Université de Liège
Faculté des Sciences Appliquées

Introduction aux Méthodes Numériques

Professeur Q. Louveaux
Département d'Électricité, Électronique et Informatique
Institut Montefiore

Table des matières

Chapitre 1

Introduction

Pourquoi un cours d'analyse numérique et à quoi cela sert-il ? Pour résumer, on peut dire que l'analyse numérique est à la frontière entre les mathématiques et l'informatique. Il s'agit en quelque sorte d'un *interprète* qui permettra de transposer la connaissance mathématique théorique à la pratique d'un ordinateur et de pouvoir ainsi résoudre des problèmes concrets.

Les deux objectifs principaux de l'analyse numérique sont, d'une part, de pouvoir résoudre *numériquement* des problèmes concrets dont on connaît ou pas la solution analytique et d'autre part, d'analyser le *comportement* des méthodes utilisées. Développons à présent ces deux objectifs.

1.1 Résolution numérique de problèmes

La plus grande partie de ce cours consiste à développer des méthodes pour résoudre numériquement des problèmes scientifiques courants.

Exemple 1.1 Imaginons que nous disposions d'une calculatrice de poche capable d'efectuer les opérations courantes d'addition, soustraction, multiplication et division. Comment pouvons-nous évaluer la constante e ?

Nous savons que le développement de Taylor de la fonction e^x est

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

c'est-à-dire, que nous pouvons approximer la constante e par, par exemple,

$$e = 1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120},$$

ce qui donnerait dans ce cas-ci un résultat (très) approximatif de $e = 2,71667$ ayant seulement deux décimales correctes. Il est évidemment possible d'obtenir un résultat plus précis en utilisant plus de termes de la série. C'est là que l'on voit tout l'intérêt de pouvoir correctement analyser une méthode. Ici, cela permettra de savoir combien de termes de la série sont nécessaires afin d'obtenir le nombre de décimales correctes désiré. ■

On peut considérer que la constante e est connue et que l'exemple précédent est uniquement un moyen de transposer concrètement à l'ordinateur une théorie abstraite. Par ailleurs, la formule donnée est *exacte* au terme d'erreur de l'expansion de Taylor près (que nous reverrons dans le Chapitre 2). Un objectif de ce cours est également de traiter des problèmes dont une solution analytique n'est pas connue, mais que l'on peut approcher numériquement de manière satisfaisante. Dans la plupart des applications pratiques, une solution donnant 10 décimales correctes n'est d'ailleurs pas toujours nécessaire. Imaginons que l'on doive dimensionner le diamètre de barres d'acier devant soutenir un pont. Il y a peu d'utilité pratique à réclamer un diamètre de 150.3429836 mm plutôt que, plus humblement, et plus raisonnablement un diamètre de 150.3 mm.

Exemple 1.2 Lors du dimensionnement de poutres de résistance, un calcul intermédiaire fréquent consiste à résoudre une équation du troisième degré. Bien qu'il existe une méthode analytique permettant de trouver une solution, il est souvent bien plus efficace de résoudre numériquement l'équation. Considérons donc l'équation

$$x^3 + x - 1 = 0. \tag{1.1}$$

La méthode numérique la plus naïve pour résoudre $f(x) = 0$, où f est continue, est la méthode de la *dichotomie* (ou bisection). Elle consiste à partir de deux valeurs x_0 et x_1 tels $f(x_0)f(x_1) < 0$, ce qui implique qu'une solution x du problème se trouve entre x_0 et x_1. Il suffit ensuite de considérer $x_2 := \frac{x_0+x_1}{2}$ et d'évaluer $f(x_2)$. En fonction du signe du résultat, on continuera sur l'intervalle compris entre x_0 et x_2 ou l'intervalle compris entre x_1 et x_2. Et ainsi de suite... L'algorithme permet de réduire la taille de l'intervalle d'un facteur 2 à chaque itération. Il est ainsi facile d'évaluer le nombre d'itérations nécessaires pour obtenir une précision désirée sur x. Sur l'exemple précité, si on pose $f(x) = x^3 + x - 1$, on voit aisément que $f(0) = -1$ et que $f(1) = 1$.

Dès lors, nous savons que f possède une racine sur l'intervalle [0,1]. Nous pouvons dès lors démarrer l'algorithme avec $x_0 = 0$ et $x_1 = 1$. La suite de l'algorithme est présentée dans le tableau suivant. Remarquons que, puisque l'on part d'un intervalle de taille 1, la taille de l'intervalle à l'itération i est de $\frac{1}{2^i}$. En particulier, si l'on souhaite obtenir 3 décimales correctes, nous en déduisons que la taille de l'intervalle doit être inférieure à $0.5\ 10^{-3}$. Dès lors 11 itérations seront suffisantes.

Itér.	x	y	$z := \frac{x+y}{2}$	$f(z)$
0	0.000000	1.000000	0.500000	-0.375000
1	0.500000	1.000000	0.750000	0.171875
2	0.500000	0.750000	0.625000	-0.130859
3	0.625000	0.750000	0.687500	0.012451
4	0.625000	0.687500	0.656250	-0.061127
5	0.656250	0.687500	0.671875	-0.024830
6	0.671875	0.687500	0.679688	-0.006314
7	0.679688	0.687500	0.683594	0.003037
8	0.679688	0.683594	0.681641	-0.001646
9	0.681641	0.683594	0.682617	0.000694
10	0.681641	0.682617	0.682129	-0.000477

Nous voyons qu'au bout de 11 itérations, 3 décimales d'une solution à l'équation sont connues avec certitude, à savoir $x = 0.682$. ■

Bien qu'extrêmement simple, la méthode que nous avons présentée dans l'exemple précédent permet, assez rapidement, de trouver une solution satisfaisante à une équation dont une solution analytique n'est pas disponible (ou peu pratique dans ce cas-ci). Nous verrons aussi qu'un grand avantage de cette méthode est sa bonne stabilité numérique. Le Chapitre 4 présentera les approches classiques pour résoudre $f(x) = 0$.

1.2 Analyse du comportement des méthodes

Nous avons vu, dans la section précédente, deux méthodes pour résoudre des problèmes avec un ordinateur. Il est très important de décrire les algorithmes qui permettent de résoudre ces problèmes, mais il n'est non moins important d'analyser le comportement de ces méthodes. En effet plusieurs questions se posent : est-ce qu'un algorithme trouvera plus vite la solution

qu'un autre, quelle est la quantité de travail nécessaire à un algorithme pour obtenir une précision désirée, est-ce que l'algorithme trouvera toujours une solution, est-ce toujours la solution souhaitée, est-ce que l'algorithme est sensible aux erreurs dans les données initiales (qui peuvent provenir de mesures par essence imprécises), est-ce que l'algorithme est sensible aux erreurs d'arrondi faites durant le calcul ? Nous montrons à présent la pertinence de ces thèmes sans rentrer dans les détails (ce que nous ferons dans les chapitres) mais en illustrant la philosophie des problèmes qui se posent.

1.2.1 Complexité d'un algorithme

Une question importante, et qui relève plus de l'algorithmique que de l'analyse numérique, est la quantité de travail nécessaire à un algorithme pour arriver à un résultat. C'est ce que l'on appelle la complexité d'un algorithme. Par quantité de travail, nous entendons le nombre d'opérations de base (additions, soustractions, multiplications, divisions) qui sont exécutées. On exprime la complexité d'un algorithme comme une fonction (souvent un polynôme) dont les variables sont les *paramètres* du problème. Ces paramètres dépendent du type de problème résolu. Souvent il s'agit de la taille du problème (nombre de lignes et de colonnes d'une matrice par exemple) et de la taille maximale des coefficients (dans certains cas un problème sera plus difficile s'il concerne des grands nombres). Comme nous l'avons dit, la fonction de complexité est fréquemment un polynôme. Dans ce cas, il est courant de considérer le degré du polynôme comme donnant une idée de l'efficacité de l'algorithme. Plus le degré du polynôme est bas, plus l'algorithme est rapide. Par ailleurs, un élément également important est de connaître la quantité de mémoire dont l'algorithme a besoin pour pouvoir s'effectuer. De la même façon que pour le nombre d'opérations, on exprime le nombre d'éléments mémoire par une fonction des paramètres de problème.

1.2.2 Convergence d'un algorithme

De nombreuses méthodes numériques sont itératives et approximent la solution à un problème de mieux en mieux au fur et à mesure des itérations. Dans la section précédente, on s'intéressait à la quantité de travail effectuée à chaque itération. Une question très importante est de savoir quelle est la précision obtenue après un certain nombre d'itérations. Le corollaire sera de

pouvoir évaluer le nombre d'itérations nécessaires pour obtenir la précision désirée.

Considérons à nouveau la résolution de l'équation donnée dans l'Exemple 1.2. Nous avons vu que, pour la méthode de recherche dichotomique, la taille de l'intervalle dans lequel se situe la racine recherchée diminue de moitié à chaque itération. Mathématiquement, considèrons la suite (x_n) correspondant au milieu de l'intervalle traité à chaque itération. Nous avons que $\lim_{n\to\infty} x_n = x$ où x est la racine recherchée. Par ailleurs, étant donné la réduction de l'intervalle de recherche de moitié à chaque itération, nous en déduisons que

$$|x_n - x| \leq \frac{|b-a|}{2^n},$$

où $[a, b]$ est l'intervalle de départ. Lorsqu'une suite (y_n) tend vers y de telle façon que $|y_n - y| \leq c_0 c^n$ avec $c_0 \in \mathbb{R}_+$ et $0 < c < 1$, on parle de convergence *linéaire*. Si par contre, la suite tend vers y avec $|y_n - y| \leq c_0 c^{pn}$ avec $c_0 \in \mathbb{R}_+$, $0 < c < 1$ et $p \in \mathbb{R}_+$, on parle de *convergence d'ordre p*.

Il existe d'autres méthodes de recherche de la racine d'une équation. Nous verrons au Chapitre 4 la méthode de *Newton-Raphson*. Celle-ci a un ordre de convergence quadratique (ordre 2) dans la plupart des cas. Sans rentrer dans les détails de la méthode, nous pouvons comparer la vitesse de convergence de celle-ci avec la méthode de la recherche dichotomique. La tableau suivant indique une comparaison des solutions approchées à chaque itération ainsi que du nombre de décimales correctes.

Itér.	Rech. Dichotomique		Newton-Raphson	
	x approché	Déc. corr.	x approché	Déc. corr.
0	0.500000	0	0.0000000000000000	0
1	0.750000	0	1.0000000000000000	0
2	0.625000	1	0.7500000000000000	0
3	0.687500	2	0.6860465116279070	2
4	0.656250	1	0.6823395825973142	4
5	0.671875	1	0.6823278039465127	9
6	0.679688	1	0.6823278038280193	16
7	0.683594	2		
8	0.681641	2		

On voit dans le tableau que l'ordre de convergence indique une performance *asymptotique*. En particulier, la méthode de Newton-Raphson n'est pas meilleure que la dichotomie dans les premières itérations. Mais elle est très efficace

au voisinage de la racine. On voit également qu'une convergence linéaire ou quadratique n'exclut pas que la méthode régresse par moments. A nouveau, la qualité de l'ordre de convergence est une propriété asymptotique.

Quand on parle de convergence d'un algorithme, il est aussi important de parler de zone de convergence en fonction d'un point initial choisi. La méthode de Newton-Raphson, par exemple, peu être très dépendante du point initial. Pour certains points de départ, elle peut même ne pas converger. La question de zone de convergence est fréquemment une question beaucoup plus difficile que la question de l'ordre de convergence au voisinage de la solution.

1.2.3 Sensibilité aux erreurs des données

Cette question concerne plus souvent les problèmes plutôt que les algorithmes. En effet, certains problèmes sont, ce qu'on appelle *mal conditionnés*. Dans ce cas-ci, un petit changement des données peut mener à un changement radical de la solution. Lorsque l'on utilise des méthodes numériques pour résoudre ces problèmes, la question devient cruciale car le fait de travailler en précision finie amène régulièrement à devoir modifier légèrement les données réelles. La modification des données peut également provenir de mesures concrètes qui sont, par essence, imprécises. Un exemple typique de problème mal conditionné est le cas de systèmes linéaires dont le déterminant est proche de 0.

Exemple 1.3 Considérons le système $Ax = b$ où

$$A = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix}, \qquad b = \begin{pmatrix} 0.8642 \\ 0.1440 \end{pmatrix}.$$

On peut vérifier que la solution unique de ce système est $x = (2 \quad -2)^T$. Imaginons maintenant qu'une petite erreur dans l'introduction de la matrice change $A(2,2)$ en 0.144. Nous obtenons un sytème très proche avec

$$A = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.144 \end{pmatrix}, \qquad b = \begin{pmatrix} 0.8642 \\ 0.1440 \end{pmatrix}.$$

La solution du sytème est maintenant $x = (0.6663 \;\; 0.0002)^T$ qui n'a plus aucun chiffre commun avec la solution précédente. ■

1.2.4 Influence des erreurs d'arrondi

Lorsque l'on travaille sur un ordinateur, on est obligé de représenter les nombres de manière finie, souvent en arrondissant les décimales les moins représentatives. Par exemple, on peut décider de travailler avec 16 chiffres significatifs. Nous définirons précisément dans le Chapitre 2 ce que cela signifie. Dans la plupart des cas, tronquer ainsi la signification d'un nombre n'a que peu d'impact sur la solution finale recherchée. Il peut arriver néanmoins que ces toutes petites erreurs s'accumulent et finissent par rendre les résultats totalement erronés. Le cas le plus fréquent de perte de précision dûe aux erreurs d'arrondis est lorsque l'on soustrait deux nombres très proches. Le Chapitre 2 traitera de ce problème et tentera de proposer quelques pistes pour le résoudre. Mais ce n'est pas le seul problème qui peut arriver comme le montre l'exemple suivant.

Exemple 1.4 Nous souhaitons trouver une valeur approchée de l'intégrale $I_n = \int_0^1 x^n e^x dx$ pour $n = 20$. En intégrant par parties, on voit que l'on a $I_n = e - n\int_0^1 x^{n-1}e^x dx$. On peut donc écrire la relation de récurrence $I_n = e - nI_{n-1}$. Par ailleurs $I_0 = e-1$ ce qui implique que $I_1 = 1$. Il est dès lors aisé de calculer par récurrence les différentes valeurs de I_i pour $i = 2, \ldots, 20$. La Figure 1.1 reporte les valeurs ainsi calculées pour $i = 1, \ldots, 20$ en trait plein. Les ronds sur la Figure 1.1 reportent les valeurs réelles de l'intégrale.

Comme on peut le voir, la valeur approchée par récurrence est complètement erronée pour $n \geq 17$. Le tableau suivant reporte les valeurs calculées et réelles pour $n \geq 17$.

n	I_n par récurrence	I_n réel
17	0.1043	0.1434
18	0.8417	0.1362
19	-13.2742	0.1297
20	268.2026	0.1238

Comment un résultat assi aberrant peut-il être obtenu à partir d'une formule à l'apparence aussi insignifiante ? La réponse est à trouver dans les erreurs d'arrondis. En effet, lors du calcul de I_2, si on considère que le calcul se fait avec une précision de 16 décimales, une infime erreur est faite dans l'expression de e. Pour fixer les idées, supposons que cette erreur soit de 10^{-17} et supposons que cela soit la seule erreur réalisée lors de tout le calcul. Si on note par $\tilde{I}_2$ la valeur calculée par récurrence, supposons donc que l'on ait

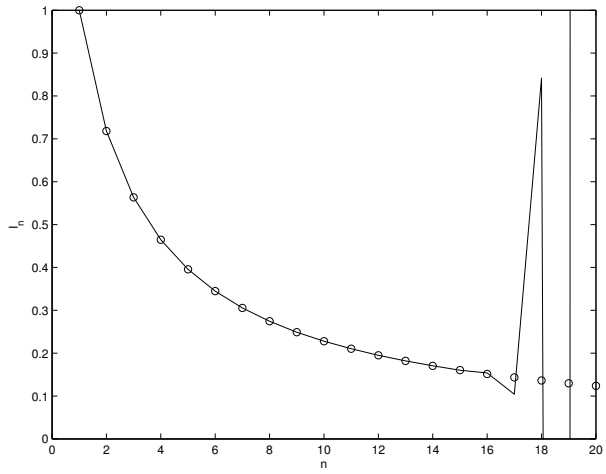

FIGURE 1.1 – Valeur réelle et approchée par récurrence de $\int_0^1 x^n e^x dx$

$\tilde{I}_2 = I_2 + e_2$ avec $|e_2| = 10^{-17}$. Nous avons, dès lors, $\tilde{I}_3 = e - 3\tilde{I}_2 = I_3 - 3e_2$. Si on continue de la sorte, on voit que l'on obtient successivement $\tilde{I}_n = I_n + e_n$ où $|e_n| = n!|e_2| = 10^{-17}n!$. Cela donne, en particulier, pour $n = 20$ une valeur de l'erreur d'environ 24 et qui rend les valeurs calculées totalement dénuées de sens. ■

L'exemple précédent montre à quel point il est crucial de vérifier qu'un algorithme n'est pas trop sensible aux erreurs d'arrondis.

1.2.5 Conclusion

L'approche traditionnelle de l'analyse numérique est de se focaliser sur la présentation des méthodes de résolution de problèmes et d'analyser leur comportement ensuite à la lumière de tous les phénomènes d'instabilité parfois inattendus qui peuvent survenir. Nous ferons de même dans ce cours. Une telle présentation qui met en avant les méthodes peut faire penser que celles-ci priment sur l'analyse détaillée qu'on peut en faire. Il est néanmoins important de garder à l'esprit les différents types d'erreur qui peuvent se présenter. La fréquence de tels problèmes numériques n'étant pas extrêmement élevée, et les logiciels modernes étant écrits de manière robuste, il est naturel d'oublier petit à petit que des problèmes numériques parfois aigus se présentent.

Un bon scientifique doit donc toujours garder l'esprit critique et prêter une attention toute particulière à la stabilité des modèles et des méthodes qu'il écrit et utilise.

Chapitre 2

Représentation des nombres et erreurs

Dans ce chapitre, nous formalisons la représentation des nombres dans un ordinateur et les erreurs qui en résultent. Tout d'abord, nous rappelons les principales notions relatives aux séries de Taylor qui constituent l'outil principal pour l'analyse numérique.

2.1 Rappel sur les séries de Taylor

Rappelons brièvement l'expansion d'une fonction en série de Taylor.

Théorème 2.1 *Soit f, une fonction possédant ses $(n+1)$ premières dérivées continues sur un intervalle fermé $[a, b]$, alors pour chaque $c, x \in [a, b]$, f peut s'écrire comme*

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

où le terme d'erreur E_{n+1} peut s'écrire sous la forme

$$E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1},$$

et ξ est un point situé entre c et x.

On dit que le terme d'erreur est *d'ordre* $n+1$. Il est parfois utile d'écrire cela de manière plus compacte sous la forme $E_{n+1} = \mathcal{O}(h^{n+1})$, où h représente

$x - c$. La notation $\mathcal{O}(x^p)$ permet de décrire le fait que la fonction tend au moins aussi vite vers 0 que x^p lorsque x tend vers 0.

Définition 2.1 *Soit $f : \mathbb{R} \mapsto \mathbb{R}$. On dit que $f = \mathcal{O}(x^p)$ au voisinage de 0 si il existe $C \in \mathbb{R}_+$ et $x_0 \in \mathbb{R}_+$ tels que*

$$|f(x)| \leq C|x^p| \quad \textit{pour tout } -x_0 \leq x \leq x_0.$$

Cette définition sera tres souvent utilisée. En effet, le degré minimal d'un polynôme donne une idée très précise de la vitesse de convergence vers 0. Si on approxime une valeur V en l'approchant itérativement par une fonction $f(h)$ quand h tend vers 0, une évaluation de $f(h) - V$ en notation $\mathcal{O}$ donne une mesure de la vitesse de convergence du processus itératif. Plus le degré de convergence est élevé, plus la convergence se fait rapidement. Le théorème de Taylor est un outil extrêmement puissant pour pouvoir moduler l'ordre de précision que l'on souhaite d'une fonction.

2.2 Erreur absolue et relative

Lorsqu'un nombre est stocké dans l'ordinateur, il y a très souvent une erreur dont il faut tenir compte. Cette erreur peut provenir soit d'une approximation de l'algorithme, soit d'erreurs dans les données, soit enfin d'erreurs dûes aux arrondis que la machine réalise pour correspondre à sa précision finie. Il y a donc lieu de modéliser ces erreurs. Dans la définition de l'erreur sur un nombre, on distingue deux cas.

