m+2}\right)$$

Soit le cas où $m = 1$, c'est-à-dire :

$$I(f) = I_t(f,\ h) + \gamma_1 h^2 + (x_n - x_0) c_2 h^4 f^{(4)}(\xi)$$

or,

$$c_1 = -\frac{B_2}{2!} = -\frac{1}{12} \text{ et } c_2 = -\frac{B_4}{4!} = \frac{1}{720}$$

donc,

$$I(f) = I_t(f,\ h) - \frac{f'(x_n) - f'(x_0)}{12} h^2 + \frac{(x_n - x_0)}{720} h^4 f^{(4)}(\xi)$$

qui s'écrit aussi,

$$I(f) = I_t(f,\ h) - \frac{(x_n - x_0)}{12} h^2 f''(\zeta) + \frac{(x_n - x_0)}{720} h^4 f^{(4)}(\xi) \quad \text{avec } \zeta,\ \xi \in [x_0, x_n]$$

Ce qui montre que cette expression montre que l'approximation de l'intégrale $I(f)$ par :

$$I_t(f,\ h) - \frac{f'(x_n) - f'(x_0)}{12} h^2$$

qui une approximation de $I(f)$ avec une erreur d'ordre 4. Remarquons aussi que $I_t(f,\ h)$ est bien une approximation de $I(f)$ avec une erreur d'ordre 2, et ce reste est exprimé par :

$$R = -\frac{f'(x_n) - f'(x_0)}{12} h^2 \equiv -\frac{h^2}{12}(x_n - x_0) f''(\zeta) \text{ avec } \zeta \in [x_0,\ x_n]$$

Dans la pratique, Il est possible d'utiliser la formule d'Euler-Maclaurin pour estimer l'intégrale à n'importe quel ordre d'erreur voulue, mais c'est possible si la fonction $f(x)$ est plusieurs fois dérivable. L'utilité de cette formule se trouve son utilisation pratique dans la méthode d'intégration de Romberg.

7. Formule d'extrapolation de Richardson

D'après la formule de quadrature de Newton-Cotes,

$$I(f) = \int_a^b f(x)dx = \sum_{j=0}^{n} \lambda_{j,n} f(x_j) + R$$

Le reste *R* est généralement exprimé sous forme :

$$R = O\left(h^m\right) \equiv \mathrm{M}h^m \quad \text{avec} \quad h = (b-a)/n$$

m est l'ordre de l'erreur d'intégration, par exemple, pour la formule des trapèzes $m=2$ et pour les deux formules de Simpson $m=4$. M, une certaine valeur considérée pour la fonction $f(x)$ sous le signe somme comme constante dans l'intervalle d'intégration [a, b]. Par exemple pour la formule des trapèzes,

$$\mathrm{M}=\frac{b-a}{12}f''(\xi)$$

et pour les deux formules de Simpson 1/3 et 3/8,

$$\mathrm{M}=\frac{b-a}{180}f^{(4)}(\xi)\quad\text{et}\quad \mathrm{M}=\frac{b-a}{80}f^{(4)}(\xi)$$

Choisissons deux pas distincts :

$$h_1=(b-a)/n_1\quad\text{et}\quad h_2=(b-a)/n_2$$

où n_1 et n_2 ($n_2 > n_1$) sont les quantités de segments partiels dans le premier et le deuxième cas. D'après la formule de quadrature,

$$I(f)=\sum_{j=0}^{n_1}\lambda_j f(x_j)+R_{n_1}\quad\Rightarrow\quad R_{n_1}=I(f)-\sum_{j=0}^{n_1}\lambda_j f(x_j)\equiv I(f)-I_{n_1}$$

$$I(f)=\sum_{j=0}^{n_2}\gamma_j f(x_j)+R_{n_2}\quad\Rightarrow\quad R_{n_2}=I(f)-\sum_{j=0}^{n_2}\gamma_j f(x_j)\equiv I(f)-I_{n_2}$$

or,

$$R_{n_1}=M\left(\frac{b-a}{n_1}\right)^m\ \text{et}\ R_{n_2}=M\left(\frac{b-a}{n_2}\right)^m$$

alors,

$$I_{n_2}-I_{n_1}=M(b-a)^m\left(\frac{1}{n_1^m}-\frac{1}{n_2^m}\right)\quad\Rightarrow\quad M=\left(\frac{n_1n_2}{b-a}\right)^m\frac{I_{n_2}-I_{n_1}}{n_2^m-n_1^m}$$

par conséquent,

$$R=\left(\frac{n_1n_2}{n}\right)^m\frac{I_{n_2}-I_{n_1}}{n_2^m-n_1^m}$$

En particulier, pour $h = h_2$, c'est-à-dire pour $n = n_2$:

$$R_{n_2}=\frac{n_1^m}{n_2^m-n_1^m}(I_{n_2}-I_{n_1})$$

alors,

$$I(f)=I_{n_2}+R_{n_2}\quad\Rightarrow\quad I=I_{n_2}+\frac{n_1^m}{n_2^m-n_1^m}(I_{n_2}-I_{n_1})=I_{n_2}+\frac{1}{\frac{n_2^m}{n_1^m}-1}(I_{n_2}-I_{n_1})$$

soit les notations,

$$\frac{n_2}{n_1}=\alpha\quad\text{et}\quad\beta=\frac{1}{\alpha^m-1}$$

soit,

$$I(f)\approx I_{n_1,n_2}=I_{n_2}+\beta(I_{n_2}-I_{n_1})=\frac{\alpha^mI_{n_2}-I_{n_1}}{\alpha^m-1}$$

Montrons que si $I_{n_2}\neq I_{n_1}$, alors I_{n_1,n_2} se trouve toujours hors du segment $[I_{n_1}, I_{n_2}]$. En effet, si $I_{n_2}>I_{n_1}$ alors, $I_{n_1,n_2}>I_{n_2}$ mais si $I_{n_2}<I_{n_1}$ alors,

$$I_{n_1,n_2} = I_{n_2} - \beta\left(I_{n_1} - I_{n_2}\right) < I_{n_2} \quad \Rightarrow \quad I_{n_1,n_2} < I_{n_1}$$

Ce qui montre bien que I_{n_1,n_2} se trouve en dehors du segment $[I_{n_1}, I_{n_2}]$, c'est-à-dire I_{n_1,n_2} s'obtient de I_{n_1} et I_{n_2} par extrapolation.

Si $I_{n_1} = I_{n_2}$, il vient évidemment

$$I_{n_1,n_2} = I_{n_1} = I_{n_2}$$

A titre d'application considérons le cas de la formule des trapèzes ($m = 2$), alors le calcul approché de l'intégrale $I(f)$ se fait par la formule :

$$I(f) \approx \frac{\alpha^2 I_{n_2} - I_{n_1}}{\alpha^2 - 1}$$

Soit à calculer par la formule d'extrapolation de Richardson l'intégrale,

$$I = \int_0^2 e^{-x^2} dx$$

Sachant que le calcul approché de cette intégrale par la formule des trapèzes pour *n* = 2 et 4 sont respectivement 0.87704 et 0.88062. Ici $\alpha = 2$, donc la formule d'extrapolation pour l'estimation de l'intégrale par extrapolation est :

$$I \approx I_n + \frac{1}{3}\left(I_{n_2} - I_{n_1}\right) = \frac{4 \times 0.88062 - 0.87704}{3} = 0.87823$$

8. Méthode d'intégration de Romberg

La méthode d'intégration de Romberg (1955) est une méthode récursive de calcul numérique d'intégrale, fondée sur l'application du procédé d'extrapolation de Richardson à la méthode des trapèzes. Soit 2^m avec *m* peut être 1, 2, ... le nombre de désertisation de l'intervalle [*a*, *b*]. L'intégrale $I(f)$ est peut-être approchée par la méthode des trapèzes :

$$I(f) \approx I_t(f,\ h_m) = h_m\left[\frac{f(a)+f(b)}{2} + \sum_{j=1}^{2^m-1} f(a + jh_m)\right] \text{ avec } h_m = \frac{b-a}{2^m}$$

Si le pas de discrétisation h_m est divisé par 2 de nouveau dans ce cas le nombre de discrétisation de l'intervalle [*a*, *b*] est 2^{m+1}, de même :

$$I(f) \approx I_t(f,\ h_{m+1}) = h_{m+1}\left[\frac{f(a)+f(b)}{2} + \sum_{j=1}^{2^{m+1}-1} f(a + jh_{m+1})\right] \text{ avec } h_{m+1} = \frac{h_m}{2} = \frac{b-a}{2^{m+1}}$$

soit,

$$I_t(f,\ h_{m+1}) = \frac{h_m}{2}\left[\frac{f(a)+f(b)}{2} + \sum_{j=1}^{2^m-1} f(a + jh_{m+1}) + \sum_{j=0}^{2^m-1} f\left(a + (2j+1)h_{m+1}\right)\right]$$

$$= \frac{h_m}{2}\left[\frac{f(a)+f(b)}{2} + \sum_{j=1}^{2^m-1} f(a + jh_{m+1})\right] + \frac{h_m}{2}\sum_{j=0}^{2^m-1} f\left(a + (2j+1)h_{m+1}\right)$$

c'est-à-dire,

$$I_t(f,\ h_{m+1}) = \frac{1}{2} I_t(f,\ h_{m+1}) + h_{m+1}\sum_{j=0}^{2^m-1} f\left(a + (2j+1)h_{m+1}\right)$$

D'après la formule d'extrapolation de Richardson l'intégrale $I(f)$ est approchée par :

$$I(f) \approx \frac{2^2 I_t(f,\ h_{m+1}) - I_t(f,\ h_m)}{2^2 - 1} = \frac{4 I_t(f,\ h_{m+1}) - I_t(f,\ h_m)}{3}$$

D'après la formule d'Euler-Maclaurin, le reste d'intégration par la méthode des trapèzes est exprimé par :

$$E(f,\ h_m) = I(f) - I_t(f,\ h_m) = \gamma_1 h_m^2 + \gamma_2 h_m^4 + \gamma_3 h_m^6 + \cdots + \gamma_\eta h_m^{2\eta} + O\left(h_m^{2\eta+2}\right)$$

alors,

$$E(f,\ h_{m+1}) = I(f) - I_t(f,\ h_{m+1}) = \gamma_1 h_{m+1}^2 + \gamma_2 h_{m+1}^4 + \gamma_3 h_{m+1}^6 + \cdots + \gamma_\eta h_{m+1}^{2\eta} + O\left(h_{m+1}^{2\eta+2}\right)$$

ou,

$$E(f,\ h_{m+1}) = I(f) - I_t(f,\ h_{m+1}) = \gamma_1 \frac{h_m^2}{2^2} + \gamma_2 \frac{h_m^4}{2^4} + \gamma_3 \frac{h_m^6}{2^6} + \cdots + \gamma_\eta \frac{h_m^{2\eta}}{2^{2\eta}} + O\left(\frac{h_m^{2\eta+2}}{2^{2\eta+2}}\right)$$

Ces deux expressions peuvent être écrites sous forme :

$$I(f) = I_t(f,\ h_m) + \gamma_1 h_m^2 + O\left(h_m^4\right) \ \Rightarrow \ I(f) - O\left(h_m^4\right) = I_t(f,\ h_m) + \gamma_1 h_m^2$$

$$I(f) = I_t(f,\ h_{m+1}) + \frac{1}{2^2}\gamma_1 h_m^2 + O\left(\frac{h_m^4}{2^4}\right) \ \Rightarrow \ 2^2 I(f) - 2^2 O\left(h_m^4\right) = 2^2 I_t(f,\ h_{m+1}) + \gamma_1 h_m^2$$

alors,

$$2^2 I(f) - 2^2 O\left(h_m^4\right) - I(f) + O\left(h_m^4\right) = 2^2 I_t(f,\ h_{m+1}) - I_t(f,\ h_m)$$

ou,

$$\left(2^2 - 1\right) I(f) - \left(2^2 - 1\right) O\left(h_m^4\right) = 2^2 I_t(f,\ h_{m+1}) - I_t(f,\ h_m)$$

alors,

$$I(f) = \frac{2^2 I_t(f,\ h_{m+1}) - I_t(f,\ h_m)}{2^2 - 1} + O\left(h_m^4\right) \equiv U(h_m) + O\left(h_m^4\right)$$

$U(h_m)$ est une approximation de l'intégrale $I(f)$ avec une erreur d'ordre 4 calculée à partir de deux approximations d'ordre 2, $I_t(f,\ h_m)$ et $I_t(f,\ h_{m+1})$.

Remarquons que $U(h_m)$ est analogue à l'expression obtenue par la formule d'extrapolation de Richardson.

Soit les approximations de l'intégrale $I(f)$ par les méthodes des trapèzes : $I_t(f,\ h_1)$ $I_t(f,\ h_2)$, $I_t(f,\ h_3)$, … D'après la formule d'extrapolation de Richardson, soit $U(h_1)$ $U(h_2)$, $U(h_3)$, … qui sont des approximations d'ordre 4, c'est-à-dire :

$$I(f) = U(h_m) + O\left(h_m^4\right) = U(h_m) + \tilde{\gamma}_2 h_m^4 + O\left(h_m^6\right)$$

aussi,

$$I(f) = U(h_{m+1}) + O\left(h_m^4\right) = U(h_{m+1}) + \tilde{\gamma}_2 h_{m+1}^4 + O\left(h_{m+1}^6\right) \equiv U(h_{m+1}) + \tilde{\gamma}_2 \frac{h_m^4}{2^4} + O\left(h_m^6\right)$$

alors,

$$2^4 I(f) = 2^4 U(h_{m+1}) + \tilde{\gamma}_2 h_m^4 + 2^4 O\left(h_m^6\right)$$

soit

$$I(f) = \frac{2^4 U(h_{m+1}) - U(h_m)}{2^4 - 1} + O\left(h_m^6\right) \equiv V(h_m) + O\left(h_m^6\right)$$

$V(h_m)$ est une approximation de l'intégrale $I(f)$ avec une erreur d'ordre 6 calculée à partir de deux approximations d'ordre 4, $U(h_m)$ et $U(h_{m+1})$.

Connaissant $V(h_1)$, $V(h_2)$, … qui sont des approximations d'ordre 4, c'est-à-dire :

$$I(f)=V(h_m)+O(h_m^6)=V(h_m)+\hat{\gamma}_3 h_m^6+O(h_m^8)$$

aussi,

$$I(f)=V(h_{m+1})+\hat{\gamma}_3\frac{h_m^6}{2^6}+O(h_m^8)$$

alors,

$$2^6 I(f)=2^6 V(h_{m+1})+\hat{\gamma}_3 h_m^6+2^6 O(h_m^8)$$

soit

$$I(f)=\frac{2^6 V(h_{m+1})-V(h_m)}{2^6-1}+O(h_m^8)\equiv W(h_m)+O(h_m^8)$$

$W(h_m)$ est une approximation de l'intégrale $I(f)$ avec une erreur d'ordre 8 calculée à partir de deux approximations d'ordre 6, $V(h_m)$ et $V(h_{m+1})$.

Soit le tableau triangulaire de Romberg des approximations de l'intégrale $I(f)$ (Fig. 5.6) :

$R(1,\ 1)=I_t(f,\ h_1)$			
$R(2,\ 1)=I_t(f,\ h_2)$	$R(2,\ 2)=U(h_1)$		
$R(3,\ 1)=I_t(f,\ h_3)$	$R(3,\ 2)=U(h_2)$	$R(3,\ 3)=V(h_1)$	
$R(4,\ 1)=I_t(f,\ h_4)$	$R(4,\ 2)=U(h_3)$	$R(4,\ 3)=V(h_2)$	$R(4,\ 4)=W(h_1)$

Fig. 5.6 : Tableau de Romberg.

avec,

$$U(h_m)=\frac{2^{2\cdot1}I_t(f,\ h_{m+1})-I_t(f,\ h_m)}{2^{2\cdot1}-1}\equiv R(m+1,\ 2),\quad m=1,\ 2,\ 3$$

$$V(h_m)=\frac{2^{2\cdot2}U(h_{m+1})-U(h_m)}{2^{2\cdot2}-1}\equiv R(m+2,\ 2),\quad m=1,\ 2$$

$$W(h_m)=\frac{2^{2\cdot3}V(h_{m+1})-V(h_m)}{2^{2\cdot3}-1}\equiv R(m+3,\ 2),\quad m=1$$

La première colonne dans le tableau, l'erreur est de l'ordre 2 et la dernière valeur dans cette colonne est la plus précise, de même pour la deuxième colonne avec une erreur d'ordre 6 et aussi pour la dernière colonne qui contient une seule valeur avec une erreur d'ordre 8, cette valeur est la plus précise estimation de l'intégrale $I(f)$.

Une généralisation c'est possible afin de construire le tableau triangulaire de Romberg. Soit $I_t(f,\ h_1)$, $I_t(f,\ h_2)$, $I_t(f,\ h_3)$, …, $I_t(f,\ h_\eta)$ qui sont les $R(i,\ 1)$:

$$R(i,\ 1)=\frac{b-a}{2^i}\left[\frac{f(a)+f(b)}{2}+\sum_{j=1}^{2^i-1}f\left(a+j\frac{b-a}{2^i}\right)\right],\quad \forall\ i=1,\ 2,\ \ldots,\ \eta$$

Le reste des éléments du tableau de Romberg est peut-être formé ainsi :

$$R(i+1,\ k)=\frac{2^{2(k-1)}R(i+1,\ k-1)-R(i,\ k-1)}{2^{2(k-1)}-1} \text{ pour } k=2,\ 3,\ \ldots,\ \eta \text{ et } i=k-1,\ k,\ \ldots,\ \eta-1$$

$R(i,\ k)$ constitue une approximation de l'intégrale $I(f)$ avec une erreur d'ordre $2k$, par conséquent pour les η approximations successives, la méthode d'intégration de Romberg permet d'avoir une bonne approximation de l'intégrale $I(f)$ avec une erreur de d'ordre 2η.

La fonction **Romberg** ci-dessous, permet l'estimation de l'intégrale d'une fonction **fun** définie et continue sur l'intervalle [**a, b**] par la méthode d'intégration de Romberg. Le script de cette fonction utilise le script de la fonction **Trap** de la méthode des trapèzes.

```
function [I,R,ErrorOrder] = Romberg(func,a,b,varargin)
% func : Nom de la fonction à intégrer,
% a, b : Bornes d'intégration (a < b)
% I : intégrale estimatée par la méthode de Romberg
% R : Tableau de Romberg
% ErroOrder : l'ordre d'erreur d'approximation de I
format long;
LocalError = 1e-05; % Critère d'arrêt entre deux estimations successives
Count=2;
while Count < 100000
  R(Count-1,1)=Trap(func,a,b,2^(Count-1));
  R(Count,1)=Trap(func,a,b,2^(Count));
  if abs(R(Count,1)-R(Count-1,1)) > LocalError
    Count=Count+1;
  else
      break
  end
end
for k=2:Count
   for i=(k-1):Count-1
       R(i+1,k)=((2^(2*k-2))*R(i+1,k-1)-R(i,k-1))/(((2^(2*k-2))-1));
   end
end
   I=R(Count,Count); ErrorOrder=2*Count;
end
```

A titre d'exemple, soit à calculer l'intégrale :

$$I(f)=\int_1^3 x^2\ln(x)dx$$

alors,

```
>> f=@(x) (x.^2).*log(x);
>> [I,R]=Romberg(f,1,3)
I =
    6.998621709124103
```

Le tableau de contrôle de Romberg est (Fig. 5.7).

7.716344									
7.177729	6.998190								
7.043377	6.998593	6.998620							
7.009809	6.998620	6.998622	6.998622						
7.001419	6.998622	6.998622	6.998622	6.998622					
6.999321	6.998622	6.998622	6.998622	6.998622	6.998622				
6.998797	6.998622	6.998622	6.998622	6.998622	6.998622	6.998622			
6.998665	6.998622	6.998622	6.998622	6.998622	6.998622	6.998622	6.998622		
6.998633	6.998622	6.998622	6.998622	6.998622	6.998622	6.998622	6.998622	6.998622	
6.998624	6.998622	6.998622	6.998622	6.998622	6.998622	6.998622	6.998622	6.998622	6.998622

Fig. 5.7 : Tableau de Romberg.

9. Méthode de quadrature adaptive

Les méthodes d'intégration approchées citées précédemment utilisent des nœuds uniformes à pas de discrétisation *h* fixe. Il est clair que si *h* est plus petit, la précision de calcul est plus améliorée, ce qui nécessite aussi beaucoup de calcul. Ces méthodes ne tiennent pas en compte la nature de la variation de la fonction $f(x)$ sur l'intervalle [*a*, *b*] plus que la précision est calculée de façon globale.

Examinons l'intégrale de la fonction $f(x)=e^{-2x}\sin 3x$ sur l'intervalle [0, 4] (Fig. 5.8), sur l'intervalle [0, 2], il y'a une forte variation de $f(x)$ par contre sur l'intervalle [2, 3] il y a une faible variation et dans la partie [3, 4] l'intégrale de la fonction est pratiquement nulle. Dans cette situation l'emploi d'une méthode composé comme la méthode des trapèzes ou de Simpson avec des nœuds équidistants est inappropriée. Pour résoudre ce type de problème en tenant compte ces variations, soit la méthode de quadrature adaptée à ce type de situation utilisé avec la méthode de Simpson 1/3. Cette méthode a été introduite pour la première fois par Kuncir (1962).

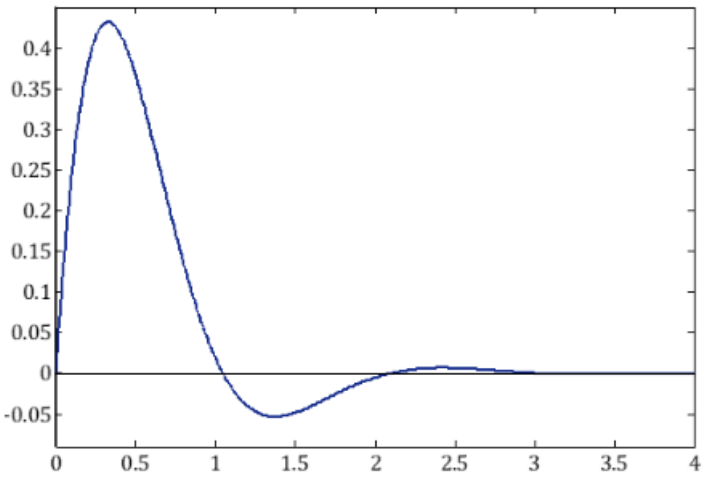
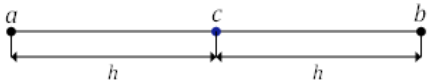


Fig. 5.8 : Illustration de la méthode de quadrature adaptée.

$$I=\int_a^b f(x)dx \quad \text{avec} \quad h=\frac{b-a}{2} \quad \text{et} \quad c=\frac{b+a}{2}$$

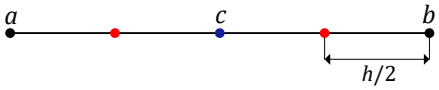


D'après la règle de Simpson 1/3 :

$$I=\int_a^b f(x)dx=\frac{h}{3}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]-\frac{1}{90}h^5 f^{(4)}(\xi), \quad \xi\in[a,b]$$

Notons,

$$S(a,\ b)=\frac{1}{3}\left(\frac{b-a}{2}\right)\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right] \text{ et } E_1=-\frac{1}{90}h^5 f^{(4)}(\xi)$$



aussi,

$$\int_a^c f(x)dx=\frac{1}{3}\left(\frac{c-a}{2}\right)\left[f(a)+4f\left(\frac{a+c}{2}\right)+f(c)\right]-\frac{1}{90}\left(\frac{h}{2}\right)^5 f^{(4)}(\xi_1),\ \ \xi_1\in[a,\,c]$$

$$\int_c^b f(x)dx=\frac{1}{3}\left(\frac{b-c}{2}\right)\left[f(c)+4f\left(\frac{c+b}{2}\right)+f(b)\right]-\frac{1}{90}\left(\frac{h}{2}\right)^5 f^{(4)}(\xi_2),\ \ \xi_2\in[c,\,b]$$

C'est-à-dire,

$$\int_a^c f(x)dx=S(a,\ c)-\frac{1}{90}\left(\frac{h}{2}\right)^5 f^{(4)}(\xi_1) \text{ et } \int_c^b f(x)dx=S(c,\ b)-\frac{1}{90}\left(\frac{h}{2}\right)^5 f^{(4)}(\xi_2)$$

or,

$$\int_a^b f(x)dx=\int_a^c f(x)dx+\int_c^b f(x)dx$$

alors,

$$I=\int_a^b f(x)dx=S(a,\ c)+S(c,\ b)-\frac{1}{90}\left(\frac{h}{2}\right)^5\left[f^{(4)}(\xi_1)+f^{(4)}(\xi_2)\right]$$

Posant,

$$E_2=-\frac{1}{90}\left(\frac{h}{2}\right)^5\left[f^{(4)}(\xi_1)+f^{(4)}(\xi_2)\right]$$

Supposant que,

$$\xi_1\approx\xi_2\approx\xi \ \Rightarrow\ f^{(4)}(\xi_1)\approx f^{(4)}(\xi_2)\approx f^{(4)}(\xi)$$

alors,

$$E_2\approx-\frac{1}{90}\frac{h^5}{2^4}f^{(4)}(\xi)\equiv\frac{E_1}{16}$$

Entre les deux expressions d'intégration,

$$S(a,\ c)+S(c,\ b)+E_2\approx S(a,\ b)+E_1 \ \Leftrightarrow\ S(a,\ c)+S(c,\ b)+E_2\approx S(a,\ b)+16E_2$$

Soit,

$$E_2\approx\frac{S(a,\ c)+S(c,\ b)-S(a,\ b)}{15}$$

Qui est une bonne estimation de l'erreur de l'intégrale *I*.

La mise en forme de cette procédure pour déterminer une bonne estimation de l'intégrale *I* avec une bonne précision connue ε. En premier lieu il faut calculer l'intégrale définie sur l'intervalle [*a*, *b*] par la méthode de Simpson 1/3 pour les pas *h*/2 et pour *h*/4, c'est-à-dire :

$$S_1=S(a,\ b)=\frac{1}{3}\left(\frac{b-a}{2}\right)\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]$$

$$S_2=S(a,\ c)+S(c,\ b)$$

si la condition,

$$|S_1-S_2|<15.\varepsilon \text{, alors } I=S_2+\frac{S_2-S_1}{15}$$

Dans le cas contraire, les deux sous intervalles [a, c] et [c, b] seront subdivisée respectivement en [a, c_1], [c_1, c] et [c, c_2], [c_2, b] avec, $c_1 = (a+c)/2$ et $c_2 = (c+b)/2$. La procédure se répète de façon récursive jusqu'à la satisfaction de la condition précédente.

Basant sur ces concepts, soit le script de la fonction récursif **`AdaptiveQuad`** qui permet le calcul approché de l'intégrale I. La fonction **`AdaptiveQuad`** utilise aussi la fonction **`simpson`** basée sur la règle de Simpson 1/3 pour le calcul approché de l'intégrale sur l'intervalle subdivisé en 2 et 4 parties égales.

```
function I = AdaptiveQuad(f,a,b,erreur)
 if nargin<4|isempty(erreur), erreur=0.0000001; end
 S1 = simpson(f,a,b,2); S2 = simpson(f,a,b,4);
 if (abs(S1-S2)<15*erreur)
   I = S2+((S2-S1)/15);
 else
   c=(a+b)/2; Left= AdaptiveQuad(f,a,c,0.5*erreur);
   Right=AdaptiveQuad(f,c,b,0.5*erreur); I = Left + Right;
end
end
```

```
function I = simpson(f,a,b,n,varargin)
   h = (b-a)/(n);  p=0;  q=0;
 for k=1:2:(n-1)
     x=a+h*k; p=p+f(x,varargin{:});
 end
  for k=2:2:(n-1)
      x=a+h*k; q=q+f(x,varargin{:});
  end
 I = h/3*(f(a,varargin{:})+f(b,varargin{:})+4*p+2*q);
end
```

Soit l'exemple :

$$\int_0^4 e^{-2x}\sin 3x dx$$

```
>> f=@(x)exp(-2*x).*sin(3*x);
>> AdaptiveQuad(f,0,4,0.00001)
ans =
    0.2307
```

10. Calcul approché des intégrales impropres

L'intégrale :

$$\int_a^b f(x)dx$$

s'appelle impropre si l'intervalle d'intégration [a, b] n'est pas finie. Soit le cas :

$$I = \int_a^{\infty} f(x)dx$$

L'intégrale est dite convergente, s'il existe une limite finie de :

$$\lim_{b\to\infty}\int_a^b f(x)dx$$

dans le cas contraire l'intégrale est dite divergente.

Soit l'intégrale sur l'intervalle $[x_i,\ x_{i+1}]$ ($x_1 = a$):

$$\int_{x_i}^{x_{i+1}} f(x)dx \quad \text{et la somme,} \quad \int_{a}^{x_{n+1}} f(x)dx = \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)dx$$

alors,

$$\lim_{n \to \infty} \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)dx = \sum_{n \geq 1} \int_{x_n}^{x_{n+1}} f(x)dx \equiv \int_{a}^{\infty} f(x)dx$$

Soit s_i le calcul approché de l'intégrale de la fonction $f(x)$ sur l'intervalle $[x_i,\ x_{i+1}]$ qui peut être obtenu par l'une des règles étudiées précédemment (trapèzes, Simpson, Gauss, ...),

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx s_i$$

soit la somme,

$$S_n = \sum_{i=1}^{n} s_i$$

Par conséquent, si $S_n \to S$ quand $n \to \infty$, alors S est une approximation de l'intégrale impropre I. Pour déterminer S, pour une tolérance ou critère d'arrêt de convergence ε, soit les deux sommes S_k, S_{k+1} et d_k, avec :

$$S_k = \sum_{i=1}^{k} s_i,\ S_{k+1} = \sum_{i=1}^{k+1} s_i,\ d_k = |S_{k+1} - S_k|$$

Si pour $i = k$, $d_k \leq \varepsilon$, la convergence est atteinte, sinon il faut vérifier pour $i = k+1$ c'est-à-dire si $d_{k+1} \leq \varepsilon$ et ainsi de suite jusqu'à la convergence.

Soit l'exemple de l'intégrale impropre :

$$\int_{0}^{\infty} \frac{dx}{x^2 + 1}$$

Cette intégrale converge analytiquement vers $\pi/2 = 1.5708\ldots$ En utilisant la méthode des trapèzes avec un pas de discrétisation h = 0.01 pour calculer approximativement les différentes intégrales sur les intervalles $[x_i,\ x_i + 1000]$, avec $x_1 = 0$ et effectuant la somme partielle jusqu'à la convergence. Soit le script MATLAB :

```
f=@(x) 1./(x.*x+1);
eps=0.000001; % Définition d'une tolérance
x(1)=0; x(2)=1000;
n(1)=numel(x(1):0.01:x(2));
S(1)=Trap(f,x(1),x(2),n(1));

for i=2:100
   x(i+1)=x(i)+1000;
       n(i)=numel(x(i):0.01:x(i+1));
     S(i)=S(i-1)+trap(f,x(i),x(i+1),n(i));
  if abs(S(i)-S(i-1))<=Tolerance, break, end
end
 i, S(i)
```

L'exécution de ce script conduit à :

```
i =
    33
ans =
    1.5708
```

La convergence atteinte dès l'ordre 33 de la somme partielle vers la valeur 1.5708

$$\int_0^{\infty} \frac{dx}{x^2+1} \approx \sum_{i=1}^{33} \int_{x_i}^{x_i+10^3} \frac{dx}{x^2+1} = 1.5708$$

Tous les formules et méthodes d'intégration vues dans les paragraphes précédentes l'expression analytique de la fonction $f(x)$ a intégrée est continue et uniformément discrétisée sur l'intervalle [a, b], mais dans certain cas en ingénierie, l'expression de la fonction $f(x)$ est inconnue et elle est donnée que par un ensemble de couples $(x_i,\ y_i)$.

L'hydrogramme de crue (Fig. 5.9) d'un bassin versant est un exemple d'une succession de débits Q_0, Q_1, …, Q_n pour les temps t_0, t_1, …, t_n. Le calcul du volume d'eau ou apport liquide écoulé entre les instants t_0 et t_n n'est autre que la surface limitée entre l'allure de l'hydrogramme et le segment $t_n - t_0$ c'est-à-dire l'intégrale :

$$\int_{t_0}^{t_n} Q(t)dt$$

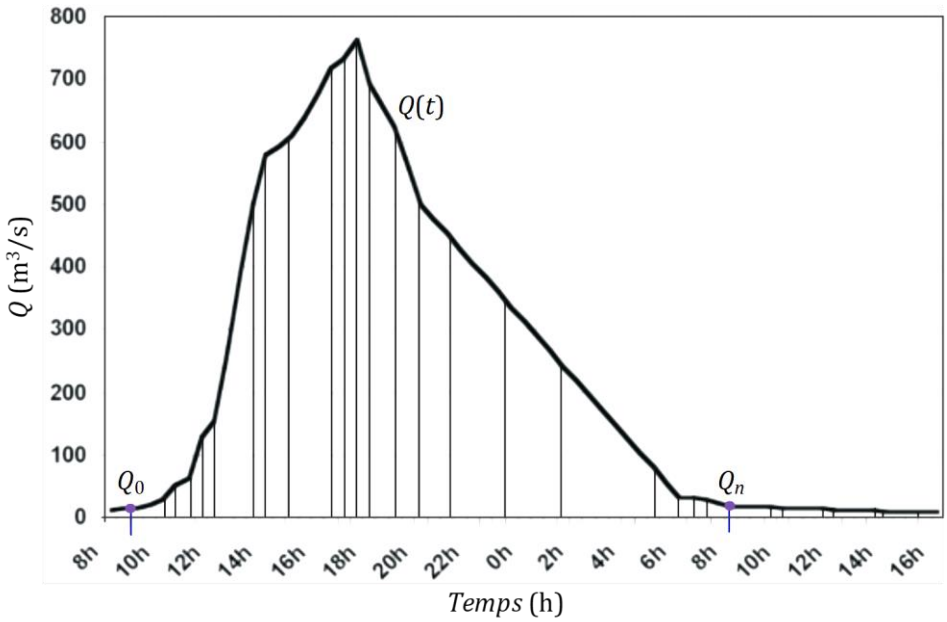


Fig. 5.9 : Exemple d'un hydrogramme de crue d'un bassin versant.

Pour évaluer l'intégrale dans ce cas, il est possible de déterminer le polynôme d'interpolation $P_n(t)$ de degré n pour l'ensemble des points (t_0,Q_0), (t_1,Q_1), …, (t_n,Q_n) et calculer l'intégrale :

$$\int_{t_0}^{t_n} Q(t)dt \approx \int_{t_0}^{t_n} P_n(t)dt$$

Mais vu le problème de Runge qui montre que les polynômes de degré supérieur ne garantissent pas une bonne approximation de $Q(t)$, les méthodes composées inspirées des méthodes des trapèzes et de Simpson 1/3 peuvent être utilisées ainsi que l'intégration des polynômes splines cubiques permettent d'avoir une bonne estimation de cette intégrale.

11. Méthodes composées

Soit les couples (x_0, y_0), (x_1, y_1), …, (x_n, y_n) formant une allure dans le plan. La surface limitée par l'allure et l'intervalle [x_0, x_n] est peut-être calculé par la somme des aires des trapèzes :

$$S \approx \frac{h_1}{2}(y_0 + y_1) + \frac{h_2}{2}(y_1 + y_2) + \ldots + \frac{h_n}{2}(y_{n-1} + y_n) = \frac{1}{2}\left(h_1 y_0 + h_n y_n + \sum_{j=1}^{n-1} (h_j + h_{j+1}) y_j \right)$$

avec,

$$h_j = x_j - x_{j-1}, \quad j = 1,\ 2,\ 3,\ \ldots,\ n$$

Soit **`Trapezoilale`** la fonction qui permet de déterminer S comme la somme de tous les trapèzes des différentes distances h_j :

```
function S = Trapezoilale(x,y)
% x = vecteur des abscisses,  y = vecteur des ordonnées
% S = la surface estimée
n=numel(x); m=numel(y);
if ~(n==m),error('les tailles des deux vecteurs sont différents'),end
if ~(n>2),error('n est faible...'),end
S=0;
for j=1:n-1
  S=S+0.5*(x(j+1)-x(j))*(y(j)+y(j+1));
end
end
```

Si n est pair $n = 2m$, il est possible d'intégrer pour chaque intervalle [x_{2k}, x_{2k+2}] pour $k = 1,\ 2,\ 3,\ \ldots,\ m$ le polynôme d'interpolation $P_{(k)}(x)$ de deuxième degré des points de coordonnées $(x_{2k},\ y_{2k})$, $(x_{2k+1},\ y_{2k+1})$ et $(x_{2k+2},\ y_{2k+2})$ et construire la somme des intégrales obtenues. Le polynôme d'interpolation $P_{(k)}(x)$ est exprimée par la méthode des différences divisées par :

$$P_{(k)}(x) = y_{2k} + a_k (x - x_{2k}) + b_k (x - x_{2k})(x - x_{2k+1})$$

a_k et b_k sont les différences divisées de second et troisième ordre :

$$a_k = \frac{y_{2k+1} - y_{2k}}{h_{2k+1}} \text{ et } b_k = \frac{y_{2k+2} - y_{2k+1}}{h_{2k+2}(h_{2k+1} + h_{2k+2})} - \frac{y_{2k+1} - y_{2k}}{h_{2k+1}(h_{2k+1} + h_{2k+2})}$$

alors,

$$s_k = \int_{x_{2k}}^{x_{2k+2}} P_{(k)}(x)dx \equiv y_{2k} \int_{x_{2k}}^{x_{2k+2}} dx + a_k \int_{x_{2k}}^{x_{2k+2}} (x - x_{2k})dx + b_k \int_{x_{2k}}^{x_{2k+2}} (x - x_{2k})(x - x_{2k+1})dx$$

c'est-à-dire,

$$s_k = (h_{2k+1} + h_{2k+2}) \left[y_{2k} + \frac{a_k}{2}(h_{2k+1} + h_{2k+2}) + b_k \left(\frac{x_{2k+2}^2 - x_{2k+2}x_{2k} + x_{2k}^2}{3} + \frac{x_{2k+2} + x_{2k}}{2} + x_{2k}x_{2k+1} \right) \right]$$

La surface limitée par l'allure et l'intervalle $[x_0,\ x_n]$ est peut-être calculé par :

$$S \approx \sum_{k=1}^{m} s_k$$

Le script MATLAB **ComposedNoUniforme** permettant de déterminer S :

```
function S = ComposedNoUniforme(x,y)
% x : vecteur des abscisses, y : vecteur des ordonnées
% S = la surface estimée
n=numel(x); m=numel(y);
if ~(n==m),error('les tailles des deux vecteurs sont différents'),end
if ~((n-1)>2),error('n est faible...'),end
Reste=mod((n-1),2);
if ~(Reste==0),error('Attention, il faut que n soit impair > 2 '),end
S=0;
for k=1:0.5*(n-1)
% a et b sont les différences divisées
j=2*k;
a=(y(j)-y(j-1))/(x(j)-x(j-1));
 b=(x(j)-x(j-1))*(y(j+1)-y(j))-(x(j+1)-x(j))*(y(j)-y(j-1));
  b=b/((x(j)-x(j-1))*(x(j+1)-x(j))*(x(j+1)-x(j-1)));
   s(k)=(y(j-1)+b*x(j-1)*x(j))*(x(j+1)-x(j-1));
  s(k)=s(k)+0.5*a*(x(j+1)-x(j-1))*(x(j+1)-x(j-1));
 s(k)=s(k)+(1/3)*b*((x(j+1)^3)-(x(j-1)^3));
s(k)=s(k)-0.5*b*(x(j-1)+x(j))*((x(j+1)^2)-(x(j-1)^2));
 S=S+s(k);
end
end
```

Rappelons que toutes les méthodes d'intégration numérique traitées précédemment, nécessite que la fonction à intégrer $f(x)$ doit être définie et continue sur l'intervalle [a, b]. Si par contre la fonction n'est pas définie sur une ou les deux bornes de l'intervalle, if faut dans ce cas prendre le problème avec une grande précaution. Penser à ce type d'intégrales :

$$\int_0^1 \frac{\sin x}{x} dx,\ \int_0^2 \frac{\sinh x}{x} dx,\ \ldots$$

Les deux fonction $\sin x/x$ et $\sinh x/x$ ne sont pas définies à 0, mais elles ont une limite qui vaut 1 quand $x \to 0$ en plus que l'intégrale de ces deux fonctions dans l'intervalle [0, b] existe. Ce type de calcul d'intégrale est peut-être résolue par l'emploi des méthodes numériques utilisées dans la résolution des équations différentielles ordinaires à valeurs initiales comme les méthodes de Runge-Kutta, la méthode d'Adams-Bashforth et la méthode d'Adams-Moulton (voir chapitre 6).

12. Application des splines cubiques

Pour chaque sous intervalle $[x_j,\ x_{j+1}]$ de $[x_0,\ x_n]$, la fonction spline cubique $S_j(x)$ est exprimée par (voir chapitre 4, section 11) :

$$S_j(x)=a_j+b_j(x-x_j)+c_j(x-x_j)^2+d_j(x-x_j)^3,\ \forall\ j=0,\ 1,\ \ldots,\ n-1$$

Par conséquent, la surface *S* est peut-être approchée par :

$$S\approx\int_{x_0}^{x_1}S_0(x)dx+\int_{x_1}^{x_2}S_1(x)dx+\cdots+\int_{x_{n-1}}^{x_n}S_{n-1}(x)dx\equiv\sum_{j=0}^{n-1}\int_{x_j}^{x_{j+1}}S_j(x)dx$$

or,

$$\int_{x_j}^{x_{j+1}}S_j(x)dx=(x_{j+1}-x_j)\left(y_j+\frac{b_j}{2}(x_{j+1}-x_j)+\frac{c_j}{3}(x_{j+1}-x_j)^2+\frac{d_j}{4}(x_{j+1}-x_j)^3\right)$$

ou,

$$\int_{x_j}^{x_{j+1}}S_j(x)dx=y_jh_{j+1}+\frac{b_j}{2}h_{j+1}^2+\frac{c_j}{3}h_{j+1}^3+\frac{d_j}{4}h_{j+1}^4$$

alors,

$$S\approx\sum_{j=0}^{n-1}y_jh_{j+1}+\frac{1}{2}\sum_{j=0}^{n-1}b_jh_{j+1}^2+\frac{1}{3}\sum_{j=0}^{n-1}c_jh_{j+1}^3+\frac{1}{4}\sum_{j=0}^{n-1}d_jh_{j+1}^4$$

En particulier si les nœuds x_0, x_1, x_2, …, x_n sont uniformes, c'est-à-dire :

$$h_1=h_2=\cdots=h_{n-1}=h$$

soit,

$$S\approx h\sum_{j=0}^{n-1}y_j+\frac{h^2}{2}\sum_{j=0}^{n-1}b_j+\frac{h^3}{3}\sum_{j=0}^{n-1}c_j+\frac{h^4}{4}\sum_{j=0}^{n-1}d_j$$

Les coefficients b_j, c_j et d_j sont déterminés selon le type de spline cubique (naturelle, périodique ou Not-a-Knot). La fonction **`IntegrationWithNaturalSpline`** permet de calculer la surface *S* sur la base d'une spline cubique naturelle.

```
function S = IntegrationWithNaturalSpline(x,y)
% x : vecteur des abscisses, y : vecteur des ordonnées
% S : la surface estimée
% x et y sont les vecteurs de données de même dimension numel(x)>=3
n=numel(x); m=numel(y);
if ~(n==m),error('les tailles des deux vecteurs sont différents'),end
if ~(n>=2),error('n est faible...'),end
 for i=1:numel(x)-1
    h(i)=x(i+1)-x(i); dy(i)=(y(i+1)-y(i))/(x(i+1)-x(i));
 end
 for i=2:numel(x)-1
    B(i)=3*(dy(i)-dy(i-1));
 end
B=B'; a(1,1)=1; a(numel(x),numel(x))=1; B(1)=0; B(numel(x))=0;
 for j=1:numel(x)
   for i=2:numel(x)-1
      if i==j
          a(i,i-1)=h(i-1); a(i,i)=2*h(i-1)+2*h(i); a(i,i+1)=h(i);
      else
        a(i,j)=0;
      end
   end
 end
```

```
c=inv(a)*B;
 for j=1:numel(x)-1
   b(j)=dy(j)-(1/3)*(2*c(j)+c(j+1))*h(j); d(j)=(c(j+1)-c(j))/(3*h(j));
 end
S=0;
for j=1:numel(x)-1
    s=y(j)*h(j)+(1/2)*b(j)*(h(j)^2);
   s=s+(1/3)*c(j)*(h(j)^3)+(1/4)*d(j)*(h(j)^4);
  S=S+s;
end
end
```

Pour vérifier l'efficacité de ces trois méthodes, soit l'intégrale

$$I(f)=\int_1^3 x^2\ln(x)dx=6.9986$$

Avec MATLAB :

```
>> f=@(x)(x.^2).*log(x);
>> x=1:(3-1)/10:3; y=f(x);
>> S1=Trapezoilale(x,y)
S1 =
    7.0273
>> S2=ComposedNoUniforme(x,y)
S2 =
    6.9986
>> S3=IntegrationWithNaturalSpline(x,y)
S3 =
    6.9954
```

Pour le même nombre de discrétisation de l'intervalle d'intégration [1, 3], la méthode composé issue du principe de la méthode de Simpson 1/3 donne une meilleure estimation de l'intégrale $I(f)$ et en deuxième c'est celle basée sur l'intégration des splines cubiques naturelles. Donc, est fort utile d'utiliser l'une de ces trois méthodes et suivant la nature de la surface a calculée.

13. Calcul approché des intégrales multiples

Soit l'intégrale double :

$$I=\iint_R f(x,\ y)dA=\int_a^b\left(\int_c^d f(x,\ y)dy\right)dx$$

ou *R* est une région rectangulaire :

$$R=\{(x,\ y)\ \ a\le x\le b,\ c\le y\le d\}$$

13.1. Méthode des trapèzes

D'après la méthode des trapèzes :

$$\int_c^d f(x,\ y)dy\approx\left(\frac{d-c}{4}\right)\left[f(x,\ c)+f(x,\ d)+2f\left(x,\ \frac{c+d}{2}\right)\right]$$

alors,

$$I=\int_a^b\left(\int_c^d f(x,\ y)dy\right)dx\approx\int_a^b\left(\left(\frac{d-c}{4}\right)\left[f(x,\ c)+f(x,\ d)+2f\left(x,\ \frac{c+d}{2}\right)\right]\right)dx$$

c'est-à-dire,

$$I=\int_a^b\left(\int_c^d f(x,\ y)dy\right)dx\approx\left(\frac{d-c}{4}\right)\left[\int_a^b f(x,\ c)dx+\int_a^b f(x,\ d)dx+2\int_a^b f\left(x,\ \frac{c+d}{2}\right)dx\right]$$

de même par la méthode des trapèzes,

$$\int_a^b f(x,\ c)dx\approx\left(\frac{b-a}{4}\right)\left[f(a,\ c)+f(b,\ c)+2f\left(\frac{a+b}{2},\ c\right)\right]$$

$$\int_a^b f(x,\ d)dx\approx\left(\frac{b-a}{4}\right)\left[f(a,\ d)+f(b,\ d)+2f\left(\frac{a+b}{2},\ d\right)\right]$$

$$\int_a^b f\left(x,\ \frac{c+d}{2}\right)dx\approx\left(\frac{b-a}{4}\right)\left[f\left(a,\ \frac{c+d}{2}\right)+f\left(b,\ \frac{c+d}{2}\right)+2f\left(\frac{a+b}{2},\ \frac{c+d}{2}\right)\right]$$

soit,

$$I=\int_a^b\left(\int_c^d f(x,\ y)dy\right)dx\approx\left(\frac{b-a}{4}\right)\left(\frac{d-c}{4}\right)\left[f(a,\ c)+f(b,\ c)+2f\left(\frac{a+b}{2},\ c\right)+f(a,\ d)+f(b,\ d)\right.$$
$$\left.+2f\left(\frac{a+b}{2},\ d\right)+2f\left(a,\ \frac{c+d}{2}\right)+2f\left(b,\ \frac{c+d}{2}\right)+4f\left(\frac{a+b}{2},\ \frac{c+d}{2}\right)\right]$$

Cette approximation de l'intégrale I à pour ordre :

$$O\left[(b-a)(d-c)\left((b-a)^2+(d-c)^2\right)\right]$$

13.2. Méthode de Simpson

Les deux intervalles $[a, b]$ et $[c, d]$ sont subdivisés en un nombre pair, c'est-à-dire :

$$a=x_0,\ x_1,\ x_2,\ \ldots,\ x_n=b \text{ et } c=y_0,\ y_1,\ y_2,\ \ldots,\ y_m=d$$

soit,

$$h=(b-a)/n \quad \text{et} \quad k=(d-c)/m$$

d'après la méthode de Simpson :

$$\int_c^d f(x,\ y)dy=\frac{k}{3}\left[f(x,\ y_0)+2\sum_{j=1}^{(m/2)-1}f(x,\ y_{2j})+4\sum_{j=1}^{m/2}f(x,\ y_{2j-1})+f(x,\ y_m)\right]-\frac{(d-c)k^4}{180}\frac{\partial^4 f}{\partial y^4}(x,\ \mu)$$

avec,

$$\mu\in[c, d]$$

alors,

$$\int_a^b\left(\int_c^d f(x,\ y)dy\right)dx=\frac{k}{3}\left[\int_a^b f(x,\ y_0)dx+2\sum_{j=1}^{(m/2)-1}\int_a^b f(x,\ y_{2j})dx+4\sum_{j=1}^{m/2}\int_a^b f(x,\ y_{2j-1})dx+\int_a^b f(x,\ y_m)dx\right]$$
$$-\frac{(d-c)k^4}{180}\int_a^b\frac{\partial^4 f}{\partial y^4}(x,\ \mu)dx$$

$\forall\ j=0,\ 1,\ 2,\ \ldots,\ m,$

$$\int_a^b f(x,\ y_j)dx=\frac{h}{3}\left[f(x_0,\ y_j)+2\sum_{i=1}^{(n/2)-1}f(x_{2i},\ y_j)+4\sum_{i=1}^{n/2}f(x_{2i-1},\ y_j)+f(x_n,\ y_j)\right]-\frac{(b-a)h^4}{180}\frac{\partial^4 f}{\partial y^4}(\xi_j,\ y_j)$$

avec,

$$\xi_j\in[a,\ b]$$

L'approximation de l'intégrale *I* par la méthode de Simpson est :

$$\int_a^b\left(\int_c^d f(x,\ y)dy\right)dx \approx \frac{hk}{9}\left\{\left[f(x_0,\ y_0)+2\sum_{i=1}^{(n/2)-1}f(x_{2i},\ y_0)+4\sum_{i=1}^{n/2}f(x_{2i-1},\ y_0)+f(x_n,\ y_0)\right]\right.$$
$$+2\left[\sum_{j=1}^{(m/2)-1}f(x_0,\ y_{2j})+2\sum_{j=1}^{(m/2)-1}\sum_{i=1}^{(n/2)-1}f(x_{2i},\ y_{2j})+4\sum_{j=1}^{(m/2)-1}\sum_{i=1}^{n/2}f(x_{2i-1},\ y_{2j})+\sum_{j=1}^{(m/2)-1}f(x_n,\ y_{2j})\right]$$
$$+4\left[\sum_{j=1}^{m/2}f(x_0,\ y_{2j-1})+2\sum_{j=1}^{m/2}\sum_{i=1}^{(n/2)-1}f(x_{2i},\ y_{2j-1})+4\sum_{j=1}^{m/2}\sum_{i=1}^{n/2}f(x_{2i-1},\ y_{2j-1})+\sum_{j=1}^{m/2}f(x_n,\ y_{2j-1})\right]$$
$$\left.+\left[f(x_0,\ y_m)+2\sum_{i=1}^{(n/2)-1}f(x_{2i},\ y_m)+4\sum_{i=1}^{n/2}f(x_{2i-1},\ y_m)+f(x_n,\ y_m)\right]\right\}$$

Le terme d'erreur *E* est exprimé par :

$$E=\frac{-k(b-a)h^4}{540}\left[\frac{\partial^4 f}{\partial x^4}(\xi_0,\ y_0)+2\sum_{j=1}^{(m/2)-1}\frac{\partial^4 f}{\partial x^4}(\xi_{2j},\ y_{2j})+4\sum_{j=1}^{m/2}\frac{\partial^4 f}{\partial x^4}(\xi_{2j-1},\ y_{2j-1})+\frac{\partial^4 f}{\partial x^4}(\xi_m,\ y_m)\right]$$
$$-\frac{(d-c)k^4}{180}\int_a^b\frac{\partial^4 f}{\partial y^4}(x,\ \mu)dx$$

Si les termes $\partial^4 f/\partial x^4$ et $\partial^4 f/\partial y^4$ sont continus et d'après le théorème de la moyenne, ils existent $(\overline{\eta},\ \overline{\mu})$ et $(\hat{\eta},\ \hat{\mu})$ de *R* tel que :

$$\frac{\partial^4 f}{\partial x^4}(\xi_0,\ y_0)+2\sum_{j=1}^{(m/2)-1}\frac{\partial^4 f}{\partial x^4}(\xi_{2j},\ y_{2j})+4\sum_{j=1}^{m/2}\frac{\partial^4 f}{\partial x^4}(\xi_{2j-1},\ y_{2j-1})+\frac{\partial^4 f}{\partial x^4}(\xi_m,\ y_m)=3m\frac{\partial^4 f}{\partial x^4}(\overline{\eta},\ \overline{\mu})$$

et,

$$\int_a^b\frac{\partial^4 f}{\partial y^4}(x,\ \mu)dx=(b-a)\frac{\partial^4 f}{\partial y^4}(\hat{\eta},\ \hat{\mu})$$

par conséquent,

$$E=\frac{-k(b-a)h^4}{540}\left[3m\frac{\partial^4 f}{\partial x^4}(\overline{\eta},\ \overline{\mu})\right]-\frac{(d-c)(b-a)}{180}k^4\frac{\partial^4 f}{\partial y^4}(\hat{\eta},\ \hat{\mu})$$

ou,

$$E=-\frac{(d-c)(b-a)}{180}\left[h^4\frac{\partial^4 f}{\partial x^4}(\overline{\eta},\ \overline{\mu})+k^4\frac{\partial^4 f}{\partial x^4}(\hat{\eta},\ \hat{\mu})\right]$$

14. MATLAB et le calcul d'intégrales

MATLAB offre des fonctions préprogrammées permettant le calcul approché des intégrales définies et impropres simples et multiples. La méthode des trapèzes se fait tout simplement avec la fonction **`trapz`**, soit l'exemple :

```
>> f=@(x) (x.^-2).*exp(-x-x.^2);
>> x=1:0.01:2; % définition d'un pas de discrétisation h = 0.01
>> y=f(x);
>> I=trapz(x,y)
I =
    0.0265
```

La fonction **quad**, est basée sur la méthode adaptée de quadrature de Simpson, soit l'exemple :

```
>> f=@(x) (x.^-2).*exp(-x-x.^2);
>> I=quad(f,1,2)
I =
   0.0265
```

La fonction **integral** basée sur la méthode adaptée de quadrature, elle permet le calcul des intégrales définies et les intégrales impropres, par exemple :

```
>> f=@(x) (x.^-2).*exp(-x-x.^2);
>> I=integral(f,1,2)
I =
   0.0265
```

Un autre exemple, consiste à calculer une intégrale impropre avec la fonction **integral** de la fonction exponentielle factorielle $ef_1(2)$ de Halphen(1955) :

$$ef_1(2)=2\int_0^{\infty} x\exp\left(-x^2+2x\right)dx$$

```
f=@(x) 2*x.*exp(-x.*x+2*x);
integral(f,0,inf);
ans =
   9.8782
```

Il est possible avec MATLAB de calculer numériquement une intégrale multiple (double ou triple) à travers une surface ou un volume par les fonctions **integral2** et **integral3** comme suit :

$$\int_{x=x_{\min}}^{x_{\max}}\int_{y=y_{\min}}^{y_{\max}} f(x,\ y)dxdy \ \approx \ \texttt{integral2(f,xmin,xmax,ymin,ymax)}$$

$$\int_{x=x_{\min}}^{x_{\max}}\int_{y=y_{\min}}^{y_{\max}}\int_{z=z_{\min}}^{z_{\max}} f(x,\ y,\ z)dxdydz \ \approx \ \texttt{integral3(f,xmin,xmax,ymin,ymax,zmin,zmax)}$$

Soit l'exemple suivant qui consiste à calculer l'intégrale double I de la fonction $f(x,\ y)=x(y+1)$ à travers la surface triangulaire A (Fig. 5.10) :

$$I=\iint_{\mathrm{A}} x(y+1)dxdy$$

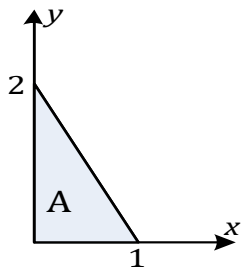


Fig. 5.10 : Surface d'intégration.

La surface A peut-être explicitée ainsi :

$$\mathrm{A}=\left\{0\le x\le 1,\ \ 0\le y\le -2x+2\right\}$$

par conséquent,

$$I=\iint_{\mathrm{A}} x(y+1)dxdy=\int_{x=0}^{1}\left(\int_{y=0}^{-2x+2} x(y+1)dy\right)dx$$

Avec MATLAB,

```
>> f=@(x,y) x.*(y+1);
>> ymax=@(x) -2*x+2;
>> I=integral2(f,0,1,0,ymax)
I =
    0.5000
```

L'exemple suivant consiste à calculer l'intégrale triple Q de la fonction $f(x,\ y,\ z)=x(\cos y+x\cos z)$ à travers le volume V de la sphère concentrique de rayon égale à l'unité (Fig. 5.11), c'est-à-dire :

$$Q=\iiint_{V} x(\cos y+x\cos z)dxdydz$$

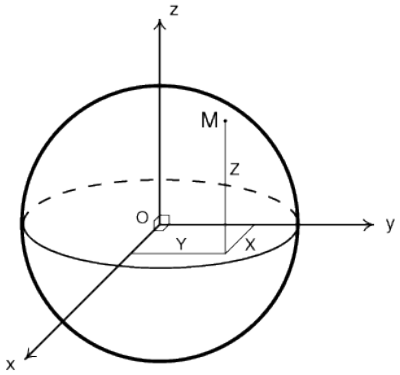


Fig. 5.11 : Volume d'intégration.

Le volume V peut être explicité ainsi :

$$V=\left\{-1\le x\le 1;\ -\sqrt{1-x^2}\le y\le\sqrt{1-x^2};\ -\sqrt{1-x^2-y^2}\le z\le\sqrt{1-x^2-y^2}\right\}$$

Avec MATLAB,

```
>> f2xyz=@(x,y,z) x.*(cos(y)+ x.*cos(z));
>> xmin = -1; xmax = 1;
>> ymin = @(x)-sqrt(1 - x.^2); ymax = @(x) sqrt(1 - x.^2);
>> zmin = @(x,y)-sqrt(1 - x.^2 - y.^2);
>> zmax = @(x,y) sqrt(1 - x.^2 - y.^2);
>> Q = integral3(f2xyz,xmin,xmax,ymin,ymax,zmin,zmax)
Q =
    0.7796
```

Le calcul des intégrales par les méthodes numériques est une alternatif pour celle ou son calcul par les méthodes analytiques est fastidieuse. Le développement de l'analyse mathématique a permis de résoudre pas mal de problèmes liés à l'intégration des fonctions (Nahin, 2015 et Cornel Ioan Vălean, 2019). Avec la fonction **`int`** de l'outil symbolique de MATLAB est un moyen très utile pour le calcul analytique des intégrales et la détermination de la fonction primitive, par exemple l'intégrale :

$$I=\int_{1}^{3} x^2\log(x)dx$$

Peut être traité ainsi :

```
>> syms x  % x ici est une variable symbolique
>> f=sym((x^2)*log(x)); % Expression symbolique de la fonction f(x)
>>F= int(f);
>> int(f,1,3);
>> 9*log(3) - 26/9
ans =
    6.9986
```

Ces commandes montrent que la fonction primitive de la fonction $f(x)=x^2\log(x)$ est la fonction $F(x)=x^3\left(\log(x)-1/3\right)/3$, et par conséquent l'intégrale I est :

$$I=\int_1^3 x^2\log(x)dx=\frac{x^3}{3}\log(x)\Big|_1^3-\frac{x^3}{9}\Big|_1^3=9\log(3)-\frac{26}{9}\Big|=6.9986$$

Certaines intégrales ne peuvent pas être calculée par l'outil symbolique de MATLAB, soit l'exemple de la fonction *intégrale sinus* :

$$\mathrm{Si}(t)=\int_0^t\frac{\sin(x)}{x}dx$$

Il est clair que cette intégrale existe pour $t>0$, mais la fonction $f(x)=\sin(x)/x$ qui n'est pas définie à $x=0$ et sa primitive ne peut pas être déterminée analytiquement, dans ce cas il faut utiliser les méthodes numériques en calculant tous d'abord sa limite à $x=0$ et déterminer par la suite la solution approchée.

15. Exercices résolus

Exercice 1

Montrer dans le cas de la méthode des trapèzes dont l'intervalle [*a*, *b*] est divisé en plusieurs segments de différentes distances que les conditions sur coefficients Cotes sont bien vérifiées.

Soit $f(x)$ une fonction continue [*a*, *b*], la méthode des trapèzes utilisée pour un calcul approché de l'intégrale est exprimé par :

$$I(f)\approx h\left[\frac{f(x_0)}{2}+f(x_1)+f(x_2)+\ldots+f(x_{n-1})+\frac{f(x_n)}{2}\right]\equiv\sum_{j=0}^{n}w_jf(x_j)$$

remarquons que les facteurs poids sont :

$$w_0=h/2,\quad w_j=h\ (\forall\ j=1,\ 2,\ 3,\ \ldots,\ n-1)\ \text{ et }\ w_n=h/2$$

par conséquent les coefficients de Cotes :

$$H_0=1/2,\quad H_j=1\ (j=1,\ 2,\ 3,\ \ldots,\ n-1)\ \text{ et }\ H_n=1/2$$

qui vérifient bien les conditions :

$$\sum_{j=0}^{n}H_j=1\ \text{ et }\ H_j=H_{n-j}$$

Exercice 2

Considérons la fonction $f(x)$ tabulé ci-dessous. Evaluer l'intégrale $\int_{0.4}^{2.0}f(x)dx$ par la méthode des trapèzes pour *n* = 1, 2, 4, et 8 intervalles.

x	*f*(*x*)	*x*	*f*(*x*)
0.4	5.1600	1.4	3.3886
0.6	3.6933	1.6	3.8100
0.8	3.1400	1.8	4.3511
1.0	3.0000	2.0	5.0000
1.2	3.1067		

La valeur exacte est 5.86. Calculer les erreurs dans ce cas.

D'après la méthode des trapèzes :

$$\int_{0.4}^{2.0} f(x)dx \approx h\left[\frac{f(x_0)}{2} + f(x_1) + f(x_2) + \ldots + f(x_{n-1}) + \frac{f(x_n)}{2}\right]$$

pour $n = 1$,

$$\int_{0.4}^{2.0} f(x)dx \approx I_{n=1} = \left(\frac{2.0-0.4}{1}\right)\left[\frac{5.1600}{2} + \frac{5.0000}{2}\right] = 8.1280$$

pour $n = 2$,

$$\int_{0.4}^{2.0} f(x)dx \approx I_{n=2} = \left(\frac{2.0-0.4}{2}\right)\left[\frac{5.1600}{2} + 3.1067 + \frac{5.0000}{2}\right] = 6.5494$$

pour $n = 4$,

$$\int_{0.4}^{2.0} f(x)dx \approx I_{n=4} = \left(\frac{2.0-0.4}{4}\right)\left[\frac{5.1600}{2} + 3.1400 + 3.1067 + 3.8100 + \frac{5.0000}{2}\right] = 6.0547$$

pour $n = 8$,

$$\int_{0.4}^{2.0} f(x)dx \approx I_{n=8} = \left(\frac{2.0-0.4}{8}\right)\left[\frac{5.1600}{2} + 3.6933 + 3.1400 + 3.0000 + 3.1067 + 3.3886 + 3.8100 + 4.3511 + \frac{5.0000}{2}\right] = 5.9139$$

Le tableau ci-dessous résumé les résultats obtenus ainsi que les erreurs d'intégration suivant cette méthode

n	***I_n***	**Erreur**	**Erreur relative (%)**
1	8.1280	2.2680	38.70
2	6.5494	0.6894	11.76
4	6.0547	0.1947	3.32
8	5.9139	0.0539	0.92

Exercice 3

En utilisant la méthode de Simpson 1/3, évaluer l'intégrale :

$$I = \int_0^1 \frac{dx}{x^2 + 6x + 10}$$

avec 2 et 4 sous intervalles et comparer avec la solution exacte.

Les sous intervalles de [0, 1] sont représentés dans les deux tableaux

$n = 2$,

x	0.0	0.5	1.0
$f(x)$	0.1	0.07547	0.05882

$n = 4$,

x	0.00	0.25	0.50	0.75	1.00
$f(x)$	0.10000	0.08649	0.07547	0.06639	0.05882

Alors, d'après la méthode Simpson 1/3 on a pour les deux cas ($n = 2$ et $n = 4$) :

$$I_1 = \frac{h}{3}\left[f(0.0) + 4f(0.5) + f(1.0)\right] = \frac{0.5}{3}\left[0.1 + 4(0.07547) + 0.05882\right] = 0.07678.$$

$$I_2 = \frac{h}{3}\left[f(0.0)+4f(0.25)+4f(0.75)+2f(0.75)+f(1.0)\right]$$
$$= \frac{0.25}{3}\left[0.1+4(0.08649)+4(0.06639)+2(0.07547)+0.05882\right] = 0.07677.$$

La valeur exacte de l'intégrale :

$$\text{Exact} = \int_0^1 \frac{dx}{x^2+6x+10} = \left[\tan^{-1}(x+3)\right]_0^1 = \tan^{-1}(4)-\tan^{-1}(3) = 0.07677.$$

L'erreur entre la valeur exacte et la valeur approchée :

$$\left|\text{Exact} - I_1\right| = \left|0.07677 - 0.07678\right| = 0.00001.$$
$$\left|\text{Exact} - I_2\right| = \left|0.07677 - 0.07677\right| = 0.00000.$$

Exercice 4

Utiliser la méthode des trapèzes pour calculer approximativement les intégrales :

$$\int_0^2 (1+\sin \pi x)dx \quad \text{et} \quad \int_0^{\pi} \sin^2 x dx$$

Comparer le résultat obtenu avec celle calculé par la formule d'intégration d'Euler-Maclaurin.

Considérons la formule d'intégration approchée d'Euler-Maclaurin suivante :

$$\int_a^b f(x)dx \approx h\left[\frac{1}{2}f(a)+\sum_{i=1}^{n-1} f(x_i)+\frac{1}{2}f(b)\right] - \sum_{k=1}^{6} \frac{B_{2k}}{(2k)!}h^{2k}\left[f^{(2k-1)}(b)-f^{(2k-1)}(a)\right]$$

Le terme,

$$h\left[\frac{1}{2}f(a)+\sum_{i=1}^{n-1} f(x_i)+\frac{1}{2}f(b)\right] \equiv I_t(f,\ h)$$

Est le calcul approché de l'intégrale par la méthode des trapèzes.

Soit le script suivant qui permet de calculer l'intégrale $\int_0^2 (1+\sin \pi x)dx$ par les deux méthodes :

```
syms x
f=1+sin(pi*x);
f1=diff(f);f2=diff(f,3);f3=diff(f,5);f4=diff(f,7);f5=diff(f,9);
f6=diff(f,11);

a=0; b=2;
n=5;
h=(b-a)/n;

f=matlabFunction(f);f1=matlabFunction(f1);f2=matlabFunction(f2);
f3=matlabFunction(f3);f4=matlabFunction(f4);f5=matlabFunction(f5);
f6=matlabFunction(f6);

B=[1/6 -1/30 1/42 -1/30 5/66 -691/2730];

Somme=(B(1)/factorial(2))*(h^2)*(f1(b)-f1(a));
Somme=Somme+(B(2)/factorial(4))*(h^4)*(f2(b)-f2(a));
Somme=Somme+(B(3)/factorial(6))*(h^6)*(f3(b)-f3(a));
Somme=Somme+(B(4)/factorial(8))*(h^8)*(f4(b)-f4(a));
Somme=Somme+(B(5)/factorial(10))*(h^10)*(f5(b)-f5(a));
Somme=Somme+(B(6)/factorial(12))*(h^12)*(f6(b)-f6(a));

ParTrapezes=Trap(f,a,b,n)
ParEulerMaclaurin=Trap(f,a,b,n)-Somme
```

L'exécution de ce script conduit à :

```
ParTrapezes =
     2
ParEulerMaclaurin =
     2
```

De même pour calculer l'intégrale $\int_0^{\pi} \sin^2 x dx$, soit le script :

```
syms x
f=sin(x)*sin(x);
f1=diff(f);f2=diff(f,3);f3=diff(f,5);f4=diff(f,7);f5=diff(f,9);
f6=diff(f,11);

a=0;
b=pi;
n=3;
h=(b-a)/n;

f=matlabFunction(f);f1=matlabFunction(f1);f2=matlabFunction(f2);
f3=matlabFunction(f3);f4=matlabFunction(f4);f5=matlabFunction(f5);
f6=matlabFunction(f6);

B=[1/6 -1/30 1/42 -1/30 5/66 -691/2730];

Somme=(B(1)/factorial(2))*(h^2)*(f1(b)-f1(a));
Somme=Somme+(B(2)/factorial(4))*(h^4)*(f2(b)-f2(a));
Somme=Somme+(B(3)/factorial(6))*(h^6)*(f3(b)-f3(a));
Somme=Somme+(B(4)/factorial(8))*(h^8)*(f4(b)-f4(a));
Somme=Somme+(B(5)/factorial(10))*(h^10)*(f5(b)-f5(a));
Somme=Somme+(B(6)/factorial(12))*(h^12)*(f6(b)-f6(a));

ParTrapezes=trap(f,a,b,n)
ParEulerMaclaurin=Trap(f,a,b,n)-Somme
```

L'exécution de ce script conduit à :

```
ParTrapezes =
    1.5708
ParEulerMaclaurin =
    1.5708
```

Exercice 5

L'équation,

$$\int_0^x \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt = 0.45$$

peut être résolue par la méthode de Newton sachant :

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-t^2/2} dt - 0.45 \quad \text{et} \quad f'(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

Déterminer une solution de l'équation $f(x) = 0$ par la méthode de Newton pour une première itération ($x_1 = 0.5$) en utilisant la méthode de Simpson 1/3 pour le calcul d'intégrale.

Le processus itératif de Newton pour la solution numérique de l'équation $f(x)=0$ est exprimé par :

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

c'est-à-dire,

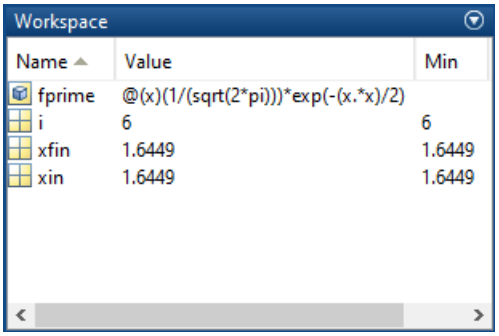
$$x_{k+1} = x_k - \frac{\frac{1}{\sqrt{2\pi}}\int_0^{x_k} e^{-t^2/2}dt - 0.45}{\frac{1}{\sqrt{2\pi}}e^{-x_k^2/2}}$$

Sachant que $x_1 = 0.5$, alors,

$$x_2 = 0.5 - \frac{\frac{1}{\sqrt{2\pi}}\int_0^{0.5} e^{-t^2/2}dt - 0.45}{\frac{1}{\sqrt{2\pi}}e^{-0.5^2/2}}$$

x_2 est calculé à partir du calcul de l'intégrale $\int_0^{0.5} e^{-t^2/2}dt$ qui sera déterminée par une des méthodes d'intégration approchée (trapèzes, Simpson ou autres...). Une fois x_2 est calculé x_3 sera calculé de la même manière jusqu'à la convergence. Soit le script MATLAB :

```
% définition de la fonction f'(x)
fprime=@(x)(1/(sqrt(2*pi)))*exp(-(x.*x)/2);
% définition de la première itération
xin=0.5;
for i=1:200
xfin=xin-((Simpson1by3(fprime,0,xin,10)-
0.45)/fprime(xin));
% Test de convergence
if abs(xfin-xin)<=0.00001
break
end
xin = xfin;
end
```



L'exécution du script montre bien que le processus converge dès la sixième itération vers 1.6449 solution de l'équation intégrale :

$$\int_0^x \frac{1}{\sqrt{2\pi}} e^{-t^2/2}dt - 0.45 = 0$$

Exercice 6

Considérons l'intégrale,

$$I = \int_{-1}^{1} |x| dx \equiv 1$$

Soit $T(h=2/n)$ l'approximation de l'intégrale I par la méthode de trapèzes.

1. Montrer que $T(2/n) = I = 1$ si n est pair.

2. Déterminer l'expression de $T(2/n)$ quand n est impair et commenter la vitesse de convergence vers la valeur exacte de l'intégrale.

1. La figure 5.12 montre la division en $n = 2m$ (nombre pair) de nœuds de l'intervalle [-1, 1], par conséquent $h = 2/2m = 1/m$. Soit alors,

$$x_0 = -1,\ x_1 = -1 + 1/m,\ x_2 = -1 + 1/m + 1/m = -1 + 2/m,\ x_3 = -1 + 3/m, \ldots$$

$$\ldots,\ x_i = -1 + i/m, \ldots,\ x_m = -1 + m/m = 0$$

par conséquent,

$$f(x_i) = 1 - i/m,\ \ \forall i = 1,\ 2,\ \ldots,\ m-1$$

aussi,

$$x_{m+1} = 1/m,\ x_{m+2} = 2/m,\ x_{m+3} = 3/m, \ldots,\ x_{m+i} = i/m, \ldots,\ x_{m+(m-1)} \equiv x_{2m-1} = (m-1)/m$$

alors,

$$f(x_{m+i}) = i/m,\ \ \forall i = 1,\ 2,\ \ldots,\ m-1$$

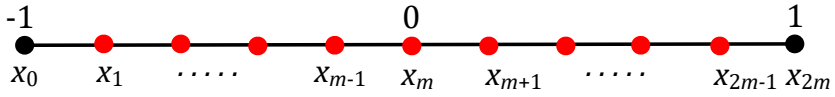


Fig. 5.12 : Division de l'intervalle [-1, 1] en un nombre pair de nœuds.

soit,

$$T(2/n) = \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{2m-1} f(x_i) + f(x_{2m}) \right]$$

$$T(2/n) = \frac{h}{2}\left[1 + 2\sum_{i=1}^{2m-1} f(x_i) + 1 \right] = h\left[1 + \sum_{i=1}^{2m-1} f(x_i) \right] = h\left[1 + \sum_{i=1}^{m-1} f(x_i) + 0 + \sum_{i=1}^{m-1} f(x_{m+i}) \right]$$

$$T(2/n) = h\left[1 + \sum_{i=1}^{m-1} \left(f(x_i) + f(x_{m+i}) \right) \right] = h\left[1 + \sum_{i=1}^{m-1} \left(1 - \frac{i}{m} + \frac{i}{m} \right) \right] = h\left[1 + \sum_{i=1}^{m-1} 1 \right] = h\left[1 + (m-1) \right] = mh = \frac{m}{m} = 1$$

2. La figure 5.13 montre la division en $n = 2m+1$ (nombre impair) de nœuds de [-1, 1], par conséquent $h = 2/(2m+1)$. Soit alors,

$$x_0 = -1,\ x_1 = -1 + 2/(2m+1),\ x_2 = -1 + 2 \times 2/(2m+1),\ x_3 = -1 + 3 \times 2/(2m+1), \ldots$$

$$\ldots,\ x_i = -1 + i \times 2/(2m+1), \ldots,\ x_m = -1 + m \times 2/(2m+1)$$

par conséquent,

$$f(x_i) = 1 - \frac{2i}{2m+1},\ \ \forall i = 1,\ 2,\ \ldots,\ m$$

aussi,

$$x_{m+1} = 0 + \frac{1}{2}\frac{2}{2m+1} = \frac{1}{2m+1},\ x_{m+2} = \frac{1}{2m+1} + \frac{2}{2m+1} = \frac{3}{2m+1},\ x_{m+3} = \frac{3}{2m+1} + \frac{2}{2m+1} = \frac{5}{2m+1}, \ldots,$$

$$x_{m+i} = \frac{2i-1}{2m+1}, \ldots,\ x_{m+m} \equiv x_{2m} = \frac{2m-1}{2m+1}$$

par conséquent,

$$f(x_{m+i}) = \frac{2i-1}{2m+1},\ \ \forall i = 1,\ 2,\ \ldots,\ m$$

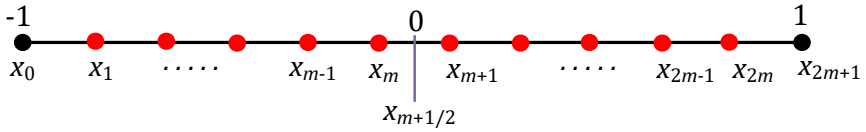


Fig. 5.13 : Division de l'intervalle [-1, 1] en un nombre impair de nœuds.

alors,

$$T(2/n)=\frac{h}{2}\left[f(x_0)+2\sum_{i=1}^{2m}f(x_i)+f(x_{2m+1})\right]$$

$$T(2/n)=\frac{h}{2}\left[1+2\sum_{i=1}^{2m}f(x_i)+1\right]=h\left[1+\sum_{i=1}^{2m}f(x_i)\right]=h\left[1+\sum_{i=1}^{m}f(x_i)+\sum_{i=1}^{m}f(x_{m+i})\right]$$

$$T(2/n)=h\left[1+\sum_{i=1}^{m}\left(1-\frac{2i}{2m+1}+\frac{2i-1}{2m+1}\right)\right]=h\left[1+\sum_{i=1}^{m}\left(1-\frac{1}{2m+1}\right)\right]=h\left[1+m\left(1-\frac{1}{2m+1}\right)\right]$$

$$T(2/n)=h\left[1+\frac{2m^2}{2m+1}\right]=\frac{2}{2m+1}\left[\frac{2m^2+2m+1}{2m+1}\right]=\frac{4m^2+4m+2}{(2m+1)^2}=\frac{(2m+1)^2+1}{(2m+1)^2}=1+\frac{1}{(2m+1)^2}$$

Exercice 7

Estimer le nombre des sous intervalles pour une approximation de l'intégrale $\int_0^1 e^{-x^2}dx$ avec une erreur de 0.5×10^{-6}, si :
(a) la méthode est celle des trapèzes, (b) la méthode est celle de Simpson.

(a) L'erreur d'intégration *E* due à l'approximation de l'intégrale par la méthode des trapèzes est :

$$E\le\frac{1}{12n^2}\max_{\forall\ 0\le x\le1}|f''(x)|=\frac{1}{12n^2}\max_{\forall\ 0\le x\le1}\left|2(2x^2-1)e^{-x^2}\right|=\frac{2}{12n^2e}=\frac{1}{6n^2e}$$

si,

$$E\le0.5\times10^{-6}\ \Rightarrow\ \frac{1}{6n^2e}=0.5\times10^{-6}$$

alors,

$$n=\sqrt{\frac{1}{6\times e\times0.5\times10^{-6}}}\approx350$$

(b) Dans le cas de la méthode de Simpson, l'erreur d'intégration :

$$E\le\frac{1}{180n^4}\max_{\forall\ 0\le x\le1}|f^{(4)}(x)|=\frac{1}{180n^4}\max_{\forall\ 0\le x\le1}\left|4(4x^4-12x^2+3)e^{-x^2}\right|=\frac{12}{180n^4}$$

si,

$$E\le0.5\times10^{-6}\ \Rightarrow\ \frac{12}{180n^4}=0.5\times10^{-6}$$

alors,

$$n=\left(\frac{12}{180\times0.5\times10^{-6}}\right)^{1/4}\approx19$$

En conclusion pour la méthode des trapèzes il faut que le nombre des sous intervalles soit plus de 350 et pour la méthode de Simpson il faut que le nombre des sous intervalles soit supérieur à 20.

Exercice 8

Soit *f* une fonction continue satisfaisant sur l'intervalle [0, 1] la condition :

$$f(x)+f(1-x)=1$$

(a) Montrer que l'intégrale

$$I = \int_0^1 f(x)dx = \frac{1}{2}$$

(b) Monter pour une subdivision symétrique de l'intervalle [0, 1], la méthode des trapèzes donne la même valeur de *I*.

