INFOF-205 : Calcul formel et numérique

R. Petit

Année académique 2015 - 2016

Contents

1 Introduction

Définition 1.1. Le *calcul numérique* est une discipline qui traite de la définition, l'analyse, et l'implémentation d'algorithmes pour la résolution numérique des problèmes mathématiques qui proviennent de la modélisation des phénomènes réels.[1]

Le calcul numérique représente donc le triplet (modèle analytique, solution théorique, résolution algorithmique).
Remarque. Le calcul numérique étant une application informatique pour la résolution de problèmes mathématiques continus, plusieurs types de problèmes peuvent arriver (section suivante). Afin de limiter les problèmes de précision sur des fonctions continues compliquées à évaluer précisément informatiquement, plusieurs outils d'approximation peuvent être utilisés. Le théorème de Taylor permet d'approcher certaines fonctions de manière plus facilement calculatoires de manière informatique.

Théorème 1.2 (Théorème de Taylor)**.** *Soient* $f : I \to \mathbb{R}$ *où* I *est un intervalle,* $a \in I$ *et* $k \geqslant 1$*. Alors* $T(f, a, k)(x)$ *est le seul polynôme de degré inférieur ou égal à* k *tel que :*

$$\lim_{x \to a} \frac{f(x) - T(f, a, k)}{(x - a)^k} = 0,$$

où $T(f, a, k)(x)$*, appelé* polynôme de Taylor *est défini par :*

$$T(f, a, k)(x) = \sum_{i=0}^{k} \frac{f^{(i)}(a)(x - a)^i}{i!}.$$

Le calcul numérique est défini comme étant une application informatique à des problèmes relatifs à des phénomènes réels. Bien souvent, ces phénomènes sont modélisés de manière *continue*. Les ordinateurs quant à eux sont *discrets*. Cette différence majeure force des approximations par le matériel informatique. Les transformations discrètes (*discrétisations*) des phénomènes réels doivent alors être rigoureusement analysés afin de déterminer leur robustesse et leur fiabilité.

Les problèmes peuvent être classifiés de plusieurs manières selon leurs propriétés et ce qui en est attendu.

Définition 1.3. Un problème est dit :

- *qualitatif* lorsque le comportement des solutions est étudié afin d'en déterminer sa stabilité, son comportement asymptotique ;
- *quantitatif* lorsque la solution (numérique) précise est étudiée afin de tirer des conclusions spécifiques.

Un problème qualitatif cherche des conclusions globales à un problème (étude d'une famille de problèmes) et un problème quantitatif est une application précise appliquée à un problème donné.

Définition 1.4. Un problème est dit sous forme :

- *explicite* quand la solution x d'un problème est une fonction donnée des données d ;
- *implicite* quand la solution ne peut être extraite explicitement des données.

[1] Nick Trefehen, Oxford.

Exemple 1. *Trouver la racine carrée d'un nombre* $d \in \mathbb{R}$ est un problème explicite que l'on peut écrire $x = f(d) = \sqrt{d}$.

Déterminer l'ensemble des solutions x *à l'équation* $ax^2 + bx + c$ est un problème implicite car la solution n'est pas séparée explicitement des données. On peut tout de même exprimer ce problème implicite comme :

$$X = \left\{ \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \right\},$$

qui est un problème explicite.
Remarque. La forme générale d'un problème explicite est $x = F(d)$, alors que la forme générale d'un problème implicite est $F(x, d) = 0$.

Définition 1.5. Le problème explicite $x = F(d)$ est dit *bien posé* si la solution x :

1. existe ;
2. est unique ;
3. dépend **continument** de d.

Si un problème n'est pas bien posé, on dit qu'il est *mal posé*.

Définition 1.6. Un problème est dit *problème inverse* s'il correspond à la détermination des données d'entrée x d'une application G en connaissant les données de sortie $G(x) =: d$.

Remarque. Les problèmes inverses sont de manière générale mal posés, entre autre parce qu'il est fréquent que l'application G ne soit pas injective et donc que plusieurs données x peuvent donner la même donnée $d = G(x)$.
Remarque. Ce cours ne traitera que de problèmes bien posés.

Définition 1.7.