Définition 2.2 *Soit $\tilde{a}$ la valeur approchée d'une quantité dont la valeur exacte est a. On appelle*

(i) *l'*erreur absolue *la quantité $\tilde{a} - a$,*

(ii) *l'*erreur relative *la quantité $\frac{\tilde{a}-a}{a}$.*

Lié à cette définition, on parle également de *décimales correctes* et de *chiffres significatifs*. Le nombre de décimales correctes est lié à l'erreur absolue. Lorsque la valeur absolue de l'erreur absolue ne dépasse pas $\frac{1}{2}10^{-t}$, on dit que $\tilde{a}$ a t décimales correctes. Le nombre de chiffres significatifs est, quant à lui, lié à l'erreur relative. Le nombre de chiffres corrects de $\tilde{a}$ à partir du premier chiffre ou à partir de la première décimale non nulle sont appelés chiffres

significatifs. En particulier lorsque la valeur absolue de l'erreur relative ne dépasse pas $\frac{1}{2}10^{-s+1}$, on dit que $\tilde{a}$ a s chiffres significatifs.

Exemple 2.1 Considérons le nombre $a = 1234.5678$ et son approximation $\tilde{a} = 1234.57$. L'erreur absolue vaut $\tilde{a} - a = 0.0022 \leq 0.5\ 10^{-2}$, donc $\tilde{a}$ a 2 décimales correctes. Si on considère l'erreur relative, on obtient $\frac{0.0022}{1234.5678} \leq \frac{1}{2}10^{-5}$, c'est-à-dire que $\tilde{a}$ a 6 chiffres significatifs. ■

En règle générale, on préférera toujours considérer l'erreur relative étant donné la représentation en virgule flottante des nombres. Nous parlerons de cette représentation dans la section suivante. Par rapport à la logique des phénomènes physiques, chimiques ou mathématiques, parler de l'erreur relative a également plus de sens. Si on demande la distance entre deux villes, on s'arrêtera en général à préciser les kilomètres, sans rentrer dans les détails des mètres et des centimètres qui ont peu de valeur pour un voyageur. On peut dire, avec 1 chiffre significatif (voire 2), que la distance entre Bruxelles et Liège est de 100 km, ce qui est, dans la plupart des applications, amplement suffisant comme mesure, bien que l'erreur absolue se compte probablement en kilomètres.

2.3 Représentation des nombres en virgule flottante

Dans la plupart des ordinateurs, les nombres réels (mais pas les nombres définis comme entiers) sont représentés en *virgule flottante*. Un nombre a est ainsi défini par deux autres *nombres* m et q,

$$a = m\ 10^q, \qquad \text{avec } 1 \leq |m| < 10 \text{ et } q \in \mathbb{Z}.$$

La partie m est appelée la *mantisse* tandis que q est appelé l'exposant. En réalité, la machine stocke la plupart des nombres en binaire, c'est-à-dire en base 2 mais nous ferons, dans ce cours, l'approximation que tout se passe réellement en base 10. Remarquons que cette approximation n'est pas totalement anodine. Un nombre aussi simple que 0.1 est stocké sans erreur dans une machine en base 10, alors qu'une erreur d'arrondi survient si on travaille en base 2.

Revenons maintenant à notre mantisse et à notre exposant. Ceux-ci sont stockés sur un nombre fini de bits. Cela implique que seulement une quantité finie de nombres réels peut être représentée dans ce système. Par exemple, si q est stocké sur 8 bits, cela signifie que l'exposant se situe entre -127 et 128. Tout nombre dont l'exposant est supérieur à 128 ne pourra pas être représenté dans ce système. On parle alors d'*overflow*. Il s'agit en général d'une erreur qui cause l'arrêt d'un programme. Un nombre x dont l'exposant est inférieur à -127 ne pourra pas non plus être correctement représenté. On parle dans ce cas d'*underflow*. Cette erreur est moins grave car on peut alors remplacer x par 0 en faisant une erreur relativement faible. Mais dans certains cas, cette approximation peut évidemment être fatale. Imaginons que l'on veuille ensuite diviser par $x\ldots$

La mantisse est également stockée sur un nombre fini de bits. Cela implique que la précision sur un nombre est limitée au nombre de décimales que la mantisse peut contenir. En particulier, si un nombre x ne peut pas être représenté en utilisant toutes les décimales, on choisit le nombre le plus proche à pouvoir être représenté comme valeur de x. C'est ce qu'on appelle l'arrondi. Nous reviendrons plus tard sur la façon d'arrondir. Mais nous pouvons déjà définir un concept important : celui d'*epsilon machine*. L'epsilon machine représente la différence entre deux mantisses consécutives. Supposons, pour simplifier, que nous travaillions en base 10 avec 5 chiffres pour la mantisse. Supposons que l'on parte de 1. Quel est le plus petit nombre qui soit plus grand que 1 et qui soit représentable exactement dans l'ordinateur ? Il s'agit de 1.0001. Tous les nombres compris entre 1 et 1.0001 ne peuvent être représentés exactement et doivent être arrondis à l'un de ces deux nombres. La différence $1.0001 - 1 = 0.0001$ est appelé l'epsilon machine. On peut également le voir comme le plus petit nombre ϵ tel que $1 + \epsilon \neq 1$ pour la machine.

De manière assez courante, on travaille avec des nombres en double précision. Ceux-ci sont stockés en base 2 avec 52 bits pour la mantisse (plus 1 pour le signe de la mantisse) et 11 bits pour l'exposant. Dans ce cas, l'epsilon machine vaut 2^{-52} soit environ $2.2\ 10^{-16}$. Le nombre le plus élevé est, quant à lui, approximativement égal à 2^{1024} soit environ $1.8\ 10^{308}$. Il peut être utile de connaître ces nombres lorsque l'on programme pour prévoir les difficultés qui peuvent survenir.

Comme nous venons de le dire, un nombre doit, en général, être arrondi pour pouvoir prendre place dans les 64 bits qui le représentent (dans le cas

d'une double précision). Il existe une façon standard d'arrondir un nombre qui est décrit par le standard dit *IEEE*. Comme on peut s'en douter, arrondir un nombre x consiste à trouver le nombre $\tilde{x}$ qui puisse être représenté et qui soit le plus proche possible de x. Considérons le cas d'une représentation en base 10 de x, et que l'on travaille avec p décimales. La mantisse de l'arrondi $\tilde{x}$ aura les mêmes $p-1$ premiers chiffres que la mantisse de p. Le dernier chiffre sera, soit le même, soit une unité plus élevée si le le $(p+1)$ème chiffre est plus grand ou égal à 5.

Exemple 2.2 Supposons que l'on travaille avec trois chiffres décimaux dans la mantisse. Alors nous avons que

125	est arrondi à	$1.25\ 10^2$
44.34	est arrondi à	$4.43\ 10^1$
59052	est arrondi à	$5.91\ 10^4$
0.1455	est arrondi à	$1.46\ 10^{-1}$

■

Pour formaliser cet arrondi, nous pouvons supposer que l'arrondi d'un nombre x à $\tilde{x}$ consiste à effectuer l'opération $\tilde{x} = fl(x) = (1+\delta)x$ et δ est borné en valeur absolue par l'epsilon machine. Lorsqu'on effectue une opération sur deux nombres machines, on peut en général faire l'approximation que le calcul se fait, dans un premier temps, correctement et qu'ensuite le résultat est arrondi pour être stocké dans une variable. On aura donc que, sur une machine,

$$\begin{aligned} fl(x \pm y) &= (x \pm y)(1+\delta), \\ fl(xy) &= (xy)(1+\delta), \\ fl(\frac{x}{y}) &= \frac{x}{y}(1+\delta). \end{aligned}$$

La chose est évidemment différente si les nombres x et y sont eux-mêmes déjà des arrondis obtenus par des opérations précédentes. Il faudra alors évaluer, par exemple, $fl(fl(x) + fl(y))$.

Exemple 2.3 Dans cet exemple, nous montrons qu'aucune borne ne peut être déduite sur l'erreur relative faite de la soustraction de deux nombres

arrondis. Nous calculons en effet $z = fl(fl(x) - fl(y))$. Nous dénotons par ϵ_M l'epsilon machine. Nous obtenons successivement

$$\begin{aligned}
z &= fl(fl(x) - fl(y)) \\
&= (1+\delta_3)((1+\delta_1)x - (1+\delta_2)y) && \text{pour } |\delta_1|, |\delta_2|, |\delta_3| \leq \epsilon_M \\
&= x - y + \delta_1 x - \delta_2 y + \delta_3 x - \delta_3 y + \delta_1\delta_3 x - \delta_2\delta_3 y \\
&\approx (x-y) + x(\delta_1 + \delta_3) - y(\delta_2 + \delta_3) && \text{puisque } \delta_1\delta_3 x + \delta_1 x \approx \delta_1 x \\
& && \text{et } \delta_2\delta_3 y + \delta_2 y \approx \delta_2 y
\end{aligned}$$

Nous sommes à présent en mesure d'évaluer l'erreur relative.

$$\begin{aligned}
\left|\frac{(x-y)-z}{x-y}\right| &\approx \left|\frac{x(\delta_1+\delta_3) - y(\delta_2+\delta_3)}{x-y}\right| \\
&= \left|\frac{(x-y)\delta_3 + x\delta_1 - y\delta_2}{x-y}\right| \\
&= \left|\delta_3 + \frac{x\delta_1 - y\delta_2}{x-y}\right|. \qquad (2.1)
\end{aligned}$$

On peut remarquer que cette dernière expression ne peut pas être bornée a priori. En effet, en particulier pour des x et y proches, la présence de $x - y$ au dénominateur peut faire craindre le pire. Mais même si on fixe $x - y = 1$ par exemple, l'expression peut s'avérer extrêmement élevée. Il suffit pour s'en convaincre de prendre x et y beaucoup plus grands que $\frac{1}{\epsilon_M}$. Comme $|\delta_1| \leq \epsilon_M$ et $|\delta_2| \leq \epsilon_M$, on pourrait, par exemple, avoir $\delta_1 = \epsilon_M$ et $\delta_2 = 0$. Dans ce cas, la valeur de (2.1) est de $\delta_3 + x\epsilon_M$. Comme x n'est pas borné (cas de $x \gg \frac{1}{\epsilon_M}$), l'expression n'est elle-même pas bornée. ■

Comme le dernier exemple le suggère, la soustraction de deux nombres proches est une opération extrêmement délicate. C'est le sujet principal de la section suivante.

2.4 Perte de précision

Illustrons tout d'abord la perte de précision qui peut résulter de la soustraction de deux nombres proches par un exemple.

Exemple 2.4 Considérons la fonction $f(x) = \sin x - x$ à évaluer au voisinage de 0. Par exemple, évaluons séparément $\sin x$ et x pour $x = 0.05$ avec 8 chiffres

de précision. Nous avons

$$\begin{aligned} fl(x) &= 5.0000000\ 10^{-2} \\ fl(\sin x) &= 4.9979169\ 10^{-2} \\ \hline fl(x) - fl(\sin x) &= 0.0020831\ 10^{-2} \\ fl(fl(x) - fl(\sin x)) &= 2.0831000\ 10^{-5}. \end{aligned}$$

On voit qu'à la dernière étape, pour stocker le résultat final en format standard de virgule flottante, l'ordinateur doit rajouter trois zéros aux trois dernières positions. Ces trois zéros n'ont évidemment aucune précision quelconque. En particulier, on peut dire que seuls les cinq premiers chiffres de la mantisse sont réellement significatifs. ■

Le théorème suivant quantifie la perte de précision résultant de la soustraction de deux nombres proches.

Théorème 2.2 *Soient $x > y > 0$ deux nombres représentés en format standard de virgule flottante. Si $10^{-p} \leq 1 - \frac{y}{x} \leq 10^{-q}$ pour deux entiers p et q, alors il y a au plus p et au moins q décimales significatives qui sont perdues lors de la soustraction $x - y$.*

Il y a plusieurs façons de réaliser des soustractions "dangereuses" de manière plus sûre.

La première méthode consiste, pour certaines opérations, à travailler avec plus de décimales dans la mantisse. Ce n'est malheureusement pas toujours la panacée. Travailler avec une plus grande précision peut être extrêmement coûteux en mémoire et en temps de calcul. On peut bien sûr garder une plus grande précision pour les calculs que l'on sait problématiques, mais on peut ne pas alors avoir à sa disposition les décimales manquantes de nombres calculés plus tôt. Enfin, travailler avec plus de précision ne fait que reporter le problème. Pour des nombres proches mais plus grands, des pertes de précision continueront à se produire.

Une deuxième méthode consiste à tirer parti, cette fois, des formules analytiques. Dans certains cas, on peut en effet évaluer certaines fonctions par des formules équivalentes mais plus stables numériquement. Illustrons cette méthode par quelques exemples.

(i) Soit $f(x) = \sqrt{x^2+1} - 1$ à évaluer au voisinage de 0. En réécrivant, on obtient

$$f(x) = (\sqrt{x^2+1} - 1)\left(\frac{\sqrt{x^2+1}+1}{\sqrt{x^2+1}+1}\right) = \frac{x^2}{\sqrt{x^2+1}+1}.$$

Dans la dernière expression, toute soustraction périlleuse a disparu.

(ii) Soit $g(x) = x - \sin x$ à évaluer au voisinage de 0. En utilisant l'expansion en série de Taylor de $\sin x$, on peut écrire

$$g(x) = x - (x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots) = \frac{x^3}{3!} - \frac{x^5}{5!} + \cdots . \tag{2.2}$$

Pour x proche de 0, il peut être plus stable d'utiliser (2.2) plutôt qu'un calcul direct. Une utilisation intelligente du Théorème 2.2 permet de savoir quelles sont les valeurs pour lesquelles il est plus sage d'utiliser la formule (2.2).

(iii) Soit $h(x) = \ln x - 1$ à évaluer au voisinage de e. A nouveau, une manipulation analytique permet d'éviter de devoir effectuer la soustraction de deux nombres proches. On aura donc

$$h(x) = \ln x - 1 = \ln x - \ln e = \ln\left(\frac{x}{e}\right). \tag{2.3}$$

Le Théorème 2.2 permet de savoir les valeurs de x pour lesquelles utiliser (2.3) est préférable.

Chapitre 3

Interpolations et Régressions

Dans ce chapitre, nous abordons le problème de l'approximaion d'une fonction inconnue mais que l'on connaît en nombre fini de points. C'est un problème fondamental d'analyse numérique car dans la plupart des méthodes numériques, on recherche une fonction que l'on n'est capable que d'approximer en des points discrets. C'est aussi un problème qui apparaît, par exemple, dans le cas de mesures d'un phénomène physique ou chimique. On ne sera en général pas capable d'obtenir les mesures en temps continu, mais seulement en un nombre prédéterminé de temps discrets où l'on aura décidé de mesurer le phénomène. Ce chapitre est donc intéressant en soi mais consiste également à créer une base dont on pourra se servir ultérieurement.

Soit une fonction inconnue $u(x)$ mais que l'on connaît aux n points $(x_1, u(x_1)), (x_2, u(x_2)), \ldots, (x_n, u(x_n))$. Il y a plusieurs techniques qui existent pour tenter d'approximer $u(x)$. Selon le type d'application nous concernant, nous chosirons donc une technique différente. Dans ce cours, nous nous concentrerons uniquement sur deux techniques qui cherchent une approximation $\tilde{u}(x)$ de $u(x)$ qui satisfasse $\tilde{u}(x_i) = u(x_i)$ pour tout $i = 1, \ldots, n$.

(i) La première approche repose sur le fait que par n points du plan, on peut faire passer exactement un polynôme de degré $n-1$. Nous allons donc élaborer une formule qui établit ce polynôme de degré $n-1$. Nous verrons que cette approche, bien que très importante pour des petits degrés, a un très mauvais comportement pour des degrés élevés et approxime en général très mal la fonction $u(x)$.

(ii) La deuxième approche consiste toujours à approximer $u(x)$ par une autre fonction $\tilde{u}(x)$. Pour trouver $\tilde{u}(x)$, nous conserverons néanmoins

la contrainte que $\tilde{u}(x_i) = u(x_i)$ pour tout i. Mais nous rajouterons des contraintes qui imposeront un certain caractère lisse à la fonction $\tilde{u}(x)$. C'est ce qu'on appelle l'interpolation par *splines*.

3.1 Interpolation polynomiale

Soit $u : \mathbb{R} \mapsto \mathbb{R}$ une fonction inconnue, mais donnée en n points $x_1, \ldots, x_n$. On recherche un polynôme de degré $n-1$,

$$P(x) = \sum_{i=0}^{n-1} a_i x^i \tag{3.1}$$

qui satisfait $P(x_i) = u(x_i)$ pour tout $i = 1, \ldots, n$. Premièrement, il est utile de remarquer que ce problème a une solution unique si tous les x_i sont différents.

Théorème 3.1 *Soit un ensemble de n paires $(x_i, u(x_i))$. Si $x_i \neq x_j$ pour tout $i \neq j$, alors il existe un et un seul polynôme $P(x)$ de degré au plus $n-1$ qui satisfait $P(x_i) = u(x_i)$ pour $i = 1, \ldots, n$.*

Démonstration: Pour le moment, nous ne prouvons que l'unicité de la solution. L'existence sera implicitement prouvée par la forme de Lagrange que nous verrons dans la Section 3.1.2. Supposons dès lors qu'il existe deux polynômes de degré au plus $n-1$ $P(x)$ et $Q(x)$ différents qui interpolent les mêmes n points. Nous pouvons alors définir le polynôme $R(x) = P(x) - Q(x)$ qui est donc, également, de degré au plus $n-1$. Par ailleurs, nous avons que $R(x_i) = 0$ pour $i = 1, \ldots, n$. Cela signifie en particulier que R a n racines. Or, le seul polynôme de degré au plus $n-1$ ayant n racines est $R(x) = 0$, ce qui prouve l'unicité de la solution. ■

3.1.1 Matrice de Vandermonde

La façon la plus naturelle pour trouver les coefficients a_i de (3.1) est d'écrire le système d'équations que l'on obtient en écrivant ce qui se passe

aux points x_i. On doit donc résoudre

$$\begin{pmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} u(x_1) \\ u(x_2) \\ \vdots \\ u(x_n) \end{pmatrix}. \tag{3.2}$$

Le membre de gauche de (3.2), que nous dénotons par $V^n(x)$ est également appelée matrice de *Vandermonde*. On peut montrer que pour des x_i différents, cette matrice n'est jamais singulière (voilà une autre façon de prouver le Théorème 3.1). En effet, on peut prouver que

$$\det(V^n(x)) = \prod_{1 \le i < j \le n} (x_i - x_j).$$

Le sytème est en réalité très instable numériquement et il n'est pas recommandé de le résoudre directement. Nous pouvons analyser $\det(V^n(x))$ en supposant que nous cherchons à approximer une fonction $u(x)$ sur l'intervalle $[0, 1]$. Nous supposons également que nous interpolons $u(x)$ sur les abscisses $\frac{k}{n}$ pour $k = 1, \ldots, n$. Nous avons donc

$$\begin{aligned} |\det(V^n(x))| &= \prod_{1 \le i < j \le n} \frac{j-i}{n} \\ &= \frac{(n-1)(n-2)^2(n-3)^3 \cdots 2^{n-2}}{n^{\frac{n(n-1)}{2}}} \\ &\le \frac{n n^2 n^3 \cdots n^{n-2}}{n^{\frac{n(n-1)}{2}}} \\ &= \frac{n^{\frac{(n-2)(n-1)}{2}}}{n^{\frac{n(n-1)}{2}}} \\ &= \frac{1}{n^{n-1}}. \end{aligned}$$

La dernière expression montre que le déterminant de la matrice de Vandermonde, dans le cas considéré, tend rapidement vers 0. Il s'ensuit que le système sera très proche d'être singulier et par conséquent sujet à de nombreuses erreurs numériques. On préfèrera donc, en général, utiliser des formules directes pour calculer le polynôme d'interpolation. Nous en dérivons deux exemples dans les deux sections suivantes.

3.1.2 Formule de Lagrange

La formule de Lagrange permet de calculer directement le polynôme d'interpolation. Pour la dériver, définissons d'abord la fonction

$$\begin{aligned} l_i(x) &= \frac{(x-x_1)(x-x_2)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_1)(x_i-x_2)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} \\ &= \frac{\prod_{k\neq i}(x-x_k)}{\prod_{k\neq i}(x_i-x_k)}. \end{aligned}$$

Premièrement remarquons que $l_i(x)$ est un polynôme de degré $n-1$. Il suffit de remarquer que le dénominateur n'est, en fait, rien d'autre qu'un nombre réel non nul. Ensuite, nous voyons que l_i satisfait

$$\begin{aligned} l_i(x_i) &= 1 \\ l_i(x_k) &= 0 \quad \text{pour tout } k \neq i. \end{aligned}$$

Il s'ensuit que le polynôme

$$P(x) = \sum_{i=1}^{n} u(x_i) l_i(x)$$

est la solution unique de notre problème. C'est ce qu'on appelle le *polynôme d'interpolation de Lagrange*. La formule est aisée à dériver mais requiert néanmoins pas mal de calculs. Nous présentons dans la section suivante une autre formule pour trouver le polynôme d'interpolation qui est un peu plus économique en temps de calcul.

3.1.3 Formule de Newton

Pour établir le polynôme d'interpolation, la formule de Newton procède en quelque sorte par induction. On va créer n polynômes $p_0, p_1, \ldots, p_{n-1}$ de degrés respectifs $0, 1, \ldots, n-1$. L'idée est que le polynôme p_k doit interpoler les $k+1$ premiers points.