(a) l'intégrale

$$\int_0^1 f(x)dx = -\int_1^0 f(1-x)dx = \int_0^1 f(1-x)dx$$

$$\Rightarrow \int_0^1 f(x)dx - \int_0^1 f(1-x)dx = 0 \Rightarrow \int_0^1 [f(x) - f(1-x)]dx =$$

or,

$$f(1-x) = 1 - f(x)$$

alors,

$$\int_0^1 [f(x) - f(1-x)]dx = 0 \Leftrightarrow \int_0^1 [2f(x) - 1]dx = 0 \Leftrightarrow 2\int_0^1 f(x)dx = \int_0^1 dx = 1$$

soit,

$$\int_0^1 f(x)dx = \frac{1}{2}$$

(b) Soit la subdivision symétrique de l'intervalle [0, 1] (Fig. 5.14) :

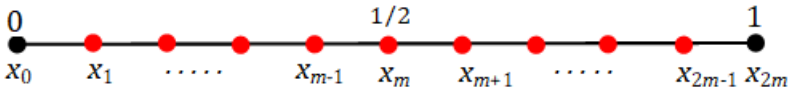


Fig. 5.14 : Subdivision symétrique de l'intervalle [0, 1].

d'après la méthode des trapèzes :

$$I \approx \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{2m-1} f(x_i) + f(x_{2m}) \right] = \frac{1}{4m}\left[f(0) + f(1) + 2\sum_{i=1}^{2m-1} f(x_i) \right] = \frac{1}{4m}\left[1 + 2\sum_{i=1}^{2m-1} f(x_i) \right]$$

d'après la figure 5.14,

$$\begin{aligned} x_1 &= h & x_{2m-1} &= 1-h \\ x_2 &= 2h & x_{2m-2} &= 1-2h \\ &\cdots\cdots & &\cdots\cdots\cdots \\ x_i &= ih & x_{2m-i} &= 1-ih \\ &\cdots\cdots & &\cdots\cdots\cdots \\ x_{m-1} &= (m-1)h & x_{m+1} &= 1-(m-1)h \end{aligned}$$

soit,

$$\sum_{i=1}^{2m-1} f(x_i) = \sum_{i=1}^{m-1} f(x_i) + f(x_m) + \sum_{i=1}^{m-1} f(x_{2m-i}) = f(1/2) + \sum_{i=1}^{m-1} [f(ih) + f(1-ih)]$$

or,

$$f(1/2) + f(1-1/2) = 1 \Rightarrow 2f(1/2) = 1 \Rightarrow f(1/2) = 1/2$$

et,

$$\sum_{i=1}^{m-1} [f(ih) + f(1-ih)] = \sum_{i=1}^{m-1} 1 = m-1$$

alors,

$$\sum_{i=1}^{2m-1} f(x_i) = 1/2 + m - 1 = m - 1/2$$

finalement,

$$I \approx \frac{1}{4m}\left[1+2(m-1/2)\right]=\frac{1}{4m}\left[1+2m-1\right]=\frac{1}{2}$$

16. Exercices supplémentaires

Exercice 1

(a) Déduire la « règle du point médian » de l'intégration, c'est-à-dire :

$$\int_{x_k}^{x_k+h} f(x)dx = hf\left(x_k+\frac{1}{2}h\right)+\frac{1}{24}h^3 f''(\xi),\quad x_k<\xi<x_k+h$$

{*indication*: Utiliser le théorème de Taylor au $x_k+\frac{1}{2}h$ }.

(b) Obtenez la règle du point médian composite pour $\int_a^b f(x)dx$ en subdivisant l'intervalle [*a*, *b*] en *n* nœuds.

Exercice 2

Déterminer c_2, b_1 et b_2 dans la formule de quadrature de Radau

$$\int_0^1 g(t)dt \approx b_1 g(t)+b_2 g(c_2)$$

afin que son ordre soit maximal.

Exercice 3

La formule des trapèzes comme :

$$\int_0^h f(x)dx = af(0)+bf(h)+E(f),\quad 0<h<\pi$$

Est-elle exacte pour $f(x)=\cos x$ et $f(x)=\sin x$.

Exercice 4

Utiliser la formule d'intégration d'Euler-Maclaurin pour démontrer les relations :

$$\cos(0)+\cos\left(\frac{\pi}{100}\right)+\cos\left(\frac{2\pi}{100}\right)+\cdots+\cos(2\pi)=1$$

$$\sin(0)+\sin\left(\frac{\pi}{100}\right)+\sin\left(\frac{2\pi}{100}\right)+\cdots+\sin(2\pi)=0$$

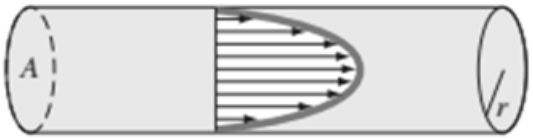
Exercice 5

Utiliser la formule d'intégration d'Euler-Maclaurin pour calculer l'intégrale :

$$\int_1^2 e^{-x^2}dx$$

Exercice 6

Si la distribution de la vitesse *v* à travers d'un écoulement dans une conduite est connue, le débit *Q* (i.e., le volume d'eau passé par unité de temps) est



calculé par l'intégrale $\int_A v dA$. Ou A la surface transversale de la conduite, si elle est circulaire $A = \pi r^2$ $A = \pi r^2$ et $dA = \pi r dr$ alors,

$$Q = 2\pi \int_0^r v r dr$$

Si l'écoulement est laminaire, la vitesse v est exprimée par :

$$v = 2\left(1 - \frac{r}{r_0}\right)^{1/6}$$

r_0 est le rayon total de la conduite.
Calculer par la méthode de Simpson le débit Q de l'écoulement.

Exercice 7

La longueur d'une courbe paramétrée $\{x(t),\ y(t)\}$ est donnée par l'intégrale :

$$\int_a^b \sqrt{[x'(t)]^2 + [y'(t)]^2}\, dt$$

Pour,

$$x(t) = \frac{3}{2}\cos t + \cos 3t \quad \text{et} \quad y(t) = \frac{3}{2}\sin t - \sin 3t, \quad 0 \le t \le 2\pi$$

Estimer la longueur de la courbe avec 10^{-6} de précision

Exercice 8

La fonction gamma est définie par l'intégrale :

$$\Gamma(\alpha) = \int_0^\infty e^{-x} x^{\alpha - 1} dx$$

Calculer approximativement $\Gamma(\alpha)$ pour $\alpha = 1.0,\ 1.4,\ \text{et}\ \ 2.0.$

Exercice 9

Supposons que la force ascendante de la résistance de l'air sur une chute l'objet est proportionnel au carré de la vitesse. Dans ce cas, la vitesse peut être calculée comme :

$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right)$$

avec c_d est le coefficient de traînée de second ordre.

(a) pour $\alpha = 1.0,\ 1.4,\ \text{et}\ 2.0.$ Si g=9.8 m/s², m=68.1 kg et c_d = 0,25 kg/m, utiliser l'intégration analytique pour déterminer à quelle distance l'objet tombe en 10 s.

(b) Faites la même évaluation, mais évaluer l'intégrale avec la règle des trapèzes à segments multiples. Utilisation un n suffisamment élevé pour que vous obteniez trois chiffres significatifs de précision.

Exercice 10

Supposons que nous souhaitions calculer la température de la terre à certaines profondeurs sur une période de temps. On suppose que la terre est plate et initialement à température nulle. La température $T(h,\ t)$ est exprimée par :

$$T(h,\ t)=\frac{h}{2a}\int_0^t \frac{e^{-\frac{h^2}{4a(t-\tau)}}}{(t-\tau)\sqrt{4\pi a(t-\tau)}}T_s(\tau)d\tau$$

$T_s(t)$ est la température à la surface de la terre. Le constant *a* est la diffusivité thermique et est fonction du milieu. Si *t* est en heures,

$$T_s(t)=15+20\sin\left(\frac{2\pi t}{8766}\right)$$

Supposons que $a=0.009\ \text{m}^2/\text{h}$, évaluer $T(h,\ t)$ pour les valeurs de *t* et *h* suivants :

(a) *t* = 200, 400, 500 et *h* = 1 m,
(b) *t* = 200, 400, 500 et *h* = 5 m,
(c) *t* = 200, 400, 500 et *h* = 10 m.

Exercice 11

Dans les problèmes de conduction thermique, le flux de chaleur d'un tuyau est donné par l'intégrale :

$$C(\varphi)=\frac{\sin^2\varphi}{16}\int_{\varphi}^{\pi/2}\sin x\left(\cos\varphi-\frac{\pi-2x-\sin 2x}{2\cos^2 x}\right)^3 dx$$

φ est en radian.

(a) Estimer l'intégrale pour π/5, π/4, π/3 en utilisant la formule de quadrature de Gauss avec *n* = 4. Noton que l'intégrale à une singularité à x = π/2.

(b) Pour lever la singularité, montrer que si on utilise les séries de puissance $C(\varphi)$ est peut être approchée par :

$$C(\varphi)\approx\frac{\sin^2\varphi}{16}\int_{\varphi 6}^{\pi/2}\left(x-\frac{x^3}{6}\right)\left[\cos\varphi-\left(\frac{\pi}{2}-2x+\frac{\pi x^2}{2}-\frac{4x^3}{3}+\frac{\pi x^4}{3}\right)\right]dx$$

Evaluer dans ce cas l'intégrale pour les mêmes valeurs de φ et comparer les résultats avec celles obtenus dans (a).

Exercice 12

Le potentiel de vitesse d'un fluide irrotationnel incompressible satisfait l'équation de Laplace. Supposons que nous puissions considérer le fluide comme étant bidimensionnel. Notons le potentiel pour que la vitesse soit :

$$\mathbf{v}=\nabla u$$

Sortons du mouvement du fluide en donnant une vitesse normale au fluide à la frontière de sorte que la condition aux limites soit donnée par une fonction *g*. Si le conteneur peut être

considéré comme un disque de rayon a dans ce cadre bidimensionnel, alors la solution u en coordonnées polaires est donnée par l'intégrale :

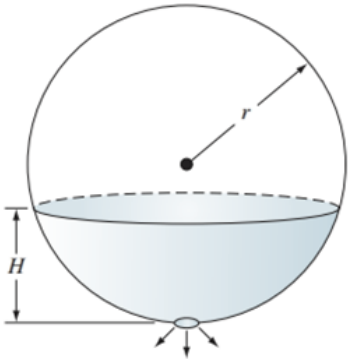
$$u(r,\ \theta)=C-\frac{a}{2\pi}\int_0^{2\pi}\ln\left(a^2-2ar\cos(\phi-\theta)+r^2\right)\ g(\phi)\ d\phi$$

ou C est un constant arbitraire et a = 1.

Evaluer l'intégrale $u(r,\ \theta)$ pour $g(\theta)=\sin\theta$, et C = 0 pour les points $(0.1,\ \pi/4)$, $(0.2,\ \pi/4)$ et $(0.4,\ 3\pi/6)$.

Exercice 13

Un réservoir sphérique de rayon $r=1.5\,\text{m}$ avec une orifice dans sa base permettant la vidange du réservoir. Soit le tableau des données :



t (s)	Q (m³/h)	t (s)	Q (m³/h)
0	10.550	3600	7.020
500	9.576	4300	6.480
1000	9.072	5200	5.688
1500	8.640	6500	4.752
2200	8.100	7000	3.348
2900	7.560	7500	1.404

Ecrire un script MATLAB permettant de calculer le volume d'eau drainé pour un hauteur d'eau H.

Exercice 14

Soi la fonction f, donnée par :

$$f(x)=a_0+\sum_{k=1}^{m}\left(a_k\cos(kx)+b_k\sin(kx)\right),\ a_k,\ b_k\in\mathbb{R}\ \text{et l'intégrale}\ I=\int_0^{2\pi}f(x)dx$$

1. Quelle est la valeur exacte de I ?
2. Appliquer la règle des trapèzes à I avec $h=2\pi/N$. A partir de quelle valeur de N le résultat est-il exact ?
3. Déterminer la formule de quadrature à trois étapes d'ordre 6.
4. Appliquer la formule précédente au calcul de I et répondez à la même question que sous (2).
5. Quelle formule de quadrature proposez-vous pour l'intégration numérique d'une fonction périodique.

Chapitre 6.

Méthodes numériques des EDOs

La résolution numérique des équations différentielles est probablement le domaine de l'analyse numérique où les applications sont les plus nombreuses. Que ce soit en mécanique des fluides, en transfert de chaleur ou en analyse de structures, on aboutit souvent à la résolution d'équations différentielles, de systèmes d'équations différentielles ou plus généralement d'équations aux dérivées partielles. La majorité de ces équations différentielles nécessitent une solution basée souvent sur des concepts numériques appropriées, par exemple l'équation d'un pendule simple de masse m en oscillation (Fig. 6.1) est explicité par l'équation différentielle :

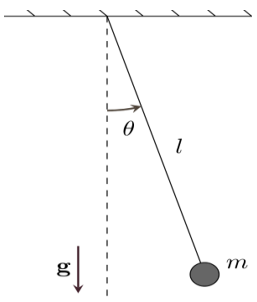


Fig. 6.1 : Pendule simple.

$$\frac{d^2\theta}{dt^2}+\frac{g}{l}\sin\theta=0$$

Il est clair que cette équation n'admet pas une solution analytique simple, ce qui nous ramène à chercher une solution par les méthodes numériques capables d'approcher la solution de l'équation différentielle.

En générale, une équation différentielle ordinaire EDO d'ordre n, c'est une fonction analytique qui peut être explicitée par :

$$F\left(t,\ u,\ \frac{du}{dt},\ \frac{d^2u}{dt^2},\ \ldots,\ \frac{d^nu}{dt^n}\right)=0$$

sa résolution consiste à déterminer la fonction $u(t)$ sous forme d'une expression analytique ou sous forme d'un graphe (Fig. 6.2).

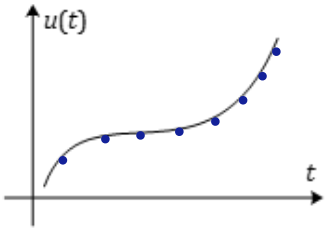


Fig. 6.2 : $u(t)$.

Certaines équations différentielles possèdent des règles et des méthodes pour les résoudre, tel que les équations linéaires, de Riccati, de Bernoulli etc., d'autres types comme celle de Bessel, Hermite et … la solution est exprimée sous forme d'une série de fonction polynomiale. Mais à part ces équations différentielles particulières, la détermination d'une solution analytique parait impossible, alors les méthodes numériques sont les seuls issus pour les résoudre (Fig. 6.2).

1. Méthodes d'Euler

En mathématiques, la méthode d'Euler, nommée ainsi en l'honneur du mathématicien Leonhard Euler (1707–1783) (Fig. 6.3), est une procédure numérique pour résoudre par approximation des équations différentielles du premier ordre avec une condition initiale nommée aussi le problème de Cauchy (Problème à valeur Initiale IVP).

$$\frac{du}{dt} = f(t,\ u),\ \ u(t_0) = u_0$$

c'est-à-dire,

$$du = f(t,\ u)dt$$

L'intégration entre deux points d'abscisses t_n et t_{n+1} (Fig. 6.4) conduit à :

$$\int_{u_n}^{u_{n+1}} du = \int_{t_n}^{t_{n+1}} f(t,\ u)dt \ \ \Rightarrow\ \ u_{n+1} - u_n = \int_{t_n}^{t_{n+1}} f(t,\ u)dt = hf(t_n,\ u_n) + R(h)$$

$R(h)$ est l'erreur de troncature ou le reste qui tends vers 0 quand t_n tends vers t_{n+1} (c'est-à-dire $h \to 0$). Cette intégrale est approchée par $hf(t_n,\ u_n)$. La méthode d'Euler explicite :

$$u_{n+1} = u_n + hf(t_n,\ u_n)$$

Connaissant la condition initiale $u(t_0) = u_0$, le processus itératif suivant peut être généré facilement :

$$u_{k+1} = u_k + hf(t_k,\ u_k),\ \ k = 0,\ 1,\ 2,\ \ldots,\ N-1$$

D'une autre façon, l'intégrale (Fig. 6.5),

$$\int_{t_n}^{t_{n+1}} f(t,\ u)dt \approx hf(t_{n+1},\ u_{n+1})$$

alors,

$$u_{n+1} = u_n + hf(t_{n+1},\ u_{n+1})$$

C'est la méthode d'Euler implicite.

Fig. 6.3 : Portrait de L. Euler par Johann Georg Brucker (1756).

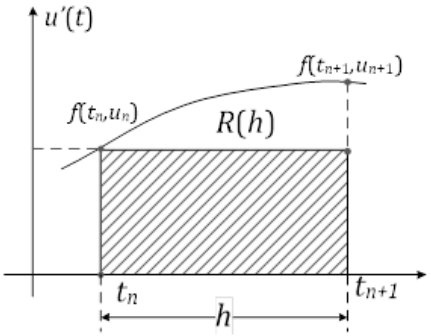


Fig. 6.4 : Méthode d'Euler explicite.

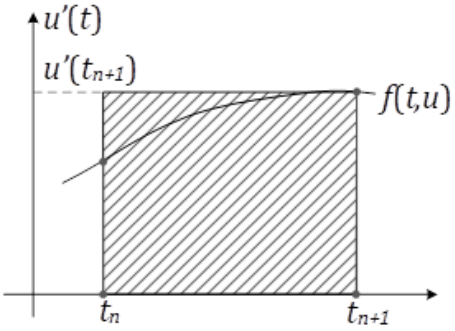


Fig. 6.5 : Méthode d'Euler implicite.

1.1. Méthode de Heun ou de Cranck-Nicholson

La somme membre à membre des deux méthodes d'Euler explicite et implicite conduit à la méthode de Heun ou de Crank-Nicholson, c'est-à-dire,

$$2u_{n+1} = 2u_n + hf(t_n,\ u_n) + hf(t_{n+1},\ u_{n+1}) \ \ \Rightarrow\ \ u_{n+1} = u_n + \frac{h}{2}f(t_n,\ u_n) + \frac{h}{2}f(t_{n+1},\ u_{n+1})$$

or,

$$u_{n+1} = u_n + hf(t_n,\ u_n)$$

soit,

$$u_{n+1} = u_n + \frac{h}{2}f(t_n,\ u_n) + \frac{h}{2}f(t_n + h,\ u_n + hf(t_n,\ u_n))$$

1.2. Erreur de troncature et stabilité

L'erreur de troncature dans la résolution des équations différentielles suivant la méthode d'Euler est de deux types *local* et *global*. L'*erreur de troncature local* est l'erreur commis dans une étape de t_n à t_{n+1}. Le développement limité de Taylor d'ordre deux de la fonction $u(t)$ au voisinage de t_{n+1} est exprimé par :

$$u(t_{n+1}) = u(t_n) + hu'(t_n) + \frac{h^2}{2}u''(\zeta_n),\ \ t_n < \zeta_n < t_{n+1}$$

ou,

$$u(t_{n+1}) = u(t_n) + hf(t_n,\ u(t_n)) + \frac{h^2}{2}u''(\zeta_n)$$

L'erreur de troncature local est de :

$$e_{n+1} = u(t_{n+1}) - u_{n+1} = \frac{h^2}{2}u''(\zeta_n)$$

C'est-à-dire que l'erreur de troncature locale est de deuxième ordre $O(h^2)$, soit :

$$e_{n+1} = u(t_n) + hf(t_n,\ u(t_n)) + \frac{h^2}{2}u''(\zeta_n) - u_{n+1} = u(t_n) + hf(t_n,\ u(t_n)) + \frac{h^2}{2}u''(\zeta_n) - u_n - hf(t_n,\ u_n)$$

$$= u(t_n) - u_n + h\left[f(t_n,\ u(t_n)) - hf(t_n,\ u_n)\right] + \frac{h^2}{2}u''(\zeta_n) = e_n + h\frac{f(t_n,\ u(t_n)) - hf(t_n,\ u_n)}{u(t_n) - u_n}e_n + \frac{h^2}{2}u''(\zeta_n)$$

Supposons que $f(t,\ u)$ et $\partial f/\partial u$ sont continues, suivant le théorème de la valeur moyenne il existe ξ_n de $[u_n,\ u_{n+1}]$ tel que :

$$\frac{f(t_n,\ u(t_n)) - hf(t_n,\ u_n)}{u(t_n) - u_n} = \frac{\partial}{\partial u}f(t_n,\ \xi_n)$$

alors,

$$e_{n+1} = e_n + h\frac{\partial}{\partial u}f(t_n,\ \xi_n)e_n + \frac{h^2}{2}u''(\zeta_n)$$

donc,

$$|e_{n+1}| \le \left(1 + h\left|\frac{\partial f}{\partial u}f(t_n,\ \xi_n)\right|\right)|e_n| + \frac{h^2}{2}|u''(\zeta_n)|$$

Pour $u_n < u < u_{n+1}$ et $t_n < t < t_{n+1}$, $\left|\frac{\partial}{\partial u}f(t,\ u)\right| \le M$ et $|u''(t)| \le K$

Avec *M* et *K* sont positives. Alors,

$$|e_{n+1}| \le (1 + hM)|e_n| + \frac{h^2}{2}K$$

qui s'écrit aussi,

$$|e_{n+1}| \le (1+hM)\left[(1+hM)|e_{n-1}| + \frac{h^2}{2}K\right] + \frac{h^2}{2}K = (1+hM)^2|e_{n-1}| + \frac{h^2}{2}K\left[(1+hM)+1\right]$$

de même,

$$|e_{n+1}| \le (1+hM)^2\left[(1+hM)|e_{n-2}| + \frac{h^2}{2}K\right] + \frac{h^2}{2}K = (1+hM)^3|e_{n-2}| + \frac{h^2}{2}K\left[(1+hM)^2 + (1+hM) + 1\right]$$

$$|e_{n+1}| \le (1+hM)^4 |e_{n-3}| + \frac{h^2}{2}K\left[(1+hM)^3 + (1+hM)^2 + (1+hM) + 1\right]$$

. . . . . .

$$|e_{n+1}| \le (1+hM)^{n+1} |e_0| + \frac{h^2}{2}K\left[(1+hM)^n + \cdots + (1+hM)^3 + (1+hM)^2 + (1+hM) + 1\right]$$

or,

$$|e_0| = 0 \text{ et } (1+hM)^n + \cdots + (1+hM)^3 + (1+hM)^2 + (1+hM) + 1 = \frac{(1+hM)^{n+1} - 1}{(1+hM) - 1} = \frac{(1+hM)^{n+1} - 1}{hM}$$

soit,

$$|e_{n+1}| \le \frac{hK}{2M}\left[(1+hM)^{n+1} - 1\right]$$

ou,

$$|e_n| \le \frac{hK}{2M}\left[(1+hM)^n - 1\right], \quad n = 0,\ 1,\ 2,\ \ldots$$

En analyse mathématique, pour un variable *x* positive, $1 + x \le e^x$, alors :

$$1 + hM \le e^{hM} \ \Rightarrow\ (1+hM)^n \le e^{nhM} = e^{M(t_n - t_0)}$$

soit,

$$|e_n| \le \frac{hK}{2M}\left[e^{M(t_n - t_0)} - 1\right], \quad n = 0,\ 1,\ 2,\ \ldots$$

C'est l'*erreur globale* pour la méthode d'Euler, elle est de première ordre $O(h)$.

Pour étudier la stabilité d'un problème de type de Cauchy, soit ε une perturbation, et le problème :

$$\frac{dz}{dt} = f(t,\ z),\ \ z(t_0) = z_0 = u_0 + \varepsilon \quad t_0 \le t \le b$$

Le problème de Cauchy est dit stable si et seulement si :

$$\|u - z\|_\infty = \max_{t_0 \le t \le b} |u(t) - z(t)| \le c\varepsilon,\ c \text{ constant positif}$$

Pour la méthode d'Euler,

$$u_{n+1} = u_n + hf(t_n,\ u_n),\ \ u(t_0) = u_0 \text{ et } z_{n+1} = z_n + hf(t_n,\ z_n),\ \ z(t_0) = u_0 + \varepsilon$$

L'erreur due à la perturbation est :

$$e_n = z_n - u_n,\ \Rightarrow\ e_0 = \varepsilon$$

Pour la méthode d'Euler,

$$e_{n+1} = e_n + h\left[f(t_n,\ z_n) - f(t_n,\ u_n)\right]$$

alors,

$$|e_{n+1}| \le |e_n| + h\left|f(t_n,\ z_n) - f(t_n,\ u_n)\right|$$

Si la fonction $f(t,\ u)$ est Lipschitzienne (Weisstein, 2003), il existe alors, un *K* de $\mathbb{R}^+$ tel que,

$$\left|f(t_n,\ z_n) - f(t_n,\ u_n)\right| \le K|z_n - u_n|$$

soit,

$$|e_{n+1}| \le |e_n| + hK|e_n| = (1+hK)|e_n|$$

alors,

$$|e_{n+1}| \le (1+hK)|e_n| \le (1+hK)^2 |e_{n-1}| \le \cdots \le (1+hK)^{n+1} |e_0| = (1+hK)^{n+1} |\varepsilon|$$

puisque,

$$1+hK \le e^{hK}$$

alors,

$$|e_n| \le e^{nhK}|\varepsilon| = e^{(b-t_0)K}|\varepsilon|$$

ce qui traduit suivant la propriété citée au-dessus que la méthode d'Euler est stable.

Suivant ce concept, n'importe quel problème de Cauchy est stable suivant cette vision qui concerne la perturbation dans la valeur initiale u_0 (Atkinson *et al.*, 2009), celle-ci qui est une vision partiale du problème de stabilité. Si en plus l'erreur $\|u-z\|_\infty$ est faction de la perturbation ($0<c<1$), le problème de Cauchy est *bien conditionné*, si par contre l'erreur $\|u-z\|_\infty$ est très large ($c>>1$), le problème de Cauchy est *mal conditionné*. Dans la pratique, il existe un continuum de problèmes allant du bien conditionné vers du mal conditionné. l'extension du mal conditionnement à un effet sur la sensibilité de la solution numérique indépendamment de la méthode numérique utilisée (Atkinson *et al.*, 2009). Le problème de Cauchy à l'égard de cette perturbation :

$$u(t)-z(t) \approx -\varepsilon \exp\left(\int_{t_0}^{t} g(s)ds\right)$$

avec,

$$g(t) = \left.\frac{\partial f(t,\ u)}{\partial u}\right|_{u=u(t)}$$

Par conséquent, le problème de Cauchy est bien conditionné si est seulement si :

$$\frac{\partial f(t,\ u(t))}{\partial u} \le 0,\ t_0 \le t \le b$$

2. Méthodes de Taylor

Le développement de Taylor autorise une généralisation immédiate de la méthode d'Euler qui permet d'obtenir des algorithmes dont l'erreur de troncature locale est d'ordre plus élevé. On cherche, à $t = t_n$, une approximation de la solution en $t = t_{n+1}$. On a immédiatement :

$$u(t_{n+1}) = u(t_n+h) = u(t_n) + hu'(t_n) + \frac{h^2}{2}u''(t_n) + O(h^3)$$

c'est-à dire,

$$u(t_{n+1}) = u(t_n) + hf(t_n,\ u(t_n)) + \frac{h^2}{2}f'(t_n,\ u(t_n)) + O(h^3)$$

or,

$$f'(t,\ u(t)) = \frac{\partial f(t,\ u(t))}{\partial t} + \frac{\partial f(t,\ u(t))}{\partial u}u'(t_n) = \frac{\partial f(t,\ u(t))}{\partial t} + \frac{\partial f(t,\ u(t))}{\partial u}f(t,\ u(t))$$

alors,

$$u(t_{n+1}) = u(t_n) + hf(t_n,\ u(t_n)) + \frac{h^2}{2}\left(\frac{\partial f(t_n,\ u(t_n))}{\partial t} + \frac{\partial f(t_n,\ u(t_n))}{\partial u}f(t_n,\ u(t_n))\right) + O(h^3)$$

Soit le schéma,

$$u_{n+1} = u_n + hf(t_n,\ u_n) + \frac{h^2}{2}\left(\frac{\partial f(t_n,\ u_n)}{\partial t} + \frac{\partial f(t_n,\ u_n)}{\partial u}f(t_n,\ u_n)\right)$$

C'est le schéma de la méthode de Taylor d'ordre 2.

Soit l'exemple d'un problème de Cauchy,

$$\begin{cases} \dfrac{du}{dt} = u + 2t - t^2 \quad \forall\ 0 \le t \le 2, \\ u(0) = 1. \end{cases}$$

dont la solution exacte est exprimée par :

$$u(t) = t^2 + e^t$$

Dans ce problème $f(t,\ u) = u + 2t - t^2$, l'application du schéma de Taylor d'ordre 2 conduit à :

$$u_{n+1} = \left(1 + h + \frac{h^2}{2}\right)u_n + 2ht_n - h\left(1 + \frac{h}{2}\right)t_n^2 + h^2$$

Pour résoudre ce problème avec les méthodes d'Euler explicite et de Taylor d'ordre 2, soit le script MATLAB :

```
f=@(t,u)u+2*t-t.^2; % Définition de la fonction f(t,u)
% La solution exacte
Uexact=@(t) t.^2+exp(t);
h=0.3; t=0:h:2;
% Définition de la condition initiale
uE(1)=1;uT(1)=1; uEx(1)= Uexact(t(1));
% uE : Calcul avec Euler Explicite
% uT : Calcul avec Taylor d'ordre 2
% uEx: Valeur exacte
for n=1:numel(t)-1
 uE(n+1)=uE(n)+h*f(t(n),uE(n));
 uT(n+1)=uT(n)+h*f(t(n),uT(n))+(h^2/2)*(2-2*t(n)+f(t(n),uT(n)));
 uEx(n+1)= Uexact(t(n+1));
end
```

t	uEx	uE	uT
0	1	1	1
0.3	1.4399	1.3000	1.4350
0.6	2.1821	1.8430	2.1690
0.9	3.2696	2.6479	3.2431
1.2	4.7601	3.7393	4.7126
1.5	6.7317	5.1491	6.6516
1.8	9.2896	6.9188	9.1602

3. Méthodes de Runge-Kutta explicites

Les méthodes de Runge-Kutta (Kutta, 1901 et Butcher, 1996) sont des méthodes d'analyse numérique d'approximation de solution d'équations différentielles. Elles ont été nommées ainsi en l'honneur des mathématiciens Carl Runge (Fig. 6.6) et Martin Wilhelm Kutta (Fig. 6.7) lesquels élaborèrent la méthode en 1901.

Ces méthodes reposent sur le principe de l'itération, c'est-à-dire qu'une première estimation de la solution est utilisée pour calculer une seconde estimation, plus précise, et ainsi de suite . Soit le problème de Cauchy :

$$\frac{du}{dt} = f(t,\ u),\ u(t_0) = u_0$$

que l'on cherche à résoudre en un ensemble discret $t_0 < t_1 < t_2 < \cdots < t_N$. Soit l'intervalle $[t_n,\ t_{n+1}]$, l'équation précédente s'écrit aussi :

$$du = f(t,\ u)dt \ \Rightarrow\ u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(t,\ u)dt$$

Fig. 6.6 : C. Runge (1856-1927).

Fig. 6.7 : M. W. Kutta (1867-1944).

D'après la méthode de quadrature, l'intégrale $\int_{t_n}^{t_{n+1}} f(t,\ u)dt$ est peut-être approchée par :

$$\int_{t_n}^{t_{n+1}} f(t,\ u)dt \approx \sum_{i=1}^{s} b_i k_i \quad \text{avec} \quad \sum_{i=1}^{s} b_i = 1$$

alors, la méthode de Runge-Kutta d'ordre *s* est explicitée comme suit :

$$u_{n+1} = u_n + \sum_{i=1}^{s} b_i k_i$$

$$k_1 = hf(t_n,\ u_n) \quad \text{et} \quad k_p = hf\left(t_n + c_p h,\ u_n + \sum_{q=1}^{p-1} a_{pq} k_q\right),\ \forall p = 2\ ,3\ ,\ldots,\ s$$

La matrice triangulaire inférieure $\mathbf{A}=[a_{ij}],\ 1 \le j \le i \le s$ s'appelle matrice de Runge-Kutta $(b_i,\ i=1,\ 2,\ \ldots,\ s)$ les coefficients poids et $(c_j,\ j=2,\ 3,\ \ldots,\ s)$ les nœuds. Ces coefficients sont rangés dans un tableau (Fig. 6.8) dit de Butcher (1963, 2016). La méthode de Runge-Kutta est consistante si :

$$\sum_{j=1}^{i-1} a_{ij} = c_i \ \text{pour} \ \ i = 2,\ 3,\ \ldots,\ s$$

$$\begin{array}{c|cccc} 0 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ \vdots & \vdots & & \ddots & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} \\ \hline & b_1 & b_2 & \cdots & b_{s-1} \quad b_s \end{array}$$

Fig. 6.8 : Tableau de Butcher.

On déduit que la méthode de Runge-Kutta d'ordre 1 (RK1) ou à un pas est la méthode d'Euler explicite, c'est-à-dire :

$$u_{n+1} = u_n + hf(t_n,\ u_n)$$

3.1. Méthodes de Runge-Kutta d'ordre 2 (RK2)

Le développement de Taylor à permet de montrer que la formule d'Euler ou Runge-Kutta d'ordre 1 est tronqué à l'erreur de troncature locale en fonction de h^2. Sur ce principe, et pour arriver à la méthode de Runge-Kutta de 2ième ordre, supposons que cette intégrale est donnée par l'expression générale suivante :

$$\int_{t_n}^{t_{n+1}} f(t,\ u)dt = ahf(t_n,\ u_n) + bhf(t_n + \alpha h,\ u_n + \beta hf(t_n,\ u_n)) + O(h^3)$$

a et b sont les coefficients poids, α et β déterminent la forme de la pente. Soit donc le processus :

$$u_{n+1} = u_n + ahf(t_n,\ u_n) + bhf(t_n + \alpha h,\ u_n + \beta hf(t_n,\ u_n)) + O(h^3)$$

Le développement de Taylor de la fonction $f(t_n + \alpha h, u_n + \beta hf(t_n, u_n))$ conduit à l'approximation :

$$f(t_n + \alpha h,\ u_n + \beta hf(t_n,\ u_n)) \approx f(t_n,\ u_n) + \alpha h \left.\frac{\partial f}{\partial t}\right|_{\substack{t=t_n \\ u=u_n}} + \beta hf(t_n,\ u_n) \left.\frac{\partial f}{\partial u}\right|_{\substack{t=t_n \\ u=u_n}}$$

Par conséquent :

$$u_{n+1} = u_n + (a+b)hf(t_n,\ u_n) + b\alpha h^2 \left.\frac{\partial f}{\partial t}\right|_{\substack{t=t_n \\ u=u_n}} + b\beta h^2 f(t_n,\ u_n) \left.\frac{\partial f}{\partial u}\right|_{\substack{t=t_n \\ u=u_n}} + O(h^3) \qquad (6.1)$$

Aussi, le développement de Taylor de la fonction $u(t_{n+1})$ au voisinage de t_n est exprimé par :

$$u(t_{n+1}) = u(t_n) + u'(t_n)h + \frac{u''(t_n)}{2!}h^2 + O(h^3)$$

or,

$$u''(t) = \frac{d}{dt}\left(\frac{du}{dt}\right) = \frac{df(t,\ u)}{dt} = \frac{\partial f(t,\ u)}{\partial t} + \frac{\partial f(t,\ u)}{\partial u}\frac{du}{dt} = \frac{\partial f(t,\ u)}{\partial t} + \frac{\partial f(t,\ u)}{\partial u} f(t,\ u)$$

Pour $t = t_n$ et $u = u_n$,

$$u'(t_n) = f(t_n,\ u_n),\ u''(t_n) = \left.\frac{\partial f}{\partial t}\right|_{\substack{t=t_n \\ u=u_n}} + f(t_n,\ u_n) \left.\frac{\partial f}{\partial u}\right|_{\substack{t=t_n \\ u=u_n}}$$

alors,

$$u_{n+1} = u_n + hf(t_n,\ u_n) + \frac{h^2}{2}\left.\frac{\partial f}{\partial t}\right|_{\substack{t=t_n \\ u=u_n}} + \frac{h^2}{2} f(t_n,\ u_n) \left.\frac{\partial f}{\partial u}\right|_{\substack{t=t_n \\ u=u_n}} + O(h^3) \qquad (6.2)$$

L'identification entre les deux expressions analytiques (6.1) et (6.2) conduit à :

$$a + b = 1, \quad b\alpha = \frac{1}{2} \quad \text{et} \quad b\beta = \frac{1}{2}$$

Par conséquent l'expression générale des méthodes de Runge-Kutta d'ordre 2 RK2 est :

$$u_{n+1} \approx u_n + h\left[(1-b) f(t_n,\ u_n) + bf\left(t_n + \frac{h}{2b},\ u_n + \frac{h}{2b} f(t_n,\ u_n)\right)\right]$$

Si $b = 1/2$, on trouve l'expression de u_{n+1} obtenu suivant la somme des deux formules explicite et implicite d'Euler dite aussi la méthode de Heun (ou d'Euler-Cauchy) :

$$u_{n+1} = u_n + \frac{h}{2} f(t_n,\ u_n) + \frac{h}{2} f(t_n + h,\ u_n + hf(t_n, u_n))$$

Si $b = 1$, on obtient la méthode de Runge-Kutta (proprement dite) :

$$u_{n+1} = u_n + hf\left(t_n + \frac{1}{2}h,\ u_n + \frac{1}{2}hf(t_n,\ u_n)\right)$$

Si $b = 2/3$, on obtient la méthode de Ralston :

$$u_{n+1} = u_n + \frac{1}{3}hf(t_n,\ u_n) + \frac{2}{3}hf\left(t_n + \frac{3}{4}h,\ u_n + \frac{3}{4}hf(t_n, u_n)\right)$$

Le script M-file suivant utilise la méthode de Heun du problème précédent :

```
% Définition de la fonction f(t,u)
f=@(t,u) u+2*t-t^2; h=0.5;  t=0:h:2;
% Définition de la condition initiale
u(1)=1;
for n=1:numel(t)-1
 k1=h*f(t(n),u(n)); k2=h*f(t(n)+h,u(n)+k1);
 u(n+1)=u(n)+(1/2)*(k1+k2);
end
```

t_i	$u(t_i)$	u_i
0.0	1	1
0.5	1.899	1.813
1.0	3.718	3.477
1.5	6.732	6.212
2.0	11.389	10.376

Les résultats obtenus avec RK2 sont meilleurs que celle obtenus par la méthode d'Euler.

3.2. Méthodes de Runge-Kutta d'ordres supérieurs

L'expression de la formule des méthodes de Runge-Kutta d'ordre 2 s'écrit :

$$k_1 = hf\left(t_n,\ u_n\right), \quad k_2 = hf\left(t_n + \alpha h,\ u_n + \beta k_1\right)$$

$$u_{n+1} = u_n + \left(1-b\right)k_1 + bk_2$$

L'idée d'utilisation de la matrice de Runge-Kutta et les coefficients poids appropriés dans le tableau de Butcher toute en respectant le critère de consistance, permet d'avoir un ensemble de méthodes de type Runge-Kutta de différents ordres. Pour que les méthodes de Runge-Kutta d'ordre 2 soient consistantes il faut que $1/2 \le b \le 1$.

Sur ce principe plusieurs méthodes de Runge-Kutta d'ordres supérieurs avec un pas *h* fixe peuvent être développées. Par exemple les méthodes de Runge-Kutta d'ordre 3 sont explicitées ainsi :

$$u_{n+1} = u_n + h\left(ak_1 + bk_2 + ck_3\right)$$

avec,

$$k_1 = f\left(t_n,\ u_n\right),\ \ k_2 = f\left(t_n + ph,\ u_n + phk_1\right) \ \text{ et } \ k_3 = f\left(t_n + rh,\ u_n + shk_1 + \left(r-s\right)hk_2\right)$$

Les développements limités de Taylor des fonctions k_2 et k_3 est :

$$k_2 = k_1 + ph\left.\frac{\partial f}{\partial t}\right|_{\substack{t=t_n\\u=u_n}} + phk_1\left.\frac{\partial f}{\partial u}\right|_{\substack{t=t_n\\u=u_n}} + \left(ph\right)^2\left[\frac{1}{2}\left.\frac{\partial^2 f}{\partial t^2}\right|_{\substack{t=t_n\\u=u_n}} + k_1\left.\frac{\partial^2 f}{\partial t\partial u}\right|_{\substack{t=t_n\\u=u_n}} + \frac{k_1^2}{2}\left.\frac{\partial^2 f}{\partial u^2}\right|_{\substack{t=t_n\\u=u_n}}\right]$$

$$k_3 = k_1 rh\left[\left.\frac{\partial f}{\partial t}\right|_{\substack{t=t_n\\u=u_n}} + k_1\left.\frac{\partial f}{\partial u}\right|_{\substack{t=t_n\\u=u_n}}\right] + h^2 p\left(r-s\right)\left(\left.\frac{\partial f}{\partial t}\right|_{\substack{t=t_n\\u=u_n}} + k_1\left.\frac{\partial f}{\partial u}\right|_{\substack{t=t_n\\u=u_n}}\right)\left.\frac{\partial f}{\partial u}\right|_{\substack{t=t_n\\u=u_n}} +$$

$$\left(rh\right)^2\left[\frac{1}{2}\left.\frac{\partial^2 f}{\partial t^2}\right|_{\substack{t=t_n\\u=u_n}} + k_1\left.\frac{\partial^2 f}{\partial t\partial u}\right|_{\substack{t=t_n\\u=u_n}} + \frac{k_1^2}{2}\left.\frac{\partial^2 f}{\partial u^2}\right|_{\substack{t=t_n\\u=u_n}}\right]$$

or, le développement de Taylor de la fonction $u\left(t_{n+1}\right)$ au voisinage de t_n est exprimé aussi par :

$$u\left(t_{n+1}\right) = u\left(t_n\right) + u'\left(t_n\right)h + \frac{u''\left(t_n\right)}{2}h^2 + \frac{u'''\left(t_n\right)}{6}h^3 + O\left(h^4\right)$$

avec,

$$u'\left(t_n\right) = f\left(t_n,\ u_n\right),\ \ u''\left(t_n\right) = \left.\frac{\partial f}{\partial t}\right|_{\substack{t=t_n\\u=u_n}} + \left.\frac{\partial f}{\partial u}\right|_{\substack{t=t_n\\u=u_n}} f\left(t_n,\ u_n\right)$$

et,

$$u'''(t_n)=\left.\frac{\partial^2 f}{\partial t^2}\right|_{\substack{t=t_n\\u=u_n}}+2f(t_n,\,u_n)\left.\frac{\partial^2 f}{\partial t\partial u}\right|_{\substack{t=t_n\\u=u_n}}+\left(f(t_n,\,u_n)\right)^2\left.\frac{\partial^2 f}{\partial u^2}\right|_{\substack{t=t_n\\u=u_n}}+\left.\frac{\partial f}{\partial u}\right|_{\substack{t=t_n\\u=u_n}}\left(\left.\frac{\partial f}{\partial t}\right|_{\substack{t=t_n\\u=u_n}}+f(t_n,\,u_n)\left.\frac{\partial f}{\partial u}\right|_{\substack{t=t_n\\u=u_n}}\right)$$

Identifiant la formule de $u(t_{n+1})$ avec celle de u_{n+1} en termes de h et h^2, on obtient :

$$a+b+c=1,\quad bp+cr=\frac{1}{2},\quad bp^2+cr^2=\frac{1}{3}\ \text{ et }\ cp(r-s)=\frac{1}{6}$$

Deux constantes parmi a, b, c, p, r et s sont choisis arbitrairement permet d'avoir une des méthodes de Runge-Kutta d'ordre 3 RK3, par exemple, si $a=c=1/6$, alors $b=4/6,\ p=1/2,\ r=1$ et $s=-1$, soit la méthode classique de Runge-Kutta d'ordre 3 :

$$k_1=f(t_n,\,u_n),\ k_2=f\left(t_n+\frac{h}{2},\,u_n+\frac{h}{2}k_1\right)\ \text{ et }\ k_3=f(t_n+h,\,u_n-hk_1+2hk_2)$$

$$u_{n+1}=u_n+\frac{h}{6}(k_1+4k_2+k_3)$$

Si $b=c=3/8\ \Rightarrow a=2/8,\ p=2/3,\ r=2/3$ et $s=0$, soit la méthode RK3 dite méthode de Nyström :

$$k_1=hf(t_n,\,u_n),\ k_2=hf\left(t_n+\frac{2}{3}h,\,u_n+\frac{2}{3}k_1\right)\text{ et }k_3=hf\left(t_n+\frac{2}{3}h,\,u_n+\frac{2}{3}k_2\right)$$

$$u_{n+1}=u_n+\frac{1}{8}(2k_1+3k_2+3k_3)$$

De la même manière les méthodes de Runge-Kutta d'ordres supérieurs peuvent être construit. La méthode de Runge-Kutta d'ordre 4 la plus utilisée est :

$$u_{n+1}=u_n+\frac{h}{6}(k_1+2k_2+2k_3+k_4)$$

avec,

$$k_1=f(t_n,\,u_n),\ k_2=f\left(t_n+\frac{h}{2},\,u_n+\frac{h}{2}k_1\right),\ k_3=f\left(t_n+\frac{h}{2},\,u_n+\frac{h}{2}k_2\right)\ \text{ et }\ k_4=f(t_n+h,\,u_n+hk_3)$$

Suivant cette formule, soit la fonction **RK4** qui utilise la méthode de Runge-Kutta d'ordre 4 pour résoudre le problème de Cauchy.

```
function [t,u]=RK4(f,u0,h,a,b,varargin)
% f: définition de la fonction f(t,u)
% u0=u(a): Condition initiale, h: le pas
% a, b: bornes de l'intervalle
t=a:h:b; u(1)=u0;
for n=1:numel(t)-1
 k1=f(t(n),u(n)); k2=f(t(n)+(h/2),u(n)+h*(k1/2));
 k3=f(t(n)+(h/2),u(n)+h*(k2/2)); k4=f(t(n+1),u(n)+h*k3);
 u(n+1)=u(n)+(h/6)*(k1+2*k2+2*k3+k4);
end
 plot(t,u,'-',t,u,'or')
end
```

Pour le même problème,

```
[t,u]=RK4(@(t,u)u+2*t-t^2,1,0.5,0,2)
```

qui donne,

```
t =
         0   0.5000   1.0000   1.5000   2.0000
u =
    1.0000   1.8978   3.7156   6.7259   11.3776
```

t_i	$u(t_i)$	u_i
0.0	1	1
0.5	1.899	1.898
1.0	3.718	3.716
1.5	6.732	6.726
2.0	11.389	11.378

Il est clair que les résultats obtenus avec RK4 sont mieux à celle obtenus par RK2.

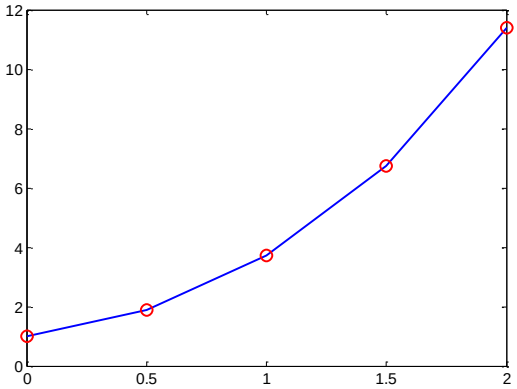


Fig. 6.9 : Exécution de la fonction **RK4**.

Il y a des méthodes modifiées de type Runge-Kutta, l'exemple suivant est la méthode de Runge-Kutta modifiée d'ordre 4 dite méthode de Runge-Kutta-Gill (Gill et Wilkes, 1951) dont l'algorithme est :

$$k_1 = f(t_n,\ u_n),\quad k_2 = f\left(t_n + \frac{h}{2},\ u_n + \frac{h}{2}k_1\right),\quad k_3 = f\left(t_n + \frac{h}{2},\ u_n + \frac{\sqrt{2}-1}{2}hk_1 + \frac{2-\sqrt{2}}{2}hk_2\right)$$

$$k_4 = f\left(t_{n+1},\ u_n + \frac{1-\sqrt{2}}{2}hk_2 + \frac{2+\sqrt{2}}{2}hk_3\right)$$

$$u_{n+1} = u_n + \frac{h}{6}\left(k_1 + \left(2-\sqrt{2}\right)k_2 + \left(2+\sqrt{2}\right)k_3 + k_4\right)$$

3.3. Méthodes de Runge-Kutta à pas adaptatif

Pour améliorer l'efficacité du calcul, on utilise des méthodes à pas variable, c'est-à-dire des méthodes dans lesquelles le pas varie à chaque itération par l'emploi de deux méthodes de Runge-Kutta emboîtées. La première méthode dont l'ordre de troncature locale p sert à calculer la solution approchée u_{n+1}, tandis que la seconde méthode dont l'ordre de l'erreur de troncature locale $q = p-1$ mais ayant le même nombre d'étapes, sert à estimer l'erreur de consistance $\tilde{u}_{n+1}$ pour contrôler le pas. On dit que la méthode est de type RKqp (Jedrzejewski, 2005). L'erreur entre les deux méthodes est explicitée à chaque étape par :

$$E_{n+1} = \left|u_{n+1} - \tilde{u}_{n+1}\right|$$

Dans la pratique, si ε désigne une tolérance imposée, l'algorithme des méthodes de Runge-Kutta à pas adaptif est utilisé avec un nouveau pas $h = h/2$, quand cette l'erreur est strictement supérieure à la tolérance imposée $\left(\text{i.e. } E_{n+1} > \varepsilon\right)$ et avec un pas h = 2h quand l'erreur est inférieure à une fraction de la tolérance $\left(\text{i.e. } E_{n+1} \le \varepsilon/2^p\right)$ et conserve le même pas h quand $\varepsilon/2^p < E_{n+1} \le \varepsilon$.

L'ajustement du pas pour une tolérance imposée ε est exprimé en fonction du pas h et de l'erreur E_{n+1} (Hull *et al.*, 1972) par :

$$h_{adj} = \alpha \left(\frac{\varepsilon \cdot h}{E_{n+1}} \right)^{\frac{1}{q}} h$$

α est un facteur d'ajustement inférieur à 1 (généralement $\alpha \approx 0.9$) . Si dans cette méthode il y a deux pas extrêmes sont imposés $h_{\min}$ et $h_{\max}$ à cause des considérations liés à la stabilité, la nature du problème etc., il faut que $h_{\min} \le h_{adj} \le h_{\max}$. Si par contre $h_{adj} < h_{\min}$, h_{adj} est choisi égal à $h_{\min}$ et si $h_{adj} > h_{\max}$, h_{adj} est choisi égal $h_{\max}$.

3.3.1. Méthode de Merson

La méthode de Merson (1957) est la première méthode de Runge-Kutta adaptive. Elle consiste à calculer à chaque pas h, u_{n+1} suivant l'algorithme :

$$k_1 = hf(t_n,\ u_n), \quad k_2 = hf\left(t_n + \frac{1}{3}h,\ u_n + \frac{1}{3}k_1\right), \quad k_3 = hf\left(t_n + \frac{1}{3}h,\ u_n + \frac{1}{6}k_1 + \frac{1}{6}k_2\right),$$

$$k_4 = hf\left(t_n + \frac{1}{2}h,\ u_n + \frac{1}{8}k_1 + \frac{3}{8}k_3\right), \quad k_5 = hf\left(t_n + h,\ u_n + \frac{1}{2}k_1 - \frac{3}{2}k_3 + 2k_4\right),$$

$$u_{n+1} = u_n + \frac{1}{6}\left(k_1 + 4k_4 + k_5\right),\ \tilde{u}_{n+1} = u_n + \frac{1}{10}\left(k_1 + 3k_3 + 4k_4 + 2k_5\right)$$

Ici, u_{n+1} est de l'ordre 4 pour le calcul de la solution et $\tilde{u}_{n+1}$ est de l'ordre 3 pour le contrôle du pas (Butcher, 2016) . L'erreur est estimée est exprimée par :

$$E_{n+1} = 0.2\left|u_{n+1} - \tilde{u}_{n+1}\right|$$

Si ε désigne la tolérance acceptée, l'algorithme de Merson sera utilisé avec un nouveau pas $h = h/2$, quand $E_{n+1} > \varepsilon$ et avec un pas $h = 2h$ quand $E_{n+1} \le \varepsilon/64$ et conserve le pas actuel quand $\varepsilon/64 < E_{n+1} \le \varepsilon$.

Soit la fonction **Merson** qui interprète cette méthodologie pour résoudre les équations différentielles sous forme de graphes.

```
function [t,u]=Merson(f,a,b,u0,h,varargin)
% f   : Fonction f(t,u)=du/dt
% a, b: Bornes de l'intervalle (b>a)
% u0  : Condition initiale u(a)
% h   : Pas initial
tn=a;
tol=0.000001; % Définition d'une tolérance
n=1; u(n)=u0; t(n)=tn; R(n)=0;
while t(n)<= b
  k1=h*f(t(n),u(n));
  k2=h*f(t(n)+(h/3),u(n)+(k1/3));
  k3=h*f(t(n)+(h/3),u(n)+(k1/6)+(k2/6));
  k4=h*f(t(n)+(h/2),u(n)+(k1/8)+3*(k3/8));
  k5=h*f(t(n)+h,u(n)+(k1/2)-3*(k3/2)+2*k4);
 u(n+1)=u(n)+(1/6)*(k1+4*k4+k5); uc(n+1)=u(n)+(1/10)*(k1+3*k3+4*k4+2*k5);
```

```
 R(n+1)=0.2*abs(u(n+1)-uc(n+1)); % Evaluation de l'erreur
  if R(n+1)>tol
    h=h/2; k1=h*f(t(n),u(n)); k2=h*f(t(n)+(h/3),u(n)+(k1/3));
    k3=h*f(t(n)+(h/3),u(n)+(k1/6)+(k2/6));
    k4=h*f(t(n)+(h/2),u(n)+(k1/8)+3*(k3/8));
    k5=h*f(t(n)+h,u(n)+(k1/2)-3*(k3/2)+2*k4);
   u(n+1)=u(n)+(1/6)*(k1+4*k4+k5);
  elseif R(n+1)<= tol/64
    h=2*h; k1=h*f(t(n),u(n)); k2=h*f(t(n)+(h/3),u(n)+(k1/3));
    k3=h*f(t(n)+(h/3),u(n)+(k1/6)+(k2/6));
    k4=h*f(t(n)+(h/2),u(n)+(k1/8)+3*(k3/8));
    k5=h*f(t(n)+h,u(n)+(k1/2)-3*(k3/2)+2*k4);
   u(n+1)=u(n)+(1/6)*(k1+4*k4+k5);
  else
 end
 tn=t(n); n=n+1; t(n)=tn+h;
end
t=t';u=u'; plot(t,u,'-',t,u,'or')
end
```

A titre d'exemple, l'application de cette fonction à l'exemple précédent, c'est à dire :

```
>> Merson(@(t,u)u+2*t-t^2,0,2,1,0.1);
```

Conduit aux résultats (Fig. 6.10).

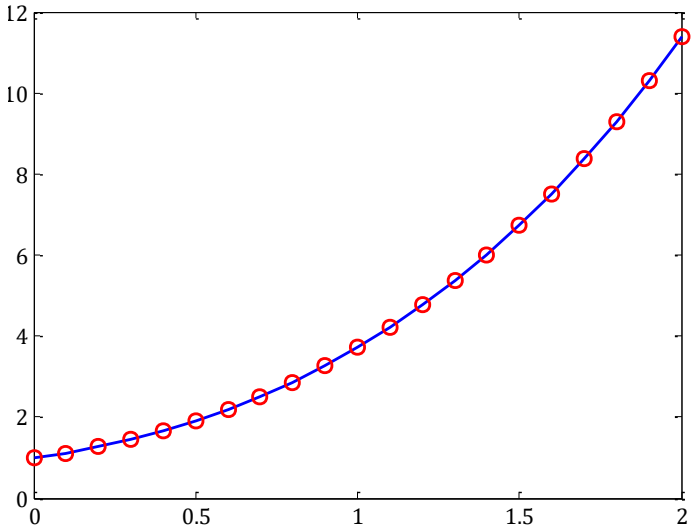


Fig. 6.10 : Exécution de la fonction **`Merson`**.

3.3.2. Méthode de Fehlberg

Cette méthode (Fehlberg, 1968, 1969 et Laplace, 1969) est du type RK45, est un schéma très populaire, elle consiste en une méthode de Runge-Kutta d'ordre 5 pour le calcul de la solution et une méthode de Runge-Kutta d'ordre 4 pour le contrôle du pas. L'algorithme de cette méthode est explicité comme suit :

$$k_1 = hf(t_n,\ u_n),\quad k_2 = hf\left(t_n + \frac{1}{4}h,\ u_n + \frac{1}{4}k_1\right),\quad k_3 = hf\left(t_n + \frac{3}{8}h,\ u_n + \frac{3}{32}k_1 + \frac{9}{32}k_2\right),$$

$$k_4 = hf\left(t_n + \frac{12}{13}h,\ u_n + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right)$$

$$k_5 = hf\left(t_n + h,\ u_n + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right),$$

$$k_6 = hf\left(t_n + \frac{1}{2}h,\ u_n - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right),$$

$$u_{n+1} = u_n + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5$$

$$\tilde{u}_{n+1} = u_n + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6$$

L'erreur entre les deux méthodes est exprimée par :

$$E_{n+1} = \left|\frac{1}{360}k_1 - \frac{128}{4275}k_3 - \frac{2197}{75240}k_4 + \frac{1}{50}k_5 + \frac{2}{55}k_6\right|$$

Le pas optimal h_{adj}, est exprimé en fonction du pas h, de l'erreur E_{n+1} et de la tolérance imposée ε par (Fehlberg, 1969):

$$h_{adj} = \left(\frac{\varepsilon \cdot h}{2E_{n+1}}\right)^{1/4} h \approx 0.84\left(\frac{\varepsilon \cdot h}{E_{n+1}}\right)^{1/4} h$$

Soit la fonction **Fehlberg** qui utilise l'algorithme de Fehlberg avec les mêmes arguments que dans la fonction **Merson**.

```
function [t,u] = Fehlberg(f,a,b,u0,h,varargin)
% f   : La fonction f(t,u)=du/dt
% a, b: Bornes de l'intervalle (b>a)
% u0  : Condition initiale u(a), h   : pas initial
% Définition d'une tolérance
tol=0.000001;
tn=a; n=1; u(n)=u0; t(n)=tn; E(n)=0;
c2=1/4; c3=3/8; c4=12/13; c5=1; c6=1/2;
a21=1/4; a31=3/32; a32=9/32; a41=1932/2197; a42=-7200/2197;
a43=7296/2197; a51=439/216; a52=-8; a53=3680/513; a54=-845/4104;
a61=-8/27; a62=2; a63=-3544/2565; a64=1859/4104; a65=-11/40;
b1=25/216; b3=1408/2565; b4=2197/4104; b5=-1/5;
bc1=16/135; bc3=6656/12825; bc4=28561/56430; bc5=-9/50; bc6=2/55;
while t(n)<= b
 k1=h*f(t(n),u(n));
 k2=h*f(t(n)+c2*h,u(n)+a21*k1);
 k3=h*f(t(n)+c3*h,u(n)+a31*k1+a32*k2);
 k4=h*f(t(n)+c4*h,u(n)+a41*k1+a42*k2+a43*k3);
 k5=h*f(t(n)+c5*h,u(n)+a51*k1+a52*k2+a53*k3+a54*k4);
 k6=h*f(t(n)+c6*h,u(n)+a61*k1+a62*k2+a63*k3+a64*k4+a65*k5);
 u(n+1)=u(n)+b1*k1+b3*k3+b4*k4+b5*k5;
 uc(n+1)=u(n)+bc1*k1+bc3*k3+bc4*k4+bc5*k5+bc6*k6;
 E(n+1)=abs(u(n+1)-uc(n+1)); % Evaluation de l'erreur
if E(n+1)<= tol/64
 h=2*h; k1=h*f(t(n),u(n));
 k2=h*f(t(n)+c2*h,u(n)+a21*k1);
 k3=h*f(t(n)+c3*h,u(n)+a31*k1+a32*k2);
 k4=h*f(t(n)+c4*h,u(n)+a41*k1+a42*k2+a43*k3);
```

```
 k5=h*f(t(n)+c5*h,u(n)+a51*k1+a52*k2+a53*k3+a54*k4);
 k6=h*f(t(n)+c6*h,u(n)+a61*k1+a62*k2+a63*k3+a64*k4+a65*k5);
 u(n+1)=u(n)+b1*k1+b3*k3+b4*k4+b5*k5;
elseif E(n+1)> tol
 h=h/2; k1=h*f(t(n),u(n));
 k2=h*f(t(n)+c2*h,u(n)+a21*k1);
 k3=h*f(t(n)+c3*h,u(n)+a31*k1+a32*k2);
 k4=h*f(t(n)+c4*h,u(n)+a41*k1+a42*k2+a43*k3);
 k5=h*f(t(n)+c5*h,u(n)+a51*k1+a52*k2+a53*k3+a54*k4);
 k6=h*f(t(n)+c6*h,u(n)+a61*k1+a62*k2+a63*k3+a64*k4+a65*k5);
 u(n+1)=u(n)+b1*k1+b3*k3+b4*k4+b5*k5;
else
end
 tn=t(n); n=n+1; t(n)=tn+h;
end
 t=t';u=u';plot(t,u,'-',t,u,'or')
end
```

A titre d'exemple, l'application de cette fonction à l'exemple précédent, c'est à dire :

```
>> Fehlberg(@(t,u)u+2*t-t^2,0,2,1,0.1);
```

Conduit aux résultats (Fig. 6.11).

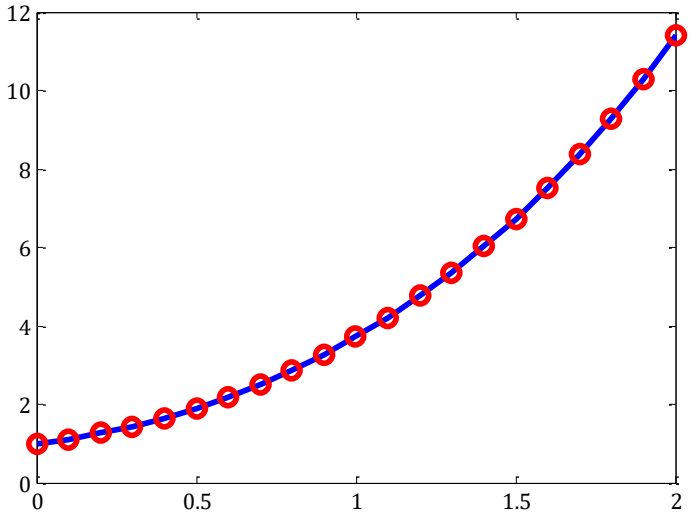


Fig. 6.11 : Exécution des fonctions **`Fehlberg`** et **`DormandAndPrince`**.

3.3.3. Méthode de Dormand et Prince

La méthode proposée par Dormand et Prince (1980) est de l'ordre globale 4, 5. Cette méthode est la base du solveur **`ode45`** de MATLAB. Elle consiste à calculer :

$$k_1 = hf\left(t_n,\ u_n\right)$$

$$k_2 = hf\left(t_n + \frac{1}{5}h,\ u_n + \frac{1}{5}k_1\right)$$

$$k_3 = hf\left(t_n + \frac{3}{10}h,\ u_n + \frac{3}{40}k_1 + \frac{9}{40}k_2\right)$$

$$k_4 = hf\left(t_n + \frac{4}{5}h,\ u_n + \frac{44}{45}k_1 - \frac{56}{15}k_2 + \frac{32}{9}k_3\right)$$

$$k_5 = hf\left(t_n + \frac{8}{9}h,\ u_n + \frac{19372}{6561}k_1 - \frac{25360}{2187}k_2 + \frac{64448}{6561}k_3 - \frac{212}{729}k_4\right)$$

$$k_6 = hf\left(t_n + h,\ u_n + \frac{9017}{3168}k_1 - \frac{355}{33}k_2 - \frac{46732}{5247}k_3 + \frac{49}{176}k_4 - \frac{5103}{18656}k_5\right)$$

$$k_7 = hf\left(t_n + h,\ u_n + \frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6\right)$$

Le calcul approché de la solution se fait par la méthode de Runge-Kutta d'ordre 5 suivante :

$$u_{n+1} = u_n + \frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6$$

Le contrôle du pas se fait par la méthode de Runge-Kutta d'ordre 4 suivante :

$$\tilde{u}_{n+1} = u_n + \frac{5179}{57600}k_1 + \frac{7571}{16695}k_3 + \frac{393}{640}k_4 - \frac{92097}{339200}k_5 + \frac{187}{2100}k_6 + \frac{1}{40}k_7$$

L'erreur entre les deux méthodes est :

$$E_{n+1} = \left|u_{n+1} - \tilde{u}_{n+1}\right| = \left|\frac{71}{57600}k_1 - \frac{71}{16695}k_3 + \frac{71}{1920}k_4 - \frac{17253}{339200}k_5 + \frac{22}{525}k_6 - \frac{1}{40}k_7\right|$$

L'algorithme de Dormand et Prince est analogue à celle de Fehlberg, soit la fonction **DormandAndPrince**. Qui permet de résoudre numériquement les équations différentielles :

```
function [t,u] = DormandAndPrince(f,a,b,u0,h,varargin)
% f   : La fonction f(t,u)=du/dt
% a, b: Bornes de l'intervalle (b>a)
% u0  : Condition initiale u(a), h   : pas initial
tol=0.000001; % Définition d'une tolérance
tn=a; n=1; u(n)=u0; t(n)=tn; E(n)=0;
c2=1/5;  a21=1/5; c3=3/10; a31=3/40; a32=9/40;
c4=4/5; a41=44/45; a42=-56/15; a43=32/9;
c5=8/9; a51=19372/6561; a52=-25360/2187; a53=64448/6561; a54=-212/729;
c6=1; a61=9017/3168; a62=-355/33; a63=46732/5247; a64=49/176;
a65=-5103/18656;
c7=1; a71=35/384; a73=500/1113; a74=125/192; a75=-2187/6784; a76=11/84;
b1=35/384; b2=0; b3=500/1113; b4=125/192; b5=-2187/6784; b6=11/84;
e1=5179/57600; e3=7571/16695; e4=393/640; e5=-92097/339200; e6=187/2100;
e7=1/40;
while t(n)<= b
  k1=h*f(t(n),u(n)); k2=h*f(t(n)+c2*h,u(n)+a21*k1);
  k3=h*f(t(n)+c3*h,u(n)+a31*k1+a32*k2);
  k4=h*f(t(n)+c4*h,u(n)+a41*k1+a42*k2+a43*k3);
  k5=h*f(t(n)+c5*h,u(n)+a51*k1+a52*k2+a53*k3+a54*k4);
  k6=h*f(t(n)+c6*h,u(n)+a61*k1+a62*k2+a63*k3+a64*k4+a65*k5);
  k7=h*f(t(n)+c7*h,u(n)+a71*k1+a73*k3+a74*k4+a75*k5+a76*k6);
 u(n+1)=u(n)+b1*k1+b3*k3+b4*k4+b5*k5+b6*k6;
 uc(n+1)=u(n)+e1*k1+e3*k3+e4*k4+e5*k5+e6*k6+e7*k7;
 E(n+1)=abs(u(n+1)-uc(n+1)); % Evaluation de l'erreur
if E(n+1)> tol
  h=h/2; k1=h*f(t(n),u(n)); k2=h*f(t(n)+c2*h,u(n)+a21*k1);
  k3=h*f(t(n)+c3*h,u(n)+a31*k1+a32*k2);
  k4=h*f(t(n)+c4*h,u(n)+a41*k1+a42*k2+a43*k3);
  k5=h*f(t(n)+c5*h,u(n)+a51*k1+a52*k2+a53*k3+a54*k4);
  k6=h*f(t(n)+c6*h,u(n)+a61*k1+a62*k2+a63*k3+a64*k4+a65*k5);
  k7=h*f(t(n)+c7*h,u(n)+a71*k1+a73*k3+a74*k4+a75*k5+a76*k6);
```

```
 u(n+1)=u(n)+b1*k1+b3*k3+b4*k4+b5*k5+b6*k6;
 elseif E(n+1)<= tol/128
   h=2*h; k1=h*f(t(n),u(n)); k2=h*f(t(n)+c2*h,u(n)+a21*k1);
   k3=h*f(t(n)+c3*h,u(n)+a31*k1+a32*k2);
   k4=h*f(t(n)+c4*h,u(n)+a41*k1+a42*k2+a43*k3);
   k5=h*f(t(n)+c5*h,u(n)+a51*k1+a52*k2+a53*k3+a54*k4);
   k6=h*f(t(n)+c6*h,u(n)+a61*k1+a62*k2+a63*k3+a64*k4+a65*k5);
   k7=h*f(t(n)+c7*h,u(n)+a71*k1+a73*k3+a74*k4+a75*k5+a76*k6);
  u(n+1)=u(n)+b1*k1+b3*k3+b4*k4+b5*k5+b6*k6;
 else
end
 tn=t(n); n=n+1; t(n)=tn+h;
end
 t=t';u=u'; plot(t,u,'-',t,u,'or')
end
```

A titre d'exemple, l'application de cette fonction à l'exemple précédent, c'est à dire :

```
>> DormandAndPrince(@(t,u)u+2*t-t^2,0,2,1,0.1);
```

Qui conduit au même résultat obtenu par la fonction Fehlberg pour le même pas initial $h=0.1$ et la même tolérance $\varepsilon=10^{-6}$ (Fig. 6.11).

D'autres méthodes d'ordre élevé ont été proposées plus récemment, comme celle de Bogacki et Shapmine (1989), Cash et Karp (1990) etc.

3.3.4. Méthode de Bogacki et Shampine

C'est une méthode de Runge-Kutta d'ordre 2 couplée avec une méthode Runge-Kutta d'ordre trois, elle consiste à calculer :

$$k_1=hf\left(t_n,\ u_n\right),\quad k_2=hf\left(t_n+\frac{1}{2}h,\ u_n+\frac{1}{2}k_1\right),\quad k_3=hf\left(t_n+\frac{3}{4}h,\ u_n+\frac{3}{4}k_2\right)$$

$$k_4=hf\left(t_n+h,\ u_n+k_1\right)$$

Pour le calcul de la solution,

$$u_{n+1}=u_n+\frac{7}{24}k_1+\frac{1}{4}k_2+\frac{1}{3}k_3+\frac{1}{8}k_4$$

Pour le control du pas,

$$\tilde{u}_{n+1}=u_n+\frac{1}{9}\left(2k_1+3k_2+4k_3\right)$$

3.3.5. Méthode de Cash et Karp

C'est une méthode d'ordre 4.5 (ou RK4,5) (Cash et Karp, 1990) dont l'algorithme est :

$$k_1=hf\left(t_n,\ u_n\right),\quad k_2=hf\left(t_n+\frac{1}{5}h,\ u_n+\frac{1}{5}k_1\right),\quad k_3=hf\left(t_n+\frac{3}{10}h,\ u_n+\frac{3}{40}k_1+\frac{9}{40}k_2\right),$$

$$k_4=hf\left(t_n+\frac{3}{5}h,\ u_n+\frac{3}{10}k_1-\frac{9}{10}k_2+\frac{6}{5}k_3\right)$$

$$k_5=hf\left(t_n+h,\ u_n-\frac{11}{54}k_1+\frac{5}{2}k_2-\frac{70}{27}k_3+\frac{35}{27}k_4\right),$$

$$k_6=hf\left(t_n+\frac{7}{8}h,\ u_n+\frac{1631}{55296}k_1+\frac{175}{512}k_2+\frac{575}{13824}k_3+\frac{44275}{110592}k_4+\frac{253}{4096}k_5\right),$$

Pour le calcul de la solution,

$$u_{n+1} = u_n + \frac{37}{378}k_1 + \frac{250}{621}k_3 + \frac{125}{594}k_4 + \frac{512}{1771}k_6,$$

Pour le control du pas,

$$\tilde{u}_{n+1} = u_n + \frac{2825}{27648}k_1 + \frac{18575}{48384}k_3 + \frac{13525}{55296}k_4 + \frac{277}{14336}k_5 + \frac{1}{4}k_6$$

4. Méthodes de Runge-Kutta implicites

Les méthodes de Runge-Kutta implicites ont été introduites dans les travaux de Butcher (1963). Comme dans les méthodes de Runge-Kutta explicites, le tableau de Butcher (Fig. 6.12) permet d'avoir la formule générale des méthodes de Runge-Kutta à s pas qui peut être exprimé par :

$$u_{n+1} = u_n + \sum_{p=1}^{s} b_p k_p$$

avec,

$$k_p = hf\left(t_n + c_p h,\ u_n + \sum_{q=1}^{s} a_{pq}k_q\right),\quad \forall p = 1,\ 2,\ \ldots,\ s$$

qui s'écrit aussi,

$$\Psi_p(k_1,\ k_2,\ \ldots,\ k_s) = k_p - hf\left(t_n + c_p h,\ u_n + \sum_{q=1}^{s} a_{pq}k_q\right) = 0,\quad p = 1,\ 2,\ \ldots,\ s$$

qui représente un système de s équations algébriques à k_1, k_2, …, k_s pentes inconnues qu'il faut calculer pour évaluer la solution approchée u_{n+1} de l'équation différentielle. A titre d'exemple, la formule de la méthode de Runge-Kutta implicite d'ordre 4 est explicitée par :

$$u_{n+1} = u_n + b_1k_1 + b_2k_2 + b_3k_3 + b_4k_4$$

c_1	a_{11}	a_{12}	$\cdots$	a_{1s}
c_2	a_{21}	a_{22}		a_{2s}
$\vdots$	$\vdots$			$\vdots$
c_s	a_{s1}	a_{s2}	$\cdots$	a_{ss}
	b_1	b_2	$\cdots$	b_s

Fig. 6.12 : Tableau de Butcher.

avec,

$$k_1 = hf\left(t_n + c_1 h,\ u_n + \sum_{i=1}^{4} a_{1i}k_i\right),\ k_2 = hf\left(t_n + c_2 h,\ u_n + \sum_{i=1}^{4} a_{2i}k_i\right),\ k_3 = hf\left(t_n + c_3 h,\ u_n + \sum_{i=1}^{4} a_{3i}k_i\right)$$

$$k_4 = hf\left(t_n + c_4 h,\ u_n + \sum_{i=1}^{4} a_{4i}k_i\right)$$

qui peuvent être exprimées aussi sous la forme d'un système d'équations algébriques :

$$\begin{cases} k_1 - hf\left(t_n + c_1 h,\ u_n + a_{11}k_1 + a_{12}k_2 + a_{13}k_3 + a_{14}k_4\right) = 0, \\ k_2 - hf\left(t_n + c_2 h,\ u_n + a_{21}k_1 + a_{22}k_2 + a_{23}k_3 + a_{24}k_4\right) = 0, \\ k_3 - hf\left(t_n + c_3 h,\ u_n + a_{31}k_1 + a_{32}k_2 + a_{33}k_3 + a_{34}k_4\right) = 0, \\ k_4 - hf\left(t_n + c_4 h,\ u_n + a_{41}k_1 + a_{42}k_2 + a_{43}k_3 + a_{44}k_4\right) = 0. \end{cases}$$

qui peut être résolue numériquement pour déterminer k_1, k_2, k_3 et k_4.

Dans un système d'équations différentielles ou une équation différentielle d'ordre *m*, le nombre d'inconnus à déterminer est de 4*m* si la méthode utilisée est de Runge-Kutta implicite d'ordre 4, ce qui complique la détermination de la solution. Pour ce faire d'une autre manière, soit l'équation intégrale du problème de Cauchy sur l'intervalle $[t_n, t]$:

$$u(t) = u(t_n) + \int_{t_n}^{t} f\left(r,\ u(r)\right)dr$$

La partie intégrante dans cette équation est peut-être approchée par l'intégrale :

$$\int_{t_n}^{t} f\left(r,\ u(r)\right)dt \approx \int_{t_n}^{t} p(r)dr$$

ou, $p(r)$ est le polynôme de degré inférieur à s interpolant les points dont les nœuds : $\left\{t_{n,i} \equiv t_n + \tau_i h,\ \ i = 1,\ 2,\ \ldots,\ s\right\}$ avec $0 \le \tau_1 < \tau_2 < \cdots < \tau_s \le 1$. $p(r)$ est exprimée suivant la formule d'interpolation de Lagrange :

$$p(r) = \sum_{j=1}^{s} f\left(t_{n,j},\ u\left(t_{n,j}\right)\right) L_j(r) \ \text{ avec, } \ L_j(r) = \prod_{\substack{k=1 \\ k\neq j}}^{s} \frac{\left(r - t_{n,k}\right)}{\left(t_{n,j} - t_{n,k}\right)} = \frac{\prod_{\substack{k=1 \\ k\neq j}}^{s}\left(r - t_{n,k}\right)}{h^{s-1}\prod_{\substack{k=1 \\ k\neq j}}^{s}\left(\tau_j - \tau_k\right)}$$

par conséquent,

$$\int_{t_n}^{t} p(r)dr = \sum_{j=1}^{s} f\left(t_{n,j},\ u\left(t_{n,j}\right)\right) \int_{t_n}^{t} L_j(r)dr$$

ou,

$$\int_{t_n}^{t} p(r)dr = \frac{1}{h^{s-1}} \sum_{j=1}^{s} \frac{f\left(t_{n,j},\ u\left(t_{n,j}\right)\right)}{\prod_{\substack{k=1 \\ k\neq j}}^{s}\left(\tau_j - \tau_k\right)} \int_{t_n}^{t} \prod_{\substack{k=1 \\ k\neq j}}^{s}\left(r - t_{n,k}\right)dr$$

l'approximation de l'équation intégrale entre t_n et $t_{n,i}$ est :

$$u_{n,i} = u_n + \sum_{j=1}^{s} f\left(t_{n,j},\ u\left(t_{n,j}\right)\right) \int_{t_n}^{t_{n,i}} L_j(r)dr \ \Rightarrow\ u_{n,i} = u_n + \frac{1}{h^{s-1}} \sum_{j=1}^{s} \frac{f\left(t_{n,j},\ u\left(t_{n,j}\right)\right)}{\prod_{\substack{k=1 \\ k\neq j}}^{s}\left(\tau_j - \tau_k\right)} \int_{t_n}^{t_{n,i}} \prod_{\substack{k=1 \\ k\neq j}}^{s}\left(r - t_{n,k}\right)dr$$

L'intégrale $\int_{t_n}^{t_{n,i}} L_j(r)dr$ est aisément calculable. Soit le cas $\tau_s = 1$, alors :

$$u_{n+1} = u_n + \sum_{j=1}^{s} f\left(t_{n,j},\ u\left(t_{n,j}\right)\right) \int_{t_n}^{t_{n+1}} L_j(r)dr \ \Rightarrow\ u_{n+1} = u_n + \frac{1}{h^{s-1}} \sum_{j=1}^{s} \frac{f\left(t_{n,j},\ u\left(t_{n,j}\right)\right)}{\prod_{\substack{k=1 \\ k\neq j}}^{s}\left(\tau_j - \tau_k\right)} \int_{t_n}^{t_{n+1}} \prod_{\substack{k=1 \\ k\neq j}}^{s}\left(r - t_{n,k}\right)dr$$

a titre particulier *s* =2,

$$u_{n+1} = u_n + \frac{1}{h}\left(\frac{f\left(t_{n,1},\ u\left(t_{n,1}\right)\right)}{\left(\tau_1 - 1\right)} \int_{t_n}^{t_{n+1}} \left(r - t_{n+1}\right)dr + \frac{f\left(t_{n+1},\ u\left(t_{n+1}\right)\right)}{\left(1 - \tau_1\right)} \int_{t_n}^{t_{n+1}} \left(r - t_{n,1}\right)dr \right)$$

c'est-à-dire,

$$u_{n+1} = u_n + \frac{h}{2(1-\tau_1)}\Big[f\big(t_n + \tau_1 h,\ u(t_n + \tau_1 h)\big) + (1-2\tau_1) f\big(t_{n+1},\ u(t_{n+1})\big)\Big]$$

C'est la formule de la méthode de Runge-Kutta implicite d'ordre 2 (RKI2). Si $\tau_1 = 0$, on retrouve la formule trapézoïdale :

$$u_{n+1} = u_n + \frac{h}{2}\Big[f\big(t_n,\ u(t_n)\big) + f\big(t_{n+1},\ u(t_{n+1})\big)\Big]$$

Cette méthodologie de forcer l'équation d'approximation à être vraie aux points de nœuds $\{t_{n,i},\ i = 1,\ 2,\ \ldots,\ s\}$ est appelée *Méthode de collocation par points*, et les points auxquels l'égalité est forcée sont appelés les *points de nœud de collocation*. Il faut noter que certaines méthodes de Runge-Kutta ne sont pas des méthodes de collocation (Iserles, 2009).