- Le *conditionnement* d'un problème $x = F(d)$ bien posé est la mesure de la *sensibilité* de x à des petits changements sur d.
- On définit par δd une faible perturbation (également dite *perturbation admissible*) sur la donnée d. On définit alors la perturbation induite sur x par $\delta x = F(d + \delta d) - F(d)$.
- Le *conditionnement relatif* du problème est la quantité :

$$\kappa(d) \coloneqq \lim_{|D| \to 0} \sup_{\delta d \in D} \frac{\|\delta x\| \|d\|}{\|x\| \|\delta d\|} = \lim_{|D| \to 0} \sup_{\delta \in D} \frac{\left\|F(d + \delta d) - F(d)\right\| \|d\|}{\left\|F(d)\right\| \|\delta d\|},$$

  où D est un voisinage de l'origine, et sup désigne la borne supérieure.

Remarque.

- si $x = 0$, alors le conditionnement relatif est forcément infini ;
- si $d = 0$, alors le conditionnement relatif est forcément nul ;
- quand $0 \in \{x, d\}$, il faut définir une autre notion, le conditionnement absolu :

$$\kappa_{\text{abs}}(d) \coloneqq \lim_{|D| \to 0} \sup_{\delta d \in D} \frac{\|\delta x\|}{\|\delta d\|}.$$

Proposition 1.8. *Si* $x = F(d)$ *est un problème explicite, et* F *est une fonction différentiable en* d*, alors, par définition :*

$$\lim_{\delta \to 0} \frac{\delta x}{\delta d}$$

ne dépend pas de la trajectoire de δd *vers* 0*. On note alors :*

$$F'(d) = \lim_{\delta d \to 0} \frac{\delta x}{\delta d},$$

et on réécrit le conditionnement par :

$$\kappa(d) = \lim_{\delta d \to 0} \frac{\|\delta x\|\|d\|}{\|\delta d\|\|x\|} = \frac{\|d\|}{\|x\|} \lim_{\delta d \to 0} \frac{\|\delta x\|}{\|\delta d\|} = \left\|F'(d)\right\| \frac{\|d\|}{\|x\|}.$$

Remarque. Si $d \in \mathbb{R}^m$, et $x \in \mathbb{R}^n$, alors $F'(d)$ est la matrice jacobienne $(\mathrm{Jac}\, F)(d)$. Et si $q = 1$, la matrice jacobienne n'a qu'une seule ligne et on parle du gradient $(\nabla F)(d)$
Exemple 2 (Conditionnement d'une différence). Soit le problème $x = F(d) = d - a$. La fonction F est différentiable, on peut donc déterminer le conditionnement :

$$\kappa(d) = \left\|F'(d)\right\| \frac{\|d\|}{\|x\|} = 1 \frac{\|d\|}{\|d - a\|}.$$

On voit alors que le conditionnement est petit pour $\|d - a\| >> 0$. Cependant, pour d proche de a, le conditionnement devient très grand. Le problème est donc *mal conditionné* pour d proche de a.
Exemple 3 (Conditionnement d'une exponentiation). Soit le problème $x = F(d) = d^r$ avec $r \in \mathbb{R}$. À nouveau, F est différentiable, et donc on peut exprimer le conditionnement comme :

$$\kappa(d) = \left\|F'(d)\right\| \frac{\|d\|}{\|x\|} = \|r\| \left\|d^{r-1}\right\| \frac{\|d\|}{\|d^r\|} = \|r\|.$$

Le conditionnement dépend donc de l'exposant r. Si r est grand, le problème est mal conditionné, peu importe la valeur de d.

Le conditionnement d'un problème est une caractéristique qui lui est intrinsèque. Si un problème est mal conditionné, peu importe l'algorithme utilisé pour le résoudre, la solution restera fortement influencé par des perturbations sur les données d. De plus, plus un problème a un grand conditionnement, plus il sera difficile à résoudre numériquement.
Remarque. les exemples ci-dessus montrent qu'un problème peut avoir un conditionnement faible pour certaines valeurs de d et un conditionnement élevé pour d'autres valeurs.

Soit $F(x, d) = g(x) - d = 0$, un problème implicite. Prenons $g : \mathbb{R} \to \mathbb{R}$ une fonction réelle différentiable. On peut alors exprimer $x = g^{-1}(d)$. Donc le conditionnement vaut :

$$\kappa(d) = \left| \frac{(g^{-1})'(d)d}{g^{-1}(d)} \right| = \left| \frac{1}{g'(g^{-1}(d))} \right| \left| \frac{d}{g^{-1}(d)} \right| = \left| \frac{1}{g'(x)} \right| \left| \frac{d}{x} \right|.$$

On remarque donc que pour $g'(x)$ petit le conditionnement devient grand. C'est intuitivement lié au fait qu'une fonction à faible dérivée (pente) en d va croitre lentement et donc pour une faible perturbation sur d (axe vertical), une grande perturbation sur x (axe horizontal) va être induite.