On a naturellement $p_0(x) = u(x_1)$.
Ensuite, on construit $p_1(x) = p_0(x) + c_1(x - x_1)$. On a clairement $p_1(x_1) = u(x_1)$. Afin d'obtenir le deuxième point d'interpolation, on calcule $p_0(x_2) + c_1(x_2 - x_1) = u(x_2)$. En d'autres termes, on obtient

$$c_1 = \frac{u(x_2) - u(x_1)}{x_2 - x_1}. \tag{3.3}$$

Si on continue de la sorte, on construira en général

$$p_i(x) = p_{i-1}(x) + c_i(x - x_1)(x - x_2)\cdots(x - x_i).$$

A nouveau, $p_i(x)$ satisfait, par construction, l'ordonnée aux i premières abscisses d'interpolation. Pour obtenir le coefficient c_i, on calculera $p_i(x_{i+1})$, c'est-à-dire

$$c_i = \frac{u(x_{i+1}) - p_{i-1}(x_{i+1})}{(x_{i+1} - x_1)\cdots(x_{i+1} - x_i)}.$$

La méthode décrite précédemment permet de dériver assez rapidement le polynôme d'interpolation. Il est encore possible de simplifier le calcul en utilisant une jolie propriété des coefficients.

Théorème 3.2 *Le polynôme d'interpolation est donné par*

$$P(x) = \sum_{i=1}^{n} u[x_1, \ldots, x_i](x - x_1)\cdots(x - x_{i-1}),$$

où on définit

$$u[x_i] = u(x_i)$$
$$u[x_i, \ldots, x_j] = \frac{u[x_{i+1}, \ldots, x_j] - u[x_i, \ldots, x_{j-1}]}{x_j - x_i}.$$

On voit que le théorème donne une valeur du coefficient en accord avec ce que donne (3.3). Les coefficients $u[x_i, \ldots, x_j]$ sont également appelés *différences divisées*. Ils sont très aisément obtenus en remplissant un tableau commençant par les $u[x_i]$ et en procédant ensuite par induction.

3.1.4 Erreur d'interpolation

Supposons que l'on interpole des données $(x_1, u(x_1)), \ldots, (x_n, u(x_n))$ par un polynôme $P(x)$ de degré $n-1$. Il est important de savoir quelle est l'erreur que l'on fait sur la fonction $u(x)$ lorsque l'on utilise $P(x)$ à sa place. En d'autres termes, on s'intéresse à la fonction $e(x) = u(x) - P(x)$. Le théorème suivant indique l'erreur réalisée.

Théorème 3.3 *Soit $u : \mathbb{R} \mapsto \mathbb{R}$ et $P(x)$ un polynôme interpolant les points $(x_1, u(x_1)), \ldots, (x_n, u(x_n))$. Si on suppose que $x_i \in [a, b]$ pour tout i, et que*

$u(x)$ est n fois continûment dérivable sur $[a,b]$, alors pour tout $x \in [a,b]$, il existe $\xi \in [a,b]$ tel que

$$e(x) = u(x) - P(x) = \frac{u^{(n)}(\xi)}{n!}(x - x_1)\cdots(x - x_n). \tag{3.4}$$

Démonstration: Il est évident que l'erreur d'interpolation sera nulle, par définition, aux points x_i. C'est bien le cas dans (3.4). Considérons maintenant $x \in [a,b]$ fixé avec $x \neq x_i$. Définissons

$$\begin{aligned} w(t) &= (t - x_1)\cdots(t - x_n) = \prod_{i=1}^{n}(t - x_i), \\ c &= \frac{u(x) - P(x)}{w(x)} = \frac{e(x)}{w(x)}, \\ \phi(t) &= u(t) - P(t) - cw(t). \end{aligned}$$

Observons que w définit un polynôme de degré n, que c est une constante car nous avons fixé x et que le dénominateur de c ne s'annule pas lorsque $x \neq x_i$ pour tout i. Occupons-nous à présent de la fonction $\phi(t)$ définie sur l'intervalle $[a,b]$. Remarquons tout d'abord que $\phi(x_i) = 0$ pour tout i. En effet, $u(x_i) = P(x_i)$ et clairement $w(x_i) = 0$. De plus, on a $\phi(x) = 0$ pour le x que nous avons fixé. Le Théorème de Rolle affirme que pour toute fonction f continue sur $[y_1, y_2]$ telle que $f(y_1) = f(y_2)$, il existe $y_3 \in [y_1, y_2]$ tel que $f'(y_3) = 0$. Nous pouvons dès lors appliquer le Théorème de Rolle sur les intervalles entre les $n+1$ racines de ϕ ($x_1, x_2, \ldots, x_n$ et x) et nous trouvons dès lors n racines de $\phi'(t)$. Par une nouvelle application du Théorème de Rolle, on trouve que $\phi^{(2)}(t)$ a $n-1$ racines et ainsi de suite. Finalement, on en conclut que $\phi^{(n)}(t)$ a une racine sur l'intervalle $[a,b]$. Soit $\xi \in [a,b]$ une racine de $\phi^{(n)}(t)$. Nous avons donc

$$0 = \phi^{(n)}(\xi) = u^{(n)}(\xi) - P^{(n)}(\xi) - cw^{(n)}(\xi). \tag{3.5}$$

Dans (3.5), $P^{(n)} = 0$ car P est un polynôme de degré au plus $n-1$. De plus, comme $w(t)$ est un polynôme de degré n dont le coefficient de t^n est 1, on en déduit que $w^{(n)}(t) = n!$. On peut dès lors réécrire (3.5) comme

$$0 = u^{(n)}(\xi) - cn! = u^{(n)}(\xi) - \frac{n!\ e(x)}{(x - x_1)\cdots(x - x_n)},$$

ce qui est exactement l'expression désirée en (3.4). ∎

La question est de savoir comment interpréter un tel théorème. Premièrement, il est aisé de voir que la borne de l'erreur aux points d'interpolation est nulle. C'est déjà rassurant ! Ensuite, on voit que pour calculer une borne sur l'erreur, il faudra connaître une borne sur la n^e dérivée de u. Enfin, le facteur $(x-x_1)\cdots(x-x_n)$ fait fortement osciller la borne de l'erreur lorsque x varie sur l'intervalle. C'est une des raisons majeures pour laquelle une interpolation polynomiale de haut degré est rarement une bonne approximation d'une fonction. Mais nous analysons cela plus en détails dans les sections suivantes.

3.1.5 Choix des abscisses d'interpolation

Il n'est pas évident a priori que le choix des abscisses d'interpolation joue un rôle crucial dans la qualité de l'approximation d'une fonction par son interpolation. On peut pourtant voir dans l'expression (3.4), que l'erreur dépend du choix des abscisses par l'intermédiaire du facteur $(x-x_1)\cdots(x-x_n)$. Cette dépendance peut même être très forte comme nous allons le voir maintenant. L'exemple suivant est classique en analyse numérique.

Exemple 3.1 (Phénomène de Runge) Considérons la fonction $u(x) = \frac{1}{1+x^2}$ sur l'intervalle $[-5,5]$. Dans un premier temps, nous prenons l'option assez naturelle de l'interpoler sur des abscisses équidistantes. Afin de comparer le choix entre peu et beaucoup de points d'interpolation, nous interpolons la fonction pour 3, 6, 11 et 21 points équidistants respectivement. Le résultat de l'interpolation de Lagrange est reporté sur la Figure 3.1. Sur la figure, la fonction $u(x)$ est représentée en trait pointillé tandis que l'interpolation est reportée en trait plein. La première importante constatation est que l'interpolation ne converge pas vers la fonction lorsque le nombre de points augmente. Nous avons en effet

$$\lim_{n\to\infty} \max_{x\in[-5,5]} |P_n(x) - u(x)| = \infty,$$

où $P_n(x)$ est le polynôme de degré $n-1$ obtenu à partir de l'interpolation de n points équidistants. Par ailleurs, on voit que l'interpolation est particulièrement mauvaise aux extrémités de l'intervalle. Si on s'éloigne de l'intervalle, l'approximation serait évidemment encore plus mauvaise. Il est à noter que ce phénomène n'est pas seulement dû au cas pathologique de la fonction étudiée ici. Des oscillations de plus en plus éloignées de la fonction se produisent en général lorsque l'on interpole une fonction par un polynôme de degré très haut.

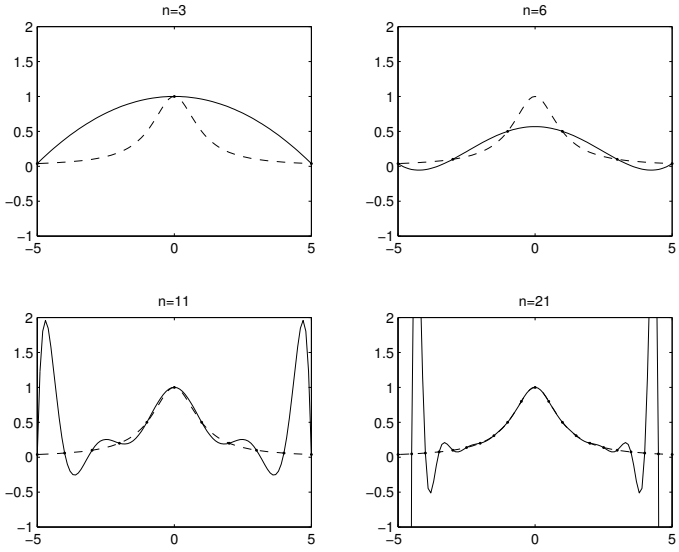

FIGURE 3.1 – Interpolation de la fonction de Runge avec des abscisses équidistantes

Nous allons étudier dans cette section l'interpolation aux abscisses de Chebyshev. Pour motiver cette étude, voyons ce que cela donne dans le cas de la fonction étudiée ici. L'idée de cette interpolation est de choisir les abscisses de manière à optimiser l'erreur, en particulier l'erreur induite par le facteur $(x - x_1) \cdots (x - x_n)$. Pour ce faire, on doit choisir plus d'abscisses aux extrémités de l'intervalle. Nous avons, à nouveau, interpolé la fonction à partir de 3, 6, 11 et 21 points. Le résultat de l'interpolation est renseigné à la Figure 3.2. A nouveau, la fonction est représentée en pointillé tandis que les interpolations sont reportées en trait plein. On voit que dans ce cas-ci, l'interpolation est bien meilleure sur l'intervalle considéré. Elle converge même lorsque n tend vers l'infini. Il est bien entendu qu'en dehors de l'intervalle, l'approximation peut devenir très mauvaise. Voyez à ce sujet la Figure 3.3 qui reporte les valeurs sur l'intervalle $[-10, 10]$. ■

Comme nous l'avons suggéré dans l'Exemple 3.1, il est intéressant d'analyser une fois encore l'expression (3.4) afin de déterminer les abscisses d'interpolation qui donneront lieu à l'erreur minimale. On voit que, pour n et u fixés, la borne de l'erreur ne varie qu'en fonction de $(x-x_1) \cdots (x-x_n)$. Il est

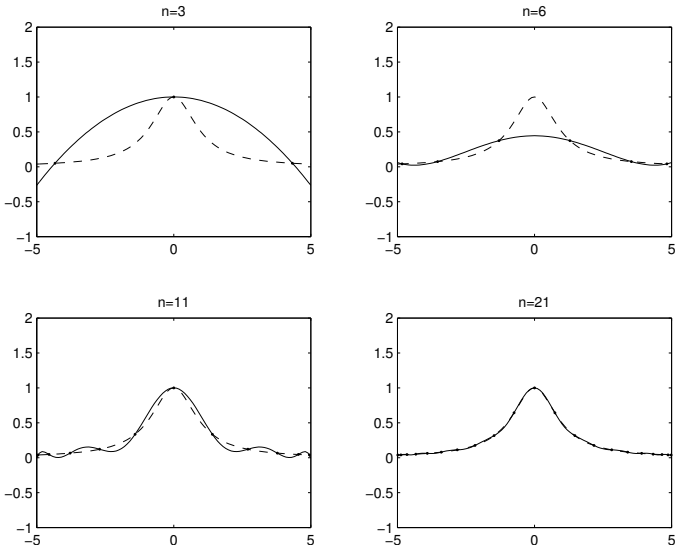

FIGURE 3.2 – Interpolation de la fonction de Runge avec les abscisses de Chebyshev

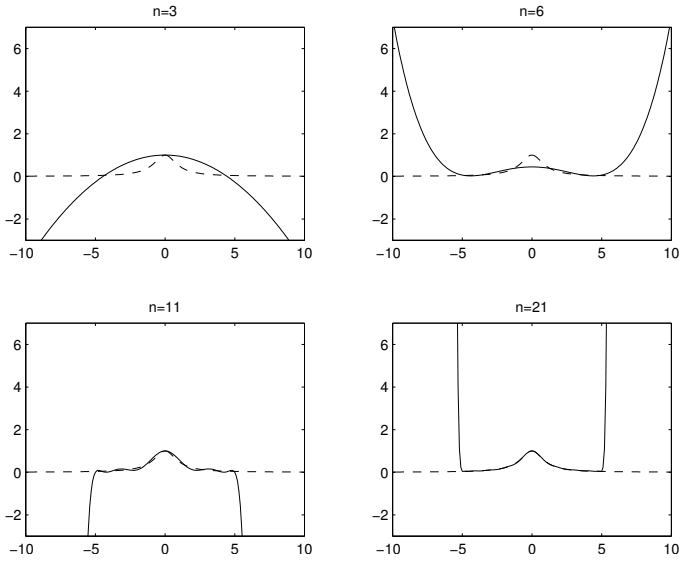

FIGURE 3.3 – Fonction de Runge et ses interpolations sur l'intervalle $[-10, 10]$

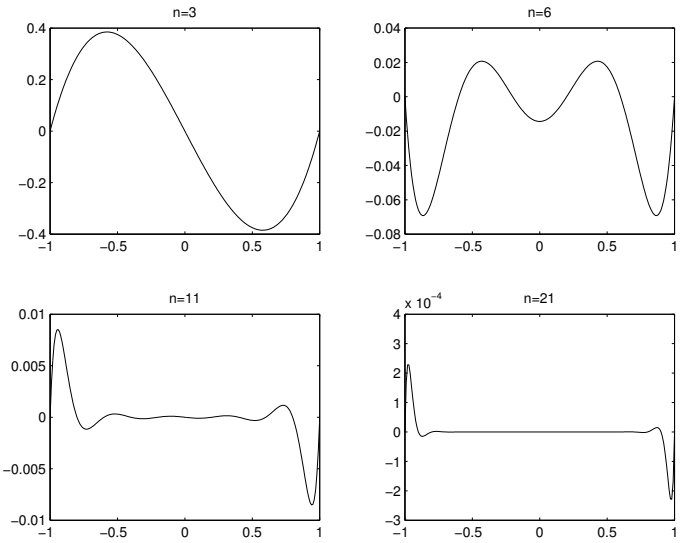

FIGURE 3.4 – Polynôme $(x-x_1)\cdots(x-x_n)$ pour des abscisses équidistantes

aussi important de noter que cette valeur change lorsque l'on change le choix des n abscisses. Pour le choix d'abscisses équidistantes, on peut se demander pourquoi l'erreur est élevée aux bords de l'intervalle. Pour ce faire, nous allons représenter le produit $(x-x_1)\cdots(x-x_n)$. La Figure 3.4 représente la valeur du produit $(x-x_1)\cdots(x-x_n)$ pour les 4 choix de n opérés dans l'exemple. On voit que proportionnellement, l'erreur est plus élevée sur les bords de l'intervalle qu'au centre. Ceci est en particulier vrai lorsque n augmente. Les polynômes de Chebyshev sont des polynômes qui minimisent leur valeur maximale atteinte sur l'intervalle. En particulier, le polynôme de Chebyshev de degré n a n racines sur l'intervalle $[-1,1]$ et $n-1$ minima et maxima. Une propriété importante est que tous ces minima et maxima atteignent exactement la même valeur (en valeur absolue). De cette façon, il n'y a aucune partie de l'intervalle où l'interpolation est extrêmement mauvaise comparée au reste de l'intervalle. Nous introduisons à présent les polynômes de Chebyshev.

Définition 3.1 *Le polyôme de Chebyshev de degré n,* $T_n : [-1,1] \mapsto \mathbb{R}$ *est défini comme*

$$T_n(x) = \cos(n \arccos(x)). \tag{3.6}$$

Au vu de la définition, il n'est pas évident que $T_n(x)$ est un polynôme. En

effet, $T_n(x)$ ne peut être défini qu'à partir de l'intervalle $[-1,1]$. Néanmoins, les propriétés suivantes vont montrer que T_n se ramène à un polynôme sur l'intervalle $[-1,1]$.

Proposition 3.1 *Sur l'intervalle* $[-1,1]$*, nous avons*

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \tag{3.7}$$

pour tout $n = 1, 2, \ldots$ *De plus,* $T_0(x) = 1$ *et* $T_1(x) = x$.

Démonstration: Clairement, on a $T_0(x) = \cos(0) = 1$ et $T_1(x) = \cos(\arccos(x)) = x$ pour $x \in [-1,1]$. Pour prouver (3.7), nous utilisons une formule d'addition de la trigonométrie, à savoir,

$$\cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta).$$

On obtient alors respectivement

$$T_{n+1}(x) = \cos(\arccos(x))\cos(n\arccos(x)) - \sin(\arccos(x))\sin(n\arccos(x)) \quad \text{et} \tag{3.8}$$

$$T_{n-1}(x) = \cos(\arccos(x))\cos(n\arccos(x)) + \sin(\arccos(x))\sin(n\arccos(x)). \tag{3.9}$$

En additionnant (3.8) et (3.9), on obtient

$$T_{n+1}(x) + T_{n-1}(x) = 2\cos(\arccos(x))\cos(n\arccos(x)).$$

Puisque $\cos(\arccos(x)) = x$ sur l'intervalle $[-1,1]$ et que $\cos(n\arccos(x)) = T_n(x)$, on obtient (3.7). ∎

Proposition 3.2 *Sur l'intervallle* $[-1,1]$*,* $T_n(x)$ *est un polynôme de degré* n.

Démonstration: Procédons par induction sur n.
On a clairement que $T_0(x) = 1$ et $T_1(x) = x$ sont des polynômes. De plus, si $T_{n-1}(x)$ et $T_n(x)$ sont des polynômes, par la formule de récurrence (3.7), il s'en suit que $T_{n+1}(x)$ est également un polynôme. ∎

Nous avons déclaré plus haut qu'il existe une propriété importante des extrema des polynômes de Chebyshev. Nous la précisons dans la Proposition suivante.

Proposition 3.3 *Pour n impair, $T_n(x)$ a, sur l'intervalle $[-1, 1]$, $(n+1)/2$ maxima où $T_n(x)$ vaut 1 et $(n+1)/2$ minima où la fonction vaut -1.*
Pour n pair, $T_n(x)$ a, sur l'intervalle $[-1, 1]$, $(n+2)/2$ maxima et $n/2$ minima où la valeur absolue de $T_n(x)$ vaut 1.

Démonstration: Supposons n impair. Comme $T_n(x)$ est de degré n, nous savons que $T'_n(x)$ est de degré $n-1$ et a, dès lors, au plus $n-1$ racines. Par conséquent, si on compte les $n-1$ points où la dérivée de $T_n(x)$ s'annule et les deux extrémités de l'intervalle, on en déduit que $T_n(x)$ a au plus $n+1$ extrema. De plus, au vu de (3.6), nous avons $-1 \leq T_n(x) \leq 1$. Nous allons montrer qu'il existe $(n+1)/2$ points pour lesquels $T_n(x)$ vaut 1 et $(n+1)/2$ points pour lesquels $T_n(x)$ vaut -1. Ces points seront par conséquent les uniques extrema de $T_n(x)$. Cherchons d'abord les maxima. On recherche $x \in [-1, 1]$ tel que

$$\cos(n \arccos(x)) = 1.$$

On a donc

$$n \arccos(x) = 2k\pi, \qquad \text{avec } k = 0, 1, \ldots$$

Dès lors, les $(n+1)/2$ maxima sont donnés par

$$x = \cos\left(\frac{2k\pi}{n}\right), \qquad \text{avec } k = 0, \ldots, \frac{n-1}{2}.$$

Similairement, on a que les $(n+1)/2$ minima sont donnés par

$$x = \cos\left(\frac{(2k+1)\pi}{n}\right), \qquad \text{avec } k = 0, \ldots, \frac{n-1}{2}.$$

Le cas où n est pair est similaire. ∎

La Figure 3.5 représente le polynôme de Chebyshev pour quatre valeurs de n. On voit que, comparé à la Figure 3.4, l'erreur dans le cas de l'utilisation des abscisses de Chebyshev, est répartie équitablement sur tout l'intervalle. On peut prouver que, de cette façon, l'erreur que l'on fait est minimale. Il est maintenant temps de déterminer quelles sont les abscisses des racines des polynômes de Chebyshev.