Considérons le cas le plus générale $0 \le \tau_1 < \tau_2 \le 1$, le polynôme d'interpolation $p(r)$ dans ce cas est exprimé par :

$$p(r) = \frac{1}{h(\tau_2 - \tau_1)}\Big[(t_{n+1} - r) f\big(t_{n,1},\ u(t_{n,1})\big) + (r - t_n) f\big(t_{n,2},\ u(t_{n,2})\big)\Big]$$

le calcul de la quantité,

$$\sum_{j=1}^{2} f\big(t_{n,j},\ u(t_{n,j})\big) \int_{t_n}^{t_{n,i}} L_j(r)\, dr$$

conduit au tableau de Butcher suivant :

τ_1	$\left(\tau_2^2 - (\tau_2 - \tau_1)^2\right)\big/2(\tau_2 - \tau_1)$	$-\tau_1^2/2(\tau_2 - \tau_1)$
τ_2	$-\tau_2^2/2(\tau_2 - \tau_1)$	$\left((\tau_2 - \tau_1)^2 - \tau_1^2\right)\big/2(\tau_2 - \tau_1)$
	$\left(\tau_2^2 - (1 - \tau_2)^2\right)\big/2(\tau_2 - \tau_1)$	$\left((1 - \tau_1)^2 - \tau_1^2\right)\big/2(\tau_2 - \tau_1)$

Soit la méthode de Gauss d'ordre 2 correspondant à :

$$\tau_1 = \frac{3 - \sqrt{3}}{6},\quad \tau_2 = \frac{3 + \sqrt{3}}{6}$$

Le tableau de Butcher est :

$\left(3 - \sqrt{3}\right)/6$	$1/4$	$\left(3 - 2\sqrt{3}\right)/12$
$\left(3 + \sqrt{3}\right)/6$	$\left(3 + 2\sqrt{3}\right)/12$	$1/4$
	$1/2$	$1/2$

en termes d'expression,

$$k_1 = hf\left(t_n + \frac{3 - \sqrt{3}}{6} h,\ u_n + \frac{1}{4} k_1 + \frac{3 - 2\sqrt{3}}{12} k_2\right)$$

$$k_2 = hf\left(t_n + \frac{3+\sqrt{3}}{6}h,\ u_n + \frac{3+2\sqrt{3}}{12}k_1 + \frac{1}{4}k_2\right)$$

Une fois k_1 et k_2 sont calculés, alors, $u_{n+1} = u_n + \frac{1}{2}(k_1 + k_2)$, l'erreur de troncature de cette méthode est de l'ordre 4 (Iserles, 2009).

Soit l'exemple du problème suivant :

$$\frac{du}{dt} = 50(\cos t - u),\ \ 0 \le t \le 3$$
$$u(0) = 1$$

Le schéma de résolution suivant cette méthode est exprimé par :

$$k_1 = hf\left(t_n + \tau_1 h,\ u_n + \frac{1}{4}k_1 + \frac{3-2\sqrt{3}}{12}k_2\right) = 50h\left(\cos(t_n + \tau_1 h) - u_n - \frac{1}{4}k_1 - \frac{3-2\sqrt{3}}{12}k_2\right)$$
$$k_2 = hf\left(t_n + \tau_2 h,\ u_n + \frac{3+2\sqrt{3}}{12}k_1 + \frac{1}{4}k_2\right) = 50h\left(\cos(t_n + \tau_2 h) - u_n - \frac{3+2\sqrt{3}}{12}k_1 - \frac{1}{4}k_2\right)$$

Qui s'écrit aussi,

$$\left(1 + \frac{50h}{4}\right)k_1 + 50h\left(\frac{3-2\sqrt{3}}{12}\right)k_2 = 50h\left(\cos(t_n + \tau_1 h) - u_n\right)$$

$$50h\left(\frac{3+2\sqrt{3}}{12}\right)k_2 + \left(1 + \frac{50h}{4}\right)k_2 = 50h\left(\cos(t_n + \tau_2 h) - u_n\right)$$

Ou sous forme matricielle,

$$\begin{pmatrix} 1+(50h/4) & 50h\left(3-2\sqrt{3}\right)/12 \\ 50h\left(3+2\sqrt{3}\right)/12 & 1+(50h/4) \end{pmatrix}\begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} 50h\left(\cos(t_n + \tau_1 h) - u_n\right) \\ 50h\left(\cos(t_n + \tau_2 h) - u_n\right) \end{pmatrix}$$

De ce système d'équations, k_1 et k_2 peuvent être déterminés. Soit le script MATLAB qui permet la résolution de ce problème (Fig. 6.13) :

```
h=0.1;
t=0:h:3;
u(1)=0;
t1=(3-sqrt(3))/6; t2=(3+sqrt(3))/6;
a=[1/4 (3-2*sqrt(3))/12 ; (3+2*sqrt(3))/12 1/4];
B=inv([1+50*h*a(1,1) 50*h*a(1,2) ; 50*h*a(2,1) 1+50*h*a(2,2)]);
for n=1:(numel(t)-1)
    b1=50*h*(cos(t(n)+t1*h)-u(n));
   b2=50*h*(cos(t(n)+t2*h)-u(n));
  k1=B(1,1)*b1+B(1,2)*b2; k2=B(2,1)*b1+B(2,2)*b2;
 u(n+1)=u(n)+(1/2)*(k1+k2);
end
plot(t,u,'-',t,u,'or')
```

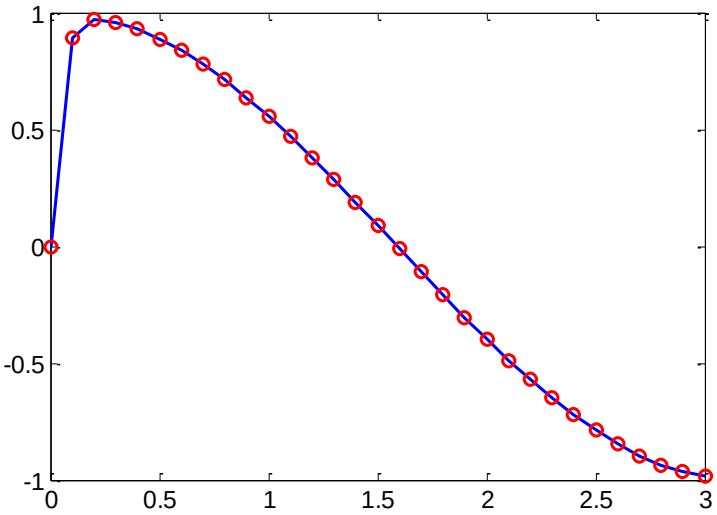


Fig. 6.13 : Solution approchée de l'EDO.

Les méthodes de Runge-Kutta implicites sont couteuses, autres schémas basés sur les méthodes Kuntzmann et Butcher d'ordre 2s et de quadrature de Radau et Lobatto peuvent être consulter dans les références comme Butcher (2016) , Hairer *et al*. (1993). Soit quelques tableaux de Butcher des méthodes de Runge-Kutta implicites qui peuvent être utilisés pour construire le schéma de résolution numérique du problème de Cauchy (Figs. 6.14, 6.15 et 6.16).

$$\begin{array}{c|ccc}
\frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\
\frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\
\frac{1}{2}+\frac{\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}+\frac{\sqrt{15}}{15} & \frac{5}{36} \\
\hline
 & \frac{5}{18} & \frac{4}{9} & \frac{5}{18}
\end{array}$$

Fig. 6.14 : Tableau de Butcher de la méthode Kuntzmann et Butcher d'ordre 6.

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\
1 & 0 & 1 & 0 \\
\hline
 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6}
\end{array}
\qquad
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
\frac{5-\sqrt{5}}{10} & \frac{5+\sqrt{5}}{60} & \frac{1}{6} & \frac{15-7\sqrt{5}}{60} & 0 \\
\frac{5+\sqrt{5}}{10} & \frac{5-\sqrt{5}}{60} & \frac{15+7\sqrt{5}}{60} & \frac{1}{6} & 0 \\
1 & \frac{1}{6} & \frac{5-\sqrt{5}}{12} & \frac{5+\sqrt{5}}{12} & 0 \\
\hline
 & \frac{1}{12} & \frac{5}{12} & \frac{5}{12} & \frac{1}{12}
\end{array}$$

Fig. 6.15 : Tableaux de Butcher des formules de Lobatto d'ordre 4 et 6.

$\frac{4-\sqrt{6}}{10}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{296-169\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{225}$
$\frac{4+\sqrt{6}}{10}$	$\frac{296+169\sqrt{6}}{1800}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{-2-3\sqrt{6}}{225}$
1	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

Fig. 6.16 : Tableau de Butcher de la méthode de Radau IIA d'ordre 5.

5. Stabilité des méthodes de Runge-Kutta

5.1. L'idée de stabilité

L'idée de stabilité pour une méthode numérique est qu'une perturbation de la donnée initiale et du second membre ne doit pas être amplifiée au-delà de tout contrôle par la méthode. D'une façon plus simple, la stabilité peut être interprétée comme,

$$\max_{t_0\le t_n\le t_N}\left|u(t_n)-u_n\right|\to 0 \text{ quand } h\to 0$$

Pour mieux illustrer le sujet de la stabilité, soit le problème de Cauchy :

$$\frac{du}{dt}=-5u+30\left(3e^{-t/5}+1\right) \text{ avec } u(0)=10$$

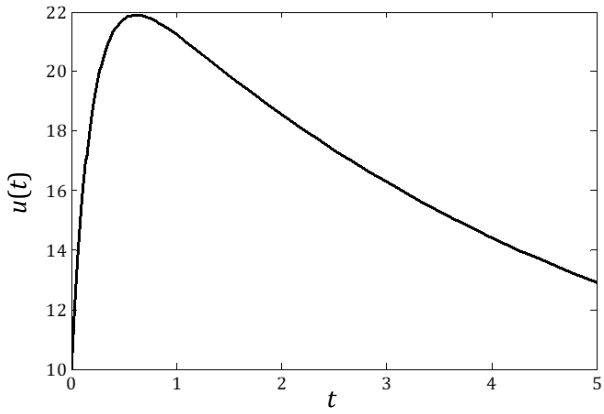


Fig. 6.17 : Solution exacte de $u'=-5u+30\left(3e^{-t/5}+1\right),\ u(0)=10.$

la solution exacte de ce problème est exprimée par (Fig. 6.17) :

$$u(t)=\frac{75}{4}e^{-t/5}-\frac{59}{4}e^{-5t}+6$$

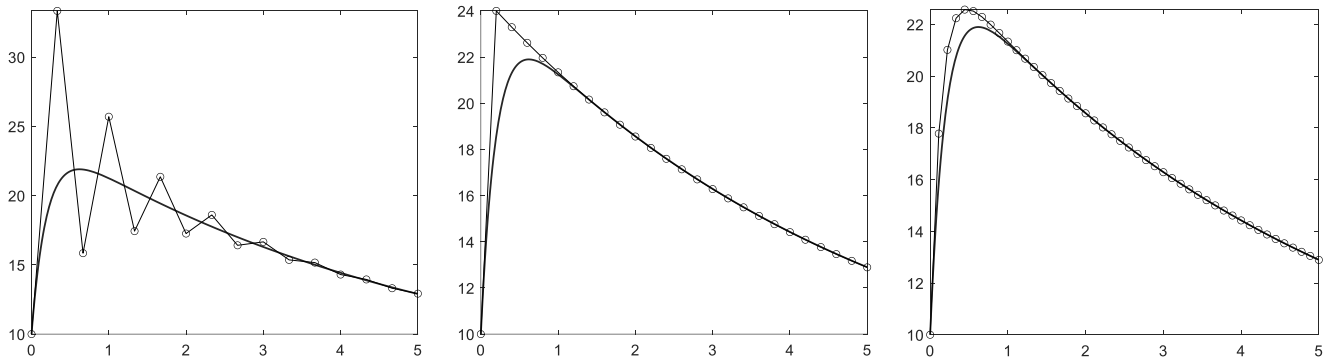
On va mener plusieurs expériences numériques avec les méthodes d'Euler explicite et implicite pour trois valeurs de h : 1/3, 1/6 et 1/9. La méthode d'Euler explicite donne le schéma :

$$\begin{cases} u_{n+1}=(1-5h)u_n+90he^{-t_n/5}+30h \quad n\ge 0,\\ u_0=10, \quad t_n=nh. \end{cases}$$

la méthode d'Euler implicite conduit à :

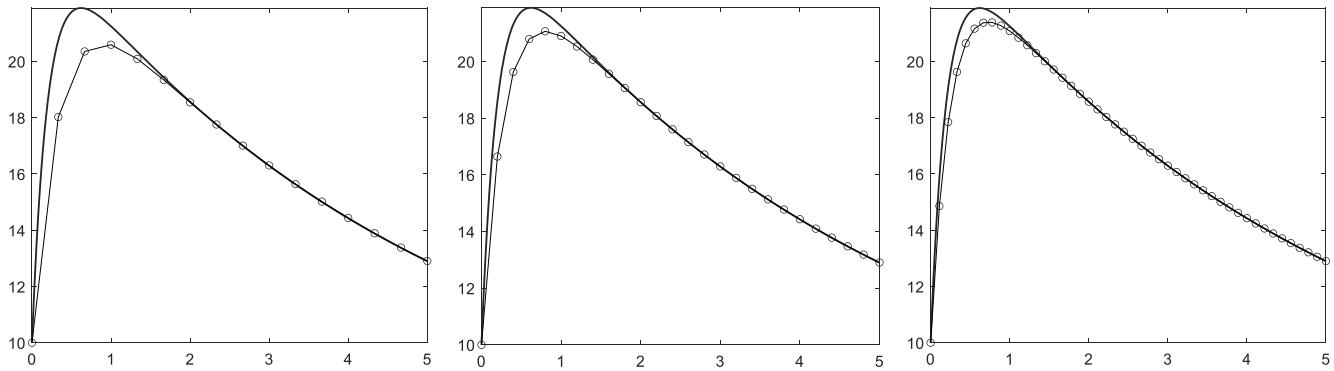
$$\begin{cases} u_{n+1}=\dfrac{1}{1+5h}\left(u_n+90he^{-t_{n+1}/5}+30h\right) \quad n\ge 0,\\ u_0=10, \quad t_n=nh. \end{cases}$$

Les solutions générées par les méthodes d'Euler explicite et implicite sont représentées sur les figures 6.18 et 6.19 pour $0\le t\le 5$,



$h = 1/3$ $h = 1/5$ $h = 1/9$

Fig. 6.18 : Solution par la méthode d'Euler explicite.



$h = 1/3$ $h = 1/5$ $h = 1/9$

Fig. 6.19 : Solution par la méthode d'Euler implicite.

Pour la méthode d'Euler explicite (Fig. 6.18), on constate pour le cas $h = 1/3$ la solution est oscillante ceci est le symptôme d'une instabilité. Pour $h = 1/5$, un effet important sur la solution numérique subsiste mais qui s'atténuent, à partir de $t = 1$, la représentation graphique semble les faire disparaitre et la solution semble devenir une courbe régulière qui s'approche de la solution exacte. Enfin, pour $h = 1/9$, la solution ressemble à la solution exacte mais les courbes ne deviennent proches qu'à partir de $t \geq 1$. La méthode d'Euler explicite souffre d'un phénomène d'instabilité à moins que h soit assez petit. Cette contrainte s'appelle *condition de stabilité*. Pour la méthode d'Euler implicite (Fig. 6.19), on constate que quel que soit h, il n' y a plus d'oscillations et le comportement est raisonnable. Ces exemples vus ci-dessus nous conduisent à la théorie de la stabilité.

Pour étudier la stabilité, on considère le problème de Cauchy suivant :

$$\begin{cases} u'(t) = \lambda u(t), & t \geq 0 \\ u(0) = 1. \end{cases} \qquad \lambda \in \mathbb{C},\ \mathrm{Re}(\lambda) < 1$$

Dont la solution exacte est $u(t) = e^{\lambda t}$. Le résultat obtenu est transféré à l'étude de la stabilité.

Soit le développement de la fonction $u'(t) = f(t,\ u(t))$ au voisinage de $(t_0,\ u_0)$,

$$u'(t) \approx f(t_0,\ u_0) + (t - t_0)\left.\frac{\partial f}{\partial t}\right|_{t_0,\ u_0} + (u(t) - u_0)\left.\frac{\partial f}{\partial u}\right|_{t_0,\ u_0}$$

c'est-à-dire,

$$u'(t) \approx \lambda u(t) + g(t) \text{ avec, } g(t) = (t - t_0)\left.\frac{\partial f}{\partial t}\right|_{t_0,\ u_0}$$

Le terme inhomogène $g(t)$ disparaîtra de toutes les dérivations concernant la stabilité numérique, car nous sommes concernés par les différences de la solution de l'équation différentielle.

L'application de la méthode d'Euler explicite à ce problème conduit à :

$$u_{n+1} = (1+\lambda h)u_n$$

c'est une suite géométrique convergente si et seulement si $|1+\lambda h|<1$ qui représente dans le plan complexe un disque sans contour de centre $\lambda h=-1$ et de rayon 1 (Fig. 6.20). Si $\lambda \in \mathbb{R}$. L'intervalle de stabilité est défini par $-1<1+\lambda h<1$ c'est-à-dire, $-2<\lambda h<0$. Par exemple, si $\lambda=-8$ alors on a la contrainte $0<h<1/4$ pour avoir la stabilité.

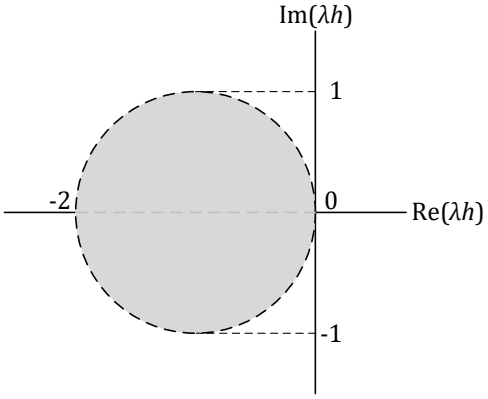


Fig. 6.20 : Région de stabilité pour le schéma d'Euler explicite.

5.2. Stabilité absolue

Par définition, une méthode numérique est dite absolument stable (ou, A-stable) si quand on l'applique au problème $u'(t)=\lambda u(t)$, sa solution tend vers 0 quand $n\to\infty$ pour toute donnée initiale.

Sachant que λh est complexe alors, l'ensemble des valeurs λh dans le plan complexe pour lesquelles une méthode est absolument stable forme la région de stabilité absolue (Fig. 6.20).

Soit la méthode de Runge-Kutta RK2 (c'est-à-dire d'ordre 3) :

$$u_{n+1} = u_n + h\left[(1-\theta)k_1 + \theta k_2\right] \text{ avec, } k_1 = f(t_n,\, u_n),\quad k_2 = f\left(t_n + \frac{h}{2\theta},\, u_n + \frac{h}{2\theta}k_1\right)$$

L'application de la méthode au problème $u'(t)=\lambda u(t),\ u(0)=1$ conduit à :

$$u_{n+1} = u_n + h\left[(1-\theta)\lambda u_n + \lambda\theta\left(u_n + \frac{h}{2\theta}\lambda u_n\right)\right] = \left(1+\lambda h + \frac{\lambda^2 h^2}{2}\right)u_n$$

on pose,

$$z=\lambda h \text{ alors, } u_{n+1} = \left(1+z+\frac{z^2}{2}\right)u_n \equiv R(z)u_n$$

$R(z)$, dite la fonction de stabilité. Pour que la méthode soit stable il faut que $|R(z)|<1$. Sachant,

$$e^z = 1+z+\frac{z^2}{2}+O(z^3) \;\Rightarrow\; R(z)=e^z+O(z^3)$$

Si z est réelle pure, et pour que la méthode de Runge-Kutta est stable c'est-à-dire $u_n\to 0$, il faut que $z<0$. Sachant que $1+z+z^2/2>1 \;\Rightarrow\; z>-2$, soit donc l'intervalle de stabilité,

$$z\in\,]-2,\ 0[$$

Pour avoir $u_n \to 0$, il faut que $\mathrm{Re}(z) < 0$, ce qui conduit à déterminer dans le plan complexe la région de stabilité. Cherchant sa frontière à la limite, en posant $z = x + iy$ et on cherche la relation entre *x* et *y* tels que $|R(z)| = 1$ c'est-à-dire, $|1 + z + z^2/2| = 1$,

$$R(z) = 1 + z + \frac{z^2}{2} = 1 + x + iy + \frac{1}{2}\left(x^2 - y^2 + 2ixy\right) = 1 + x + \frac{1}{2}\left(x^2 - y^2\right) + i(x+1)y$$

$$4\left|1 + z + \frac{z^2}{2}\right|^2 = \left(2 + 2x + x^2 - y^2\right)^2 + 4y^2(x+1)^2 = \left(1 + (1+x)^2 - y^2\right)^2 + 4y^2(x+1)^2$$

$$= 1 + (1+x)^4 + y^4 + 2(1+x)^2 - 2y^2 - 2y^2(x+1)^2 + 4y^2(x+1)^2$$

$$= 1 + (1+x)^4 + y^4 + 2(1+x)^2 + 2y^2(x+1)^2 - 2y^2$$

$$= \left(1 + (1+x)^2 + y^2\right)^2 - 4y^2$$

donc,

$$4\left|1 + z + \frac{z^2}{2}\right|^2 = 4 \;\Leftrightarrow\; \left(1 + (1+x)^2 + y^2\right)^2 - 4y^2 = 4 \;\Rightarrow\; \left(1 + (1+x)^2 + y^2\right)^2 = 4\left(y^2 + 1\right)$$

soit encore,

$$1 + (1+x)^2 + y^2 = 2\sqrt{1+y^2} \;\Rightarrow\; (1+x)^2 + \left(y^2 + 1\right) - 2\sqrt{1+y^2} + 1 = 1$$

c'est-à-dire,

$$(1+x)^2 + \left(\sqrt{1+y^2} - 1\right)^2 = 1$$

La paramétrisation est donnée par :

$$1 + x = \cos\phi \;\Rightarrow\; x = \cos\phi - 1 \text{ et } \sqrt{1+y^2} - 1 = \sin\phi \;\Rightarrow\; y = \pm\sqrt{(2 + \sin\phi)\sin\phi}$$

Les résultats exprimés ci-dessus pour la méthode de Runge-Kutta d'ordre 2 RK2 se généralisent aux méthodes de Runge-Kutta d'ordre *m* (RK*m*) . La fonction de stabilité est exprimée par :

$$R(z) = e^z + O\left(z^{m+1}\right) = 1 + z + \frac{z^2}{2!} + \cdots + \frac{z^m}{m!} + O\left(z^{m+1}\right)$$

Soit le tableau (Tab. 6.1) des intervalles de stabilité de quatre premières méthodes de Runge-Kutta.

Tab. 6.1 : Intervalles de stabilité des méthodes de Runge-Kutta.

Méthode	*R(z)*	Intervalle de stabilité absolue
RK1	$1+z$	]–2, 0[
RK2	$1+z+\frac{z^2}{2}$	]–2, 0[
RK3	$1+z+\frac{z^2}{2}+\frac{z^3}{6}$	]–2.513, 0[
RK4	$1+z+\frac{z^2}{2}+\frac{z^3}{6}+\frac{z^4}{24}$	]–2.785, 0[

Les régions de stabilité correspondantes sont représentées dans la figure ci-contre (Fig. 6.21).

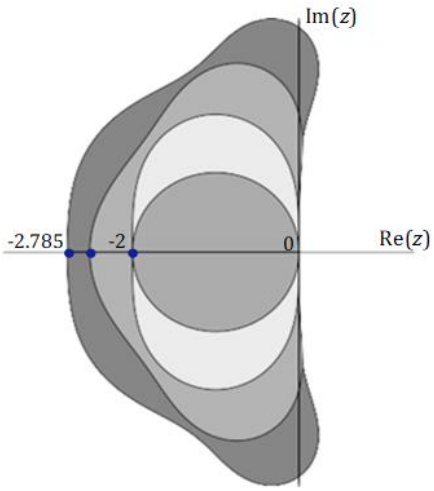


Fig. 6.21 : Régions de stabilité.

6. Méthodes multi-pas

Une amélioration de la méthode d'Euler a été envisagée encore plus tôt que les méthodes Runge-Kutta, les méthodes d'Adams. Ceux-ci ont été conçus par John Couch Adams afin de résoudre un problème de F. Bashforth, qui s'est produit dans une enquête sur l'action capillaire. Le problème et les schémas d'intégration numérique sont publiés dans Bashforth (1883). L'origine réelle de ces méthodes doit dater au moins en 1855, car cette année-là F. Bashforth fit une demande à la Royal Society pour l'aide de la subvention du gouvernement. Là, il a écrit: "*..., mais Je suis redevable à M. Adams pour une méthode de traitement de l'équation différentielle* :

$$\frac{\frac{d^2z}{du^2}}{\left(1+\left(\frac{dz}{du}\right)^2\right)^{3/2}}+\frac{\frac{1}{u}\frac{dz}{du}}{\left(1+\left(\frac{dz}{du}\right)^2\right)^{1/2}}-2\alpha z=\frac{2}{b}$$

une fois mis sous la forme,

$$\frac{b}{\rho}+\frac{b}{x}\sin\phi=2+2\alpha b^2\frac{z}{b}=2+\beta\frac{z}{b}$$

ce qui donne la forme théorique de la goutte avec une précision dépassant celle des mesures les plus raffinées".

Les méthodes de Runge-Kutta sont à un pas unique c'est-à-dire le problème de Cauchy est transformé en une relation sous la forme d'une fonction entre u_{n+1} et $(t_n,\ u_n)$, c'est-à-dire :

$$u_{n+1}=F(t_n,\ u_n)$$

Si le problème de Cauchy est transformé sous la forme d'une équation généralisée telle que :

$$u_{n+m}=-\sum_{i=0}^{m-1}\alpha_i u_{n+i}+h\sum_{i=0}^{m}\beta_i f_{n+i},\quad f_\eta\equiv f(t_\eta,\ u_\eta)$$

C'est La méthode de résolution de multi-pas à m étapes (Iserles, 2009), cette méthode est implicite si $\beta_m\neq 0$ et explicite dans le cas contraire. L'intérêt de ces méthodes vient du fait qu'on peut obtenir un ordre élevé pour une complexité de calcul nettement inférieure à celle des méthodes de Runge-Kutta. A partir du problème de Cauchy, soit :

$$u(t_{n+1})=u(t_{n-m})+\int_{t_{n-m}}^{t_{n+1}}f(t,\ u(t))dt$$

Qui peut être approchée par :

$$u_{n+1}=u_{n-m}+\int_{t_{n-m}}^{t_{n+1}}p(t)dt$$

Le principe de base des méthodes multi-pas est d'approcher $p(t)$ par un polynôme d'interpolation, et par conséquent le calcul de l'intégrale dans cette formule par une méthode approchée issue des méthodes de quadratures.

6.1. Méthodes d'Adams-Bashforth

Dans ces méthodes, la fonction $f\left(t,\ u(t)\right)$ est approchée par la formule d'interpolation de Gregory-Newton régressive des m points $\left(t_n,\ u_n\right)$, ..., $\left(t_{n-m+1},\ u_{n-m+1}\right)$, c'est-à-dire (Fig. 6.22) :

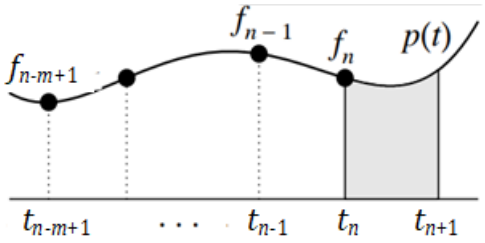


Fig. 6.22 : Méthodes d'Adams–Bashforth.

$$f\left(t,\ u(t)\right)\approx p(t)=\sum_{k=0}^{m-1}(-1)^k\binom{-s}{k}\nabla^k f\left(t_n,\ u_n\right)$$

soit,

$$u_{n+1}=u_n+\int_{t_n}^{t_{n+1}}\sum_{k=0}^{m-1}(-1)^k\binom{-s}{k}\nabla^k f\left(t_n,\ u_n\right)dt$$

ou,

$$u_{n+1}=u_n+\sum_{k=0}^{m-1}(-1)^k\nabla^k f\left(t_n,\ u_n\right)\int_{t_n}^{t_{n+1}}\binom{-s}{k}dt$$

sachant que, $t=t_n+sh\ \Rightarrow\ dt=hds$. Pour $t=t_n,\ s=0$ et pour $t=t_{n+1},\ s=1$ alors :

$$u_{n+1}=u_n+h\sum_{k=0}^{m-1}(-1)^k\nabla^k f\left(t_n,\ u_n\right)\int_0^1\binom{-s}{k}ds$$

ou,

$$u_{n+1}=u_n+h\sum_{k=0}^{m-1}\gamma_k\nabla^k f_n \text{ avec, } f_n\equiv f\left(t_n,\ u_n\right)$$

C'est la formule des méthodes d'Adams-Bashforth à m pas (ABm) pour la résolution du problème de Cauchy.

$$\nabla^k f_n=h^k k!\,f\left[\left(t_{n-k},\ u_{n-k}\right),\ \left(t_{n-k+1},\ u_{n-k+1}\right),\ \ldots,\ \left(t_n,\ u_n\right)\right]=\nabla\left(\nabla^{k-1}f_n\right)$$

et

$$\gamma_k=(-1)^k\int_0^1\binom{-s}{k}ds=\int_0^1\frac{s(s+1)(s+2)\cdots(s+k-1)}{k!}ds$$

γ_k sont les intégrales des coefficients binomiaux, elles peuvent être calculées facilement, a titre d'exemples :

$$\gamma_0=\int_0^1 ds=1,\quad \gamma_1=\int_0^1 sds=\frac{1}{2},\quad \gamma_2=\frac{1}{2}\int_0^1 s(s+1)ds=\frac{5}{12},\quad \gamma_3=\frac{1}{6}\int_0^1 s(s+1)(s+2)ds=\frac{3}{8}$$

La formule des méthodes d'Adams-Bashforth à m pas s'écrit,

$$u_{n+1}=u_n+h\sum_{k=0}^{m-1}b_{k+1}f\left(t_{n-k},\ u_{n-k}\right)$$

Pour $m=1$,

$$u_{n+1}=u_n+h\sum_{k=0}^{0}\gamma_k\nabla^k f_n=u_n+h\gamma_0 f_n=u_n+hf\left(t_n,\ u_n\right)$$

(AB1) c'est la méthode d'Euler.

Pour $m=2$,

$$u_{n+1}=u_n+h\sum_{k=0}^{1}\gamma_k\nabla^k f_n=u_n+h\left(\gamma_0 f_n+\gamma_1\nabla f_n\right)$$

$$u_{n+1} = u_n + h\left(\gamma_0 f_n + \gamma_1 f_n - \gamma_1 f_{n-1}\right) = u_n + h\left[\left(\gamma_0 + \gamma_1\right) f_n - \gamma_1 f_{n-1}\right]$$

$$u_{n+1} = u_n + h\left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1}\right) = u_n + \frac{h}{2}\left[3f\left(t_n,\ u_n\right) - f\left(t_{n-1},\ u_{n-1}\right)\right]$$

C'est la méthode d'*Adams-Bashforth à deux pas* (AB2). Cette méthode nécessite les deux premiers termes u_0 et u_1 pour résoudre le problème de Cauchy pour $n = 1,\ 2,\ \ldots,\ N-1$.

Pour $m = 3$,

$$u_{n+1} = u_n + h\sum_{k=0}^{2} \gamma_k \nabla^k f_n = u_n + h\left(\gamma_0 f_n + \gamma_1 \nabla f_n + \gamma_2 \nabla^2 f_n\right)$$

$$u_{n+1} = u_n + h\left[\gamma_0 f_n + \gamma_1 f_n - \gamma_1 f_{n-1} + \gamma_2 \nabla\left(\nabla f_n\right)\right] = u_n + h\left[\gamma_0 f_n + \gamma_1 f_n - \gamma_1 f_{n-1} + \gamma_2 \nabla f_n - \gamma_2 \nabla f_{n-1}\right]$$

$$u_{n+1} = u_n + \left[\gamma_0 f_n + \gamma_1 f_n - \gamma_1 f_{n-1} + \gamma_2 f_n - 2\gamma_2 f_{n-1} + \gamma_2 f_{n-2}\right]$$

$$u_{n+1} = u_n + h\left[\left(\gamma_0 + \gamma_1 + \gamma_2\right) f_n - \left(\gamma_1 + 2\gamma_2\right) f_{n-1} + \gamma_2 f_{n-2}\right]$$

$$u_{n+1} = u_n + h\left[\left(1 + \frac{1}{2} + \frac{5}{12}\right) f_n - \left(\frac{1}{2} + \frac{5}{6}\right) f_{n-1} + \frac{5}{12} f_{n-2}\right] = u_n + h\left[\frac{23}{12} f_n - \frac{4}{3} f_{n-1} + \frac{5}{12} f_{n-2}\right]$$

c'est-à-dire :

$$u_{n+1} = u_n + \frac{h}{12}\left[23f\left(t_n,\ u_n\right) - 16f\left(t_{n-1},\ u_{n-1}\right) + 5f\left(t_{n-2},\ u_{n-2}\right)\right]$$

C'est la méthode d'*Adams-Bashforth à trois pas* (AB3). Cette méthode nécessite les trois premiers termes u_0, u_1 et u_2 pour résoudre le problème de Cauchy pour $n = 2,\ 3,\ \ldots,\ N-1$.

Pour $m = 4$,

$$u_{n+1} = u_n + h\sum_{k=0}^{3} \gamma_k \nabla^k f_n = u_n + h\left(\gamma_0 f_n + \gamma_1 \nabla f_n + \gamma_2 \nabla^2 f_n + \gamma_3 \nabla^3 f_n\right)$$

or,

$$\nabla f_n = f_n - f_{n-1},\ \nabla^2 f_n = \nabla f_n - \nabla f_{n-1} = f_n - 2f_{n-1} + f_{n-2}$$

$$\begin{aligned}\nabla^3 f_n = \nabla\left(\nabla^2 f_n\right) &= \nabla f_n - 2\nabla f_{n-1} + \nabla f_{n-2} \\ &= f_n - f_{n-1} - 2\left(f_{n-1} - f_{n-2}\right) + f_{n-2} - f_{n-3} \\ &= f_n - 3f_{n-1} + 3f_{n-2} - f_{n-3}\end{aligned}$$

soit,

$$\begin{aligned}\gamma_0 f_n + \gamma_1 \nabla f_n + \gamma_2 \nabla^2 f_n + \gamma_3 \nabla^3 f_n &= \left(\gamma_0 + \gamma_1 + \gamma_2 + \gamma_3\right) f_n - \left(\gamma_1 + 2\gamma_2 + 3\gamma_3\right) f_{n-1} \\ &\quad + \left(\gamma_2 + 3\gamma_3\right) f_{n-2} - \gamma_3 f_{n-3} \\ &= \left(1 + \frac{1}{2} + \frac{5}{12} + \frac{3}{8}\right) f_n - \left(\frac{1}{2} + 2\frac{5}{12} + 3\frac{3}{8}\right) f_{n-1} + \left(\frac{5}{12} + 3\frac{3}{8}\right) f_{n-2} - \frac{3}{8} f_{n-3}\end{aligned}$$

alors,

$$u_{n+1} = u_n + \frac{h}{24}\left[55f\left(t_n,\ u_n\right) - 59f\left(t_{n-1},\ u_{n-1}\right) + 37f\left(t_{n-2},\ u_{n-2}\right) - 9f\left(t_{n-3},\ u_{n-3}\right)\right]$$

C'est la méthode d'*Adams-Bashforth à quatre pas* (AB4). Cette méthode nécessite les quatre premiers termes u_0, u_1, u_2 et u_3 pour résoudre le problème de Cauchy pour $n = 3,\ 4,\ \ldots,\ N-1$.

Les méthodes d'Adams-Bashforth à d'ordre plus élevé nécessitent en plus de la première condition u_0, les autres conditions u_1, u_2 et u_3 comme dans la méthode (AB4) qui correspond respectivement à t_1, t_2 et t_3. Dans la pratique, ces valeurs peuvent être déterminé avec la méthode de Runge-Kutta d'ordre 4 (RK4) puisque les deux méthodes (AB4) et (RK4) ont le même ordre d'erreur de troncature local $O\left(h^5\right)$ (Kharab et Guenther, 2019).

Soit, la fonction **AB4** permet la solution numérique du problème de Cauchy avec la méthode d'Adams-Bashforth à 4 étapes (AB4) :

```
function [t,u]= AB4(f,a,b,u0,h,varargin)
t=a:h:b; u(1)=u0;
for n=1:3
 k1=f(t(n),u(n));
 k2=f(t(n)+(h/2),u(n)+h*(k1/2));
 k3=f(t(n)+(h/2),u(n)+h*(k2/2));
 k4=f(t(n+1),u(n)+h*k3);
 u(n+1)=u(n)+(h/6)*(k1+2*k2+2*k3+k4);
end
for n=4:numel(t)-1
 K1=f(t(n),u(n));K2=f(t(n-1),u(n-1));
 K3=f(t(n-2),u(n-2)); K4=f(t(n-3),u(n-3));
 u(n+1)=u(n)+(h/24)*(55*K1-59*K2+37*K3-9*K4);
end
 plot(t,u,'-',t,u,'or')
end
```

A titre d'exemple soit le problème :

$$\frac{du}{dt}=3-2u+e^{-t} \quad \text{pour} \quad 0\le t\le 5 \quad \text{et} \quad u\left(0\right)=1$$

L'application de cette fonction conduit aux résultats (Fig. 6.23).

```
>> f = @(t,u) 3 - 2*u + exp(-t);
>> AB4(f,0,5,1,0.1);
```

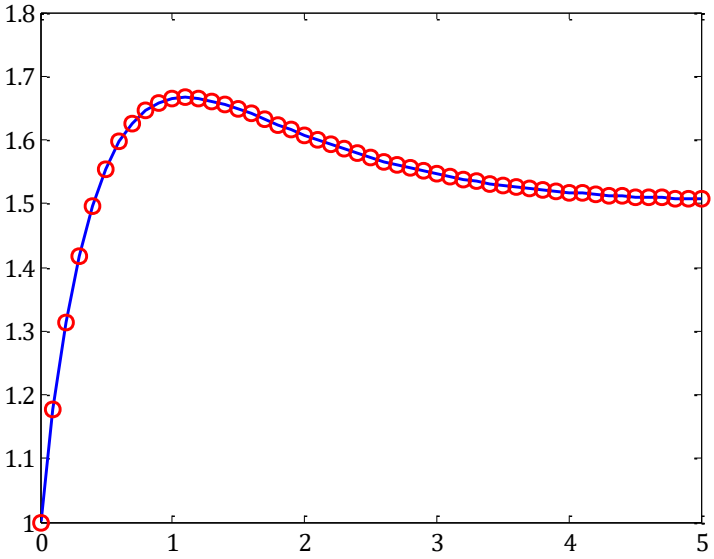


Fig. 6.23 : Exécution des fonctions **AB4** et **AM4**.

6.2. Méthodes d'Adams-Moulton

Si $f\left(t,\ u(t)\right)$ est approchée par la formule d'interpolation de Gregory-Newton régressive des *m* points $\left(t_{n+1},\ u_{n+1}\right)$, …, $\left(t_{n-m+2},\ u_{n-m+2}\right)$, c'est-à-dire :

$$\int_{t_n}^{t_{n+1}} p(t)dt = h\sum_{k=0}^{m-1}(-1)^k \nabla^k f\left(t_{n+1},\ u_{n+1}\right)\int_0^1 \binom{1-s}{k} ds = h\sum_{k=0}^{m-1}\gamma'_k \nabla^k f_{n+1}$$

Soit la formule d'Adams-Moulton à *m* pas,

$$u_{n+1} = u_n + h\sum_{k=0}^{m-1}\gamma'_k \nabla^k f_{n+1}$$

γ'_k est exprimée par :

$$\gamma'_k = (-1)^k \int_0^1 \binom{1-s}{k} ds$$

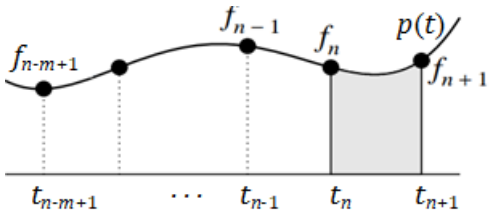


Fig. 6.24 : Méthodes d'Adams–Moulton.

Par exemple,

$$\gamma'_0 = \int_0^1 ds = 1,\quad \gamma'_1 = \int_0^1 (1-s)ds = -\frac{1}{2}\quad \gamma'_2 = \int_0^1 (-s)(1-s)ds = -\frac{1}{12},\ \gamma'_3 = \int_0^1 (-s)(1-s)(-s-1)ds = -\frac{1}{24}$$

Pour *m* = 1,

$$u_{n+1} = u_n + h\gamma'_0 f_{n+1}$$

ou,

$$u_{n+1} = u_n + hf\left(t_{n+1},\ u_{n+1}\right)$$

C'est la méthode Adams-Moulton à un pas (AM1) qui est la formule d'Euler implicite.

Pour *m* = 2,

$$u_{n+1} = u_n + h\left(\gamma'_0 f_{n+1} + \gamma'_1 \nabla f_{n+1}\right)$$

or,

$$\gamma'_0 f_{n+1} + \gamma'_1 \nabla f_{n+1} = \gamma'_0 f_{n+1} + \gamma'_1 f_{n+1} - \gamma'_1 f_n$$

soit,

$$u_{n+1} = u_n + h\left[\left(\gamma'_0 + \gamma'_1\right)f_{n+1} - \gamma'_1 f_n\right]$$

ou,

$$u_{n+1} = u_n + \frac{h}{2}\left[f\left(t_{n+1},\ u_{n+1}\right) + f\left(t_n,\ u_n\right)\right]$$

C'est la méthode Adams-Moulton à deux pas (AM2) qui est le schéma d'Euler-centré ou la méthode de Heun.

Pour *m* = 4,

$$u_{n+1} = u_n + h\left[\gamma'_0 f_{n+1} + \gamma'_1 \nabla f_{n+1} + \gamma'_2 \nabla^2 f_{n+1} + \gamma'_3 \nabla^3 f_{n+1}\right]$$

or,

$$\nabla f_{n+1} = f_{n+1} - f_n,\ \nabla^2 f_{n+1} = \nabla f_{n+1} - \nabla f_n = f_{n+1} - 2f_n + f_{n-1}$$

alors,

$$\nabla^3 f_n = \nabla\left(\nabla^2 f_n\right) = \nabla f_{n+1} - 2\nabla f_n + \nabla f_{n-1} = f_{n+1} - f_n - 2\left(f_n - f_{n-1}\right) + f_{n-1} - f_{n+2} = f_{n+1} - 3f_n + 3f_{n+1} - f_{n+2}$$

soit,

$$\gamma'_0 f_{n+1} + \gamma'_1 \nabla f_{n+1} + \gamma'_2 \nabla^2 f_{n+1} + \gamma'_3 \nabla^3 f_{n+1} = \left(\gamma'_0 + \gamma'_1 + \gamma'_2 + \gamma'_3\right)f_{n+1} - \left(\gamma'_1 + 2\gamma'_2 + 3\gamma'_3\right)f_n + \left(\gamma'_2 + 3\gamma'_3\right)f_{n-1} - \gamma'_3 f_{n-2}$$

donc,

$$u_{n+1}=u_n+\frac{h}{24}\left[9f\left(t_{n+1},\ u_{n+1}\right)+19f\left(t_n,\ u_n\right)-5f\left(t_{n-1},\ u_{n-1}\right)+f\left(t_{n-2},\ u_{n-2}\right)\right]$$

C'est la méthode d'Adams-Moulton à quatre pas (AM4).

6.3. Méthodologie de la prédiction-correction

Les méthodes implicites comme les méthodes d'Adams-Moulton ne sont pas utilisées comme telles. Elles sont plus couteuses car le caractère implicite doit être résolu par une méthode itérative comme du point fixe, Newton-Raphson etc. D'où l'idée, pour diminuer le coût, d'utiliser une méthode explicite pour calculer un bon prédicteur de la solution et de ne faire qu'une itération (ou quelques itérations) de la méthode implicite utilisée comme correcteur explicite.

Il existe de nombreuses variantes des méthodes prédicteur-correcteur. Le prédicteur $\tilde{u}$ est généralement explicite c'est-à-dire :

$$\tilde{u}_{n+1}=-\sum_{i=0}^{m-1}\tilde{\alpha}_i u_{n+i-m+1}+h\sum_{i=0}^{m-1}\tilde{\beta}_i f\left(t_{n+i-m+1},\ u_{n+i-m+1}\right)$$

et le correcteur qui peut être sous la forme :

$$u_{n+1}=-\sum_{i=0}^{m-1}\alpha_i u_{n+i-m+1}+h\sum_{i=0}^{m-1}\beta_i f\left(t_{n+i-m+1},\ u_{n+i-m+1}\right)+h\beta_m f\left(t_{n+1},\ \tilde{u}_{n+1}\right)$$

ou sous la forme,

$$u_{n+1}=-\sum_{i=0}^{m-1}\alpha_i u_{n+i-m+1}+h\sum_{i=0}^{m-1}\beta_i f\left(t_{n+i-m+1},\ \tilde{u}_{n+i-m+1}\right)+h\beta_m f\left(t_{n+1},\ \tilde{u}_{n+1}\right)$$

La contribution de l'erreur du prédicteur est d'un ordre inférieur à celle du correcteur. Pour avoir une méthode d'ordre m on pourra prendre un prédicteur d'ordre $m-1$ et un correcteur d'ordre m.

Pour les méthodes d'Adams, on utilise plutôt conjointement une méthode d'Adams-Bashforth (explicite) et une méthode d'Adams-Moulton (implicite) pour construire un schéma prédicteur-correcteur. Le schéma le plus utilisé est composé de la méthode (AB3) pour la prédiction et de la méthode (AM4) pour la correction :

$$\text{Prédicteur :}\quad \tilde{u}_{n+1}=u_n+\frac{h}{12}\left[23f\left(t_n,\ u_n\right)-16f\left(t_{n-1},\ u_{n-1}\right)+5f\left(t_{n-2},\ u_{n-2}\right)\right]$$

$$\text{Correcteur :}\quad u_{n+1}=u_n+\frac{h}{24}\left[9f\left(t_{n+1},\ \tilde{u}_{n+1}\right)+19f\left(t_n,\ u_n\right)-5f\left(t_{n-1},\ u_{n-1}\right)+f\left(t_{n-2},\ u_{n-2}\right)\right]$$

Soit la fonction **AM4** qui permet la solution numérique du problème de Cauchy de premier ordre avec la méthode d'Adams-Bashforth (AB4) comme prédicteur et la méthode d'Adam-Moulton (AM4) comme correcteur :

```
function [t,u]= AM4(f,a,b,u0,h,varargin)
t=a:h:b; u(1)=u0; t=a:h:b;
for n=1:2
 k1=f(t(n),u(n)); k2=f(t(n)+(h/2),u(n)+h*(k1/2));
```

```
 k3=f(t(n)+ h,u(n)-h*k1+2*h*k2);
 u(n+1)=u(n)+(h/6)*(k1+4*k2+k3);
end
for n=3:numel(t)-1
 K1=f(t(n),u(n)); K2=f(t(n-1),u(n-1)); K3=f(t(n-2),u(n-2));
 K4=f(t(n-3),u(n-3));
 up(n+1)=u(n)+(h/24)*(55*K1-59*K2+37*K3-9*K4);
 u(n+1)=u(n)+(h/24)*(9*f(t(n)+h,up(n+1))+19*K1-5*K2+K3);
end
plot(t,u,'-',t,u,'or')
end
```

L'application de cette fonction au problème précédent, conduit aux mêmes résultats obtenus par la fonction **AB4** résultats (Fig. 6.13).

```
>> f = @(t,u) 3 - 2*u + exp(-t);
>> AM4(f,0,5,1,0.1);
```

La méthode d'Adams-Moulton (AM4) est très utile pour le calcul de l'intégrale définie :

$$I=\int_a^b f(t)dt$$

Ainsi le problème d'intégration est formé comme problème de Cauchy :

$$\frac{du}{dt}=f(t)\ \Rightarrow\ u(t_{n+1})=u(t_n)+\int_{t_n}^{t_{n+1}} f(t)dt\ \text{ et }\ u(a)\equiv u_0=\int_a^a f(t)dt=0$$

D'après la méthode d'Adams-Moulton (AM4), le processus d'approximation du calcul de l'intégrale s'écrit :

$$u_{n+1}=u_n+\frac{h}{24}\left[9f(t_{n+1})+19f(t_n)-5f(t_{n-1})+f(t_{n-2})\right],\ \text{avec}\ \ h=\frac{b-a}{N}$$

alors,

$$\int_a^b f(t)dt=u_N-\frac{19}{720}h^5 f^{(5)}(\xi),\ \ \xi\in[a,\ b]$$

C'est-à-dire que u_N est une approximation d'ordre 5. Le processus de calcul nécessite en plus de u_0, u_1 et u_2. Les deux dernières valeurs peuvent être calculées par une des méthodes de Runge-Kutta (RK4), par exemple,

$$u_{n+1}=u_n+\frac{h}{6}\left[f(t_n)+4f\left(t_n+\frac{h}{2}\right)+f(t_n+h)\right],\ \ u_0\equiv u(t_0)=u(a)=0$$

remarquons que,

$$u_1=\frac{h}{6}\left[f(a)+4f\left(a+\frac{h}{2}\right)+f(a+h)\right]\approx\int_a^{a+h} f(s)ds$$

u_1 est le calcul approché par la méthode de Simpson 1/3 de l'intégrale de la fonction $f(t)$ sur l'intervalle $[a,\ a+h]$.

$$u_2=u_1+\frac{h}{6}\left[f(a+h)+4f\left(a+\frac{3h}{2}\right)+f(a+2h)\right]$$

L'intégrale I est explicité ainsi par,

$$u_N = u_{N-1} + \frac{h}{24}\left[9f\left(a+Nh\right)+19f\left(a+\left(N-1\right)h\right)-5f\left(a+\left(N-2\right)h\right)+f\left(a+\left(N-3\right)h\right)\right]$$

Soit la fonction **`IntegralByAM4`** qui utilise cette procédure pour calculer l'intégrale I de la fonction $f(t)$ définie et continue dans l'intervalle $[a,\ b]$.

```
function I = IntegralByAM4(f,a,b,h)
if nargin<4, h=0.01; end
t=a:h:b;
u(1)=0;
for n=1:2
 u(n+1)=u(n)+(h/6)*(f(t(n))+4*f(t(n)+(h/2))+f(t(n+1)));
end
for n=3:numel(t)-1
 u(n+1)=u(n)+(h/24)*(9*f(t(n+1))+19*f(t(n))-5*f(t(n-1))+f(t(n-2)));
end
I=u(numel(t));
end
```

Soit l'exemple :

$$\int_0^4 e^{-2x}\sin 3x dx$$

L'application de la fonction **`IntegralByAM4`** conduit à :

```
>> f=@(x)exp(-2*x).*sin(3*x);
>> I=IntegralByAM4(f,0,4)
I =
    0.2307
```

6.4. Méthodes explicites de Nyström

Dans son article de synthèse sur l'intégration numérique des équations différentielles Nyström (1925) présente une classe de méthodes multi-pas. Il considère l'équation intégrale du problème de Cauchy comme (Fig. 6.25) :

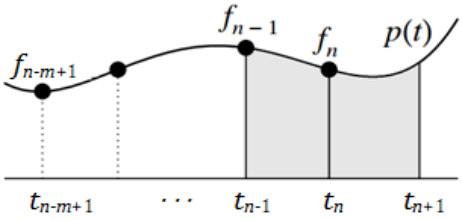


Fig. 6.25 : Méthodes explicites de Nyström.

$$u\left(t_{n+1}\right) = u\left(t_{n-1}\right) + \int_{t_{n-1}}^{t_{n+1}} f\left(t,\ u(t)\right)dt$$

Comme dans la méthode d'Adams-Bashforth, l'expression précédente peut être approché par :

$$u_{n+1} = u_{n-1} + \int_{t_{n-1}}^{t_{n+1}} \sum_{k=0}^{m-1} (-1)^k \binom{-s}{k} \nabla^k f\left(t_n,\ u_n\right)dt = u_{n-1} + h\sum_{k=0}^{m-1} \kappa_k \nabla^k f_n$$

avec,

$$\kappa_k = (-1)^k \int_{-1}^{1} \binom{-s}{k} ds = \int_{-1}^{1} \frac{s(s+1)(s+2)\cdots(s+k-1)}{k!} ds$$

par exemple,

$$\kappa_0 = 2,\ \kappa_1 = 0,\ \kappa_2 = \kappa_3 = \frac{1}{3},\ \text{et}\ \kappa_4 = \frac{29}{90}$$

La formule des méthodes de Nyström à *m* pas s'écrit,

$$u_{n+1} = u_{n-1} + h\sum_{k=0}^{m-1} \beta_{k+1} f\left(t_{n-k},\ u_{n-k}\right)$$

Pour *m* = 1,

$$u_{n+1} = u_{n-1} + 2hf_n = u_{n-1} + 2hf\left(t_n,\ u_n\right)$$

c'est la méthode d'Euler centrale, c'est la simple méthode de second ordre.

Pour *m* = 2,

$$u_{n+1} = u_{n-1} + h\sum_{k=0}^{1} \kappa_k \nabla^k f_n = u_{n-1} + h\left(\kappa_0 f_n + \kappa_1 \nabla f_n\right)$$

$$u_{n+1} = u_{n-1} + h\left(\kappa_0 f_n + \kappa_1 f_n - \kappa_1 f_{n-1}\right) = u_{n-1} + h\left[\left(\kappa_0 + \kappa_1\right) f_n - \kappa_1 f_{n-1}\right] = u_{n-1} + 2hf_n$$

C'est la même méthode que la précédente quand *m* = 1.

Pour *m* = 3,

$$u_{n+1} = u_{n-1} + h\sum_{k=0}^{2} \kappa_k \nabla^k f_n = u_{n-1} + h\left(\kappa_0 f_n + \kappa_1 \nabla f_n + \kappa_2 \nabla^2 f_n\right)$$

$$u_{n+1} = u_{n-1} + h\left[\kappa_0 f_n + \kappa_1 f_n - \kappa_1 f_{n-1} + \kappa_2 \nabla\left(\nabla f_n\right)\right] = u_{n-1} + h\left[\kappa_0 f_n + \kappa_1 f_n - \kappa_1 f_{n-1} + \kappa_2 \nabla f_n - \kappa_2 \nabla f_{n-1}\right]$$

$$u_{n+1} = u_{n-1} + \left[\kappa_0 f_n + \kappa_1 f_n - \kappa_1 f_{n-1} + \kappa_2 f_n - 2\kappa_2 f_{n-1} + \kappa_2 f_{n-2}\right]$$

$$u_{n+1} = u_{n-1} + h\left[\left(\kappa_0 + \kappa_1 + \kappa_2\right) f_n - \left(\kappa_1 + 2\kappa_2\right) f_{n-1} + \kappa_2 f_{n-2}\right]$$

$$u_{n+1} = u_{n-1} + \frac{h}{3}\left(7f_n - 2f_{n-1} + f_{n-2}\right)$$

Pour *m* = 4,

$$u_{n+1} = u_{n-1} + h\sum_{k=0}^{3} \kappa_k \nabla^k f_n = u_{n-1} + h\left(\kappa_0 f_n + \kappa_1 \nabla f_n + \kappa_2 \nabla^2 f_n + \kappa_3 \nabla^3 f_n\right)$$

c'est-à-dire,

$$u_{n+1} = u_{n-1} + h\left[\left(\kappa_0 + \kappa_1 + \kappa_2 + \kappa_3\right) f_n - \left(\kappa_1 + 2\kappa_2 + 3\kappa_3\right) f_{n-1} + \left(\kappa_2 + 3\kappa_3\right) f_{n-2} - \kappa_3 f_{n-3}\right]$$

soit,

$$u_{n+1} = u_{n-1} + \frac{h}{3}\left(8f_n - 5f_{n-1} + 4f_{n-2} - f_{n-3}\right)$$

La méthode de Nyström d'ordre 4 nécessite en plus de la première condition u_0, les termes u_1, u_2 et u_3, qui peuvent être calculés par la méthode de Runge-Kutta d'ordre 4 (RK4).

6.5. Méthodes de Milne–Simpson

Comme dans la méthode d'Adams-Moulton, l'intégrale :

$$\int_{t_{n-1}}^{t_{n+1}} f\left(t,\ u(t)\right)dt$$

est approchée par l'intégrale (Fig. 6.26) :

$$\int_{t_{n-1}}^{t_{n+1}} \sum_{k=0}^{m} (-1)^k \binom{-s+1}{k} \nabla^k f\left(t_{n+1},\ u_{n+1}\right)dt = h\sum_{k=0}^{m} \kappa'_k \nabla^k f_{n+1} \text{ avec, } \kappa'_k = (-1)^k \int_{-1}^{1} \binom{-s+1}{k} ds$$

par exemple,

$$\kappa_0' = 2\ \ \kappa_1' = -2,\ \ \kappa_2' = \frac{1}{3},\ \ \kappa_3' = 0 \text{ et } \kappa_4' = -\frac{1}{90}$$

La formule des méthodes de Milne-Simpson à m pas s'écrit,

$$u_{n+1} = u_{n-1} + h\sum_{k=0}^{m} \kappa_k' \nabla^k f_{n+1}$$

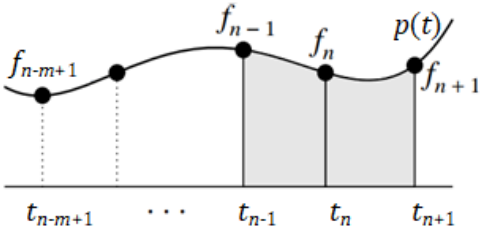


Fig. 6.26 : Méthodes de Milne-Simpson.

Pour $m = 0$,

$$u_{n+1} = u_{n-1} + 2hf_{n+1} = u_{n-1} + 2hf\left(t_{n+1},\ u_{n+1}\right)$$

C'est la méthode d'Euler implicite avec un pas de $2h$.

Pour $m = 1$,

$$u_{n+1} = u_{n-1} + h\sum_{k=0}^{1} \kappa_k' \nabla^k f_{n+1} = u_{n-1} + h\left(\kappa_0' f_{n+1} + \kappa_1' \nabla f_{n+1}\right) = u_{n-1} + 2hf_n$$

C'est la méthode d'Euler centrale.

Pour $m = 2$,

$$u_{n+1} = u_{n-1} + h\sum_{k=0}^{2} \kappa_k' \nabla^k f_{n+1} = u_{n-1} + h\left(\kappa_0' f_{n+1} + \kappa_1' \nabla f_{n+1} + \kappa_2' \nabla^2 f_{n+1}\right)$$

c'est-à dire,

$$u_{n+1} = u_{n-1} + \frac{h}{3}\left(f_{n+1} + 4f_n + f_{n-1}\right)$$

C'est la méthode de Milne, très intéressante (Milne, 1926) qui est une généralisation directe de la règle de Simpson.

Pour $m = 4$,

$$u_{n+1} = u_{n-1} + h\sum_{k=0}^{4} \kappa_k' \nabla^k f_{n+1} = u_{n-1} + h\left(\kappa_0' f_{n+1} + \kappa_1' \nabla f_{n+1} + \kappa_2' \nabla^2 f_{n+1} + \kappa_3' \nabla^3 f_{n+1} + \kappa_4' \nabla^4 f_{n+1}\right)$$

c'est-à dire,

$$u_{n+1} = u_{n-1} + \frac{h}{90}\left(29f_{n+1} + 124f_n + 24f_{n-1} + 4f_{n-2} - f_{n-3}\right)$$

De nombreuses autres méthodes similaires ont été étudiées. Ils sont tous basés sur un équation intégrale de la forme :

$$u\left(t_{n+1}\right) = u\left(t_{n-l}\right) + \int_{t_{n-l}}^{t_{n+1}} f\left(t,\ u(t)\right)dt$$

Qui peut être approché par :

$$u_{n+1} = u_{n-l} + \int_{t_{n-l}}^{t_{n+1}} \sum_{k=0}^{m-1} (-1)^k \binom{-s}{k} \nabla^k f\left(t_n,\ u_n\right)dt$$

En particulier si $l = 3$,

$$u_{n+1} = u_{n-3} + \frac{4h}{3}\left(2f_n - f_{n-1} + 2f_{n-2}\right)$$

Cette méthode est utilisée par Milne (1926) comme prédicteur avec l'une des méthodes implicites précédentes comme correcteurs.

6.6. Formules de différentiation rétrograde BDF

Les formules de différentiation rétrograde (notées BDF pour *Backward Differentiation Formulae*) sont des méthodes multi-pas implicites obtenues par une approche complémentaire de celle suivie pour les méthodes d'Adams. Pour ces dernières, on a effectué une intégration numérique de la fonction source $f\left(t,\ u(t)\right)$, tandis que dans les méthodes BDF on approche directement la valeur de la dérivée première de u au nœud t_{n+1} par la dérivée première du polynôme interpolant u aux $(m+1)$ nœuds t_{n+1}, ,t_{n+1}, …, t_{n-m+1}. Le schéma général de la méthode BDF d'ordre m (Cryer, 1972) est:

$$\sum_{k=1}^{m}\frac{1}{k}\nabla^k u_{n+1}=hf\left(t_{n+1},\ u_{n+1}\right)$$

Qui peut être transformé sous la forme :

$$u_{n+1}=-\sum_{i=0}^{m-1}\alpha_i u_{n+i-m+1}+h\beta_m f\left(t_{n+1},\ u_{n+1}\right)$$

A titre d'exemple, la méthode BDF3 correspond à :

$$\sum_{k=1}^{3}\frac{1}{k}\nabla^k u_{n+1}=hf\left(t_{n+1},\ u_{n+1}\right)$$

c'est-à-dire,

$$\nabla u_{n+1}+\frac{1}{2}\nabla^2 u_{n+1}+\frac{1}{3}\nabla^3 u_{n+1}=hf\left(t_{n+1},\ u_{n+1}\right)$$

or,

$$\nabla u_{n+1}=u_{n+1}-u_n,\ \nabla^2 u_{n+1}=\nabla\left(\nabla u_{n+1}\right)=\nabla u_{n+1}-\nabla u_n=u_{n+1}-2u_n+u_{n-1}$$

et

$$\nabla^3 u_{n+1}=\nabla\left(\nabla^2 u_{n+1}\right)=\nabla\left(u_{n+1}-2u_n+u_{n-1}\right)=u_{n+1}-3u_n+3u_{n-1}-u_{n-2}$$

on obtient après remplacement,

$$\frac{11}{6}u_{n+1}-3u_n+\frac{3}{2}u_{n-1}-\frac{1}{3}u_{n-2}=hf\left(t_{n+1},\ u_{n+1}\right)$$

ou,

$$u_{n+1}=\frac{18}{11}u_n-\frac{9}{11}u_{n-1}+\frac{2}{11}u_{n-2}+\frac{6}{11}hf\left(t_{n+1},\ u_{n+1}\right)$$

On peut avoir la même formule de BDF3 à partir du l'expression :

$$u_{n+1}=-\sum_{i=0}^{2}\alpha_i u_{n+i-2}+h\beta_3 f\left(t_{n+1}\ u_{n+1}\right)=-\left(\alpha_0 u_{n-2}+\alpha_1 u_{n-1}+\alpha_2 u_n\right)+h\beta_3 f\left(t_{n+1},\ u_{n+1}\right)$$

sachant que,

$$u\left(t+\nu h\right)\approx u_{n+\nu}\ \text{ et }\ u'\left(t+h\right)\approx f\left(t_{n+1},\ u_{n+1}\right)$$

D'après le développement de Taylor d'ordre 4 de la fonction $u\left(t+\nu h\right)$:

$$u\left(t+\nu h\right)=u(t)+\nu h u'(t)+\frac{\nu^2 h^2}{2}u''(t)+\frac{\nu^3 h^3}{6}u'''(t)+\frac{\nu^4 h^4}{24}u^{(4)}(t)+O\left(h^5\right)$$

alors,

$$u\left(t+h\right)=u(t)+h u'(t)+\frac{h^2}{2}u''(t)+\frac{h^3}{6}u'''(t)+\frac{h^4}{24}u^{(4)}(t)+O\left(h^5\right)$$

$$u(t-2h)=u(t)-2hu'(t)+2h^2u''(t)-\frac{4h^3}{3}u'''(t)+\frac{2h^4}{3}u^{(4)}(t)+O(h^5)$$

$$u(t-h)=u(t)-hu'(t)+\frac{h^2}{2}u''(t)-\frac{h^3}{6}u'''(t)+\frac{h^4}{24}u^{(4)}(t)+O(h^5)$$

et

$$hu'(t+h)=hu'(t)+h^2u''(t)+\frac{h^3}{2}u'''(t)+\frac{h^4}{6}u^{(4)}(t)+O(h^5)$$

alors,

$$-\alpha_0u(t-2h)-\alpha_1u(t-h)-\alpha_2u(t)+h\beta_3u'(t+h)=-(\alpha_0+\alpha_1+\alpha_2)u(t)+$$
$$(2\alpha_0+\alpha_1+\beta_3)hu'(t)+\left(-2\alpha_0-\frac{1}{2}\alpha_1+\beta_3\right)h^2u''(t)+\left(\frac{4}{3}\alpha_0+\frac{1}{6}\alpha_1+\frac{1}{2}\beta_3\right)h^3u'''(t)+$$
$$\left(-\frac{2}{3}\alpha_0-\frac{1}{24}\alpha_1+\frac{1}{6}\beta_3\right)h^4u^{(4)}(t)+O(h^5)$$

puisque,

$$u(t+h)=-\alpha_0u(t-2h)-\alpha_1u(t-h)-\alpha_2u(t)+h\beta_3u'(t+h)$$

et par identification avec le développement de Taylor de $u(t+h)$ $u(t+h)$ et pour que la différence entre $u(t+h)$ et $-\alpha_0u(t-2h)-\alpha_1u(t-h)-\alpha_2u(t)+h\beta_3u'(t+h)$ soit de même ordre c'est-à-dire $O(h^5)$ il faut que :

$$\begin{cases}-\alpha_0-\alpha_1-\alpha_2=1\\ 2\alpha_0+\alpha_1+\beta_3=1\\ -2\alpha_0-\frac{1}{2}\alpha_1+\beta_3=\frac{1}{2}\\ \frac{4}{3}\alpha_0+\frac{1}{6}\alpha_1+\frac{1}{2}\beta_3=\frac{1}{6}\end{cases}\Rightarrow\begin{cases}\alpha_0=-\frac{2}{11}\\ \alpha_1=\frac{9}{11}\\ \alpha_2=-\frac{18}{11}\\ \beta_3=\frac{6}{11}\end{cases}$$

Soit donc l'expression de BDF3 :

$$u_{n+1}=\frac{18}{11}u_n-\frac{9}{11}u_{n-1}+\frac{2}{11}u_{n-2}+\frac{6}{11}hf(t_{n+1},\ u_{n+1})$$

On déduit aussi la différence :

$$u(t+h)+\alpha_0u(t-2h)+\alpha_1u(t-h)+\alpha_2u(t)-h\beta_3u'(t+h)=Ch^4u^{(4)}(t)+O(h^5)$$

avec,

$$C=\frac{1}{24}+\frac{2}{3}\alpha_0+\frac{1}{24}\alpha_1-\frac{1}{6}\beta_3=\frac{1}{24}+\frac{2}{3}\left(-\frac{2}{11}\right)+\frac{1}{24}\left(\frac{9}{11}\right)-\frac{1}{6}\left(\frac{6}{11}\right)=-\frac{3}{22}$$

soit :

$$u(t+h)+\alpha_0u(t-2h)+\alpha_1u(t-h)+\alpha_2u(t)-h\beta_3u'(t+h)=-\frac{3}{22}h^4u^{(4)}(t)+O(h^5)$$

C'est l'erreur de troncature locale de BDF3 elle est de l'ordre 4

Le tableau ci-dessous (Tab. 6.2) donne les méthodes BDF les plus utilisées.

Tab. 6.2 : Méthodes BDF pour m = 1, 2, 3, 4, 5 et 6.

m	**Méthode BDF*m***
1	$u_{n+1} = u_n + hf\left(t_{n+1},\ u_{n+1}\right)$
2	$u_{n+1} = \frac{4}{3}u_n - \frac{1}{3}u_{n-1} + \frac{2}{3}hf\left(t_{n+1},\ u_{n+1}\right)$
3	$u_{n+1} = \frac{18}{11}u_n - \frac{9}{11}u_{n-1} + \frac{2}{11}u_{n-2} + \frac{6}{11}hf\left(t_{n+1},\ u_{n+1}\right)$
4	$u_{n+1} = \frac{48}{25}u_n - \frac{36}{25}u_{n-1} + \frac{16}{25}u_{n-2} - \frac{3}{25}u_{n-3} + \frac{12}{25}hf\left(t_{n+1},\ u_{n+1}\right)$
5	$u_{n+1} = \frac{300}{137}u_n - \frac{300}{137}u_{n-1} + \frac{200}{137}u_{n-2} - \frac{75}{137}u_{n-3} + \frac{12}{137}u_{n-4} + \frac{60}{137}hf\left(t_{n+1}\ u_{n+1}\right)$
6	$u_{n+1} = \frac{360}{147}u_n - \frac{450}{147}u_{n-1} + \frac{400}{147}u_{n-2} - \frac{225}{147}u_{n-3} + \frac{72}{147}u_{n-4} - \frac{10}{147}u_{n-3} + \frac{60}{147}hf\left(t_{n+1}\ u_{n+1}\right)$

7. Consistance, convergence et stabilité

Pour étudier la consistance, soit l'opérateur linéaire obtenue de la formule générale de la méthode multi-pas d'ordre m :

$$L\left[u(t)\right] = u(t+mh) + \sum_{i=0}^{m-1}\alpha_i u(t+ih) - h\sum_{i=0}^{m}\beta_i u'(t+ih)$$

La méthode multi-pas est consistante si $L\left[u(t)\right] = O\left(h^{m+1}\right)$ pour tout entier $m > 0$.

Le développement de Taylor de $L\left[u(t)\right]$ au voisinage de $t+ph,\ \forall p = 1,\ \ldots,\ m$ s'écrit :

$$L\left[u(t)\right] = C_0 u(t) + C_1 h u'(t) + \cdots + C_m h^m u^{(m)}(t) + C_{m+1} h^{m+1} u^{(m+1)}(t) + O\left(h^{m+2}\right)$$

les coefficients $C_0,\ C_1,\ \ldots,\ C_m,\ C_{m+1}$ sont des combinaisons linéaires des coefficients $\alpha_0,\ \alpha_1,\ \ldots,\ \alpha_{m-1},\ \alpha_m$ et $\beta_0,\ \beta_1,\ \ldots,\ \beta_m$ c'est-à-dire :

$$C_0 = \sum_{p=0}^{m}\alpha_p$$

$$C_k = \frac{1}{k!}\sum_{p=0}^{m}\alpha_p p^k - \frac{1}{(k-1)!}\sum_{p=0}^{m}\beta_p p^{k-1} \quad \forall\ k \geq 1$$

La méthode multi-pas est consistante si est seulement si :

$$C_k = 0 \ \Rightarrow\ \sum_{p=0}^{m}\alpha_p p^k = k\sum_{p=0}^{m}\beta_p p^{k-1},\ \forall\ k \geq 0 \ \text{ avec, } \alpha_m = 1$$

Si m est l'ordre de la méthode et $C_{m+1} \neq 0$ est dit le coefficient d'erreur est exprimé par :

$$C_{m+1} = \frac{1}{(m+1)!}\sum_{p=0}^{m}\alpha_p p^{m+1} - \frac{1}{m!}\sum_{p=0}^{m}\beta_p p^m \ \text{ avec, } \alpha_m = 1$$

On appelle constante d'erreur de la méthode, la quantité :

$$C = C_{m+1} \Big/ \sum_{p=0}^{m}\beta_p$$

$C_{m+1}h^{m+1}u^{(m+1)}(t)$ est l'erreur de troncature locale principale, qui est très utilisée dans la définition de stratégies adaptatives pour les méthodes multi-pas.

A titre d'exemple, soit le développement de Taylor de l'expression $L[u(t)]$ de la méthode de Dahlquist (1956) exprimée par :

$$u_{n+2} + 4u_{n+1} - 5u_n = h(4f_{n+1} + 2f_n)$$

or,

$$u(t+2h) = u(t) + 2hu'(t) + 2h^2u''(t) + \frac{4}{3}h^3u'''(t) + \frac{2}{3}h^4u^{(4)}(t) + O(h^5)$$

$$u(t+h) = u(t) + hu'(t) + \frac{h^2}{2}u''(t) + \frac{1}{6}h^3u'''(t) + \frac{1}{24}h^4u^{(4)}(t) + O(h^5)$$

$$u'(t+h) = u'(t) + hu''(t) + \frac{h^2}{2}u'''(t) + \frac{1}{6}h^3u^{(4)}(t) + O(h^4)$$

alors,

$$L[u(t)] = \frac{1}{6}h^4u^{(4)}(t) + O(h^5)$$

Si on utilise la formule de C_k, pour la méthode de Dahlquist sachant que :

$$\alpha_0 = -5,\ \alpha_1 = 4,\ \alpha_2 = 1,\ \beta_0 = 2,\ \beta_1 = 4 \text{ et } \beta_2 = 0$$

alors,

$$C_0 = \sum_{p=0}^{m}\alpha_p = -5 + 4 + 1 = 0$$

$$C_1 = \frac{1}{1!}\sum_{p=0}^{2}\alpha_p p - \frac{1}{0!}\sum_{p=0}^{2}\beta_p = (\alpha_1 + 2\alpha_2) - (\beta_0 + \beta_1 + \beta_2) = (4+2) - (2+4+0) = 0$$

$$C_2 = \frac{1}{2!}\sum_{p=0}^{2}\alpha_p p^2 - \frac{1}{1!}\sum_{p=0}^{2}\beta_p p = \frac{1}{2}(\alpha_1 + 4\alpha_2) - \frac{1}{1}(\beta_1 + 2\beta_2) = \frac{1}{2}(4+4) - (4+0) = 0$$

$$C_3 = \frac{1}{3!}\sum_{p=0}^{2}\alpha_p p^3 - \frac{1}{2!}\sum_{p=0}^{2}\beta_p p^2 = \frac{1}{6}(\alpha_1 + 8\alpha_2) - \frac{1}{2}(\beta_1 + 4\beta_2) = \frac{1}{6}(4+8) - \frac{1}{2}(4+0) = 0$$

$$C_4 = \frac{1}{4!}\sum_{p=0}^{2}\alpha_p p^4 - \frac{1}{3!}\sum_{p=0}^{2}\beta_p p^3 = \frac{1}{24}(\alpha_1 + 16\alpha_2) - \frac{1}{6}(\beta_1 + 8\beta_2) = \frac{1}{24}(4+16) - \frac{1}{6}(4+0) = \frac{1}{6}$$

$C_4 \neq 0$, ce qui montre que la méthode de Dahlquist est consistante à l'ordre 3.

En conséquence de la consistance, l'erreur locale de troncature d'une méthode multi-pas d'ordre *m*, est :

$$e_m = \sum_{p=0}^{m-1}\alpha_p u(t_{n+p}) + u(t_{n+m}) - h\sum_{p=0}^{m}\beta_p f(t_{n+p},\ u(t_{n+p})) = C_{m+1}h^{m+1}u^{(m+1)}(t_n) + O(h^{m+1})$$

La consistance est une condition nécessaire mais non suffisante pour la convergence de la méthode multi-pas d'ordre *m*. En effet,

$$\lim u_{n+m} = \sum_{i=0}^{m-1}(h\beta_i \lim f_{n+i} - \alpha_i \lim u_{n+i}) + h\beta_m \lim f_{n+m}$$

soit,

$$\lim u_{n+k} = l \text{ et } \lim f_{n+k} = f(\lim t_{n+k},\ \lim u_{n+k}) = f(t^*,\ l)\ \ \forall\ k = 0,\ 1,\ \ldots,\ m$$

donc

$$l \to \sum_{i=0}^{m-1}\left(h\beta_i f\left(t^*,\ l\right) - l\alpha_i\right) + h\beta_m f\left(t^*,\ l\right) \Rightarrow\ l\left(1 + \sum_{i=0}^{m-1}\alpha_i\right) \to hf\left(t^*,\ l\right)\sum_{i=0}^{m}\beta_i$$

or,

$$1 + \sum_{i=0}^{m-1}\alpha_i = 0 \text{ et } \ f\left(t^*,\ l\right)\sum_{i=0}^{m}\beta_i \neq 0 \ \Rightarrow\ h \to 0$$

c'est-à-dire, la méthode est convergente.

Une méthode multi-pas consistante est suffisamment convergente si et seulement est stable (Isaacson et Keller, 1966).

Pour étudier la stabilité d'une méthode multi-pas, considérons le problème de Cauchy suivant :

$$u'\left(t\right) = \lambda u\left(t\right),\ \ t \geq 0 \ \text{ et } \ u\left(0\right) = 1,\ \lambda \in \mathbb{C},\ \text{Re}\left(\lambda\right) < 1$$

alors,

$$u_{n+m} = -\sum_{p=0}^{m-1}\alpha_p u_{n+p} + h\sum_{p=0}^{m}\beta_p f_{n+p} \ \Rightarrow\ \sum_{p=0}^{m}\alpha_p u_{n+p} = h\sum_{p=0}^{m}\beta_p f_{n+p},\ \alpha_m = 1$$

ou,

$$\sum_{p=0}^{m}\alpha_p u_{n+p} - h\sum_{p=0}^{m}\beta_p f_{n+p} = 0 \ \Rightarrow\ \sum_{p=0}^{m}\alpha_p u_{n+p} - h\lambda\sum_{p=0}^{m}\beta_p u_{n+p} = 0$$

Ceci est une équation linéaire homogène d'ordre *m*. Considérant une classe de solution de cette équation sous la forme :

$$u_\eta = r^\eta,\ \ \eta \geq 0$$

soit,

$$\sum_{p=0}^{m}\alpha_p r^{n+p} - h\lambda\sum_{p=0}^{m}\beta_p r^{n+p} = 0 \ \Leftrightarrow\ \sum_{p=0}^{m}\alpha_p r^p - h\lambda\sum_{p=0}^{m}\beta_p r^p = 0$$

L'équation polynomiale de degré *m* obtenue s'appelle l'équation caractéristique, les polynômes $\rho\left(r\right)$ et $\sigma\left(r\right)$ dont :

$$\rho\left(r\right) = \sum_{p=0}^{m}\alpha_p r^p \text{ et } \ \sigma\left(r\right) = \sum_{p=0}^{m}\beta_p r^p$$

sont respectivement le premier et le second polynôme caractéristique de la méthode multi-pas. D'après le théorème fondamental de l'algèbre, l'équation caractéristique à *m* solutions dans l'ensemble des nombres complexes notés par :

$$r_1\left(h\lambda\right),\ r_2\left(h\lambda\right),\ \ldots,\ r_m\left(h\lambda\right)$$

Dans le cas où $h\lambda = 0$, l'équation caractéristique se rendre à $\rho\left(r\right) = 0$ et $r_j\left(0\right) = r_j$. Si les racines $r_1\left(h\lambda\right),\ r_2\left(h\lambda\right),\ \ldots,\ r_m\left(h\lambda\right)$ sont distincts, alors,

$$u_n = \sum_{p=1}^{m} a_p\left[r_p\left(h\lambda\right)\right]^n$$

$a_p,\ p = 1,\ \ldots, m$ sont des constantes qui peuvent être déterminés suivant les valeurs de u_n.

Pour que la méthode multi-pas est stable, c'est-à-dire $u_n \to 0$ quand $n \to \infty$ ou $h \to 0$, il faut que $\text{Re}\left[r_j\left(h\lambda\right)\right] < 0,\ \forall\ j = 1,\ 2,\ \ldots,\ m$. Si $r_1,\ r_2,\ \ldots,\ r_m$ ses racines de l'équation $\rho\left(r\right) = 0$

d'après Dahlquist (1956), la méthode multi-pas est zéro-stable (c'est-à-dire $\lambda = 0$) si et seulement si la *condition de racine* est satisfaite, c'est-à-dire :

$$\begin{cases} |r_j| \le 1, \ \forall \ j = 1, \ 2, \ \ldots, \ m \\ \rho'(r_j) \ne 0 \ \text{pour} \ |r_j| = 1 \end{cases}$$

Si en plus la fonction $f(t, \ u)$ est Lipschitzienne en *u*, la méthode multi-pas est suffisamment stable. Une méthode multi-pas est fortement stable si 1 est la seule racine de module 1 du polynôme $\rho(r)$. La méthode est faiblement stable si le polynôme $\rho(r)$ à plusieurs racines de module 1. Par exemple, les méthodes d'Adams-Bashforth et d'Adams-Moulton sont fortement stables, les méthodes de Nyström et de Milne-Simpson sont faiblement stables et les méthodes de différentiation rétrograde à *m* pas sont fortement stable uniquement pour $m \le 6$.

Soit l'exemple de la méthode :

$$u_{n+1} = u_{n-2} + \frac{3}{4} h \left(f_{n+1} + f_n + f_{n-1} + f_{n-2} \right)$$

qui s'écrit aussi,

$$u_{n+3} - u_n = \frac{3}{4} h \left(f_{n+3} + f_{n+2} + f_{n+1} + f_n \right)$$

alors,

$$\alpha_0 = -1, \ \alpha_1 = \alpha_2 = 0, \ \alpha_3 = 1 \ \text{et} \ \beta_0 = \beta_1 = \beta_2 = \beta_3 = 3/4$$

donc,

$$C_0 = \sum_{p=0}^{3} \alpha_p = -1 + 1 = 0, \quad C_1 = \sum_{p=0}^{3} \alpha_p p - \beta_0 \sum_{p=0}^{3} 1 = 3 - \frac{3}{4}(4) = 0$$

$$C_2 = \frac{1}{2} \sum_{p=0}^{3} \alpha_p p^2 - \beta_0 \sum_{p=0}^{3} p = \frac{1}{2} 3^2 - \frac{3}{4}(1 + 2 + 3) = \frac{9}{2} - \frac{3 \times 6}{4} = 0$$

$$C_3 = \frac{1}{6} \sum_{p=0}^{3} \alpha_p p^3 - \frac{1}{2} \beta_0 \sum_{p=0}^{3} p^2 = \frac{9}{6} - \frac{3}{8}(1 + 4 + 9) = \frac{3}{2} - \frac{3 \times 14}{8} = -\frac{15}{4} \ne 0$$

Ainsi, cette méthode est consistante à l'ordre 2.