Définition 1.9. Soit $F(x, d) = 0$ un problème bien posé. On définit l'*algorithme numérique* pour la résolution du problème F par la suite de problèmes approchés $F_1(x^{(1)}, d^{(1)}), F_2(x^{(2)}, d^{(2)}), \ldots, F_n(x^{(n)}, d^{(n)})$ dépendant d'un paramètre n.

L'idée derrière la division en sous-problèmes est d'avoir des F_k plus simples à résoudre que F, ce qui permet une résolution numérique.

Définition 1.10. Un algorithme numérique est dit :

- *direct* si le nombre n de sous-problèmes est fixé (du moins majoré), habituellement par une fonction de la taille du problème ;
- *itératif* si n n'est pas borné et s'il existe une routine $f(u)$ admissible, indépendante de n telle que $x^{(n)} = f(x^{(n-1)})$.

Définition 1.11. Un algorithme numérique $\{F_i(x^{(i)}, d^{(i)})\}_{1 \leqslant i \leqslant n}$ est consistant si :

$$\lim_{n \to +\infty} F_n(x^{(n)}, d^{(n)}) = F(x, d^{(n)}),$$

et donc si x, la solution précise, est une solution ses sous-problèmes pour $n \to +\infty$.

Remarque. Un algorithme itératif $x^{(n)} = f(x^{(n-1)})$ est consistant si $x^{(n)} = x$ implique que $x^{(n+1)} = x$. Ce qui revient à dire qu'une fois la solution exacte atteinte, l'algorithme itératif la conserve et ne diverge pas.

Définition 1.12. Un algorithme $F_n(x^{(n)}, d^{(n)})$ est dit *fortement* consistant si :

$$\forall n \geqslant 1 : F_n(x, d^{(d)}) = 0,$$

c'est-à-dire si x est une solution admissible de tous les sous-problèmes $F_n(x^{(n)}, d^{(n)})$.

Intéressons-nous au problème $F(x, d)$ et à $F_n(x^{(n)}, d^{(n)})$, un algorithme numérique de résolution du problème. Supposons que cet algorithme numérique soit composé d'étapes telles que :

$$x^{(n)} = F_n^{(m)}\left(d_{m-1}^{(n)}\right),$$

où :

$$d_k^{(n)} = F_n^{(k)}\left(d_{k-1}^{(n)}\right) \qquad \text{et} \qquad d_0^{(n)} = d^{(n)}.$$

On considère que la résolution numérique est l'application de cet algorithme sur des valeurs perturbées :

$$\begin{cases} \bar{x}^{(n)} & = F_m^{(n)}\left(\bar{d}_{m-1}^{(n)}\right) + \delta d_m^{(n)} \\ \bar{d}_k^{(n)} & = F_k^{(n)}\left(\bar{d}_{k-1}^{(n)}\right) + \delta d_k^{(n)} \\ \bar{d}_0^{(n)} & = d^{(n)} \end{cases}$$

Remarque. On voit bien que l'algorithme à l'étape n commence avec $\bar{d}_0^{(n)} = d^{(n)}$ qui est une donnée initiale *non perturbée*.

Définition 1.13. Soit $F_n(x^{(n)}, d^{(n)})$ un algorithme numérique. On dit qu'il est *stable* (ou *bien posé*) si :

- $\forall n : \exists! x^{(n)}$, solution ;
-

$$\forall \epsilon > 0 : \exists \eta > 0, n_0 \in \mathbb{N}^* \text{ t. q. } \forall n > n_0 : \left(\forall i \in \{1, \dots, m\} : \left\|\delta d_i^{(n)}\right\| < \eta\right) \Rightarrow \left(\left\|\bar{x}^{(n)} - x^{(n)}\right\| < \epsilon\right).$$

Définition 1.14. Si l'opération $F_m^{(n)}(d)$ ne dépend ni de m, ni de n, on parle d'algorithme *itératif*.

Remarque. Un algorithme est donc dit itératif si c'est constamment la même opération qui est appliquée sur des données consécutives. On a alors :

$$x^{(n)} = f(x^{(n-1)}) = (f \circ f)(x^{(n-2)}) = \cdots = (f^n)\,(x^{(0)}) = (f^n)\,(d).$$

On peut traduire la définition de stabilité de manière plus informelle par le fait qu'un algorithme est stable s'il existe un voisinage de x, la solution tel que pour tout x_1, x_2 dans ce voisinage, on a un $R \in (0,1)$ tel que :

$$\left\|f(x_2) - f(x_1)\right\| \leqslant R\|x_2 - x_1\|\,.$$

Si f est une fonction différentiable, alors on peut dire que :

$$\left\|f'(x_2)\right\| \leqslant R.$$

2 Analyse et mesure d'erreurs

2.1 Définitions

Définition 2.1. $\mathbb{F}$ est l'ensemble des nombres représentés exactement par un ordinateur.