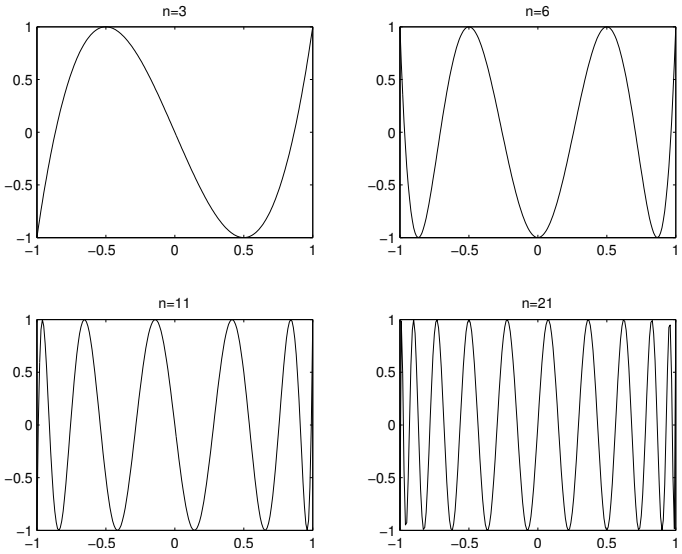

FIGURE 3.5 – Polynôme de Chebyshev pour quatre valeurs de n

Proposition 3.4 *Les n racines de $T_n(x)$, sur l'intervalle $[-1,1]$ sont*

$$x_k = \cos\left(\frac{(2k+1)\pi}{2n}\right) \qquad k = 0, \ldots, n-1. \tag{3.10}$$

Démonstration: Nous cherchons $x \in [-1,1]$, tel que

$$n \arccos(x) = \frac{\pi}{2} + k\pi \qquad k = 0, 1, \ldots$$

Par conséquent, on a que

$$x_k = \cos\left(\frac{\pi}{2n} + \frac{k\pi}{n}\right) \qquad k = 0, \ldots, n-1,$$

ce qui peut se réécrire comme (3.10). ∎

L'interpolation aux abscisses de Chebyshev consiste à utiliser les valeurs données par (3.10) pour interpoler. Nous avons vu sur l'Exemple 3.1 que cette manière de faire est plus stable. Lorsque l'intervalle d'interpolation

n'est pas $[-1, 1]$, on peut, par un simple changement de variables, se ramener au cas standard. Ainsi si l'on travaille sur l'intervalle $[a, b]$, les abscisses de Chebyshev sont données par

$$x_k = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(2k+1)\pi}{2n}\right) \qquad k = 0, \ldots, n-1.$$

3.2 Interpolation par splines

Le message principal à retenir de la section précédente est que l'interpolation de nombreux points par un polynôme de très haut degré donne souvent des résultats décevants. On peut néanmoins vouloir faire passer une courbe par un certain nombre de points tout en demandant que la courbe soit suffisamment lisse et ne contienne pas d'oscillations non souhaitées. C'est le thème de cette section. Ces courbes sont appelées splines. Elles peuvent être utiles, par exemple, dans le design. Le designer fixe quelques points et demande à l'ordinateur de déterminer une courbe "jolie" à l'oeil qui passe par tous ces points.

3.2.1 Interpolation par splines cubiques

Dans la section précédente, nous avons cherché un unique polynôme qui passe par tous les points. Dans cette section, nous allons réaliser l'interpolation *par morceaux*. En d'autres termes, entre chaque paire de points, nous aurons un polynôme différent. La première façon naturelle de faire passer une courbe par un certain nombre de points et de considérer des fonctions linéaires entre chaque paire de points consécutifs. Si on a initialement n points, la fonction sera alors décrite par $n-1$ fonctions linéaires *différentes*. La Figure 3.6 montre l'exemple d'une interpolation linéaire par morceaux. Ce genre d'interpolation peut être très pratique lorsqu'il s'agit d'approximer la valeur d'une fonction qui est connue expérimentalement pour de nombreux points. En effet, on peut montrer à l'aide de la formule d'erreur (3.4) de la section précédente appliquée à des polynômes de degré 1 que si on augmente le nombre de points, l'erreur commise va tendre vers 0. Néanmoins, la fonction est très rudimentaire et n'est certainement pas "jolie" (bien que cela soit là un critère particulièrement subjectif). Une façon simple de généraliser la simple interpolation linéaire par morceaux est toujours de considérer un polynôme différent pour chaque intervalle de points consécutifs. Mais au lieu

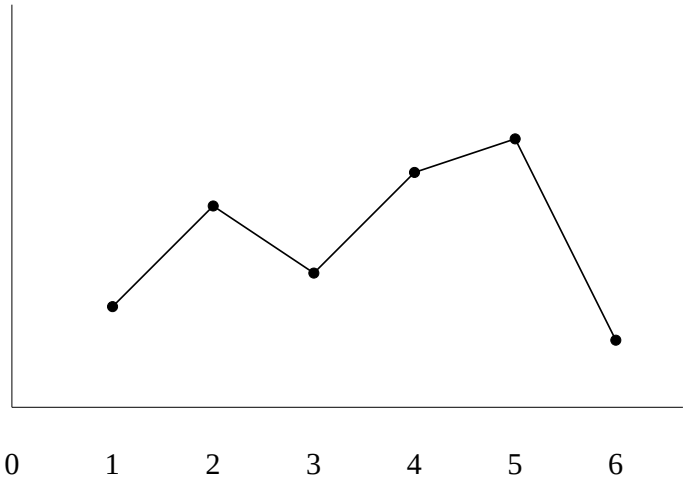

FIGURE 3.6 – Exemple d'interpolation linéaire par morceaux

de calculer un polynôme de degré 1 à chaque intervalle, on peut augmenter le degré du polynôme et imposer des conditions de *continuité* sur les dérivées successives. Le cas le plus courant en pratique est de considérer des polynômes de degré 3 à chaque intervalle. Dénotons par $p_i(x)$ le polynôme de degré 3 passant par les points $(x_i, u(x_i))$ et $(x_{i+1}, u(x_{i_+1}))$. Nous imposerons dès lors les conditions successives

$$p_i(x_i) = u(x_i) \qquad \text{pour } i = 1, \ldots, n-1 \tag{3.11}$$
$$p_i(x_{i+1}) = u(x_{i+1}) \qquad \text{pour } i = 1, \ldots, n-1 \tag{3.12}$$
$$p'_{i-1}(x_i) = p'_i(x_i) \qquad \text{pour } i = 2, \ldots, n-1 \tag{3.13}$$
$$p''_{i-1}(x_i) = p''_i(x_i) \qquad \text{pour } i = 2, \ldots, n-1. \tag{3.14}$$

Les $2n-2$ conditions (3.11)-(3.12) assurent la continuité de la fonction. Les $n-2$ conditions (3.13) et les $n-2$ conditions (3.14) assurent le caractère *lisse* de la courbe en jouant sur la continuité de la dérivée et de la dérivée seconde respectivement. On voit donc que l'on dispose de $4n-6$ conditions. Or on doit calculer $n-1$ polynômes de degré 3 $p_i(x) = a_{i,3}x^3 + a_{i,2}x^2 + a_{i,1}x + a_{i,0}$, c'est-à-dire que l'on a 4 coefficients à calculer pour chaque polynôme. Au total, cela fait donc $4n-4$ coefficients. Il nous manque donc deux conditions pour pouvoir déterminer une solution unique au problème. Plusieurs choix peuvent être faits :

(i) Le choix le plus typique consiste à imposer une courbure nulle au début et à la fin de l'intervalle, à savoir

$$p_1''(x_1) = 0, \qquad p_{n-1}''(x_n) = 0.$$

On appelle une *spline naturelle* la courbe ainsi calculée.

(ii) Un autre choix possible est d'imposer la même courbure au début et à la fin des premier et dernier intervalles, à savoir

$$p_1''(x_1) = p_1''(x_2), \qquad p_{n-1}''(x_{n-1}) = p_{n-1}''(x_n).$$

(iii) Dans certains cas, la pente est connue au début et à la fin de l'intervalle. On aura alors $p_1'(x_1) = u'(x_1)$ et $p_{n-1}'(x_n) = u'(x_n)$.

(iv) Finalement la solution choisie par matlab est d'imposer une dérivée troisième constante sur les deux premiers et les deux derniers intervalles respectivement. De manière équivalente, ceci revient à imposer le même coefficient de x^3 sur les deux premiers et les deux derniers intervalles.

La Figure 3.7 représente la juxtaposition de l'interpolation linéaire par morceaux (en pointillés) avec une spline cubique calculée pour les mêmes points d'interpolation (en trait plein).

3.2.2 Qualité de l'interpolation par spline cubique

Dans un certain sens, on peut dire que l'interpolation par spline cubique est l'interpolation la plus lisse que l'on puisse trouver. Dans cette section, nous prouvons que c'est en particulier l'inteprolation qui minimise les oscillations. Dans le cas de l'interpolation polynomiale, nous avons vu qu'un grand nombre d'oscillations non désirées apparaissaient. Cela induit en particulier une valeur élevée de la dérivée seconde du polynôme. La proposition suivante quantifie la qualité de l'interpolation par spline cubique.

Proposition 3.5 *Soit S la spline cubique naturelle interpolant aux points $(x_1, u(x_1)), \ldots, (x_n, u(x_n))$ une fonction $u(x)$ deux fois continûment dérivable. On a*

$$\int_{x_1}^{x_n} (S''(x))^2 dx \leq \int_{x_1}^{x_n} (u''(x))^2 dx. \tag{3.15}$$

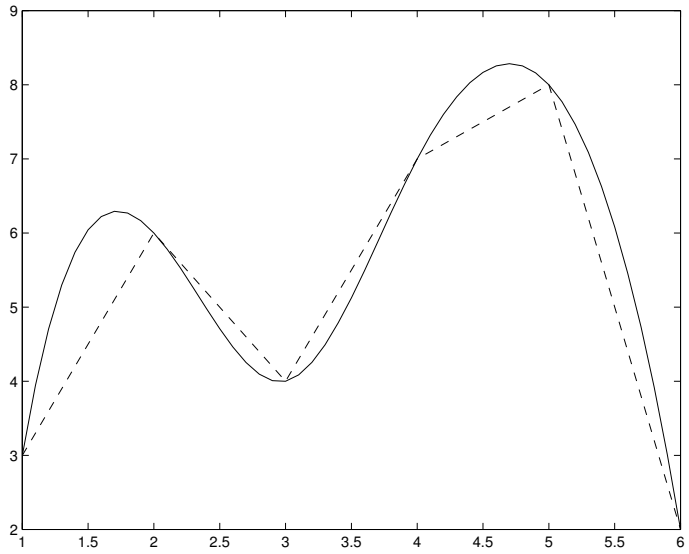

FIGURE 3.7 – Une spline cubique

Démonstration: Nous définissons l'erreur d'interpolation comme

$$e(x) = u(x) - S(x).$$

Comme u et S sont deux fois dérivables, nous avons $u''(x) = S''(x) + e''(x)$. Dès lors,

$$\int_{x_1}^{x_n} (u''(x))^2 dx = \int_{x_1}^{x_n} (S''(x))^2 dx + \int_{x_1}^{x_n} (e''(x))^2 dx + 2 \int_{x_1}^{x_n} S''(x) e''(x) dx. \tag{3.16}$$

Nous allons prouver que le dernier terme de cette expression est nul. En intégrant par parties, on obtient

$$\begin{aligned} \int_{x_1}^{x_n} S''(x) e''(x) dx &= [S''(x) e'(x)]_{x_1}^{x_n} - \int_{x_1}^{x_n} S'''(x) e'(x) dx \\ &= - \int_{x_1}^{x_n} S'''(x) e'(x) dx \end{aligned}$$

puisque, pour une spline cubique naturelle, on a que la dérivée seconde s'annule aux deux extrémités de l'intervalle. Si on continue de développer, on

obtient

$$\int_{x_1}^{x_n} S'''(x)e'(x)dx = \sum_{i=1}^{n-1} \int_{x_i}^{x_{i+1}} S'''(x)e'(x)dx.$$

Remarquons que, comme S est un polynôme de degré 3 sur chaque intervalle, sa dérivée troisième est une constante c_i sur chaque intervalle $[x_i, x_{i+1}]$. Nous en déduisons finalement que

$$\begin{aligned}\int_{x_1}^{x_n} S'''(x)e'(x) &= \sum_{i=1}^{n-1} c_i \int_{x_i}^{x_{i+1}} e'(x)dx \\ &= \sum_{i=1}^{n-1} c_i(e(x_{i+1}) - e(x_i)) \\ &= 0\end{aligned}$$

puisque l'erreur e s'annule pour tous les points x_i. Si on insère ce dernier résultat dans (3.16), on obtient maintenant

$$\begin{aligned}\int_{x_1}^{x_n} (u''(x))^2 dx &= \int_{x_1}^{x_n} (S''(x))^2 dx + \int_{x_1}^{x_n} (e''(x))^2 dx \\ &\geq \int_{x_1}^{x_n} (S''(x))^2 dx\end{aligned}$$

puisque l'intégrale définie du carré d'une fonction est positive. ■

Par conséquent, si on considère comme critère, de minimiser l'intégrale du carré de la dérivée seconde, tout en conservant des fonctions deux fois dérivables, la spline cubique naturelle est la meilleure interpolation possible.

Chapitre 4

Résolution d'équations non linéaires

Dans ce chapitre, nous passons en revue les principales méthodes numériques qui existent pour trouver une racine d'une équation ou d'un système non linéaire. Au delà des méthodes elles-mêmes, nous insisterons sur l'étude de leur comportement. En particulier, une question importante qui sera abordée est la notion de convergence des méthodes. C'est cette question cruciale qui peut être déterminante dans le choix d'une méthode par rapport à une autre.

Pour mettre les idées en place, et pour commencer, nous recherchons une racine de l'équation

$$f(x) = 0. \tag{4.1}$$

A l'exception de la recherche des racines d'un polynôme, il n'existe pas de méthode capable de rechercher *toutes les racines* de (4.1). Toutes les méthodes que nous développons ici sont itératives, c'est-à-dire qu'elles construisent une suite (x_n) avec la propriété que

$$\lim_{n\to\infty} x_n = \bar{x},$$

où $f(\bar{x}) = 0$. En règle générale, il est très difficile de prévoir vers quelle racine x une méthode converge. Si la racine trouvée n'est pas celle souhaitée, il faudra recommencer l'algorithme avec un autre point de départ.

Nous commençons ce chapitre par la méthode la plus simple et que nous avons également présentée brièvement dans l'introduction, à savoir la méthode de la dichotomie ou bisection.

4.1 Méthode de la bisection

Supposons que l'on cherche à localiser une racine d'une fonction continue $f(x)$. Si de plus, on connaît deux points a_0 et b_0 tels que $f(a_0) < 0$ et $f(b_0) > 0$. On suppose sans perte de généralité que $a_0 < b_0$. Dès lors, le théorème des valeurs intermédiaires nous garantit qu'une racine x de f existe sur l'intervalle $[a_0, b_0]$. L'algorithme de la bisection consiste à définir $x_i := \frac{a_{i-1}+b_{i-1}}{2}$. Si $f(x_i) < 0$, alors $a_i := x_i$ et $b_i := b_{i-1}$. Si, en revanche, $f(x_i) > 0$, alors $a_i := a_{i-1}$ et $b_i := x_i$. Il n'est pas difficile de prouver que la méthode de la bisection converge vers une racine x avec un taux de convergence linéaire.

Proposition 4.1 *Soit $\bar{x}$ une racine (supposée unique) de f sur l'intervalle $[a_0, b_0]$. Alors on a*

$$|\bar{x} - x_n| \leq \frac{(b_0 - a_0)}{2^n}.$$

Démonstration: La taille de l'intervalle $[a_i, b_i]$ est exactement divisée par 2 à chaque itération. De plus, comme la racine $\bar{x}$ se trouve soit dans la première, soit dans la deuxième partie de l'intervalle, on en déduit que la distance entre $\bar{x}$ et x_n est inférieure ou égale à la moitié de la taille de l'intervalle à l'itération n. Le résultat découle ensuite du fait que la taille de l'intervalle à l'itération n est égal à $\frac{b_0-a_0}{2^{n-1}}$. ∎

La méthode de la bisection admet donc un taux de convergence linéaire. Par souci de précision, nous rappelons ici la définition du taux de convergence qui avait été abordé dans l'introduction.

Définition 4.1 *Lorsque $y_n \to y$ avec $|y_n - y| \leq c_0 c^{pn}$, pour $c_0, p > 0$ et $0 < c < 1$, on dit que la suite (y_n) a un taux de convergence d'ordre p.*

Lorsque $p = 1$ comme dans le cas de la méthode de la bisection, on parle de convergence *linéaire*. Pour $p > 1$ en général, on parle de convergence *superlinéaire* et pour $p = 2$, de convergence quadratique.

Comme dit précédemment, la méthode de la bisection est très facile à mettre en oeuvre et admet une bonne stabilité. Le seul point négatif est qu'elle nécessite deux points de départ a_0 et b_0 qui peuvent être difficiles à trouver. Dans certains cas, de tels points n'existent même pas. Par exemple, pour une fonction qui admet une racine double, il est impossible de mettre la méthode de la bisection en oeuvre au voisinage de la racine. De plus, la méthode nécessite la continuité de la fonction. Des résultats erronés peuvent apparaître si on néglige ce point.

4.2 Méthode du point fixe

La force de la méthode de la bisection est sa robustesse. Sa faiblesse est le relatif mauvais ordre de convergence. La méthode du point fixe ne corrige pas cette faiblesse. Il est néanmoins intéressant de l'étudier car il s'agit d'une méthode générique. Nous verrons plus tard la méthode de Newton-Raphson qui en est une généralisation. On peut également utiliser une généralisation de la méthode de point fixe dans le cas de la résolution de systèmes linéaires. Supposons à nouveau que nous cherchions une solution à $f(x) = 0$. Par ailleurs, imaginons que nous arrivions à réécrire l'équation sous la forme $x = g(x)$. Arrêtons nous quelques instants sur cette réécriture. Il s'agit en réalité d'une opération plus simple qu'il n'y paraît sauf si nous ne disposons pas de la forme analytique de f. C'est ce que l'exemple suivant tente de montrer.

Exemple 4.1 Supposons que nous cherchions la racine de l'équation donnée dans le chapitre 1

$$x^3 + x - 1 = 0.$$

Il existe plusieurs façons de réécrire l'équation sous la forme $x = g(x)$. On écrira, par exemple,

$$\begin{aligned} x &= -x^3 + 1 \\ x &= \sqrt[3]{-x+1} \\ x &= x^3 + 2x - 1 \\ x &= \frac{1}{x^2+1}. \end{aligned}$$

Nous comparerons plus tard l'efficacité de ces différentes réécritures et nous verrons qu'elles sont loin d'être équivalentes pour les performances de la méthode du point fixe. Le but de cet exemple est surtout de montrer qu'il existe souvent un grand nombre de façons d'obtenir la forme $x = g(x)$. ■

La méthode du point fixe est encore plus simple à mettre en oeuvre que la méthode de la bisection. L'algorithme s'écrit en effet en une ligne

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

Il faut bien entendu choisir un point de départ x_1. Graphiquement, on part d'un point x_1, on calcule la valeur $g(x_1)$ (trait vertical), on reporte cette

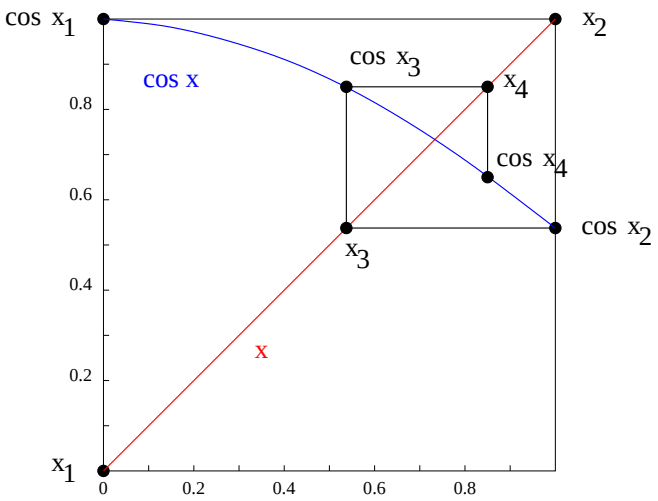

FIGURE 4.1 – 4 itérations de la méthode du point fixe pour $x = \cos x$

valeur sur l'abscisse (trait horizontal) et on itère. La Figure 4.1 représente quelques itérations de la méthode du point fixe dans le cas de l'équation $x = \cos x$. Le tableau 4.1 indique les 8 premières itérations. Sur la Figure et en faisant quelques itérations, on remarque que la méthode converge vers une solution de l'équation. Il n'en est pourtant pas toujours ainsi. Considérons à présent la troisième version que nous avons écrite dans l'Exemple 4.1 pour résoudre $x^3 + x - 1 = 0$. On va calculer des valeurs successives $x_{k+1} = x_k^3 + 2x_k - 1$. Si on part de $x_1 = 1$ ou de $x_1 = 0$, on obtient le tableau 4.2 avec les deux suites. Dans les deux cas, la méthode diverge assez rapidement. Ce sera d'ailleurs le cas pour tout point x_1 choisi (sauf la racine évidemment). Ce que l'on peut remarquer, c'est que c'est $|g'(x)|$ qui détermine si la méthode converge ou pas. Pour une valeur absolue inférieure à 1 au voisinage de la racine, la méthode peut être convergente. Dans le cas d'une valeur absolue supérieure à 1, la méthode diverge. C'est ce qu'indique intuitivement la Figure 4.2. Le théorème suivant formalise une condition suffisante de convergence relativement générale. Nous verrons par la suite que la condition intuitive $|g'(x)| < 1$ en est un cas particulier.