Le polynôme caractéristique de la méthode est :

$$\rho(r) = r^3 - 1 = (r - 1)\left(r^2 + r + 1\right)$$

dont les racines sont :

$$r_1 = 1, \ r_2 = -\frac{1}{2} + \frac{\sqrt{3}}{2} i \ \text{ et } \ r_3 = -\frac{1}{2} - \frac{\sqrt{3}}{2} i$$

Remarquons que $|r_k| = 1$ et $\rho'(r_k) \ne 0, \ \forall \ k = 1, \ 2, \ 3$ donc la condition de racine est bien satisfaite, par conséquent la méthode est zéro-stable et puisque la méthode est consistante alors elle est convergente.

8. Extension vers les systèmes d'équations différentielles

Les méthodes numériques traitées précédemment consistent à résoudre les équations différentielles ordinaires du type $du/dt = f\left(t, \ u(t)\right)$. Toutes ces méthodes restent valables pour un problème de Cauchy sous forme d'un système d'équations différentielles :

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(t,\ \mathbf{u}),\quad \mathbf{u}(t_0)$$

u et **f** sont des vecteurs,

$$\frac{d\mathbf{u}}{dt} = \left(\frac{du_1}{dt},\ \frac{du_2}{dt},\ \ldots,\ \frac{du_n}{dt}\right)^t$$

$$\mathbf{f}(t,\ \mathbf{u}) = \left[f_1(t,\ \mathbf{u}),\ f_2(t,\ \mathbf{u}),\ \ldots,\ f_n(t,\ \mathbf{u})\right]^t \text{ et } \mathbf{u} = (u_1,\ u_2,\ \ldots,\ u_n)^t$$

Par exemple, le schéma général de la méthode de Runge-Kutta pour un tel système est :

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \sum_{i=1}^{s} b_i \mathbf{k}_i$$

avec,

$$\mathbf{k}_1 = h\mathbf{f}(t_n,\ \mathbf{u}_n),\ \mathbf{k}_2 = h\mathbf{f}(t_n + c_2 h,\ \mathbf{u}_n + a_{21}\mathbf{k}_1),\ \mathbf{k}_3 = h\mathbf{f}(t_n + c_3 h,\ \mathbf{u}_n + a_{31}\mathbf{k}_1 + a_{32}\mathbf{k}_2)$$

$$\mathbf{k}_4 = h\mathbf{f}(t_n + c_4 h,\ \mathbf{u}_n + a_{41}\mathbf{k}_1 + a_{42}\mathbf{k}_2 + a_{43}\mathbf{k}_3)$$

. . . . . .

$$\mathbf{k}_s = h\mathbf{f}(t_n + c_s h,\ \mathbf{u}_n + a_{s1}\mathbf{k}_1 + a_{s2}\mathbf{k}_2 + a_{s3}\mathbf{k}_3 + \ldots + a_{ss-1}\mathbf{k}_{s-1})$$

L'utilité des systèmes d'équations différentielles réside dans la résolution des équations ordinaires d'ordre supérieure (2ième, 3ième...). Soit par exemple l'équation différentielle de second ordre :

$$\frac{d^2u}{dt^2} = f\left(t,\ u,\ \frac{du}{dt}\right)$$

Si on pose $du/dt = v$ alors, $dv/dt = f(t,\ u,\ v)$, vectoriellement l'équation devient :

$$\frac{d\mathbf{u}}{dt} = \left(\frac{du}{dt},\ \frac{dv}{dt}\right)^T,\ \mathbf{f}(t,\ \mathbf{u}) = \left[v,\ f(t,\ \mathbf{u})\right]^T \text{ et } \mathbf{u} = (u\ ,\ v)^T$$

L'algorithme de Runge-Kutta d'ordre 4 pour résoudre le problème est le suivant :

$$\frac{du}{dt} = f(t,\ u,\ v) \text{ et } \frac{dv}{dt} = g(t,\ u,\ v)$$

avec comme conditions initiales

$$u_1 = u(t_1),\ v_1 = v(t_1)$$

alors,

$$k_1 = hf(t_n,\ u_n,\ v_n),\quad K_1 = hg(t_n,\ u_n,\ v_n)$$

$$k_2 = hf\left(t_n + \frac{h}{2},\ u_n + \frac{1}{2}k_1,\ v_n + \frac{1}{2}K_1\right),\quad K_2 = hg\left(t_n + \frac{h}{2},\ u_n + \frac{1}{2}k_1,\ v_n + \frac{1}{2}K_1\right)$$

$$k_3 = hf\left(t_n + \frac{h}{2},\ u_n + \frac{1}{2}k_2\ v_n + \frac{1}{2}K_2\right),\quad K_3 = hg\left(t_n + \frac{h}{2},\ u_n + \frac{1}{2}k_2\ v_n + \frac{1}{2}K_2\right)$$

$$k_4 = hf(t_n + h,\ u_n + k_3,\ v_n + K_3),\quad K_4 = hg(t_n + h,\ u_n + k_3,\ v_n + K_3)$$

$$u_{n+1} = u_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),\quad v_{n+1} = v_n + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

A titre d'exemple, soit l'équation différentielle du Hermite de second ordre :

$$\frac{d^2u}{dt^2}-2t\frac{du}{dt}+8u=0 \text{ avec comme condition initiales, } u(0)=12,\ \left.\frac{du}{dt}\right|_{t=0}=0$$

Cette équation différentielle possède une solution exacte sous forme de polynôme de Hermite de 4ième degré :

$$u(t)=16t^4-48t^2+12$$

Pour résoudre ce problème par l'algorithme de Runge-Kutta d'ordre 4, soit la transformation :

$$\frac{du}{dt}=v \ \Rightarrow\ \frac{dv}{dt}-2tv+8u=0 \ \Rightarrow\ \frac{dv}{dt}=2tv-8u$$

vectoriellement, s'écrit comme :

$$\begin{pmatrix} du/dt \\ dv/dt \end{pmatrix}=\begin{pmatrix} v \\ 2tv-8u \end{pmatrix}\equiv\begin{pmatrix} f(t,\ u,\ v) \\ g(t,\ u,\ v) \end{pmatrix}$$

Le script MATLAB ci-dessous permet de résoudre cette équation pour $0\le t\le 1$ et pour un pas $h=0.1$:

```
Uexact=@(t) 16*t^4-48*t^2+12; % la solution exacte Uexact
f=@(t,u,v) v; % Définition de la fonction f(t,u,v)
g=@(t,u,v) 2*t*v-8*u; % Définition de la fonction g(t,u,v)

h=0.1; % Définition du pas h
t=0:h:1;

u(1)=12; v(1)=0;  U(1)=Uexact(0); % Définition de la condition initiale

for n=1:numel(t)-1
   k1=h*f(t(n),u(n),v(n));K1=h*g(t(n),u(n),v(n));
 k2=h*f(t(n)+(h/2),u(n)+(k1/2),v(n)+(K1/2));
   K2=h*g(t(n)+(h/2),u(n)+(k1/2),v(n)+(K1/2));
 k3=h*f(t(n)+(h/2),u(n)+(k2/2),v(n)+(K2/2));
K3=h*g(t(n)+(h/2),u(n)+(k2/2),v(n)+(K2/2));
 k4=h*f(t(n+1),u(n)+k3,v(n)+K3); K4=h*g(t(n+1),u(n)+k3,v(n)+K3);

 u(n+1)=u(n)+(1/6)*(k1+2*k2+2*k3+k4);
 v(n+1)=v(n)+(1/6)*(K1+2*K2+2*K3+K4);
 U(n+1)= Uexact(t(n+1));
end

plot(t,U,'-b',t,u,'*g')
```

Le graphe de la figure 6.27, illustre la solution par la méthode de Runge-Kutta ainsi la comparaison avec la solution exacte de l'équation différentielle.

Soit l'exemple suivant d'un système d'équations différentielles :

$$\begin{cases} \dfrac{du}{dt}=2u+4v, \\ \dfrac{dv}{dt}=-u+6v. \end{cases} \quad \begin{cases} u(0)=-1, \\ v(0)=6. \end{cases}$$

De ce système on peut avoir :

$$\frac{du}{dt}=2u+4v=f(t,\ u,\ v)=f(u,\ v),\ \ \frac{dv}{dt}=-u+6v=g(u,\ v)$$

Par conséquent, l'application de l'algorithme de Runge-Kutta d'ordre 4 est peut-être explicité suivant le script MATLAB suivant :

```
% Définition des fonctions f(u,v) et g(u,v)
f=@(u,v) 2*u+4v;
g=@(u,v) -u+6*v;
h=0.1; % Définition du pas h
t=0:h:2;
u(1)=-1; v(1)=6; % Définition de la condition initiale
for n=1:numel(t)-1
 k1=h*f(u(n),v(n)); K1=h*g(u(n),v(n));
 k2=h*f(u(n)+(k1/2),v(n)+(K1/2)); K2=h*g(u(n)+(k1/2),v(n)+(K1/2));
 k3=h*f(u(n)+(k2/2),v(n)+(K2/2)); K3=h*g(u(n)+(k2/2),v(n)+(K2/2));
 k4=h*f(u(n)+k3,v(n)+K3); K4=h*g(u(n)+k3,v(n)+K3);
 u(n+1)=u(n)+(1/6)*(k1+2*k2+2*k3+k4);v(n+1)=v(n)+(1/6)*(K1+2*K2+2*K3+K4);
end
plot(t,u,'*b',t,v,'*r');
```

Ce script fournie la solution sous forme de graphes en points astérisques (Fig. 6.28). La solution analytique de ce système est :

$$\begin{cases}u(t)=(26t-1)e^{4t}\\ v(t)=(13t+1)e^{4t}\end{cases}$$

Ces deux solutions sont représentées dans le même graphe (Fig. 6.28) par des trais continus ou en remarque qu'il y a une identification parfaite entre la solution approchée et la solution exacte.

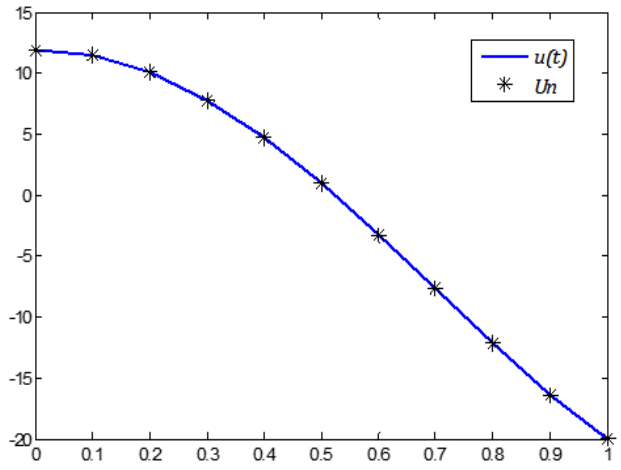


Fig. 6.27 : Résolution par RK4 d'une équation différentielle de 2ième ordre.

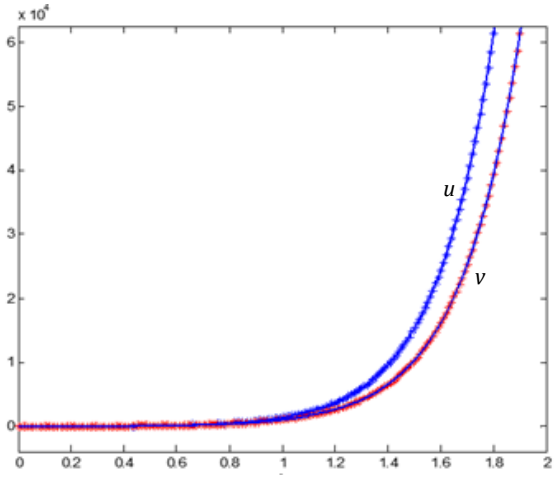


Fig. 6.28 : Résolution du système avec RK4.

9. Equations différentielles raides

Nous avons vu dans l'étude de la stabilité des méthodes numériques, que le choix du pas h joue un rôle dans la stabilité, sachant que le schéma numérique est appliqué à un cas type de problème de Cauchy ($u'=\lambda u$, $u(0)=1$), mais si la sensibilité de l'équation différentielle aux paramètres va rendre difficile la résolution par des méthodes numériques explicites (surtout), l'équation est dite raide (en arabe dite سَاق et en anglais dite *stiff*). Il existe plusieurs définitions formelles d'une équation différentielle raide. Une des plus simples est celle de Curtiss et Hirschfelder (1951) : "*Stiff equations are equations where certain implicit methods, in particular BDF, perform better, usually tremendously better, than explicit ones*".

Les équations différentielles raides existent souvent dans les problèmes de Cauchy d'ordre deux ou plus ou plus particulièrement dans les problèmes de Cauchy explicité sous forme d'un système d'équations différentielles. Pour illustrer le sujet (Quarteroni *et al.*, 2010), soit à titre d'exemple le système différentiel :

$$\frac{d\mathbf{u}}{dt}=\mathbf{A}\mathbf{u}(t)+\boldsymbol{\varphi}(t),\quad \mathbf{A}\in\mathbb{R}^{n\times n},\quad \boldsymbol{\varphi}(t)\in\mathbb{R}^{n}$$

où la matrice **A** possède les valeurs propres distinctes λ_j avec $\mathrm{Re}(\lambda_j)<0$, $\forall j=1,\ 2,\ \ldots,\ n$. Ce système est dite raide si :

$$r_s=\frac{\max_j\left|\mathrm{Re}(\lambda_j)\right|}{\min_j\left|\mathrm{Re}(\lambda_j)\right|}>>1$$

La solution exacte du système est exprimée par :

$$\mathbf{u}(t)=\sum_{j=1}^{n}C_j e^{\lambda_j t}\mathbf{v}_j+\boldsymbol{\psi}(t)$$

où $C_1,\ C_2,\ \ldots,\ C_n$ sont n constantes, $\{\mathbf{v}_j\}$ est une base constituée par les vecteurs propres de la matrice **A**, et $\boldsymbol{\psi}(t)$ est une solution particulière de l'équation différentielle. Si $r_s>>1$, on constate la présence dans la solution **u** de composantes qui tendent vers zéro avec des vitesses différentes. La composante qui tend le plus vite vers zéro quand $t\to\infty$ (celle qui est associée à la valeur propre de plus grand module) est celle qui impose la restriction la plus sévère sur le pas d'intégration, à moins bien sûr d'utiliser une méthode inconditionnellement absolument stable.

Nous avons vu que les méthodes implicites comme celle de Runge-Kutta implicites, multi-pas implicites et les BDF ont l'avantage d'être plus stables à celle des méthodes explicites, le seul problème dans la mise en œuvre de ces méthodes est plus couteux à cause de la transformation mathématique qui permet d'avoir un système d'équations numérique à résoudre à chaque pas de calcul, surtout si le problème est non linéaire.

Pour améliorer l'efficacité du calcul, on utilise des méthodes à pas variable, c'est-à-dire des méthodes dans lesquelles le pas varie à chaque itération. Soit les méthodes semi-implicites connu sous le nom des méthodes de Rosenbrock, parmi lesquelles la méthode de Kaps et Rentrop qui est une bonne implémentation de cette idée. Les méthodes de Rosenbrock à s pas se définissent par le schéma :

$$\mathbf{u}_1 = \mathbf{u}_0 + \sum_{i=1}^{s} b_i \mathbf{k}_i$$

les facteurs de $\mathbf{k}_i$ correction sont obtenus par la résolution du système de s équations linéaires :

$$\left(\mathbf{I}_s - \gamma h \mathbf{f}'\right).\mathbf{k}_i = h\mathbf{f}\left(\mathbf{u}_0 + \sum_{j=1}^{i-1} \alpha_{ij}\mathbf{k}_j\right) + h\mathbf{f}'.\sum_{j=1}^{i-1} \gamma_{ij}\mathbf{k}_j, \qquad i = 1,\ 2,\ \ldots,\ s$$

$\mathbf{f}'$ est la matrice jacobienne, les coefficients γ, b_i, α_{ij} et γ_{ij} sont des constants fixés indépendamment du problème posé. Pour minimiser le produit matrice-vecteur dans le membre droit du système peuvent être s'écrit en fonction de la quantité :

$$\mathbf{g}_i = \sum_{j=1}^{i-1} \gamma_{ij}\mathbf{k}_j + \gamma \mathbf{k}_i$$

Pour une méthode d'ordre 4 (s = 4), le système d'équations s'écrit :

$$\left(\frac{1}{\gamma h}\mathbf{I}_4 - \mathbf{f}'\right).\mathbf{g}_1 = h\mathbf{f}\left(\mathbf{u}_0\right)$$

$$\left(\frac{1}{\gamma h}\mathbf{I}_4 - \mathbf{f}'\right).\mathbf{g}_2 = h\mathbf{f}\left(\mathbf{u}_0 + a_{21}\mathbf{g}_1\right) + \frac{1}{h}c_{21}\mathbf{g}_1$$

$$\left(\frac{1}{\gamma h}\mathbf{I}_4 - \mathbf{f}'\right).\mathbf{g}_3 = h\mathbf{f}\left(\mathbf{u}_0 + a_{31}\mathbf{g}_1 + a_{32}\mathbf{g}_2\right) + \frac{1}{h}\left(c_{31}\mathbf{g}_1 + c_{32}\mathbf{g}_2\right)$$

$$\left(\frac{1}{\gamma h}\mathbf{I}_4 - \mathbf{f}'\right).\mathbf{g}_4 = h\mathbf{f}\left(\mathbf{u}_0 + a_{41}\mathbf{g}_1 + a_{42}\mathbf{g}_2 + a_{43}\mathbf{g}_3\right) + \frac{1}{h}\left(c_{41}\mathbf{g}_1 + c_{42}\mathbf{g}_2 + c_{43}\mathbf{g}_3\right)$$

a_{ij} et c_{ij} sont exprimés en fonction de α_{ij} et γ_{ij}.

10. Problèmes aux limites

Certaines équations différentielles de la physique, les conditions du problème sont imposées au niveau des limites supérieure et inférieure de l'intervalle [a, b], qui peut être sous forme $u(a) = \alpha$ et $u(b) = \beta$ pour l'équation $u'' = f(t,\ u,\ u')$, un tel problème aux limites s'appelle problème de Dirichlet, si les conditions concernent surtout les dérivées premières aux limites $u'(a) = \alpha$ et $u'(b) = \beta$, le problème est dit de Neumann ou bien mixte et dans ce cas le problème est dit de Robin. Pour résoudre ces problèmes deux méthodes sont utilisées, la méthode de tir (*shooting method*) et la méthode des différences finis.

10.1. Méthode de tir

Le principe de cette méthode est assez explicite qui consiste à remplacer le problème aux conditions aux limites par un problème de Cauchy et le résoudre et voir si $u(b)$ est proche de β. Pour le problème aux limites :

$$\begin{cases} u'' = f(t,\ u,\ u') \\ u(a) = \alpha,\ u(b) = \beta \end{cases}$$

on s'intéresse au problème de Cauchy :

$$\begin{cases} u'' = f(t,\ u,\ u') \\ u(a) = \alpha,\ u'(a) = d \end{cases}$$

d, ici est imposé de façon à ce que l'intégration numérique du problème de Cauchy aboutisse à ce que $u(b) = \beta$. L'emploi de telle méthode nécessite une programmation appropriée, l'essentielle est l'estimation de $u'(a) = d$ qui conduit à avoir $u(b) \approx \beta$.

Considérant le cas d'un problème de Dirichlet dont l'équation différentielle est linéaire :

$$\begin{cases} \dfrac{d^2u}{dt^2} = P(t)\dfrac{du}{dt} + Q(t)u + R(t) \\ u(a) = \alpha,\ u(b) = \beta \end{cases}$$

Soit le problème de Cauchy,

$$\begin{cases} \dfrac{d^2u}{dt^2} = P(t)\dfrac{du}{dt} + Q(t)u + R(t) \\ u(a) = \alpha,\ u'(a) = d_1 \end{cases}$$

d_1, une valeur choisie pour la dérivée de la solution $u(t)$, alors le problème peut être résolue par une des méthodes numériques vue précédemment (Runge-Kutta, multi-pas…) qui conduisent à la solution numérique $u_1(t)$ (Fig. 6.29) permettant de fournir la valeur $u_1(b) = \beta_1$ il est clair que si, $\beta_1 = \beta$ le problème est résolu et par conséquent la solution $u_1(t) \equiv u(t)$ sinon le processus est répété mais avec $u'(a) = d_2$ avec $d_2 \neq d_1$ qui conduit à $u_2(b) = \beta_2$ (Fig. 6.29).

Vu la linéarité de l'équation différentielle, la solution $u(t)$ peut être tirée approximativement à partir d'une combinaison linéaire des solutions $u_1(t)$ et $u_2(t)$:

$$u(t) = \lambda u_1(t) + (1-\lambda)u_2(t)$$

λ est un constante positive inférieur à 1, alors,

$$u(b) = \lambda u_1(b) + (1-\lambda)u_2(b)$$

équivalent a :

$$\beta = \lambda\beta_1 + (1-\lambda)\beta_2 \ \Rightarrow\ \lambda = \frac{\beta - \beta_2}{\beta_1 - \beta_2}$$

Finalement, la solution de l'équation différentielle est :

$$u(t) = \left(\frac{\beta - \beta_2}{\beta_1 - \beta_2}\right)u_1(t) - \left(\frac{\beta - \beta_1}{\beta_1 - \beta_2}\right)u_2(t)$$

La solution ainsi obtenue est approximative, et dépend des valeurs choisi d_1 et d_2 qui conduisent à β_1 et

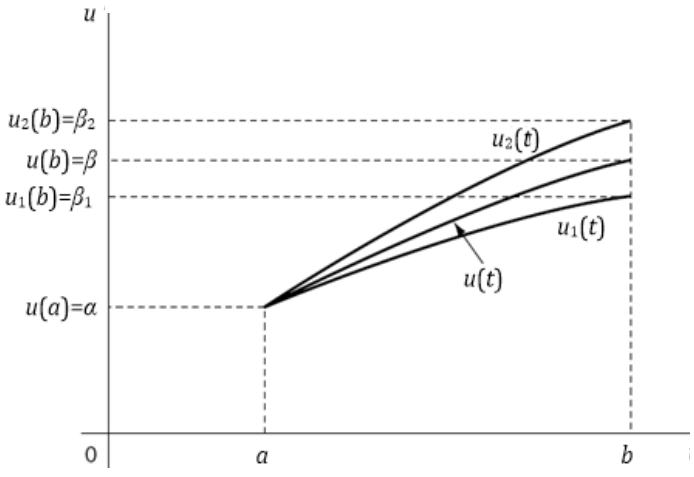


Fig. 6.29 : Illustration de la méthode de tir.

β_2, et pour qu'elle soit acceptable il faut que β est compris entre β_1 et β_2 ainsi que β_1 soit proche de β_2 dans le cas contraire la solution tirée est loin de la solution réelle.

Dans certaines méthodes de tir au lieu de choisir deux valeurs d_1 et d_2 pour la dérivée, une succession de valeurs d_1, d_2, . . . sont choisi ce qui conduit à une succession de valeurs β_1, β_2, … et pour tirer la solution, il faut tout d'abord chercher la valeur $d = u'(a)$ conduit à une valeur proche de $\beta = u(b)$, qui se fait par interpolation.

La méthode de tir peut être utilisée dans certains cas de problèmes non linéaires.

10.2. Méthodes des différences finies

Le développement d'ordre un en série de Taylor (1715) de la fonction $u(t_n)$ au voisinage de $t_n + h$ s'écrit :

$$u(t_n + h) = u(t_n) + hu'(t_n) + O(h^2)$$

De même le développement d'ordre un en série de Taylor de la fonction $u(t_n)$ au voisinage de $t_n - h$ s'écrit (en remplaçant *h* par –*h* dans la même expression précédente de $u(t_n + h)$:

$$u(t_n - h) = u(t_n) - hu'(t_n) + O(h^2)$$

de $u(t_n + h)$, la dérivée $u'(t_n)$ est peut-être tirée :

$$u'(t_n) = \frac{u(t_n + h) - u(t_n)}{h} + O(h)$$

de même, de l'expression de $u(t_n - h)$, la dérivée $u'(t_n)$ est donnée par :

$$u'(t_n) = \frac{u(t_n) - u(t_n - h)}{h} + O(h)$$

$O(h)$ est l'erreur de troncature d'ordre 1, qui tends vers zéro quand *h* tends vers zéro.

Ces approximations de la dérivée $u'(t_n)$ sont de premier ordre, par conséquent soit :

$$u'(t_n) \approx \frac{u(t_n + h) - u(t_n)}{h} \quad \text{ou} \quad \left(\frac{du}{dt}\right)_n \approx \frac{u_{n+1} - u_n}{h} \equiv \Delta u_n \text{ dite schéma avant}$$

$$u'(t_n) \approx \frac{u(t_n) - u(t_n - h)}{h} \quad \text{ou} \quad \left(\frac{du}{dt}\right)_n \approx \frac{u_n - u_{n-1}}{h} \equiv \nabla u_n \text{ dite schéma arrière}$$

Le développement d'ordre 2 en série de Taylor de la fonction $u(t_n)$ au voisinage de $t_n + h$ est :

$$u(t_n + h) = u(t_n) + hu'(t_n) + \frac{h^2}{2}u''(t_n) + O(h^3)$$

De même le développement d'ordre deux en série de Taylor de la fonction $u(t_n)$ au voisinage de $t_n - h$ est :

$$u(t_n - h) = u(t_n) - hu'(t_n) + \frac{h^2}{2}u''(t_n) + O(h^3)$$

par conséquent,

$$u(t_n+h)-u(t_n-h)=2hu'(t_n)+O(h^3) \;\Rightarrow\; u'(t_n)=\frac{u(t_n+h)-u(t_n-h)}{2h}+O(h^2)$$

alors,

$$u'(t_n)\approx\frac{u(t_n+h)-u(t_n-h)}{2h} \quad \text{ou} \quad \left(\frac{du}{dt}\right)_n\approx\frac{u_{n+1}-u_{n-1}}{2h}\equiv\delta u_n \text{ dite schéma centrale}$$

Graphiquement (Fig. 6.30), $u'(t_n)$ est le gradient de la droite tangente au point d'abscisse t_n de la courbe dont la fonction est $u=u(t)$ (point M_n). L'approximation avant Δu_n et arrière ∇u_n de la dérivée $u'(t_n)$ est exprimée par la droite (M_nM_{n+1}) et la droite (M_nM_{n-1}), tandis-que l'approximation centrale δu_n de la dérivée $u'(t_n)$ est exprimée par la droite $(M_{n+1}M_{n-1})$. Remarquons bien (Fig. 6.30) que les deux droites (M_nM_{n+1}) et (M_nM_{n-1}) ont un point d'intersection avec la droite tangente, par contre la droite $(M_{n+1}M_{n-1})$ est pratiquement parallèle à droite tangente ce qui justifier que l'approximation δu_n de la dérivée $u'(t_n)$ est meilleure que celle de Δu_n et ∇u_n.

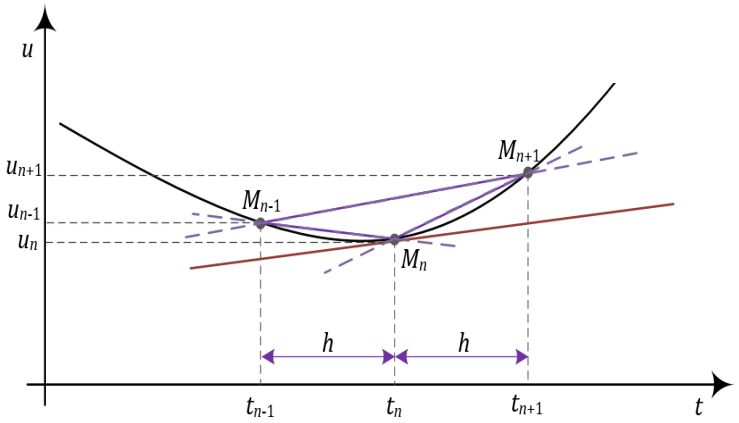


Fig. 6.30 : Approximations par différences finies.

Remarquons qu'on peut retrouver les trois méthodes d'Euler, explicite, implicite et du point central respectivement à partir du schéma avant, arrière et centrale de la dérivée $u'(t_n)$ pour l'équation différentielle de première ordre $du/dt=f(t,\ u)$ avec la condition initiale $u(a)=\alpha$.

Pour la méthode d'Euler explicite, le schéma aux différences finies avant :

$$\Delta u_n=\frac{du_n}{dt}=\frac{u_{n+1}-u_n}{h} \;\Rightarrow\; u_{n+1}-u_n=h\frac{du_n}{dt}=hf(t_n,\ u_n) \;\Rightarrow\; u_{n+1}=u_n+hf(t_n,\ u_n)$$

Pour la méthode d'Euler implicite, le schéma aux différences finies arrière :

$$\nabla u_n=\frac{du_n}{dt}=\frac{u_n-u_{n-1}}{h} \;\Rightarrow\; u_n-u_{n-1}=h\frac{du_n}{dt}=hf(t_n,\ u_n) \;\Rightarrow\; u_n=u_{n-1}+hf(t_n,\ u_n)$$

Par substitution de *n* par *n*+1, on obtient :

$$u_{n+1}=u_n+hf(t_{n+1},\ u_{n+1})$$

Pour la méthode d'Euler du point centrale, le schéma aux différences finies central :

$$\delta u_n=\frac{du_n}{dt}=\frac{u_{n+1}-u_{n-1}}{2h} \;\Rightarrow\; u_{n+1}-u_{n-1}=2h\frac{du_n}{dt}=2hf(t_n,\ u_n) \;\Rightarrow\; u_{n+1}=u_{n-1}+2hf(t_n,\ u_n)$$

Pour déterminer un schéma aux différences finies de la dérivée seconde $u''(t_n)$, soit les développements d'ordre deux en série de Taylor de la fonction $u(t_n)$ au voisinage de $t_n + h$ et $t_n - h$:

$$u(t_n + h) = u(t_n) + hu'(t_n) + \frac{h^2}{2}u''(t_n) + O(h^3)$$

$$u(t_n - h) = u(t_n) - hu'(t_n) + \frac{h^2}{2}u''(t_n) + O(h^3)$$

la somme $u(t_n + h) + u(t_n - h)$ conduit à :

$$u(t_n + h) + u(t_n - h) = 2u(t_n) + h^2 u''(t_n) + O(h^2) \;\Rightarrow\; u''(t_n) = \frac{u(t_n + h) - 2u(t_n) + u(t_n - h)}{h^2} + O(h^2)$$

C'est l'approximation d'ordre 2 de la dérivée seconde $u''(t_n)$ qui s'écrit aussi par l'expression :

$$u''(t_n) \approx \frac{u(t_n + h) - 2u(t_n) + u(t_n - h)}{h^2} \equiv \frac{u_{n+1} - 2u_n + u_{n-1}}{h^2}$$

alors, le schéma aux différences finies approprié pour l'équation différentielle $u'' = f(t,\ u,\ u')$ est peut-être exprimé par :

$$\frac{u_{\eta+1} - 2u_\eta + u_{\eta-1}}{h^2} = f\left(t_\eta,\ u_\eta,\ \frac{u_{\eta+1} - u_{\eta-1}}{2h}\right) \;\Rightarrow\; u_{\eta+1} - 2u_\eta + u_{\eta-1} = h^2 f\left(t_\eta,\ u_\eta,\ \frac{u_{\eta+1} - u_{\eta-1}}{2h}\right)$$

ou,

$$u_{\eta-1} - 2u_\eta + u_{\eta+1} - h^2 f\left(t_\eta,\ u_\eta,\ \frac{u_{\eta+1} - u_{\eta-1}}{2h}\right) = 0$$

Pour les conditions aux limites deux cas peuvent se présenter,

- *Cas du problème de Dirichlet*, c'est-à-dire $u(a) = \alpha$ et $u(b) = \beta$, soit :

$$\varphi_1(u_1,\ u_2,\ \ldots,\ u_N) \equiv u_1 - \alpha = 0 \quad \text{et} \quad \varphi_N(u_1,\ u_2,\ \ldots,\ u_N) \equiv u_N - \beta = 0$$

- *Cas du problème de Neumann*, c'est-à-dire $u'(a) = \alpha$ et $u'(b) = \beta$. Soit les deux valeurs hors de l'intervalle $[a,\ b]$ $u_0 \equiv u(a - h)$ et $u_{N+1} \equiv u(b + h)$, qui sont utilisées comme des valeurs intermédiaires pour préparer les équations aux limites (Fig. 6.31).

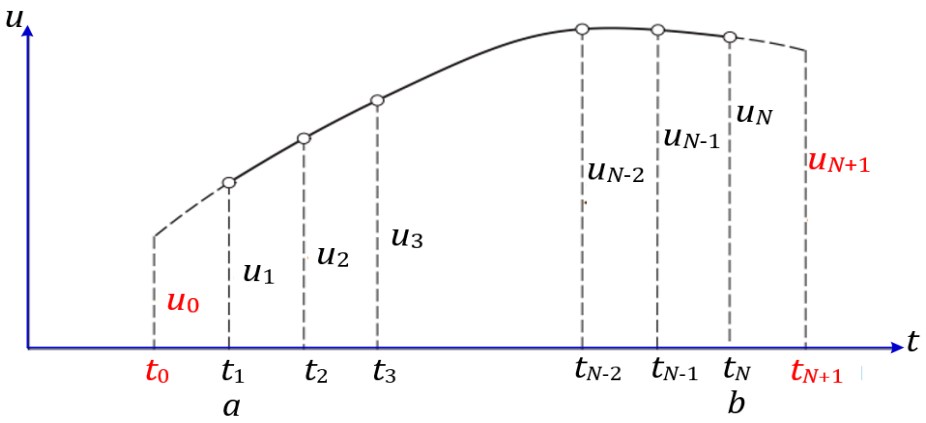


Fig. 6.31 : Méthode de différences finies dans un problème de Neumann.

Pour la condition au limite amont,

$$u'(a)=\alpha \;\Rightarrow\; \frac{u_2-u_0}{2h}=\alpha \;\Rightarrow\; u_0=u_2-2h\alpha$$

En remplaçant dans la relation numérique :

$$u_0-2u_1+u_2=h^2 f\left(t_1,\ u_1,\ \frac{u_2-u_0}{2h}\right) \;\Rightarrow\; -2u_1+2u_2-h^2 f\left(t_1,\ u_1,\ \alpha\right)-2h\alpha=0$$

soit,

$$\varphi_1\left(u_1,\ u_2,\ \ldots,\ u_N\right)\equiv -2u_1+2u_2-h^2 f\left(t_1,\ u_1,\ \alpha\right)-2h\alpha=0$$

Pour la condition à la limite avale,

$$u'(b)=\beta \;\Rightarrow\; \frac{u_{N+1}-u_{N-1}}{2h}=\beta \;\Rightarrow\; u_{N+1}=u_{N-1}+2h\beta$$

En remplaçant dans la relation numérique :

$$u_{N-1}-2u_N+u_{N+1}=h^2 f\left(t_N,\ u_N,\ \frac{u_{N+1}-u_{N-1}}{2h}\right) \;\Rightarrow\; 2u_{N-1}-2u_N-h^2 f\left(t_N,\ u_N,\ \beta\right)+2h\beta=0$$

soit,

$$\varphi_N\left(u_1,\ u_2,\ \ldots,\ u_N\right)\equiv 2u_{N-1}-2u_N-h^2 f\left(t_N,\ u_N,\ \beta\right)+2h\beta=0$$

A part les points frontières à t_1 et t_N,

$$\varphi_\eta\left(u_1,\ u_2,\ \ldots,\ u_N\right)=u_{\eta-1}-2u_\eta+u_{\eta+1}-h^2 f\left(t_\eta,\ u_\eta,\ \frac{u_{\eta+1}-u_{\eta-1}}{2h}\right)=0 \text{ pour } \eta=2,\ 3,\ \ldots,\ N-1$$

Soit finalement le système d'équations numériques de N équations et N inconnus :

$$\begin{cases}
\varphi_1\left(u_1,\ u_2,\ \ldots,\ u_N\right)=0,\\
\varphi_2\left(u_1,\ u_2,\ \ldots,\ u_N\right)=0,\\
\varphi_3\left(u_1,\ u_2,\ \ldots,\ u_N\right)=0,\\
\quad . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ ,\\
\varphi_{N-1}\left(u_1,\ u_2,\ \ldots,\ u_N\right)=0,\\
\varphi_N\left(u_1,\ u_2,\ \ldots,\ u_N\right)=0.
\end{cases}$$

Pour le problème de Dirichlet,

$$\begin{cases}
u_1-\alpha=0,\\
u_1-2u_2+u_3-h^2 f\left(t_2,\ u_2,\ \frac{u_3-u_1}{2h}\right)=0,\\
u_2-2u_3+u_4-h^2 f\left(t_3,\ u_3,\ \frac{u_4-u_2}{2h}\right)=0,\\
\quad . \ ,\\
u_{N-2}-2u_{N-1}+u_N-h^2 f\left(t_{N-1},\ u_{N-1},\ \frac{u_N-u_{N-2}}{2h}\right)=0,\\
u_N-\beta=0.
\end{cases}$$

Pour le problème de Neumann,

$$\begin{cases}
2u_1-2u_2+h^2 f\left(t_1,\ u_1,\ \alpha\right)+2h\alpha=0,\\
u_1-2u_2+u_3-h^2 f\left(t_2,\ u_2,\ \frac{u_3-u_1}{2h}\right)=0,\\
u_2-2u_3+u_4-h^2 f\left(t_3,\ u_3,\ \frac{u_4-u_2}{2h}\right)=0,\\
\quad . \ ,\\
u_{N-2}-2u_{N-1}+u_N-h^2 f\left(t_{N-1},\ u_{N-1},\ \frac{u_N-u_{N-2}}{2h}\right)=0,\\
2u_{N-1}-2u_N-h^2 f\left(t_N,\ u_N,\ \beta\right)+2h\beta=0.
\end{cases}$$

Le système peut être résolue par les méthodes numériques pour déterminer $u_1,\ u_2,\ u_3,\ \ldots,\ u_N$ solution de l'équation différentielle.

Etudions le cas de l'équation différentielle linéaire de seconde ordre suivante,

$$A(t)\frac{d^2u}{dt^2}+B(t)\frac{du}{dt}+B(t)u=g(t)\ \Rightarrow\ f(t,\ u,\ u')=\frac{g(t)}{A(t)}-\frac{B(t)}{A(t)}u'-\frac{C(t)}{A(t)}u$$

Par conséquent, le schéma aux différences de cette équation différentielle est :

$$A(t_\eta)\frac{u_{\eta+1}-2u_\eta+u_{\eta-1}}{h^2}+B(t_\eta)\frac{u_{\eta+1}-u_{\eta-1}}{2h}+C(t_\eta)u_\eta=g(t_\eta)$$

qui s'écrit aussi,

$$\left[2A(t_\eta)+hB(t_\eta)\right]u_{\eta+1}+2\left[C(t_\eta)h^2-2A(t_\eta)\right]u_\eta+\left[2A(t_\eta)-hB(t_\eta)\right]u_{\eta-1}=2h^2g(t_\eta)$$

on pose,

$$d_\eta^+=2A(t_\eta)+hB(t_\eta),\ d_\eta=2\left[C(t_\eta)h^2-2A(t_\eta)\right]\text{ et }d_\eta^-=2A(t_\eta)-hB(t_\eta)\ d_\eta^-=2A(t_\eta)-hB(t_\eta)$$

soit,

$$d_\eta^-u_{\eta-1}+d_\eta u_\eta+d_\eta^+u_{\eta+1}=2h^2g(t_\eta)$$

- Si le problème est de type de Dirichlet, le système d'équations linéaires devient :

$$\begin{cases}u_1=\alpha,\\ d_2^-u_1+d_2u_2+d_2^+u_3=2h^2g(t_2),\\ d_3^-u_2+d_3u_3+d_3^+u_4=2h^2g(t_3),\\ \ldots\ldots\ldots\ldots\ldots\ldots\\ d_{N-1}^-u_{N-2}+d_{N-1}u_{N-1}+d_{N-1}^+u_N=2h^2g(t_N),\\ u_N=\beta.\end{cases}$$

Sous forme matricielle tridiagonale **Mu=G** :

$$\mathbf{Mu}=\mathbf{G}\ \Leftrightarrow\ \begin{bmatrix}1 & 0 & 0 & \cdots & 0\\ d_2^- & d_2 & d_2^+ & & \\ 0 & d_3^- & d_3 & d_3^+ & \\ & & \ddots & \ddots & \ddots\\ & & d_{N-2}^- & d_{N-2} & d_{N-2}^+ & 0\\ & & & d_{N-1}^- & d_{N-1} & d_{N-1}^+\\ 0 & \cdots & & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}u_1\\ u_2\\ u_3\\ \vdots\\ u_{N-1}\\ u_N\end{bmatrix}=\begin{bmatrix}\alpha\\ 2h^2g_2\\ 2h^2g_3\\ \vdots\\ 2h^2g_{N-1}\\ \beta\end{bmatrix}$$

L'algorithme de la construction de chaque composante de la matrice **M** et du vecteur **G** se fait comme suit :

$$M(1,\ 1)=1,\ \ M(1,\ j)=0,\ \ j=2,\ \ldots,\ N,$$

$$M(i,\ j)=0,\ \ i=2,\ \ldots,\ N-2,\ \ j=i+2,\ \ldots,\ N,$$

$$M(i,\ j)=0,\ \ i=3,\ \ldots,\ N-1,\ \ j=1,\ \ldots,\ i-2,$$

$$M(N,\ j)=0,\ \ j=1,\ \ldots,\ N-1,\ \ M(N,\ N)=1.$$

$$\left.\begin{aligned} M(i,\ i-1)&=d_i^-,\\ M(i,\ i)&=d_i,\\ M(i,\ i+1)&=d_i^+, \end{aligned}\right\}\ i=2,\ \ldots,\ N-1, \qquad \begin{aligned} G(1)&=\alpha,\\ G(i)&=2h^2g_i,\ \ i=2,\ \ldots,\ N-1,\\ G(N)&=\beta. \end{aligned}$$

- Si le problème est de type de Neumann, le système d'équations devient :

$$\begin{cases} d_1u_1+(d_1^-+d_1^+)u_2=2h(hg_1+d_1^-\alpha),\\ d_2^-u_1+d_2u_2+d_2^+u_3=2h^2g_2,\\ d_3^-u_2+d_3u_3+d_3^+u_4=2h^2g_3,\\ \ldots\ldots\ldots\ldots\ldots\ldots\\ d_{N-1}^-u_{N-2}+d_{N-1}u_{N-1}+d_{N-1}^+u_N=2h^2g_{N-1},\\ (d_N^-+d_N^+)u_{N-1}+d_Nu_N=2h(hg_N-d_N^+\beta). \end{cases}$$

Le système d'équations obtenu peut être résolu numériquement par la méthode itérative de Gauss–Seidel ou par la méthode de surrelaxation successive (SOR) permet la détermination de $u_1,\ u_2,\ u_3,\ \ldots,\ u_{N-1}$ et u_N solution de l'équation différentielle.

Sous forme matricielle tridiagonale **Mu=G** :

$$\mathbf{Mu}=\mathbf{G} \quad\Leftrightarrow\quad \begin{bmatrix} d_1 & d_1^-+d_1^+ & 0 & \cdots & 0\\ d_2^- & d_2 & d_2^+ & & \\ 0 & d_3^- & d_3 & d_3^+ & \\ & & \ddots & \ddots & \ddots\\ & & d_{N-2}^- & d_{N-2} & d_{N-2}^+\\ & & & d_{N-1}^- & d_{N-1} & d_{N-1}^+\\ 0 & \cdots & & 0 & d_N^-+d_N^+ & d_N \end{bmatrix} \begin{bmatrix} u_1\\ u_2\\ u_3\\ \vdots\\ u_{N-1}\\ u_N \end{bmatrix} = \begin{bmatrix} 2h(hg_1+d_1^-\alpha)\\ 2h^2g_2\\ 2h^2g_3\\ \vdots\\ 2h^2g_{N-1}\\ 2h(hg_N-d_N^+\beta) \end{bmatrix}$$

L'algorithme de la construction de la matrice tridiagonale **M** ainsi que le vecteur **G** sont :

$$\begin{aligned} &M(1,\ 1)=d_1,\ \ M(1,\ 2)=d_1^-+d_1^+,\ \ M(1,\ j)=0,\ \ j=3,\ \ldots,\ N,\\ &M(i,\ j)=0,\ \ i=2,\ \ldots,\ N-2,\ \ j=i+2,\ \ldots,\ N\\ &\left.\begin{aligned} M(i,\ i-1)&=d_i^-,\\ M(i,\ i)&=d_i,\\ M(i,\ i+1)&=d_i^+, \end{aligned}\right\}\ i=2,\ \ldots,\ N-1,\\ &M(i,\ j)=0,\ \ i=3,\ \ldots,\ N-1,\ \ j=1,\ \ldots,\ i-2,\\ &M(N,\ j)=0,\ \ j=1,\ \ldots,\ N-2,\ \ M(N,\ N-1)=d_N^-+d_N^+,\\ &M(N,\ N)=d_N. \end{aligned}$$

$$\begin{cases} G(1)=2h(hg_1+d_1^-\alpha),\\ G(i)=2h^2g_i,\ \ i=2,\ \ldots,\ N-1,\\ G(N)=2h(hg_N-d_N^+\beta). \end{cases}$$

Ce système d'équations peut être résolu numériquement par la méthode itérative de Gauss–Seidel ou par la méthode de surrelaxation successive (SOR) permet la détermination de $u_1,\ u_2,\ u_3,\ \ldots,\ u_{N-1}$ et u_N solution de l'équation différentielle.

Soit l'exemple suivant :

$$\frac{d^2u}{dt^2}+4u=4t\ \text{ avec, }\ u(0)=0,\ u\left(\frac{3\pi}{4}\right)=2$$

dans ce cas,

$$\alpha = 0,\ \beta = 2,\ A(t) = 1,\ B(t) = 0,\ C(t) = 4 \text{ et } g(t) = 4t,$$

par conséquent,

$$d_\eta^+ = 2,\ d_\eta^+ = 4\left(2h^2 - 1\right) \text{ et } d_\eta^- = 2.$$

qui s'écrit sous la forme matricielle **Mu** = **G** :

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 2 & 4(2h^2-1) & 2 & & \\ 0 & 2 & 4(2h^2-1) & 2 & \\ & \ddots & \ddots & \ddots & 0 \\ & & 2 & 4(2h^2-1) & 2 \\ 0 & & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} 0 \\ 8h^2t_2 \\ 8h^2t_3 \\ \vdots \\ 8h^2t_{N-1} \\ 2 \end{bmatrix}$$

Les éléments de la matrice tridiagonale **M** et du vecteur **B** sont interprétés comme suit :

$$\begin{cases} M(1,1) = 1,\ M(1,j) = 0,\ j = 2,\ \ldots,\ N, \\ M(i,j) = 0,\ i = 2,\ \ldots,\ N-2,\ j = i+2,\ \ldots,\ N, \\ \left.\begin{array}{l} M(i,i-1) = 2, \\ M(i,i) = 4(2h^2-1), \\ M(i,i+1) = 2, \end{array}\right\} i = 2,\ \ldots,\ N-1, \\ M(i,j) = 0,\ i = 3,\ \ldots,\ N-1,\ j = 1,\ \ldots,\ i-2, \\ M(N,j) = 0,\ j = 1,\ \ldots,\ N-1,\ M(N,N) = 1. \end{cases}$$

$$\begin{cases} G(1) = 0, \\ G(i) = 8h^2t_i,\ i = 2,\ \ldots,\ N-1, \\ G(N) = 2. \end{cases}$$

Le script suivant permet la résolution de l'équation différentielle pour un pas h = 0.2, sachant que la solution exacte est $u(t) = t + ((3\pi - 8)/4)\sin 2t$.

```
tDebut=0; tFin=3*pi/4;
h=0.2; % Définition du pas h
t=tDebut:abs(h):tFin; N=numel(t);
% Construction de la Matrice M
M(1,1)=1; M(1,2:N)=0;
for i=2:N-2
  for j=i+2:N
      M(i,j)=0;
  end
end
for i=2:N-1
  M(i,i-1)=2; M(i,i)=4*(2*h^2-1);
  M(i,i+1)=2;
end
for i=3:N-1
  for j=1:i-2
     M(i,j)=0;
  end
end
M(N,1:N-1)=0; M(N,N)=1;
% Construction du vecteur G
G(1)=0; G(2:N-1)=8*(h^2)*t(2:N-1); G(N)=2;
u=inv(M)*G';
for i=1:N
```

i	t_i	u_i	$u(t_i)$
1	0.0	0.000	0.000
2	0.2	0.282	0.339
3	0.4	0.550	0.656
4	0.6	0.795	0.932
5	0.8	1.008	1.156
6	1.0	1.188	1.324
7	1.2	1.338	1.441
8	1.4	1.466	1.519
9	1.6	1.583	1.579
10	1.8	1.703	1.642
11	2.0	1.839	1.730
12	2.2	2.000	1.861

```
 U(i)=t(i)+((3*pi-8)/4)*sin(2*t(i)); % Solution exacte de l'équation
end
plot(t,U,'-r',t,u,'*b')
```

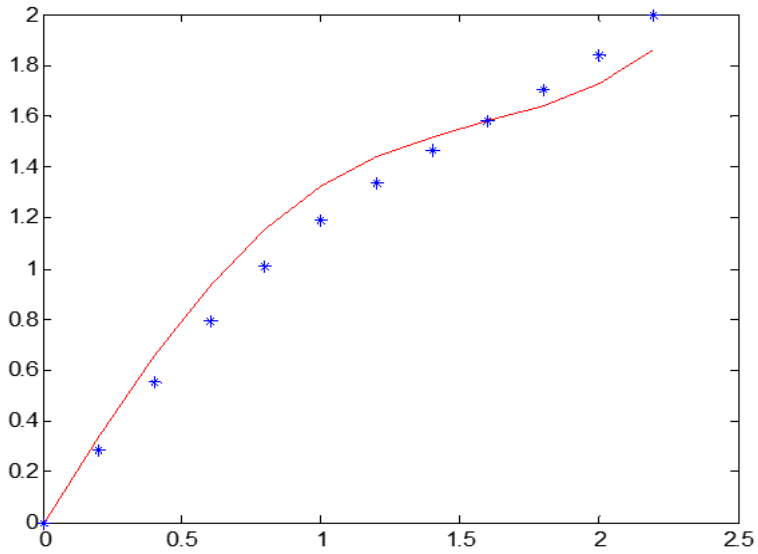


Fig. 6. 32 : Résolution d'un problème de Dirichlet par la méthode des différences finies.

La figure 6.32 illustre la solution approchée de l'équation différentielle par la méthode des différences finies ainsi que la solution exacte (courbe).

Soit aussi, l'exemple :

$$t^2\frac{d^2u}{dt^2}-3t\frac{du}{dt}+4u=\ln t \text{ avec, } u'(1)=1/4,\ u'(3)=1/12$$

Dans ce cas,

$$\alpha=1/4,\ \beta=1/12,\ A(t)=t^2,\ B(t)=-3t,\ C(t)=4 \text{ et } g(t)=\ln t$$

par conséquent,

$$d_\eta^+=2t_\eta^2-3ht_\eta,\ \ d_\eta=4\left(2h^2-t_\eta^2\right) \text{ et } \ d_\eta^-=2t_\eta^2+3ht_\eta$$

Soit l'algorithme de la construction des composantes de la matrice **M** et du vecteur **G** :

$$\begin{bmatrix} M(1,\ 1)=4(2h^2-1),\ M(1,\ 2)=4,\ M(1,\ j)=0,\ j=3,\ldots,\ N, \\ M(i,\ j)=0,\ i=2,\ \ldots,\ N-2,\ \ j=i+2,\ \ldots,\ N \\ \left.\begin{matrix} M(i,\ i-1)=2t_i^2+3ht_i, \\ M(i,\ i)=4(2h^2-t_i^2), \\ M(i,\ i+1)=2t_i^2-3ht_i, \end{matrix}\right\}\ i=2,\ \ldots,\ N-1, \\ M(i,\ j)=0,\ i=3,\ \ldots,\ N-1,\ \ j=1,\ \ldots,\ i-2, \\ M(N,\ j)=0,\ \ j=1,\ \ldots,\ N-2,\ M(N,\ N-1)=36, \\ M(N,\ N)=4(2h^2-9). \end{bmatrix}$$

$$\begin{bmatrix} G(1)=\frac{2h}{4}(2+3h), \\ G(i)=2h^2\ln t_i,\ \ i=2,\ \ldots,\ N-1, \\ G(N)=2h\left(h\ln 3-\frac{3}{4}(2-h)\right). \end{bmatrix}$$

La solution exacte de l'équation différentielle est utilisée pour la comparaison avec la solution approchée par la méthode des différences finies.

$$u(t)=\frac{1}{4}(1+\ln t)$$

Le script suivant, permet de résoudre l'équation différentielle par la méthode des différences finies et le compare avec la solution exacte (Fig. 6.33) :

```
tDebut=1; tFin=3;
h=0.1; % le pas h
t=tDebut:h:tFin; N=numel(t);
%Construction de la Matrice M
M(1,1)=4*(2*h*h-1);
M(1,2)=4; M(1,3:N)=0;
for i=2:N-2
   for j=i+2:N
       M(i,j)=0;
   end
end
for i=2:N-1
 M(i,i-1)=2*t(i)*t(i)+3*h*t(i);
 M(i,i)=4*(2*h^2-t(i)^2);
 M(i,i+1)=2*t(i)^2-3*h*t(i);
end
for i=3:N-1
  for j=1:i-2
      M(i,j)=0;
  end
end
 M(N,1:N-2)=0;
 M(N,N-1)=36;
 M(N,N)=4*(2*h^2-9);
% Construction du vecteur G
 G(1)=(2*h/4)*(2+3*h);
 G(2:N-1)=2*(h^2)*log(t(2:N-1));
 G(N)=2*h*(h*log(3)-(3/4)*(2-h));
 G=G';
 u=inv(M)*G; % Solution du système linéaire obtenue
for i=1:N
    U(i)=(1/4)*(1+log(t(i))); % Solution exacte de l'équation
end
 plot(t,U,'-r',t,u,'*b')
```

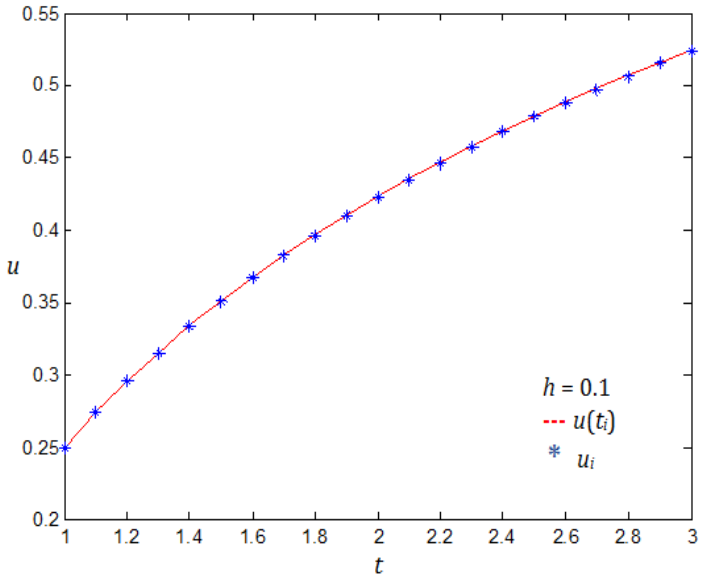


Fig. 6.33 : Résolution d'un problème de Neumann par la méthode des différences finies.

11. MATLAB et équations différentielles ordinaires

Pour résoudre numériquement les équations différentielles ordinaires de type problème de Cauchy de premier ordre, une variété de solveurs est incluse en MATLAB qui peut être appliqués suivant la syntaxe :

```
>> f=@(t,u) f(t,u); % Définition de l'expression de f(t,u)
>> [t,u]=solveur(f,[a,b],u(a)); % Renvoi la solution sous forme de tableaux
>> solveur(f,[a,b],u(a)) % Renvoi la solution sous forme graphique
```

solveur est une fonction près installée dans MATLAB comme :

- **ode45**, utilise la méthode de Dormand-Prince qui est une méthode de de Runge-Kutta d'ordre 4 et 5 à pas adaptatif (Dormand et Prince, 1980 ; Cash et Karp, 1990).

- **ode23**, utilise la méthode de Bogacki et Shampine qui est une méthode de Runge-Kutta d'ordre 2 et 3 à pas adaptatif (Shampine et Reichelt, 1997).

- **ode113**, utilise la méthode d'Adams-Bashforth comme prédicteur et la méthode d'Adams Moulton comme correcteur (Moler, 2004) .

- **ode15s**, solveur pour les équations différentielles raides, il utilise la méthode de différenciation arrière BDF (Süli *et al.*, 2003) ou la méthode de Gear pour les EDOs raides.

- **ode23s**, solveur pour les équations différentielles raides basé sur la formule de Rosenbrock modifiée d'ordre 2.

- **ode23t**, utilise la méthode des trapèzes, ce solveur est utilisé pour les équations différentielles modérément sévères.

- **ode23tb**, utilise la méthode des trapèzes avec la formule de différenciation arrière d'ordre deux, connue aussi par TR–BDF2 (Bank *et al.*, 1985 ; Dharmaraja, 2007).

- **ode15i**, utilise la formule de différenciation arrière pour résoudre les équations différentielles implicite de la forme $f\left(t,\ u,\ u'\right)=0$.

Dans le cas d'un problème de Cauchy de deuxième ordre, les solveurs du MATLAB précédents peuvent être utilisés après la formulation d'une fonction MATLAB. Soit l'exemple de l'équation différentielle du Hermite de second ordre :

$$\frac{d^2u}{dt^2}-2t\frac{du}{dt}+8u=0 \text{ avec comme condition initiales, } u\left(0\right)=12,\ \left.\frac{du}{dt}\right|_{t=0}=0$$

Cette équation différentielle possède une solution exacte sous forme de polynôme de Hermite de 4ième degré :

$$u\left(t\right)=16t^4-48t^2+12$$

Pour utiliser un solveur de MATLAB comme par exemple **ode45** ou **ode23**, soit la fonction :

```
function dudt = twoeq(t,u)
dudt(1)=u(2); dudt(2)=2*t*u(2)-8*u(1); dudt=dudt';
end
```

Dans "Command Window" du MATLAB, l'ajout de la commande :

```
>> ode45(@twoeq,[0,1],[12,0])
```

ou la commande :

```
>> ode23(@twoeq,[0,1],[12,0])
```

permettant de fournir la solution sous forme graphique (Fig. 6.34 et 6.35). Le graphe en bleu représente la solution $u(t)$ et le vert représente sa dérivée $u'(t)=v(t)$.

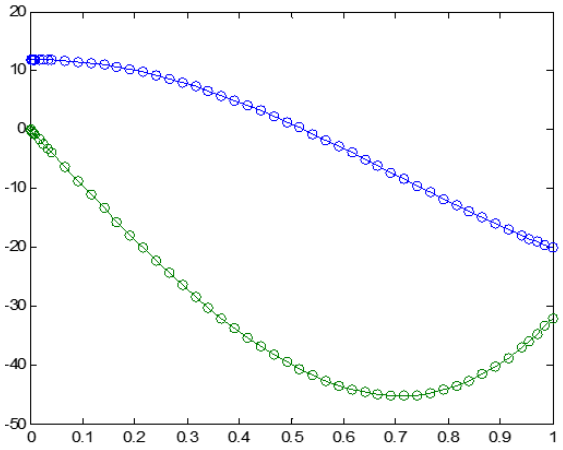


Fig. 6.34 : Résolution avec le solveur **ode45**.

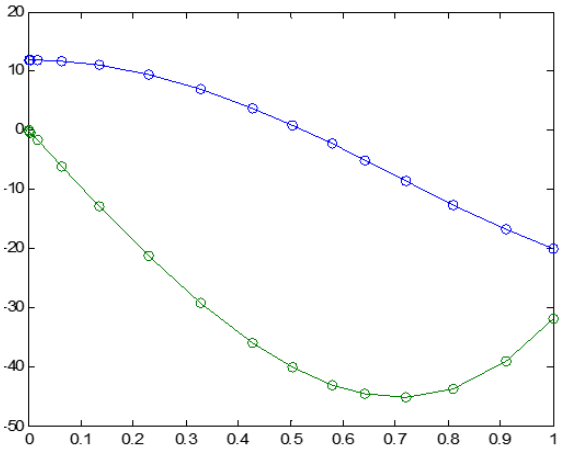


Fig. 6.35 : Résolution avec le solveur **ode23**.

Pour les problèmes aux limites, comme par exemple le problème de Dirichlet suivante,

$$\begin{cases} \dfrac{d^2u}{dt^2}=P(t)\dfrac{du}{dt}+Q(t)u+R(t) \\ u(a)=\alpha,\ u(b)=\beta \end{cases} \Rightarrow \frac{du}{dt}=v \Rightarrow \frac{dv}{dt}=P(t)v+Q(t)u+R(t)$$

La résolution du problème avec les solveurs de MATLAB (Kierzenka et Shampine, 2001) nécessite en premier lieu de formuler le *code de l'équation* par la fonction :

```
function dudt = bvpfcn(t,u)
dudt(1)=u(2);
dudt(2)=P*u(2)+Q*u(1)+R; % P, Q et R sont des fonctions de t
end
```

Soit aussi à formuler la fonction dite *code des conditions aux limites*,

```
function res = bcfcn(ua,ub)
res(1)= ua(1)-alpha;
res(2)= ub(1)-beta;
end
```

Pour *définir les options*, en utilisant la fonction MATLAB **bvpset** pour activer l'affichage des statistiques du solveur et spécifier des tolérances d'erreur brutes pour mettre en évidence la différence de contrôle d'erreur entre les solveurs. Aussi, pour plus d'efficacité, soit la matrice jacobienne :

$$\mathbf{J}=\begin{pmatrix} \dfrac{\partial f_1}{\partial u} & \dfrac{\partial f_1}{\partial v} \\ \dfrac{\partial f_2}{\partial u} & \dfrac{\partial f_2}{\partial v} \end{pmatrix}=\begin{pmatrix} 0 & 1 \\ Q(t) & P(t) \end{pmatrix}$$

Soit alors la fonction,

```
function dfdu = jac(t, u)
dfdu(1,1)= 0; dfdu(1,2)= 1;
dfdu(2,1)= Q; dfdu(2,2)= P;
end
```

Alors, la définition des options se fait comme :

```
Opts=bvpset('Fjacobian',@jac,'RelTol',0.1,'AbsTol',0.1,'Stats','on')
```

Utilisant maintenant la fonction MATLAB **bvpinit** pour la *création d'une estimation initiale de la solution*,

```
tmesh=linspace(a, b, n) % n est le nombre de discrétisation
solinit= bvpinit(tmesh, [a,b]);
```

La résolution de l'équation différentielle peut être par l'une des deux fonctions MATLAB suivantes.

```
Sol4c=bvp4c(@bvpfcn,@bcfcn, solinit, Opts);
```

ou,

```
Sol5c=bvp4c(@bvpfcn,@bcfcn, solinit, Opts);
```

L'affichage des résultats se fait par :

```
plot(Sol4c.x,Sol4c.y(1,:),'r*')
```

ou par :

```
plot(Sol5c.x,Sol5c.y(1,:),'o')
```

A titre d'exemple soit le problème aux limites :

$$\begin{cases} \dfrac{d^2u}{dt^2}=\dfrac{3}{t}\dfrac{du}{dt}-\dfrac{4}{t^2}u+\dfrac{\ln t}{t^2} \\ u(1)=\dfrac{1}{4},\ u(3)=\dfrac{\ln 3+1}{4} \end{cases} \Rightarrow \begin{cases} \dfrac{du}{dt}=v \\ \dfrac{dv}{dt}=\dfrac{3}{t}v-\dfrac{4}{t^2}u+\dfrac{\ln t}{t^2} \end{cases}$$

Avec MATLAB, soit les fonctions **bvpfcn**, **bcfcn** et **jac** propre à l'équation différentielle et ses valeurs aux limites :

```
function dudt = bvpfcn(t,u)
 dudt(1)=u(2);
 dudt(2)=(3/t)*u(2)-(4/(t^2))*u(1)+(log(t)/(t^2));
end
```

```
function res = bcfcn(ua,ub)
 res(1)= ua(1)-(1/4);
 res(2)= ub(1)-((1+log(3))/4);
end
```

```
function dfdu = jac(t, u)
 dfdu(1,1)= 0; dfdu(1,2)= 1;
 dfdu(2,1)= -4/(t^2); dfdu(2,2)= 3/t;
end
```

Soit le script suivant permettant donc de résoudre le problème :

```
Opts=bvpset('Fjacobian',@jac,'RelTol',0.1,'AbsTol',0.1,'Stats','on');
t = linspace(1, 3,20);
solinit = bvpinit(t, [1,3]);
Solution=bvp4c(@bvpfcn,@bcfcn, solinit, Opts);
plot(Solution.x,Solution.y(1,:),'b-*')
```

L'exécution du script permet d'avoir dans le "Command Windows" de MATLAB le résumé et la solution sous forme graphique (Fig. 6.36):

```
The solution was obtained on a mesh of 20 points.
The maximum residual is  1.052e-05.
There were 117 calls to the ODE function.
There were 10 calls to the BC function.
```

bvp4c et **bvp5c** sont basés sur la méthode des différences finies dont le code implémente la formule de Lobatto IIIa à trois pas et quatre pas respectivement qui sont des méthodes de collocation par points (Kierzenka et Shampine, 2001).

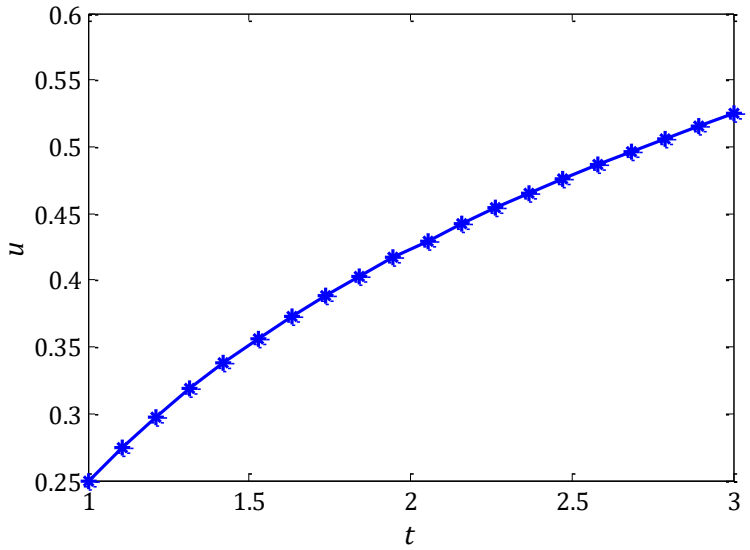


Fig. 6.36 : Résolution avec le solveur **bvp4c**.

12. Conclusion

La résolution numérique des équations différentielles ordinaires (ou un système d'équations différentielles) dépend de la nature des conditions imposées, initiales ou aux limites.

Pour un problème aux conditions initiales dit aussi problème de Cauchy, seules les méthodes à base de Runge-Kutta et multi-pas qui peuvent être utilisées pour déterminer une solution approchée. Mais lors de l'application de ces méthodes numériques, deux catégories de problèmes s'imposent, ceux qui sont raides et non raides. Pour un problème non raide, tous les méthodes numériques d'ordre élevés que ce soit explicites ou implicites peuvent permettant d'avoir une bonne approximation vers la solution réelle. Par contre dans un problème raide, seule les méthodes implicites stable qu'il faut utilisée.

Pour les problèmes aux limites (Dirichlet ou de Neumann), deux types de méthodes sont utilisées à savoir, la méthode de tir et la méthode des différences finis qui sont utilisée pour déterminer une solution approchée du problème posé. La seule difficulté de l'emploi de ces méthodes réside surtout sur la non linéarité du problème.

Plusieurs solveurs de MATLAB permet de résoudre ces problèmes que ce soit initiales et aux limites sont très intéressants et leurs utilisations se fait aussi avec précautions dans certains problèmes.

13. Exercices résolus

Exercice 1

En utilisant la méthode d'Euler pour h = 0.1, résoudre le problème de Cauchy suivant :

$$\begin{cases} u' = -2u + t^2 - 3t, \quad 0 \le t \le 1 \\ u(0) = 1. \end{cases}$$

La méthode d'Euler pour la résolution de l'équation différentielle est :

$$u_{n+1} = u_n + hf(t_n,\ u_n)$$

Ici $f(t_n,\ u_n) = -2u_n + t_n^2 - 3t_n$ par conséquent, soit le processus récursif :

$$u_{n+1} = u_n + h(-2u_n + t_n^2 - 3t_n),\ \ u_1 = 1$$

La discrétisation de l'intervalle [0, 1] avec le pas h = 0.1, conduit à $0 = t_1 < t_2 < t_3 < \ldots < t_{11} = 1$. La résolution de l'équation différentielle consiste à calculer pour chaque valeur t_η la valeur u_η. Sachant que $u_1 = 1$, alors,

$$u_2 = u_1 + h(-2u_1 + t_1^2 - 3t_1) = 1 + 0.1(-2\times 1) = 0.8$$
$$u_3 = u_2 + h(-2u_2 + t_2^2 - 3t_2) = 0.8 + 0.1\times(-2\times 0.8 + 0.1^2 - 3\times 0.1) = 0.611$$
$$u_4 = u_3 + h(-2u_3 + t_3^2 - 3t_3) = 0.611 + 0.1\times(-2\times 0.611 + 0.2^2 - 3\times 0.2) = 0.433$$
$$u_5 = u_4 + h(-2u_4 + t_4^2 - 3t_4) = 0.433 + 0.1\times(-2\times 0.433 + 0.3^2 - 3\times 0.3) = 0.265$$
$$u_6 = u_5 + h(-2u_5 + t_5^2 - 3t_5) = 0.265 + 0.1\times(-2\times 0.265 + 0.4^2 - 3\times 0.4) = 0.108$$
$$u_7 = u_6 + h(-2u_6 + t_6^2 - 3t_6) = 0.108 + 0.1\times(-2\times 0.108 + 0.5^2 - 3\times 0.5) = -0.039$$
$$u_8 = u_7 + h(-2u_7 + t_7^2 - 3t_7) = -0.039 + 0.1\times(2\times 0.039 + 0.6^2 - 3\times 0.6) = -0.175$$
$$u_9 = u_8 + h(-2u_8 + t_8^2 - 3t_8) = -0.175 + 0.1\times(2\times 0.175 + 0.7^2 - 3\times 0.7) = -0.301$$
$$u_{10} = u_9 + h(-2u_9 + t_9^2 - 3t_9) = -0.301 + 0.1\times(2\times 0.301 + 0.8^2 - 3\times 0.8) = -0.417$$
$$u_{11} = u_{10} + h(-2u_{10} + t_{10}^2 - 3t_{10}) = -0.417 + 0.1\times(2\times 0.417 + 0.9^2 - 3\times 0.9) = -0.522$$

Exercice 2

Soit l'équation différentielle : $\dfrac{du}{dt} = 2t - 3u$ avec $u(1) = 4$ comme condition initiale.

Pour résoudre cette équation, en utilise une des méthodes de Runge-Kutta d'ordre 2 dite aussi la méthode de Heun,

$$u_{n+1}=u_n+\frac{h}{2}f\left(t_n,\ u_n\right)+\frac{h}{2}f\left(t_n+h,\ u_n+hf\left(t_n,u_n\right)\right)$$

1- Que signifié *h* ?

2- Expliciter l'expression de u_{n+1} en fonction de u_n, t_n et *h*.

3- Ecrire un script MATLAB permettant de résoudre l'équation différentielle dans l'intervalle [1, 5] pour un choix de *h* de] 0, 1].

1. *h* est le pas de discrétisation est fixé et généralement choisie de l'intervalle]0, 1].

2. La méthode de Heun ainsi explicitée par la formule :

$$u_{n+1}=u_n+\frac{h}{2}f\left(t_n,\ u_n\right)+\frac{h}{2}f\left(t_n+h,\ u_n+hf\left(t_n,u_n\right)\right)$$

Est une méthode de Runge-Kutta d'ordre 2 c'est-à-dire elle peut être formulée comme suit :

$$k_1=hf\left(t_n,\ u_n\right)$$

$$k_2=hf\left(t_n+h,\ u_n+k_1\right)$$

et,

$$u_{n+1}=u_n+\frac{1}{2}\left(k_1+k_2\right)$$

alors,

$$k_1=hf\left(t_n,\ u_n\right)=h\left(2t_n-3u_n\right)$$

$$k_2=hf\left(t_n+h,\ u_n+k_1\right)=h\left(2\left(t_n+h\right)-3\left(u_n+k_1\right)\right)$$

$$k_2=h\left(2t_n+2h-3u_n-3k_1\right)=h\left(2t_n+2h-3u_n-3h\left(2t_n-3u_n\right)\right)=h\left(2\left(1-3h\right)t_n-3\left(1-3h\right)u_n+2h\right)$$

$$\frac{k_1+k_2}{h}=2t_n-3u_n+2t_n+2h-3u_n-6ht_n+9hu_n=2\left(2-3h\right)t_n-3\left(2-3h\right)u_n+2h$$

$$\frac{k_1+k_2}{2}=h\left[\left(2-3h\right)t_n-\frac{3}{2}\left(2-3h\right)u_n+h\right]$$

finalement

$$u_{n+1}=u_n+\frac{k_1+k_2}{2}=h\left(2-3h\right)t_n+\left[1-\frac{3h}{2}\left(2-3h\right)\right]u_n+h^2$$

ou,

$$u_{n+1}=h\left(2-3h\right)t_n+\left[1-\frac{3h}{2}\left(2-3h\right)\right]u_n+h^2$$

On remarque que si on choisit $h=2/3$, le processus devient $u_{n+1}=u_n+4/9$ qui est une suite numérique de raison $4/9$ n'a aucun lien avec l'évolution de *t*. De même, si $h\geq 2/3$, le processus donne de mauvaise résolution (problème de stabilité).

3. Soit le script MATLAB, pour résoudre l'équation différentielle.

```
f=@(t,u) 2*t-3*u; % Définition de la fonction f(t,y)
h=0.01; t=1:h:5;
u(1)=4; % Définition de la condition initiale
for n=1:numel(t)-1
 k1=h*f(t(n),u(n)); k2=h*f(t(n)+h,u(n)+k1);
 u(n+1)=u(n)+(1/2)*(k1+k2);
end
plot(t,u,'*b');
```

Exercice 3

La fonction erreur $\text{erf}(t)$ est définie par l'intégrale suivant :

$$\text{erf}(t)=\frac{2}{\sqrt{\pi}}\int_0^t e^{-x^2}dx$$

Mais aussi peut être définie comme une solution de l'équation différentielle :

$$\frac{du}{dt}=\frac{2}{\sqrt{\pi}}e^{-t^2},\ u(0)=0$$

Utiliser le solveur **`ode23tb`** pour résoudre cette équation pour $0\le t\le 2$ et comparer la solution obtenue avec le solveur **`ode23tb`** et les résultats de la fonction **`erf`** programmée dans MATLAB.

Dans la fenêtre "Command Window" de MATLAB on applique les commandes suivantes :

```
>> f=@(t,u) (2/sqrt(pi))*exp(-t^2); % Définition de la fonction
>> [t,u]=ode23tb(f,[0,2],0); % Application du solveur ode23tb
>> erf(t); % Calcul de erf(t) ou t est un vecteur
```

Qui permettent d'avoir les résultats explicités dans le tableau ci-dessous.

D'après le tableau l'erreur absolue $|u_i - \text{erf}(t_i)|$ à chaque ordre de calcul est de l'ordre de 10^{-3} au maximum, ce qui montre bien que le solveur de MATLAB **`ode23tb`** est peut-être utilisée pour calculer numériquement de telle intégrale.

Ordre i	t_i	u_i	$\text{erf}(t_i)$	$\lvert u_i - \text{erf}(t_i)\rvert$
1	0.0000	0.0000	0.0000	0.0000
2	0.0954	0.1072	0.1073	0.0001
3	0.1907	0.2125	0.2126	0.0001
4	0.2861	0.3140	0.3142	0.0002
5	0.3815	0.4102	0.4104	0.0002
6	0.5122	0.5308	0.5311	0.0003
7	0.6725	0.6579	0.6584	0.0005
8	0.8725	0.7823	0.7827	0.0004
9	1.0725	0.8705	0.8707	0.0002
10	1.2725	0.9282	0.9281	0.0001
11	1.4725	0.9632	0.9627	0.0005
12	1.6725	0.9827	0.9820	0.0007
13	1.8725	0.9928	0.9919	0.0009
14	2.0000	0.9962	0.9953	0.0009

Exercice 4

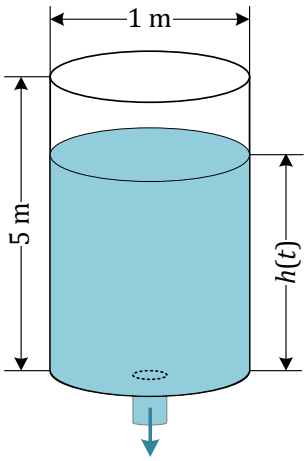
Soit un réservoir cylindrique de section de base de diamètre de 1 m, la hauteur d'eau dans le réservoir été de 5 m. En utilisant la méthode d'Euler $(\Delta h=1/3)$, calculer le temps de vidange de réservoir par un orifice circulaire de diamètre de 2 cm, situé au fond du réservoir. On suppose que la vitesse de l'écoulement à l'orifice est exprimée par $v=0.6\sqrt{2gh(t)}$.

En premier lieu il faut établir l'équation différentielle de vidange du réservoir. D'après l'équation de continuité, Il est clair que pour un temps *dt*, la quantité d'eau descendue d'un niveau *dh* est la même qui sort par l'orifice qui existe au fond du réservoir, c'est-à-dire :

$$Sdh(t) = 0.6s\sqrt{2gh(t)}dt$$

qui peut être se transformer en :

$$\frac{dt}{dh} = \frac{S}{s}\frac{1}{0.6\sqrt{2gh}} = f(h,\ t)$$



L'application de la méthode d'Euler à cette équation conduit à :

$$t_{n+1} = t_n + \Delta h f(h_n,\ t_n) \text{ c'est-à-dire, } t_{n+1} = t_n + \frac{S}{s}\frac{\Delta h}{0.6\sqrt{2gh_n}}$$

A $t_1 = 0$ la hauteur dans le réservoir est à $h_1 = 5$ mètres et à l'instant t_2 correspond à la hauteur $h_2 = 5 - \Delta h$ ce qui implique aussi $h_3 = 5 - 2\Delta h,\ \ldots,\ h_n = 5 - (n-1)\Delta h$.

Si $\Delta h = 1/3$,

$$h_n = 5 - \frac{(n-1)}{3} = \frac{16-n}{3}$$

or,

$$\frac{S}{s} = 2500$$

Alors, le calcul des temps $t_1,\ t_2,\ \ldots,\ t_{15}$ en secondes :

$$t_{n+1} = t_n + \frac{2500}{1.8\sqrt{2g}}\left(\frac{3}{16-n}\right)^{\frac{1}{2}}$$

Avec g = 9.81 m/s^2, le tableau des résultats ci-contre montre que le temps de vidange du réservoir correspond à t_{16} est :

$$T = t_{16} = 3656.76\,\text{s} = 60.9\,\text{min}$$

i	h_i (m)	t_i (s)
1	5	0
2	4.67	145.15
3	4.33	295.78
4	4.00	452.56
5	3.67	616.31
6	3.33	788.05
7	3.00	969.08
8	2.67	1161.10
9	2.33	1366.37
10	2.00	1588.09
11	1.67	1830.97
12	1.33	2102.52
13	1.00	2416.08
14	0.67	2800.11
15	0.33	3343.20
16	0.00	3656.76

La détermination du temps de vidange du réservoir peut être déterminé par la résolution directe de l'équation différentielle :

$$dt = \frac{S}{s}\frac{dh}{0.6\sqrt{2gh}}$$

Le temps de vidange T est obtenue en intégrant l'équation entre 0 et H pour la hauteur, c'est-à-dire :

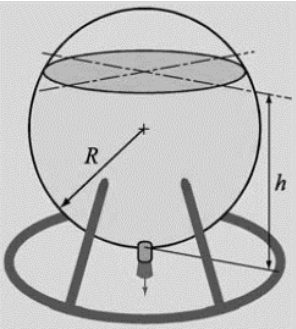
$$\int_0^T dt = \frac{S}{s}\frac{1}{0.6\sqrt{2g}}\int_0^H \frac{dh}{\sqrt{h}} \ \Rightarrow\ T = \frac{S}{s}\frac{2\sqrt{H}}{0.6\sqrt{2g}}$$

Numériquement,

$$T = \frac{2500\times 2\times\sqrt{5}}{0.6\sqrt{2\times 9.81}} = 4206.82\ \text{s} = 70.1\,\text{min}$$

Exercice 5

Un réservoir sphérique de rayon R = 4 m ouvert en haut, se vide à travers une orifice située dans le fond de rayon r = 0.02 m. le niveau h de l'eau dans le réservoir est obtenu à partir de l'équation différentielle :



$$\frac{dh}{dt}=-\frac{r^2\sqrt{2gh}}{2hR-h^2}$$

avec, g = 9.81m/s^2. Si a l'instant initiale $t = 0$ le niveau de l'eau est à $h=6.5$ m. Calculer le temps de vidange pour que le niveau d'eau dans le réservoir soit à $h=0.5$ m en utilisant la méthode d'Euler et la méthode de Runge-Kutta d'ordre 4.