Théorème 4.1 *Soit $g(x)$ une fonction dont $\bar{x}$ est un point fixe $g(\bar{x}) = \bar{x}$. Considérons l'intervalle $I = \{x \mid |x - \bar{x}| \leq r\}$ pour $r > 0$. Si g satisfait la*

Itération	x_k	$\cos x_k$
1	0	1
2	1	0.5403
3	0.5403	0.8576
4	0.8576	0.6543
5	0.6543	0.7935
6	0.7935	0.7014
7	0.7014	0.7640
8	0.7640	0.7221

TABLE 4.1 – 8 itérations de la méthode du point fixe pour $x = \cos x$

Itération	x_k	$x_k^3 + 2x_k - 1$	Itération	x_k	$x_k^3 + 2x_k - 1$
1	1	2	1	0	-1
2	2	11	2	-1	-4
3	11	1352	3	-4	-73

TABLE 4.2 – 3 itérations divergentes de la méthode du point fixe

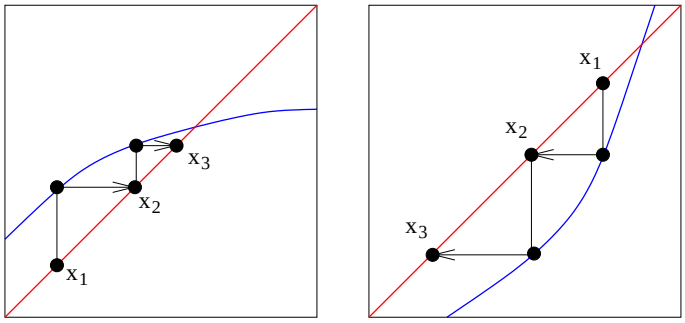

FIGURE 4.2 – La méthode converge pour $|g'(x)| < 1$ et diverge pour $|g'(x)| > 1$

condition de Lipschitz

$$|g(x) - g(\bar{x})| \leq L|x - \bar{x}| \qquad \textit{pour tout } x \in I,$$

avec $0 < L < 1$, *alors l'itération* $x_{k+1} = g(x_k)$ *converge vers* $\bar{x}$ *pour tout point de départ* $x_1 \in I$.

Démonstration: Nous allons prouver que

(i) tous les itérés x_n appartiennent à I,

(ii) dans l'intervalle I, la racine $\bar{x}$ est unique,

(iii) les itérés x_n convergent vers $\bar{x}$.

Nous démontrons le point (i) par récurrence. Par hypothèse, x_1 appartient à I. Supposons alors que x_{n-1} appartient à I, on en déduit

$$\begin{aligned} |x_n - \bar{x}| &= |\, g(x_{n-1}) - g(\bar{x})| \\ &\leq L|x_{n-1} - \bar{x}| \\ &\leq r, \end{aligned}$$

c'est-à-dire que x_n appartient également à I.

En ce qui concerne l'unicité (point (ii)), remarquons que s'il y avait deux points fixes $\bar{x}$ et $\bar{y}$, on aurait

$$|\bar{x} - \bar{y}| = |g(\bar{x}) - g(\bar{y})| \leq L|\bar{x} - \bar{y}| < |\bar{x} - \bar{y}|.$$

Cette contradiction indique que l'on doit avoir $\bar{x} = \bar{y}$.

Quant au point (iii), on obtient successivement :

$$\begin{aligned} |x_n - \bar{x}| &\leq L|x_{n-1} - \bar{x}| \\ &\leq L^2|x_{n-2} - \bar{x}| \\ &\vdots \\ &\leq L^{n-1}|x_1 - \bar{x}| \end{aligned} \tag{4.2}$$

d'où, pour $n \to \infty$, $x_n \to \bar{x}$ puisque $L < 1$. ∎

Un cas particulier intéressant du Théorème 4.1 est quand la valeur absolue de la dérivée est inférieure à 1 sur tout l'intervalle.

Proposition 4.2 *Soit g une fonction dérivable qui admet un point fixe $\bar{x}$. Considérons l'intervalle $I = \{x \mid |x - \bar{x}| \leq r\}$. Si pour tout $x \in I$, on a*

$$|g'(x)| \leq C < 1,$$

alors l'itération $x_{k+1} = g(x_k)$ converge vers $\bar{x}$ pour tout point de départ $x_1 \in I$.

Démonstration: Nous allons montrer que l'hypothèse du Théorème 4.1 est satisfaite. Nous avons en effet, pour tout $x \in I$,

$$\begin{aligned} |g(x) - g(\bar{x})| &= |g'(\xi)(x - \bar{x})| \qquad (4.3) \\ &= |g'(\xi)||x - \bar{x}| \\ &\leq C|x - \bar{x}|, \end{aligned}$$

où (4.3) est obtenu en vertu du théorème des accroissements finis pour un ξ entre x et $\bar{x}$. ∎

Il est intéressant de remarquer que (4.2) implique que l'ordre de convergence de la méthode est linéaire. Il est également important de voir que plus L est petit, plus la méthode sera rapide. Nous avons également vu que la preuve de la Proposition 4.2 implique que si $|g'(x)| \leq C$ sur l'intervalle, nous avons $C \leq L$. A nouveau, une borne peu élevée sur la dérivée mènera à une convergence plus rapide.

Exemple 4.2 Si on reprend les expressions trouvées dans l'Exemple 4.1, on trouve des comportements totalement différents en fonction de $g(x)$. Reportons-les dans le tableau suivant.

$g(x)$	$g'(x)$	Borne de $\lvert g'(x)\rvert$ sur $[0,1]$	Intervalle de convergence autour de $\bar{x} \approx 0.682$
$-x^3 + 1$	$-3x^2$	3	$\emptyset$
$\sqrt[3]{-x+1}$	$-\frac{1}{3(-x+1)^{\frac{2}{3}}}$	$+\infty$	$(2\bar{x} - 1 + \left(\frac{1}{3}\right)^{\frac{3}{2}}, 1 - \left(\frac{1}{3}\right)^{\frac{3}{2}})$ $\approx (0.5565, 0.8075)$
$x^3 + 2x - 1$	$3x^2 + 2$	5	$\emptyset$
$\frac{1}{x^2+1}$	$\frac{-2x}{(x^2+1)^2}$	1	$\mathbb{R}$

∎

Remarquons que le Théorème 4.1 produit une condition suffisante de convergence. La condition n'est cependant pas toujours nécessaire. Dans l'Exemple 4.2, on peut prouver, par exemple, que la deuxième forme produit une convergence pour tout point de départ dans $(0, 1)$. Il arrive, en effet, quelquefois que dans le courant de l'agorithme, on retombe presque par hasard dans une zone où il y a convergence.

4.3 Méthode de la sécante

4.3.1 Exposé de la méthode

Jusqu'à présent, toutes les méthodes que nous avons vues jouissent d'une convergence linéaire, dans le cas où elles convergent. Nous allons à présent améliorer la méthode de la bisection et obtenir un processus dont l'ordre de convergence est superlinéaire. L'idée de la méthode de la sécante est assez similaire à celle de la bisection. A chaque itération, on conserve les deux derniers itérés x_{i-1} et x_i. Mais cette fois, nous ne nécessitons aucune hypothèse sur le signe des deux itérés. A la place, nous allons approximer la fonction f par la droite qui relie les deux points $(x_{i-1}, f(x_{i-1}))$ et $(x_i, f(x_i))$ et rechercher la racine de cette droite. Cette racine sera le nouvel itéré x_{i+1}. Mathématiquement, on calcule l'itéré par la formule

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

L'interprétation géométrique est indiquée sur la Figure 4.3. On voit qu'il s'agit d'une amélioration par rapport à la bisection. En effet, dans certains cas, la méthode de la bisection trouve par hasard de très bons itérés. Mais à cause de son extrême rigidité (l'obligation de toujours diviser l'intervalle par 2), elle doit parfois s'éloigner de ces bonnes approximations de la racine. La méthode de la sécante, quant à elle, tire parti de la valeur donnée par $f(x_i)$ et non pas uniquement de son signe.

Exemple 4.3 Appliquons la méthode de la sécante pour trouver la racine réelle de $f(x) = x^3 + x - 1 = 0$. Nous partons, comme dans le cas présenté dans l'Exemple 1.2, des itérés $x_1 = 0$ et $x_2 = 1$. Pour une meilleure comparaison des méthodes, nous reportons les résultats de la sécante et de la bisection en parallèle dans le tableau suivant. Nous reportons 6 chiffres après la virgule.

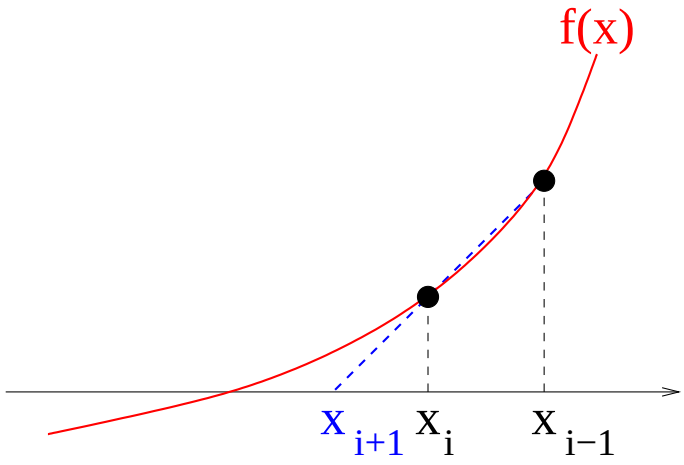

FIGURE 4.3 – Interprétation géométrique de la méthode de la sécante

Itération	Sécante		Bisection	
	x_i	$f(x_i)$	x_i	$f(x_i)$
1	0	-1.000000	0	-1.000000
2	1	1.000000	1	1.000000
3	0.500000	-0.375000	0.500000	-0.375000
4	0.636364	-0.105935	0.750000	0.171875
5	0.690052	0.018636	0.625000	-0.130859
6	0.682020	-0.000737	0.687500	0.012451
7	0.682326	-0.000005	0.656250	-0.061127
8	0.682328	0.000000	0.671875	-0.024830
9	0.682328	0.000000	0.679688	-0.006314
10	0.682328	0.000000	0.683594	0.003037

■

4.3.2 Convergence de la méthode de la sécante

Il est clair à partir de l'exemple que la méthode converge plus rapidement vers la racine. Nous allons maintenant analyser cet ordre de convergence.

Théorème 4.2 *La méthode de la sécante jouit d'une convergence d'ordre*

$\frac{1+\sqrt{5}}{2} \approx 1.618$. *En d'autres termes, on a*

$$|x_{n+1} - \bar{x}| \leq C|x_n - \bar{x}|^{1.618},$$

pour $C > 0$ *et où* $f(\bar{x}) = 0$.

La preuve de ce théorème est extrêmement technique et dépasse le cadre de ce cours. On peut néanmoins essayer de comprendre comment un ordre de convergence aussi bizarre peut apparaître. L'idée est que l'on peut écrire l'erreur comme

$$\begin{aligned} e_{n+1} := (x_{n+1} - \bar{x}) &\approx -\frac{1}{2}\left(\frac{f''(\bar{x})}{f'(\bar{x})}\right) e_n e_{n-1} \\ &\approx K e_n e_{n-1}. \end{aligned} \tag{4.4}$$

En prenant le logarithme de la dernière expression et en notant $z_i = \log|Ke_i|$, on peut écrire

$$z_{n+1} \approx z_n + z_{n-1}.$$

Ceci est une relation de récurrence qui ressemble à s'y méprendre à la suite de Fibonacci. On peut montrer que tout élément de cette récurrence peut s'écrire comme

$$z_n = A\alpha^n + B\beta^n$$

où α et β sont les racines de l'équation quadratique provenant de la récurrence

$$z^2 = z + 1.$$

On a par conséquent, $\alpha = \frac{1+\sqrt{5}}{2}$ et $\beta = \frac{1-\sqrt{5}}{2}$. On aura donc

$$\begin{aligned} \log|Ke_n| &\approx A\left(\frac{1+\sqrt{5}}{2}\right)^n + B\left(\frac{1-\sqrt{5}}{2}\right)^n \\ &\approx A\left(\frac{1+\sqrt{5}}{2}\right)^n \end{aligned}$$

puisque dans ce cas-ci, c'est le terme puissance de α qui domine. On a donc finalement

$$\begin{aligned} |Ke_n| &\approx 10^{A\alpha^n} \\ |Ke_{n-1}| &\approx 10^{A\alpha^{n-1}}, \end{aligned}$$

c'est-à-dire

$$\frac{|e_n|}{|e_{n-1}|^\alpha} \approx C.$$

Il s'agit bien entendu d'un essai d'explication et non d'une preuve rigoureuse.

4.3.3 Méthode de la regula falsi (fausse position)

La méthode de la sécante augmente assez sensiblement la vitesse de convergence si on la compare à la méthode de la bisection. Cependant, elle ne conserve pas la robustesse de celle-ci. En particulier, on ne peut rien dire sur sa convergence globale. On ne peut savoir a priori ni si elle va converger ni vers quelle racine elle va converger le cas échéant. La méthode présentée dans cette section tente de réaliser un mariage équilibré entre les deux processus décrits précédemment.

Le principe de la méthode est le suivant. Comme dans le cas de la bisection, on part de deux points x_0 et x_1 tels que $f(x_0)f(x_1) < 0$. Pour calculer le point suivant, nous utilisons la méthode de la sécante, nous avons donc

$$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)}. \tag{4.5}$$

Mais cette fois, au lieu de remplacer x_0 par x_2 et d'itérer, nous préférons conserver la robustesse de la bisection en choisissant de conserver x_2 et le point $y \in \{x_0, x_1\}$ tel que $f(y)f(x_2) < 0$. Nous continuons ensuite le processus où x_2 prend le rôle de x_1 et y celui de x_0. On voit que, si on connaît l'existence de deux points tels que $f(x_0)f(x_1) < 0$, on aura la garantie de converger vers une racine dans l'intervalle considéré, ce qui est une amélioration par rapport à la méthode de la sécante.

Lorsque l'on analyse la vitesse de convergence de la méthode de la fausse position, on remarque malheureusement que celle-ci n'est pas superlinéaire 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 par (4.5) a toujours le même signe. Dès lors, en supposant sans perte de généralité, que $f(x_2)$ a le même signe que $f(x_1)$ dans (4.5), x_0 ne sera jamais remplacé dans le courant de l'algorithme. Si on se réfère à (4.4), on voit que l'erreur suivra, asymptotiquement la formule

$$\begin{aligned} |e_{n+1}| &\approx K|e_n||e_{n-1}| \\ &= K|e_n||x_{n-1} - \bar{x}| \\ &= K|e_n||e_0| \end{aligned} \tag{4.6}$$

puisque, dans le cas de la regula falsi, comme on l'a dit, le point x_{n-1} ne change en réalité pas et reste x_0 durant tout l'algorithme. L'équation (4.6)

exprime bien la convergence linéaire de la méthode. Pour illustrer le fait que l'itéré x_0 est utilisé dans tout l'algorithme comme l'un des deux points de la méthode, on peut se référer à la Figure 4.4.

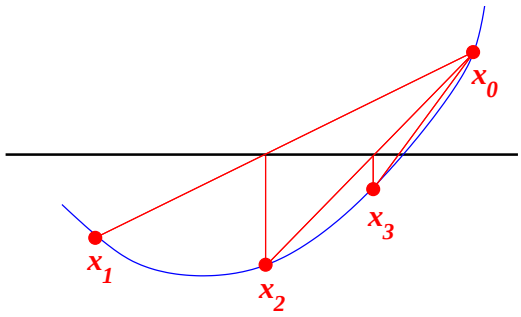

FIGURE 4.4 – Si la convexité est constante au voisinage de la racine, le point x_0 reste fixe pendant tout l'algorithme de la regula falsi

4.3.4 Extension de la méthode de la sécante : méthode de Müller

La méthode de la sécante se base sur l'approximation linéaire de la fonction à partir des deux itérés précédents. Rien n'empêche d'utiliser plus de points précédemment calculés et de produire une approximation polynomiale d'un degré plus élevé. On peut, par exemple, considérer les trois précédents itérés $(x_n, f(x_n))$, $(x_{n-1}, f(x_{n-1}))$ et $(x_{n-2}, f(x_{n-2}))$ pour créer un polynôme de degré 2 qui approxime notre fonction f. C'est ce qu'on appelle la méthode de Müller. Deux problèmes apparaissent. Premièrement, ayant un polynôme quadratique, nous aurons le choix entre 2 points (les 2 racines du polynôme) comme nouvel itéré. Cette question est résolue en général en prenant la racine la plus proche de x_n. Le deuxième problème apparaît lorsque le polynôme créé n'a que des racines complexes. Il ne sera alors pas possible de continuer sur la droite réelle. Néanmoins, ce problème peut également être un avantage. La méthode de Müller est en effet la seule méthode qui, partant d'un point réel, peut converger vers une racine complexe (si une telle racine est recherchée). A nouveau, l'ordre de convergence est assez technique à analyser. On peut prouver que son ordre de convergence est environ de 1.84, ce qui améliore la méthode de la sécante, mais pas drastiquement.

4.4 Méthode de Newton-Raphson

4.4.1 Idée de la méthode

Nous allons encore améliorer le taux de convergence en profitant de l'information de la dérivée de f si celle-ci est disponible. Le principe de la méthode est, une fois de plus, d'approximer la fonction f par une droite. Mais au lieu de considérer la droite reliant $(x_n, f(x_n))$ à $(x_{n-1}, f(x_{n-1}))$, nous allons considérer l'approximation linéaire donnée en x_n par le développement en série de Taylor tronqué au terme linéaire. Rappelons que nous avons

$$f(x) = f(x_n) + (x - x_n)f'(x_n) + \frac{(x - x_n)^2}{2}f''(x_n) + \cdots$$

Si nous approximons f uniquement par les deux premiers termes de cette série, nous obtenons $f(x) = f(x_n) + (x - x_n)f'(x_n)$. Une approximation d'une racine de cette fonction est donc

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

L'algorithme de Newton-Raphson consiste à considérer cette approximation comme itéré suivant. Nous aurons donc

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

L'interprétation géométrique de la méthode est indiquée sur la Figure 4.5. A partir du point $(x_n, f(x_n))$, on trace la tangente à f et recherchons l'intersection de cette tangente avec l'axe des abscisses. Cela donne l'itéré suivant.

4.4.2 Convergence de la méthode de Newton-Raphson

Etudions à présent le taux de convergence de la méthode de Newton-Raphson. Le théorème suivant prouve que la méthode jouit d'une convergence d'ordre quadratique.

Théorème 4.3 *Soit $\bar{x}$, une racine de f. Si $f'(\bar{x}) \neq 0$, la méthode de Newton-Raphson converge quadratiquement vers $\bar{x}$ dans un voisinage de $\bar{x}$, c'est-à-dire*

$$|x_n - \bar{x}| \leq C|x_{n-1} - \bar{x}|^2,$$

pour $C > 0$, n suffisamment grand, et x_n suffisamment proche de $\bar{x}$.