De l'équation,

$$\frac{dh}{dt}=-\frac{r^2\sqrt{2gh}}{2hR-h^2}\ \Rightarrow\ \frac{dt}{dh}=-\frac{2hR-h^2}{r^2\sqrt{2gh}}=f(h,\ t)\equiv f(h)$$

alors,

$$t_{n+1}=t_n+\int_{h_n}^{h_{n+1}} f(h)dh$$

D'après la méthode d'Euler,

$$t_{n+1}=t_n+\Delta h f(h_n)=t_n-\frac{2h_nR-h_n^2}{r^2\sqrt{2gh_n}}\Delta h$$

D'après la méthode de Runge-Kutta d'ordre 4,

$$t_{n+1}=t_n+\frac{1}{6}(k_1+2k_2+2k_3+k_4)$$

avec,

$$k_1=\Delta h f(h_n)=-\frac{2h_nR-h_n^2}{r^2\sqrt{2gh_n}}\Delta h$$

$$k_2=k_3=\Delta h f\left(h_n+\frac{\Delta h}{2}\right)=-\frac{2\left(h_n+\frac{\Delta h}{2}\right)R-\left(h_n+\frac{\Delta h}{2}\right)^2}{r^2\sqrt{2g\left(h_n+\frac{\Delta h}{2}\right)}}\Delta h$$

$$k_4=\Delta h f(h_n+\Delta h)=-\frac{2(h_n+\Delta h)R-(h_n+\Delta h)^2}{r^2\sqrt{2g(h_n+\Delta h)}}\Delta h$$

Sachant que, R = 4 m, r = 0.02 m, t_1 = 0, h_1 = 6.5 m et g = 9.81m/s^2. En choisi par exemple comme pas de discrétisation $\Delta h=-0.5$ m. Soit le script MATLAB :

```
R=4;r=0.02;g=9.81;
Deltah=-0.5; %Pas de discrétisation
f=@(h)(-2*R*h+h.*h)./(r*r*sqrt(2*g*h));
h=6.5:Deltah:0.5; te(1)=0; trk4(1)=0;
for n=1:numel(h)-1
  te(n+1)=te(n)+Deltah*f(h(n));
```

```
  k1=Deltah*f(h(n)); k2=Deltah*f(h(n)+0.5*Deltah);
  k3=k2; k4=Deltah*f(h(n)+Deltah);
 trk4(n+1)=trk4(n)+(1/6)*(k1+2*k2+2*k3+k4);
end
%Temps de Vidange en heure
Temps2VidangeParEuler=te(numel(h))/3600;
Temps2VidangeParRK4=trk4(numel(h))/3600;
```

L'exécution de ce script permet d'avoir le temps de vidange est de 6.74 heures par la méthode d'Euler et de 6.82 heures par la méthode de Runge-Kutta d'ordre 4.

Exercice 6

Utiliser la méthode de Runge-Kutta d'ordre 4 avec h=0.1 pour déterminer $u(0.5)$ sachant que $u(t)$ est la solution du problème de Cauchy :

$$\begin{cases} u' = (t+u-1)^2, \\ u(0) = 2. \end{cases}$$

L'algorithme de Runge-Kutta d'ordre 4 appliqué à cette équation est donné par :

$$k_1 = h(t_n + u_n - 1)^2, \quad k_2 = h\left(t_n + \frac{1}{2}h + u_n + \frac{1}{2}k_1 - 1\right)^2,$$

$$k_3 = h\left(t_n + \frac{h}{2} + u_n + \frac{1}{2}k_2 - 1\right)^2, \quad k_4 = h(t_{n+1} + u_n + k_3 - 1)^2,$$

$$u_{n+1} = u_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad u_1 = 2.$$

Le script MATLAB ci-dessous permet d'avoir les résultats représentés dans le tableau ci-dessous.

```
% Définition de la fonction f(t,u), le pas h et l'intervalle de solution
f=@(t,u) (t+u-1)^2 ; h=0.1; t=0:h:0.5;
u(1)=2; % Définition de la condition initiale
for n=1:numel(t)-1
 k1=h*f(t(n),u(n));K1(n)=k1; k2=h*f(t(n)+(h/2),u(n)+(k1/2));
 K2(n)=k2; k3=h*f(t(n)+(h/2),u(n)+(k2/2));K3(n)=k3;
 k4=h*f(t(n+1),u(n)+k3);K4(n)=k4;
 u(n+1)=u(n)+(k1+2*k2+2*k3+k4)/6;
end
```

N	1	2	3	4	5	6
t_n	0	0.1	0.2	0.3	0.4	0.5
k_1	-	0.100	0.150	0.228	0.359	0.608
k_2	-	0.121	0.182	0.280	0.452	0.794
k_3	-	0.123	0.186	0.288	0.472	0.848
k_4	-	0.150	0.228	0.360	0.609	1.165
u_n	2	2.123	2.308	2.596	3.065	3.908

D'après les résultats obtenus $u(0.5) = u_6 = 3.908$.

Exercice 7

Utiliser la méthode d'Adams-Moulton (AM4) avec h = 0.01 pour approcher l'intégrale :

$$\mathrm{Si}(x)=\int_0^x \frac{\sinh(t)}{t}dt$$

Pour x = 0.5, 1.0, 1.5 et 2.0. Comparer les résultats avec la fonction **`sinhint`** de MATLAB.

Considérer $\sinh(t)/t=1$ pour t = 0.

La méthode d'Adams-Moulton (AM4) est explicitée par :

$$u_{n+1}=u_n+\frac{h}{24}\left[9f(t_{n+1})+19f(t_n)-5f(t_{n-1})+f(t_{n-2})\right]$$

Pour calculer u_n, il faut connaître u_0, u_1, u_2 et u_3 qui peuvent être calculer par la méthode de Runge-Kutta d'ordre 4 (RK4) :

$$u_{n+1}=u_n+\frac{h}{6}\left[f(t_n)+4f\left(t_n+\frac{h}{2}\right)+f(t_{n+1})\right],\quad u_0=0$$

Soit la fonction **`Shi`** qui permet de calculer l'intégrale de la fonction $\mathrm{Si}(x)$.

```
function Int2Shi=Shi(x,h)
f=@(t)sinh(t)/t;
if nargin<2, h=0.01; end
t=0:h:x; u(1)=0;
for n=1:2
 if n==1
  u(n+1)=u(n)+(h/6)*(1+4*f(t(n)+(h/2))+f(t(n+1)));
 else
  u(n+1)=u(n)+(h/6)*(f(t(n))+4*f(t(n)+(h/2))+f(t(n+1)));
 end
end
for n=3:numel(t)-1
 if n==3
  u(n+1)=u(n)+(h/24)*(9*f(t(n+1))+19*f(t(n))-5*f(t(n-1))+1);
 else
  u(n+1)=u(n)+(h/24)*(9*f(t(n+1))+19*f(t(n))-5*f(t(n-1))+f(t(n-2)));
 end
end
Int2Shi=u(numel(t));
end
```

L'application des fonctions **`Shi`** et **`sinhint`** conduisent à :

```
>> Shi(0.5)
ans =
    0.5070
>> Shi(1)
ans =
    1.0573
>> Shi(1.5)
```

```
>> sinhint(0.5)
ans =
    0.5070
>> sinhint(1.0)
ans =
    1.0573
>> sinhint(1.5)
```

```
ans =
    1.7007
>> Shi(2.0)
ans =
    2.5016
```

```
ans =
    1.7007
>> sinhint(2.0)
ans =
    2.5016
```

D'après les résultats obtenus, la méthode d'Adams-Moulton (AM4) est peut-être un moyen efficace pour le calcul d'intégration.

Exercice 8

Utiliser la méthode d'Euler (h = 0.1) pour calculer $y(0.2)$ ou $y(t)$ est la solution de l'équation différentielle : $y''+ty'+y=0$ avec, $y(0)=1$ et $y'(0)=2$.
Résoudre la même équation par la méthode RK4 dans l'intervalle [0, 10] avec un script MATLAB.

L'équation différentielle

$$y''+ty'+y=0 \;\Rightarrow\; y''=-ty'-y$$

soit,

$$\frac{dy}{dt}=z \;\text{ alors }\; \frac{dz}{dt}=-tz-y$$

d'après ces deux équations différentielles, la méthode d'Euler peut être appliquée ainsi :

$$y_{n+1}=y_n+hz_n,\quad z_{n+1}=z_n+h(-t_nz_n-y_n) \;\text{ avec, }\; y_1=1,\; z_1=2$$

supposons h = 0.1, soit :

$$y_2=y_1+hz_1=1+0.1\times 2=1.2,\quad z_2=z_1+h(-t_1z_1-y_1)=2+0.1\times(-0\times 2-1)=1.9$$

$$y_3=y_2+hz_2=1.2+0.1\times 1.9=1.39,\quad z_3=z_2+h(-t_2z_2-y_2)=1.9+0.1\times(-0.1\times 1.9-1.2)=1.761$$

alors,

$$y(0.2)\approx 1.39$$

L'équation différentielle $y''+ty'+y=0$ peut être s'écrit sous forme vectorielle :

$$\begin{cases}\dfrac{dy}{dt}=z,\\[2mm] \dfrac{dz}{dt}=-tz-y.\end{cases}\Rightarrow\begin{pmatrix}\dfrac{dy}{dt}\\[2mm]\dfrac{dz}{dt}\end{pmatrix}=\begin{pmatrix}z\\-tz-y\end{pmatrix}\Rightarrow\frac{d}{dt}\begin{pmatrix}y\\z\end{pmatrix}=\begin{pmatrix}z\\-tz-y\end{pmatrix}=\begin{pmatrix}f_1(t,y,z)\\f_2(t,y,z)\end{pmatrix}\Rightarrow\frac{d\mathbf{u}}{dt}=\mathbf{f}(t,\mathbf{u})$$

La condition initiale est donc explicitée par le vecteur $\mathbf{u}_1=(1,\;2)^T$, l'algorithme de RK4 est :

$$\mathbf{k}_1=h\mathbf{f}(t_n,\;\mathbf{u}_n),\;\mathbf{k}_2=h\mathbf{f}\left(t_n+\frac{h}{2},\;\mathbf{u}_n+\frac{1}{2}\mathbf{k}_1\right),\;\mathbf{k}_3=h\mathbf{f}\left(t_n+\frac{h}{2},\;\mathbf{u}_n+\frac{1}{2}\mathbf{k}_2\right),\;\mathbf{k}_4=h\mathbf{f}(t_{n+1},\;\mathbf{u}_n+\mathbf{k}_3)$$

$$\mathbf{u}_{n+1}=\mathbf{u}_n+\frac{1}{6}(\mathbf{k}_1+2\mathbf{k}_2+2\mathbf{k}_3+\mathbf{k}_4)$$

$$\mathbf{k}_i=(k_i,\;K_i)^T,\;i=1,\;2,\;3,\;4$$

qui peut être s'écrit analytiquement :

$$k_1=hf_1(t_n,\;y_n,\;z_n),\quad K_1=hf_2(t_n,\;y_n,\;z_n),$$

$$k_2 = hf_1\left(t_n + \frac{h}{2},\ y_n + \frac{1}{2}k_1,\ z_n + \frac{1}{2}K_1\right), \quad K_2 = hf_2\left(t_n + \frac{h}{2},\ y_n + \frac{1}{2}k_1,\ z_n + \frac{1}{2}K_1\right),$$

$$k_3 = hf_1\left(t_n + \frac{h}{2},\ y_n + \frac{1}{2}k_2\ z_n + \frac{1}{2}K_2\right), \quad K_3 = hf_2\left(t_n + \frac{h}{2},\ y_n + \frac{1}{2}k_2\ z_n + \frac{1}{2}K_2\right),$$

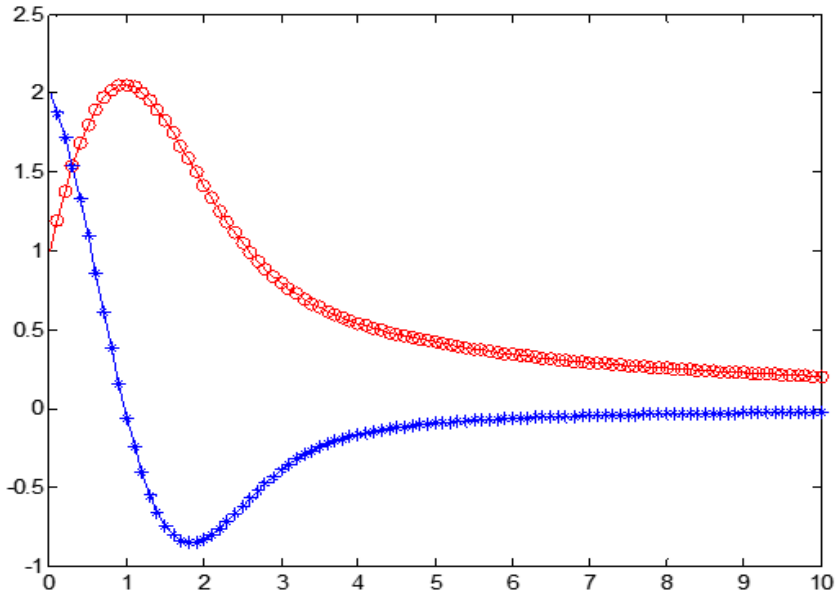
$$k_4 = hf_1\left(t_n + h,\ y_n + k_3,\ z_n + K_3\right), \quad K_4 = hf_2\left(t_n + h,\ y_n + k_3,\ z_n + K_3\right).$$

$$y_{n+1} = y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \quad z_{n+1} = z_n + \frac{1}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right)$$

Soit le script MATLAB qui permet de résoudre cette équation pour l'intervalle [0, 10] :

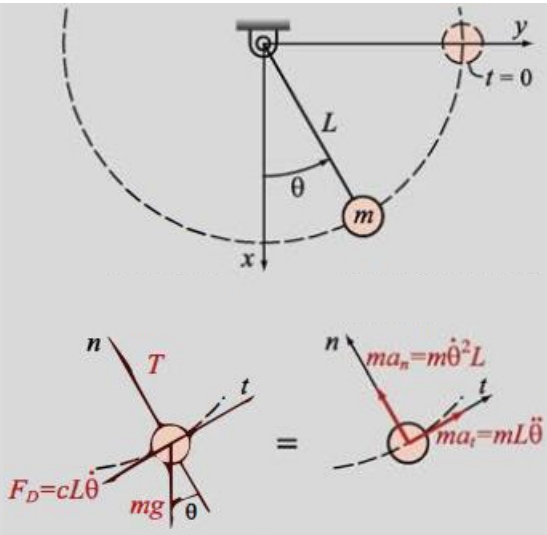
```
%Définition des fonctions f1(t,y,z) et f2(t,y,z)
f1=@(t,y,z) z; f2=@(t,y,z) -t*z-y;
h=0.1;  t=0:h:10;
% Définition des conditions initiales
y(1)=1; z(1)=2;
for n=1:numel(t)-1
 k1=h*f1(t(n),y(n),z(n));
 K1=h*f2(t(n),y(n),z(n));
 k2=h*f1(t(n)+(h/2),y(n)+(k1/2),z(n)+(K1/2));
 K2=h*f2(t(n)+(h/2),y(n)+(k1/2),z(n)+(K1/2));
 k3=h*f1(t(n)+(h/2),y(n)+(k2/2),z(n)+(K2/2));
 K3=h*f2(t(n)+(h/2),y(n)+(k2/2),z(n)+(K2/2));
 k4=h*f1(t(n+1),y(n)+k3,z(n)+K3);
 K4=h*f2(t(n+1),y(n)+k3,z(n)+K3);
y(n+1)=y(n)+(1/6)*(k1+2*k2+2*k3+k4);
z(n+1)=z(n)+(1/6)*(K1+2*K2+2*K3+K4);
end
  plot(t,y,'-or',t,z,'-*b');
```

Dans la représentation graphique de la solution $y(t)$ (en rouge) et la fonction dérivée $z(t)$ (en bleu).



Exercice 9

Un pendule de masse m attachée à une tige rigide. D'après la deuxième loi de Newton, lorsque le pendule oscille, la somme des forces agissant sur la masse est égal à la masse multipliée par l'accélération. L'écriture de l'équation d'équilibre dans la direction tangentielle est exprimée par :



$$\sum F_t = -cL\frac{d\theta}{dt} - mg\sin\theta = mL\frac{d^2\theta}{dt^2}$$

θ est l'angle d'inclinaison du pendule,
c = 0.16 N.s.m^{-1} le coefficient d'amortissement,
m = 0.5 kg, L = 1.2 m et g = 9.81 m.s^{-2}.

Cette équation peut être réécrite comme:

$$\frac{d^2\theta}{dt^2} = -\frac{c}{m}\frac{d\theta}{dt} - \frac{g}{L}\sin\theta$$

À l'instant t = 0, l'angle initiale est θ = 90° et la vitesse est $d\theta/dt = 0$. Déterminer l'angle du pendule en fonction du temps $\theta(t)$, pendant les 18 premières secondes après son relâchement.

Pour résoudre l'équation soit le changement de variable

$$w = \frac{d\theta}{dt} \quad \Rightarrow \quad \frac{dw}{dt} = \frac{d^2\theta}{dt^2}$$

Par conséquent l'équation différentielle de deuxième ordre s'écrit sous forme d'un systèmes d'équations différentielle de première ordre :

$$\frac{d\theta}{dt} = w \text{ avec, } \theta(0) = \frac{\pi}{2} \text{ la première condition initiale}$$

$$\frac{dw}{dt} = \frac{c}{m}w - \frac{g}{L}\sin\theta \text{ avec, } w(0) = 0 \text{ la première deuxième condition initiale}$$

Soit l'implémentation du problème sous forme d'une fonction **`Sys2EDOsRK4`** permet de déterminer une solution numérique de l'équation différentielle par la méthode de Runge-Kutta d'ordre 4.

```
function [t,u,v] = Sys2EDOsRK4(f,a,b,u0,v0,h)
% Sys2EDOsRK4 résoudre l'équation différentielle par la méthode de Runge-
% Kutta d'ordre 4.
% La variable indépendante est t, les variables dépendantes sont u et v
% f est le nom de la fonction qui calcul du/dt=v.
% a la première valeur de l'intervalle du temps t.
% b la dernière valeur de l'intervalle du temps t.
% u0 et v0 sont les condition initiales
% h le pas de discrétisation.
if nargin<6, h=0.01; end
t=a:h:b; u(1)=u0; v(1)=v0;
```

```
for n=1:numel(t)-1
 k1=h*v(n); K1=h*f(t(n),u(n),v(n));
 k2=h*(v(n)+(K1/2)); K2=h*f(t(n)+(h/2),u(n)+(k1/2),v(n)+(K1/2));
 k3=h*(v(n)+(K2/2)); K3=h*f(t(n)+(h/2),u(n)+(k2/2),v(n)+(K2/2));
 k4=h*(v(n)+K3); K4=h*f(t(n+1),u(n)+k3,v(n)+K3);
 u(n+1)=u(n)+(1/6)*(k1+2*k2+2*k3+k4);
 v(n+1)=v(n)+(1/6)*(K1+2*K2+2*K3+K4);
end
end
```

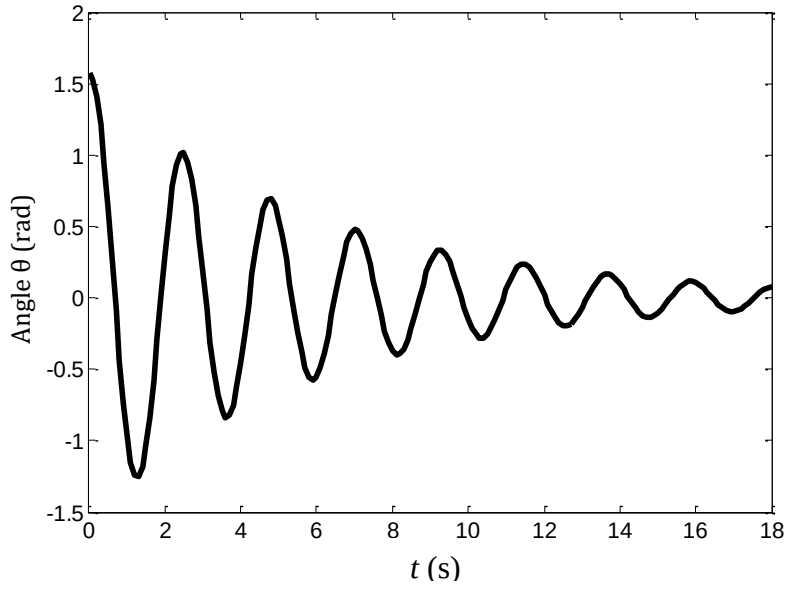
La fonction **f** dans **Sys2EDOsRK4** est donnée par l'expression :

$$f(t,\ \theta,\ w)=\frac{c}{m}w-\frac{g}{L}\sin\theta$$

Alors avec MATLAB,

```
>> c = 0.16; m = 0.5; g = 9.81; L = 1.2;
>> f=@(t,u,v) -(c/m)*v-(g/L)*sin(u);
>> [t,u,v] = Sys2EDOsRK4(f,0,18,pi/2,0,0.1);
```

qui donne après l'exécution le graphe de la fonction $\theta(t)$.



Exercice 10

Utiliser la méthode Runge-Kutta d'ordre 4 pour résoudre les systèmes différentiels suivants pour $0\le t\le 1$.

$$(a)\ \begin{cases}\dfrac{dx}{dt}=6x+y+6t,\\[2mm] \dfrac{dy}{dt}=4x+3y-10t+4.\end{cases}\qquad x(0)=0.5,\ \ y(0)=0.2$$

$$(b)\ \begin{cases}\dfrac{dx}{dt}+\dfrac{dy}{dt}=4t,\\[2mm] -\dfrac{dx}{dt}+\dfrac{dy}{dt}+y=6t^2+10.\end{cases}\qquad x(0)=3,\ \ y(0)=-1$$

Choisir h = 0.1

(*a*) Ce système différentiel peut être s'écrit :

$$\begin{cases} \dfrac{dx}{dt} = 6x + y + 6t = f(t,\ x,\ y) \\ \dfrac{dy}{dt} = 4x + 3y - 10t + 4 = g(t,\ x,\ y) \end{cases}$$

par conséquent soit l'algorithme de RK4 :

$$k_1 = hf(t_n,\ x_n,\ y_n), \quad K_1 = hg(t_n,\ x_n,\ y_n)$$

$$k_2 = hf\left(t_n + \frac{h}{2},\ x_n + \frac{1}{2}k_1,\ y_n + \frac{1}{2}K_1\right), \quad K_2 = hg\left(t_n + \frac{h}{2},\ x_n + \frac{1}{2}k_1,\ y_n + \frac{1}{2}K_1\right)$$

$$k_3 = hf\left(t_n + \frac{h}{2},\ x_n + \frac{1}{2}k_2,\ y_n + \frac{1}{2}K_2\right), \quad K_3 = hg\left(t_n + \frac{h}{2},\ x_n + \frac{1}{2}k_2,\ y_n + \frac{1}{2}K_2\right)$$

$$k_4 = hf(t_n + h,\ x_n + k_3,\ y_n + K_3), \quad K_4 = hg(t_n + h,\ x_n + k_3,\ y_n + K_3)$$

$$u_{n+1} = u_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad v_{n+1} = v_n + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

c'est-à-dire,

$$k_1 = h(6x_n + y_n + 6t_n), \quad K_1 = h(4x_n + 3y_n - 10t_n + 4)$$

$$k_3 = h\left(6x_n + 3k_2 + y_n + \frac{1}{2}K_2 + 6t_n + 3h\right), \quad K_3 = h\left(4x_n + 2k_2 + 3y_n + \frac{3}{2}K_2 - 10t_n + 5h + 4\right)$$

$$k_4 = h(6x_n + 6k_3 + y_n + K_3 + 6t_n + 6h), \quad K_4 = h(4x_n + 4k_3 + 3y_n + 3K_3 - 10t_n + 10h + 4)$$

$$x_{n+1} = x_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad y_{n+1} = y_n + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

Le script MATLAB suivant, permet de déterminer la solution du système différentielle :

```
% Définition de la fonction f(t,y,z)
f=@(t,x,y) 6*x+y+6*t;
% Définition de la fonction g(t,y,z)
g=@(t,x,y) 4*x+3*y-10*t+4;
h=0.1;  t=0:h:1;
% Définition des conditions initiales
x(1)=0.5;
y(1)=0.2;
for n=1:numel(t)-1
k1=h*f(t(n),x(n),y(n));
K1=h*g(t(n),x(n),y(n));
k2=h*f(t(n)+(h/2),x(n)+(k1/2),y(n)+(K1/2));
K2=h*g(t(n)+(h/2),x(n)+(k1/2),y(n)+(K1/2));
k3=h*f(t(n)+(h/2),x(n)+(k2/2),y(n)+(K2/2));
K3=h*g(t(n)+(h/2),x(n)+(k2/2),y(n)+(K2/2));
k4=h*f(t(n+1),x(n)+k3,y(n)+K3);
K4=h*g(t(n+1),x(n)+k3,y(n)+K3);
x(n+1)=x(n)+(1/6)*(k1+2*k2+2*k3+k4);
y(n+1)=y(n)+(1/6)*(K1+2*K2+2*K3+K4);
end
```

i	t_i	x_i	y_i
1	0.0	0.500	0.200
2	0.1	1.021	1.011
3	0.2	2.190	2.359
4	0.3	4.648	4.857
5	0.4	9.675	9.757
6	0.5	19.847	19.595
7	0.6	40.338	39.502
8	0.7	81.552	79.825
9	0.8	164.417	161.425
10	0.9	331.032	326.318
11	1.0	666.092	659.097

(*b*) Ce système différentiel peut être se transforme en :

$$\begin{cases} \dfrac{dx}{dt}+\dfrac{dy}{dt}=4t, \\ -\dfrac{dx}{dt}+\dfrac{dy}{dt}+y=6t^2+10. \end{cases} \Rightarrow \begin{cases} \dfrac{dx}{dt}=\dfrac{1}{2}y-3t^2+2t-5, \\ \dfrac{dy}{dt}=-\dfrac{1}{2}y+3t^2+2t+5. \end{cases}$$

On peut utiliser le même script en remplaçant seulement :

`f=@(t,x,y)6*x+y+6*t;` par `f=@(t,x,y)(1/2)*y-3*t^2+2*t-5;`

`g=@(t,x,y)4*x+3*y-10*t+4;` par `g=@(t,x,y)(-1/2)*y+3*t^2+2*t+5;`

`x(1)=0.5;` par `x(1)=3;`

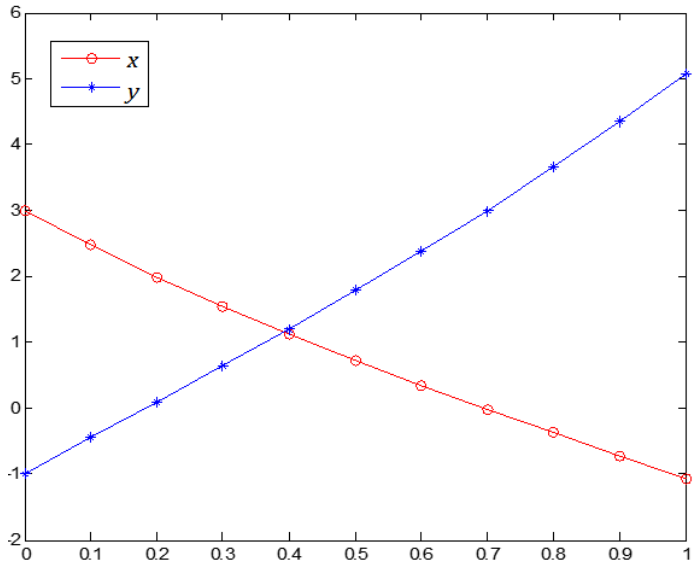
`y(1)=0.2;` par `y(1)=-1;`

Soit le script MATLAB qui permet la résolution de ce système différentielle :

```
% Définition des fonctions f(t,y,z)et g(t,y,z)
f=@(t,x,y) (1/2)*y-3*t^2+2*t-5; g=@(t,x,y) (-1/2)*y+3*t^2+2*t+5;
h=0.1; t=0:h:1;
x(1)=3; y(1)=-1; % Définition des conditions initiales
for n=1:numel(t)-1
   k1=h*f(t(n),x(n),y(n));
    K1=h*g(t(n),x(n),y(n));
     k2=h*f(t(n)+(h/2),x(n)+(k1/2),y(n)+(K1/2));
      K2=h*g(t(n)+(h/2),x(n)+(k1/2),y(n)+(K1/2));
       k3=h*f(t(n)+(h/2),x(n)+(k2/2),y(n)+(K2/2));
      K3=h*g(t(n)+(h/2),x(n)+(k2/2),y(n)+(K2/2));
    k4=h*f(t(n+1),x(n)+k3,y(n)+K3);
   K4=h*g(t(n+1),x(n)+k3,y(n)+K3);
  x(n+1)=x(n)+(1/6)*(k1+2*k2+2*k3+k4);
 y(n+1)=y(n)+(1/6)*(K1+2*K2+2*K3+K4);
end
plot(t,x,'-or',t,y,'-*b');
```

Ci-dessous, le tableau des résultats et le graphe qui illustre la solution numérique par la méthode de Runge-Kutta d'ordre 4.

i	t_i	x_i	y_i
1	0.0	3.000	-1.000
2	0.1	2.473	-0.453
3	0.2	1.987	0.093
4	0.3	1.536	0.644
5	0.4	1.115	1.205
6	0.5	0.719	1.781
7	0.6	0.342	2.378
8	0.7	-0.021	3.001
9	0.8	-0.374	3.654
10	0.9	-0.721	4.341
11	1.0	-1.067	5.067



Exercice 11

La méthode multi-pas suivante est-elle consistante ?

$$u_{n+1}-u_n=\frac{1}{4}h\left(3f_{n+1}-f_n\right)$$

Pour quelles valeurs des paramètres a et b les méthodes multi-pas suivantes sont consistantes ?

1. $u_{n+2}-au_{n+1}-2u_n=hbf_n$
2. $u_{n+2}+u_{n+1}+au_n=h\left(f_{n+2}+bf_n\right)$

La première méthode multi-pas est de l'ordre 1, c'est-à-dire sa forme générale est :

$$\alpha_1u_{n+1}+\alpha_0u_n=h\left(\beta_1f_{n+1}+\beta_0f_n\right) \text{ ou, } \alpha_1=1,\ \alpha_0=-1 \text{ et } \beta_1=\frac{3}{4},\ \beta_0=-\frac{1}{4}$$

suivant les conditions de consistance :

$$\alpha_1+\alpha_0=1-1=0 \text{ et } 1\times\alpha_1+0\times\alpha_0=1\neq\beta_1+\beta_0=\frac{1}{2}$$

cette méthode n'est pas consistante.

Les deux méthodes multi-pas suivantes sont de l'ordre 2, dont la forme générale est :

$$\alpha_2u_{n+2}+\alpha_1u_{n+1}+\alpha_0u_n=h\left(\beta_2f_{n+1}+\beta_1f_{n+1}+\beta_0f_n\right)$$

Pour que cette méthode est consistante, il faut que :

$$\alpha_2+\alpha_1+\alpha_0=0,\ 2\alpha_2+\alpha_1=\beta_2+\beta_1+\beta_0 \text{ et } 4\alpha_2+\alpha_1=4\beta_2+2\beta_1$$

Pour la méthode, $u_{n+2}-au_{n+1}-2u_n=hbf_n$, $\alpha_2=1,\ \alpha_1=-a,\ \alpha_0=-2$ et $\beta_2=\beta_1=0,\ \beta_0=b$,

suivant la première condition de consistance, $\alpha_2+\alpha_1+\alpha_0=0\Leftrightarrow1-a-2=0\Rightarrow a=-1$,

suivant la deuxième condition de consistance, $2\alpha_2+\alpha_1=\beta_2+\beta_1+\beta_0\Leftrightarrow2+1=b\Rightarrow b=3$,

suivant la troisième condition de consistance, $4\alpha_2+\alpha_1=4\beta_2+2\beta_1\Leftrightarrow5=0$ impossible.

Ce qui signifié que cette méthode n'est pas consistante à l'ordre 2, mais elle est consistante à l'ordre 1 pour $a=-1$ et $b=3$.

Pour la méthode, $u_{n+2}+u_{n+1}+au_n=h\left(f_{n+2}+bf_n\right)$, $\alpha_2=\alpha_1=1,\ \alpha_0=a$ et $\beta_2=1,\ \beta_1=0,\ \beta_0=b$,

suivant la première condition de consistance, $\alpha_2+\alpha_1+\alpha_0=0\Leftrightarrow1+1+a=0\Rightarrow a=-2$,

suivant la deuxième condition de consistance, $2\alpha_2+\alpha_1=\beta_2+\beta_1+\beta_0\Leftrightarrow2+1=1+b\Rightarrow b=2$,

suivant la troisième condition de consistance, $4\alpha_2+\alpha_1=4\beta_2+2\beta_1\Leftrightarrow5=4$ impossible.

Ce qui signifié que cette méthode n'est pas consistante à l'ordre 2, mais elle est consistante à l'ordre 1 pour a = -2 et b = 2.

Exercice 12

Etudier la zéro-stabilité des méthodes multi-pas

$$(1)\ u_{n+2}-4u_{n+1}+3u_n=-2hf_n$$

$$(2)\ 3u_{n+2}-4u_{n+1}+u_n=ahf_n$$

Y a-t-il des valeurs de a telles que (2) soit convergente ?

Le polynôme caractéristique de la première méthode est :

$$\rho(r)=r^2-4r+3=(r-1)(r-3)$$

Le module de la racine $r = 3$ ne vérifié pas la condition de racines donc la méthode n'est pas stable et n'est pas convergente.

Le polynôme caractéristique de la deuxième méthode s'écrit :

$$\rho(r)=r^2-\frac{4}{3}r+\frac{1}{3}=(r-1)\left(r-\frac{1}{3}\right)$$

Le polynôme caractéristique possède les deux racines $r=1$ et $r=1/3$ dont les modules sont inférieurs ou égale à 1. Autre, $\rho'(1)=2-4/3=2/3\neq 0$ donc la condition de racines est bien vérifiée, par conséquent la méthode est zéro-stable.
D'après la consistance :

$$\sum_{p=0}^{2}\alpha_p p=\sum_{p=0}^{m}\beta_p \;\Rightarrow\; \alpha_1+2\alpha_2=\beta_0+\beta_1+\beta_2$$

or,

$$\alpha_1=-\frac{4}{3},\ \alpha_2=1 \text{ et } \beta_0=\frac{a}{3},\ \beta_1=\beta_2=0 \;\Rightarrow\; a=2$$

Il faut que $a=2$ pour que la méthode soit convergente.

Exercice 13

Soit la méthode BDF3 :

$$u_{n+1}=\frac{18}{11}u_n-\frac{9}{11}u_{n-1}+\frac{2}{11}u_{n-2}+\frac{6}{11}hf\left(t_{n+1},\ u_{n+1}\right)$$

Montrer que cette méthode est consistante.

La méthode BDF3 s'écrit aussi comme :

$$u_{n+3}-\frac{18}{11}u_{n+2}+\frac{9}{11}u_{n+1}-\frac{2}{11}u_n=\frac{6}{11}hf_{n+3}$$

Elle est de la forme d'une méthode multi-pas de 3ième ordre dont la forme générale est :

$$\sum_{m=0}^{3}\alpha_m u_{n+m}=h\sum_{m=0}^{3}\beta_m f_{n+m}$$

avec,

$$\alpha_0=-\frac{2}{11},\ \alpha_1=\frac{9}{11},\ \alpha_2=-\frac{18}{11},\ \alpha_3=1 \text{ et } \beta_0=\beta_1=\beta_2,\ \beta_3=\frac{6}{11}$$

Pour que la méthode soit consistante, il faut que :

$$\sum_{m=0}^{3}\alpha_m=0 \text{ et } \sum_{m=0}^{3}m\alpha_m=\sum_{m=0}^{3}\beta_m$$

alors,

$$\sum_{m=0}^{3}\alpha_m=-\frac{2}{11}+\frac{9}{11}-\frac{18}{11}+\frac{11}{11}=\frac{-2+9-18+11}{11}=\frac{0}{11}=0$$

$$\sum_{m=0}^{3}m\alpha_m=-0\times\frac{2}{11}+1\times\frac{9}{11}-2\times\frac{18}{11}+3\times\frac{11}{11}=\frac{9-36+33}{11}=\frac{6}{11}$$

Ce qui montre que les conditions de consistance sont bien vérifiées pour BDF3.

Exercice 14

Soit a un paramètre réel et la méthode multi-pas :

$$u_{n+2}-2au_{n+1}+(2a-1)u_n=h\left[af_{n+2}+(2-3a)f_{n+1}\right]$$

1. Calculer les 1ère et 2ième polynôme caractéristique.
2. Etudier la consistance.
3. Etudier la zéro-stabilité.
4. Etudier la convergence.
5. Quels sont l'ordre et la constante d'erreur ?.

1. Les deux polynômes caractéristiques de la méthode multi-pas sont :

$$\rho(r)=r^2-2ar+2a-1 \text{ et } \sigma(r)=ar^2+(2-3a)r$$

2. La méthode est à deux pas pour qu'elle soit consistante il faut que,

$$\sum_{p=0}^{m}\alpha_p p^k=k\sum_{p=0}^{m}\beta_p p^{k-1} \text{ pour } k=0,\ 1 \text{ et } 2$$

Or, $\alpha_0=2a-1,\ \alpha_1=-2a,\ \alpha_2=1,\ \beta_0=0,\ \beta_1=2-3a$ et $\beta_2=a$ alors :

$$\sum_{p=0}^{m}\alpha_p=0 \Rightarrow 2a-1-2a+1=0 \text{ vérifiée}$$

$$\sum_{p=0}^{m}\alpha_p p-\sum_{p=0}^{m}\beta_p=0 \Rightarrow -2a+2-2+3a-a=0 \text{ vérifiée}$$

$$\sum_{p=0}^{m}\alpha_p p^2-2\sum_{p=0}^{m}\beta_p p=0 \Rightarrow -2a+4-2(2-3a+2a)=-2a+4-4+2a=0 \text{ vérifiée}$$

Les trois égalités sont bien vérifiées, donc la méthode est consistante à l'ordre 2.

3. Pour étudier la zéro-stabilité, les racines du polynôme $\rho(r)$ sont, $r_1=1$ et $r_2=2a-1$. Pour que la méthode est zéro-stable, il faut que $|2a-1|<1$ c'est-à-dire $0<a<1$.

4. La méthode est zéro-stable, c'est-à-dire,

$$u_n=A+B(2a-1)^n$$

Et puisque $0<a<1$ alors $u_n\to A\neq 0$ donc les racines r_1 et r_2 ne vérifient pas la condition de racines, donc la convergence ne suffit pas.

5. Le coefficient d'erreur est exprimé par :

$$C_3=\frac{1}{3!}\sum_{p=0}^{2}\alpha_p p^3-\frac{1}{2!}\sum_{p=0}^{2}\beta_p p^2=\frac{1}{6}(-2a+8)-\frac{1}{2}(2-3a+4a)=\frac{1}{6}(2-5a)$$

La constante d'erreur de la méthode est :

$$C=\frac{C_3}{\sum_{p=0}^{2}\beta_p}=\frac{2-5a}{6}\frac{1}{2(1-a)}=\frac{1}{12}\frac{2-5a}{1-a}$$

Exercice 15

Soit l'équation différentielle de deuxième ordre :

$$\frac{d^2u}{dt^2}+3u\frac{du}{dt}=0 \quad \text{avec,} \quad u(0)=0 \quad \text{et} \quad u(2)=1$$

Utiliser la méthode de tir pour déterminer une solution numérique de l'équation différentielle pour $0\le t\le 2$.

L'équation différentielle ainsi donnée est une équation non linéaire, peut s'écrit comme suit :

$$\frac{d^2u}{dt^2}=-3u\frac{du}{dt}$$

On pose,

$$\frac{du}{dt}=v=f(t,\ u,\ v) \quad \text{alors,} \quad \frac{dv}{dt}=-3uv=g(t,\ u,\ v)$$

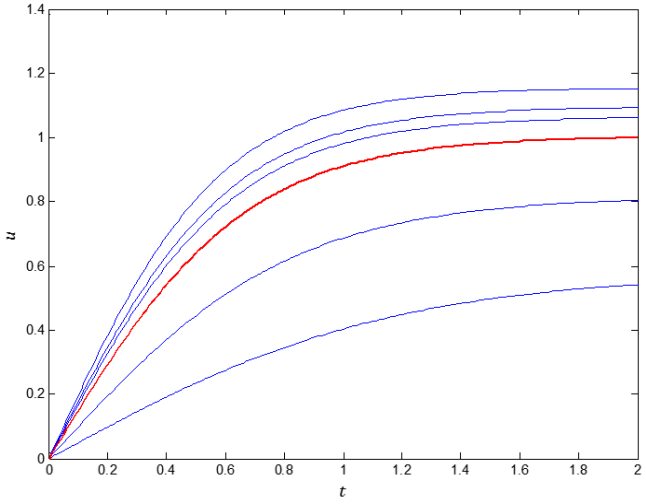
On a donc obtenu un système d'équations différentielles de premier ordre dont les inconnues sont les fonctions inconnues sont, $u(t)$ et $v(t)$. Mais le problème obtenu n'est pas un problème de Cauchy car on a qu'une seule condition $u(0)$ $(v(0)$ est absent$)$. Pour rendre ce problème type de Cauchy, on suppose une valeur pour $v(t)$ et appliquer la méthode de Runge-Kutta et vérifier si $u(2)\approx 1$. Pour ce faire soit le script MATLAB suivant qui utilise la méthode de Runge-Kutta d'ordre 4.

```
% Définition des fonctions f(t,u,v) et g(t,u,v)
f=@(t,u,v) v; g=@(t,u,v)-3*u*v;
h=0.01; t=0:h:2; % Définition du pas de discrétisation h
u(1)=0; % Condition à la limite inférieure fixée
LimiteSupReel = 1; % Condition imposé à la limite supérieure fixée
% Définition de la condition initiale qui concerne la dérivée
LimiteInf2Derivee = 0.5;
v(1)=LimiteInf2Derivee;
for n=1:numel(t)-1
   k1=h*f(t(n),u(n),v(n));
    K1=h*g(t(n),u(n),v(n));
     k2=h*f(t(n)+(h/2),u(n)+(k1/2),v(n)+(K1/2));
      K2=h*g(t(n)+(h/2),u(n)+(k1/2),v(n)+(K1/2));
     k3=h*f(t(n)+(h/2),u(n)+(k2/2),v(n)+(K2/2));
    K3=h*g(t(n)+(h/2),u(n)+(k2/2),v(n)+(K2/2));
   k4=h*f(t(n+1),u(n)+k3,v(n)+K3); K4=h*g(t(n+1),u(n)+k3,v(n)+K3);
  u(n+1)=u(n)+(1/6)*(k1+2*k2+2*k3+k4);
 v(n+1)=v(n)+(1/6)*(K1+2*K2+2*K3+K4);
end
LimiteSupCalcule=u(numel(t));
if abs(LimiteSupCalcule-LimiteSupReel)<=0.0001
   msgbox 'Valeur de la dérivée trouvée'
end
plot(t,u,'-b')
```

Ce script permet de résoudre l'équation différentielle avec la méthode RK4, et utilise plusieurs fois avec différente valeur de $v(0)$ et calcul aussi la valeur de $u(2)$ et la compare avec la condition imposée à la limite supérieure de l'intervalle qui égale à 1.

L'utilisation du script pour $u'(0)=1.5143$ conduit à $u(2)=0.9999\approx 1$ qui est la valeur trouvée, par conséquent, le problème devient de Cauchy alors, on peut utiliser d'autres solveurs de MATLAB pour avoir une solution numérique plus précise.

$u'(0)$ **LimiteInf2Derivee**	$u(2)$ **LimiteSupCalcule**
0.5000	0.5423
1.0000	0.8044
1.5143	0.9999
1.7000	1.0610
1.8000	1.0924
2.0000	1.1524



Exercice 16

Soit l'équation différentielle :

$$\frac{d^2u}{dt^2}+2\frac{du}{dt}+u=0 \quad \text{avec,} \quad u(0)=0 \quad \text{et} \quad u(1)=1$$

1. Utiliser la méthode de tir pour déterminer une solution approchée de l'équation différentielle.
2. Utiliser la méthode des différences finies avec $h=1/4$, pour déterminer aussi une solution approchée de l'équation différentielle.
3. Comment appréciez-vous les deux méthodes.

1. L'équation différentielle ainsi donnée est une équation différentielle homogène linéaire :

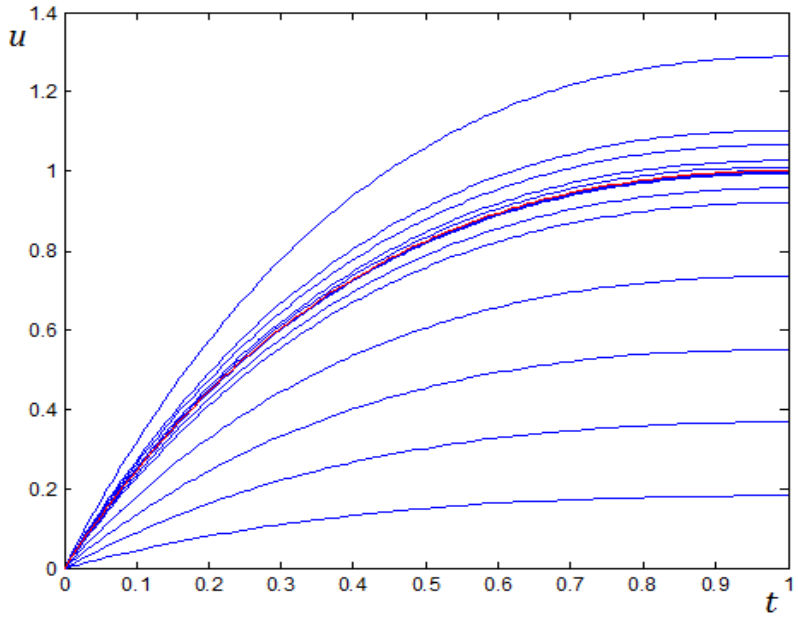
$$\frac{d^2u}{dt^2}+2\frac{du}{dt}+u=0 \quad \Rightarrow \quad \frac{d^2u}{dt^2}=-2\frac{du}{dt}-u$$

Considérons $du/dt=v$, soit par conséquent le système différentiel :

$$\begin{cases} f(t,\ u,\ v)\equiv \dfrac{du}{dt}=v, \\ g(t,\ u,\ v)\equiv \dfrac{dv}{dt}=2v-u. \end{cases}$$

Le problème obtenu n'est pas un problème de Cauchy car on a qu'une seule condition $u(0)$ $\left(v(0) \text{ est absent}\right)$. Pour rendre ce problème type de Cauchy, on suppose une valeur pour $v(0)$ et appliquer la méthode de Runge-Kutta et vérifier si $u(1)\approx 1$. Pour ce faire soit le script MATLAB suivant qui utilise la méthode de tir avec la méthode de Runge-Kutta d'ordre 4.

```
% Définition des fonctions f(t,u,v) et g(t,u,v)
f=@(t,u,v) v; g=@(t,u,v) -2*v-u; % Définition des fonctions f et g
h=0.01;  t=0:h:1; % Définition du pas de discrétisation h
u(1)=0; % Condition à la limite inférieure fixée
% Condition imposé à la limite supérieure fixée
LimiteSupReel = 1;
% Définition de la condition initiale qui concerne la dérivée
LimiteInf2Derivee = 2.7190; v(1)=LimiteInf2Derivee;
for n=1:numel(t)-1
 k1=h*f(t(n),u(n),v(n));
 K1=h*g(t(n),u(n),v(n));
 k2=h*f(t(n)+(h/2),u(n)+(k1/2),v(n)+(K1/2));
 K2=h*g(t(n)+(h/2),u(n)+(k1/2),v(n)+(K1/2));
 k3=h*f(t(n)+(h/2),u(n)+(k2/2),v(n)+(K2/2));
 K3=h*g(t(n)+(h/2),u(n)+(k2/2),v(n)+(K2/2));
 k4=h*f(t(n+1),u(n)+k3,v(n)+K3);
 K4=h*g(t(n+1),u(n)+k3,v(n)+K3);
 u(n+1)=u(n)+(1/6)*(k1+2*k2+2*k3+k4);
 v(n+1)=v(n)+(1/6)*(K1+2*K2+2*K3+K4);
end
LimiteSupCalcule=u(numel(t))
if abs(LimiteSupCalcule-LimiteSupReel)<=0.001
   msgbox 'Valeur de la dérivée trouvée'
end
plot(t,u,'-b')
```



$v(0)=u'(0)$	$u(1)$
0.50	0.1839
1.00	0.3679
1.50	0.5518
2.00	0.7358
2.50	0.9197
2.60	0.9565
2.70	0.9933
2.71	0.9970
2.72	1.0006
2.75	1.0117
3.00	1.1036
3.50	1.2876

Ce script permet de résoudre l'équation différentielle avec la méthode RK4, et utilise plusieurs fois avec différentes valeurs de $v(0)$ et calcul aussi la valeur de $u(0)$ et la compare avec la condition imposée à la limite supérieure de l'intervalle qui égale à 1.

L'utilisation du script pour $v(0)=u'(0)\approx 2.72$ conduit à $u(0)=1.0006\approx 1$, qui est la valeur trouvée. Par conséquent, le problème devient de Cauchy formé par le système différentiel :

$$\frac{du}{dt}=v,\quad \frac{dv}{dt}=2v-u \quad \text{avec} \quad u(0)=0 \quad \text{et} \quad v(0)=2.72$$

L'utilisation du script pour $u'(0)=1.5143$, conduit à $u(2)=0.9999\approx 1$, qui est la valeur trouvée, par conséquent le problème devient de Cauchy alors, on peut utiliser d'autres solveurs de MATLAB pour avoir une solution numérique plus précise.

2. L'équation :

$$\frac{u_{i+1}-2u_i+u_{i-1}}{h^2}+2\frac{u_{i+1}-u_{i-1}}{2h}+u_i=0 \Rightarrow (1-h)u_{i-1}+(h^2-1)u_i+(1+h)u_{i+1}=0$$

Si $h=1/4$, l'équation aux différences devient :

$$12u_{i-1}-31u_i+20u_{i+1}=0$$

Dans l'intervalle [0, 1], il faut donc chercher la solution dans les points dont l'abscisse correspondent à : t_2, t_3 et t_4. A partir de l'équation aux différences soit le système d'équations :

$$\begin{cases} u_1=0, \\ 12u_1-31u_2+20u_3=0, \\ 12u_2-31u_3+20u_4=0, \\ 12u_3-31u_4+20u_5=0, \\ u_5=1. \end{cases} \Rightarrow \begin{cases} -31u_2+20u_3=-12u_1=0, \\ 12u_2-31u_3+20u_4=0, \\ 12u_3-31u_4=-20u_5=-20. \end{cases}$$

La solution de ce système d'équations est :

$$u_2=0.5365,\ u_3=0.8316 \text{ et } u_4=0.9671$$

3. Le schéma aux différences finies est d'ordre deux, c'est-à-dire l'erreur de troncature de type $O(h^2)$, pour comparer la méthode des différences finies, utilisons la méthode de Runge-Kutta d'ordre 2, soit alors le script suivant :

```
f=@(u,v)v; % Définition de la fonction f(u,v)
g=@(u,v)-2*v-u; % Définition de la fonction g(u,v)
h=1/4; % Définition du pas de discrétisation h
t=0:h:1;
u(1)=0; % Définition de la condition initiale
v(1)=2.72;
for n=1:numel(t)-1
 k1=h*f(u(n),v(n)); K1=h*g(u(n),v(n));
 k2=h*f(u(n)+k1,v(n)+K1); K2=h*g(u(n)+k1,v(n)+K1);
 u(n+1)=u(n)+(1/2)*(k1+k2);
 v(n+1)=v(n)+(1/2)*(K1+K2);
end
```

Qui permet d'avoir la solution de l'équation différentielle par la méthode de Runge-Kutta d'ordre 2 :

$$u_2=0.5100,\ u_3=0.7969,\ u_4=0.0.9338 \text{ et } u_5=0.9727$$

Remarquons qu'il y a une différence négligeable entre les résultats obtenues par la méthode des différences finies et celle obtenues par la méthode de Runge-Kutta d'ordre 2.

La solution exacte de cette équation différentielle est :

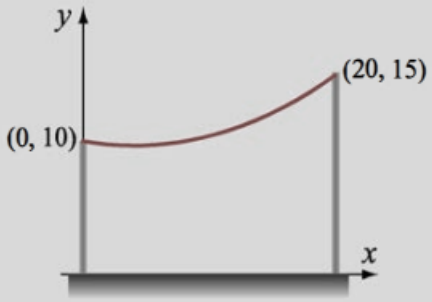
$$u(t)=te^{1-t}$$

Par conséquent,

$$u_2=0.5293,\ u_3=0.8244 \text{ et } u_4=0.9630$$

Exercice 17

Un câble flexible de densité uniforme est suspendu entre deux points, comme indiqué sur la figure. La forme du câble $y(x)$ est régi par l'équation différentielle :



$$\frac{d^2y}{dx^2} = C\sqrt{1+\left(\frac{dy}{dx}\right)^2}$$

ou C (C = 0.041 m^{-1})est une constante égale au rapport du poids par unité de longueur du câble à l'amplitude de la composante horizontale de la tension dans le câble à son point le plus bas. Le câble est suspendu entre deux points spécifiés par $y(0)$ = 10 m et $y(20)$ = 15 m.

Utiliser les fonctions intégrées de MATLAB pour déterminer et tracer la forme du câble entre x = 0 m et x =20 m.

Le problème traité dans cette exercice est de type de Dirichlet à conditions aux limites, ou l'équation différentielle :

$$\frac{d^2y}{dx^2} = C\sqrt{1+\left(\frac{dy}{dx}\right)^2}$$

Peut être transformée sous la forme vectorielle suite un changement de variable comme :

$$\begin{pmatrix} \frac{dy}{dx} \\ \frac{dw}{dx} \end{pmatrix} = \begin{pmatrix} w \\ C\sqrt{1+w^2} \end{pmatrix}$$

Pour déterminer une solution avec les fonctions intégrées de MATLAB pour les problèmes aux limites, soit alors :

bvpfcn la fonction qui calcule $dy/dx = w$ et $dw/dx = C\sqrt{1+w^2}$ pour les données de x, w et y.

```
function dydx = bvpfcn(x,y)
dydx(1)=y(2);
dydx(2)=0.041*sqrt(1+y(2)*y(2));
end
```

bcfcn la fonction qui calcule le résidu dans les conditions aux limites. Le résidu est la différence entre la solution numérique et la condition aux limite prescrite.

```
function res = bcfcn(ya,yb)
res(1)=ya(1)-10;
res(2)=yb(1)-15;
end
```

jac la fonction qui calcule les éléments de la matrice jacobienne pour les valeurs données de x, w et y.

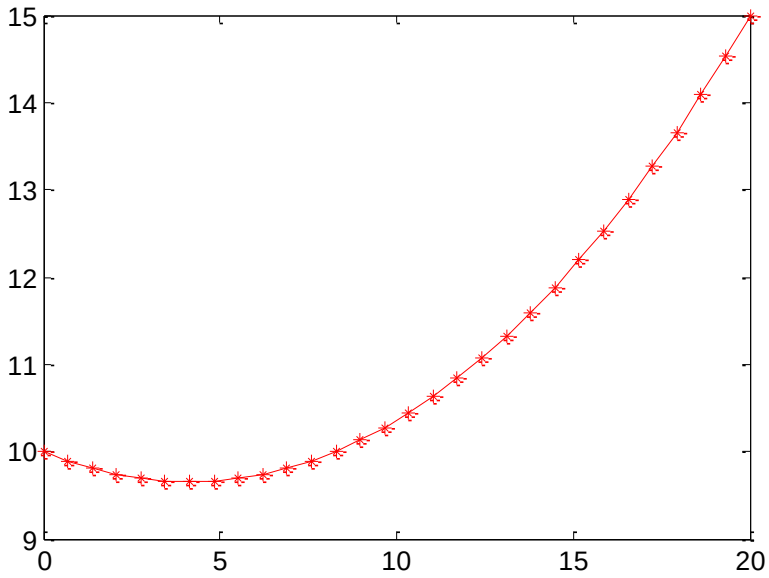
```
function dfdy = jac(x,y)
dfdy(1,1)= 0; dfdy(1,2)= 1;
dfdy(2,1)= 0; dfdy(2,2)= 0.041*y(2)/(sqrt(1+y(2)*y(2)));
end
```

Ainsi, les solveurs de MATLAB sont utilisés dans le script suivant :

```
Opts=bvpset('Fjacobian',@jac,'RelTol',0.1,'AbsTol',0.1,'Stats','on');
x=linspace(0,20,30);
solinit= bvpinit(x, [0,20]);
Solution=bvp4c(@bvpfcn,@bcfcn, solinit, Opts);
plot(x, Solution.y(1,:),'r-*')
```

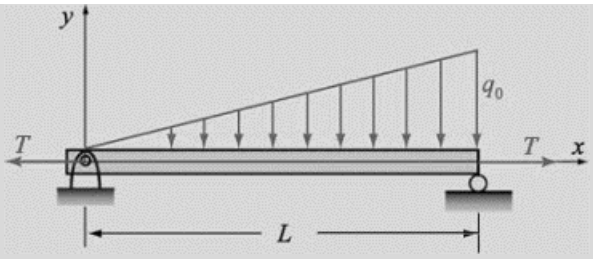
L'exécution du script permet d'obtenir l'information suivante et le graphique ci-dessous de la solution du problème.

```
The solution was obtained on a mesh of 30 points.
The maximum residual is  1.357e-07.
There were 354 calls to the ODE function.
There were 22 calls to the BC function.
```



Exercice 18

Une poutre de longueur L = 4 m supporte une charge répartie selon la figure ci-contre. La flexion de la poutre est exprimée par y solution de l'équation différentielle :



$$\frac{d^2y}{dx^2}=\frac{1}{EI}\left[\frac{q_0}{6}\left(Lx-\frac{x^3}{L}\right)+Ty\right]\left[1+\left(\frac{dy}{dx}\right)^2\right]^{3/2}$$

Avec les conditions aux limites, $y(0)=y(L)=0$.

$EI = 1.2\times10^7$ N-m^2 est la rigidité à la flexion, q_0 = 30 kN/m et T = 20 kN.

Utiliser les fonctions de MATLAB, déterminer et tracer la flexion de la poutre en fonction de x.

Dans cet exercice, on a un le problème aux limites de type de Dirichlet. L'équation différentielle du second ordre peut être transformée en un système d'équation de premier ordre, comme suit :

$$\frac{dy}{dx}=w \Rightarrow \frac{dw}{dx}=\frac{1}{EI}\left[\frac{q_0}{6}\left(Lx-\frac{x^3}{L}\right)+Ty\right]\left[1+w^2\right]^{3/2}$$

soit les fonctions,

$$f_1(y,\ w)=w \quad \text{et} \quad f_2(y,\ w)=\frac{1}{EI}\left[\frac{q_0}{6}\left(Lx-\frac{x^3}{L}\right)+Ty\right]\left[1+w^2\right]^{3/2}$$

et la matrice jacobienne associée

$$\mathbf{J}=\begin{pmatrix} 0 & 1 \\ \dfrac{T}{EI}\left[1+w^2\right]^{3/2} & \dfrac{3}{EI}\left[\dfrac{q_0}{6}\left(Lx-\dfrac{x^3}{L}\right)+Ty\right]w\sqrt{1+w^2} \end{pmatrix}$$

Soit les scripts des fonctions suivantes définissant les fonctions f_1 et f_2, les résidus aux limites et la matrice jacobienne.

```
function dydx = bvpfcn(x,y)
EI=1.2e7; q0=30*1000; T=20*1000;L=4;
dydx(1)=y(2);
dydx(2)=(1/EI)*((sqrt(1+y(2)*y(2)))^3)*(((q0*x/6)*(L-(x*x/L)))+T*y(1));
end
```

```
function res = bcfcn(ya,yb)
res(1)=ya(1);
res(2)=yb(1);
end
```

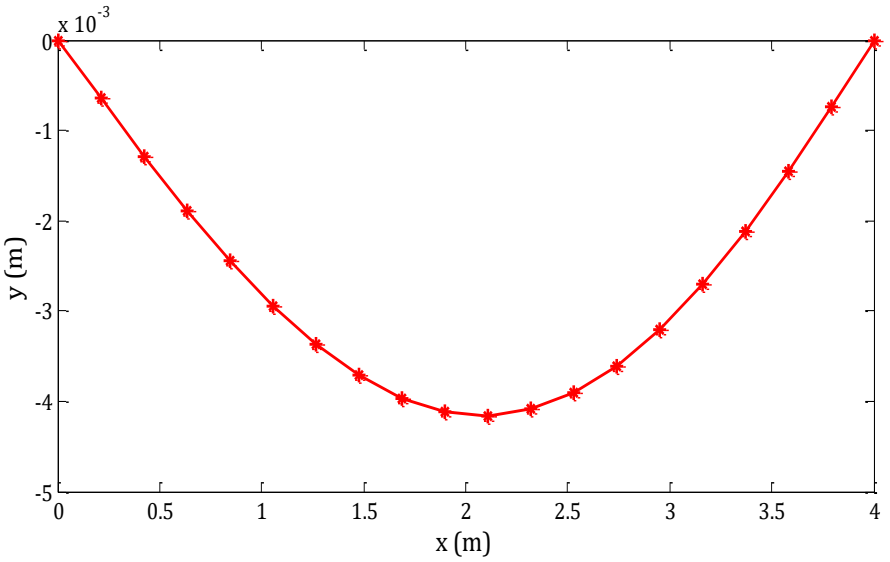
```
function dfdy = jac(x,y)
EI=1.2e7; q0=30*1000; T=20*1000;L=4;
dfdy(1,1)= 0;
dfdy(1,2)= 1;
dfdy(2,1)= (T/EI)*((sqrt(1+y(2)*y(2)))^3);
dfdy(2,2)= (3*y(2)/EI)*sqrt(1+y(2)*y(2))*(((q0*x/6)*(L-(x*x/L)))+T*y(1));
end
```

Pour déterminer la solution du problème et tracer la flexion de la poutre *y* en fonction de *x*, soit le script suivant qui contient les fonctions appropriées de MATLAB.

```
Opts=bvpset('Fjacobian',@jac,'RelTol',0.1,'AbsTol',0.1,'Stats','on');
x=linspace(0,4,20);
solinit= bvpinit(x, [0,4]);
Solution=bvp4c(@bvpfcn,@bcfcn, solinit, Opts);
plot(x,Solution.y(1,:),'r-*')
```

L'exécution de ce script dans l'environnement MATLAB permet d'avoir l'information et le graphe (ci-dessous) de la flexion de la poutre.

```
The solution was obtained on a mesh of 20 points.
The maximum residual is  3.082e-06.
There were 195 calls to the ODE function.
There were 19 calls to the BC function.
```



14. Exercices supplémentaires

Exercice 1

Calculer $u(0.1)$ et $u(0.2)$ en utilisant la méthode de Taylor d'ordre 5 pour le problème de Cauchy suivant :

$$\frac{du}{dt} = u + t;\ \ u(0) = 2$$

Déterminer la solution exacte du problème et comparer les résultats obtenus à l'ordre de 10^{-5}.

Exercice 2

Montrer que les méthodes de Heun et de Crank-Nicolson sont des méthodes d'ordre 2.

Exercice 3

Considérons un corps ponctuel de masse m et de température interne T situé dans un environnement de température constante T_e. Le transfert de chaleur entre le corps et l'extérieur peut être décrit par la loi de Stefan-Boltzmann

$$v(t) = \varepsilon\gamma S\left(T^4(t) - T_e^4\right)$$

où t est la variable temporelle, ε la constante de Boltzmann (égale à $5.6 \cdot 10^{-8}$ J/m²K⁴s, J est l'abréviation de Joule, K celle de Kelvin et, naturellement, m et s celles de mètre et seconde), γ est la constante d'émissivité du corps, S sa surface et v est la vitesse de transfert de chaleur. Le taux de variation de l'énergie $E(t) = mCT(t)$ (où C est la capacité calorifique du corps) est égal, en valeur absolue, à la vitesse v. La température est peut-être obtenue à partir de l'équation différentielle :

$$\frac{dT}{dt} = -\frac{v}{mC}$$

Calculer la température T en résolvant l'équation par les méthodes de Crank-Nicolson et de Heun quand le corps est un cube de coté 1 m et de masse 1 kg. On posera T_0 = 180K, T_e = 200K γ = 0.5 et C =100 J/(kg/K). Comparer les résultats obtenus en prenant h = 20 et h = 10, pour t allant de 0 à 200 secondes.

Exercice 4

L'équation différentielle :

$$u'(t)=u(t)+e^{2t},\quad u(0)=2$$

Possède la solution analytique $u(t)=e^{t}\left(1+e^{t}\right)$

(a) En prenant h = 0.1, faire 3 itérations de la méthode d'Euler modifiée et calculer l'erreur commise sur u_3 en comparant les résultats avec la solution analytique $u(0.3)$.

(b) En prenant h = 0.05, faire 6 itérations de la méthode d'Euler modifiée et calculer l'erreur commise sur u_6 en comparant les résultats avec la solution analytique $u(0.3)$.

(c) Faire le rapport des erreurs commises en (a) et en (b) et commenter le résultat en fonction de l'erreur de troncature locale liée à la méthode utilisée.

(d) Utiliser l'extrapolation de Richardson pour obtenir une meilleure approximation de $u(0.3)$

Refaire l'exercice précédent, mais cette fois à l'aide de la méthode de Runge-Kutta d'ordre 4.

Exercice 5

On considère l'équation différentielle :

$$\begin{cases}u'(t)=2u(t)\\ u(0)=5\end{cases}$$

a) Vérifier que la solution analytique est $u(t)=5e^{2t}$.

b) En posant $h=1/N$, montrer que les approximations fournies par la méthode d'Euler explicite peuvent s'écrire $u_n=5(1+2h)^n$, pour $n=0,\ 1,\ \ldots,\ N$.

c) Vérifier numériquement que l'erreur $e(h)$ se comporte suivant la relation $e(h)\approx Kh$ où K est une constante à déterminer.

Exercice 6

Soit le problème à valeur initiale,

$$u'=-3u\sin t,\quad u(0)=\frac{1}{2}$$

Par la méthode d'Euler, avec N=10, calculer une solution approchée et comparer avec la solution exacte de l'équation différentielle,

$$u=\frac{1}{2}e^{3(\cos t-1)}$$

Exercice 7

Utiliser la méthode de Runge-Kutta d'ordre 4 RK4 pour résoudre numériquement les équations de Lotka–Volterra sur l'intervalle [0, 80]:

$$u'(x)=0.1u(x)-0.01u(x)v(x),\ \ v'(x)=-0.5v(x)+0.01u(x)v(x)$$

$$u(0)=60,\ \ v(0)=20$$

Tracer les graphes des deux fonctions $u(x)$ et $v(x)$.

Exercice 8

Vérifier la consistance de la méthode de RK3 suivante :

$$u_{n+1} = u_n + \frac{1}{6}\left(k_1 + 4k_2 + k_3\right)$$

avec,

$$k_1 = hf\left(t_n,\ u_n\right),\ k_2 = hf\left(t_n + \frac{h}{2},\ u_n + \frac{k_2}{2},\right) \text{ et } k_3 = hf\left(t_n + h,\ u_n - k_1 + 2k_2\right)$$

Implémenter dans un programme MATLAB pour résoudre le problème de Cauchy :

$$\begin{cases} \dfrac{du}{dt} = \sin t - u, \quad 0 < t \le 1, \\ u(0) = 0. \end{cases}$$

et vérifier expérimentalement que la méthode est d'ordre 3 en *h*. Les méthodes de Heun et RK3 sont à la base du programme MATLAB **`ode23`**.

Exercice 9

Soit l'équation différentielle ordinaire de première ordre,

$$\frac{du}{dt} = \frac{t^2}{u},\ 0 \le t \le 2.1 \text{ et } u(0) = 2$$

(a) Utiliser la méthode d'Euler explicite pour calculer une solution approchée de cette EDO pour $h = 0.7$.

(b) Utiliser la fameuse méthode de Runge-Kutta RK4 pour calculer une solution approchée de cette EDO pour $h = 0.7$.

(c) Calculer l'erreur commise dans chaque nœud pour les deux cas de méthodes, sachant que la solution exacte du problème est :

$$u(t) = \sqrt{\frac{2t^3}{3} + 4}$$

Exercice 10

Etudier la méthode multi-pas linéaire :

$$u_{n+1} = \theta u_n + (1-\theta)u_{n-1} + 2hf_n + \frac{h\theta}{2}\left(f_{n-1} - 3f_n\right), \text{ où } \theta \in \mathbb{R}$$

Exercice 11

Etudier la famille de méthodes multi-pas linéaires dépendant d'un paramètre θ définies par :

$$u_{n+1} = u_n + h\left[\left(1 - \frac{\theta}{2}\right)f_n + \frac{\theta}{2}f_{n+1}\right]$$

Considérer la méthode correspondant à $\theta = 1$ et l'appliquer au problème de Cauchy suivant :

$$\begin{aligned} u'(t) &= -10u(t), \quad t > 0, \\ u(0) &= 1 \end{aligned}$$

Trouver les valeurs de *h* pour lesquelles cette méthode est absolument stable.

Exercice 12

Ecrire un programme MATLAB qui trace la région de stabilité absolue d'une méthode de Runge-Kutta pour laquelle on dispose de la fonction de stabilité $R(h\lambda)$. Tester le code dans le cas particulier où,

$$R(h\lambda)=1+h\lambda+(h\lambda)^2/2+(h\lambda)^3/6+(h\lambda)^4/24+(h\lambda)^5/120+(h\lambda)^6/600$$

et vérifier que cette région n'est pas connexe.

Exercice 13

Montrer que la méthode :

$$u_{n+1}=-u_n+2u_{n-1}+h\left(\frac{5}{2}f_n+\frac{1}{2}f_{n-1}\right)$$

st d'ordre 2 et instable. De plus, montrez directement qu'il n'est pas nécessaire de converger lors de la résolution de $u'(t)=f(t,\,u(t))$ en considérant le problème particulier $u'(t)=0,\quad u(0)=0$.

Pour la méthode numérique, pensez à utiliser les valeurs initiales $u_0=h,\quad u_1=-2h$.

Exercice 14

Déterminer la fonction $R(h\lambda)$ associée à la méthode de Merson dont le tableau de Butcher est donné par :

0	0	0	0	0	0
$\frac{1}{3}$	$\frac{1}{3}$	0	0	0	0
$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$	0	0	0
$\frac{1}{2}$	$\frac{1}{8}$	0	$\frac{3}{8}$	0	0
1	$\frac{1}{2}$	0	$-\frac{3}{2}$	2	0
	$\frac{1}{6}$	0	0	$\frac{2}{3}$	$\frac{1}{6}$

Exercice 15

Soit la formule générale de la méthode explicite de deux pas :

$$u_{n+1}=a_0u_n+a_1u_{n-1}+h\left[b_0f(t_n,\,u_n)+b_1f(t_{n-1},\,u_{n-1})\right]$$

(a) Envisagez de trouver toutes les méthodes de deux pas et qui sont d'ordre 2. Montrez que les coefficients doivent satisfaire les équations :

$$a_0+a_1=1,\qquad -a_1+b_0+b_1=1,\qquad a_1-2b_1=1$$

Résoudre pour $\{a_1,\,b_0,\,b_1\}$ en fonction de a_0.

(b) Déterminer la formule de l'erreur de troncature en fonction a_0.

(c) Quelle est la condition sur a_0 pour que la méthode soit stable ? soit convergent ?

Exercice 16

Donner le système tridiagonal requis pour résoudre l'équation différentielle:

$$\begin{cases} u''(t) = \left(1 + \dfrac{2}{t}\right) u(t) - t - 2, \\ u(0) = 0 \text{ et } u(1) = 2. \end{cases}$$

à l'aide de la méthode des différences finies centrées (prendre 5 intervalles).

Exercice 17

On veut résoudre l'équation différentielle avec conditions aux limites :

$$\begin{cases} u''(t) = \dfrac{1}{8}\left(32 + 2t^3 - u(t)u'(t)\right) \\ u(1) = 17,\ u(3) = \dfrac{43}{3} \end{cases}$$

a) Cette équation différentielle est-elle linéaire ?
Pour résoudre ce problème, on va servir de l'équation différentielle avec conditions initiales suivante :

$$\begin{cases} u''_\beta(t) = \dfrac{1}{8}\left(32 + 2t^3 - u_\beta(t)u'_\beta(t)\right) \\ u_\beta(1) = 17,\ u'_\beta(1) = \beta \end{cases}$$

Il suffit ensuite de trouver la valeur du paramètre β de sorte que $u_\beta(3) = 43/3$.

b) Donner une interprétation géométrique de cette méthode.

c) Écrire cette dernière équation différentielle sous la forme d'un système de 2 équations différentielles d'ordre 1 avec conditions initiales. On a résolu le système obtenu en c) pour $\beta_1 = -10$ et $\beta_2 = -12$ par une méthode de Runge-Kutta d'ordre 4 en utilisant un pas de temps h = 0.1 (la solution est indiquée seulement à tous les 4 pas de temps). On a ainsi obtenu :

$\beta_1 = -10$

t	$u_\beta(t)$	$u_{\beta'}(t)$
1.0	1.700 000 × 10	-1.000 000 × 10^{1}
1.4	1.452 246 × 10	-3.341 190 × 10^{0}
1.8	1.388 176 × 10	-1.877 628 × 10^{-1}
2.2	1.420 276 × 10	+1.656 797 × 10^{0}
2.6	1.513 410 × 10	+2.941 136 × 10^{0}
3.0	1.652 553 × 10	+3.994 271 × 10^{0}

$\beta_2 = -12$

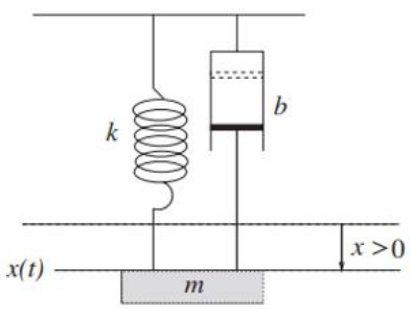
t	$u_\beta(t)$	$u_{\beta'}(t)$
1.0	1.700 000 × 10	-1.200 000 × 10^{1}
1.4	1.395 818 × 10	-4.336 713 × 10^{0}
1.8	1.301 724 × 10	-7.342 975 × 10^{-1}
2.2	1.318 164 × 10	+1.404 508 × 10^{0}
2.6	1.405 758 × 10	+2.905 261 × 10^{0}
3.0	1.546 717 × 10	+4.110 557 × 10^{0}

d) En vous servant de ces données et à l'aide de la méthode de la sécante, proposer une nouvelle valeur de β qui permette de s'approcher de la solution de l'équation différentielle avec conditions aux limites de départ.

Exercice 18

Dans l'étude d'un ressort vibrant avec amortissement, on est conduit au problème des valeurs aux limites de la forme :

$$mx''(t)+bx'(t)+kx(t)=0,\quad x(0)=x_0,\ x(a)=x_1$$



Ou m est la masse, h est la constante d'amortissement, k est coefficient du raideur du ressort, x_0 est le déplacement initial, $x(t)$ est le déplacement par rapport à l'équilibre du système de ressort à l'instant t.

Déterminer le déplacement de ce système de ressort dans l'intervalle de temps [0, 10] pour m = 26 kg, b = 12 kg.s^{-1}, k = 37 kg.s^{-2}, x_0= 70.0 cm et x_1= - 13.32 cm en utilisant la méthode des différences finies et les fonction intégrées de Matlab et tracer le graphe suivant les deux méthodes.

Chapitre 7.

Méthodes numériques des EDPs

Les équations différentielles aux dérivées partielles EDPs ce sont les équations de la physique. Dans le domaine de l'engineering, les équations différentielles aux dérivées partielles existent généralement soit en d'ordre 1 comme une fonction :

$$F\left(x,\ t,\ u,\ \frac{\partial u}{\partial x},\ \frac{\partial u}{\partial t}\right)=0$$

Ou en d'ordre 2 comme :

$$F\left(x,\ t,\ u,\ \frac{\partial u}{\partial x},\ \frac{\partial u}{\partial t},\ \frac{\partial^2 u}{\partial x^2},\ \frac{\partial^2 u}{\partial t^2},\ \frac{\partial^2 u}{\partial x \partial t}\right)=0$$

La plus simple équation différentielle linéaire aux dérivées partielles est :

$$A(x,\ t)\frac{\partial^2 u}{\partial x^2}+2B(x,\ t)\frac{\partial^2 u}{\partial x \partial t}+C(x,\ t)\frac{\partial^2 u}{\partial t^2}+a(x,\ t)\frac{\partial u}{\partial x}+b(x,\ t)\frac{\partial u}{\partial t}+c(x,\ t)u=f(x,\ t)$$

$A(x,\ t)$, $B(x,\ t)$, $C(x,\ t)$, $a(x,\ t)$, $b(x,\ t)$ et $c(x,\ t)$ sont des fonctions de x et y. Si $f(x,\ t)=0$ l'équation différentielle est dite homogène. La classification de ces modèles peut être faite algébriquement par la considération du discriminant $\Delta=B^2-AC$ de l'EDP.
Une EDP *hyperbolique* si $\Delta>0$, *parabolique* si $\Delta=0$ et *elliptique* si $\Delta<0$.

Les fameuses équations différentielles aux dérivées partielles sont :
- L'*équation de la convection et l'équation des ondes* sont des exemples d'équations de type *hyperbolique*,

$$\frac{\partial u}{\partial t}+a\frac{\partial u}{\partial x}=0 \quad \text{et} \quad \frac{\partial^2 u}{\partial t^2}=c^2\frac{\partial^2 u}{\partial x^2}$$

- L'*équation de la chaleur* ou *équation de la diffusion* est une équation de type *parabolique*,

$$\frac{\partial u}{\partial t}=\lambda\frac{\partial^2 u}{\partial x^2}$$

- L'*équation de Laplace* est une équation de type *elliptique*,

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0$$

u est une fonction de l'espaces x et y ; $u=u(x,\ y)$.

Les équations différentielles aux dérivées partielles existent aussi sous forme de systèmes différentielles comme les équations de Navier–Stockes ou de Barré de Saint–Venant.

La solution numérique des EDPs est un sujet très important, est faite généralement par les méthodes, des différences finies, des éléments finis et des volumes finis.

1. Approximations par différences finies

Soit $u=u(x,\ t)$, une fonction de à deux variables *x* et *t*, généralement *x* traduit l'espace ou la position par rapport à un point de référence et *t* traduit le temps, dans certain cas la fonction dépend que des coordonnées *x*, *y* et de *t* c'est-à-dire $u=u(x,\ y,\ t)$. Par exemple, un point matériel déplace dans un plan *Oxy*, sa vitesse *u* est une fonction de sa position $(x_i,\ y_j)$ et du temps t^n, c'est-à-dire $u=u(x_i,\ y_j,\ t^n)$ qui peut être notée aussi par $u_{i,j}^n$.

Soit le domaine (Fig. 7.1) pour *x* de $[x_0,\ x_N]$ et pour *y* de $[y_0,\ y_M]$ quadrillé par un réseau orthogonal dont les côtés sont parallèles aux axes *x* et *y*. Le pas de discrétisation en *x* est noté *h* et le pas de discrétisation en *y* est noté *k*. Les coordonnées d'un point du maillage (nœud du réseau) seront donc $(x_0+ih,\ y_0+jk)$.

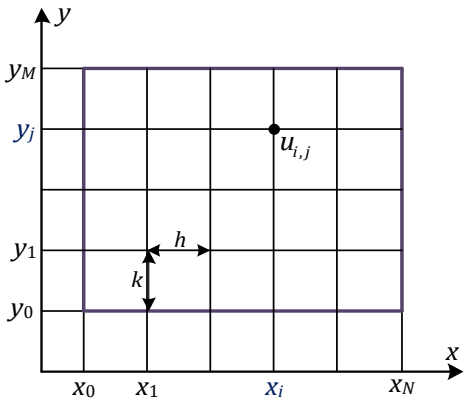


Fig. 7.1 : Domaine de travail.

Le développement de Taylor de la fonction $u(x_i+h,\ y_j)$ au voisinage du point de coordonnées $(x_i,\ y_j)$ en termes de *x* est :

$$u(x_i+h,\ y_j)=u(x_i,\ y_j)+h\left(\frac{\partial u}{\partial x}\right)_{i,j}+\frac{1}{2}h^2\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j}+O(h^2)$$

Ce qui permet d'approcher la dérivée partielle de premier ordre par un schéma aux différences finies avant :

$$u(x_i+h,\ y_j)=u(x_i,\ y_j)+h\left(\frac{\partial u}{\partial x}\right)_{i,\ j}+O(h)\ \Rightarrow\ \left(\frac{\partial u}{\partial x}\right)_{i,\ j}=\frac{u(x_i+h,\ y_j)-u(x_i,\ y_j)}{h}+O(h)$$

où, $O(h)$ est l'erreur de troncature d'ordre.