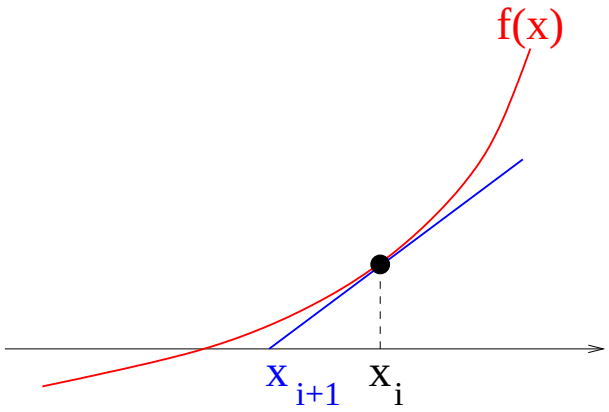

FIGURE 4.5 – Interprétation géométrique de la méthode de Newton-Raphson

Démonstration: Ecrivons l'erreur réalisée à l'itération n. Nous avons

$$|x_n - \bar{x}| = |x_{n-1} - \bar{x} - \frac{f(x_{n-1})}{f'(x_{n-1})}|. \tag{4.7}$$

Par le théorème de Taylor, nous avons

$$0 = f(\bar{x}) = f(x_{n-1}) + f'(x_{n-1})(\bar{x} - x_{n-1}) + \frac{f''(\xi_n)}{2!}(\bar{x} - x_{n-1})^2,$$

pour ξ_n compris entre $\bar{x}$ et x_{n-1}. Utilisant l'expression de $f(x_{n-1})$, on peut donc réécrire (4.7) comme

$$\begin{aligned} |x_n - \bar{x}| &= |(x_{n-1} - \bar{x}) + \frac{f'(x_{n-1})}{f'(x_{n-1})}(\bar{x} - x_{n-1}) + \frac{f''(\xi_n)}{2f'(x_{n-1})}(\bar{x} - x_{n-1})^2| \\ &= |\frac{f''(\xi_n)}{2f'(x_{n-1})}(\bar{x} - x_{n-1})^2| \\ &\leq C|x_{n-1} - \bar{x}|^2 \end{aligned}$$

car $f'(\bar{x}) \neq 0$ par hypothèse et dès lors, on peut prouver que $\frac{f''(\xi_n)}{f'(x_{n-1})}$ est borné pour ξ_n et x_{n-1} dans un voisinage de $\bar{x}$. ■

Lorsque l'on se trouve proche de la racine, la méthode de Newton-Raphson converge extrêmement vite. Malheureusement, cette convergence n'est pas globale. Pour certains points de départ, en effet, il peut arriver que la méthode

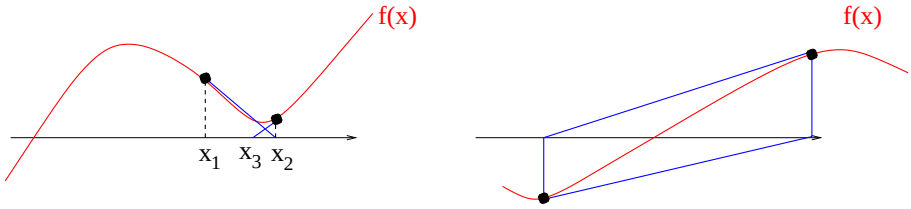

FIGURE 4.6 – Deux cas de non-convergence pour la méthode de Newton-Raphson

diverge ou cycle, elle aussi. Cela peut être aisément illustré. La Figure 4.6 indique quelques mauvais cas qui peuvent se produire. Dans le cas de plusieurs racines pour f, il n'est, en général, pas évident de savoir a priori quels points de départ convergeront vers quelles racines. C'est la raison pour laquelle la méthode de Newton sera souvent utilisée pour améliorer rapidement une approximation obtenue par une autre méthode plus robuste (comme par exemple la bisection). Enfin, il y a un cas particulier important pour lequel la méthode n'admet pas une convergence quadratique. Nous avons vu, en effet, que la convergence quadratique n'est obtenue dans le Théorème 4.3 que pour $f'(\bar{x}) \neq 0$. Dans le cas d'une racine multiple, et donc avec $f'(\bar{x}) = 0$, la méthode converge néanmoins mais seulement linéairement.

Théorème 4.4 *Soit $\bar{x}$ une racine de f. Si $f'(\bar{x}) = 0$, l'itéré de la méthode de Newton-Raphson converge linéairement vers $\bar{x}$ dans un voisinage de $\bar{x}$.*

Démonstration: On a $x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$. On peut donc écrire l'erreur

$$|x_n - \bar{x}| = \left| x_{n-1} - \bar{x} - \frac{f(x_{n-1})}{f'(x_{n-1})} \right|. \tag{4.8}$$

Remarquons que nous ne pouvons plus utiliser le même développement de Taylor d'ordre 2 qu'auparavant car cela nécessitait de borner f''/f' au voisinage de $\bar{x}$. Nous pouvons cependant utiliser un développement d'ordre 1 et écrire

$$0 = f(\bar{x}) = f(x_{n-1}) + f'(\xi_n)(\bar{x} - x_{n-1}).$$

Nous pouvons maintenant réécrire (4.8) comme

$$\begin{aligned} |x_n - \bar{x}| &= |x_{n-1} - \bar{x} - \frac{f'(\xi_n)}{f'(x_{n-1})}(x_{n-1} - \bar{x})| \\ &= |(1 - \frac{f'(\xi_n)}{f'(x_{n-1})})(x_{n-1} - \bar{x})|. \end{aligned} \tag{4.9}$$

Cette fois, nous pouvons borner l'expression $|f'(\xi_n)/f'(x_n)|$ car ξ_n est compris entre $\bar{x}$ et x_n. Dans un voisinage de $\bar{x}$, on a dès lors que $|f'(\bar{x})| \leq |f'(\xi_n)| \leq |f'(x_n)|$. On peut donc déduire de (4.9) que

$$|x_n - \bar{x}| \leq C|x_{n-1} - \bar{x}|,$$

avec $C > 0$. ■

Si on sait à l'avance que l'on cherche une racine double, il est possible de retrouver un ordre quadratique de convergence. Pour une racine de multiplicité m, les itérations

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

produisent à nouveau une convergence quadratique.

4.4.3 Lien avec d'autres méthodes

La méthode du point fixe Lorsque nous avons étudié les méthodes de point fixe pour résoudre un problème $f(x) = 0$, nous avons dit qu'il existe une multitude de façons de réécrire l'équation afin de résoudre le problème $x = g(x)$. Une manière simple de faire est de remarquer que l'on peut toujours écrire g comme $g(x) = x - af(x)$ où a est une constante. Il s'agit d'une expression très similaire à celle de la méthode de Newton-Raphson. En réalité, c'est comme si la méthode de Newton-Raphson était une méthode de point fixe où l'on a choisi $a = \frac{1}{f'(x_n)}$. En revanche la méthode de Newton-Raphson utilise une valeur a différente à chaque itération. Ce détail permet de passer d'un ordre de convergence linéaire à quadratique.

La méthode de la sécante Que se passe-t-il si nous ne disposons pas d'une expression analytique de la dérivée de f ? Nous ne pourrons alors pas

utiliser la méthode de Newton-Raphson telle quelle. Nous pouvons cependant appliquer la définition et considérer $f'(x_n) \approx \frac{f(x_{n-1})-f(x_n)}{x_{n-1}-x_n}$. En utilisant cette approximation numérique de la dérivée, on retombe en réalité sur la méthode de la sécante. Il est intéressant de remarquer que le fait d'utiliser cette approximation ramène l'ordre de convergence de 2 à 1.618.

Optimisation La méthode de Newton-Raphson est souvent utilisée pour maximiser ou minimiser une fonction f deux fois dérivable. On parle alors de *méthode de Newton*. Dans ce cas, on se base sur le fait que l'on recherche un point $\bar{x}$ où la dérivée $f'(\bar{x}) = 0$. Le principe de la méthode est exactement le même. On part d'un point x_1, et on itère

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

Dans ce cas, il conviendra bien entendu ensuite de vérifier si l'on obtient un minimum ou un maximum. Dans le cas d'optimisation de fonctions convexes ou concaves, la question ne se pose plus puisqu'elles admettent un extremum unique qui est soit un mimumum (cas d'une fonction convexe) soit un maximum (cas d'une fonction concave).

Chapitre 5

Résolution numérique d'équations différentielles ordinaires

De nombreux problèmes réels font intervenir la résolution d'équations différentielles. Il est dès lors naturel d'étudier la résolution numérique de celles-ci.

Essayons tout d'abord de comprendre ce qu'est une équation différentielle. Il s'agit d'un problème où l'on doit trouver une fonction $x(t)$ dont on connaît la *dérivée* en fonction de t et de $x(t)$ elle-même

$$x'(t) = f(x(t), t). \tag{5.1}$$

Si la fonction $f(x(t), t)$ ne dépend que de t, on a alors affaire à un problème d'intégration. Comme pour un problème d'intégration, l'ensemble des solutions d'un problème de type (5.1) est défini à une constante près. Pour déterminer une solution unique à (5.1), une condition nécessaire (et pas nécessairement toujours suffisante) est de fixer la valeur de $x(t)$ pour un t_0 donné. Un problème de ce type est appelé *problème aux valeurs initiales.*

Définition 5.1 *Un problème aux valeurs initiales est un problème déterminé par une équation différentielle et une condition initiale*

$$\begin{aligned} x'(t) &= f(x(t), t) \\ x(t_0) &= x_0. \end{aligned} \tag{5.2}$$

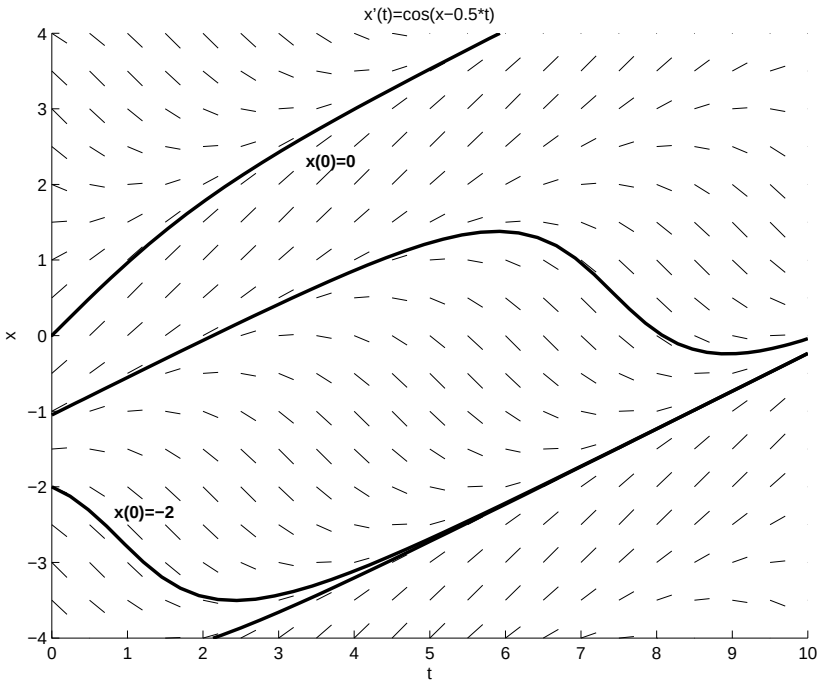

FIGURE 5.1 – La Figure représente le champ de vecteurs correspondant à l'équation différentielle $x'(t) = \cos(x(t) - t/2)$ et des solutions pour quatre conditions initiales différentes

Pour mieux comprendre le principe d'une équation différentielle, et pour donner une bonne introduction aux techniques de résolution, nous allons en faire une représentation graphique. Dessinons pour ce faire le champ de vecteurs $(t, x(t), f(x(t), t))$, c'est-à-dire, que pour chaque paire $(t, x(t))$ choisis sur une grille, on représente $f(x(t), t)$ par une flèche de pente égale à $f(x(t), t)$. Le problème de l'équation différentielle consiste alors à trouver une fonction $x(t)$ qui soit *tangente* en tous points à ces flèches. Un exemple est donné à la Figure 5.1 où l'on résout le problème aux valeurs initiales

$$\begin{aligned} x'(t) &= \cos(x(t) - \tfrac{t}{2}) \\ x(t_0) &= x_0. \end{aligned}$$

Le problème est résolu pour quatre conditions initiales différentes et donne quatre solutions clairement identifiables. Dans la suite, pour comprendre au mieux les techniques de résolution des problèmes aux valeurs initiales, il

sera assez intuitif de se rapporter à la représentation de la Figure 5.1. Nous verrons plus tard qu'il y a de multiples façons de résoudre numériquement un problème aux valeurs initiales. Mais avant de passer à l'exposé des méthodes, il est important de s'attarder sur quelques propriétés d'un problème aux valeurs initiales.

5.1 Stabilité d'une équation différentielle ordinaire

Une question importante lors de la résolution numérique d'une équation différentielle est la stabilité de l'équation. La stabilité détermine si une méthode numérique peut être appliquée, ou le cas échéant, impose une borne sur la taille du pas afin d'obtenir une solution fiable.

Afin de comprendre ce qu'est la stabilité d'une équation différentielle, nous allons tout d'abord étudier une équation différentielle instable.

Exemple 5.1 Equation différentielle ordinaire instable
Soit le problème aux valeurs initiales

$$\begin{aligned} x'(t) &= x(t) \\ x(0) &= C. \end{aligned} \tag{5.3}$$

Le champ de vecteurs et la solution pour $C = 1$ sont représentés sur la Figure 5.2. Pour le problème (5.3), la solution analytique peut être trouvée. En effet

$$x(t) = Ce^t$$

est la solution unique à (5.3). Considérons à présent une technique de résolution numérique de (5.3). Le principe de toutes les méthodes numériques est d'approximer $x(t)$ pour des temps discrétisés $t_0, t_0 + h, t_0 + 2h, t_0 + 3h, \ldots$ Inévitablement une erreur par rapport à la solution analytique sera introduite à chaque itération. Pour voir l'effet qu'a une erreur sur la suite du calcul, nous allons modéliser l'erreur ϵ faite lors du calcul, par une différence ϵ sur la condition initiale. Si l'on résout donc un problème perturbé $x'(t) = x(t), x(0) = C - \epsilon$, la solution obtenue est $x_\epsilon = (C - \epsilon)e^t$. Comparant $x(t)$ à $x_\epsilon(t)$, on obtient

$$e(t) := x(t) - x_\epsilon(t) = \epsilon e^t. \tag{5.4}$$

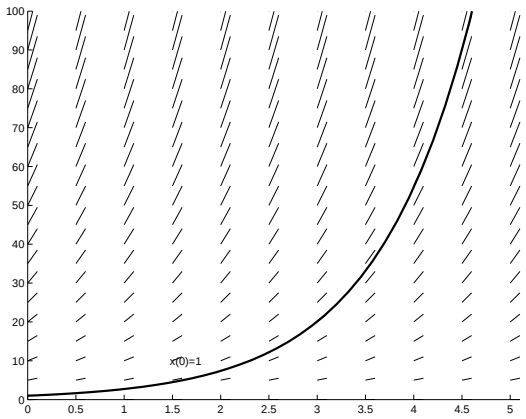

FIGURE 5.2 – La solution du problème $x'(t) = x(t), x(0) = 1$.

On voit que l'erreur introduite grandit exponentiellement lorsque t augmente. En d'autres termes, une erreur introduite au départ du calcul va être amplifiée exponentiellement en cours de calcul. Cela explique pourquoi on parle d'une équation différentielle *instable*. On comprend dès lors, pourquoi il sera plus difficile de résoudre une telle équation. A titre d'illustration, la Figure 5.3 représente la différence entre la solution pour $x(0) = 1$ et pour $x(0) = 0.9$. ■

Nous passons tout naturellement à l'exemple d'une équation différentielle stable.

Exemple 5.2 Similairement à l'exemple précédent, nous considérons à présent le problème

$$\begin{aligned} x'(t) &= -x(t) \\ x(0) &= C, \end{aligned} \tag{5.5}$$

dont la solution analytique est $x(t) = Ce^{-t}$. Une petite perturbation dans la condition initiale $x(0) = C - \epsilon$ nous donne cette fois comme solution $x_\epsilon(t) = (C - \epsilon)e^{-t}$ et comme erreur $e(t) = x(t) - x_\epsilon(t) = \epsilon e^{-t}$. Cette fois, l'erreur décroît exponentiellement. En d'autres termes, une erreur commise en début de calcul ne se répercutera pratiquement pas sur la suite du calcul. La Figure 5.4 représente les deux solutions pour les conditions initiales $x(0) = 1$ et $x(0) = 0.9$. Cette fois, il est difficile de distinguer les deux solutions au-delà de $t = 2$. ■

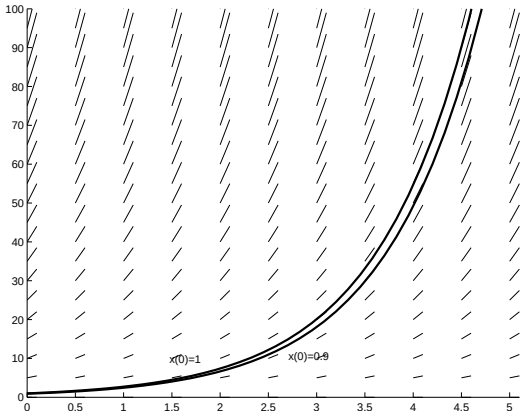

FIGURE 5.3 – La solution aux problèmes $x'(t) = x(t), x(0) = 1$ et $x'(t) = x(t), x(0) = 0.9$.

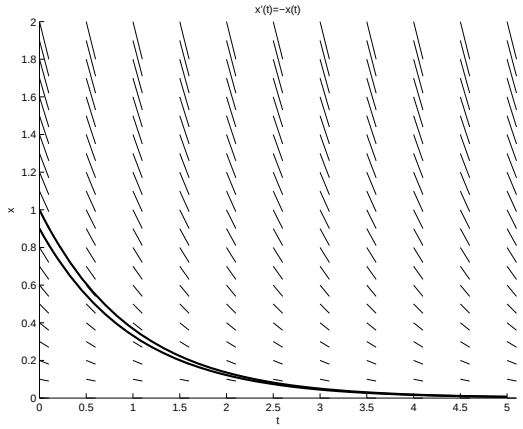

FIGURE 5.4 – La solution aux problèmes $x'(t) = -x(t), x(0) = 1$ et $x'(t) = -x(t), x(0) = 0.9$.

Dans certains cas, la stabilité d'une équation ne peut pas être déterminée pour toute la zone de caclul. L'équation sera stable dans certaines zones et instable dans d'autres. Dans ce genre de cas, il faudra choisir le pas d'intégration en s'adaptant à la circonstance.

Exemple 5.3 Considérons le problème

$$\begin{aligned} x'(t) &= -2tx(t) \\ x(0) &= C, \end{aligned} \tag{5.6}$$

dont la solution analytique est $x(t) = Ce^{-t^2}$. Cette fois, des solutions de conditions initiales proches s'écartent losrque t est négatif et se rapprochent lorsque t est positif. Le phénomène est illustré à la Figure 5.5. ■

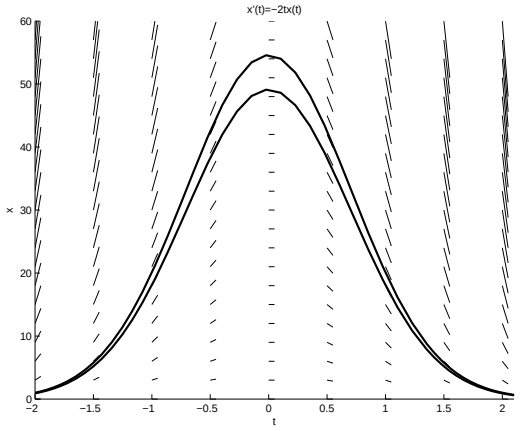

FIGURE 5.5 – La solution aux problèmes $x'(t) = -2tx(t), x(-2) = 1$ et $x(-2) = 0.9$

Les problèmes (5.3) et (5.5) sont les exemples typiques d'équations différentielles stables et instables. En fait, nous allons voir que chaque équation différentielle peut se ramener localement à ces deux cas canoniques.

Définition 5.2 *Une équation différentielle*

$$\begin{aligned} x'(t) &= f(x(t), t) \\ x(t_0) &= x_0 \end{aligned}$$

est dite stable *en* $(x(t),t)$ *si son* Jacobien $J(x(t),t) = \frac{\partial f(x(t),t)}{\partial x}(x(t),t) < 0$ *et* instable *si son* Jacobien $J(x(t),t) = \frac{\partial f(x(t),t)}{\partial x}(x(t),t) > 0$.

La définition précédente se justifie car l'erreur commise en modifiant légèrement la condition initiale s'amplifie dans le cas instable et s'amenuise dans le cas stable comme la Proposition suivante nous l'indique.

Proposition 5.1 *Soient les deux problèmes aux valeurs initiales*

$$\begin{array}{lll} x'(t) = f(x(t),t) & & x'(t) = f(x(t),t) \\ x(t_0) = x_0 & (*) & x(t_0) = x_0 - \epsilon \qquad (**) \end{array}$$

dont les solutions sont $x^*(t)$ *et* $x^{**}(t)$ *respectivement. Si on définit* $e(t) := x^{**}(t) - x^*(t)$*, on a*

$$e'(t) \approx J(x(t),t)e(t),$$

et dès lors

$$e(t) \approx \epsilon exp\left(\int_{t_0}^{t} J(x(s),s)ds\right). \tag{5.7}$$

Démonstration: On a

$$\begin{aligned} e'(t) &= x^{**'}(t) - x^{*'}(t) \\ &= f(x^{**}(t),t) - f(x^*(t),t) \\ &\approx f(x^*(t),t) + J(x^*(t),t)(x^{**}(t) - x^*(t)) - f(x^*(t),t) \\ &= J(x^*(t),t)e(t), \end{aligned}$$

où l'approximation de l'avant-dernière ligne est obtenue grâce à un développement de Taylor tronqué à l'ordre 1. Finalement, l'expression (5.7) est obtenue en résolvant l'équation différentielle. ∎

On déduit de la Proposition 5.1 que l'erreur va croître exponentiellement pour une équation différentielle instable et décroître exponentiellement pour une équation différentielle stable.