Le développement de Taylor de la fonction $u(x_i-h,\ y_j)$ au voisinage du point de coordonnées $(x_i,\ y_j)$ en termes de *x* est peut-être obtenu à partir de $u(x_i+h,\ y_j)$ par la substitution de *h* par –*h*, c'est-à-dire :

$$u(x_i-h,\ y_j)=u(x_i,\ y_j)-h\left(\frac{\partial u}{\partial x}\right)_{i,\ j}+O(h)\ \Rightarrow\ \left(\frac{\partial u}{\partial x}\right)_{i,\ j}=\frac{u(x_i,\ y_j)-u(x_i-h,\ y_j)}{h}+O(h)$$

C'est le schéma aux differences finies arrière de la dérivée partielle d'ordre 1 en *h*.

Les développements de Taylor précédents peuvent être s'écrit aussi comme suit :

$$u(x_i+h,\ y_j)=u(x_i,\ y_j)+h\left(\frac{\partial u}{\partial x}\right)_{i,\ j}+\frac{1}{2}h^2\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,\ j}+O(h^2)$$

$$u(x_i-h,\ y_j)=u(x_i,\ y_j)-h\left(\frac{\partial u}{\partial x}\right)_{i,\ j}+\frac{1}{2}h^2\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,\ j}+O(h^2)$$

Ce qui conduit à exprimer la dérivée partielle par le schéma aux différences finies centrale :

$$\left(\frac{\partial u}{\partial x}\right)_{i,\ j} = \frac{u\left(x_i + h,\ y_j\right) - u\left(x_i - h,\ y_j\right)}{2h} + O\left(h^2\right)$$

La dérivée partielle $\left(\partial u/\partial x\right)_{i,\ j}$ a été faite avec trois schémas aux différences finies, pour les schémas avant et arrière, l'erreur de troncature est de premier ordre et pour le schéma centrale est de deuxième ordre.

Alors la dérivée partielle par rapport à *x* peut être approchée par les formules suivantes :

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} \approx \frac{u\left(x_i + h,\ y_j\right) - u\left(x_i,\ y_j\right)}{h} \equiv \frac{u_{i+1,j} - u_{i,j}}{h}$$

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} \approx \frac{u\left(x_i,\ y_j\right) - u\left(x_i - h,\ y_j\right)}{h} \equiv \frac{u_{i,j} - u_{i-1,j}}{h}$$

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} \approx \frac{u\left(x_i + h,\ y_j\right) - u\left(x_i - h,\ y_j\right)}{2h} \equiv \frac{u_{i+1,j} - u_{i-1,j}}{2h}$$

La somme $u\left(x_i + h,\ y_j\right) + u\left(x_i - h,\ y_j\right)$ conduit à exprimer la dérivée partielle seconde par rapport à *x*,

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} = \frac{u\left(x_i + h,\ y_j\right) - 2u\left(x_i,\ y_j\right) + u\left(x_i - h,\ y_j\right)}{h^2} + O\left(h^2\right)$$

Qui peut être approchée par :

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

De même, les dérivées partielles par rapport à *y* sont exprimées approximativement par :

$$\left(\frac{\partial u}{\partial y}\right)_{i,j} \approx \frac{u_{i,j+1} - u_{i,j}}{k},\ \text{schéma avant,}$$

$$\left(\frac{\partial u}{\partial y}\right)_{i,j} \approx \frac{u_{i,j} - u_{i,j-1}}{k},\ \text{schéma arrière,}$$

$$\left(\frac{\partial u}{\partial y}\right)_{i,j} \approx \frac{u_{i,j+1} - u_{i,j-1}}{2k},\ \text{schéma centrale}$$

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

Où *k* est le pas de discrétisation suivant la variable *y*.

De développement en série de Taylor de $u\left(x_i + h,\ y_j\right) = u_{i+1,j}$ s'écrit aussi :

$$u_{i+1,j} = u_{i,j} + h\left(\frac{\partial u}{\partial x}\right)_{i,j} + \frac{1}{2}h^2\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} + \sum_{\eta \geq 3}\left(\frac{\partial^\eta u}{\partial x^\eta}\right)_{i,j}\frac{h^\eta}{\eta!}$$

alors,

$$\left(\frac{\partial u}{\partial y}\right)_{i+1,j} = \left(\frac{\partial u}{\partial y}\right)_{i,j} + h\left(\frac{\partial^2 u}{\partial y \partial x}\right)_{i,j} + \frac{1}{2}h^2\left(\frac{\partial^3 u}{\partial y \partial x^2}\right)_{i,j} + \sum_{\eta \geq 3}\left(\frac{\partial^{\eta+1} u}{\partial y \partial x^\eta}\right)_{i,j}\frac{h^\eta}{\eta!}$$

aussi,

$$\left(\frac{\partial u}{\partial y}\right)_{i-1,j}=\left(\frac{\partial u}{\partial y}\right)_{i,j}-h\left(\frac{\partial^2 u}{\partial y\partial x}\right)_{i,j}+\frac{1}{2}h^2\left(\frac{\partial^3 u}{\partial y\partial x^2}\right)_{i,j}+\sum_{\eta\geq 3}\left(\frac{\partial^{\eta+1}u}{\partial y\partial x^{\eta}}\right)_{i,j}\frac{(-h)^{\eta}}{\eta!}$$

par conséquent,

$$\left(\frac{\partial u}{\partial y}\right)_{i+1,j}-\left(\frac{\partial u}{\partial y}\right)_{i-1,j}=2h\left(\frac{\partial^2 u}{\partial y\partial x}\right)_{i,j}+\sum_{\eta\geq 3}\left(\frac{\partial^{\eta+1}u}{\partial y\partial x^{\eta}}\right)_{i,j}\left(1-(-1)^{\eta}\right)\frac{h^{\eta}}{\eta!}$$

ou,

$$\left(\frac{\partial^2 u}{\partial y\partial x}\right)_{i,j}=\frac{(\partial u/\partial y)_{i+1,j}-(\partial u/\partial y)_{i-1,j}}{2h}+O\left(h^2\right)$$

or,

$$\left(\frac{\partial u}{\partial y}\right)_{i+1,j}=\frac{u_{i+1,j+1}-u_{i+1,j-1}}{2k}+O\left(k^2\right)\ \text{ et }\ \left(\frac{\partial u}{\partial y}\right)_{i-1,j}=\frac{u_{i-1,j+1}-u_{i-1,j-1}}{2k}+O\left(k^2\right)$$

alors,

$$\left(\frac{\partial^2 u}{\partial y\partial x}\right)_{i,j}=\frac{u_{i+1,j+1}-u_{i+1,j-1}-u_{i-1,j+1}+u_{i-1,j-1}}{4hk}+O\left(h^2,k^2\right)$$

Avec le même raisonnement,

$$\left(\frac{\partial^2 u}{\partial x\partial y}\right)_{i,j}=\frac{u_{i+1,j+1}-u_{i+1,j-1}-u_{i-1,j+1}+u_{i-1,j-1}}{4hk}+O\left(h^2,k^2\right)=\left(\frac{\partial^2 u}{\partial y\partial x}\right)_{i,j}$$

Les dérivées partielles, $\partial^2 u/\partial x\partial y$ et $\partial^2 u/\partial y\partial x$, peuvent-être approchées par le schéma aux différences finies :

$$\left(\frac{\partial^2 u}{\partial x\partial y}\right)_{i,j}\approx\frac{u_{i+1,j+1}-u_{i+1,j-1}-u_{i-1,j+1}+u_{i-1,j-1}}{4hk}$$

Les schémas aux différences finies des dérivées partielles, peuvent être utilisés dans l'approximation des expressions différentielles, par exemple l'approximation du Laplacien $\Delta u=\partial^2 u/\partial x^2+\partial^2 u/\partial y^2$ est exprimé par :

$$\Delta u\approx\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h^2}+\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{k^2}$$

La même approche est appliquée si en plus la fonction solution *u* dépend du temps *t*, c'est-à-dire $u(x,\ y,\ t)$. Si τ est le pas de discrétisation pour le temps, la dérivée partielle de la fonction *u* par rapport à *t* est approchée par le schéma aux différences finies centrale :

$$\left(\frac{\partial u}{\partial t}\right)_{i,j}^{n}\approx\frac{u_{i,j}^{n+1}-u_{i,j}^{n-1}}{2\tau}$$

Les schémas aux différences finies (Fig. 7.2) ainsi obtenus représentent que le ''sommet de l'iceberg'' dans l'approximation des dérivées partielles, nous avons vu que les schémas aux différences finie arrière et avant de la dérivée $(\partial u/\partial x)_{i,\ j}$ ont une erreur de troncature de l'ordre de $O(h)$, par conséquent ces deux approximations sont appelées aussi les schémas aux différences de première ordre, on peut montrer (Hirsch, 2007) que les schémas aux

différences de deuxième ordre avant et arrière et centrales de la dérivée $(\partial u/\partial x)_{i,j}$ sont explicités par exemple par :

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{-u_{i+2,j} + 4u_{i+1,j} - 3u_{i,j}}{2h} + O\left(h^2\right)$$

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{3u_{i,j} - 4u_{i-1,j} + 3u_{i-2,j}}{2h} + O\left(h^2\right)$$

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{u_{i+1,j+1} - u_{i-1,j+1}}{6h} + 2\frac{u_{i+1,j} - u_{i-1,j}}{3h} + \frac{u_{i+,j-1} - u_{i-1,j-1}}{6h} + O\left(h^2\right)$$

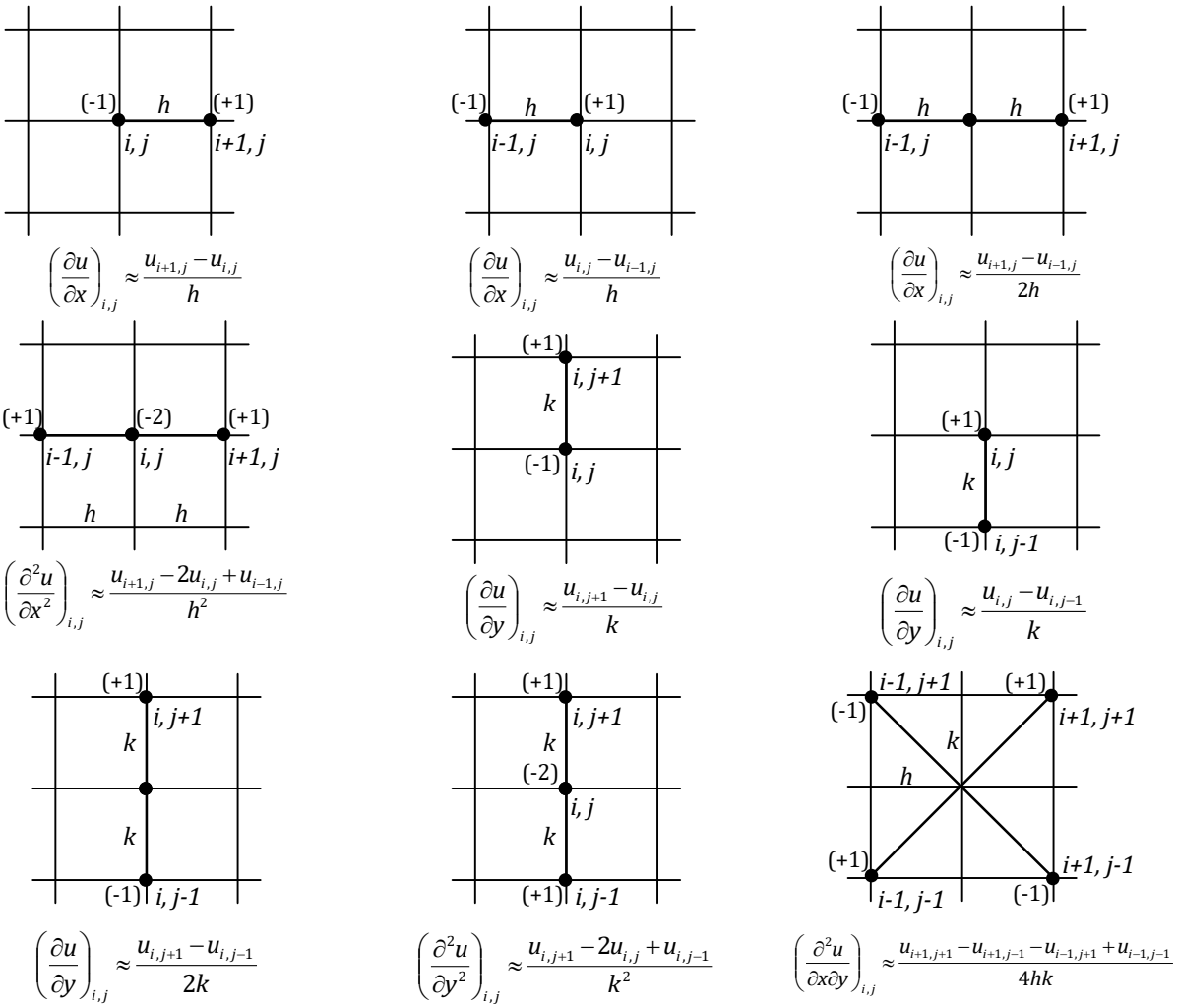


Fig. 7.2 : Schémas aux différences finies des dérivées partielles.

Aussi les schémas aux différences avant et arrière de troisième ordre de la dérivée partielle $(\partial u/\partial x)_{i,j}$ peuvent être explicités par exemple comme :

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{-u_{i+2,j} + 6u_{i+1,j} - 3u_{i,j} - 2u_{i-1,j}}{6h} + O\left(h^3\right)$$

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{2u_{i+1,j} + 3u_{i,j} - 6u_{i-1,j} + u_{i-2,j}}{6h} + O\left(h^3\right)$$

La dérivée partielle $\left(\partial^2 u/\partial x^2\right)_{i,\ j}$ a été approchée par $\left(u_{i+1,j}-2u_{i,j}+u_{i-1,j}\right)/h^2$ avec une erreur de troncature de deuxième ordre. L'approximation de la même dérivée avec une erreur de troncature de l'ordre 4 est exprimé par :

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} \approx \frac{-u_{i+2,j}+16u_{i+1,j}-30u_{i,j}+16u_{i-1,j}-u_{i-2,j}}{12h^2}$$

Cette approximation, dépend de cinq termes $u_{i+2,j}$, $u_{i+1,j}$, $u_{i,j}$, $u_{i-1,j}$ et $u_{i-2,j}$ (Fig. 7.3).

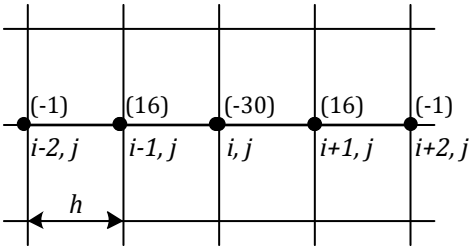


Fig. 7.3 : Schéma aux différences finies d'ordre 4 de la dérivée $\left(\partial^2 u/\partial x^2\right)_{i,\ j}$.

2. Equations aux différences

Les expressions algébriques citées précédemment sont des approximations des différents types de dérivée partielle. Soit à titre d'exemple l'équation différentielle aux dérivées partielles :

$$\frac{\partial u}{\partial t}-\frac{\partial^2 u}{\partial x^2}=0$$

Qui peut être approché par l'équation :

$$\frac{u_i^{n+1}-u_i^{n-1}}{2\tau}-\frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{h^2}=0 \quad \text{ou,} \quad h^2\left(u_i^{n+1}-u_i^{n-1}\right)-2\tau\left(u_{i+1}^n-2u_i^n+u_{i-1}^n\right)=0$$

L'équation obtenue est une relation algébrique entre les termes u_i^{n+1}, u_i^{n-1}, u_{i+1}^n, u_i^n et u_{i-1}^n s'appelle *équation aux différences* ou le *schéma aux différences de l'équation différentielle* (Fig. 7.4). Il est clair que l'équation aux différences, c'est une approximation de l'équation différentielle dont l'erreur de l'approximation peut être évaluée puisque :

$$\left(\frac{\partial u}{\partial t}\right)_i^n=\frac{u_i^{n+1}-u_i^{n-1}}{2\tau}+O\left(\tau^2\right)$$

et,

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

soit,

$$\left(\frac{\partial u}{\partial t}\right)_i^n-\left(\frac{\partial^2 u}{\partial t^2}\right)_i^n=\frac{u_i^{n+1}-u_i^{n-1}}{2\tau}-\frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{h^2}+O\left(h^2,\ \tau^2\right)=0$$

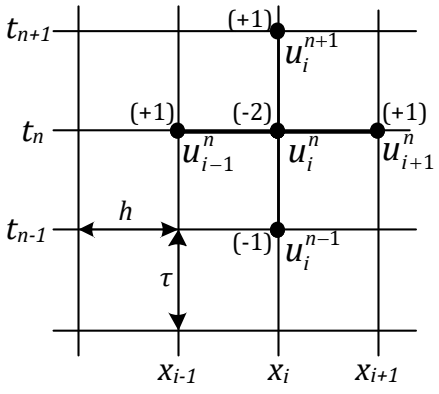


Fig. 7.4 : Disposition des différents points d'une équation aux différences.

$O\left(h^2,\ \tau^2\right)$ est l'erreur de troncature de l'équation aux différences.

La résolution numérique d'une équation différentielle au dérivée partielle consiste à calculer pour chaque point $\left(x_i,\ t_n\right)$ (nœud) la valeur du terme u_i^n au moyen de l'équation aux différences, celle-ci qui est un schéma récursif de deux (i et n) ou de trois (i, j et n) degrés de libertés, dont son emploi nécessite une grande précaution. Une équation aux différences formée doit être consistante, convergente et stable pour quelle donne des résultats acceptables (Franz *et al.*, 1997).

Un schéma aux différences finies est dit consistant ou compatible, si les équations aux différences sont construites à partir d'une expression équivalente à l'équation de départ au second ordre près. Cela signifie que l'erreur de troncature $O\left(h^2,\ \tau^2\right)$ tend vers zéro, lorsque les dimensions du maillage h et τ tendent vers zéro.

Un schéma aux différences est stable si la différence entre la solution numérique et la solution exacte reste borné si t tend vers l'infini. La stabilité d'un schéma aux différences finies est une propriété globale de l'algorithme qui en découle. Elle concerne essentiellement le comportement numérique qui se manifeste lorsque les pas de discrétisation tendent tous vers 0.

La stabilité d'un schéma aux différences finies, peut être déterminée grâce à l'analyse de von Neumann (Charney *et al.*, 1950). Pour étudier la stabilité d'un schéma aux différences considérant, $u_i^n = g^n e^{ji\theta}$ ou g est le facteur d'amplification qui est une fonction de θ et j est le nombre imaginaire c'est-à-dire $\left(j^2 = -1\right)$ par substitution dans l'équation aux différences comme suit :

$$\frac{g^{n+1}e^{ji\theta} - g^{n-1}e^{ji\theta}}{2\tau} - \frac{g^n e^{j(i+1)\theta} - 2g^n e^{ji\theta} + g^n e^{j(i-1)\theta}}{h^2} = 0 \ \Rightarrow\ \frac{g - g^{-1}}{2\tau} - \frac{e^{j\theta} - 2 + e^{-j\theta}}{h^2} = 0$$

Pour que le schéma soit stable, il faut que $\left|g\left(\theta\right)\right| \le 1$.

Un schéma aux différences finies est dit convergent si la différence entre les solutions numériques en un point fixe tends vers zéro quand lorsque les dimensions du maillage h et τ tendent vers zéro. Sous certaines hypothèses, le théorème de Lax (Lax et Richtmyer, 1956) montre que la stabilité est une condition nécessaire et suffisante pour assurer la convergence

Généralement, il y a deux formes de schémas aux différences qui sont utilisés pour résoudre numériquement les équations différentielles aux dérivées partielles : le schéma explicite et le schéma implicite.

3. Schémas explicites et implicites

Dans un schéma explicite, les valeurs inconnues sont déterminées d'une façon séquentielle le long du temps. Plusieurs schémas explicites (Chaudhry, 2008, Sturm, 2001) sont utilisés pour exprimer les différences finies $\partial u/\partial t$ et $\partial u/\partial x$, leurs buts consistent à la détermination de u_i^{n+1} sachant que u_i^n sont connues quelques soit i. Parmi les schémas explicites citons à titre d'exemple les schémas de Lax–Friedrish et Leapfrog qui peuvent être utilisés dans n'importe quel type d'EDP. Autres types de schémas aux différences tel que le schéma de Upwind, Lax &

Wendroff (1960) ou de MacCormack *et al.*, (1972) sont utilisés pour les EDPs de type hyperbolique.

Suivant le schéma explicite de Lax–Friedrichs, les deux dérivées partielles $\partial u/\partial t$, $\partial u/\partial x$ et la fonction u sont exprimées pour un point de coordonnées $(x_i,\ t_n)$ comme suit (Fig. 7.5) :

$$\left(\frac{\partial u}{\partial t}\right)_i^n \approx \frac{1}{\tau}\left(u_i^{n+1} - \frac{1}{2}\left(u_{i+1}^n + u_{i-1}^n\right)\right), \quad \left(\frac{\partial u}{\partial x}\right)_i^n \approx \frac{u_{i+1}^n - u_{i-1}^n}{2h} \quad \text{et}$$

$$u_i^n \approx \frac{1}{2}\left(u_{i+1}^n + u_{i-1}^n\right)$$

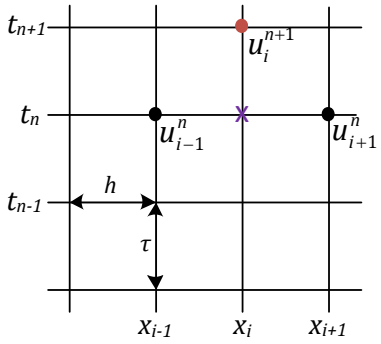


Fig. 7.5 : Schéma diffusive de Lax-Friedrichs.

Et suivant le schéma explicite Leapfrog, les dérivées partielles $\partial u/\partial t$, $\partial u/\partial x$ sont exprimées par les schémas aux différences finies centrales pour un point de coordonnées $(x_i,\ t_n)$, c'est-à-dire (Fig. 7.6) :

$$\left(\frac{\partial u}{\partial t}\right)_i^n \approx \frac{u_i^{n+1} - u_i^{n-1}}{2\tau} \text{ et } \left(\frac{\partial u}{\partial x}\right)_i^n \approx \frac{u_{i+1}^n - u_{i-1}^n}{2h}$$

u est approché par u_i^n (Liggett *et al.*, 1975). Suivant les conditions initiales et aux limites du problème le schéma explicite permet de calculer pour un temps t_{n+1} fixé séquentiellement les valeurs u_i^{n+1}, $\forall i$, par contre le schéma implicite peut être déduit du schéma explicite du problème, comme il peut être obtenu à partir du schéma aux différences des de la fonction u et les deux dérivées partielles $\partial u/\partial t$, $\partial u/\partial x$ dans le point de coordonnées $(x_{i+1},\ t_{n+1})$ par le schéma de Preissmann (1961) explicité comme suit :

$$\frac{\partial u}{\partial t} \approx \varphi\left(\frac{u_{i+1}^{n+1} - u_{i+1}^n}{\tau}\right) + (1-\varphi)\left(\frac{u_i^{n+1} - u_i^n}{\tau}\right)$$

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

$$u \approx \theta\left[\varphi u_{i+1}^{n+1} + (1-\varphi)u_i^{n+1}\right] + (1-\theta)\left[\varphi u_{i+1}^n + (1-\varphi)u_i^n\right]$$

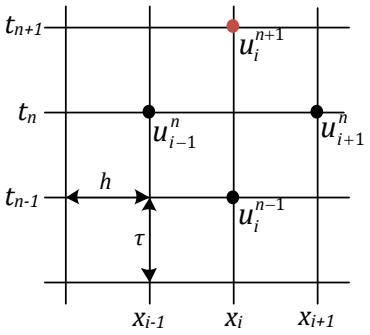


Fig. 7.6 : Schéma leapfrog.

φ et θ sont respectivement des coefficients de pondération en temps et en espace, prenant une valeur entre 0 et 1. Pour $\varphi = 0.5$, ces équations donnent le schéma classique de Preissmann ; il en résulte que :

- pour $\theta = 0$; le schéma devient complètement explicite,
- pour $\theta = 1$; le schéma devient complètement implicite,
- pour $\theta = 0.5$; le schéma est centré-implicite à quatre points.

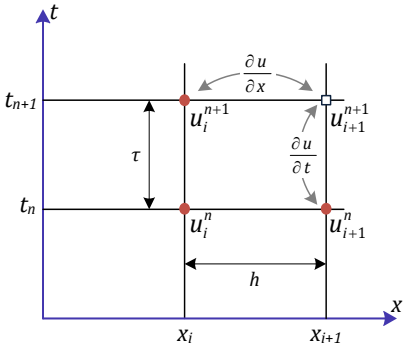


Fig. 7.7 : Schéma implicite.

Le schéma implicite sert à calculer la valeur u_{i+1}^{n+1} sachant

que les valeurs u_{i+1}^n, u_i^n et u_i^{n+1} sont connues (Fig. 7.7). Le schéma implicite de Preissmann est utilisé dans le calcul des écoulements non permanent à surface libre (Liggett *et al.*, 1975 ; Cunge *et al.*, 1980).

3.1. EDPs hyperboliques. Equation de transport

L'équation de transport ou de convection unidimensionnelle est une EDP de type hyperbolique, traduit le transport d'une quantité scalaire u définie par une unité de volume avec une vitesse de convection a. Soit le problème avec les conditions initiale et au limite inférieure :

$$\begin{cases} \dfrac{\partial u}{\partial t} + a\dfrac{\partial u}{\partial x} = 0, \\ u(x,\ 0) = \varphi(x), \quad 0 \le x \le L, \\ u(0,\ t) = \phi(t), \quad t \ge 0. \end{cases}$$

Ce type de problème possède la solution exacte suivante (Strikwerda, 2004) :

$$u(x,\ t) = \begin{cases} \varphi(x - at) & \text{si} \quad x - at > 0, \\ \phi\left(t - \dfrac{x}{a}\right) & \text{si} \quad x - at < 0. \end{cases}$$

Pour appliquer la méthode des différences finies, soit h et τ les pas de discrétisation de l'espace et du temps, par conséquent $u_i^n = u(x_i,\ t_n) = u(ih,\ n\tau)$ (Fig. 7.8). Les conditions du problème peuvent être interprété comme suit :

$$u(x,\ 0) = \varphi(x) \ \Rightarrow\ u_i^0 = \varphi(ih) \quad \forall i = 0,\ 1,\ \ldots,\ N$$

$$u(0,\ t) = \phi(x) \ \Rightarrow\ u_0^n = \phi(n\tau) \quad \forall n = 0,\ 1,\ 2,\ \ldots$$

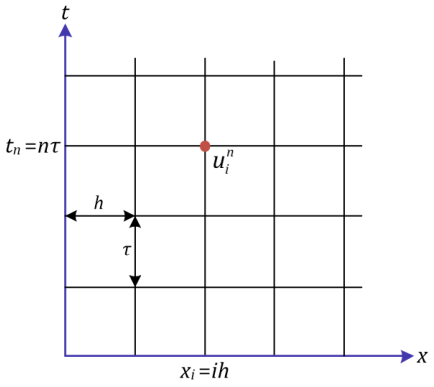


Fig. 7.8 : Discrétisation du domaine.

La résolution numérique du problème revient donc à calculer u_i^n quel que soit $i = 0,\ 1,\ \ldots,\ N$ et $n = 0,\ 1,\ 2,\ \ldots$

3.1.1. Schéma d'Euler

Le schéma explicite d'Euler est le plus simple schéma aux différences finies, qui consiste à approcher $(\partial u/\partial x)_i^n$ par un schéma central et $(\partial u/\partial t)_i^n$ par un schéma avant de première ordre, par conséquent l'équation aux différences de l'EDP est :

$$\frac{u_i^{n+1} - u_i^n}{\tau} + a\frac{u_{i+1}^n - u_{i-1}^n}{2h} = 0 \ \Rightarrow\ u_i^{n+1} = \frac{\lambda}{2}u_{i-1}^n + u_i^n - \frac{\lambda}{2}u_{i+1}^n \quad \text{ou,} \quad \lambda = a\frac{\tau}{h}$$

Pour utiliser cette équation aux différences, une analyse de la stabilité est essentielle, soit $u_i^n = g^n e^{ji\theta}$ par conséquent :

$$g^{n+1}e^{ji\theta} = \frac{\lambda}{2}g^n e^{j(i-1)\theta} + g^n e^{ji\theta} - \frac{\lambda}{2}g^n e^{j(i+1)\theta} \ \Rightarrow g(\theta) = 1 - j\lambda\sin\theta$$

La fonction $g(\theta)$ est une fonction complexe, alors la valeur absolue de cette fonction est :

$$|g(\theta)| = \sqrt{1 + \lambda^2\sin^2\theta}$$

Elle strictement supérieure à 1, c'est-à-dire ce schéma aux différences est instable.

De ce schéma explicite d'Euler, soit le schéma d'Euler implicite, qui consiste à approcher le second terme de l'équation pour le niveau n+1, c'est-à-dire :

$$\frac{u_i^{n+1}-u_i^n}{\tau}+a\frac{u_{i+1}^{n+1}-u_{i-1}^{n+1}}{2h}=0 \;\Rightarrow\; \lambda u_{i-1}^{n+1}-2u_i^{n+1}-\lambda u_{i+1}^{n+1}=-2u_i^n$$

L'analyse de la stabilité conduit à la fonction complexe :

$$g(\theta)=\frac{1}{1+j\lambda\sin\theta} \;\Rightarrow\; |g(\theta)|=\sqrt{\frac{1}{\left(1+\lambda^2\sin^2\theta\right)^2}}\le 1$$

Qui montre que ce schéma est stable.

L'équation aux différences obtenue conduit à un système d'équations linéaires qui peut être s'écrit sous la forme matricielle tridiagonale suivante :

$$\begin{bmatrix} -2 & -\lambda & 0 & & & 0 \\ \lambda & -2 & -\lambda & & & \\ 0 & \lambda & -2 & -\lambda & & \\ & & \lambda & -2 & -\lambda & 0 \\ & & & \lambda & -2 & -\lambda \\ 0 & & & 0 & \lambda & -2 \end{bmatrix} \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ . \\ . \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{bmatrix} = -2 \begin{bmatrix} u_1^n \\ u_2^n \\ u_3^n \\ . \\ . \\ u_N^n \\ u_N^n \end{bmatrix}$$

Sous forme compacte :

$$\mathbf{M}\mathbf{U}^{n+1}=-2\mathbf{U}^n \;\Rightarrow\; \mathbf{U}^{n+1}=-2\mathbf{M}^{-1}\mathbf{U}^n \;\Rightarrow\; \mathbf{U}^{n+1}=(-2)^{n+1}\left(\mathbf{M}^{-1}\right)^{n+1}\mathbf{U}^0$$

3.1.2. Schémas Upwind

Les schémas explicites Upwind sont généralement de trois formes suivant l'ordre de l'approximation de la dérivée partielle $(\partial u/\partial x)_i^n$. Pour le schéma Upwind de premier ordre, la dérivée partielle $(\partial u/\partial x)_i^n$ est approchée par un schéma aux différences de premier ordre arrière, par conséquent l'équation aux différences est explicitée comme suit :

$$\frac{u_i^{n+1}-u_i^n}{\tau}+a\frac{u_i^n-u_{i-1}^n}{h}=0 \;\Rightarrow\; u_i^{n+1}=\lambda u_{i-1}^n+(1-\lambda)u_i^n$$

Le schéma Upwind implicite de premier ordre peut être déduit :

$$\frac{u_i^{n+1}-u_i^n}{\tau}+a\frac{u_i^{n+1}-u_{i-1}^{n+1}}{h}=0 \;\Rightarrow\; \lambda u_{i-1}^n-(1+\lambda)u_i^{n+1}=-u_i^n$$

Pour le schéma Upwind de deuxième ordre, la dérivée partielle $(\partial u/\partial x)_i^n$ est approchée par un schéma aux différences de deuxième ordre arrière, par conséquent l'équation aux différences pour ces deux schémas i = 1, 2, 3, . . ., N et n = 0, 1, 2, . . . est explicitée comme suit :

$$\frac{u_i^{n+1}-u_i^n}{\tau}+a\frac{3u_i^n-4u_{i-1}^n+u_{i-2}^n}{2h}=0 \;\Rightarrow\; u_i^{n+1}=-\frac{\lambda}{2}u_{i-2}^n+2\lambda u_{i-1}^n+\left(1-\frac{3}{2}\lambda\right)u_i^n$$

Le schéma Upwind implicite de seconde ordre peut être s'écrit alors :

$$\frac{u_i^{n+1}-u_i^n}{\tau}+a\frac{3u_i^{n+1}-4u_{i-1}^{n+1}+u_{i-2}^{n+1}}{2h}=0 \;\Rightarrow\; \lambda u_{i-2}^{n+1}-4\lambda u_{i-1}^{n+1}+(2+3\lambda)u_i^{n+1}=2u_i^n$$

Dans le cas d'un schéma Upwind de troisième ordre, la dérivée partielle $(\partial u/\partial x)_i^n$ est approchée par un schéma aux différences arrière de troisième ordre, l'équation aux différences devient :

$$\frac{u_i^{n+1}-u_i^n}{\tau}+a\frac{2u_{i+1}^n+3u_i^n-6u_{i-1}^n+u_{i-2}^n}{6h}=0 \;\Rightarrow\; u_i^{n+1}=-\frac{1}{6}\lambda u_{i-2}^n+\lambda u_{i-1}^n+\left(1-\frac{1}{2}\lambda\right)u_i^n-\frac{1}{3}\lambda u_{i+1}^n$$

Mais pour le calcul de u_N^{n+1}, soit $(\partial u/\partial x)_{N-1}^n=(\partial u/\partial x)_N^n$, c'est-à-dire :

$$\frac{2u_N^n+3u_{N-1}^n-6u_{N-2}^n+u_{N-3}^n}{6h}=\frac{2u_{N+1}^n+3u_N^n-6u_{N-1}^n+u_{N-2}^n}{6h}\Rightarrow u_{N+1}^n=\frac{-u_N^n+9u_{N-1}^n-7u_{N-2}^n+u_{N-3}^n}{2}$$

Par substitution dans l'équation aux différences u_N^{n+1} peut être calculé :

$$u_N^{n+1}=-\frac{\lambda}{6}u_{N-3}^n+\frac{\lambda}{6}u_{N-2}^n-\frac{\lambda}{2}u_{N-3}^n+\left(1-\frac{\lambda}{3}\right)u_N^n$$

Le schéma Upwind implicite de seconde ordre peut être s'écrit alors :

$$\frac{u_i^{n+1}-u_i^n}{\tau}+a\frac{2u_{i+1}^{n+1}+3u_i^{n+1}-6u_{i-1}^{n+1}+u_{i-2}^{n+1}}{6h}=0$$

c'est-à-dire,

$$\lambda u_{i-2}^{n+1}-6\lambda u_{i-1}^{n+1}+3(2+\lambda)u_i^{n+1}+2\lambda u_{i+1}^{n+1}=6u_i^n$$

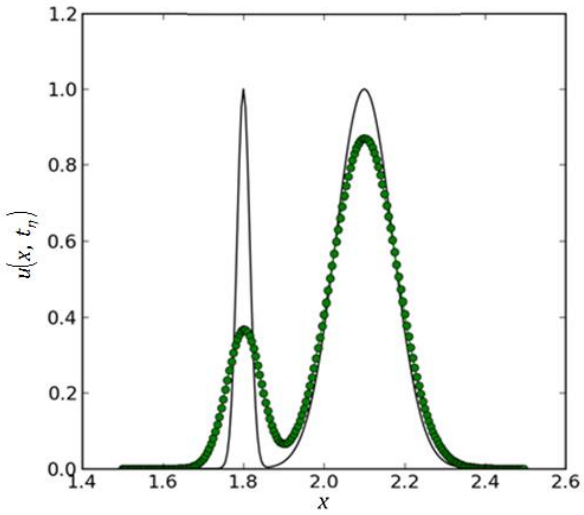


Fig. 7.9 : Exemple d'application du schéma Upwind à l'équation du transport.

3.1.3. Schéma de Lax-Friedrichs

L'équation aux différences obtenue suivant le schéma de Lax-Friedrichs est :

$$\frac{u_i^{n+1}-\frac{1}{2}\left(u_{i+1}^n+u_{i-1}^n\right)}{\tau}+a\frac{u_{i+1}^n-u_{i-1}^n}{2h}=0 \;\Rightarrow\; \frac{1}{2}(1+\lambda)u_{i-1}^n+\frac{1}{2}(1-\lambda)u_{i+1}^n=u_i^{n+1}$$

Pour étudier la stabilité de ce schéma soit, $u_i^n=g^n e^{ji\theta}$ par conséquent :

$$\frac{1}{2}(1+\lambda)g^n e^{j(i-1)\theta}+\frac{1}{2}(1-\lambda)g^n e^{j(i+1)\theta}=g^{n+1}e^{ji\theta} \;\Rightarrow\; g(\theta)=\cos\theta-j\lambda\sin\theta$$

soit,

$$|g(\theta)|=\sqrt{\cos^2\theta+\lambda^2\sin^2\theta}=\sqrt{1+(\lambda^2-1)\sin^2\theta}$$

Il est clair que $|g(\theta)|\le 1 \;\Rightarrow\; \lambda\le 1$ c'est la condition de stabilité du schéma appelé aussi la condition CFL (Courant–Friedrichs–Lewy) (Courant *et al.*, 1928).

Remarquons que ce schéma est un système linéaire tri-diagonale, peut-être s'écrit sous la forme matricielle suivante :

$$\mathbf{U}^{n+1}=\frac{1}{2}\mathbf{M}\mathbf{U}^n$$

alors,

$$\mathbf{U}^1=\frac{1}{2}\mathbf{M}\mathbf{U}^0 \text{ et } \mathbf{U}^2=\frac{1}{2}\mathbf{M}\mathbf{U}^1=\frac{1}{2^2}\mathbf{M}^2\mathbf{U}^0,\ \ldots,\ \mathbf{U}^{n+1}=\frac{1}{2^{n+1}}\mathbf{M}^{n+1}\mathbf{U}^0$$

avec,

$$\mathbf{M}=\begin{bmatrix} 0 & 1-\lambda & 0 & & & & 0 \\ 1+\lambda & 0 & 1-\lambda & & & & \\ 0 & 1+\lambda & 0 & 1-\lambda & & & \\ & & . & . & . & & \\ & & & 1+\lambda & 0 & 1-\lambda & 0 \\ & & & & 1+\lambda & 0 & 1-\lambda \\ 0 & & & & 0 & 1+\lambda & 0 \end{bmatrix},\quad \mathbf{U}^n=\begin{bmatrix} u_1^n \\ u_2^n \\ u_3^n \\ . \\ . \\ u_N^n \\ u_N^n \end{bmatrix},\quad \mathbf{U}^{n+1}=\begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ . \\ . \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{bmatrix} \text{ et } \mathbf{U}^0=\begin{bmatrix} \varphi(h) \\ \varphi(2h) \\ \varphi(3h) \\ . \\ . \\ \varphi((N-1)h) \\ \varphi(Nh) \end{bmatrix}$$

3.1.4. Schéma Leapfrog

L'équation aux différences obtenue suivant le schéma Leapfrog est :

$$\frac{u_i^{n+1}-u_i^{n-1}}{2\tau}+a\frac{u_{i+1}^n-u_{i-1}^n}{2h}=0 \;\Rightarrow\; u_i^{n+1}=\lambda u_i^n+u_i^{n-1}-\lambda u_{i+1}^n$$

Dans ce schéma, on suppose que $(\partial u/\partial x)_{N-1}^n=(\partial u/\partial x)_N^n$ par conséquent :

$$\frac{u_N^n-u_{N-2}^n}{2h}=\frac{u_{N+1}^n-u_{N-1}^n}{2h} \;\Rightarrow\; u_{N+1}^n=u_N^n+u_{N-1}^n-u_{N-2}^n$$

Aussi on remarque que pour calculer u_i^2 les valeurs u_i^0 sont connues et u_i^1 peuvent être estimées suivant la proposition suivante :

$$\text{quand } \tau\to 0 \;\Rightarrow\; u_i^1\approx u_i^0$$

alors,

$$u_i^2=\begin{cases}(\lambda+1)u_i^0-\lambda u_{i+1}^1, & i=1,\ 2,\ 3,\ \ldots,\ N-1,\\ u_i^0-\lambda u_{i-1}^0+\lambda u_{i-2}^0, & i=N.\end{cases}$$

Par conséquent le calcul de u_i^{n+1} pour n = 2, 3, 4, . . . se fait comme suit :

$$u_i^{n+1}=\begin{cases}\lambda u_i^n+u_i^{n-1}-\lambda u_{i+1}^n, & i=1,\ 2,\ 3,\ \ldots,\ N-1,\\ u_i^n-\lambda u_{i-1}^n+\lambda u_{i-2}^n, & i=N.\end{cases}$$

On peut considérer aussi $u_{N-1}^{n+1}=u_N^{n+1}$:

$$u_i^2=\begin{cases}(\lambda+1)u_i^0-\lambda u_{i+1}^1, & i=1,\ 2,\ 3,\ \ldots,\ N-1,\\ u_{i-1}^2, & i=N.\end{cases}$$

Et le calcul de u_i^{n+1} pour, n = 2, 3, 4, . . . se fait comme suit :

$$u_i^{n+1}=\begin{cases}\lambda u_i^n+u_i^{n-1}-\lambda u_{i+1}^n, & i=1,\ 2,\ 3,\ \ldots,\ N-1,\\ u_{i-1}^{n+1}, & i=N.\end{cases}$$

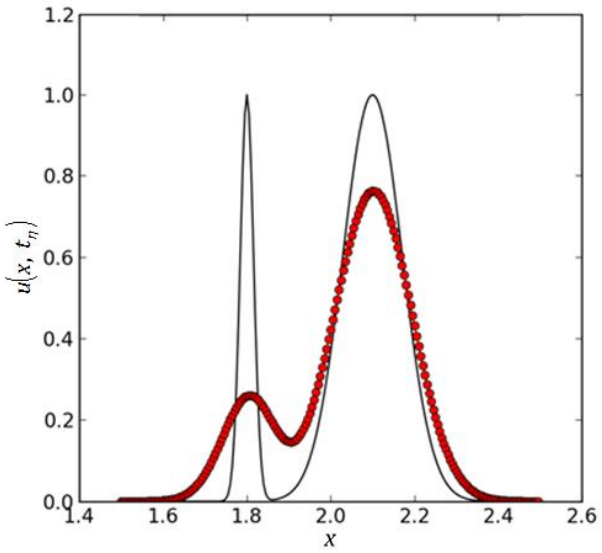


Fig. 7.10 : Exemple d'application du schéma de Lax–Friedrichs à l'équation du transport.

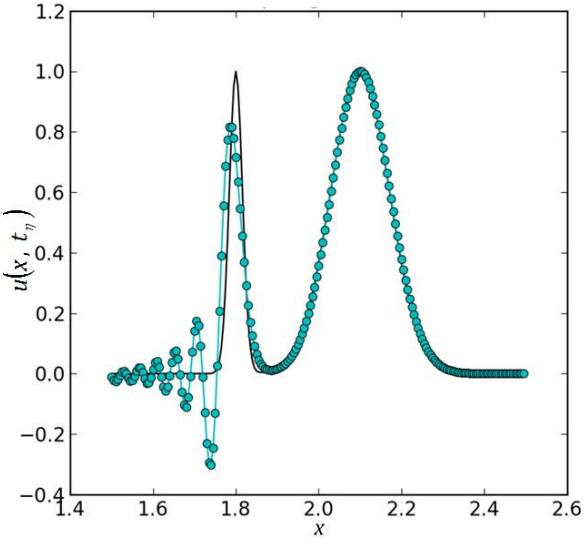


Fig. 7.11 : Exemple d'application du schéma leapfrog à l'équation du transport.

et le calcul de u_i^{n+1} pour, n = 2, 3, 4, . . . se fait comme suit :

$$u_i^{n+1} = \begin{cases} \lambda u_i^n + u_i^{n-1} - \lambda u_{i+1}^n, & i = 1,\ 2,\ 3,\ ...,\ N-1, \\ u_{i-1}^{n+1}, & i = N. \end{cases}$$

Le schéma Leapfrog (Fig. 7.11) est stable si la condition de Courant–Friedrichs–Lewy CFL (Courant *et al.*, 1928) est vraie.

3.1.5. Schéma MacCormack

Le schéma explicite de MacCormack (1969) s'effectue en deux étapes :

- *Etape de prédiction*, qui consiste à calculer la quantité $\tilde{u}_i^{n+1}$ obtenue à partir de la considération des schémas pour les deux dérivées partielles :

$$\left(\frac{\partial u}{\partial t}\right)_i^n \approx \frac{\tilde{u}_i^{n+1} - u_i^n}{\tau} \quad \text{et} \quad \left(\frac{\partial u}{\partial x}\right)_i^n \approx \frac{u_{i+1}^n - u_i^n}{h}$$

par conséquent,

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \ \Rightarrow\ \frac{\tilde{u}_i^{n+1} - u_i^n}{\tau} + a\frac{u_{i+1}^n - u_i^n}{h} = 0$$

ou,

$$\tilde{u}_i^{n+1} = u_i^n - \lambda\left(u_{i+1}^n - u_i^n\right)$$

- *Etape de correction*, qui consiste à calculer la quantité u_i^{n+1} par :

$$u_i^{n+1} = \frac{u_i^n + \tilde{u}_i^{n+1}}{2} - \frac{\lambda}{2}\left(\tilde{u}_i^{n+1} - \tilde{u}_{i-1}^{n+1}\right)$$

3.2. EDPs paraboliques. Equation de la chaleur ou de la diffusion

En physique, l'équation de la chaleur ou de la diffusion unidimensionnelle est une équation aux dérivées partielles de seconde ordre de type parabolique, soit le problème suivant avec les conditions initiales et au limite pour $0 \le x \le L$ et $t \ge 0$:

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} \quad \text{avec} \quad u(x,\ 0) = \varphi(x),\ \ u(0,\ t) = T_1(t) \quad \text{et} \quad u(L,\ t) = T_2(t)$$

$u(x,\ t)$ est la variation de la température pour un point de position x pendant le temps t. Plusieurs phénomènes ont le même modèle mathématique, par exemple le rabattement du niveau phréatique d'une nappe libre lors du pompage en régime permanent, diffusion des substances (pollution) dans un écoulement d'un fluide visqueux…

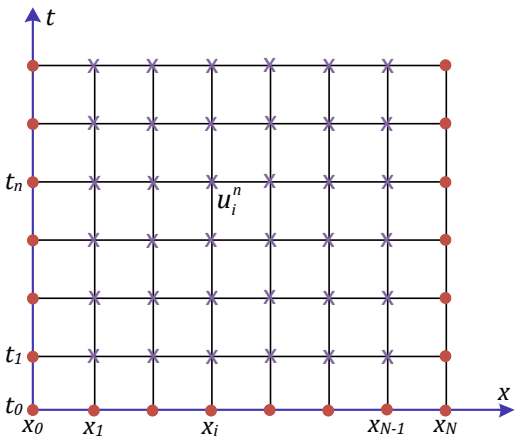


Fig. 7.12 : Conditions initiale et aux limites.

Sur le domaine discrétisé (Fig. 7.12) $u_i^n = u(x_i,\ t_n) = u(x_0 + ih,\ t_0 + n\tau)$ ou $i = 0,\ 1,\ ...,\ N$ et $n = 0,\ 1,\ . . .$, la condition initiale $u(x,\ 0) = \varphi(x)$ s'écrit $u_i^0 = \varphi(x_i)$ pour $i = 0,\ 1,\ ...,\ N$ et les deux conditions

aux limites $u(0,\ t)=T_1(t)$ et $u(L,\ t)=T_2(t)$ s'écrivent respectivement comme $u_0^n=T_1(t_n)$ et $u_N^n=T_2(t_n)$ pour $n=0,\ 1,\ \ldots$

Suivant ces conditions, il est clair que les valeurs de la solution numérique de l'équation différentielle qu'il faut déterminer correspondent à u_i^n pour $i=1,\ 2,\ \ldots,\ N-1$ et $n=1,\ 2,\ \ldots,$

3.2.1. Schémas classiques

Le plus simple schéma aux différences est le schéma avant pour la dérivée partielle $\partial u/\partial t$, soit :

$$\frac{u_i^{n+1}-u_i^n}{\tau}=a\frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{h^2}\ \Rightarrow\ u_i^{n+1}=\mu u_{i+1}^n+(1-2\mu)u_i^n+\mu u_{i-1}^n$$

où, μ est une grandeur adimensionnelle exprimée par :

$$\mu=a\frac{\tau}{h^2}$$

Pour étudier la stabilité de ce schéma soit : $u_i^n=g^n e^{ji\theta}=g^n(\cos(i\theta)+j\sin(i\theta))$, par substitution dans l'équation aux différences on obtient :

$$g^{n+1}e^{ji\theta}=\mu g^n e^{j(i+1)\theta}+(1-2\mu)g^n e^{ji\theta}+\mu g^n e^{j(i-1)\theta}\ \Rightarrow\ g=\mu\left(e^{j\theta}+e^{-j\theta}\right)+(1-2\mu)$$

où,

$$g(\theta)=1-4\mu\sin^2\frac{1}{2}\theta$$

Pour que ce schéma soit stable il faut que $|g(\theta)|\le 1,$ par conséquent :

$$-1\le 1-4\mu\sin^2\frac{1}{2}\theta\le 1\ \Rightarrow\ -2\le -4\mu\sin^2\frac{1}{2}\theta\le 0\ \Rightarrow\ 4\mu\sin^2\frac{1}{2}\theta\le 20\ \Rightarrow\ \mu\le\frac{1}{2}$$

Il faut choisir les pas de discrétisation τ et h de façon à avoir $\mu<1/2$ pour que le schéma aux différences soit stable.

Suivant les conditions initiale et aux limites, ce schéma permet de calculer facilement u_i^{n+1} pour $i=1,\ 2,\ \ldots,\ N-1$ et $n=0,\ 1,\ \ldots$ Soit l'algorithme :

$$\left.\begin{cases} u_i^0=\varphi(x_0+ih),\ \ \forall\ i=0,\ 2,\ \ldots,\ N, \\ u_0^n=T_1(t_0+n\tau),\ \ u_N^n=T_2(t_0+n\tau), \\ u_i^{n+1}=\mu u_{i+1}^n+(1-2\mu)u_i^n+\mu u_{i-1}^n,\ \ \forall\ i=1,\ 2,\ \ldots,\ N-1. \end{cases}\right\}\ \forall\ n=0,\ 1,\ \ldots$$

De même schéma explicite, le schéma implicite peut être obtenu facilement :

$$\frac{u_i^{n+1}-u_i^n}{\tau}=a\frac{u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}}{h^2}\ \Rightarrow\ -\mu u_{i-1}^{n+1}+(1+2\mu)u_i^{n+1}-\mu u_{i+1}^{n+1}=u_i^n$$

Remarquons que ce schéma est un système linéaire tri-diagonale symétrique, peut-être s'écrit sous la forme matricielle suivante :

$$\mathbf{M}\mathbf{U}^{n+1}=\mathbf{U}^n\ \Rightarrow\ \mathbf{U}^{n+1}=\mathbf{M}^{-1}\mathbf{U}^n\ \Rightarrow\ \mathbf{U}^{n+1}=\left(\mathbf{M}^{-1}\right)^{n+1}\mathbf{U}^0$$

avec,

$$\mathbf{M}=\begin{bmatrix} 1+2\mu & -\mu & 0 & \cdots & 0 \\ -\mu & 1+2\mu & -\mu & & \\ 0 & -\mu & 1+2\mu & -\mu & \\ & \ddots & \ddots & \ddots & \\ 0 & & -\mu & 1+2\mu & -\mu \\ & & 0 & -\mu & 1+2\mu \end{bmatrix},\quad \mathbf{U}^n=\begin{bmatrix} u_1^n \\ u_2^n \\ u_3^n \\ \vdots \\ u_N^n \\ u_N^n \end{bmatrix} \text{ et } \mathbf{U}^{n+1}=\begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{bmatrix}$$

3.2.2. Schéma de Crank–Nicholson

Ce type de schéma est peut-être formé par la moyenne des deux schémas explicite et implicite, c'est-à-dire :

$$u_i^{n+1}=u_i^n+\frac{1}{2}\mu\left(u_{i+1}^n-2u_i^n+u_{i-1}^n\right)+\frac{1}{2}\mu\left(u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}\right)$$

Ce schéma est d'ordre deux en temps et en espace. Sa fonction d'amplification est :

$$g(\theta)=\frac{1-2\mu\sin^2\frac{1}{2}\theta}{1+2\mu\sin^2\frac{1}{2}\theta} \;\Rightarrow\; |g(\theta)|\le 1$$

Le schéma de Crank-Nicholson est un schéma implicite et stable par conséquent il est convergent d'après le théorème de Lax.

3.2.3. Schéma de Lax - Friedrichs

L'application du schéma explicite de Lax–Friedrichs à cette équation conduit à l'équation aux différences suivante :

$$\frac{u_i^{n+1}-\frac{1}{2}\left(u_{i+1}^n+u_{i-1}^n\right)}{\tau}=a\frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{h^2} \;\Rightarrow\; u_i^{n+1}=\left(\frac{1}{2}+\mu\right)\left(u_{i+1}^n+u_{i-1}^n\right)-2\mu u_i^n$$

Suivant les conditions initiales et aux limites, ce schéma permet de calculer facilement u_i^{n+1} pour $i=1,\ 2,\ \ldots,\ N-1$ et $n=0,\ 1,\ \ldots$ soit par conséquent l'algorithme :

$$\left.\begin{cases} u_i^0=\varphi(x_0+ih),\ \forall\ i=0,\ 2,\ \ldots,\ N, \\ u_0^n=T_1(t_0+n\tau),\ \ u_N^n=T_2(t_0+n\tau), \\ u_i^{n+1}=\left(\frac{1}{2}+\mu\right)\left(u_{i+1}^n+u_{i-1}^n\right)-2\mu u_i^n,\ \forall\ i=1,\ 2,\ \ldots,\ N-1. \end{cases}\right\}\forall\ n=0,\ 1,\ \ldots$$

4. EDPs elliptiques. Equation de Poisson et de Laplace

Une interprétation physique des équations elliptiques provient de la notion de flux conservatif donné par un gradient. Cette notion fournit un modèle mathématique pour des lois de conservation à l'équilibre en comportement linéaire. Cela peut être appliqué à de nombreux domaines des sciences pour l'ingénieur.

L'équation différentielle de type elliptique ou équation de Poisson est donnée par :

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=f(x,\ y)$$

x et y sont respectivement l'abscisse et l'ordonnée. L'opérateur $\Delta = \partial^2/\partial x^2 + \partial^2/\partial y^2$ s'appelle le laplacien, dans le cas où $\Delta u = 0$ on a l'équation de Laplace.

Le schéma aux différences finies de l'équation de Poisson pour un point $(x_i,\ y_j)$

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} = f_{i,j}$$

c'est-à-dire :

$$u_{i,j} = \frac{1}{2\left(1 + \frac{h^2}{k^2}\right)}\left[u_{i+1,j} + u_{i-1,j} + \frac{h^2}{k^2}u_{i,j+1} + \frac{h^2}{k^2}u_{i,j-1} - h^2 f_{i,j}\right]$$

L'équation aux différences est un schéma à cinq points d'ordre deux (Fig. 7.13).

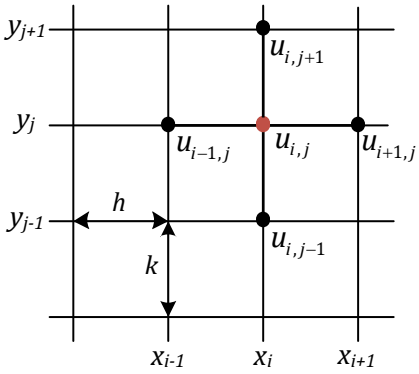


Fig. 7.13 : Schéma de cinq points de l'opérateur de Laplace.

Dans le cas ou le domaine de la solution est de type circulaire, l'équation différentielle devient en coordonnées polaire $(r,\ \theta)$ comme suit :

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} = f(r,\ \theta)$$

Avec $0 \le r \le R$ et $0 \le \theta \le 2\pi$, par conséquent le schéma aux différences finies pour $r_i = r_0 + i\rho$ et $\theta_j = \theta_0 + j\vartheta$ est :

$$\frac{1}{r_i}\left(r_{i+1/2}\frac{u_{i+1,j} - u_{i,j}}{\rho} - r_{i-1/2}\frac{u_{i,j} - u_{i-1,j}}{\rho}\right)\frac{1}{\rho} + \frac{1}{r_i^2}\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\vartheta^2} = f_{i,j}$$

Le schéma aux différences finies fournie un systèmes d'équations linéaires **Au** = **B** dont la matrice **A** est symétrique. Pour résoudre ce système deux types deux méthodes itératives sont souvent utilisées, la méthode de Gauss–Seidel et la méthode de surrelaxation successive. L'algorithme de calcul des cinq points basé sur la méthode itérative de Gauss–Seidel se fait par un script MATLAB comme suit :

```
A=(h/k)^2;
B=0.5/(1+A);
for N=1:NombreMaxIteration
 for i=1:N
   for j=1:M
    u(i,j)=B*(u(i+1,j)+u(i-1,j)+A*u(i,j+1)+A*u(i,j-1)-f(i,j)*h^2);
   end
 end
end
```

Pour comprendre la mise en œuvre pratique de cette méthode, soit le problème elliptique :

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

avec comme conditions aux limites de type de Dirichlet :

$$u(x,\ 0)=2x,\ \ u(x,\ 1)=2\sin\frac{1}{2}\pi x\ \ \forall\ 0\le x\le 1$$

$$u(0,\ y)=0,\ \ u(1,\ y)=2\ \ \forall\ 0\le y\le 1$$

Considérant h = k =1/3 comme pas de discrétisation, par conséquent les points qu'il faut déterminer sont (Fig. 7.14) :

$$U_1=u_{1,1},\ U_2=u_{1,2},\ U_3=u_{2,1}\ \text{ et }\ U_4=u_{2,2}$$

Le schéma aux différences finies de cette équation différentielle est :

$$u_{i,j}=\frac{1}{4}\left[u_{i+1,j}+u_{i-1,j}+u_{i,j+1}+u_{i,j-1}\right]$$

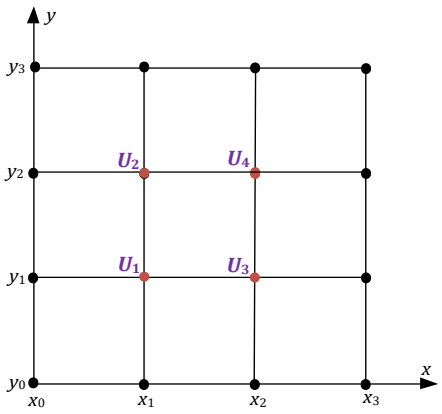


Fig. 7.14 : Domaine de calcul.

Son application permet de donner le système d'équations :

$$\left.\begin{aligned}
u_{1,1}&=\frac{1}{4}\left[u_{2,1}+u_{0,1}+u_{1,2}+u_{1,0}\right]\\
u_{1,2}&=\frac{1}{4}\left[u_{2,2}+u_{0,2}+u_{1,3}+u_{1,1}\right]\\
u_{2,1}&=\frac{1}{4}\left[u_{3,1}+u_{1,1}+u_{2,2}+u_{2,0}\right]\\
u_{2,2}&=\frac{1}{4}\left[u_{3,2}+u_{1,2}+u_{2,3}+u_{2,1}\right]
\end{aligned}\right\}\Leftrightarrow\left\{\begin{aligned}
U_1&=\frac{1}{4}\left[U_3+u_{0,1}+U_2+u_{1,0}\right]\\
U_2&=\frac{1}{4}\left[U_4+u_{0,2}+u_{1,3}+U_1\right]\\
U_3&=\frac{1}{4}\left[u_{3,1}+U_1+U_4+u_{2,0}\right]\\
U_4&=\frac{1}{4}\left[u_{3,2}+U_2+u_{2,3}+U_3\right]
\end{aligned}\right.\ \text{ou}\ \left\{\begin{aligned}
4U_1-U_2-U_3&=u_{0,1}+u_{1,0}\\
-U_1+4U_2-U_4&=u_{0,2}+u_{1,3}\\
-U_1+4U_3-U_4&=u_{3,1}+u_{2,0}\\
-U_2-U_3+4U_4&=u_{3,2}+u_{2,3}
\end{aligned}\right.$$

Suivant les conditions aux limites,

$$u(x_i,\ 0)\equiv u_{i,0}=2x_i,\ \ u(x_i,\ 1)\equiv u_{i,3}=2\sin\frac{1}{2}\pi x_i$$

$$u(0,\ y_j)\equiv u_{0,j}=0,\ \ u(1,\ y_j)\equiv u_{3,j}=2$$

Par conséquent,

$$u_{0,1}=0,\ \ u_{1,0}=2/3,\ \ u_{0,2}=0,\ \ u_{1,3}=1,\ \ u_{3,1}=2,\ \ u_{2,0}=4/3,\ \ u_{3,2}=2\ \text{ et }\ u_{2,3}=1.73$$

Le système d'équations obtenu, s'écrit sous forme matricielle :

$$\begin{pmatrix}4&-1&-1&0\\-1&4&0&-1\\-1&0&4&-1\\0&-1&-1&4\end{pmatrix}\begin{pmatrix}U_1\\U_2\\U_3\\U_4\end{pmatrix}=\begin{pmatrix}u_{0,1}+u_{1,0}\\u_{0,2}+u_{1,3}\\u_{3,1}+u_{2,0}\\u_{3,2}+u_{2,3}\end{pmatrix}$$

Ce système permet de donner facilement les valeurs de U_1, U_2, U_3 et U_4 correspondant respectivement à $u_{1,1}$, $u_{1,2}$, $u_{2,1}$ et $u_{2,2}$, c'est-à-dire :

$$u_{1,1}=0.7111,\ \ u_{1,2}=0.7971,\ \ u_{2,1}=1.3804\ \text{ et }\ u_{2,2}=1.4774$$

5. Application de la méthode des différences finies

5.1. Equation de Burgers

L'équation de Burgers est une équation aux dérivées partielles quasi-linéaire parabolique issue de la mécanique des fluides. Elle apparaît dans divers domaines, comme la modélisation

de la dynamique des gaz, de l'acoustique ou du trafic routier. L'équation de Burgers est exprimée par :

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}=\nu\frac{\partial^2 u}{\partial x^2}$$

u la vitesse et ν la viscosité cinématique. Les conditions :

$$u(x,\ 0)=f(x),\ 0<x<1$$
$$\begin{cases}u(0,\ t)=g_1(t)\\ u(1,\ t)=g_2(t)\end{cases},\ 0<x\le T$$

Le schéma explicite aux différences est exprimé par :

$$\frac{u_i^{n+1}-u_i^n}{\tau}+u_i^n\frac{u_{i+1}^n-u_{i-1}^n}{2h}=\nu\frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{h^2}$$

c'est-à-dire,

$$u_i^{n+1}=(1-2\lambda\nu)u_i^n+\lambda\left(\nu-\frac{h}{2}u_i^n\right)u_{i+1}^n+\lambda\left(\nu+\frac{h}{2}u_i^n\right)u_{i-1}^n,$$

avec,

$$\lambda=\frac{\tau}{h^2},\ i=2,\ 3,\ \ldots,\ i_{\max}-1 \text{ et } n=1,\ 2,\ \ldots$$

Le schéma aux différences pour les conditions est :

$$u_i^1=f(ih),\ i=2,\ 3,\ \ldots,\ i_{\max}$$
$$\begin{cases}u_1^n=g_1(n\tau)\\ u_{i_{\max}}^n=g_2(n\tau)\end{cases},\ n=1,\ 2,\ \ldots$$

Le schéma aux différence est stable pour $\lambda\nu\le 0.5$.

Le schéma implicite aux différences finies de l'équation différentielle est exprimé par :

$$\frac{u_i^{n+1}-u_i^n}{\tau}+u_i^n\frac{u_{i+1}^{n+1}-u_{i-1}^{n+1}}{2h}=\nu\frac{u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}}{h^2}$$

c'est-à-dire,

$$\left(\frac{\tau}{2h}u_i^n-\lambda\nu\right)u_{i+1}^{n+1}+(1+2\lambda\nu)u_i^{n+1}-\left(\frac{\tau}{2h}u_i^n+\lambda\nu\right)u_{i-1}^{n+1}=u_i^n$$

Avec les mêmes conditions initiale et aux limites, l'avantage du schéma implicite est stable et pour n ou, $n=1,\ 2,\ \ldots$, le système d'équations est à résoudre pour déterminer les inconnues u_i^{n+1} pour $i=2,\ 3,\ \ldots,\ i_{\max}-1$

5.2. Equation de réaction-diffusion

L'équation aux différentielle aux dérivées partielles de réaction-diffusion est dite aussi équation de Zeldovich est une équation non linéaire est donnée par :

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}+u(1-u)(u-\vartheta),\ \vartheta \text{ est un paramètre}$$

avec les conditions initiale et aux limites

$$u(x,\ 0)=f(x),\ 0<x<10$$
$$u(0,\ t)=c_1,\ u(10,\ t)=c_2,\ 0<x\le T$$

Le schéma explicite de l'équation différentielle est explicité par :

$$\frac{u_i^{n+1}-u_i^n}{\tau}=\frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{h^2}+u_i^n\left(1-u_i^n\right)\left(u_i^n-\vartheta\right)$$

soit,

$$u_i^{n+1}=(1-2\lambda)u_i^n+\lambda\left(u_{i+1}^n+u_{i-1}^n\right)+\tau u_i^n\left(1-u_i^n\right)\left(u_i^n-\vartheta\right),\ i=2,\ 3,\ \ldots,\ i_{\max}-1,\ n=1,\ 2,\ \ldots$$

Le schéma implicite stable aux différences finies de l'équation différentielle est obtenu par le schéma de Crank-Nicolson :

$$\frac{u_i^{n+1}-u_i^n}{\tau}=\frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{2h^2}+\frac{u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}}{2h^2}+u_i^n\left(1-u_i^n\right)\left(u_i^n-\vartheta\right)$$

5.3. Équation des milieux poreux

Soit l'équation des milieux poreux :

$$\frac{\partial u}{\partial t}=\left(\frac{\partial u}{\partial x}\right)^2+u\frac{\partial^2 u}{\partial x^2}$$

avec les conditions initiale et aux limites :

$$u(x,\ 0)=x,\ 0<x<1$$
$$u(0,\ t)=t,\ u(1,\ t)=1+t,\ 0<x\le T$$

Cette équation est rencontrée dans les problèmes non linéaires de transfert de chaleur et de masse, de la théorie de la combustion et d'écoulement en milieu poreux. Il a également des applications à de nombreux systèmes physiques, y compris la dynamique des fluides des couches minces.

Le schéma explicite de l'équation différentielle est explicité par :

$$\frac{u_i^{n+1}-u_i^n}{\tau}=\left(\frac{u_{i+1}^n-u_{i-1}^n}{2h}\right)^2+u_i^n\frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{h^2}$$

c'est-à-dire :

$$u_i^{n+1}=u_i^n+\frac{\lambda}{4}\left(u_{i+1}^n-u_{i-1}^n\right)^2+\lambda u_i^n\left(u_{i+1}^n-2u_i^n+u_{i-1}^n\right)\ i=2,\ 3,\ \ldots,\ i_{\max}-1,\ n=1,\ 2,\ \ldots$$

5.4. Ecoulement non permanent à surface libre

La méthode des différences finies comme technique numérique de recherche d'une solution approchée aux équations différentielles aux dérivées partielles est très utilisée dans le domaine de l'engineering. Dans le domaine de l'hydraulique par exemple, le problème des écoulements non permanent à surface libre, les écoulements dans les milieux poreux, la propagation de la pollution dans l'eau etc…, sont parmi les problèmes ou la méthode des différences finies se mis en jeu afin de déterminer une solution ou un ordre de grandeur de la solution de ces types de problèmes…

Le modèle mathématique d'un écoulement non permanent unidimensionnelle à surface libre dans un canal prismatique est explicité par les équations de Saint–Venant est un

problème hyperbolique exprimé par le système d'équations différentielles aux dérivées partielles :

$$\begin{cases} \dfrac{\partial S}{\partial t}+\dfrac{\partial Q}{\partial x}=0, \\ \dfrac{1}{S}\dfrac{\partial Q}{\partial t}+\dfrac{1}{S}\dfrac{\partial}{\partial x}\left(\beta\dfrac{Q^2}{S}\right)+g\dfrac{\partial y}{\partial x}-g(I-J)=0. \end{cases}$$

Les inconnus sont le débit $Q=Q(x,\ t)$ et la hauteur $y=y(x,\ t)$ de la surface libre. La section transversale S de l'écoulement est exprimée en fonction de la hauteur y et la perte de charge J est exprimée en fonction de Q, S et y suivant la formule de Manning.

Plusieurs schémas explicites et implicites ont été utilisés pour chercher une solution approchée suivant les conditions initiale et aux limites (Fig. 7.15).

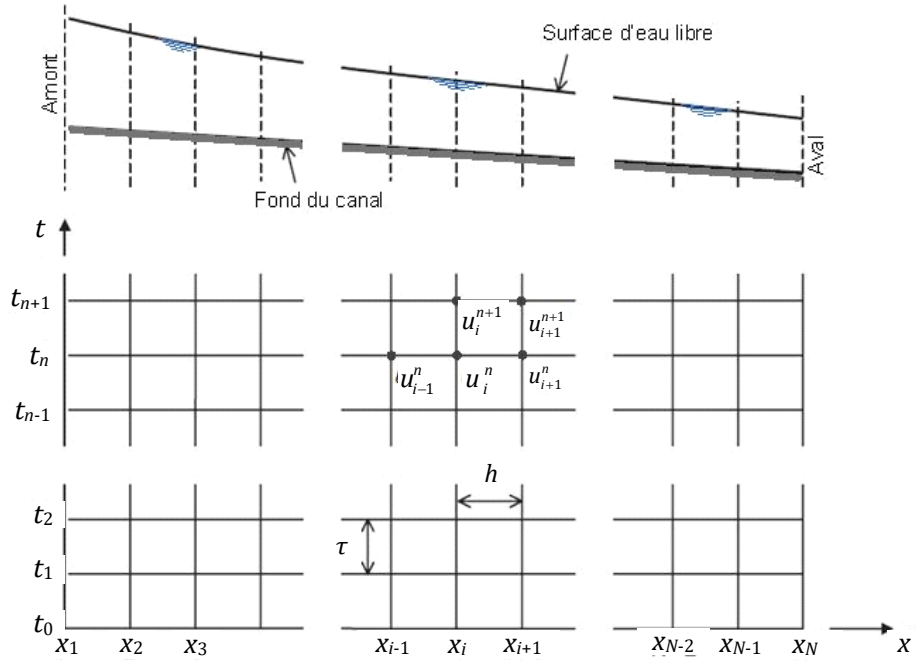


Fig. 7.15 : Grille de calcul pour la résolution numérique des équations de Saint-Venant.

5.4.1. Schéma explicite de Lax-Friedrichs

L'application du schéma explicite de Lax–Friedrichs à ce système conduit à :

$$S_i^{n+1}=\frac{1}{2}\left(S_{i+1}^n+S_{i-1}^n\right)-\frac{\tau}{2h}\left(Q_{i+1}^n-Q_{i-1}^n\right)$$

$$Q_i^{n+1}=\frac{1}{2}\left(Q_{i+1}^n+Q_{i-1}^n\right)-\frac{\tau}{2h}\beta\left(\frac{\left(Q_{i+1}^n\right)^2}{S_{i+1}^n}-\frac{\left(Q_{i-1}^n\right)^2}{S_{i-1}^n}\right)-g\frac{\tau}{2}\left(\left(S_{i+1}^n+S_{i-1}^n\right)\left(\frac{y_{i+1}^n-y_{i-1}^n}{2h}-I\right)+\left(S_{i+1}^nJ_{i+1}^n+S_{i-1}^nJ_{i-1}^n\right)\right)$$

Le schéma obtenu est consistant si $h^2/\tau\to 0$ quand h et $\tau\to 0$.

5.4.2. Schéma explicite de Leapfrog

L'application du schéma explicite de Leapfrog à ce système conduit à :

$$S_i^{n+1}=S_i^{n-1}-\frac{\tau}{h}\left(Q_{i+1}^n-Q_{i-1}^n\right)$$

$$Q_i^{n+1} = Q_i^n - \frac{\tau}{h}\beta\left(\frac{\left(Q_{i+1}^n\right)^2}{S_{i+1}^n} - \frac{\left(Q_{i-1}^n\right)^2}{S_{i-1}^n}\right) - g\tau\left(\frac{\left(y_{i+1}^n - y_{i-1}^n\right)}{h} + 2\left(I - \frac{\left|Q_i^n\right|}{\left(K_i^n\right)^2}\frac{\left(Q_{i+1}^n + Q_{i-1}^n\right)}{2}\right)\right)S_i^n$$

K, est la débitance du canal, elle dépend de y, et elle est exprimée en fonction de la section S du périmètre mouillé p_m, le coefficient de rugosité de Manning n et le facteur $k_n = 1.0\ \mathrm{m}^{1/3}\mathrm{s}^{-1}$. Son expression analytique est : $K(y) = k_n S^{5/3}/np_m^{2/3}$.

5.4.3. Schéma explicite de Lax-Wendroff

Ce schéma est composé de deux étapes (Fig. 7.16) : *Prédiction* et *Correction*.

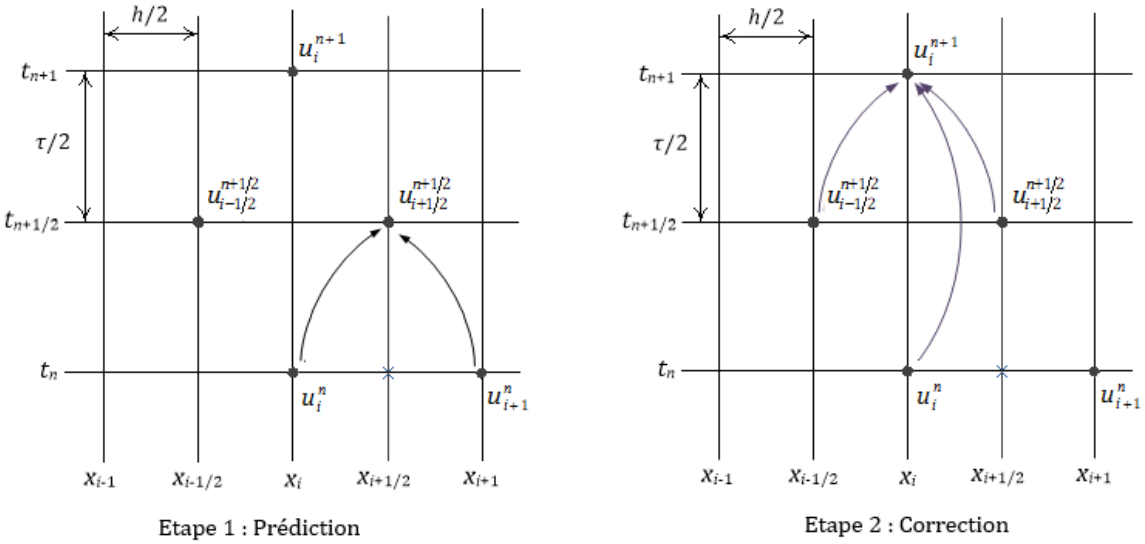


Fig. 7.16 : Schéma de Lax–Wendroff.

Dans la l'étape *Prédiction* :

$$\tilde{S}_{i+1/2}^{n+1/2} = \frac{1}{2}\left(S_{i+1}^n + S_i^n\right) - \frac{\tau}{2h}\left(Q_{i+1}^n - Q_i^n\right)$$

$$\tilde{Q}_{i+1/2}^{n+1/2} = \frac{1}{2}\left(Q_{i+1}^n + Q_i^n\right) - \frac{\tau}{2h}\beta\left(\frac{\left(Q_{i+1}^n\right)^2}{S_{i+1}^n} - \frac{\left(Q_i^n\right)^2}{S_i^n}\right) - g\frac{\tau}{2}\left(\left(S_{i+1}^n + S_i^n\right)\left(\frac{y_{i+1}^n - y_i^n}{2h} - I\right) + \left(S_{i+1}^n J_{i+1}^n + S_i^n J_i^n\right)\right)$$

Dans l'étape *Correction*, le schéma Leapfrog est appliqué pour déterminer S_i^{n+1} et Q_i^{n+1} :

$$S_i^{n+1} = S_i^n - \frac{\tau}{h}\left(\tilde{Q}_{i+1/2}^{n+1/2} - \tilde{Q}_{i-1/2}^{n+1/2}\right)$$

$$Q_i^{n+1} = Q_i^n - \frac{\tau}{h}\left(\frac{\left(\tilde{Q}_{i+1/2}^{n+1/2}\right)^2}{\tilde{S}_{i+1/2}^{n+1/2}} - \frac{\left(\tilde{Q}_{i-1/2}^{n+1/2}\right)^2}{\tilde{S}_{i-1/2}^{n+1/2}}\right) - g\tau S_i^k\left(\frac{\left(y_{i+1/2}^{n+1/2} - y_{i-1/2}^{n+1/2}\right)}{h} - 2\left(I - \frac{\left|Q_i^n\right|}{\left(K_i^n\right)^2}\frac{\left(\tilde{Q}_{i+1/2}^{n+1/2} + \tilde{Q}_{i-1/2}^{n+1/2}\right)}{2}\right)\right)$$

5.4.4. Schéma explicite de MacCormack

Le schéma explicite de MacCormack (Anderson *et al.*, 1984) est un cas particulier du schéma de Lax–Wendroff, il est très utilisé dans la résolution numérique de ce type de problème d'écoulement non permanent à surface libre (Fennema et Chaudhry, 1986). Il permet de déterminer la solution approchée suivant deux étapes, étape de prédiction et étape de correction. Pour appliquer ce schéma, le système d'équations de Barré de Saint–Venant est mis sous la forme vectorielle suivante :

$$\frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{F}}{\partial x}=\mathbf{A}$$

avec,

$$\mathbf{U}=\begin{pmatrix} S \\ Q \end{pmatrix},\quad \mathbf{F}=\begin{pmatrix} Q \\ \beta\dfrac{Q^2}{S}+gSy \end{pmatrix} \text{ et } \quad \mathbf{A}=\begin{pmatrix} 0 \\ gS(I-J) \end{pmatrix}$$

L'étape de prédiction consiste à former à partir des schémas :

$$\frac{\partial \mathbf{U}}{\partial t}\approx\frac{\mathbf{U}_i^p-\mathbf{U}_i^n}{\tau} \text{ et } \frac{\partial \mathbf{F}}{\partial x}\approx\frac{\mathbf{F}_i^n-\mathbf{F}_{i-1}^n}{h}$$

la quantité,

$$\mathbf{U}_i^p=\mathbf{U}_i^n-\frac{\tau}{h}\left(\mathbf{F}_i^n-\mathbf{F}_{i-1}^n\right)+\tau\mathbf{A}_i^n$$

Calculer $\mathbf{U}_i^p$ signifié que la section S_i^p et le débit Q_i^p alors $\mathbf{F}_i^p$ et $\mathbf{A}_i^p$ sont déduite en fonction de S_i^p et Q_i^p comme suit :

$$\mathbf{F}_i^p=\begin{pmatrix} Q_i^p \\ \beta\dfrac{\left(Q^2\right)_i^p}{S_i^p}+gS_i^p y_i^p \end{pmatrix} \text{ et } \quad \mathbf{A}_i^p=\begin{pmatrix} 0 \\ gS_i^p\left(I-\dfrac{\left|Q_i^p\right|Q_i^p}{\left(K^2\right)_i^p}\right) \end{pmatrix}$$

L'étape de correction consiste à former à partir des schémas :

$$\frac{\partial \mathbf{U}}{\partial t}\approx\frac{\mathbf{U}_i^c-\mathbf{U}_i^n}{\tau} \text{ et } \frac{\partial \mathbf{F}}{\partial x}\approx\frac{\mathbf{F}_i^p-\mathbf{F}_{i-1}^p}{h}$$

La quantité $\mathbf{U}_i^c$ obtenue aussi par substitution dans la forme vectorielle, c'est-à-dire :

$$\mathbf{U}_i^c=\mathbf{U}_i^n-\frac{\tau}{h}\left(\mathbf{F}_i^p-\mathbf{F}_{i-1}^p\right)+\tau\mathbf{A}_i^p$$

Finalement la solution cherchée est la valeur moyenne de la prédiction et la correction (Chaudhry, 2008),

$$\mathbf{U}_i^{n+1}=\frac{1}{2}\left(\mathbf{U}_i^p+\mathbf{U}_i^c\right)$$

Généralement l'étape de prédiction est calculée pour $p=n-1$ (arrière) et la partie correction pour $c=n+1$ (avant) (Chaudhry, 2008).

Ces schémas aux différences sont stables si la condition de Koren est vérifiée, celle-ci qui découle de la condition de Courant-Friedrichs-Lewy (CFL), c'est-à-dire :

$$\tau\le\frac{\sqrt{1+2F_{r0}}-1}{F_{r0}\dfrac{gS_0}{v_0}}$$

Où, F_{r0} est le nombre de Froude de l'écoulement uniforme à l'état initial.