Nous reviendrons plus tard sur l'importance de ces notions dans le cadre de la résolution numérique d'équations différentielles ordinaires. Il est, par exemple, assez intuitif qu'une équation instable est très difficile à résoudre numériquement. Il faut utiliser des méthodes très particulières pour ce faire. La situation des équations stables n'en est pas toute rose pour autant. Nous verrons que dans le cas de certains systèmes très stables, la situation peut également être problématique.

5.2 Méthodes de Taylor

Nous commençons notre tour parmi les différentes méthodes numériques pour résoudre les équations différentielles ordinaires par les plus intuitives de toutes. Tout d'abord, il est utile de remarquer, et c'est le cas pour toutes les méthodes que nous exposons dans ce cours, que pour trouver la fonction $x(t)$ recherchée, nous allons en réalité approximer $x(t)$ en $t_0, t_0 + h, t_0 + 2h, t_0 + 3h, \ldots$ La notation que nous adoptons dans tout le reste de ce chapitre est présentée ci-dessous.

Notation 5.1

- *Les temps pour lesquels une approximation de $x(t)$ est calculée sont notés par $t_0, t_1, t_2, \ldots$*
- *Les approximations de $x(t)$ calculées aux temps $t_0, t_1, t_2, \ldots$ sont notées respectivement $\bar{x}_0, \bar{x}_1, \bar{x}_2, \ldots$*

La première méthode que nous considérons ici consiste à écrire le développement de Taylor de la fonction recherchée $x(t)$ autour de t afin d'approximer au mieux la valeur de x en $t+h$. La longueur du développement choisie indique le degré de la méthode considérée.

5.2.1 Méthode d'Euler explicite

La méthode d'Euler explicite considère un développement de Taylor tronqué à l'ordre 1. Cela implique que l'on doit connaître la première dérivée de x en t. Cette dérivée est toutefois connue, puisqu'elle nous est donnée par l'intermédiaire de f. Rappelons-nous en effet que $x'(t) = f(x(t), t)$. Si on procède de la sorte, on obtient

$$\begin{aligned} x(t+h) &\approx x(t) + hx'(t) \\ &= x(t) + hf(x(t), t). \end{aligned}$$

Le processus $x(t + h) = x(t) + hf(x(t), t)$ est mieux connu sous le nom de *méthode d'Euler explicite*.

Méthode 5.1 (Méthode d'Euler explicite) *Les itérés sont calculés successivement par*

$$\bar{x}_{i+1} = \bar{x}_i + hf(\bar{x}_i, t_i).$$

Exemple 5.4 Soit le problème aux valeurs initiales

$$x'(t) = -x^2(t) + t$$
$$x(0) = 2.$$

Nous allons appliquer la méthode d'Euler explicite. Les valeurs obtenues par la méthode sont notées $\bar{x}_i$. Prenons un pas de $h = 0.3$. On a $\bar{x}_0 = 2$, et $\bar{x}_1 = 2 + hf(2,0) = 2 + 0.3(-4) = 0.8$. Ensuite $\bar{x}_2 = 0 + hf(0.8, 0.3) = 0 + 0.3(-0.64 + 0.3) = 0.698$. La Figure 5.6 présente le résultat du calcul

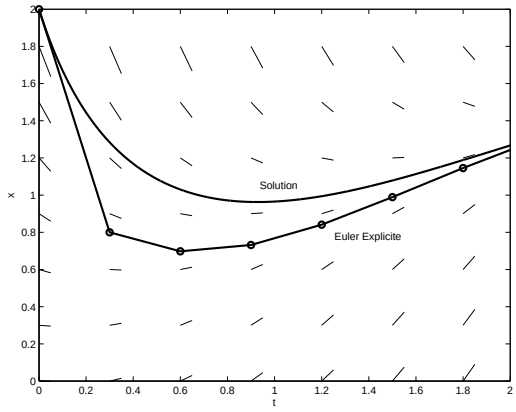

FIGURE 5.6 – L'algorithme d'Euler explicite avec un pas de $h = 0.3$ comparé à la solution réelle pour le problème $x'(t) = t - x^2(t), x(0) = 2$.

avec la méthode d'Euler explicite et la vraie solution du problème. On voit que l'approximation du 1e ordre est assez mauvaise au début du calcul et bien meilleure sur la fin. De plus, les erreurs commises en début de calcul s'amenuisent au fur et à mesure que l'algorithme progresse. Ce comportement n'est bien entendu pas général et dépend de plusieurs paramètres que nous discuterons plus tard. ■

Nous allons à présent analyser l'erreur commise lorsque l'on résout un problème aux valeurs initiales en utilisant la méthode d'Euler explicite. On se doute qu'à chaque pas dit d'intégration, une erreur va s'introduire dans le calcul. Mais il faut se rendre compte que l'erreur introduite va impliquer que l'approximation calculée à l'étape suivante le sera pour un problème

légèrement différent résultant en une nouvelle erreur. On va donc voir dans la Proposition suivante que l'erreur peut être décomposée en une erreur commise localement et une erreur globale résultant de l'accumulation des différentes erreurs locales et menant à la résolution d'un problème légèrement modifié.

Proposition 5.2 *Soit le problème aux valeurs initiales*

$$\begin{aligned} x'(t) &= f(x(t), t) \\ x(t_0) &= x_0 \end{aligned} \tag{5.8}$$

et sa solution $x^*(t)$. On définit également par $\bar{x}_0, \bar{x}_1, \bar{x}_2, \ldots$ les différentes approximations de $x^*(t)$ obtenues en utilisant la méthde d'Euler explicite avec un pas h. L'erreur EG_i commise en $t_i = t_0 + ih$ peut s'exprimer comme

$$EG_i = (1 + hJ_i)EG_{i-1} + EL_i,$$

où EG signifie erreur globale *et EL* erreur locale *et où $J_i = \frac{\partial f}{\partial x}(\zeta_i, t_i)$ et $EL_i = -\frac{h^2}{2}x''(\xi_i)$, où ζ_i est compris entre $\bar{x}_{i-1}$ et $x^*(t_{i-1})$ et où $\xi_i \in [t_{i-1}, t_i]$.*

Démonstration: Ecrivons l'erreur commise au pas i. On a

$$\begin{aligned} EG_i &= \bar{x}_i - x^*(t_i) \\ &= \bar{x}_{i-1} + hf(\bar{x}_{i-1}, t_{i-1}) - x^*(t_i) && (5.9) \\ &= \bar{x}_{i-1} + hf(\bar{x}_{i-1}, t_{i-1}) - (x^*(t_{i-1}) + hf(x^*(t_{i-1}), t_{i-1}) + \frac{h^2}{2}(x^*)''(\xi_i)) && (5.10) \\ &= EG_{i-1} + h(f(\bar{x}_{i-1}, t_{i-1}) - f(x^*(t_{i-1}), t_{i-1})) - \frac{h^2}{2}(x^*)''(\xi_i) \\ &= EG_{i-1} + h(\bar{x}_{i-1} - x^*(t_{i-1}))\frac{\partial f}{\partial x}(\zeta_i, t_{i-1}) + EL_i && (5.11) \\ &= EG_{i-1}(1 + hJ_i) + EL_i, \end{aligned}$$

où (5.9) est obtenue en exprimant comment $\bar{x}_i$ est obtenu en utilisant la méthode d'Euler explicite, (5.10) est obtenue en développant $x^*(t)$ en série de Taylor autour de t_{i-1}, et (5.11) est obtenue en appliquant le théorème des accroissements finis à la première composante de f. ∎

Il est important de comprendre ce que signifie exactement la Proposition 5.2. L'exemple suivant tente de clarifier la situation.

Exemple 5.5 Soit le problème

$$\begin{aligned} x'(t) &= -x^2(t) + t \\ x(0) &= 2 \end{aligned}$$

que nous avons déjà considéré dans l'exemple précédent. Lors de la première itération, on obtient

$$\bar{x}_1 = 2 + 0.3f(2, 0) = 0.8$$

L'erreur obtenue ici est uniquement locale, c'est-à-dire qu'elle peut être exclusivement interprétée par l'intermédiaire du développement de Taylor. Dans ce cas, l'erreur peut être approximée par 0.48. A la deuxième itération, la méthode d'Euler calcule

$$\bar{x}_{2,0.3} = 0.8 + 0.3f(0.8, 0.3) = 0.698.$$

Dans ce cas-ci, une erreur s'est introduite par rapport à la résolution du problème $x'(t) = -x^2(t) + t, x(0.3) = 0.8$. En effet, la solution en 0.6 de ce problème est 0.76, ce qui implique qu'une erreur de 0.04 a été introduite en plus à la deuxième étape. Mais le point important à remarquer est que nous avons résolu l'équation diférentielle pour la condition initiale $x(0.3) = 0.8$ au lieu de $x(0.3) = 1.28$. En d'autres termes, nous avons utilisé, dans l'approximation de Taylor, une pente de $f(0.8, 0.3)$ au lieu de $f(1.28, 0.3)$, ce qui fait une erreur approximative de 1 dans la *pente utilisée*. Cette erreur implique une accumulation d'erreurs venant des itérations précédentes. Tout ceci est illustré à la Figure 5.7. Dans la Figure 5.7, l'erreur commise à la première itération est indiquée par $EL_1 = EG_1$. A la deuxième itération, l'erreur peut être décomposée en un trait plein (EL_2) et un trait pointillé venant de l'itération précédente et multipliée par le facteur $(1 + hJ_i)$. Dans ce cas-ci, on voit que le facteur multiplicatif a rendu l'erreur plus petite par rapport à l'itération précédente. En réalité, la différence de 1 dans la valeur de f a fait *se rapprocher* l'itéré suivant de la solution réelle. ■

Nous venons de voir dans l'exemple précédent qu'une partie importante de l'erreur commise en utilisant la méthode d'Euler explicite provient des erreurs commises lors des itérations précédentes. On en vient à présent au choix du

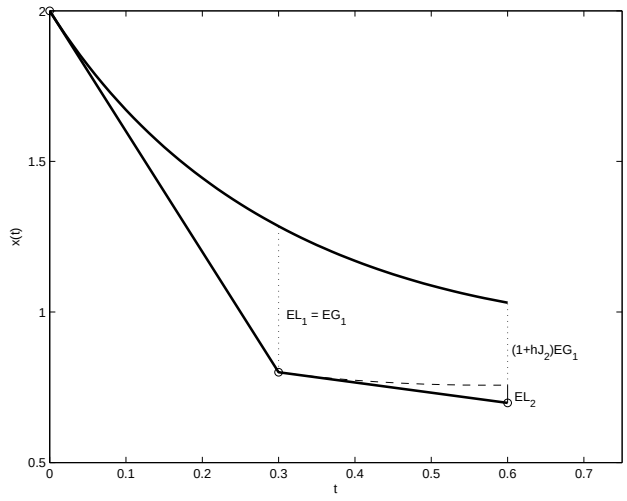

FIGURE 5.7 – Description de l'erreur locale et globale lors de la méthode d'Euler explicite

pas de la méthode. Dans ce cas-ci, on choisira donc un pas de façon à ce que l'erreur qui se propage d'une itération à l'autre *s'amenuise* (comme dans le cas de l'exemple) au lieu de *croître*. Dans le cas où les erreurs propagées d'une itération à l'autre restent sous contrôle, on dit que la *méthode est stable*. Si, au contraire, les erreurs provenant des itérations précédentes croissent, on dit que la *méthode est instable*.

Proposition 5.3 *La méthode d'Euler explicite est stable si on a*

$$-2 < hJ_i < 0 \quad \textit{pour tout } i.$$

Démonstration: L'erreur globale à l'itération i est EG_i et est donnée par $EG_i = (1 + hJ_i)EG_{i-1} + EL_i$. On aura en particulier

$$EG_i = (1 + hJ_i)(1 + hJ_{i-1}) \cdots (1 + hJ_2)EL_1 + \ldots + (1 + hJ_i)EL_{i-1} + EL_i.$$

Pour que tous les termes tendent vers 0, il faut donc $|1 + hJ_i| < 1$ ce qui est équivalent au résultat annoncé. ■

On voit donc que la méthode d'Euler explicite n'est jamais stable lorsque l'équation différentielle n'est elle-même pas stable. Par contre, et c'est plus

surprenant, il faut choisir un pas extrêmement petit lorsque l'équation différentielle est fortement stable, c'est-à-dire lorsque $J_i << 0$. Les équations très stables sont donc également des problèmes particulièrement ardus pour les méthodes numériques traditionnelles.

5.2.2 Méthodes d'ordre supérieur

Rien n'empêche de construire un développement de Taylor comportant plus de termes afin d'obtenir une approximation plus précise de l'itéré $\bar{x}_{i+1}$ en fonction de $\bar{x}_i$. Nous allons voir que ceci implique une connaissance approfondie de la fonction f et que ce n'est pas toujours très praticable. En effet, si on écrit le développement de Taylor de $x(t+h)$ autour du point t, on obtient

$$\begin{aligned} x(t+h) &= x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \frac{h^3}{3!}x'''(t) + \cdots \\ &= x(t) + hf(x(t),t) + \frac{h^2}{2}\frac{df}{dt}(x(t),t) + \frac{h^3}{3!}\frac{d^2f}{dt^2}(x(t),t) + \cdots \\ &= x(t) + hf(x(t),t) + \frac{h^2}{2}(\frac{\partial f}{\partial x}(x(t),t)f(x(t),t) + \frac{\partial f}{\partial t}(x(t),t)) + \cdots \end{aligned} \tag{5.12}$$

$$\begin{aligned} &= x(t) + hf(x(t),t) + \frac{h^2}{2}(\frac{\partial f}{\partial x}(x(t),t)f(x(t),t) + \frac{\partial f}{\partial t}(x(t),t)) + \\ &+ \frac{h^3}{3!}(\frac{\partial^2 f}{\partial x^2}f^2 + 2\frac{\partial^2 f}{\partial x \partial t}f + \frac{\partial^2 f}{\partial t^2} + \frac{\partial f}{\partial x}\frac{\partial f}{\partial t} + (\frac{\partial f}{\partial x})^2 f)(x(t),t) + \cdots \end{aligned} \tag{5.13}$$

L'expression (5.12) donne la méthode de Taylor d'ordre 2. L'expression (5.13) donne l'expression d'ordre 3.

Méthode 5.2 (Méthode de Taylor d'ordre 2) *Après calcul préalable de* $\frac{\partial f}{\partial x}$ *et de* $\frac{\partial f}{\partial t}$*, on calcule successivement les itérés*

$$\bar{x}_{i+1} = \bar{x}_i + hf(\bar{x}_i, t_i) + \frac{h^2}{2}(\frac{\partial f}{\partial x}(\bar{x}_i, t_i)f(\bar{x}_i, t_i) + \frac{\partial f}{\partial t}(\bar{x}_i, t_i))$$

Méthode 5.3 (Méthode de Taylor d'ordre 3) *Après calcul préalable des différentes dérivées partielles premières et secondes, on calcule successive-*

ment les itérés

$$\begin{aligned}\bar{x}_{i+1} =& \bar{x}_i + hf(\bar{x}_i, t_i) + \frac{h^2}{2}(\frac{\partial f}{\partial x}(\bar{x}_i, t_i)f(\bar{x}_i, t_i) + \frac{\partial f}{\partial t}(\bar{x}_i, t_i)) + \\ &+ \frac{h^3}{3!}(\frac{\partial^2 f}{\partial x^2}f^2 + 2\frac{\partial^2 f}{\partial x \partial t}f + \frac{\partial^2 f}{\partial t^2} + \frac{\partial f}{\partial x}\frac{\partial f}{\partial t} + (\frac{\partial f}{\partial x})^2 f)(\bar{x}_i, t_i).\end{aligned}$$

On le voit, la complexité de ces formules croît très rapidement. Pour pouvoir appliquer ces méthodes, il faudra donc passer au préalable par une étape de dérivation symbolique. Dans la pratique, les méthodes de Taylor d'ordre supérieur à 1 sont très peu utilisées. Ceci dit, avec la venue de logiciels de calcul symbolique, il n'est pas inintéressant de considérer ces méthodes dans certaines applications où la dérivation symbolique est possible. Finalement, il est également possible d'analyser les conditions de stabilité de telles méthodes. Au fur et à mesure que l'ordre du développement de Taylor considéré augmente, la région de stabilité augmente également. Il n'y a pas, ceci dit, de différence drastique avec la méthode d'Euler explicite.

5.2.3 La méthode d'Euler implicite

On peut adapter la méthode d'Euler dite explicite de façon à ce qu'elle adopte un comportement beaucoup plus stable. Nous verrons plus tard qu'il y a malheureusement un lourd coût à payer au niveau du temps de calcul à effectuer à chaque itération. Souvenons-nous que pour déterminer la méthode d'Euler explicite, nous avons simplement écrit un développement de Taylor autout du point t, pour en déduire une expression de $x(t+h)$. L'idée de la méthode d'Euler implicite est d'écrire le développement en $t+h$ plutôt qu'en t. On a donc $x(t) = x(t+h) - hx'(t+h) + \cdots$ et dès lors

$$x(t+h) \approx x(t) + hf(x(t+h), t+h). \tag{5.14}$$

Le problème dans (5.14) est évidemment que l'on *ne connaît pas* $f(x(t+h), t+h)$ si on ne connaît pas encore $x(t+h)$. C'est la raison pour laquelle la méthode est qualifiée d'*implicite* puisqu'il faudra résoudre une équation non linéaire à chaque pas de temps.

Méthode 5.4 (Méthode d'Euler implicite) *L'itéré $\bar{x}_{i+1}$ est obtenu comme étant une solution de l'équation*

$$\bar{x}_{i+1} = \bar{x}_i + hf(\bar{x}_{i+1}, t_{i+1}).$$

Exemple 5.6 Soit à nouveau le problème

$$x'(t) = -x^2(t) + t$$
$$x(0) = 2.$$

On considère une itération de l'algorithme d'Euler implicite. On part de $\bar{x}_0 = 2$ et on recherche $\bar{x}_1$ tel que

$$\bar{x}_0 = \bar{x}_1 - 0.3(-\bar{x}_1^2 + 0.3).$$

Dans ce cas-ci, on voit qu'il suffit de résoudre l'équation non linéaire $0.3\bar{x}_1^2 + \bar{x}_1 - 2.09 = 0$. Deux solutions sont possibles, $\bar{x}_1 = 1.254$ ou $\bar{x}_1 = -2.254$. En choisissant la solution la plus proche de $\bar{x}_0$, on obtient donc $\bar{x}_1 = 1.254$. Cette fois, l'erreur n'est plus que de 0.03. ■

La méthode semble donc être une bonne alternative. Malheureusement, la résolution d'une équation non linéaire à chaque pas rend son utilisation impraticable. On peut malgré tout analyser la stabilité de la méthode. Une analyse similaire au cas de la méthode d'Euler explicite nous mène à la Proposition suivante que nous énonçons sans démonstration.

Proposition 5.4 *La méthode d'Euler implicite est stable si*

$$\left| \frac{1}{1 - hJ_i} \right| < 1 \quad \textit{pour tout } i.$$

On voit, en particulier, que lorsque l'équation est stable ($J_i < 0$), la méthode est stable pour tout choix de pas h. La méthode d'Euler implicite admet donc des conditions de stabilité très robustes. Pour des valeurs très positives de J_i, c'est-à-dire pour un problème très instable, il semblerait que la méthode d'Euler implicite soit également stable. Ceci n'a évidemment aucune valeur car la stabilité apparente de la méthode numérique n'aura rien à voir avec la solution analytique.

5.3 Méthodes de Runge-Kutta

Les méthodes de Runge-Kutta sont aux méthodes de Taylor ce que la méthode de la sécante est à la méthode de Newton dans le cadre de la résolution d'équations non linéaires. On se souvient que la méthode de la

sécante approxime numériquement la dérivée nécessaire à la méthode de Newton. Dans le cadre d'équations différentielles ordinaires, nous avons vu que les méthodes de Taylor requièrent une lourde phase de différentiation anaytique. Les méthodes de Runge-Kutta vont remplacer cette partie par une approximation numérique des différentes dérivées partielles.

Méthode 5.5 (Runge-Kutta d'ordre 2) *Les différents itérés de la méthode de Runge-Kutta d'ordre 2 sont obtenus par le processus*

$$\bar{x}_{i+1} = \bar{x}_i + \frac{h}{2} f(\bar{x}_i, t_i) + \frac{h}{2} f(\bar{x}_i + hf(\bar{x}_i, t_i), t_{i+1}) \tag{5.15}$$

Explication: L'idée de la méthode est que l'on va copier le plus possible de termes du développement de Taylor de $x(t+h)$ en utilisant le calcul de f en deux points seulement, à savoir $F_1 = f(x(t), t)$ et $F_2 = f(x(t) + \beta h f(x(t), t), t + \alpha h)$ où α et β sont inconnus. Ces deux points sont utilisés en écrivant

$$x(t+h) \approx x(t) + w_1 h F_1 + w_2 h F_2 \tag{5.16}$$

où w_1 et w_2 sont également inconnus. La suite de cette "explication" est donc de montrer qu'on peut déterminer α, β, w_1, w_2 de manière à ce que la formule (5.16) se rapproche le plus possible du développement de Taylor de $x(t+h)$. Pour ce faire, nous allons tout d'abord faire un développement de Taylor tronqué à l'ordre 1 de F_2. On a

$$F_2 = f(x(t) + \beta h f(x(t), t), t + \alpha h) \tag{5.17}$$

$$\approx f(x(t), t) + \beta h f(x(t), t) \frac{\partial f}{\partial x}(x(t), t) + \alpha h \frac{\partial f}{\partial t}(x(t), t). \tag{5.18}$$

Si on utilise (5.18) dans (5.15), on trouve l'approximation

$$x(t+h) \approx x(t) + (w_1 + w_2) h f(x(t), t) + \alpha w_2 h^2 \frac{\partial f}{\partial t}(x(t), t) + \beta h^2 w_2 f(x(t), t) \frac{\partial f}{\partial x}(x(t), t). \tag{5.19}$$

Par ailleurs, nous avons vu lors de l'exposition des méthodes de Taylor, qu'une expression d'ordre 2 de $x(t+h)$ est (5.12) à savoir

$$x(t+h) \approx x(t) + hf(x(t), t) + \frac{h^2}{2} \left(f(x(t), t) \frac{\partial f}{\partial x}(x(t), t) + \frac{\partial f}{\partial t} \right). \tag{5.20}$$

Si on compare (5.19) à (5.20), on voit qu'il faut avoir

$$w_1 + w_2 = 1,\ \alpha w_2 = \frac{1}{2},\ \beta w_2 = \frac{1}{2}. \tag{5.21}$$

Une solution possible et pratique à (5.21) est de choisir $\alpha = \beta = 1$ et $w_1 = w_2 = \frac{1}{2}$. ■

Remarquons que la méthode proposée n'est pas la seule qui pourrait donner un ordre 2. On pourrait par exemple choisir pour satisfaire (5.21) $\alpha = \beta$ et $w_1 = 1 - \frac{1}{2\alpha}$ et $w_2 = \frac{1}{2\alpha}$.

Dans la pratique, les méthodes de Runge-Kutta d'ordre 2, bien que très simples à mettre en oeuvre sont assez peu utilisées car leur erreur n'est que de $\mathcal{O}(h^3)$. La méthode de Runge-Kutta la plus utilisée est celle d'ordre 4. Déterminer une telle formule est un travail très fastidieux que nous ne détaillerons pas ici. Nous présentons la formule de la méthode sans l'expliquer.

Méthode 5.6 (Runge-Kutta d'ordre 4) *Les différents itérés de la méthode de Runge-Kutta d'ordre 4 sont obtenus par le processus*

$$\bar{x}_{i+1} = \bar{x}_i + \frac{1}{6}(K_1 + K_2 + K_3 + K_4)$$

où

$$\begin{aligned} K_1 &= hf(\bar{x}_i, t_i) \\ K_2 &= hf(\bar{x}_i + \frac{1}{2}K_1, t_i + \frac{1}{2}h) \\ K_3 &= hf(\bar{x}_i + \frac{1}{2}K_2, t_i + \frac{1}{2}h) \\ K_4 &= hf(\bar{x}_i + K_3, t_i + h). \end{aligned}$$

Comme son nom l'indique, la méthode de Runge-Kutta d'ordre 4 copie le développement de Taylor jusqu'aux termes d'ordre 4. Le terme d'erreur est donc en $\mathcal{O}(h^5)$.

5.4 Méthodes adaptatives

Comme on l'a vu précédemment, il est souvent difficile de déterminer le pas à choisir pour assurer une stabilité de la méthode numérique tout

en conservant une quantité limitée de calculs. En général, on aimerait que l'utilisateur puisse déterminer une tolérance endéans laquelle la solution doit se trouver. Mais même en ayant accès à l'erreur locale commise par une méthode, il est souvent difficile de déterminer le pas à utiliser. Il se pourrait qu'il soit nécessaire de choisir un pas très petit sur certaines portions du problème alors que l'on pourrait se contenter de pas plus grands sur d'autres portions. Pour cette raison, plusieurs méthodes de choix automatiques du pas ont été imaginées.

Pour comprendre le principe des méthodes adaptatives parfois aussi appelées de Runge-Kutta-Fehlberg, imaginons tout d'abord le principe suivant. On considère la méthode de Runge-Kutta d'ordre 4 avec un pas h. On peut aussi considérer la même méthode avec un double pas de $h/2$. Si le pas h est satisfaisant, la différence entre l'approximation obtenue avec un pas h ou deux pas de $h/2$ sera très faible. Dans ce cas, le pas h est suffisant. Dans le cas contraire, il faudra réduire le pas. Le problème de cette méthode est qu'elle nécessite quatre appels à la fonction f pour le pas h et 7 autres appels pour le double pas de $h/2$. Cela fait un total de 11 appels à la fonction f par itération, ce qui peut s'avérer coûteux en temps de calcul dans certaines applications. Or, nous avons vu dans la section précédente qu'il y a une certaine flexibilité dans le choix des coefficients des méthodes de Runge-Kutta. L'idée est de chosir une méthode de Runge-Kutta d'ordre 5 et une méthode d'ordre 4 qui *partagent le plus possible d'évaluations communes de* f de façon à minimiser la quantité de travail à chaque itération. L'avantage est de disposer de *deux évaluations de* $x(t+h)$. En comparant les deux évaluations, nous pouvons ainsi décider si le pas h est adapté ou pas. La méthode suivante est un exemple courant de paire de méthodes de Runge-Kutta donnant lieu à une méthode adaptative.

Méthode 5.7 (Runge-Kutta-Fehlberg d'ordres 4 et 5)

$$
\begin{aligned}
K_1 &= hf(\bar{x}_i, t_i) \\
K_2 &= hf(\bar{x}_i + \frac{1}{4}K_1, t_i + \frac{1}{4}h) \\
K_3 &= hf(\bar{x}_i + \frac{3}{32}K_1 + \frac{9}{32}K_2, t_i + \frac{3}{8}h) \\
K_4 &= hf(\bar{x}_i + \frac{1932}{2197}K_1 - \frac{7200}{2197}K_2 + \frac{7296}{2197}K_3, t_i + \frac{12}{13}h) \\
K_5 &= hf(\bar{x}_i + \frac{439}{216}K_1 - 8K_2 + \frac{3680}{513}K_3 - \frac{845}{4104}K_4, t_i + h) \\
K_6 &= hf(\bar{x}_i - \frac{8}{27}K_1 + 2K_2 - \frac{3544}{2565}K_3 + \frac{1859}{4104}K_4 - \frac{11}{40}K_5, t_i + \frac{1}{2}h)
\end{aligned}
$$

On obtient deux approximations de $x(t+h)$, à savoir

$$
\begin{aligned}
\bar{x}_{i+1}^{[4]} &= x(t) + \frac{25}{216}K_1 + \frac{1408}{2565}K_3 + \frac{2197}{4104}K_4 - \frac{1}{5}K_5 \\
\bar{x}_{i+1}^{[5]} &= x(t) + \frac{16}{135}K_1 + \frac{6656}{12825}K_3 + \frac{28561}{56430}K_4 - \frac{9}{50}K_5 + \frac{2}{55}K_6
\end{aligned}
$$

qui sont respectivement une approximation d'ordre 4 et d'ordre 5 obtenues à l'aide de 6 évaluations de fonction. La différence $|\bar{x}_{i+1}^{[5]} - \bar{x}_{i+1}^{[4]}|$ est une estimation de l'erreur en t_{i+1}.

La fonction `ode45` de matlab est une méthode de cette famille mais utilisant une autre paire de méthodes de Runge-Kutta d'ordre 4 et 5 : la paire de *Dormand-Prince*. La fonction `ode23` utilise la paire de *Bogacki-Shampine* qui est une paire de méthodes de Runge-Kutta d'ordre 2 et 3 respectivement, mais partageant un certain nombre de points où la fonction est évaluée.

5.5 Méthodes à pas liés

Jusquà présent, nous avons uniquement analysé des méthodes à pas séparés. Une méthode est à *pas séparés* lorsque l'on se sert uniquement de l'intervalle $[t, t+h]$ et de l'expression de f dans celui-ci pour calculer la nouvelle valeur $x(t+h)$. L'idée d'une méthode à *pas liés* est que l'on peut se servir de la connaissance des points précédemment calculés afin d'avoir une meilleure perception de la manière dont f se comporte et ceci sans devoir procéder à une différentiation analytique qui s'avérerait trop lourde. La forme générique d'une méthode à pas liés peut être formulée comme suit.

Méthode 5.8 (Méthode à pas liés) *Si on dénote par $\bar{x}_i$ les différentes approximations obtenues par la méthode aux points t_i, on calcule successivement*

$$\bar{x}_{i+1} = \bar{x}_i + h \sum_{j=-1}^{n} \beta_j f(\bar{x}_{i-j}, t_{i-j}). \tag{5.22}$$

Dans (5.22), on remarque que j peut prendre la valeur -1 ce qui correspond à considérer que pour obtenir la valeur $\bar{x}_{i+1}$, on se sert de la valeur $\bar{x}_{i+1}$. On reconnaît là le principe d'une méthode implicite. Si $\beta_{-1} = 0$, on ne se sert que de points connus pour calculer la nouvelle valeur $\bar{x}_{i+1}$, il s'agit alors d'une méthode explicite.

Pour calculer les coefficients d'une formule de type (5.22), on doit évaluer l'intégrale

$$\bar{x}_{i+1} = \bar{x}_i + \int_{t_i}^{t_{i+1}} f(x(s), s) ds.$$

En particulier, pour obtenir les coefficients de (5.22) dans le cas d'une méthode explicite, on peut écrire le polynôme qui interpole les $n+1$ points obtenus lors des itérations précédentes $(\bar{x}_{i-n}, f(\bar{x}_{i-n}, t_{i-n})), \ldots, (\bar{x}_i, f(\bar{x}_i, t_i))$ et l'intégrer sur l'intervalle $[t_i, t_{i+1}]$. Nous donnons les méthodes explicites et implicites d'ordre 2 et 3 à titre informatif. Remarquons que les méthodes explicites à pas liés sont appelées *méthodes d'Adams-Bashforth* et les méthodes implicites *méthodes d'Adams-Moulton.*

Méthode 5.9 (Adams-Bashforth d'ordre 2)

$$\bar{x}_{i+1} = \bar{x}_i + \frac{h}{2}(-f(\bar{x}_{i-1}, t_{i-1}) + 3f(\bar{x}_i, t_i))$$

Méthode 5.10 (Adams-Bashforth d'ordre 3)

$$\bar{x}_{i+1} = \bar{x}_i + \frac{h}{12}(5f(\bar{x}_{i-2}, t_{i-2}) - 16f(\bar{x}_{i-1}, t_{i-1}) + 23f(\bar{x}_i, t_i))$$

Méthode 5.11 (Adams-Moulton d'ordre 2)

$$\bar{x}_{i+1} = \bar{x}_i + \frac{h}{2}(f(\bar{x}_i, t_i) + f(\bar{x}_{i+1}, t_{i+1}))$$

Méthode 5.12 (Adams-Moulton d'ordre 3)

$$\bar{x}_{i+1} = \bar{x}_i + \frac{h}{2}(-f(\bar{x}_{i-1}, t_{i-1}) + 8f(\bar{x}_i, t_i) + 5f(\bar{x}_{i+1}, t_{i+1}))$$

L'intérêt des méthodes à pas liés est qu'elles n'utilisent qu'une seule évaluation de la fonction f à chaque pas d'intégration. Cela peut s'avérer un gain de temps conséquent par rapport à une méthode de Runge-Kutta d'ordre élevé qui requiert un grand nombre d'évaluations à chaque pas, et ce, surtout lorsque l'évaluation de la fonction est assez coûteuse. Dans le cadre des méthodes de Runge-Kutta, nous avons vu l'amélioration adaptative proposée par Fehlberg. Les méthodes à pas liés se prêtent également très bien à une version *adaptative* ou *prédicteur-correcteur*.

La méthode prédicteur-correcteur consiste à utiliser une méthode explicite et implicite conjointement. On va ainsi se servir de l'approximation donnée par la méthode explicite comme $\bar{x}_{i+1}$ dans la formule implicite. On évitera ainsi la coûteuse phase de résolution d'une équation non linéaire. La version adaptative consiste à utiliser la différence entre la sortie de la formule explicite et de la formule implicite pour savoir s'il faut considérer un changement de la taille du pas.

Exemple 5.7 Soit le problème

$$\begin{aligned} x'(t) &= -x^2(t) + t \\ x(0) &= 2. \end{aligned}$$

On peut remarquer que les méthodes d'Euler implicite et explicite sont en réalité les méthodes à pas liés d'ordre 1. Nous allons ici uniquement montrer comment on peut, pour l'ordre 1, mettre en pratique la méthode prédicteur-correcteur. Rappelons les formules d'Euler explicite $\bar{x}_{i+1} = \bar{x}_i + hf(x_i, t_i)$ et implicite $\bar{x}_{i+1} = \bar{x}_i + hf(\bar{x}_{i+1}, t_{i+1})$. Dans notre cas, et pour un pas de 0.3, on obtient le prédicteur donné par Euler explicite $\bar{x}_{i+1} := 2 - 1.2 = 0.8$. Le correcteur est ensuite donné en utilisant la première approximation comme $\bar{x}_{i+1} = 2 + 0.3f(0.8, 0.3) = 2 - 0.102 = 1.898$. Remarquons qu'ici, vu la différence entre les deux approximations obtenues, il serait judicieux de réduire le pas. ■

Remarquons qu'il est possible d'itérer plusieurs fois le processus *prédicteur-correcteur* afin d'obtenir une approximation plus précise. La pratique montre cependant qu'une seule itération suffit à donner de très bonnes approximations de la valeur réelle.

5.6 Systèmes d'équations différentielles ordinaires

Un système d'équations dfférentielles est très similaire à une équation différentielle scalaire. Dans ce cas, on cherche une fonction $\underline{x}(t) : \mathbb{R} \mapsto \mathbb{R}^n$ telle que

$$\begin{aligned} \frac{d\underline{x}}{dt}(t) &= f(\underline{x}(t), t) \\ \underline{x}(t_0) &= \underline{x}_0, \end{aligned} \tag{5.23}$$

où $f : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}^n$ et $\underline{x}_0 \in \mathbb{R}^n$.

5.6.1 Systèmes d'ordre supérieur à un

Les systèmes d'équations différentielles sont évidemment les plus courants dans la pratique. Ils sont également importants dans le cas scalaire car ils permettent de résoudre également les équations d'ordre supérieur à un. Supposons en effet que l'on veuille résoudre l'équation

$$x^{[n]}(t) = g(x(t), x'(t), \ldots, x^{[n-1]}(t), t), \tag{5.24}$$

avec des conditions initiales appropriées et où $g : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}$. On peut résoudre ce problème en passant à un système d'équations du premier ordre. En effet, on peut écrire $y_0(t) := x(t), y_1(t) := x'(t), y_2(t) := x''(t), \ldots$ De cette façon, on peut maintenant réécrire (5.24) de manière équivalente comme

$$\frac{d}{dt}\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ g(y_0, y_1, \ldots, y_{n-1}, t) \end{pmatrix}. \tag{5.25}$$

Il s'agit cette fois d'un système vectoriel du premier ordre où on peut voir la fonction $f : \mathbb{R}^{n+1} \times \mathbb{R} \mapsto \mathbb{R}^{n+1}$ de (5.23) comme étant $f(y_0, \ldots, y_n, t) = (y_1, y_2, \ldots, y_n, g(y_0, y_1, \ldots, y_{n-1}, t))^T$. On peut également appliquer cette astuce dans le cas de systèmes d'équations d'ordre supérieur à un. On voit donc que le système de premier ordre est le modèle complètement général qui englobe les autres.

5.6.2 Résolution de systèmes d'équations différentielles

La résolution numérique de systèmes vectoriels d'équations différentielles ne comporte pas de différence majeure par rapport au cas scalaire, à condition de considérer toutes les méthodes vues précédemment vectoriellement. A titre d'exemple, nous passons en revue quatre méthodes vues en début de chapitre pour le cas scalaire. Il est aisé de les adapter au cas vectoriel.

Méthode 5.13 (Euler explicite)

$$\underline{\bar{x}}_{i+1} = \underline{\bar{x}}_i + hf(\underline{\bar{x}}_i, t_i)$$

Méthode 5.14 (Runge-Kutta vectoriel d'ordre 4)

$$\underline{\bar{x}}_{i+1} = \underline{\bar{x}}_i + \frac{1}{6}(\underline{K}_1 + \underline{K}_2 + \underline{K}_3 + \underline{K}_4)$$

où

$$\begin{aligned}
\underline{K}_1 &= hf(\underline{\bar{x}}_i, t) \\
\underline{K}_2 &= hf(\underline{\bar{x}}_i + \frac{1}{2}\underline{K}_1, t + \frac{1}{2}h) \\
\underline{K}_3 &= hf(\underline{\bar{x}}_i + \frac{1}{2}\underline{K}_2, t + \frac{1}{2}h) \\
\underline{K}_4 &= hf(\underline{\bar{x}}_i + \underline{K}_3, t + h).
\end{aligned}$$

Méthode 5.15 (Adams-Bashforth-Moulton vectoriel d'ordre 2) *Les méthodes vectorielles d'Adams-Bashforth et d'Adams-Moulton sont respectivement*

$$\begin{aligned}
\underline{\bar{x}}_{i+1} &= \underline{\bar{x}}_i + \frac{h}{2}(-f(\underline{\bar{x}}_{i-1}, t_{i-1}) + 3f(\underline{\bar{x}}_i, t_i)) \\
\underline{\bar{x}}_{i+1} &= \underline{\bar{x}}_i + \frac{h}{2}(f(\underline{\bar{x}}_i, t_i) + f(\underline{\bar{x}}_{i+1}, t_{i+1})).
\end{aligned}$$

5.6.3 Stabilité et équations différentielles raides

Lors de l'étude de la stabilité de la méthode d'Euler, nous avons vu que le choix du pas doit être choisi de manière judicieuse pour éviter les problèmes d'instabilité. C'est évidemment le cas pour la plupart des méthodes que nous avons considérées dans ce chapitre. Nous avons également vu que la

stabilité est essentiellement régie par la valeur de $J_i = \frac{\partial f}{\partial x}$. Il est possible d'étendre cette analyse au cas d'un système $\underline{x}' = f(\underline{x}, t)$. Dans ce cas-ci, ce qui détermine la stabilité du système est le rayon spectral de sa matrice Jacobienne $J = (\frac{\partial f}{\partial x_i})$. Si toutes les valeurs propres sont telles que leur partie réelle $\mathcal{R}e(\lambda_j(J)) < 0$, on dit que le système est stable. Similairement, les valeurs propres de J en tout point vont permettre de déterminer la taille requise du pas de façon à avoir une méthode numérique stable. Un cas pathologique notoire arrive lorsque les différentes valeurs propres de la matrice Jacobienne sont de modules très différents. On parle dans ce cas d'un système *raide*. Ce genre de systèmes s'avère particulièrement délicat à résoudre. L'étude de la résolution de problèmes raides est un sujet en soi. On a déjà vu néanmoins que certaines méthodes, telle la méthode d'Euler implicite, sont assez adaptées du fait de leur très grande région de stabilité. Pour clôturer cette discussion sur les problèmes raides, nous allons l'illustrer par un exemple.

Exemple 5.8 Soit le système

$$\begin{aligned} x' &= -20x - 19y \qquad x(0) = 2 \\ y' &= -19x - 20y \qquad y(0) = 0. \end{aligned}$$

On peut résoudre analytiquement ce problème et voir que la solution peut s'exprimer comme $x(t) = e^{-39t} + e^{-t}$ et $y(t) = e^{-39t} - e^{-t}$. Quand t augmente, la partie correspondant à e^{-39t} devient rapidement négligeable et la solution se rapproche de $x(t) = -y(t) = e^{-t}$. Si on regarde la matrice Jacobienne du problème qui est obtenue aisément comme $\begin{pmatrix} -20 & -19 \\ -19 & -20 \end{pmatrix}$, on obtient naturellement les valeurs propres -39 et -1, correspondant aux deux fonctions reconnues dans la forme analytique. Prenons à présent le cas de la méthode d'Euler explicite. Cette méthode requiert que $|hJ_i| < 2$ pour garantir sa stabilité. Dans ce cas, on obtient donc respectivement que $h < 2/39$ ou $h < 2$. On en déduit que c'est la partie la plus négligeable du problème (correspondant à e^{-39t}) qui impose que le pas choisi soit très petit. ■

L'exemple illustre que dans des cas pathologiques, on peut être forcé de devoir choisir un pas d'intégration ridiculement petit à cause d'une valeur propre de la Jacobienne trop négative. Ceci peut ne pas être en adéquation avec la "physique" naturelle du phénomène.