5.4.5. Schéma implicite

Le schéma implicite le plus utilisé est de Preissman(1961), les termes du système d'équations de Saint–Venant, sont explicités en différence finie comme suit :

$$\frac{\partial S}{\partial t} \approx \frac{1}{\tau}\left(\frac{S_i^{n+1}+S_{i+1}^{n+1}}{2}\right)-\frac{1}{\tau}\left(\frac{S_i^{n}+S_{i+1}^{n}}{2}\right)=\frac{\overline{S}_i^{n+1}-\overline{S}_i^{n}}{\tau}$$

$$\frac{\partial Q}{\partial x} \approx \theta\frac{Q_{i+1}^{n+1}-Q_i^{n+1}}{h}+(1-\theta)\frac{Q_{i+1}^{n}-Q_i^{n}}{h}$$

$$\frac{\partial Q}{\partial t} \approx \frac{1}{\tau}\left(\frac{Q_i^{n+1}+Q_{i+1}^{n+1}}{2}\right)-\frac{1}{\tau}\left(\frac{Q_i^{n}+Q_{i+1}^{n}}{2}\right)=\frac{\overline{Q}_i^{n+1}-\overline{Q}_i^{n}}{\tau}$$

$$\frac{\partial}{\partial x}\left(\beta\frac{Q^2}{S}\right) \approx \theta\frac{\left(\beta Q^2/S\right)_{i+1}^{n+1}-\left(\beta Q^2/S\right)_i^{n+1}}{h}+(1-\theta)\frac{\left(\beta Q^2/S\right)_{i+1}^{n}-\left(\beta Q^2/S\right)_i^{n}}{h}$$

$$S\frac{\partial h}{\partial x} \approx \theta\overline{S}_i^{n+1}\,\frac{h_{i+1}^{n+1}-h_i^{n+1}}{h}+(1-\theta)\overline{S}_i^{n}\,\frac{h_{i+1}^{n}-h_i^{n}}{h}$$

$$JS \approx \theta\overline{J}_i^{n+1}\overline{S}_i^{n+1}+(1-\theta)\overline{J}_i^{n}\overline{S}_i^{n}$$

D'après Fread (1973 et 1974), le coefficient de pondération θ est compris entre 0.55 et 0.6. La substitution dans le système d'équations, permet d'obtenir le système aux schémas de différences :

$$\begin{cases}
\dfrac{h}{\tau}\left(\overline{S}_i^{n+1}-\overline{S}_{i+1}^{n}\right)+\theta\left(Q_{i+1}^{n+1}-Q_i^{n+1}\right)+(1-\theta)\left(Q_{i+1}^{n}-Q_i^{n}\right)=0, \\
\dfrac{h}{\tau}\left(\overline{Q}_i^{n+1}-\overline{Q}_i^{n}\right)+\theta\left[\left(\beta\dfrac{Q^2}{S}\right)_{i+1}^{n+1}-\left(\beta\dfrac{Q^2}{S}\right)_i^{n+1}\right]+(1-\theta)\left[\left(\beta\dfrac{Q^2}{S}\right)_{i+1}^{n}-\left(\beta\dfrac{Q^2}{S}\right)_i^{n}\right]+ \\
\qquad \theta g\overline{S}_i^{n+1}\left(H_{i+1}^{n+1}-H_i^{n+1}\right)+(1-\theta)g\overline{S}_i^{n}\left(H_{i+1}^{n}-H_i^{n}\right)+\theta h\overline{J}_i^{n+1}\overline{S}_i^{n+1}+(1-\theta)h\overline{J}_i^{n}\overline{S}_i^{n}=0.
\end{cases}$$

Les termes inconnues dans ce système sont ceux qui ont l'indice supérieure $n+1$, c'est-à-dire Q^{n+1}, S^{n+1}, H^{n+1} et J^{n+1}, or S^{n+1} pouvant être exprimés en fonction de H^{n+1} et J^{n+1} s'exprime en fonction de Q^{n+1} et S^{n+1} suivant la formule de Manning. Par conséquent J^{n+1} peut être exprimé en fonction de Q^{n+1} et H^{n+1}. Alors, le système d'équations possède les inconnues Q^{n+1} et H^{n+1}, en d'autres termes les vrais inconnues sont Q_i^{n+1}, Q_{i+1}^{n+1} et H_i^{n+1}, H_{i+1}^{n+1}.

Pour $i=1,\ 2,\ 3\ ,...,\ N-1$, les équations du système contiennent chacune quatre inconnues, qui peuvent être symbolisées par $C_i\left(Q_i^{n+1},\ H_i^{n+1},\ Q_{i+1}^{n+1},\ H_{i+1}^{n+1}\right)=0$ pour l'équation de continuité et par $M_i\left(Q_i^{n+1},\ H_i^{n+1},\ Q_{i+1}^{n+1},\ H_{i+1}^{n+1}\right)=0$ pour l'équation de la conservation de la quantité de mouvement $M_i\left(Q_i^{n+1},\ H_i^{n+1},\ Q_{i+1}^{n+1},\ H_{i+1}^{n+1}\right)=0$.

Dans le domaine discrétisé (Fig. 7.16), les deux équations du système (continuité et conservation de la quantité de mouvement) sont considérées pour $N-1$ grilles rectangulaires chacune, la limite supérieure (amont) correspond à $i=1$, et la limite inférieure (aval) à $i=N$ ce qui conduit à résoudre $2N-2$ équations.

Une variété de conditions aux limites (amont et aval) peut être prise en considération. Si le débit à l'entrée est constant, l'équation des conditions aux limites supérieures est de type de Dirichlet :

$$Q_1^{n+1} = Q_0$$

Si l'écoulement dans la limite aval est critique, alors :

$$\frac{Q_N^{n+1}\sqrt{L_N^{n+1}}}{\sqrt{g\left(S_N^{n+1}\right)^3}} = 1$$

L_N^{n+1} dans le nombre de Froude, est la largeur équivalent dans la limite inférieur du canal. Pour un long canal, l'écoulement à la limite inférieure est considéré normal, et la condition à la limite aval peut être exprimée par l'équation :

$$J_N^{n+1} = I$$

Les équations aux conditions aux limites (amont et aval) peuvent être symbolisées pour la limite supérieure (amont) par $B_1\left(Q_1^{n+1},\ H_1^{n+1}\right)=0$ et pour la limite inférieure (aval) par $B_N\left(Q_N^{n+1},\ H_N^{n+1}\right)=0$. Alors, la détermination de Q_i^{n+1}, Q_{i+1}^{n+1}, H_i^{n+1} et H_{i+1}^{n+1} pour $i=1,\ 2,\ 3\ ,...,\ N$ se fait par la résolution du système d'équations non linéaires suivant :

$$\left.\begin{cases} B_1\left(Q_1^{n+1},\ H_1^{n+1}\right)=0, \\ C_i\left(Q_i^{n+1},\ H_i^{n+1},\ Q_{i+1}^{n+1},\ H_{i+1}^{n+1}\right)=0, \\ M_i\left(Q_i^{n+1},\ H_i^{n+1},\ Q_{i+1}^{n+1},\ H_{i+1}^{n+1}\right)=0, \\ B_N\left(Q_N^{n+1},\ H_N^{n+1}\right)=0. \end{cases}\right\} \forall\ i=1,\ 2,\ ...,\ N-1$$

Ce système d'équations peut être résolu par la méthode de Newton–Raphson afin de déterminer les Q_i^{n+1}, H_i^{n+1}, Q_{i+1}^{n+1} et H_{i+1}^{n+1} pour $i=1,\ 2,\ ...,\ N-1$.

6. Exercices résolus

Exercice 1

En utilisant le développement de Taylor trouver :

$$\left(\frac{\partial u}{\partial x}\right)_i^n = \frac{1}{6h}\left(-11u_i^n + 18u_{i+1}^n - 9u_{i+2}^n + 2u_{i+3}^n\right) + O\left(h^3\right)$$

Pour simplifier les formules, soit la notation compacte de la dérivée partielle :

$$\left(\frac{\partial u}{\partial x}\right)_i^n = \left(u_x\right)_i^n,\ \left(\frac{\partial^2 u}{\partial x^2}\right)_i^n = \left(u_{xx}\right)_i^n \text{ et } \left(\frac{\partial^3 u}{\partial x^3}\right)_i^n = \left(u_{xxx}\right)_i^n$$

considérant,

$$\left(u_x\right)_i^n = \frac{1}{6h}\left(au_i^n + bu_{i+1}^n + cu_{i+2}^n + du_{i+3}^n\right) + O\left(h^3\right)$$

et cherchant les facteurs *a*, *b*, *c* et *d*. soit les développements de Taylor d'ordre 3 de u_{i+1}^n, u_{i+2}^n et u_{i+3}^n :

$$u_{i+1}^n = u_i^n + h\left(u_x\right)_i^n + \frac{h^2}{2}\left(u_{xx}\right)_i^n + \frac{h^3}{6}\left(u_{xxx}\right)_i^n + O\left(h^3\right)$$

$$u_{i+2}^n = u_i^n + 2h(u_x)_i^n + \frac{(2h)^2}{2}(u_{xx})_i^n + \frac{(2h)^3}{6}(u_{xxx})_i^n + O\left((2h)^3\right) =$$
$$u_i^n + 2h(u_x)_i^n + 4\frac{h^2}{2}(u_{xx})_i^n + 8\frac{h^3}{6}(u_{xxx})_i^n + O\left(h^3\right)$$

$$u_{i+3}^n = u_i^n + 3h(u_x)_i^n + \frac{(3h)^2}{2}(u_{xx})_i^n + \frac{(3h)^3}{6}(u_{xxx})_i^n + O\left((3h)^3\right) =$$
$$u_i^n + 3h(u_x)_i^n + 9\frac{h^2}{2}(u_{xx})_i^n + 27\frac{h^3}{6}(u_{xxx})_i^n + O\left(h^3\right)$$

alors,

$$au_i^n + bu_{i+1}^n + cu_{i+2}^n + du_{i+3}^n = (a+b+c+d)u_i^n + (b+2c+3d)h(u_x)_i^n + (b+4c+9d)\frac{h^2}{2}(u_{xx})_i^n$$
$$+ (b+8c+27d)\frac{h^3}{6}(u_{xxx})_i^n + O\left(h^3\right) = 6h(u_x)_i^n$$

L'identification entre les deux termes droits et à gauche conduit au système d'équations :

$$\begin{cases} a+b+c+d=0, \\ b+2c+3d=6, \\ b+4c+9d=0, \\ b+8c+27d=0. \end{cases} \Rightarrow \begin{cases} a=-11 \\ b=18 \\ c=-9 \\ d=2 \end{cases}$$

finalement,

$$(u_x)_i^n = \frac{1}{6h}\left(-11u_i^n + 18u_{i+1}^n - 9u_{i+2}^n + 2u_{i+3}^n\right) + O\left(h^3\right)$$

ce qu'il faut trouver comme résultat.

Exercice 2

Déterminer la famille des schémas de second ordre de la dérivée partielle $(\partial u/\partial x)_i^n$ sous la forme :

$$\left(\frac{\partial u}{\partial x}\right)_i^n = \frac{1}{h}\left(au_{i-2}^n + bu_{i-1}^n + cu_i^n + du_{i+1}^n\right) + O\left(h^2\right)$$

Soit les développements de u_{i+1}^n, u_{i-1}^n et u_{i-2}^n d'ordre deux en série de Taylor :

$$u_{i+1}^n = u_i^n + h(u_x)_i^n + \frac{h^2}{2}(u_{xx})_i^n + O\left(h^2\right), \quad u_{i-1}^n = u_i^n - h(u_x)_i^n + \frac{h^2}{2}(u_{xx})_i^n + O\left(h^2\right)$$

$$u_{i-2}^n = u_i^n - 2h(u_x)_i^n + 4\frac{h^2}{2}(u_{xx})_i^n + O\left(h^2\right)$$

alors,

$$au_{i-2}^n + bu_{i-1}^n + cu_i^n + du_{i+1}^n = (a+b+c+d)u_i^n + (-2a-b+d)h(u_x)_i^n + (4a+b+d)\frac{h^2}{2}(u_{xx})_i^n + O\left(h^2\right)$$

ou,

$$(a+b+c+d)u_i^n + (-2a-b+d)h(u_x)_i^n + (4a+b+d)\frac{h^2}{2}(u_{xx})_i^n + O\left(h^2\right) = h(u_x)_i^n$$

par identification membre à membre :

$$\begin{cases} a+b+c+d=0, \\ -2a-b+d=1, \\ 4a+b+d=0. \end{cases} \Rightarrow \begin{cases} a=c/3, \\ b=-(1+2c)/2, \\ d=(3-2c)/6. \end{cases}$$

par conséquent,

$$\left(\frac{\partial u}{\partial x}\right)_i^n = \frac{1}{6h}\left(2cu_{i-2}^n - 3(1+2c)u_{i-1}^n + 6cu_i^n + (3-2c)u_{i+1}^n\right) + O\left(h^2\right)$$

Si $c=0$ on retrouve l'expression du schéma centrale de la dérivée partielle :

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

Exercice 3

Analyser la stabilité du schéma au différences suivant :

$$\frac{u_i^{n+1}-u_i^n}{\tau} + a\frac{u_{i+1}^n - u_{i-1}^n}{2h} = 0$$

Pour illustrer l'analyse de stabilité, en remplaçant u_i^n par $g^n e^{ji\theta}$ alors le schéma aux différences devient :

$$\frac{g^{n+1}e^{ji\theta} - g^n e^{ji\theta}}{\tau} + a\frac{g^n e^{j(i+1)\theta} - g^n e^{j(i-1)\theta}}{2h} = g^n e^{ji\theta}\left(\frac{g-1}{\tau} + a\frac{e^{j\theta}-e^{-j\theta}}{2h}\right) = 0$$

qui donne :

$$g = 1 - j\lambda\sin\theta$$

alors,

$$|g(\theta)|^2 = 1 + \lambda^2\sin^2\theta$$

Puisque $|g(\theta)| > 1,\ \forall\ 0 \le \theta \le \pi$ alors le schéma est instable.

Exercice 4

Soit l'équation différentielle aux dérivées partielle :

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} - u = 0$$

Etudier la stabilité du schéma aux différences, si le schéma appliqué à l'équation est celle de Lax–Friedrichs modifiée.

La méthode de Lax–Friedrichs modifiée est un schéma explicite qui consiste à :

$$\left(\frac{\partial u}{\partial t}\right)_i^n \approx \frac{1}{\tau}\left(u_i^{n+1} - \frac{1}{2}\left(u_{i+1}^n + u_{i-1}^n\right)\right),\quad \left(\frac{\partial u}{\partial x}\right)_i^n \approx \frac{u_{i+1}^n - u_{i-1}^n}{2h} \text{ et } u \approx u_i^n$$

par conséquent l'EDP devient :

$$\frac{1}{\tau}\left(u_i^{n+1} - \frac{1}{2}\left(u_{i+1}^n + u_{i-1}^n\right)\right) + a\frac{u_{i+1}^n - u_{i-1}^n}{2h} - u_i^n = 0$$

remplaçant u_i^n par $g^n e^{ji\theta}$, l'expression précédente devient :

$$\frac{1}{\tau}\left(g^{n+1}e^{ji\theta} - \frac{1}{2}\left(g^n e^{j(i+1)\theta} + g^n e^{j(i-1)\theta}\right)\right) + \frac{a}{2h}\left(g^n e^{j(i+1)\theta} - g^n e^{j(i-1)\theta}\right) - g^n e^{ji\theta} = 0$$

c'est-à-dire,

$$\frac{1}{\tau}\left(g - \frac{1}{2}\left(e^{j\theta} + e^{-j\theta}\right)\right) + \frac{a}{2h}\left(e^{ji\theta} - e^{-ji\theta}\right) - 1 = 0 \Rightarrow g = \cos\theta + \tau - j\lambda\sin\theta$$

par conséquent,

$$|g|^2 = (\cos\theta + \tau)^2 + \lambda^2 \sin^2\theta = 1 + 2\tau\cos\theta + \tau^2 + \lambda^2$$

or,

$$2\tau\cos\theta \le 2\tau$$

alors,

$$|g|^2 = 1 + 2\tau\cos\theta + \tau^2 + \lambda^2 \le 1 + 2\tau + \tau^2 + \lambda^2$$

mais,

$$1 + 2\tau + \tau^2 + \lambda^2 = 1 + 2\tau + (1 + a^2/h^2)\tau^2 \approx 1 + 2\tau + \tau^2 = (1+\tau)^2$$

ce qui conduit à,

$$|g|^2 \le (1+\tau)^2 \quad \Leftrightarrow \quad |g| \le 1 + \tau$$

La fonction g est une fonction qui dépend de θ, τ et h. Pour étudier la stabilité, soit le théorème suivant :

Si la fonction g dépend de θ, τ et h, alors, le schéma aux différences finies est stable s'il existe un constant K indépendant de θ, τ et h sachant que :

$$|g| \le 1 + K\tau$$

Si la fonction g dépend que de θ, la condition de stabilité se résume à la condition :

$$|g(\theta)| \le 1$$

Pour ce schéma $K = 1$, par conséquent le schéma aux différences de l'équation différentielle est stable selon ce théorème.

Exercice 5

Soit le schéma aux différences constitué suivant deux étapes Prédiction–Correction :

Prédiction :

$$\tilde{u}_i^{n+1} = u_i^n - \frac{\lambda}{2}\left(u_{i+1}^n - u_{i-1}^n\right) + \tau f_i^n$$

Correction :

$$u_i^{n+1} = \frac{1}{4}\left(\tilde{u}_{i+1}^{n+1} + 2\tilde{u}_i^{n+1} + \tilde{u}_{i-1}^{n+1}\right)$$

Etudier la stabilité de ce schéma suivant l'analyse de von Neumann.

L'analyse de von Neumann consiste à remplacer u_i^n par $g^n e^{ji\theta}$, par conséquent le schéma prédicteur devient :

$$\tilde{u}_i^{n+1} = g^n e^{ji\theta} - \frac{\lambda}{2}\left(g^n e^{j(i+1)\theta} - g^n e^{j(i-1)\theta}\right) = g^n e^{ji\theta}\left(1 - j\lambda\sin\theta\right)$$

alors,

$$\tilde{u}_{i+1}^{n+1} = g^n e^{j(i+1)\theta}\left(1 - j\lambda\sin\theta\right) \text{ et } \tilde{u}_{i-1}^{n+1} = g^n e^{j(i-1)\theta}\left(1 - j\lambda\sin\theta\right)$$

Le schéma correcteur devient :

$$u_i^{n+1} = \frac{1}{4}\left(\tilde{u}_{i+1}^{n+1} + 2\tilde{u}_i^{n+1} + \tilde{u}_{i-1}^{n+1}\right) \Rightarrow g^{n+1}e^{ji\theta} = \frac{1}{4}\left(g^n e^{j(i+1)\theta} + 2g^n e^{ji\theta} + g^n e^{j(i-1)\theta}\right)\left(1 - j\lambda\sin\theta\right)$$

ou,

$$g=\frac{1}{4}\left(e^{j\theta}+2+e^{-j\theta}\right)\left(1-j\lambda\sin\theta\right)=\frac{1}{2}\left(\cos\theta+1\right)\left(1-j\lambda\sin\theta\right)$$

alors,

$$|g|^2=\frac{1}{4}\left(\cos\theta+1\right)^2\left(1+\lambda^2\sin^2\theta\right)=\cos^4\frac{1}{2}\theta\left(1+\lambda^2\sin^2\theta\right)$$

Pour que ce schéma soit stable, il faut que $|g|^2\le 1$, c'est-à-dire :

$$\cos^4\frac{1}{2}\theta\left(1+\lambda^2\sin^2\theta\right)\le 1 \quad\Leftrightarrow\quad \cos^4\frac{1}{2}\theta\left(1+4\lambda^2\sin^2\frac{1}{2}\theta\cos^2\frac{1}{2}\theta\right)\le 1$$

$$4\lambda^2\sin^2\frac{1}{2}\theta\cos^2\frac{1}{2}\theta\cos^4\frac{1}{2}\theta\le 1-\cos^4\frac{1}{2}\theta=\left(1-\cos^2\frac{1}{2}\theta\right)\left(1+\cos^2\frac{1}{2}\theta\right)=\sin^2\frac{1}{2}\theta\left(1+\cos^2\frac{1}{2}\theta\right)$$

et puisque,

$$\sin^2\frac{1}{2}\theta\approx 1 \quad et \quad \cos^2\frac{1}{2}\theta\approx 1$$

alors ,

$$4\lambda^2\sin^2\frac{1}{2}\theta\cos^2\frac{1}{2}\theta\cos^4\frac{1}{2}\theta\le 1+\cos^2\frac{1}{2}\theta \quad\Rightarrow\quad 4\lambda^2\le 2 \quad\Rightarrow\quad \lambda^2\le\frac{1}{2}$$

Pour que le schéma Prédicteur – Correcteur soit stable, il faut que :

$$\lambda\le\sqrt{\frac{1}{2}}$$

Exercice 6

En utilisant la méthode des différences finies avant pour $h=0.2$ et $\tau=0.02$, déterminer une solution numérique du problème suivant :

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}=\dfrac{\partial u}{\partial t},\ 0<x<1,\ 0<t<0.2,\\ u(0,\ t)=u(1,\ t)=0,\ 0\le t\le 0.2,\\ u(x,\ 0)=4x(1-x),\ 0\le x\le 1.\end{cases}$$

On a les schémas aux différences :

$$\frac{\partial^2 u}{\partial x^2}\simeq\frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{h^2}$$

$$\frac{\partial u}{\partial t}\simeq\frac{u_i^{n+1}-u_i^n}{\tau}$$

On a les schémas aux différences :

$$\frac{\partial^2 u}{\partial x^2}=\frac{\partial u}{\partial t}\ \Rightarrow\ \frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{h^2}=\frac{u_i^{n+1}-u_i^n}{\tau}$$

soit,

$$u_i^{n+1}=\frac{\tau}{h^2}u_{i+1}^n+\left(1-2\frac{\tau}{h^2}\right)u_i^n+\frac{\tau}{h^2}u_{i-1}^n$$

$h=0.2$, le pas de discrétisation de l'intervalle [0, 1], soit $x_i,\ i=1,\ \ldots,\ 6$

$\tau=0.02$, le pas de discrétisation de l'intervalle [0, 0.02], soit $t_n,\ n=1,\ \ldots,\ 11$

Pour les conditions aux limites du problèmes

$$u(0,\ t)=u(1,\ t)=0,\ 0\le t\le 0.2\ \Rightarrow\ u_1^n=u_6^n=0,\ \forall n=1,\ 2\ ,\ldots,\ 11$$

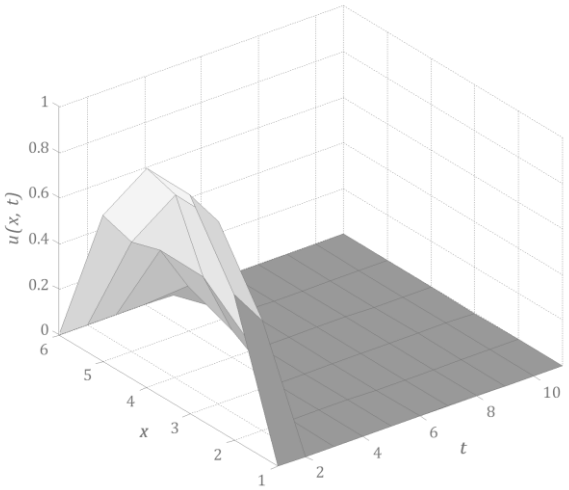
$$u(x,\ 0)=4x(1-x),\ 0\le x\le 1\ \Rightarrow\ u_i^1=4(i-1)\left(1-(i-1)h\right)h,\ \forall i=2,\ 3\ ,4,\ 5$$

Le schéma aux différences finies du problème est :

$$\begin{cases} u_i^{n+1}=\dfrac{\tau}{h^2}u_{i+1}^n+\left(1-2\dfrac{\tau}{h^2}\right)u_i^n+\dfrac{\tau}{h^2}u_{i-1}^n,\ \forall i=2,\ 3,\ 4,\ 5 \text{ et } n=1,\ 3,\ \ldots,\ 10, \\ u_1^n=u_6^n=0,\ \forall n=1,\ 2\ ,\ldots,\ 11, \\ u_i^1=4(i-1)\left(1-(i-1)h\right)h,\ \forall i=2,\ 3\ ,4,\ 5. \qquad (h=0.2,\ \ \tau=0.02) \end{cases}$$

qui peut être programmé dans le script MATLAB suivant :

```
tau=0.02; h=0.2;
t=0:tau:0.2; x=0:h:1;
for n=1:numel(t)
    u(1,n)=0;
    u(6,n)=0;
end
for i=2:numel(x)-1
    u(i,1)=4*(i-1)*(1-(i-1)*h)*h;
end
l=to/(h^2);
for i=2:numel(x)-1
 for n=1:numel(t)-1
  u(i,n+1)=l*u(i+1,n)+(1-2*l)*u(i,n)+l*u(i-1,n);
 end
end
surfc(t,x,u)
```



son exécution conduit au résultats représentés dans la figure ci-dessus.

Exercice 7

Soit l'équation différentielle aux dérivées partielles suivante :

$$\frac{\partial u}{\partial t}+a\frac{\partial^3 u}{\partial x^3}=f$$

- Appliquer le schéma de Lax–Friedrichs pour déterminer l'équation aux différences finies de l'EDP.
- Etudier la consistance et la stabilité du schéma obtenu.

- Le schéma de Lax–Friedrichs concerne un schéma spécifique de la dérivée par rapport au temps c'est-à-dire :

$$\left(\frac{\partial u}{\partial t}\right)_i^n\approx\frac{1}{\tau}\left(u_i^{n+1}-\frac{1}{2}\left(u_{i+1}^n+u_{i-1}^n\right)\right)$$

et un schéma central de la dérivée par rapport à l'espace. L'approximation de la dérivée partielle d'ordre trois par un schéma centrale est explicitée comme suit :

$$\left(\frac{\partial^3 u}{\partial x^3}\right)_i^n \approx \frac{u_{i+2}^n - 2u_{i+1}^n + 2u_{i-1}^n - u_{i-2}^n}{2h^3}$$

remplaçant dans l'EDP :

$$\frac{1}{\tau}\left(u_i^{n+1} - \frac{1}{2}\left(u_{i+1}^n + u_{i-1}^n\right)\right) + a\frac{u_{i+2}^n - 2u_{i+1}^n + 2u_{i-1}^n - u_{i-2}^n}{2h^3} = f_i^n$$

ou,

$$u_i^{n+1} = \frac{1}{2}\left(u_{i+1}^n + u_{i-1}^n\right) - \left(\frac{a\tau}{2h^3}\right)\left(u_{i+2}^n - 2u_{i+1}^n + 2u_{i-1}^n - u_{i-2}^n\right) + \tau f_i^n$$

- Pour étudier la consistance, soit le développement en série de Taylor de u_{i+1}^n et u_{i-1}^n :

$$u_{i\pm1}^n = u_i^n \pm h\left(\frac{\partial u}{\partial x}\right)_i^n + \frac{1}{2}h^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n \pm \frac{1}{6}h^3\left(\frac{\partial^3 u}{\partial x^3}\right)_i^n + O\left(h^4\right)$$

alors,

$$\frac{1}{2}\left(u_{i+1}^n + u_{i-1}^n\right) = u_i^n + \frac{1}{2}h^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n + O\left(h^4\right) \text{ et } \frac{u_{i+1}^n - u_{i-1}^n}{2h} = \left(\frac{\partial u}{\partial x}\right)_i^n + \frac{1}{6}h^2\left(\frac{\partial^3 u}{\partial x^3}\right)_i^n + O\left(h^4\right)$$

aussi,

$$\left(\frac{\partial^3 u}{\partial x^3}\right)_i^n \approx \frac{u_{i+2}^n - u_{i-2}^n}{2h^3} - \frac{2}{h^2}\left(\frac{u_{i+1}^n - u_{i-1}^n}{2h}\right)$$

ou,

$$\left(\frac{\partial^3 u}{\partial x^3}\right)_i^n = \frac{u_{i+2}^n - u_{i-2}^n}{2h^3} - \frac{2}{h^2}\left(\frac{\partial u}{\partial x}\right)_i^n - \frac{1}{3}\left(\frac{\partial^3 u}{\partial x^3}\right)_i^n + O\left(h^4\right)$$

donc,

$$\left(\frac{\partial^3 u}{\partial x^3}\right)_i^n = \frac{u_{i+2}^n - 2u_{i+1}^n + 2u_{i-1}^n - u_{i-2}^n}{2h^3} + O\left(h^4\right)$$

et,

$$\left(\frac{\partial u}{\partial t}\right)_i^n = \frac{1}{\tau}\left(u_i^{n+1} - u_i^n + \frac{1}{2}h^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n + O\left(h^4\right)\right) + O\left(\tau^2\right) = \frac{1}{\tau}\left(u_i^{n+1} - u_i^n + \frac{1}{2}h^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n\right) + O\left(\tau^2,\ \frac{h^4}{\tau}\right)$$

L'erreur de troncature de cette EDP est $O\left(\tau^2,\ h^4/\tau,\ h^4\right)$ qui tend vers 0 quand h et τ tendent vers 0, par conséquent le schéma aux différences est consistant.

Pour étudier la stabilité du schéma, soit $u_i^n = g^n e^{ji\theta}$ en remplaçant dans l'équation aux différences, on obtient :

$$g = \cos\theta - j\left(\frac{a\tau}{h^3}\right)\left(\sin 2\theta - 2\sin\theta\right)$$

alors,

$$|g|^2 = \cos^2\theta + \left(\frac{a\tau}{h^3}\right)^2\left(\sin 2\theta - 2\sin\theta\right)^2$$

or,

$$\sin 2\theta - 2\sin\theta = 2\sin\theta\cos\theta - 2\sin\theta = 2\sin\theta\left(\cos\theta - 1\right)$$

$$2\sin\theta\left(\cos\theta - 1\right) = 2\sin\theta\left(\cos^2\frac{1}{2}\theta - \sin^2\frac{1}{2}\theta - 1\right) = 2\sin\theta\left(\cos^2\frac{1}{2}\theta - \sin^2\frac{1}{2}\theta - \cos^2\frac{1}{2}\theta - \sin^2\frac{1}{2}\theta\right)$$

$$= -4\sin\theta\sin^2\frac{1}{2}\theta$$

alors,

$$|g|^2 = \cos^2\theta + 16\left(\frac{a\tau}{h^3}\right)^2 \sin^2\theta \sin^4\frac{1}{2}\theta^2 \le 1 + \left(\frac{16a^2}{h^3}\right)\tau^2$$

de cette formule, il peut exister un constant *K* de façon à avoir,

$$|g| \le 1 + K\tau$$

Donc le schéma aux différences de l'EDP est stable.

Exercice 8

Soit l'EDP avec les conditions initiales et aux limites :

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0,\ -2 \le x \le 3,\ \forall t \ge 0$$

$$u(x,\ 0) = \varphi(x) = \begin{cases} 1 - |x| & \text{si } |x| \le 1, \\ 0 & \text{si } |x| > 1. \end{cases}$$

$$u(-2,\ t) = 0,\ \forall t \ge 0$$

1. En utilisant le schéma de Lax–Friedrichs pour $\lambda = 0.8$ et $h = 0.1$, déterminer une solution approchée de l'EDP.

2. Tracer la solution approchée pour *t* = 1.6 et la comparez avec la solution exacte $u(x,\ 1.6) = \varphi(x - 1.6)$.

3. Comparez la solution approchée par le même schéma avec la solution exacte $u(x,\ 1.6) = \varphi(x - 1.6)$ si $\lambda = 1.6$.

1. l'application du schéma de Lax–Friedrichs à cette équation conduit à :

$$\frac{1}{\tau}\left(u_i^{n+1} - \frac{1}{2}\left(u_{i+1}^n + u_{i-1}^n\right)\right) + \frac{u_{i+1}^n - u_{i-1}^n}{2h} = 0 \ \Rightarrow\ u_i^{n+1} = \frac{1}{2}(1+\lambda)u_{i-1}^n + \frac{1}{2}(1-\lambda)u_{i+1}^n$$

Or, $\lambda = 0.8$, alors l'équation aux différences du problème est :

$$u_i^{n+1} = 0.9u_{i-1}^n + 0.1u_{i+1}^n$$

Le domaine de travail de ce problème est $x \in \left]-2,\ 3\right]$ et $t \in \left[0,\ +\infty\right[$ la génération des variables *x* et *t* se font comme suit :

$$\begin{aligned} x_1 &= -2, \\ x_2 &= x_1 + h, \\ x_3 &= x_2 + h = x_1 + 2h, \\ &\cdots\cdots\cdots \\ x_i &= x_1 + (i-1)h, \\ &\cdots\cdots\cdots \\ x_N &= x_1 + (N-1)h = 3 \end{aligned} \qquad \begin{aligned} t_1 &= 0, \\ t_2 &= t_1 + \tau = \tau, \\ t_3 &= t_2 + \tau = t_1 + 2\tau = 2\tau, \\ &\cdots\cdots\cdots \\ t_n &= t_1 + (n-1)\tau = (n-1)\tau, \\ &\cdots\cdots\cdots \end{aligned}$$

$\lambda = 0.8$ et $h = 0.1$ alors $\tau = 0.08$ et par conséquent $N - 1 = (3 - x_1)/h \ \Rightarrow\ N = 1 + (3+2)/0.1 = 51$

pour la condition initiale,

$$u(x,\ 0) = \varphi(x) = \begin{cases} 1 - |x| & \text{si } |x| \le 1, \\ 0 & \text{si } |x| \ge 1. \end{cases} \Rightarrow\ u_i^1 = \begin{cases} 1 - |x_i| & \text{si } |x_i| \le 1, \\ 0 & \text{si } |x_i| \ge 1. \end{cases} \quad \forall\ i = 1,\ 2,\ \ldots,\ N$$

pour la condition à la limite,

$$u(-2,\ t)=0 \ \Rightarrow\ u_1^n=0 \ \ \forall\ n=1,\ 2,\ \ldots$$

D'après l'équation aux différences, la condition initiale et la condition à la limite supérieure, ce processus explicite permet de calculer que N-(n+1) valeurs de u_i^{n+1} c'est-à-dire seulement pour $i=1,\ 2,\ 3,\ \ldots,\ N-(n+1)$, et on remarque très bien que le nombre de valeurs de u_i^{n+1} diminue en fonction de l'augmentation de n (Fig. 7.17). Et pour résoudre ce problème on considère (Strikwerda, 2004) l'égalité :

$$u_N^{n+1}=u_{N-1}^{n+1}$$

Finalement, le processus qui permet de déterminer une solution numérique de l'équation différentielle par la méthode des différences finies :

$$u_i^1=\begin{cases}1-|x_i| & \text{si}\ \ |x_i|\le 1,\\ 0 & \text{si}\ \ |x_i|\ge 1.\end{cases}\quad \forall\ i=1,\ 2,\ \ldots,\ N$$

$$u_1^n=0\ \ \forall\ n=1,\ 2,\ \ldots$$

$$u_i^{n+1}=\begin{cases}0.9u_{i-1}^n+0.1u_{i+1}^n & \text{si}\ \ i=2,\ 3,\ \ldots,\ N-1,\\ u_{i-1}^{n+1} & \text{si}\ \ i=N.\end{cases}$$

Le script MATLAB basant sur ce processus permet de trouver la solution numérique (Fig. 7.18) pour une évolution du phénomène dans un temps fini (t = 1.6).

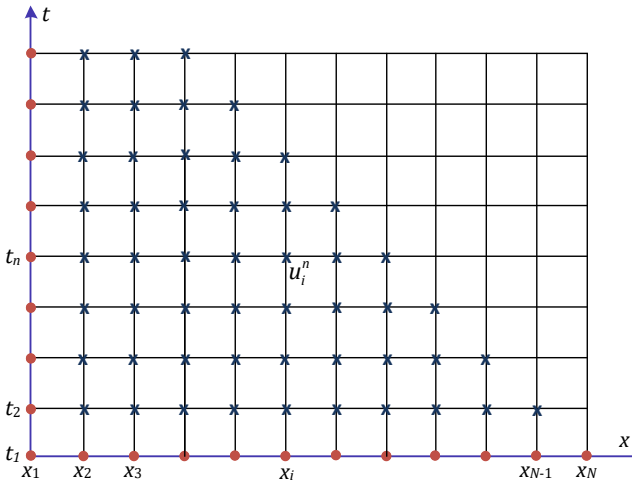


Fig. 7.17 : Les points qui peuvent être calculés.

```
a=1;
tau=0.08;
h=0.1;
x=-2:h:3;
t=0:tau:1.6;
N=numel(x); M=numel(t);
for i=1:N
 if abs(x(i))<=1
   u(i,1)=1-abs(x(i));
 else
   u(i,1)=0;
 end
end
for n=1:M
  u(1,n)=0;
end
for n=1:M-1
 for i=2:N
  if i==N
   u(i,n+1)=u(i-1,n+1);
  else
   u(i,n+1)=0.9*u(i-1,n)+0.1*u(i+1,n);
  end
```

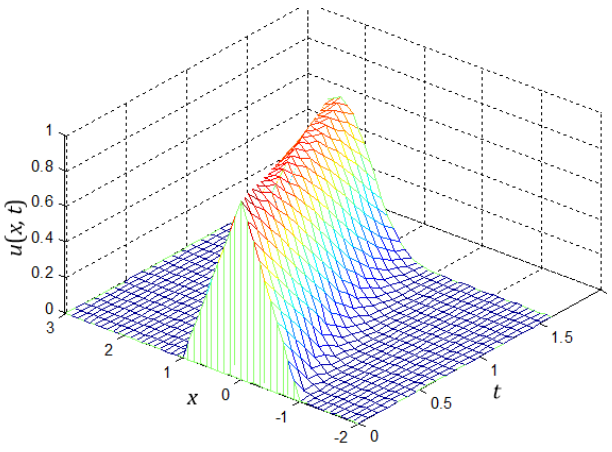


Fig. 7.18 : Résolution avec Lax–Friedrichs.

```
 end
end
mesh(t,x,u) %(Fig. 7.18)
```

2. La comparaison de la solution approchée par la méthode des différences finies et la solution exacte de l'équation $u(x,\ 1.6)=\varphi(x-1.6)$ pour $\lambda=0.8$ est représentée dans la figure ci-dessous (Fig. 7.19).

3. De même, la comparaison de la solution approchée par la méthode des différences finies et la solution exacte de l'équation $u(x,\ 1.6)=\varphi(x-1.6)$ pour $\lambda=1.6$ avec la précédente (cas de $\lambda=0.8$) est représentée dans la figure ci-dessous (Fig. 7.20).

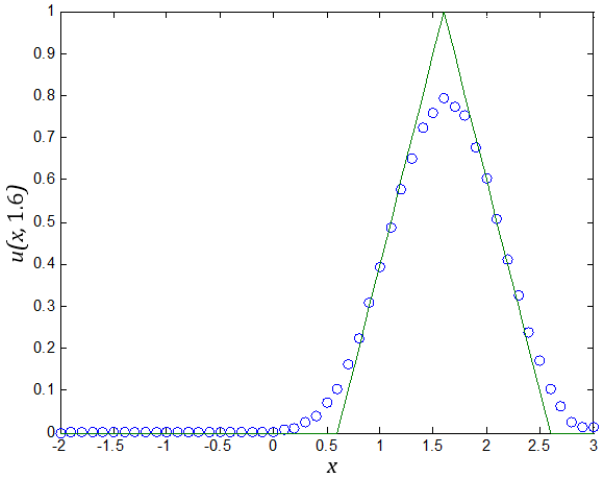


Fig. 7.19 : Comparaison ente la solution approchée et la solution exacte pour (λ = 0.8).

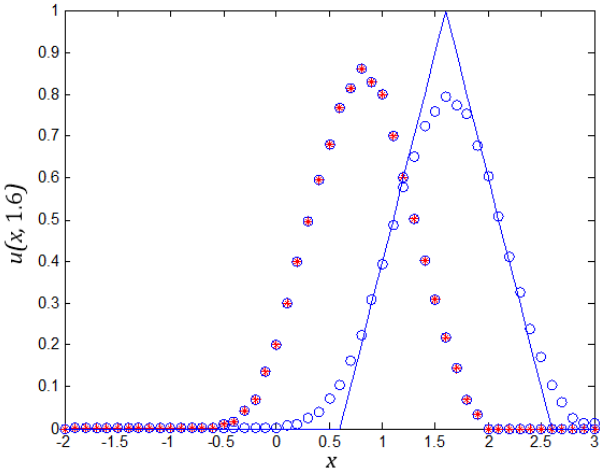


Fig. 7.20 : Comparaison ente les solutions approchées pour (λ = 0.8) et pour (λ = 1.6) et la solution exacte.

Exercice 9

Soit l'équation de convection avec les conditions initiale et aux limites suivants :

$$\begin{cases} \dfrac{\partial u}{\partial t}+\dfrac{\partial u}{\partial x}=0, \\ u(x,\ 0)=\varphi(x),\ \ 0\le x\le 2, \\ u(0,\ t)=0,\ \ t\ge 0. \end{cases}$$

Ou $\varphi(x)$ est une fonction dépend de la position x est exprimée par :

$$\varphi(x)=\begin{cases} 0, & x\le 0.9 \\ 10(x-0.9), & 0.9\le x\le 1 \\ 10(1.1-x), & 1\le x\le 1.1 \\ 0, & x\ge 1.1 \end{cases}$$

1. Sachant que les pas de discrétisation $h=0.05$ et $\tau=0.04$, déterminer une solution numérique de cette équation en utilisant le schéma explicite de Lax–Friedrichs.

2. Comparer la solution obtenue avec celle de la solution exacte du problème est qui exprimée par :

$$u(x,\ t)=\varphi(x-1.5)$$

1. Soit le script, qui utilise la fonction **`FonctionPhi2x`** :

```
h=0.05; tau=0.04; lamda=tau/h;
```

```
x=0:h:2; t=0:tau:0.5;
N=numel(x); M=numel(t);
for i=1:N, u(i,1)=FonctionPhi2x(x(i)),end
 for n=1:M, u(1,n)=0,end
for n=1:M-1
 for i=2:N
  if i==N, u(i,n+1)=u(i-1,n+1);
  else,u(i,n+1)=0.5*(1+lamda)*u(i-1,n)+0.5*(1-lamda)*u(i+1,n),end
 end
end
mesh(t,x,u) %(Fig. 7.21)
```

```
function [phi] = FonctionPhi2x(x)
if x<=0.9
   phi=0;
elseif (x>0.9) && (x<=1)
   phi=10*(x-0.9);
elseif (x>1) && (x<=1.1)
   phi=10*(1.1-x);
else
   phi=0;
end
end
```

Permettant d'avoir la solution sous la forme graphique suivante (Fig. 7.21)

2. La comparaison de la solution approchée par la méthode des différences finies et la solution exacte $u(x,\ t)=\varphi(x-1.5)$ est représentée dans la figure ci-contre dessous (Fig. 7.22).

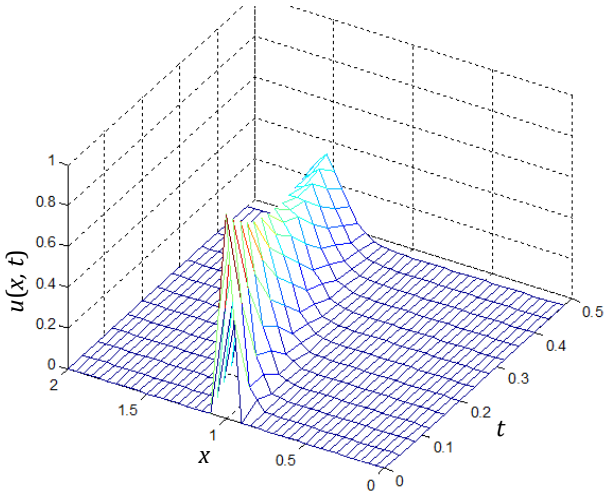


Fig. 7.21 : Résolution avec Lax–Friedrichs.

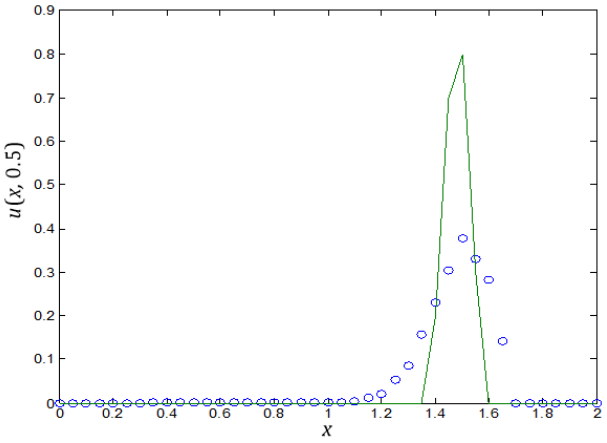


Fig. 7.22 : Comparaison ente la solution approchée et la solution exacte.

Exercice 10

On considère, pour l'équation de la chaleur : $\begin{cases} \dfrac{\partial u}{\partial t}-\nu\dfrac{\partial^2 u}{\partial x^2}=0 \quad \forall x\in\Re,\ \ \forall t>0 \\ u(x,\ 0)=\varphi(x) \quad \forall x\in\Re. \end{cases}$

Le schéma d'approximation par différences finies sur un maillage régulier de pas h en espace et τ en temps (schéma de Gear) :

$$\frac{3u_i^{n+1}-4u_i^n+u_i^{n-1}}{2\tau}-\nu\frac{u_{i+1}^{n+1}+u_{i-1}^{n+1}-2u_i^{n+1}}{h^2}=0$$

Note on pourra poser $\xi=2\nu\tau/h^2$

1. Dessiner le schéma, de quel type de schéma s'agit-il ?
2. Analysez la stabilité du schéma par la méthode de von Neumann.
3. Quel est l'ordre du schéma ?

1. La figure suivante (Fig. 7.23) donne le stencil du schéma aux différences de l'équation différentielle.

2. Pour l'analyse de la stabilité par von Neumann, soit :

$$u_i^n=g^n e^{ji\theta}$$

en remplaçant dans le schéma aux différences :

$$\frac{3g^{n+1}e^{ji\theta}-4g^n e^{ji\theta}+g^{n-1}e^{ji\theta}}{2\tau}-\nu\frac{g^{n+1}e^{j(i+1)\theta}+4g^{n+1}e^{j(i-1)\theta}-2g^{n+1}e^{ji\theta}}{h^2}=0$$

ou,

$$\frac{3g-4+g^{-1}}{2\tau}-\nu\frac{ge^{j\theta}+ge^{-j\theta}-2g}{h^2}=0$$

Devient après simplification,

$$3g-4+\frac{1}{g}-2\xi g(\cos\theta-1)=0$$

sachant que,

$$\cos\theta=\cos\left(\tfrac{1}{2}\theta+\tfrac{1}{2}\theta\right)=\cos^2\tfrac{1}{2}\theta-\sin^2\tfrac{1}{2}\theta$$

et :

$$\cos^2\tfrac{1}{2}\theta+\sin^2\tfrac{1}{2}\theta=1$$

alors,

$$\cos\theta-1=\cos^2\tfrac{1}{2}\theta-\sin^2\tfrac{1}{2}\theta-\cos^2\tfrac{1}{2}\theta-\sin^2\tfrac{1}{2}\theta=-2\sin^2\tfrac{1}{2}\theta$$

soit :

$$\left(3+4\xi\sin^2\tfrac{1}{2}\theta\right)g^2-4g+1=0$$

L'équation obtenue est de second degré, son discriminant est :

$$\Delta'=(-2)^2-\left(3+4\xi\sin^2\tfrac{1}{2}\theta\right)=1-4\xi\sin^2\tfrac{1}{2}\theta$$

Si $\Delta'\geq 0$, c'est-à-dire, $1-4\xi\sin^2\tfrac{1}{2}\theta\geq 0$ l'équation possède alors deux racines :

$$\begin{cases} g_1=\dfrac{2-\sqrt{1-4\xi\sin^2\frac{1}{2}\theta}}{3+4\xi\sin^2\frac{1}{2}\theta}, \\ g_2=\dfrac{2+\sqrt{1-4\xi\sin^2\frac{1}{2}\theta}}{3+4\xi\sin^2\frac{1}{2}\theta}. \end{cases}$$

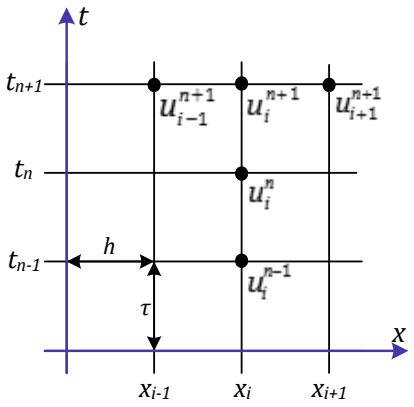


Fig. 7.23 : Stencil du schéma aux différences.

or,

$$1-4\xi\sin^2\tfrac{1}{2}\theta\geq 0 \;\Rightarrow\; 4\xi\sin^2\tfrac{1}{2}\theta\leq 1 \;\Rightarrow\; \xi\leq 1/4$$

par conséquent,

$$|g_1|\leq 1 \quad \text{et} \quad |g_2|\leq 1$$

le schéma aux différences est stable si :

$$\frac{\tau\nu}{h^2} \le \frac{1}{8}$$

3. Pour l'équation différentielle :

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

et cherchons l'ordre du schéma :

$$\left(\frac{\partial u}{\partial t}\right)_i^n \approx \frac{3u_i^{n+1} - 4u_i^n + u_i^{n-1}}{2\tau}$$

soit les développements :

$$\begin{cases} u_i^{n+1} = u_i^n + \tau\left(\dfrac{\partial u}{\partial t}\right)_i^n + O(\tau), \\ u_i^{n-1} = u_i^n - \tau\left(\dfrac{\partial u}{\partial t}\right)_i^n + O(\tau). \end{cases} \Rightarrow \begin{cases} 3u_i^{n+1} = 3u_i^n + 3\tau\left(\dfrac{\partial u}{\partial t}\right)_i^n + O(\tau), \\ u_i^{n-1} = u_i^n - \tau\left(\dfrac{\partial u}{\partial t}\right)_i^n + O(\tau). \end{cases}$$

alors,

$$3u_i^{n+1} + u_i^{n-1} = 4u_i^n + 2\tau\left(\frac{\partial u}{\partial t}\right)_i^n + O(\tau) \Rightarrow 3u_i^{n+1} + u_i^{n-1} - 4u_i^n = 2\tau\left(\frac{\partial u}{\partial t}\right)_i^n + O(\tau)$$

$$\Rightarrow \left(\frac{\partial u}{\partial t}\right)_i^n = \frac{3u_i^{n+1} - 4u_i^n + u_i^{n-1}}{2\tau} + O(\tau)$$

finalement,

$$\left(\frac{\partial u}{\partial t}\right)_i^n - \nu\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n = 0 \Leftrightarrow \frac{3u_i^{n+1} - 4u_i^n + u_i^{n-1}}{2\tau} + \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2} + O\left(h^2\right) + O(\tau) = 0$$

donc le schéma aux différences est de l'ordre 1 en temps et d'ordre 2 en espace.

Exercice 11

Considérons l'équation de Laplace d'une fonction de deux variables indépendantes $z = f(x,\ y)$:

$$\frac{\partial^2 f(x,\ y)}{\partial x^2} + \frac{\partial^2 f(x,\ y)}{\partial y^2} = 0$$

Avec $x, y \in [0, 1]$ et les conditions aux limites du problème sont :

$$f(x,\ y) = \begin{cases} 0 & x = 0,\ \forall\ y \\ 0 & y = 0,\ \forall\ x \\ 1 & x = 1,\ y \neq 0 \\ 1 & y = 1,\ x \neq 0 \end{cases}$$

Résoudre par la méthode des différences finies en prend comme pas de discrétisation $h = 0.25$.

D'après le schéma aux différences finies centrales, l'équation de Laplace devient :

$$f(x_{i+1},\ y_j) + f(x_{i-1},\ y_j) + f(x_i,\ y_{j+1}) + f(x_i,\ y_{j-1}) u_{i,j-1} - 4f(x_i,\ y_j) = 0$$

Suivant les conditions aux limites et le pas de discrétisation suivant x et y qui est $h = 0.25$ alors il faut déterminer les valeurs de la fonction f dans les points dont les coordonnées $(0.25,\ 0.25)$, $(0.25,\ 0.50)$, $(0.25,\ 0.75)$, $(0.50,\ 0.25)$, $(0.50,\ 0.50)$, $(0.50,\ 0.75)$, $(0.75,\ 0.25)$, $(0.75,\ 0.50)$ et $(0.75,\ 0.75)$.

Les valeurs des 16 points correspondants aux conditions aux limites sont connues (Fig. 7.24). L'équation pour chacun des points du maillage, dans l'ordre indiqué ci-dessus, est :

$$f(0.50,\ 0.25)+f(0.25,\ 0.50)-4f(0.25,\ 0.25)=0-f(0.00,\ 0.25)-f(0.25,\ 0.00)=0$$
$$f(0.50,\ 0.50)+f(0.25,\ 0.75)+f(0.25,\ 0.25)-4f(0.25,\ 0.50)=0-f(0.00,\ 0.50)=0$$
$$f(0.50,\ 0.75)+f(0.25,\ 0.50)-4f(0.25,\ 0.75)=0-f(0.00,\ 0.75)-f(0.25,\ 1.00)=-1$$
$$f(0.75,\ 0.25)+f(0.25,\ 0.25)+f(0.50,\ 0.50)-4f(0.50,\ 0.25)=0-f(0.50,\ 0.00)=0$$
$$f(0.75,\ 0.50)+f(0.25,\ 0.50)+f(0.50,\ 0.75)+f(0.50,\ 0.25)-4f(0.50,\ 0.50)=0$$
$$f(0.75,\ 0.75)+f(0.25,\ 0.75)+f(0.50,\ 0.50)-4f(0.50,\ 0.75)=0-f(0.50,\ 1.00)=-1$$
$$f(0.50,\ 0.25)+f(0.75,\ 0.50)-4f(0.75,\ 0.25)=0-f(1.00,\ 0.25)-f(0.75,\ 0.00)=-1$$
$$f(0.50,\ 0.50)+f(0.75,\ 0.75)+f(0.75,\ 0.25)-4f(0.75,\ 0.50)=0-f(1.00,\ 0.50)=-1$$
$$f(0.50,\ 0.75)+f(0.75,\ 0.50)-4f(0.75,\ 0.75)=0-f(1.00,\ 0.75)-f(0.75,\ 1.00)=-2$$

Qui peut être s'écrit sous la forme matricielle :

$$\begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} f(0.25,\ 0.25) \\ f(0.25,\ 0.50) \\ f(0.25,\ 0.75) \\ f(0.50,\ 0.25) \\ f(0.50,\ 0.50) \\ f(0.50,\ 0.75) \\ f(0.75,\ 0.25) \\ f(0.75,\ 0.50) \\ f(0.75,\ 0.75) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ -1 \\ -1 \\ -1 \\ -2 \end{bmatrix}$$

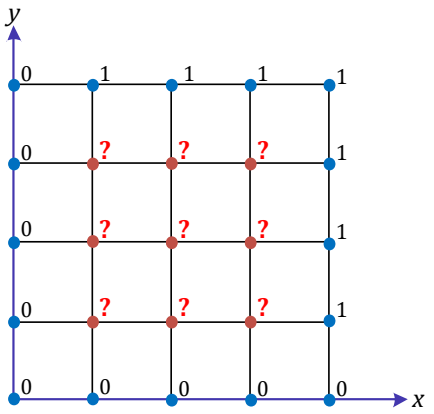


Fig. 7.24 : The mesh représentation.

La résolution du système d'équations linéaires avec MATLAB conduit à la solution suivante :

$$f(0.25,\ 0.25)=0.14 \quad f(0.50,\ 0.25)=0.29 \quad f(0.75,\ 0.25)=0.50$$
$$f(0.25,\ 0.50)=0.29 \quad f(0.50,\ 0.50)=0.50 \quad f(0.75,\ 0.50)=0.71$$
$$f(0.25,\ 0.75)=0.50 \quad f(0.50,\ 0.75)=0.71 \quad f(0.75,\ 0.75)=0.86$$

ce qui est en effet une bonne approximation du comportement de l'équation de Laplace, même s'il est approximatif en raison de la grande valeur de pas utilisée. Résoudre le problème de la valeur limite en utilisant une valeur plus petite de *h* donnera une meilleure approximation.

7. Exercices supplémentaires

Exercice 1

1. Dans quelle partie du plan Oxy l'équation suivante est de type parabolique ?

$$\frac{\partial^2 u}{\partial x^2}+4\frac{\partial^2 u}{\partial x\partial y}+\left(x^2+y^2\right)\frac{\partial^2 u}{\partial y^2}=\sin(xy)$$

2. Vérifier que $u(x,\ t)=e^{-t}\cos\pi\left(x-\frac{1}{2}\right)$ est une solution de l'équation de la chaleur :

$$\frac{1}{\pi^2}\frac{\partial^2 u}{\partial x^2}=\frac{\partial u}{\partial t}$$

Exercice 2

En utilisant la méthode des différences finies avant pour $h=0.2$ et $\tau=0.02$, déterminer une solution numérique du problème suivant :

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}=\dfrac{\partial u}{\partial t},\ 0<x<1,\ 0<t<0.2,\\ u(0,\ t)=u(1,\ t)=0,\ 0\le t\le 0.2,\\ u(x,\ 0)=4x-4x^2,\ 0\le x\le 1.\end{cases}$$

Exercice 3

En utilisant la méthode des différences finies avant pour $h=10$ et $\tau=0.01$, déterminer une solution numérique du problème suivant :

$$\begin{cases}10\dfrac{\partial^2 u}{\partial x^2}=\dfrac{\partial u}{\partial t},\ 0<x<100,\ 0<t<0.05,\\ u(0,\ t)=u(100,\ t)=0,\ 0\le t\le 0.05,\\ u(x,\ 0)=\begin{cases}0 \text{ si } 0\le x<25\\ 50 \text{ si } 25\le x\le 75\\ 0 \text{ si } 75<x\le 100\end{cases}\end{cases}$$

Exercice 4

En utilisant la méthode des différences finies de Crank-Nicholson pour $h=0.1$ et $\tau=0.01$, déterminer une solution numérique du problème :

$$\begin{cases}\dfrac{1}{\pi^2}\dfrac{\partial^2 u}{\partial x^2}=\dfrac{\partial u}{\partial t},\\ u(0,\ t)=u(1,\ t)=0,\ 0\le t\le 0.1,\\ u(x,\ 0)=\sin\pi x,\ 0\le x\le 1.\end{cases}$$

Comparer la solution numérique à celle de la solution exacte exprimée par : $u(x,\ t)=e^{-t}\sin\pi x$.

Exercice 5

En utilisant la méthode des différences finies explicite pour $h=0.1$ et $\tau=0.05$, déterminer une solution numérique du problème :

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}=\dfrac{\partial^2 u}{\partial t^2},\\ u(0,\ t)=u(1,\ t)=0,\ 0\le t\le 0.1,\\ \left.\dfrac{\partial u}{\partial t}\right|_{(x,\ 0)}=x^2,\ u(x,\ 0)=x\ 0\le x\le 1.\end{cases}$$

Exercice 6

En utilisant la méthode des différences finies explicite pour $h=0.4$ et $\tau=0.25$, déterminer une solution numérique du problème :

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2}=\dfrac{\partial^2 u}{\partial t^2}, \\ u(0,\ t)=u(4,\ t)=0,\ 0\le t\le 1, \\ u(x,\ 0)=\begin{cases} 16x^2(1-x)^2 \text{ si } 0\le x\le 1, \\ 0 \text{ si } 1\le x\le 4. \end{cases} \\ \left.\dfrac{\partial u}{\partial t}\right|_{(x,\ 0)}=0,\ x\ 0\le x\le 4. \end{cases}$$

Exercice 7

Déterminer la solution de l'équation de Laplace pour le carré $0<x<a$ et $0<y<a$:

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0$$

Pour les conditions :

(a) $u(x,\ 0)=u(x,\ \pi)=u(\pi,\ y)=0,\ u(0\ y)=\sin^2 y$.

(b) $u(x,\ 0)=u(0,\ y)=u(x,\ 1)=0,\ u(1\ y)=-y(y-1)$.

(c) $u(x,\ 0)=u(0,\ y)=u(x,\ 2)=0,\ u(2\ y)=10$.

(d) $u(x,\ 0)=u(0,\ y)=u(x\ \pi)=0,\ u(\pi\ y)=\sin y$.

Exercice 8

Dans le rectangle $\Omega=\{(x,\ y),\ 0\le x\le a,\ 0\le y\le b\}$, on considère le problème (E) suivant : trouver u tel que :

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=-f\ ,\ u|_{\Gamma}=g$$

où f et g sont des fonctions données. On admettra que ce problème admet une solution unique. On choisit un maillage régulier de pas h dans chaque direction (on suppose a/b rationnel) et on appelle D_h l'ensemble des points du maillage situés dans Ω et C_h les points du maillage sur Γ.

1) En utilisant la méthode d'approximation par différences finies construite à partir du schéma à cinq points, montrer en posant

$$Lv_{i,\ j}=\frac{1}{h^2}\{v_{i+1,\ j}+v_{i-1,\ j}+v_{i,\ j+1}+v_{i,\ j-1}-4v_{i,\ j}\}$$

que l'on a :

$$\max_{D_h}|v_{i,\ j}|\le\max_{C_h}|v_{i,\ j}|+\frac{1}{h^2}(a^2+d^2)\max_{D_h}|Lv_{i,\ j}|$$

Pour cela, on suggère de procéder comme suit :

a) Montrer que si $Lv_{i,\ j}\ge 0$ sur D_h, alors,

$$\max_{D_h} v_{i,\ j}\le\max_{C_h} v_{i,\ j}$$

b) Considérer la fonction définie sur D_h par $\Phi_{i,\,j}=\dfrac{h^2}{4}\left(i^2+j^2\right)$ et appliquer les résultats de la question a) à la fonction $w^{\varepsilon}=\varepsilon v+M_L\Phi$, avec : $M_L=\max\limits_{D_h}\left|v_{i,\,j}\right|$ et $\varepsilon=\pm 1$. Conclure.

2) On suppose que les hypothèses sur f et g sont telles que la solution u du problème (E) est dans $C^4(\bar{\Omega})$; on pose, pour tout indice i, j :

$$e_{i,\,j}=u\left(ih,\ jh\right)-v_{i,\,j}$$

où les $v_{i,\,j}$ sont calculés par le schéma de la question précédente, *i.e.* : $Lv_{i,\,j}=-f_{i,\,j}$ dans D_h et $v_{i,\,j}=g\left(ih,\ jh\right)$ sur C_h. Montrer que l'on a :

$$\max_{D_h}\left|Le_{i,\,j}\right|\le\frac{1}{6}\mathrm{M}h^2$$

où

$$\mathrm{M}=\max\left\{\max_{(x,\,y)\in\bar{\Omega}}\left|\frac{\partial^4 u}{\partial x^4}\left(x,\ y\right)\right|,\ \max_{(x,\,y)\in\bar{\Omega}}\left|\frac{\partial^4 u}{\partial y^4}\left(x,\ y\right)\right|\right\}$$

3) Donner une estimation de l'erreur $\left|e_{i,\,j}\right|$ et conclure.

Exercice 9

Considérons une tige de fer homogène de 100 cm de long qui est chauffée à la distribution de température initiale $f\left(x\right)=100-x,\ 0\le x\le 100$. A l'instant t = 0 la surface latérale est isolée, et l'extrémité gauche de la tige est plongée dans un réservoir d'huile maintenu à 30°C, tandis que l'extrémité droite reste à la température constante de 70°C. La diffusivité thermique α du fer est 0.15 cm^2/sec. Use h = 10 et une valeur appropriée de k de sorte que $\lambda = \alpha 2k/h^2 = 0.5$. Calculer la solution pour cinq étapes.

Exercice 10

Résoudre l'équation des ondes

$$\frac{\partial^2 u}{\partial x^2}=\frac{\partial^2 u}{\partial t^2},\ 0\le x\le 4,\ t>0$$

Pour les conditions initiale et aux limites :

$$\begin{cases} u\left(0,\ t\right)=0,\ u\left(4,\ t\right)=0 \\ u\left(x,\ 0\right)=0,\quad \dfrac{\partial u}{\partial t}\left(0,\ 0\right)=\begin{cases} 0 & 0\le x<1 \\ 4 & 1\le x\le 1.5 \\ 0 & 1.5<x<3 \\ -4 & 3\le x\le 3.5 \\ 0 & 3.5<x\le 4 \end{cases} \end{cases}$$

L'énergie totale dans la corde avec des extrémités épinglées à x = 0 et x = 4 est composé de l'énergie cinétique plus l'énergie potentielle et à tout instant t donné par :

$$E\left(t\right)=\frac{1}{2}\int_0^4\left[\left(\frac{\partial u\left(x,\ t\right)}{\partial t}\right)^2+\left(\frac{\partial u\left(x,\ t\right)}{\partial x}\right)^2\right]dx$$

Si au départ $u(x,\ t)=f(x)$ et $\dfrac{\partial u(x,\ 0)}{\partial t}=g(x)$, l'énergie initiale est :

$$E(0)=\frac{1}{2}\int_0^4\left[\left(g(x)\right)^2+\left(f'(x)\right)^2\right]dx$$

L'énergie est conservée si $E(t)=E(0)$ pour n'importe temps *t*.

Annexe 1.

Algèbre des matrices

Dans cette annexe, nous rappelons les notions élémentaires d'algèbre des matrices (Bradley, 1975, Noble, 1969 et Halmos, 1958), les résultats complémentaires sur les valeurs et vecteurs propres (Householder, 1975 et Wilkinson, 1965) qui sont très utilisées dans les méthodes numériques.

1. Espaces vectoriels

Un espace vectoriel sur un corps E (ou E-espace vectoriel) un corps des nombres (réels $\mathbb{R}$ ou complexes $\mathbb{C}$) est un ensemble non vide V sur lequel l'addition interne + et la multiplication externe par un scalaire possèdent les propriétés suivantes :

1. $(V,\ +)$ est un groupe abélien commutatif,
2. $\forall\ \alpha \in K,\ \forall\ \mathbf{v},\ \mathbf{w} \in V,\ \alpha(\mathbf{v}+\mathbf{w}) = \alpha\mathbf{v} + \alpha\mathbf{w}$,
3. $\forall\ \alpha,\ \beta \in K,\ \forall\ \mathbf{v} \in V,\ \alpha(\alpha+\beta)\mathbf{v} = \alpha\mathbf{v} + \beta\mathbf{v}$, $(\alpha\beta)\mathbf{v} = \alpha(\beta\mathbf{v})$ et $1.\mathbf{v} = \mathbf{v}$

où 1 est l'élément unité de E. Les éléments de V sont appelés *vecteurs,* ceux de E sont les *scalaires*.

Par exemple, l'ensemble des polynômes $P_n(x) = \sum_{i=0}^{n} a_i x^i$ de degré inférieur ou égal à n est $\mathbb{C}$-espace vectoriel.

Par définition, une partie non vide W de V est un sous-espace vectoriel de V si et seulement si :

$$\forall\ (\mathbf{v},\ \mathbf{w}) \in W^2,\ \forall\ (\alpha,\ \beta) \in E^2,\ \alpha\mathbf{v} + \beta\mathbf{w} \in W$$

En particulier, l'ensemble W des combinaisons linéaires d'une famille de k vecteurs de V $\{\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots,\ \mathbf{v}_k\}$ est un sous-espace vectoriel de V, appelé sous espace engendré par la famille de vecteurs, c'est-à-dire :

$$\mathbf{v} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \cdots + \alpha_k\mathbf{v}_k$$

La famille $\{\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots,\ \mathbf{v}_k\}$ est appelée *famille génératrice* de W.

Si W_1, W_2, ..., W_m sont des sous espaces vectoriels de V, alors l'ensemble :

$$S = \{\mathbf{w} :\ \mathbf{w} = \mathbf{v}_1 + \mathbf{v}_2 + \cdots + \mathbf{v}_m \text{ avec } \mathbf{v}_i \in W_i,\ i = 1,\ 2,\ \ldots,\ m\}$$

est aussi un sous espace vectoriel de V.

S est la *somme directe* des sous espaces W_i si tout élément $\mathbf{s} \in S$ admet une unique décomposition de la forme $\mathbf{s} = \mathbf{v}_1 + \mathbf{v}_2 + \cdots + \mathbf{v}_m$ avec $\mathbf{v}_i \in W_i$ et $i = 1,\ 2,\ \ldots,\ m$. Dans ce cas, S s'écrit comme :

$$S = W_1 \oplus W_2 \oplus \cdots \oplus W_m$$

Une famille de vecteurs $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ d'un espace vectoriel V est dite *libre* si les vecteurs $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$ sont *linéairement indépendants* c'est-à-dire si la relation :

$$\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \cdots + \alpha_m\mathbf{v}_m = \mathbf{0} \Rightarrow \alpha_1 = \alpha_2 = \cdots = \alpha_m = 0$$

Dans le cas contraire, la famille est dite *liée*.

On appelle base de V toute famille libre et génératrice de V. Si $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ est une base de V, l'expression $\mathbf{v} = \lambda_1\mathbf{u}_1 + \lambda_2\mathbf{u}_2 + \cdots + \lambda_n\mathbf{u}_n$ est appelée *décomposition* de $\mathbf{v}$ et les scalaires $\lambda_1, \lambda_2, \ldots, \lambda_n \in E$ sont les *composantes* de $\mathbf{v}$ sur la base donnée.

2. Matrices

Soient m et n deux entiers positifs. On appelle matrice à m lignes et n colonnes à coefficients dans E, un ensemble de mn scalaires $a_{ij} \in E$ avec $i = 1, 2, \ldots, m$ et $j = 1, 2, \ldots, n$ représentés dans le tableau rectangulaire :

$$\mathbf{A} = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Si $E = \mathbb{R}$ ou $E = \mathbb{C}$, alors $\mathbf{A} \in \mathbb{R}^{m\times n}$ ou $\mathbf{A} \in \mathbb{C}^{m\times n}$. Si $n = m$ $\mathbf{A}$ est une matrice carrée ou d'ordre n. Le n-uple $(a_{11}, a_{22}, \ldots, a_{nn})$ est le diagonale principale.

On appelle vecteur ligne (resp. vecteur colonne) une matrice n'ayant qu'une ligne (resp. colonne). Sauf mention explicite du contraire, nous supposerons toujours qu'un vecteur est un vecteur colonne. Dans le cas $n = m = 1$, la matrice désigne simplement un scalaire de E. Il est quelquefois utile de distinguer, à l'intérieur d'une matrice, l'ensemble constitué de lignes et de colonnes particulières.

2.1. Opérations, trace et déterminant

Soit $\mathbf{A} = (a_{ij})$ et $\mathbf{B} = (b_{ij})$ deux matrices de dimension $m\times n$ sur E. On dit que $\mathbf{A}$ est égale à $\mathbf{B}$, si $a_{ij} = b_{ij}$ pour $i = 1, 2, \ldots, m$ et $j = 1, 2, \ldots, n$. On définit de plus les opérations suivantes :

- La somme des matrices $\mathbf{A}$ et $\mathbf{B}$ est la matrice $\mathbf{C}$ de dimension $m\times n$ dont les coefficients $c_{ij} = a_{ij} + b_{ij}$ pour $i = 1, 2, \ldots, m$ et $j = 1, 2, \ldots, n$. L'élément neutre pour la somme matricielle est la matrice nulle $\mathbf{0}$ de dimension $m\times n$. constituée de coefficients tous nuls.

- La multiplication de la matrice $\mathbf{A}$ de dimension $m\times n$ par un scalaire $\lambda \in E$ est la matrice $\mathbf{C} = \lambda\mathbf{A}$ de dimension $m\times n$ dont les coefficients $c_{ij} = \lambda a_{ij}$ pour $i = 1, 2, \ldots, m$ et $j = 1, 2, \ldots, n$.

- Le produit d'une matrice $\mathbf{A}$ de dimension $m\times p$ par une matrice $\mathbf{B}$ de dimension $p\times n$ est la matrice $\mathbf{C}$ de $m\times n$, dont les coefficients $c_{ij} = \sum_{k=1}^{p} a_{ik}b_{kj}$ pour $i = 1, 2, \ldots, m$ et $j = 1, 2, \ldots, n$.

Le produit matriciel est associatif et distributif par rapport à la somme matricielle, mais il n'est pas commutatif en général.

Dans le cas des matrices carrées, l'élément neutre pour le produit matriciel est la matrice carrée d'ordre *n*, appelée aussi matrice unité d'ordre *n* ou, plus fréquemment, matrice identité, définie par $\mathbf{I}_n = \left(\delta_{ij}\right)$ $\left(\delta_{ij} \text{ est le symbole de Kronecker}\right)$. La matrice identité est, par définition, la seule matrice telle que $\mathbf{I}_n\mathbf{A} = \mathbf{A}\mathbf{I}_n = \mathbf{A}$ pour toutes les matrices carrées **A**.

Si $\mathbf{AB} = \mathbf{I}_n$ ou $\mathbf{BA} = \mathbf{I}_n$ alors **A** (de **B**) est la matrice inverse de **B** (de **A**), ceci est possible si et seulement si la matrice **A** (ou **B**) est régulière ou non singulière. La matrice inverse de **A** est notée par $\mathbf{A}^{-1}$. De plus, si **A** et **B** sont deux matrices inversibles d'ordre *n*, leur produit **AB** est aussi inversible et $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$.

Si **A** une matrice de dimension $m \times n$ le transpose de la matrice est la matrice $\mathbf{A}^T$ de dimension $n \times m$ obtenue en échangeant les lignes et les colonnes de **A**. On a clairement $\left(\mathbf{A}^T\right)^T = \mathbf{A}$, $\left(\mathbf{A}+\mathbf{B}\right)^T = \mathbf{A}^T + \mathbf{B}^T$, $\left(\mathbf{AB}\right)^T = \mathbf{A}^T\mathbf{B}^T$, $\left(\mathbf{A}^T\right)^T = \mathbf{A}$ et $\left(\alpha\mathbf{A}\right)^T = \alpha\mathbf{A}^T$, $\forall \alpha \in \mathbb{R}$. Si A est inversible, on a aussi $\left(\mathbf{A}^T\right)^{-1} = \left(\mathbf{A}^{-1}\right)^T$.

Une matrice $\mathbf{A} \in \mathbb{R}^{n\times n}$ est dite symétrique si $\mathbf{A} = \mathbf{A}^T$, et antisymétrique si $\mathbf{A} = -\mathbf{A}^T$. Elle est dite orthogonale si $\mathbf{A}\mathbf{A}^T = \mathbf{A}^T\mathbf{A} = \mathbf{I}_n$, c'est-à-dire si $\mathbf{A}^{-1} = \mathbf{A}^T$.

Soit $\mathbf{A} \in \mathbb{C}^{m\times n}$ une matrice dont les éléments sont des nombres complexes, la matrice $\mathbf{B} = \mathbf{A}^*$ est appelée adjointe ou transposée conjuguée de **A** si $b_{ij} = \overline{a}_{ij}$ où $\overline{a}_{ij}$ est le complexe conjugué de a_{ij}, on a :

$$\left(\mathbf{A}+\mathbf{B}\right)^* = \mathbf{A}^* + \mathbf{B}^*,\ \left(\mathbf{AB}\right)^* = \mathbf{A}^*\mathbf{B}^* \text{ et } \left(\alpha\mathbf{A}\right)^* = \overline{\alpha}\,\mathbf{A}^*$$

La matrice $\mathbf{A} \in \mathbb{C}^{n\times n}$ est dite hermitienne ou auto adjointe si $\mathbf{A}^T = \overline{\mathbf{A}}$, c'est-à-dire si $\mathbf{A}^*=\mathbf{A}$, et elle est dite unitaire si $\mathbf{A}^*\mathbf{A}=\mathbf{A}\mathbf{A}^*=\mathbf{I}_n$ et si $\mathbf{A}^*\mathbf{A}= \mathbf{A}^*\mathbf{A}$ est dite normale.

Par conséquent, une matrice unitaire est telle que $\mathbf{A}^{-1}=\mathbf{A}^*$. Naturellement, une matrice unitaire est également normale, mais elle n'est en général pas hermitienne.

La trace d'une matrice carrée **A** d'ordre *n* est la somme des coefficients diagonaux de **A** :

$$\mathrm{tr}\left(\mathbf{A}\right) = \sum_{i=1}^{n} a_{ii}$$

Le déterminant de la matrice carrée **A** le scalaire défini par la formule :

$$\det\left(\mathbf{A}\right) = \sum_{\boldsymbol{\pi}\in P} \mathrm{signe}\left(\boldsymbol{\pi}\right) a_{1\pi_1} a_{2\pi_2} \cdots a_{n\pi_n}$$

où, $P = \left\{\boldsymbol{\pi} = \left(\pi_1,\ \pi_2,\ \ldots,\ \pi_n\right)^T\right\}$ est l'ensemble des *n*! multi-indices obtenus par permutation du multi-indice $\mathbf{i} = \left(1,\ 2,\ \ldots,\ n\right)^T$ et $\mathrm{signe}\left(\boldsymbol{\pi}\right)$ vaut 1 (respectivement -1) si on effectue un nombre pair (respectivement impair) de transpositions pour obtenir $\boldsymbol{\pi}$ à partir de **i**.
Si **A** est une matrice carrée d'ordre *n*, on a les propriétés suivantes :

$$\det\left(\mathbf{A}\right) = \det\left(\mathbf{A}^T\right),\ \det\left(\mathbf{AB}\right) = \det\left(\mathbf{A}\right)\det\left(\mathbf{B}\right) \Rightarrow \det\left(\mathbf{A}^{-1}\right) = 1/\det\left(\mathbf{A}\right)$$

$$\det\left(\mathbf{A}^*\right) = \overline{\det\left(\mathbf{A}\right)},\ \det\left(\alpha\mathbf{A}\right) = \alpha^n \det\left(\mathbf{A}\right),\ \forall \alpha \in E$$

De plus, si deux lignes ou deux colonnes d'une matrice coïncident, le déterminant de cette matrice est nul. Quand on échange deux lignes (ou deux colonnes), on change le signe du déterminant. Enfin, le déterminant d'une matrice diagonale est le produit des éléments diagonaux.

Si on note $\mathbf{A}_{ij}$ la matrice d'ordre $n-1$ obtenue à partir de $\mathbf{A}$ en éliminant la i-ième ligne et la j-ième colonne, on appelle mineur associé au coefficient a_{ij} le déterminant de la matrice $\mathbf{A}_{ij}$. On appelle k-ième mineur principal de $\mathbf{A}$ le déterminant d_k de la sous-matrice principale d'ordre k, $\mathbf{A}_k = \mathbf{A}(1:k,\ 1:k)$.

Le cofacteur de a_{ij} est défini par $\Delta_{ij} = (-1)^{i+j} \det(\mathbf{A}_{ij})$, le calcul effectif du déterminant de $\mathbf{A}$ peut être effectué en utilisant la relation de récurrence :

$$\det(\mathbf{A}) = \begin{cases} a_{11} & \text{si } n = 1, \\ \sum_{j=1}^{n} \Delta_{ij} a_{ij} & \text{si } n > 1. \end{cases}$$

qui est connue sous le nom de la loi de Laplace. Si $\mathbf{A}$ est une matrice inversible d'ordre n, alors :

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \mathbf{C}$$

où $\mathbf{C}$ est la matrice de coefficients Δ_{ij}, $i,\ j = 1,\ 2,\ \ldots,\ n$. Par conséquent, une matrice carrée est inversible si et seulement si son déterminant est non nul. Dans le cas d'une matrice diagonale inversible, l'inverse est encore une matrice diagonale ayant pour éléments les inverses des éléments de la matrice.

2.2. Matrices et applications linéaires

Une application linéaire de $\mathbb{C}^n$ sur $\mathbb{C}^m$ est une fonction $f : \mathbb{C}^n \to \mathbb{C}^m$ telle que :

$$f(\alpha \mathbf{x} + \beta \mathbf{y}) = \alpha f(\mathbf{x}) + \beta f(\mathbf{y}),\ \forall \alpha,\ \beta \in E \text{ et } \forall \mathbf{x},\ \mathbf{y} \in \mathbb{C}^n$$

Si $f : \mathbb{C}^n \to \mathbb{C}^m$ est une application linéaire, alors il existe une unique matrice $\mathbf{A} \in \mathbb{C}^{m \times n}$ telle que :

$$f(\mathbf{x}) = \mathbf{A}\mathbf{x},\ \forall \mathbf{x} \in \mathbb{C}^n$$

Inversement, si $\mathbf{A} \in \mathbb{C}^{m \times n}$, la fonction $f(\mathbf{x})$ est une application linéaire de $\mathbb{C}^n$ sur $\mathbb{C}^m$.

2.3. Rang et noyau d'une matrice

Soit $\mathbf{A}$ une matrice rectangulaire de dimension $m \times n$. Le déterminant extrait d'ordre $q \geq 1$ est le déterminant de n'importe quelle matrice d'ordre q obtenue à partir de $\mathbf{A}$ en éliminant $m - q$ lignes et $n - q$ colonnes. Le rang de $\mathbf{A}$, noté $\mathrm{rg}(\mathbf{A})$, est l'ordre maximum des déterminants extraits non nuls de $\mathbf{A}$. Une matrice de rang maximum si $\mathrm{rg}(\mathbf{A}) = \min(m,\ n)$. Le rang de la matrice $\mathbf{A}$ représente le nombre maximum de vecteurs colonnes ou vecteurs lignes linéairement indépendants. Autrement dit, c'est la dimension de l'image de $\mathbf{A}$, qui est un sous-espace vectoriel définie par :

$$\mathrm{Im}(\mathbf{A}) = \left\{ \mathbf{y} \in \mathbb{R}^m \ : \ \mathbf{y} = \mathbf{A}\mathbf{x} \text{ pour } \mathbf{x} \in \mathbb{R}^n \right\}$$

Le noyan de la matrice **A** est le sous-espace vectoriel défini par :

$$\mathrm{Ker}(\mathbf{A}) = \left\{ \mathbf{x} \in \mathbb{R}^n \ : \ \mathbf{y} = \mathbf{A}\mathbf{x} = 0 \right\}$$

Soit $\mathbf{A} \in \mathbb{R}^{m\times n}$, on a les relations suivantes :

$$\mathrm{rg}(\mathbf{A}) = \mathrm{rg}\left(\mathbf{A}^T\right) \ \left(\text{si } \mathbf{A} \in \mathbb{C}^{m\times n},\ \mathrm{rg}(\mathbf{A}) = \mathrm{rg}\left(\mathbf{A}^T\right)\right)$$

$$\mathrm{rg}(\mathbf{A}) + \dim\left(\mathrm{Ker}(\mathbf{A})\right) = n$$

En général,

$$\dim\left(\mathrm{Ker}(\mathbf{A})\right) \neq \dim\left(\mathrm{Ker}\left(\mathbf{A}^T\right)\right)$$

Si **A** est une matrice carrée inversible, alors,

$$\mathrm{rg}(\mathbf{A}) = n \text{ et } \dim\left(\mathrm{Ker}(\mathbf{A})\right) = 0$$

On note enfin que pour une matrice $\mathbf{A} \in \mathbb{C}^{n\times n}$ est inversible alors $\det(\mathbf{A}) \neq 0$, $\mathrm{Ker}(\mathbf{A}) = \{\mathbf{0}\}$ $\mathrm{rg}(\mathbf{A}) = n$ et les colonnes et les lignes de **A** sont linéairement indépendantes.

2.4. Valeurs propres et vecteurs propres

Soit **A** une matrice carrée d'ordre *n*. à valeurs réelles ou complexes, on dit que $\lambda \in \mathbb{C}$ est une valeur propre de **A** s'il existe un vecteur non nul $\mathbf{x} \in \mathbb{C}^n$ tel que $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$. **x** est le vecteur propre associé à la valeur propre λ et l'ensemble des valeurs propres de **A** est appelé spectre de **A**.

La valeur propre λ correspondant au vecteur propre **x** est la solution de l'équation caractéristique :

$$P_{\mathbf{A}}(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}_n) = 0$$

$P_{\mathbf{A}}(\lambda)$ est le polynôme caractéristique. Ce polynôme étant de degré *n* par rapport à λ, par conséquent et suivant le théorème fondamentale de l'algèbre, il existe *n* valeurs propres non nécessairement distinctes. On peut démontrer la propriété suivante :

$$\det(\mathbf{A}) = \lambda_1\lambda_2\cdots\lambda_n, \quad \mathrm{rg}(\mathbf{A}) = \sum_{i=1}^{n}\lambda_i$$

Le plus grand des modules des valeurs propres de **A** est appelé rayon spectral de **A** :

$$\rho(\mathbf{A}) = \max\left(|\lambda_1|,\ |\lambda_2|,\ \ldots,\ |\lambda_n|\right)$$

2.5. Produit scalaires et normes vectoriels

Pour quantifier des erreurs ou mesurer des distances, de calculer la "grandeur" d'un vecteur ou d'une matrice. Pour cela, soit la notion de norme vectorielle et celle de norme matricielle. Un produit scalaire sur un *E*-espace vectoriel *V* est une application de $V\times V$ sur *E* qui possèdent les propriétés suivante :

1. Elle est linéaire par rapport aux vecteurs de *V*, c'est-à-dire.
2. Elle est hermitienne.
3. Elle est définie positive.

Soit $\mathbf{v}_1$ et $\mathbf{v}_2$ deux vecteurs de *E* d'ordre *n* :

$$\mathbf{v}_1 = \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix}, \quad \mathbf{v}_2 = \begin{pmatrix} b_1 & b_2 & \dots & b_n \end{pmatrix}^T$$

Le produit scalaire est un produit matricielle qui s'effectue entre le vecteur ligne $\mathbf{v}_1$ et un vecteur colonne $\mathbf{v}_2$, il est exprimé par :

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = \sum_{i=1}^{n} a_i b_i$$

Remarquons que le produit $\mathbf{v}_2 \cdot \mathbf{v}_1$ conduit la production de la matrice suivante :

$$\mathbf{v}_2 \cdot \mathbf{v}_1 = \begin{pmatrix} b_1 & b_2 & \dots & b_n \end{pmatrix}^T \cdot \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix} = \begin{pmatrix} b_1 a_1 & b_1 a_2 & \cdots & b_1 a_n \\ b_2 a_1 & b_2 a_2 & \cdots & b_2 a_n \\ \vdots & \vdots & \cdots & \vdots \\ b_n a_1 & b_n a_2 & \cdots & b_n a_n \end{pmatrix}$$

Soit *V* un *E*-espace vectoriel. On dit qu'une application $\| \cdot \|$ de *V* dans $\mathbb{R}$ est une norme sur *V* si :

1. $\|\mathbf{v}\| \geq 0, \ \forall \mathbf{v} \in V$ et $\|\mathbf{v}\| = 0$ si et seulement si $\mathbf{v} = \mathbf{0}$;
2. $\|\alpha \mathbf{v}\| = |\alpha| \|\mathbf{v}\| \ \ \forall \mathbf{v} \in V, \ \forall \alpha \in E$ (propriété d'homogénéité) ;
3. $\|\mathbf{v} + \mathbf{w}\| \leq \|\mathbf{v}\| + \|\mathbf{w}\| \ \ \forall \mathbf{v}, \ \mathbf{w} \in V$ (inégalité triangulaire)

où $|\alpha|$ désigne la valeur absolue (resp. le module) de α si $E = \mathbb{R}$ (resp. $E = \mathbb{C}$).

On appelle espace vectoriel normé le couple $(V, \| \cdot \|)$. Un vecteur de *V* dont la norme 1 est dit un vecteur unitaire.
On défit la *p*-norme (ou norme de Hölder) d'un vecteur $\mathbf{v} = (v_1, v_2, \dots, v_n)$ par :

$$\|\mathbf{v}\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p} \text{ pour } 1 \leq p < \infty$$

La limite $\|\mathbf{v}\|_p$ Quand *p* tend vers l'infini existe, est finie et égale au maximum des modules des composantes de $\mathbf{v}$. Cette limite définit à son tour une norme, appelée norme infinie (ou norme du maximum), donnée par :

$$\|\mathbf{v}\|_\infty = \max_{1 \leq i \leq n} |v_i|$$

Quand on prend *p* = 2, on retrouve la définition classique de la norme euclidienne :

$$\|\mathbf{v}\|_2 = \left(\sum_{i=1}^{n} |v_i|^2 \right)^{1/2} = \left(\mathbf{v} \cdot \mathbf{v}^T \right)^{1/2}$$

qui possède la propriété, pour tout vecteur $\mathbf{v}, \ \mathbf{w} \in \mathbb{R}^n$,

$$\left| \mathbf{v} \cdot \mathbf{w}^T \right| \leq \|\mathbf{v}\|_2 \|\mathbf{w}\|_2$$

où l'égalité est très stricte si et seulement si $\mathbf{w} = \alpha \mathbf{v}$ pour $\alpha \in \mathbb{R}$.

Rappelons que l'inégalité de Hölder fournit une relation entre le produit scalaire dans $\mathbb{R}^n$ et les *p*-normes :

$$\left|\mathbf{v}\cdot\mathbf{w}^T\right| \le \|\mathbf{v}\|_p \|\mathbf{w}\|_q \quad \text{avec } \frac{1}{p}+\frac{1}{q}=1$$

2.6. Normes matricielles

Une norme matricielle est une application $\|\cdot\|$ de $\mathbb{R}^{m\times n}$ dans $\mathbb{R}$ telle que :

1. $\|\mathbf{A}\| \ge 0,\ \forall \mathbf{A}\in\mathbb{R}^{m\times n}$ et $\|\mathbf{A}\|=0$ si et seulement si $\mathbf{A} = \mathbf{0}$;
2. $\|\alpha\mathbf{A}\| = |\alpha|\|\mathbf{A}\|\ \ \forall \mathbf{A}\in\mathbb{R}^{m\times n},\ \forall\alpha\in\mathbb{R}$ (propriété d'homogénéité) ;
3. $\|\mathbf{A}+\mathbf{B}\| \le \|\mathbf{A}\|+\|\mathbf{B}\|\ \ \forall \mathbf{A},\ \mathbf{B}\in\mathbb{R}^{m\times n}$ (inégalité triangulaire)

On dit qu'une norme matricielle est compatible ou consistante avec une norme vectorielle si :

$$\|\mathbf{Ax}\| \le \|\mathbf{A}\|\ \|\mathbf{x}\|, \quad \forall \mathbf{x}\in\mathbb{R}^n$$

Plus généralement, on dit que trois normes, toutes notées $\|\cdot\|$ et respectivement définies sur $\mathbb{R}^m$, $\mathbb{R}^n$ et $\mathbb{R}^{m\times n}$, sont consistantes si $\forall \mathbf{x}\in\mathbb{R}^n$, $\forall \mathbf{y}\in\mathbb{R}^m$ et $\mathbf{A}\in\mathbb{R}^{n\times m}$ tels que $\mathbf{Ax}=\mathbf{y}$, on a $\|\mathbf{y}\| \le \|\mathbf{A}\|\ \|\mathbf{x}\|$.

La norme,

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i,j=1}^{n} |a_{ij}|^2} = \sqrt{\mathrm{tr}\left(\mathbf{AA}^*\right)}$$

est une norme matricielle appelée norme de Frobenius et elle est compatible avec la norme vectorielle euclidienne $\|\cdot\|_2$. En effet,

$$\|\mathbf{Ax}\|_2^2 = \sum_{i=1}^{n}\left|\sum_{j=1}^{n} a_{ij}x_j\right|^2 \le \sum_{i=1}^{n}\left(\sum_{j=1}^{n}|a_{ij}|^2\sum_{j=1}^{n}|x_j|^2\right) = \|\mathbf{A}\|_F^2\,\|\mathbf{x}\|_2^2$$

Remarquer que pour cette norme $\|\mathbf{I}_n\|_F = \sqrt{n}$.

Annexe 2.

L'essentiel du MATLAB

MATLAB® (Fig. A2.1) est un système interactif de programmation scientifique, pour le calcul numérique et la visualisation graphique, basé sur la représentation matricielle des données.

La plate-forme MATLAB est optimisée pour résoudre les problèmes scientifiques et techniques. Le langage utilisé dans MATLAB est basé sur les matrices, est le moyen le plus naturel au monde pour exprimer les mathématiques computationnelles. Les graphiques intégrés permettent de visualiser facilement les données afin d'en dégager des informations. Grâce à la vaste bibliothèque de boîtes à outils prédéfinies.

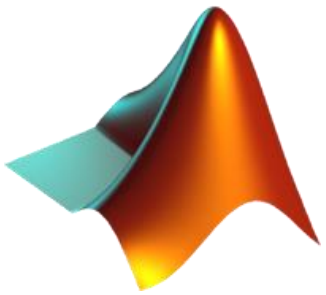


Fig. A2.1 : Logo de MATLAB®.

MATLAB trouve ses applications dans de nombreuses disciplines. Il constitue un outil numérique puissant pour la modélisation de systèmes physiques, la simulation de modèles mathématiques, la conception et la validation (tests en simulation et expérimentation) d'applications. Le logiciel de base est complété par de multiples toolboxes, c'est-à-dire des boîtes à outils. Celles-ci sont des bibliothèques de fonctions dédiées à des domaines particuliers. Par exemple, génie civil, hydraulique, automatique, le traitement du signal l'analyse statistique, l'optimisation . . .

1. Environnement de MATLAB

MATLAB (Yahiaoui, 2017) propose un véritable environnement de travail (Fig. A2.2) composé de multiples fenêtres.

- **Command Window** (console d'exécution) : à l' « invite » de commande le signe prompt "**>>**", l'utilisateur peut entrer les instructions à exécuter. Il s'agit de la fenêtre principale de l'environnement MATLAB.
- **Current Folder** (répertoire courant) : permet de naviguer et de visualiser le contenu du répertoire courant de l'utilisateur. Les programmes de l'utilisateur doivent être situés dans ce répertoire pour être visible et donc exécutable.
- **Workspace** (espace de travail) : permet de visualiser les variables définies, leur type, la taille occupée en mémoire…
- **Command History** : historique des commandes que l'utilisateur a exécutées. Il est possible de faire glisser ces commandes vers la fenêtre de commande.

La fenêtre "Command Window" existe dans le centre de l'interface, c'est à partir de là que l'utilisateur pourra lancer les commandes et fonctions interprétées par MATLAB. Le principe est simple et intuitif, le tout est de connaitre les fonctions appropriées et de respecter leur syntaxe. Premier exemple élémentaire : à l'invite de commande, taper « **5*9** », puis entrer :

```
>> 5*9
ans =
```

```
    45
```

A la validation de l'instruction, l'interface affiche le résultat de cette dernière. Afin d'alléger l'affichage, un point-virgule "**;**" en fin de commande empêche le renvoie du résultat dans la fenêtre (évidemment l'instruction est toujours exécutée). Par exemple :

```
>> 5*9;
>>
```

Le calcul a été effectué mais le résultat n'est pas affiché.

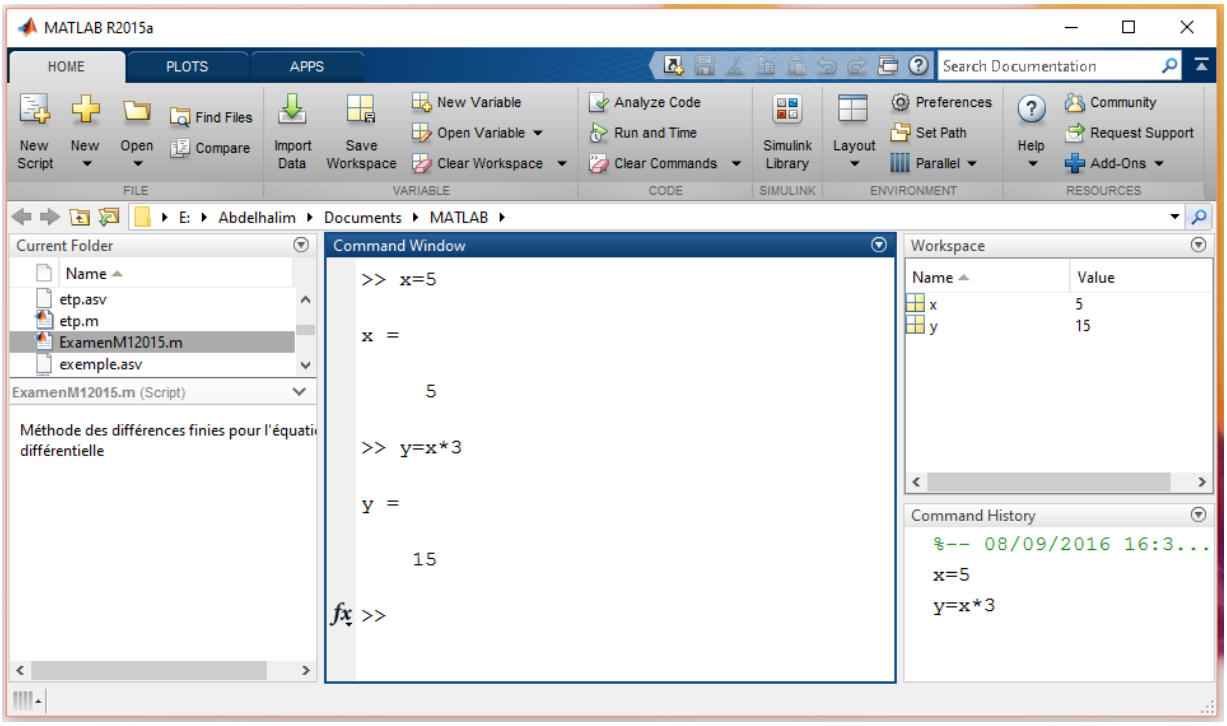


Fig. A2.2 : Environnement de MATLAB.

Soit en premier lieu un démarrage rapide basé sur un exemple très simple. Celui-ci a pour objectif de montrer comment se pratique MATLAB dans une première utilisation basique. Cette section doit donc être parcourue avec MATLAB à côté. La manipulation commence dans la fenêtre «Command Window », à l'invite de commande :

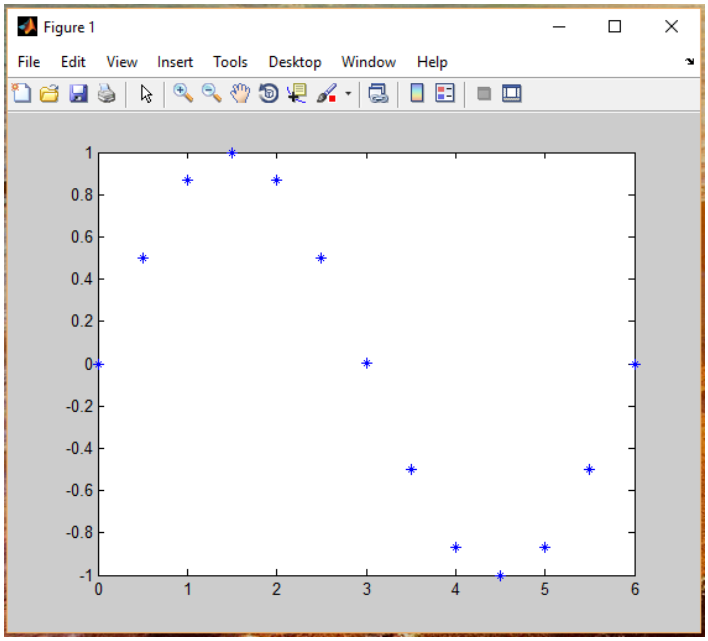


Fig. A2.3 : Représentation graphique dans MATLAB.

```
>> V = [0 0.5 1 1.5 2 2.5 3 3.5
4 4.5 5 5.5 6]
V =
  Columns 1 through 7
    1.0000    0.5000    1.0000
1.5000   2.0000   2.5000   3.0000
  Columns 8 through 13
    3.5000    4.0000    4.5000
5.0000   7.5000   6.0000
```

Cette ligne de commande définit un

tableau de 13 valeurs (allant de 0 à 6 par incrément de 0.5) nommé **V**. On comprend l'utilité de mettre un "**;**" à la fin de la ligne pour éviter l'affichage systématique du résultat des opérations envoyées. Cela peut devenir très lourd si, par exemple, la taille du tableau est importante.

```
>> w=2*pi/6;
>> y=sin(w*V);
```

Le terme **pi** est une constante prédéfinie et donne la valeur de π. Le tableau (ou vecteur) **y**, de même dimension que **V**, contient les valeurs résultantes de l'opération appliquée à chaque composante de **V**. **y**, représente donc la fonction sinus de période 6 (de pulsation **w**). Il est possible de représenter graphiquement les points du tableau à l'aide de la fonction suivante :

```
>> plot(V,y,'*');
```

Une nouvelle fenêtre s'ouvre (Fig. A2.3).

2. Opérateurs dans MATLAB

Dans MATLAB, le signe égale "**=**" signifie l'opérateur d'affectation d'un constante ou une chaine de caractère à une variable, à titre d'exemple, l'écriture de la commande :

```
>> x=5
```

Conduit après l'exécution de la touche entrée au résultat :

```
x =
     5
```

qui signifie que la valeur **5** est affectée à la variable **x**, ou la variable **x** prend pour l'instant la valeur **5**.

Pour distinguer la différence entre l'affectation et l'égalité, soit l'exemple suivant :

```
>> x+7=50
```

La réponse du MATLAB après l'exécution de cette commande est :

```
??? x+7=50
         |
Error: The expression to the left of the equals sign is not a valid
target for an assignment.
```

Ce message d'erreur montre bien que l'opérateur d'affectation ne peut être pas utilisé comme une signe d'égalité, malgré que la commande "**x+7=50**" est équation algébrique et possède une solution **x=50-7=43**.

L'opérateur d'affectation est utilisé d'une manière récursive comme dans l'exemple suivant :

```
>> x=5
x =
    5
>> x=x+3
x =
     8
>> x=2*x*x
x =
    128
```

Dans MATLAB, certains opérateurs mathématiques sont très utile, sachant que les fameuses opérations mathématiques utilisées dans les calculs numériques telles que l'addition "**+**", la soustraction "**-**", la multiplication "*****", la division "**/**" et l'opération puissance "**^**", il y a aussi d'autres opérations mathématiques sont propres a MATLAB sont utilisée dans certain cas de manipulation des scalaires et tableau, telle que la division inverse "****" et le reste d'une division euclidienne "**mod**"et les opérations liées aux calculs symboliques…

A titre d'exemple, soit le volume *V* d'une sphère de diamètre *D* est exprimé par :

$$V = \frac{4}{3}\pi\left(\frac{D}{2}\right)^3$$

Dans cette expression il y a le constant π qui est prédéfinie en MATLAB par le terme "**pi**". Les deux commandes suivantes permettant le calcul de ce volume à partir d'un diamètre $D = 4$:

```
>> D = 4;
>> V = (4/3)*pi*(D/2)^3
V =
   33.5103
```

Dans cet exemple, il y a trois opérations principales à savoir, le produit, la division et la puissance.

Un autre nombre est très souvent utilisé, c'est le nombre $e \approx 2.718$, qui peut être obtenue à partir de la commande :

```
>> exp(1)
ans =
    2.7183
```

exp(a)

Est la fonction exponentielle e^a dont l'argument est *a*. Pour calculer e^2 soit :

```
>> exp(2)
ans =
    7.3891
```

Pour calculer la racine carrée d'un nombre (par exemple 16) avec MATLAB, soit la commande :

```
>> x= sqrt(16)
x =
    4
```

Aussi, pour calculer le logarithme naturel d'un nombre positif (par exemple 3.2) avec MATLAB, soit la commande :

```
>> y= log(3.2)
y =
    1.6094
```

Le logarithme décimal du même nombre se fait comme suit :

```
>> z= log10(3.2)
z =
    0.5051
```

Les fonctions trigonométriques de base **sin**, **cos** et **tan** et leurs inverses **asin**, **acos** et **atan** sont utilisées pour un argument en radian par défaut, soit les exemples suivants :

```
>> cos(pi/4)
ans =
    0.7071
>> sin(pi/3)
ans =
    0.8660
>> tan(pi/3)
ans =
    1.7321
>> atan(pi/3)
ans =
    0.8084
>> format rat
>> atan(pi/3)
ans =
    1110/1373
```

Ce sont quelques opérateurs de bases qui existent en MATLAB, une large bibliothèque qui contient de multitude de fonctions et opérateurs, qui peuvent être cherché et voir leurs utilités dans l'aide de MATLAB.

3. Vecteurs et matrices dans MATLAB

Un vecteur ou une matrice dans MATLAB, sont des tableaux de série de données qui peuvent être rangées dans une seule ligne (vecteur ligne), dans une seule colonne (vecteur colonne) ou dans des lignes et colonnes pour le cas d'une matrice (tableau).

Plusieurs méthodes sont utilisées dans la formation des tableaux, soit par exemple les deux vecteurs **V** et **W** suivants :

```
>> V=[1 ;2 ;3]
V =
     1
     2
     3
>> W=[2 3 1]
W =
     2     3     1
```

V est un vecteur colonne, c'est un tableau composé de trois lignes et une colonne, c'est-à-dire de dimension 3×1. **W** est un vecteur ligne, c'est un tableau composé d'une ligne et trois colonnes, c'est-à-dire de dimension 1×3. La commande **whos** donne :

```
>> whos
Name        Size              Bytes  Class     Attributes
  V         3x1                  24  double
  W         1x3                  24  double
```

Remarque.

Dans MATLAB, vecteur ou matrice se sont des tableaux de différents dimensions, voir même un scalaire est aussi un tableau de dimension 1×1.

Parmi les vecteurs utilisés dans MATLAB, sont de nombres et qui suit certain suite numérique bien particulière. Soit la suite $u_n = \sqrt{n}$ ou $n = 0,\ 1,\ 2,\ ...,\ 10$, dans cette suite il y a deux vecteurs : le vecteur $n = (1\ \ 2\ \ 3\ \ ...\ \ 10)$ et le vecteur $Un = (u_1\ \ u_2\ \ u_3\ \ ...\ \ u_{10})$. Ces deux vecteurs peuvent être formés comme suit :

```
>> n=0:10
n =
     0     1     2     3     4     5     6     7     8     9    10
>> Un=sqrt(n)
Un =
  Columns 1 through 6
         0    1.0000    1.4142    1.7321    2.0000    2.2361
  Columns 7 through 11
    2.4495    2.6458    2.8284    3.0000    3.1623
```

n et **Un**, sont deux tableaux de dimension 1×11. Le vecteur n a été forme à partir d'une simple commande qui est formé par une succession de valeurs, la structure générale de ces vecteurs est :

V=vd:pas:vf

vd est la valeur début de la série
vf est la valeur finale de la série
pas est le pas (step size)

vd, **vf** et **pas** sont des nombres réels. Mais la formation de ce type de vecteur nécessite une certaine logique, soit l'exemple suivant :

```
>> X=1.5:-0.1:2
X =
   Empty matrix: 1-by-0
```

La commande **whos** donne :

```
>> whos
  Name       Size              Bytes  Class     Attributes
  X          1x0                   0  double
```

X est un tableau vide, n'a aucun sens ici.

Aussi, une autre manière de générer une suite numérique de raison fixe sous forme un vecteur est très intéressant avec la commande **linspace** qui signifie l'espacement linéaire comme suit :

V=linspace(vd,vf,n)

vd et **vf** sont les bornes amont et aval de l'intervalle, **n** est le nombre des valeurs intérimaires (dans certains cas n'est pas forcément être un entier), soit les deux exemples suivants :

```
>> V=linspace(3,9,7)
V =
     3     4     5     6     7     8     9
>> W=linspace(9,3,7)
W =
     9     8     7     6     5     4     3
```

Dans la commande **linspace(vd,vf)**, **n**'est prise par défaut égale à 100, c'est-à-dire,

linspace(vd,vf)= linspace(vd,vf,100)

Il est possible de créer un vecteur à partir d'autres vecteurs, soit **U** et **V** deux vecteurs colonnes :

```
>> U=[3; 6; 7; 9];
>> V=[1; 2; 5];
>> W=[U;V]
W =
     3
     6
     7
     9
     1
     2
     5
```

De même si **U** et **V** deux vecteurs lignes :

```
>> U=[3, 6, 7, 9];
>> V=[1, 2, 5];
>> W=[U, V]
W =
     3     6     7     9     1     2     5
```

Comme dans les vecteurs, une matrice dans MATLAB peut être construite facilement comme peut être importés directement par Copier–Coller à partir d'un autre éditeur de tableaux comme Excel, PlanMaker, Bloc-notes, etc… Soit, l'exemple suivant qui permet de construire un tableau de dimension 3×3 :

```
>> u=[1, 2, 3; 4, 5, 6; 7, 8, 9]
u =
     1     2     3
     4     5     6
     7     8     9
```

Ou bien,

```
>> u=[1 2 3; 4 5 6; 7 8 9]
u =
     1     2     3
     4     5     6
     7     8     9
```

Pour importer un tableau depuis un autre éditeur (PlanMaker par exemple), il faut ouvrir avec "**[**"et le coller, par la suite il faut aussi fermer avec "**]**" (Fig. A2.4).

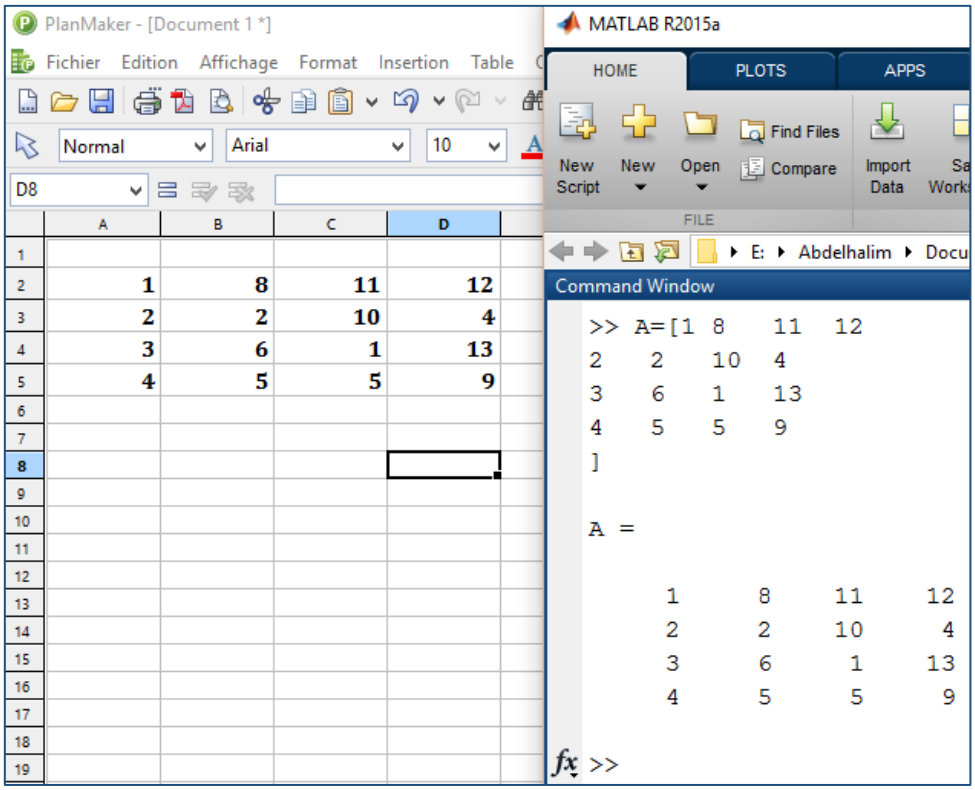


Fig. A2.4 : Importation d'un tableau depuis un autre éditeur de tableaux.

3.1. Opérateurs et opérations sur les tableaux

Plusieurs commandes peuvent être utilisée pour manipuler les vecteurs, par exemple la commande **numel(nom_du_vecteur)** donne le nombre des valeurs qui existe dans un vecteur ou dans un tableau, la commande **sort(nom_du_vecteur)** permet de classer les valeurs d'un vecteur par ordre croissant, la fonction commande **exp(nom_du_vecteur)** permet de donner un autre vecteur dont les valeurs sont les exponentielles des valeurs etc… voici quelques exemples :

```
>> x=[1, 0, 2, 3, 2.5];
>> y=exp(x)
y =
       2.7183    1.0000    7.3891   20.0855   12.1825
>> z=log(x)
z =
         0      -Inf    0.6931    1.0986    0.9163
>> w=sort(x)
w =
         0    1.0000    2.0000    2.5000    3.0000
>> n=numel(w)
n =
     5
```

Dans certain cas de fonctions par exemple avoir un vecteur dont les composantes sont les puissances des composantes du vecteur d'origine, par exemple soit le vecteur :

```
>> u = [4:-0.9:1]
u =
    4.0000    3.1000    2.2000    1.3000
```

Le vecteur **v=u^2** qui est normalement est :

```
v =
   16.0000    9.6100    4.8400    1.6900
```

Alors que MATLAB vous donne la réponse suivante :

```
>> v=u^2
??? Error using ==> mpower
Inputs must be a scalar and a square matrix.
```

Pour effectuer l'opération voulue, il faut utiliser pour les opérations du produit (*****), de division (**/**), de division inverse (****) et de puissance (**^**) les opérations avancé avec un point (dot), comme (**.***) pour le produit, (**./**) pour la division, (**.**) pour la division inverse et (**.^**) pour la puissance. Ces opérations s'effectuent sur les vecteurs ou tableaux de même dimension membre à membre. Pour obtenir le vecteur **v** dont les composantes sont les puissances des composantes du vecteur **u** un par un se fait comme suit :

```
>> v=u.^2
v =
   16.0000    9.6100    4.8400    1.6900
```

Le transposé d'un vecteur est une méthode très importante, elle est tout simplement rend un vecteur ligne en un vecteur colonne et vice versa, soit l'exemple :

```
v=[1; 3; 5]
v =
     1
     3
     5
>> w=v'
w =
     1     3     5
```

Les opérations additions et soustraction peuvent être faites entre les vecteurs de même dimension, par contre le produit entre les vecteurs s'effectue sur la base du nombre de colonnes dans le premier vecteur égale au nombre de lignes dans le second vecteur.

```
>> v=[1; 3; 5]
v =
     1
     3
     5
>> w=[7,6,1]
w =
     7     6     1
>> v*w
ans =
     7     6     1
    21    18     3
    35    30     5
```

```
>> w*v
ans =
    30
```

Les opérations de multiplication "**.***", de division "**./**" ou de division inverse "**.\'**' s'effectue entre les vecteurs ou tableaux de même dimensions membre à membre, par exemple :

```
>> v=[2, 4, 6]; u=[4, 6, 8];
>> v.*u
ans =
     8    24    48
>> u./v
ans =
    2.0000    1.5000    1.3333
>> u.\v
ans =
     0.5000    0.6667    0.7500
```

Chaque composante dans un vecteur peut être référenciée facilement suivant sa position soit l'exemple suivant :

```
>> W =[3, 6, 7, 9, 1, 2, 5]
W =
     3     6     7     9     1     2     5
>> W(1,6)
ans =
     2
```

Il est possible dans MATLAB de faire l'addition d'un vecteur ou un tableau avec un scalaire, soit l'exemple suivant :

```
>> W =[3, 6, 7, 9, 1, 2, 5];
>> 1+W
ans =
     4     7     8    10     2     3     6
```

Tous les opérateurs et opérations applicables aux vecteurs sont applicables aussi pour les matrices ou tableaux. Certains opérateurs sont utilisés spécialement pour les tableaux à savoir : le rang, le déterminant, l'inverse, les valeurs et vecteurs propres d'une matrice carrée etc. par exemple :

```
>> A=[1 8 11 12; 2 2 10 4; 3 6 1 13; 4 5 5 9]
A =
     1     8    11    12
     2     2    10     4
     3     6     1    13
     4     5     5     9
>> Determinant2A=det(A)
Determinant2A =
 -702.0000
>> Rang2A=rank(A)
Rang2A =
     4
>> Inverse2A=inv(A)
```

```
Inverse2A =
   -0.1396   -0.0385   -0.1453    0.4131
    0.2593   -0.5000   -0.4444    0.5185
    0.0114    0.1154   -0.0085   -0.0541
   -0.0883    0.2308    0.3162   -0.3305
```

Ces opérateurs sont très utiles, surtout dans le cas de la résolution des systèmes d'équations linéaires. En algèbre, si le déterminant de la matrice **A** est différent de zéro alors le système $\mathbf{Ax} = \mathbf{b}$ possède une solution comme suit :

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

A titre d'exemple, soit le système d'équations linéaires suivante :

$$\begin{cases} 3x+2y-9z=-65, \\ -9x+5y+2z=16, \\ 6x+7y+3z=5. \end{cases} \text{ ou bien, } \begin{pmatrix} 3 & 2 & -9 \\ -9 & 5 & 2 \\ 6 & 7 & 3 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -65 \\ 16 \\ 5 \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 & 2 & -9 \\ -9 & 5 & 2 \\ 6 & 7 & 3 \end{pmatrix}^{-1}\begin{pmatrix} -65 \\ 16 \\ 5 \end{pmatrix}$$

Qui peut être résolue avec MATLAB comme suit :

```
>> A=[3 2 -9; -9 5 2; 6 7 3]; % Introduction de la matrice A
>> b=[-65; 16; 5]; % Introduction du vecteur colonne b
>> x=inv(A)*b  % Résolution
x =
   -1.0065
   -1.2549
    6.6078
```

Donc la solution de ce système est : $x = -1.0065$, $y = -1.2549$ et $z = 6.06078$

3.2. Norme d'un tableau

Soit **A** une matrice ou un tableau de dimension $n \times n$, c'est-à-dire :

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

Les normes usuelles d'une matrice les plus utilisés sont :

$$\|\mathbf{A}\|_1 = \max\left(\sum_{i=1}^{n}|a_{ij}|\right) \text{ pour } j = 1,\ 2,\ 3,\ \ldots,\ n$$

$$\|\mathbf{A}\|_\infty = \max\left(\sum_{j=1}^{n}|a_{ij}|\right) \text{ pour } i = 1,\ 2,\ 3,\ \ldots,\ n$$

$$\|\mathbf{A}\|_E = \left(\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}^2\right)^{1/2} \text{ (Norme Euclidienne)}$$

La norme d'un tableau **A** peut être calculée dans MATLAB avec la commande **`norm(A,p)`**, ou **`p`** est un paramètre,

$\|\mathbf{A}\|_1 \equiv$ **`norm(A,1)`**

$\|\mathbf{A}\|_\infty \equiv$ **`norm(A,inf)`**

$\|\mathbf{A}\|_f \equiv$ **`norm(A,'fro')`** norme de Frobenius

soit l'exemple :

```
>> A=[3 2 -9; -9 5 2; 6 7 3]
A =
     3     2    -9
    -9     5     2
     6     7     3
>> norm(A,1)
ans =
    18
>> norm(A,inf)
ans =
    16
>> norm(A,'fro')
ans =
   17.2627
```

3.3. Valeurs et vecteurs propres

En algèbre linéaire, chaque colonne (resp. ligne) est un vecteur dont les éléments sont les composantes par rapport à la base canonique, il est possible de trouver une autre base de façon que la matrice soit purement diagonale.

Si **A** une matrice carrée, les vecteurs propres et les valeurs propres de **A**, se fait à l'aide de la fonction **`eig`**, par exemple :

```
>> A=[2 1 -1;3 2 -3;3 1 -2]
A =
     2     1    -1
     3     2    -3
     3     1    -2
>> [V,D]=eig(A)
V =
         0   -0.5774    0.7071
   -0.7071   -0.5774   -0.0000
   -0.7071   -0.5774    0.7071
D =
   -1.0000         0         0
         0    2.0000         0
         0         0    1.0000
```

Les colonnes du tableau **`V`** sont les vecteurs propres normalisés **`V(:,1)`**, **`V(:,2)`** et **`V(:,3)`** et la diagonale du tableau **`D`** est formé par les valeurs propres **`D(1,1)`**, **`D(2,2)`** et **`D(3,3)`**.

4. Représentation graphique dans MATLAB

La représentation graphique est l'une des outils les plus importants en MATLAB que ce soit dans le plan (2D) ou dans l'espace (3D).

4.1. La fonction `plot`

L'ensemble des couples $(x_i,\ y_i)\ \forall i = 1,\ 2,\ 3,\ \ldots,\ n$ peut être considérés dans MATLAB comme deux vecteurs **x** et **y** :

```
>> x=[x1 x2 x3 . . . xn];
>> y=[y1 y2 y3 . . . yn];
```

Pour tracer le graph formé par le nuage du points $(x_i,\ y_i)\ \forall i = 1,\ 2,\ 3,\ \ldots,\ n$ avec MATLAB, il suffit d'appliquer la fonction **plot(x,y)**. Soit l'exemple :

```
>> x=[1:0.5:2*pi];
>> y=sin(x);
>> plot(x,y)
```

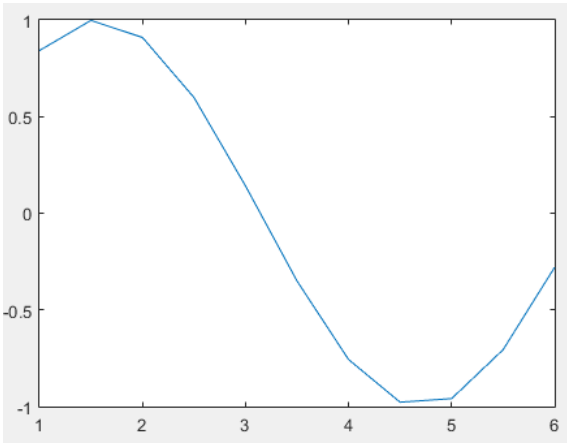


Fig. A2.5 : Utilisation de la fonction **plot**.

La fonction **plot**, possède les deux arguments **x** et **y**, une fois exécutée, MATLAB affiche une fenêtre de figure, (Fig. A2.5) qui contient le graphique tracé par défaut en bleu reliant les points par des bissectrices en parcourant l'axe des abscisses. Cette fenêtre est un éditeur de figure ou le graph obtenu peut être personnalisé, sauvegardé etc…

La commande **figure**, permet l'affichage d'une fenêtre figure vide prête à recevoir un nouveau graphe si la fonction **plot** est exécuté une autre fois. Pour comprendre mieux, soit le vecteur :

```
>> x=pi*(0:1/10:2);
```

et les vecteurs **y1** et **y2** :

```
>> y1=sin(x) ; y2=cos(x);
```

Pour tracer des deux fonctions **y1=sin(x)** et **y2=cos(x)** dans deux figures différentes.

```
>> plot(x,y1) % Figure 1 pour tracer y1=sin(x)
>> figure     % Création d'une seconde fenêtre Figure 2
>> plot(x,y2) % Figure 2 pour tracer y2=cos(x)
```

Il est possible de tracer deux ou plus de graphes dans le même système de coordonnées (Fig. A2.7). Pour l'exemple précédent :

```
>> plot(x,y1,x,y2)
```

Il est possible d'avoir le même résultat avec la commande **hold on** comme suit :

```
>> plot(x,y1)
>> hold on
>> plot(x,y2)
```

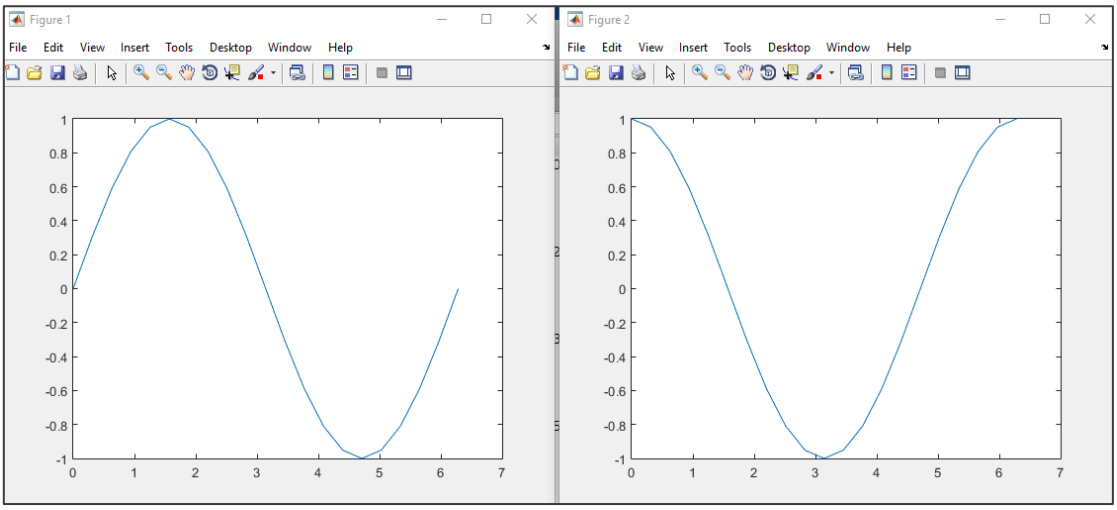


Fig. A2.6 : Affichage des graphes dans des fenêtres séparées.

La fonction **`plot`** peut être utilisée avec d'autres arguments pour personnaliser les graphes comme suit (Fig. A2.8) :

```
>> x1=0:0.1:7;
>> y1=sin(x1);
>> x2=0:0.2:8;
>> y2=cos(x2);
>> plot(x1,y,'*r',x2,y2,'o-b')
```

Il est très important de signaler que la fonction **`plot`** ne marche pas si les deux vecteurs **`x`** et **`y`** n'ont pas la même dimension, par exemple :

```
>> x=1:0.5:10;
>> numel(x)
ans =
      19
```

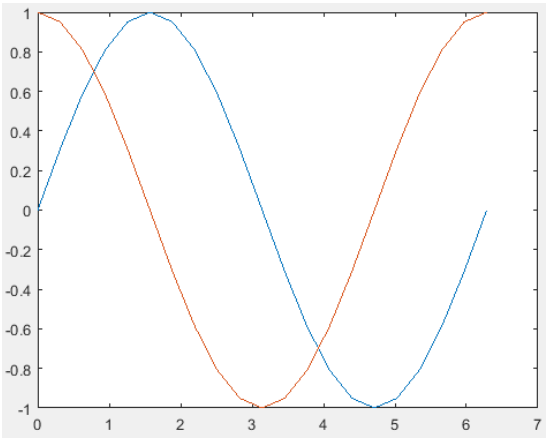


Fig. A2.7 : Graphes dans une seule figure.

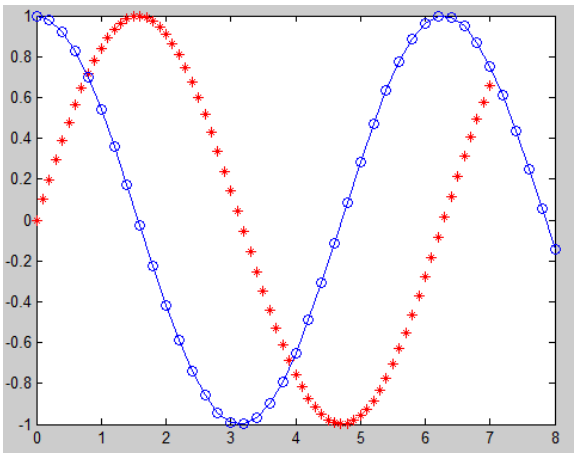


Fig. A2.8 : Graphes personnalisés.

```
>> y=2:0.1:15;
>> numel(y)
ans =
   131
>> plot(x,y)
Error using plot
Vectors must be the same length.
```

4.2. La fonction `fplot`

Le tracer d'une fonction $f(x)$ définie sur un intervalle se fait avec la fonction **`fplot`**. Soit à titre d'exemple la fonction $f(x)=e^{-2x}\sin 2x$ défini dans l'intervalle [0, 5], c'est-à-dire :

```
>> y = 'exp(-2*x)*sin(2*x)'
y =
exp(-2*x)*sin(2*x)
```

y est une chaine de caractères, le tracé de cette fonction (Fig. A2.9) avec **`fplot`** se fait suivant la syntaxe suivante :

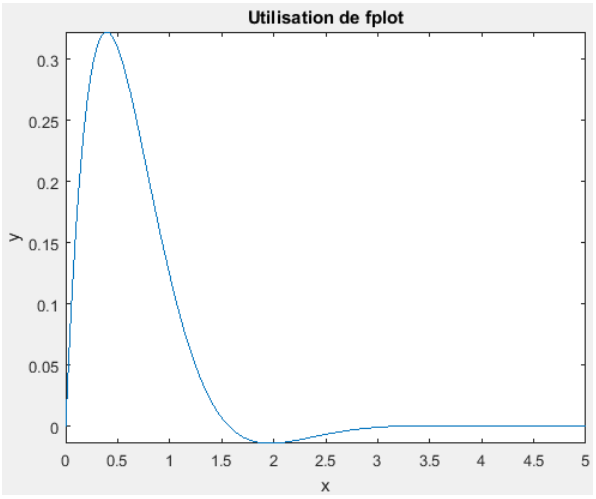


Fig. A2.9 : Tracé de la fonction $f(x)=e^{-2x}\sin 2x$.

```
>> fplot(f,[0,5]), xlabel('x'), ylabel('y'), title ('Utilisation de fplot')
```

Avec d'autres arguments, tel que la couleur, l'épaisseur du graph, ainsi d'autres commandes supplémentaires peuvent être personnalisés concerne le nom des axes, nom du graphe et la légende…

4.3. Graphiques en 3D

Dans l'espace tridimensionnel, il est possible de tracer une courbe définie par une suite de triplet $(x_i,\ y_i,\ t_i),\ \forall i=1,\ \ldots,\ n$ avec la fonction **`plot3`**, soit l'exemple (Fig. A2.10) :

```
>> t = 0:pi/50:10*pi;
>> x = sin(t); y = cos(t);
>> plot3(x,y,t)
```

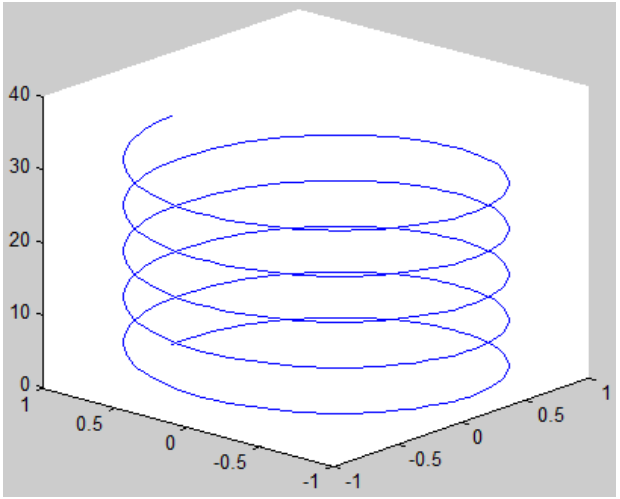


Fig. A2.10 : Graphique avec la fonction plot3.

Le tracé d'une surface dont la fonction est $z=f(x,\ y)$, nécessite la grille des points $(x_i,\ y_j),\ i=1,\ \ldots,\ n$ et $j=1,\ \ldots,\ m$ c'est-à-dire, $z_k,\ k=1,\ldots,n\times m$ à calculer afin de représenter avec MATLAB la surface de la fonction $z=f(x,\ y)$. Dans MATLAB, le tracé de la fonction $z=f(x,\ y)$ se fait par l'une des fonctions : **`mesh`**, **`surfl`** et **`surfc`** etc…

Soit l'exemple d'une fonction $f(x,\ y)$:

$$z=f(x,\ y)=(1-x)^2 e^{-x^2-(y+1)^2}-\left(x-x^3-y^5\right)e^{-x^2-y^2}\ \text{ avec }\ -4\le x\le 4,\ -4\le y\le 4$$

Le tracer de la surface $z=f(x,\ y)$, avec les fonctions citées précédemment nécessite la formation en premier lieu la grille dont les nœuds sont les points de coordonnées $(x_i,\ y_j),\ i=1,\ \ldots,\ n,\ j=1,\ \ldots,\ m$ et choisir deux pas de discrétisation Δx et Δy.

Soit la suite de commandes :

```
>> Pas2x=0.1; Pas2y=0.2;
```

```
>> x=-4:Pas2x:4; y=-4:Pas2y:4;
>> [X,Y]=meshgrid(x,y)
>> z=(1-X).^2.*exp(-(X.^2)-(Y+1).^2)-(X-X.^3-Y.^5).*exp(-X.^2-Y.^2)
>> mesh(X,Y,z)
>> figure
>> surfl(X,Y,z)
>> figure
>> surfc(X,Y,z)
```

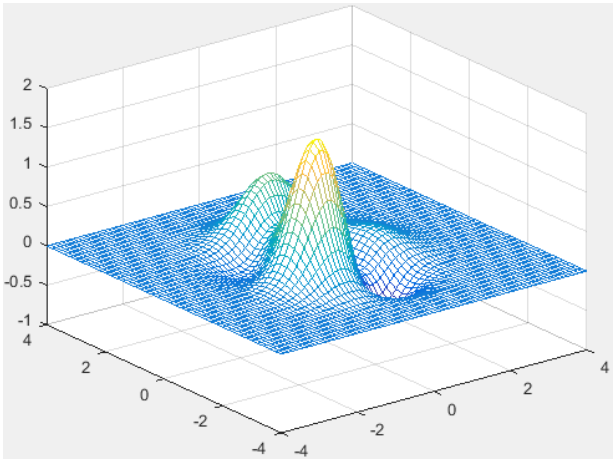


Fig. A2.11 : Surface tracée avec la fonction **mesh**.

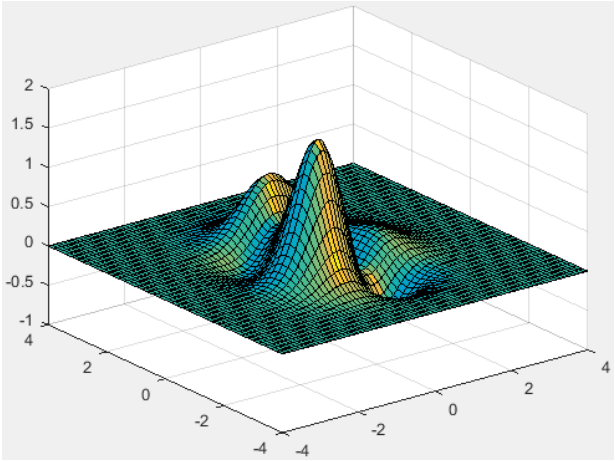


Fig. A2.12 : Surface tracée avec la fonction **surfl**.

Remarquons que la fonction **surfc** permet de tracer la surface $z = f(x,\ y)$ et tracer aussi une projection des courbes de niveau sur le plan horizontale. Pour avoir que la projection de la surface sur le plan, il suffit d'appliquer la fonction contour comme suit (Fig. A2.14).

```
>> contour(X,Y,z);
```

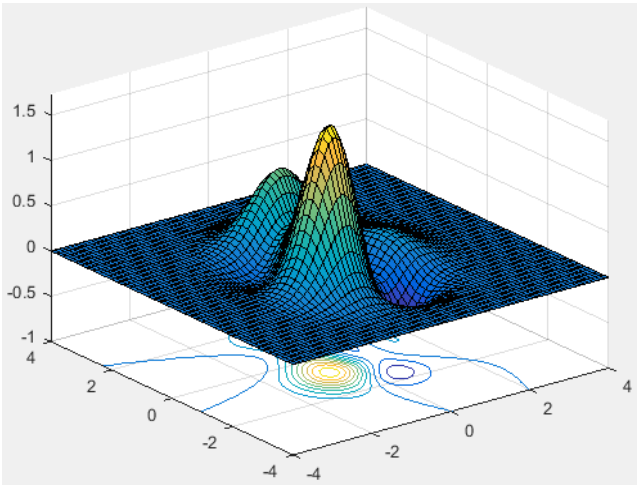


Fig. A2.13 : Surface tracée avec la fonction **surfc**.

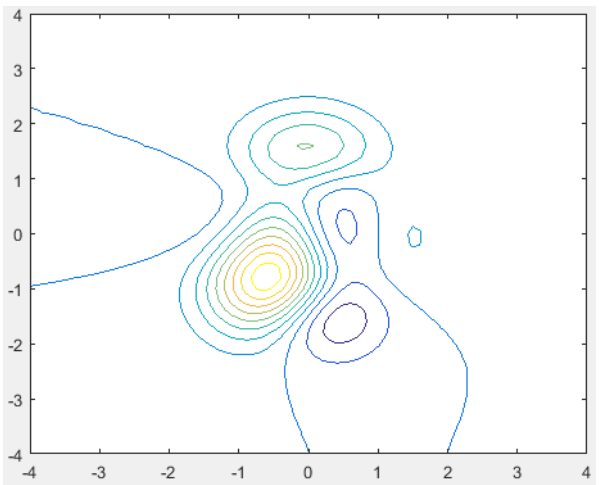


Fig. A2.14 : Projection de la surface sur le plan.

Il y a d'autres fonctions qui sont utilisées pour tracer les surfaces qui sont paramétrées avec des arguments afin de personnaliser la présentation graphique.

5. Scripts et Fonctions

Un script est un fichier texte permet regrouper une série de commandes MATLAB. Cela évite d'avoir à saisir plusieurs fois de longues suites d'instructions. Si le script est écrit dans un

fichier comme par exemple **`NameOfMFile.m`** qui peut être exécuté à partir de la fenêtre de commande de MATLAB comme :

```
>> NameOfMFile
```

A son lancement, les instructions qu'il contient s'exécutent séquentiellement comme si elles étaient lancées depuis l'invite de commande. Un script stocke ses variables dans le Workspace, lequel est partagé par tous autres scripts. Ainsi, toutes les variables créées dans les scripts sont visibles depuis la Command window et vice versa. Lorsque MATLAB détecte une erreur, le programme s'arrête et un message d'erreur s'affiche à l'écran (avec le numéro de la ligne où l'erreur est détectée).

Une fonction est aussi un script, consiste à effectuer des opérations à partir d'une ou plusieurs entrées et fournir une ou plusieurs sorties (résultat). Les variables d'entrées sont des paramètres à spécifier en argument de la fonction, tandis que les variables de sorties sont des valeurs qu'elle renvoie. Un M-File function est tout à fait semblable aux fonctions intégrées de MATLAB. Un fichier function à deux rôles, de définir des fonctions ou des procédures qui ne figurent pas parmi les fonctions incorporées de MATLAB et de les utiliser de la même manière que les fonctions utilisateur.

Les fichiers fonctions sont des éléments importants dans la programmation d'applications où les fonctions jouent le rôle des fonctions et procédures des langages de programmation usuels.

Un fichier M-File function (ou une procédure) est sauvegardé avec un nom qui est le même nom de la fonction. La structure générale d'un fichier fonction est :

```
function [Sortie1,Sortie2,Sortie3 ] = Nom2LaFonction(arg1,arg2)
%   Summary of this function goes here
%   Detailed explanation goes here

%   Commands and Functions here

end
```

La fonction contient les entrées ou arguments **`(arg1, arg2)`** et les sorties **`[Sortie1, Sortie1, Sortie3]`**. L'exemple suivant permet de calculer les valeurs de la fonction $z = f(x,\ y)$ à partir de deux vecteur *x* et *y* et trace la surface correspondante :

$$z = f(x,\ y) = (1-x)^2 e^{-x^2-(y+1)^2} - (x - x^3 - y^5) e^{-x^2-y^2}$$

```
function [z] = MaFonction(x,y)
x=x(1):0.1:x(numel(x));
y=y(1):0.2:x(numel(x));
[X,Y]=meshgrid(x,y);
z=(1-X).^2.*exp(-(X.^2)-(Y+1).^2);
z=z-(X-X.^3-Y.^5).*exp(-X.^2-Y.^2);
surfl(X,Y,z)
end
```

Dans le cas où la fonction possède qu'une seule sortie et plusieurs arguments d'entrées, il est plus commande d'utiliser une fonction anonyme (*anonymous function*) au lieu d'utiliser un fichier fonction. L'exemple de la fonction :

$$f = (1-x)^2 e^{-x^2-(y+1)^2}$$

S'écrit comme suit :

```
>> f=@(x,y)  (1-x).^2.*exp(-(x.^2)-(y+1).^2)
```

Cette écriture signifie que **x** et **y** sont les arguments de la fonction, **(x,y)** peuvent être des tableaux de mêmes dimensions, à titre d'exemple,

```
>> f(2,5)
ans =
  4.2484e-018
>> f(5,2)
ans =
  2.7423e-014
```

ou,

```
>> x=[2 1]; y=[0 1];
>> f(x,y)
  ans =
    0.0067         0
```

Un point important concerne la gestion des variables entre le programme principale (script) ou du workspace et les fonctions. Toutes les variables définies à l'intérieur d'une fonction sont des variables locales à cette fonction. La communication avec des variables du programme principal (ou du workspace) ou avec des variables d'autres fonctions se fait uniquement par les variables d'entrée et sortie de la fonction. Une alternative existe toutefois : il est possible de déclarer certaines variables comme des variables globales. Une variable globale peut être partagée entre un programme principal et plusieurs fonctions sans qu'il soit besoin de la spécifier parmi les variables d'entrée-sortie des différentes fonctions. Une variable globale est déclarée grâce au mot clé **global** suivit par le nom de variable dans chaque fonction qui utilise cette même variable.

6. Opérateurs logiques et de comparaison

Les opérateurs logiques et de comparaison sont utilisés essentiellement dans les instructions de contrôle. Les opérateurs logiques sont :

& : signifie **et** par exemple **(x & y)**
| : signifie **ou** par exemple **(x | y)**
~ : signifie **non** par exemple **(~ x)**
xor : signifie **ou exclusif** par exemple **(x xor y)**
any(x) : retourne 1 si un des éléments de **x** est non nul
all(x) : retourne 1 si tous les éléments de **x** sont nuls

Et les opérateurs de comparaison sont :

== : signifie **égal à** par exemple **(x == y)**

> : signifie **strictement plus grand que** par exemple **(x > y)**
< : signifie **strictement plus petit que** par exemple **(x < y)**
>= : signifie **plus grand ou égal à** par exemple **(x >= y)**
<= : signifie **plus petit ou égal à** par exemple **(x <= y)**
~ = : signifie **différent de** par exemple **(x ~ = y)**

7. Les structures de contrôle

Les structures de contrôle définissent l'ordre d'exécution des instructions d'un programme Dans sa forme la plus simple, le déroulement d'un programme est linéaire dans le sens où les instructions qui le composent s'exécutent successivement. Les structures de contrôle sont des mécanismes qui permettent de modifier la séquence d'exécution des instructions. Plus précisément, lors de l'exécution, en fonction des conditions réalisées certaines parties précises du code seront exécutées.

7.1. Branchement conditionnel (if … elseif … else … end)

Cette structure permet d'exécuter un bloc d'instructions en fonction de la valeur logique d'une expression. Sa syntaxe est :

```
if Expression_Logique
     Bloc d'instructions
end
```

L'ensemble des instructions (**Bloc d'instructions**) est exécuté seulement si (**Expression_Logique**) est vraie. Plusieurs tests exclusifs peuvent être combinés.

```
if Expression_Logique_1
     Bloc d'instructions_1 …
elseif Expression_Logique_2
     Bloc d'instructions_2 …
else
     Bloc d'instructions_3 …
end
```

Plusieurs **elseif** peuvent être concaténés, leur bloc est exécuté si l'expression correspondante est vraie et si toutes les conditions précédentes n'ont pas été satisfaites. Le **Bloc d'instructions_3** associé au **else** est quant à lui exécuté si aucune des conditions précédentes n'a été réalisées. Soit l'exemple :

```
if x < 0
      disp('x est négatif');
elseif x > 1
     disp('x est supérieur à 1');
else
     x = 1;
end
```

Bien évidemment, la variable **x** doit être définie auparavant. La fonction **disp** permet d'afficher une chaîne de caractère. Si **x** n'est ni négatif ni supérieur à 1, il reçoit la valeur **1**.

7.2. Choix ventilé, l'instruction `switch`

Dans cette structure, une expression numérique est comparée successivement à différentes valeurs. Dès qu'il y a identité, le bloc d'instructions correspondant est exécuté. Sa syntaxe est :

```
switch expression
   case valeur1,
     instructions_1. . .
   case valeur2,
     instructions_2. . .
     . . . . . .
   otherwise
     instructions. . .
end
```

L'expression testée, **`expression`**, doit être un scalaire ou une chaîne de caractère. Une fois qu'un bloc **`instructions_i`** est exécuté, le flux d'exécution sort de la structure et reprend après **`end`**. Si aucune **`case`** vérifie l'égalité, le bloc qui suit **`otherwise`** est exécuté. Soit l'exemple :

```
switch x
  case 0,
      resultat = a + b;
  case 1,
      resultat = a * b;
  case 2,
      resultat = a/b;
  case 3,
     resultat = a^b;
  otherwise
     resultat = 0;
end
```

En fonction de la valeur de **`x`** une opération particulière est effectuée. Par défaut, **`resultat`** prend la valeur **`0`**.

7.3. Boucle iterative `for...end`

La boucle itérative est très utilisée pratiquement dans tous les langages de programmation. La boucle for, exécute le bloc interne autant de fois que spécifié par une variable jouant un rôle de compteur, sa syntaxe est :

```
for  compteur = debut:increment:fin
     instructions …
end
```

Le **`compteur`** est initialisé à la valeur **`debut`** et évolue jusqu'à la valeur **`fin`** par pas de **`increment`**. A chaque itération, le bloc **`instructions`** est exécuté. Généralement, le **`compteur`** est un scalaire, et souvent un entier. Soit l'exemple :

```
n = 11;
for k = 1:n
      x(k)= (2*k+1)/k;
end
```

Cet exemple construit élément par élément un vecteur **x** de dimension 11.

7.4. Boucle conditionnelle while...end

Ce mécanisme permet de répéter une série d'instructions tant qu'une condition est vérifiée. Sa syntaxe est :

```
while expression
     instructions …
end
```

Le terme **expression** est une expression logique. Si cette dernière est vraie, le bloc **instructions** est exécuté. Puis, **expression** est de nouveau testée. L'exécution du bloc est répétée tant que le test est vrai. Soit l'exemple :

```
compteur = 0;
while compteur < 11
        disp('toujours dans la boucle') ;
     compteur = compteur + 1;
end
```

Cet exemple affiche 11 fois la chaîne de caractère toujours dans la boucle.

7.5. Interruption d'une boucle de contrôle

Il est possible de provoquer une sortie prématurée d'une boucle de contrôle. L'instruction **break** permet de sortir d'une boucle **for** ou d'une boucle **while**. L'exécution se poursuit alors séquentiellement à partir de l'instruction suivant le mot clé **end** fermant la boucle. En cas de boucles imbriquées, on interrompt seulement l'exécution de la boucle intérieure contenant l'instruction **break**. L'instruction **return** provoque un retour au programme appelant (ou au clavier). Les instructions suivant le **return** ne sont donc pas exécutées. L'instruction **return** est souvent utilisée conjointement avec une instruction conditionnée par exemple pour tester dans le corps d'une fonction si les paramètres d'entrée ont les valeurs attendues.

L'instruction **error** permet d'arrêter un programme et d'afficher un message d'erreur. La syntaxe est **error('message d''erreur')**. L'instruction **warning** permet d'afficher un message de mise en garde sans suspendre l'exécution du programme. La syntaxe est **warning('message de mise en garde')**. Il est possible d'indiquer à MATLAB de ne pas afficher les messages de mise en garde d'un programme en tapant **warning off** dans la fenêtre de commandes. L'affichage se rétablit suite à **warning on**.

Références bibliographiques

Abramowitz, M. & Stegun, C. A. 1972. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing. New York: Dover.

Anderson, D. A., Tannehil, J. D., & Pletcher, R. H. 1984. Computational Fluid Dynamics and Heat Transfer. McGraw-Hill, New York, NY.

Arfken, G. B. & Weber, H. J. 2005. Mathematical Methods for Physicists. 6th Edition. Elsevier Academic Press. USA.

Atkinson, K. E. 1989. An Introduction to Numerical Analysis, Second Edition. John Wiley & Sons.

Atkinson, K. E., Han, W. & Stewart, D. 2009. Numerical Solution of Ordinary Differential Equations. John Wiley & Sons, Inc., Hoboken, New Jersey, USA.

Bank, R. E., W. M. Coughran, W. Fichtner, E. H. Grosse, D. J. Rose, and R. K. Smith. 1985. "Transient Simulation of Silicon Devices and Circuits," IEEE Transactions on Computer-Aided Design, CAD-4, 436–451.

Bashforth, F. 1883. An attempt to test the theories of capillary action by comparing thetheoretical and measured forms of drops of fluid. With an explanation of the method of integration employed in constructing the tables which give the theoretical form of such drops by J. C. Adams. Cambridge Univ. Press.

Bogacki, P. & Shampine, L.F. 1989. A 3(2) pair of Runge-Kutta formulas. Applied Mathematics Letters, 2(4): 321–325.

Bradley, G. 1975. A Primer of Linear Algebra. Prentice-Hall, Englewood Cliffs, New York.

Brent, R. P. 1973. Algorithms for Minimization Without Derivatives. Englewood Cliffs, NJ: Prentice-Hall.

Burden, L. R. & Faires, J. D. 2010. Numerical Analysis, 9th Edition. Brooks/Cole, Cengage Learning.

Butcher, J. C. 1963. Coefficients for the study of Runge-Kutta integration processes. Jour. Australian Math. Soc., Vol. 3. 185–201.

Butcher, J. C. 1996. A history of Runge-Kutta methods. Applied Numerical Mathematics, Vol. 20 (3), p. 247-260.

Butcher, J. C. 2016. Numerical Methods for Ordinary Differential Equations. Third Edition. John Wiley & Sons, Ltd. UK.

Cash, J. R. and Karp, A. H. 1990. A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides. ACM Transactions on Mathematical Software, vol. 16. 201-222.

Chapra, S. C. & Canale, R. P. 2010. Numerical Methods for Engineers. Sixth Edition. McGraw-Hill, NY.

Chapra, S. C. 2012. Applied Numerical Methods with MATLAB for Engineers and Scientists. Third Edition. McGraw-Hill, NY.

Charney, J. G., Fjørtoft, R. et von Neumann, J. 1950. Numerical Integration of the Barotropic Vorticity Equation. Tellus, vol. 2, 237–254.

Chaudhry, M. H. 2008. Open – channel Flow. Second Edition. Springer.

Cornel Ioan Vălean, 2019. (Almost) Impossible Integrals, Sums, and Series. Springer Nature Switzerland AG.

Courant, R., Friedrichs, K., & Lewy, H. (March 1967) 1928. "On the partial difference equations of mathematical physics", IBM Journal of Research and Development 11 (2): 215–234.

Cryer, C. W. 1972. On the instability if high order Backward-Difference multistep methods. BIT 12. 17–25.

Cunge, J., Holly, F. M., & Verwey, A. 1980. Practical Aspects of Computational River Hydraulics, Pitman, London.

Demailly, J-P. 2006. Analyse numérique et équations différentielles. Collection Grenoble Sciences. EDP Sciences.

Démidovitch, B et Maron, I. 1979. Eléments de calcul numérique. 2ième Edition. Editions Mir, Moscou. USSR.

Curtiss, C. F. and Hirschfelder, J. O. 1951. Integration of stiff equations. Mathematics, Vol. 38, 235 – 243.

Dahlquist, G. 1956. Convergence and stability in the numerical integration of ordinary differential equations. Math. Scand. 4, 33–56.

Dharmaraja, S. 2007. An analysis of the TR-BDF2 integration scheme. Submitted to the School of Engineering in Partial Fulfilment of the Requirements for the degree of Master of Science in Computation for Design and Optimization at the Massachusetts Institute of Technology.

Dormand, J. R. and Prince, P. J. 1980. A family of embedded Runge-Kutta formulae, Journal of Computational and Applied Mathematics, vol. 6(1). 19-26.

Epperson, J. F. 2013. An introduction to numerical methods and analysis. 2nd Edition. John Wiley & Sons, Inc., Hoboken, New Jersey. USA.

Fehlberg, E. 1968. Classical fifth, sixth, seventh, and eighth-order Runge-Kutta formulas with stepsize control. NASA Technical Report 287.

Fehlberg, E. 1969. Low-order classical Runge-Kutta formulas with step size control and their application to some heat transfer problems. NASA Technical Report 315.

Fennema, R. & Chaudhry, M. H. 1986. Explicit Numerical Schemes for Unsteady Free Surface Flows with Shoks. Water Reources Res. 22, no. 13. 1923–1930.

Franz, D. D., Melching, C. S. 1997. Full Equations (FEQ) Model for the Solution of the Full Dynamic Equations of Motion for One-Dimensional Unsteady Flow in Open Channels and

through Control Structures. Report 96-4240, US Geological Survey, Water-Resources Investigations, Washington.

Fread, D. L. 1973. Effect of time step size in implicit dynamic routing. Water Resources Bull. vol. 9, N°2: 338-351.

Fread, D. L. 1974. Numerical properties of implicit four-point finite difference equations of unsteady flow. NOAA technical memorandum NWS HYDRO-18. National Weather Service, NOAA, U.S. Department of Commerce, Silver Spring, Md.

Forsythe, G. E., Malcolm, M. A., and Moler, C. B. 1977. Computer Methods for Mathematical Computations. Englewood Cliffs, NJ: Prentice-Hall.

Gilat, A. & V. Subramaniam 2014. Numerical Methods for Engineers and Scientists. An Introduction with Applications using MATLAB®. Third Edition. John Wiley & Sons, Inc.

Gill, S. & M. V. Wilkes 1951. A process for the step-by-step integration of differential equations in an automatic digital computing machine. Mathematical Proceeding of the Cambridge Philosophical Society, vol. 47(01), p. 96-108.

Glyn, J. 2011. Advanced Modern Engineering Mathematics. Fourth Edition. Pearson Education Limited.

Glyn, J. 2015. Modern Engineering Mathematics. Fifth Edition. Pearson Education Limited.

Hackbusch, W. 2016. Iterative Solution of Large Sparse Systems of Equations. 2nd Editions. Springer Switzerland.

Hairer, E., Nørsett, S. P. & Wanner, G. 1993. Solving Ordinary Differential Equations I. Nonstiff Problems. Springer-Verlag Berlin Heidelberg.

Halmos, P. 1958. Finite-Dimensional Vector Spaces. Van Nostrand, Princeton, New York.

Halphen, E. 1955. Les fonctions factorielles. Publications de l'institut de statistique de l'université de Paris, Vol. IV, Fascicule 1, 21-39.

Higham, N. J. 2002. Accuracy and Stability of Numerical Algorithms. 2nd Edition. Society for Industrial and Applied Mathematics.

Hirsch, C. 2007. Numerical Computation of Internal & External Flows. Volume 1 Fundamentals of Computational Fluid Dynamics. 2nd Edition. Butterworth-Heinemann–Elsevier. UK.

Householder, A. 1975. The Theory of Matrices in Numerical Analysis. Dover Publications New York.

Hull, T. E., Enright, W. H., Fellen, B. M. & Sedgwick, A. E. 1972. Comparing Numerical Methods for Ordinary Differential Equations. SIAM Journal on Numerical Analysis, Vol. 9, No. 4 p. 603-637.

Iserles, A. 2009. A First Course in the Numerical Analysis of Differential Equations. 2nd Ed. Cambridge Texts in Applied Mathematics. Cambridge University Press, UK.

Isaacson, E. et Keller, H. 1966. Analysis of Numerical Methods. Wiley, New York.

Iyengar, S.R.K. & Jain, R.K. hapra, S. C. 2009. Numerical Methods. New Age International (P) Limited Publishers. New Delhi, India.

Jedrzejewski, F. 2005. Introduction aux méthodes numériques. Deuxième édition. Springer-Verlag France, Paris.

Kharab, A. & R. B. Guenther 2019. An Introduction to Numerical Methods. A MATLAB® Approach. Fourth Edition. CRC Press. Taylor & Francis Group.

Kierzenka, J. & Shampine, L. F. 2001. A BVP solver based on residual control and the MATLAB PSE. ACM TOMS 27(3):299–316.

Kiusalaas, J. 2010. Numerical methods in engineering with MATLAB. Second Edition. Cambridge University Press.

Kuncir, G.F. 1962. Algorithm 103: Simpson's rule integrator, Communications of the ACM, 5 (6): 347.

Kutta, W . 1901. Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. Z . Math . Phys . Band 46 , 1901 , p. 435 - 453.

Laplace, A. 1969. Méthodes de Runge-Kutta, Fehlberg. Thèse de doctorat de 3ième Cycle en Mathématiques Appliquées. Université Joseph Fourier – Grenoble I. France.

Lax, P.D. & Richtmyer, R. D. 1956. Survey of the Stability of Linear Finite Difference Equations. Communications on Pure and Applied Mathematics, Vol IX, 267–293.

Lax, P.D. & Wendroff, B. 1960. Systems of conservation laws. Comm. Pure Appl. Math. 13, 217–237.

Liggett, J. A. & Cunge, J. A. 1975. Numerical methods for the solution of the unsteady flow equations. In: Mahmood and Yevjevich (eds), Unsteady Flow in Open Channels, p. 89–172. Water Resources Publications, Littleton, CO.

MacCormack, R. W. & Paullay, A. J. 1972. Computational efficiency achieved by time splitting of finite difference operators. AIAA Paper 72–154.

MacCormack, R. W. 1969. The Effect of viscosity in hypervelocity impact cratering, AIAA Paper, 69-354.

Merson, R. H. 1957. An operational method for the study of integration processes, Proceedings of the Symposium on Data Processing, Weapons Research Establishment. Salisbury, South Australia, p. 110–125.

Milne, W. E. 1926. Numerical integration of ordinary differential equations. Amer. Math. Monthly, Vol. 33, 455–460.

Moler, C. 2004. Numerical Computing with MATLAB. Society for Industrial and Applied Mathematics, Philadelphia, U.S.

Morozov, V. 1984. Methods for Solving Incorrectly Posed Problems. Springer-Verlag, New York.

Moulton, F. R. 1926. New methods in exterior ballistics. University of Chicago Press. Rééditée. Methods in exterior ballistics. Dover, New York.

Nahin, P. J. 2015. Inside Interesting Integrals. Springer Science+Business Media New York.

Noble, B. 1969. Applied Linear Algebra. Prentice-Hall, Englewood Cliffs, New York.

Nyström, E.J. 1925. Ueber die numerische Integration von Differentialgleichungen. Acta Soc. Sci. Fenn., Vol. 50, No. 13, p. 1-54.

Phillips, G. M. & Taylor, P. J. 1996. Theory and Applications of Numerical Analysis. 2nd Edition. Academic Press. Elsevier Science & Technology Books.

Press, H. W., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. 2007. Numerical Recipes. The Art of Scientific Computing. Third Edition. Cambridge University Press. 1235 p.

Preissmann, A. 1961. Propagation des intumescences dans les canaux et rivières. 1er Congrès de l'Association Française de Calcul, Grenoble, France, 433–442.

Quarteroni, A., Sacco, R. & Saleri, F. 2007. Méthodes Numériques. Algorithmes, analyse et applications. Traduction à partir de l'ouvrage italien : Matematica Numerica – A. Quarteroni, R. Sacco, F. Saleri. Springer-Verlag Italia, Milano.

Quarteroni, A., Saleri, F. & Gervasio, 2010. Calcul Scientifique. Cours, exercices corrigés et illustrations en MATLAB et Octave. 2ième Edition. Traduction à partir de l'ouvrage italien : Calcolo Scientifico - Esercizi e problemi risolti con MATLAB e Octave A. Quarteroni, F. Saleri – 4 edizione. Springer-Verlag Italia.

Romberg, W. 1955. Vereinfachte numerische Integration. Det Kong. Norske Videnskabernes Selskab Forhandlingar, Bind 28, Nr. 7.

Shampine, L. F. & Reichelt, M. W. 1997. The Matlab ODE Suite. SIAM Journal on Scientific Computing, 18(1): 1–22.

Stewart, G. W. 1998. Matrix Algorithms. Vol 1: Basic Decompositions. Society for Industrial and Applied Mathematics. Philadelphia.

Sturm, T. W. 2001. Open Channel Hydraulics. McGraw-Hill, New York, NY.

Strikwerda, J. C. 2004. Finite Difference Schemes and Partial Differential Equations. 2nd Edition. Society for Industrial and Applied Mathematics SIAM, Philadelphia. USA.

Süli, E. Mayers, D. 2003. An Introduction to Numerical Analysis, Cambridge University Press.

Taylor, B. 1715. Methodus incrementorum directa & inversa. Londini.

Timothy Sauer 2012. Numerical Analysis. Second Edition. Pearson Education, Inc.

Watson, G. N. 1966. A Treatise on the Theory of Bessel Functions. Cambrige University Press.

Weisstein, E. W. 2003. CRC Concise Encyclopedia of Mathematics. 2nd Ed. Chapmann & Hall/CRC. USA.

Whittaker, E. T. & Robinson, G. 1944. The Calculus of Observations: A Treatise on Numerical Mathematics. 4th Edition, Blackie & Son Limited. London and Glasgow.

Wilkinson, J. 1963. T Rounding Errors in Algebraic Processes. Prentice-Hall, Englewood Cliffs, New York.

Yahiaoui, A. 2017. Introduction à MATLAB. Université de Béchar. Algérie.

Young, M. D. 1971. Iterative solution of large linear systems. Academic Press, INC. London. UK.

Zill, D. G. & Wright, W. S. 2014. Advanced Engineering Mathematics. Fifth Edition. Jones & Bartlett Learning.

Zwillinger, D. 2003. CRC Standard Mathematical Tables and Formulae, 31st Edition. Chapman & Hall/CRC Press Company.

Les méthodes numériques est une discipline à l'interface des mathématiques et de l'informatique. Elle s'intéresse tant aux fondements qu'à la mise en pratique des méthodes permettant de résoudre, par des calculs numériques, des problèmes d'analyse mathématique et de la physique.

Plus formellement, les méthodes numériques en ingénierie, est l'étude des algorithmes permettant de résoudre numériquement par discrétisation les problèmes de mathématiques continues. Cela signifie qu'elle s'occupe principalement de répondre de façon numérique à des questions à variable réelle ou complexe comme l'algèbre linéaire numérique sur les champs réels ou complexes, la recherche de solution numérique d'équations différentielles et d'autres problèmes liés survenant dans les sciences physiques et l'ingénierie.

Le développement des méthodes numériques est étroitement lié à celui des outils de la programmation en informatique ou certains problèmes mathématiques peuvent être résolus numériquement avec l'ordinateur de façon exacte par un algorithme en un nombre fini d'opérations.

L'utilisation des méthodes numériques est grandement facilitée par les ordinateurs. L'accroissement de la disponibilité et de la puissance des ordinateurs depuis la seconde moitié du XXe siècle a permis l'application de l'analyse numérique dans de nombreux domaines scientifiques, techniques et économiques, avec souvent des effets importants.