Remarque. On remarque aisément que $\mathbb{F} \subset \mathbb{R}$ car $\mathbb{F} \not\subset \mathbb{C} \setminus \mathbb{R}$. On peut tout de même raffiner car il semble évident que $\mathbb{F} \subset \mathbb{Q}$: un nombre irrationnel n'a pas de représentation physique finie, et ne peut donc pas être représentés par un ordinateur.

Définition 2.2. Soit x un nombre dans $\mathbb{Q}$. Une approximation de x est une valeur $\widehat{x}$ légèrement différente de x et qui remplace x dans les calculs effectués. On note également $x \simeq \widehat{x}$ pour montrer que $\widehat{x}$ est une approximation de x.

Si $\widehat{x} < x$, on parle d'approximation par défaut, et si $\widehat{x} > x$, on parle d'approximation par excès.

Définition 2.3. On définit l'*erreur absolue* entre x et son approximation $\widehat{x}$ par :

$$\delta_x = |\widehat{x} - x| .$$

On définit également l'*écart relatif* entre x et $\widehat{x}$ par :

$$\rho_x = \frac{\widehat{x} - x}{x}.$$

On définit alors l'écart absolue défini comme $\epsilon_x = |\rho_x|$.

Remarque. L'écart relatif permet d'écrire l'approximation sous la forme $\widehat{x} = x(1 + \rho_x)$ où ρ_x est censé être faible (car l'écart entre x et son approximation doit rester relativement faible). Cette notation est fortement utilisée.

Les écarts ne sont cependant définis qu'en $x \neq 0$, ce dont il faut tenir compte.

Définition 2.4. La *borne supérieure d'erreur relative* d'une approximation $\widehat{x}$, notée u est nombre positif quelconque tel que $u \geqslant \epsilon_x$.

Mis à part les erreurs de calcul (qui consistent à déterminer un résultat erroné sur base de données bien définies) et les erreurs d'implémentation (qui consiste en une implémentation erronée amenant à des erreurs de calcul), il existe cinq catégories d'erreurs que l'on peut regrouper en deux familles :

- les erreurs de modèle qui consistent en simplifier les phénomènes en omettant des éléments afin de le rendre plus facile à étudier ;
- les erreurs numériques qui consistent en l'approximation due à l'observation et non à la résolution théorique.

2.2 Les erreurs numériques[2]

Définition 2.5. Les *erreurs de troncature* sont les erreurs liées aux processus théoriques mathématiques infinis (série infinie par exemple).

[2]Les erreurs de modélisation ne sont pas traitées ici.

Définition 2.6. Les *erreurs d'arrondi* sont les erreurs liées au système numérique de la machine venant du fait qu'un ordinateur ne peut représenter qu'un sous-ensemble fini $\mathbb{F}$ des nombres.

Définition 2.7. Les *erreurs de génération et de propagation* sont les erreurs liées à l'évaluation d'une opération sur des opérandes pas exactes.

Remarque. Les erreurs de troncature sont compliquées à éviter : il est nécessaire d'avoir un procédé fini pour y échapper. Cette catégorie d'erreurs ne seront donc pas étudiées dans ce cours.

Les différents types d'erreurs concernent différentes propriétés des problèmes ou des algorithmes numériques utilisés.

Proposition 2.8.

- *Les erreurs d'approximation ou de troncature se mesurent par la* ***consistance*** *de l'algorithme ;*
- *les erreurs de propagation se mesurent par le* ***conditionnement*** *de l'algorithme ;*
- *les erreurs de génération se mesurent par la* ***stabilité de l'algorithme*** *;*
- *les erreurs d'arrondi sont liées à la représentation interne des nombres.*

2.3 Représentation interne des nombres par l'ordinateur

Il existe deux manières de représenter les nombres décimaux par un ordinateur : la *virgule fixe* et la *virgule flottante*.

2.3.1 Représentation en virgule fixe

Soit $x \in \mathbb{F}$ un nombre. Sa représentation en virgule fixe est donnée par le triplet $(\{a_i\}_{-m \leqslant i \leqslant n}, b, s)$ où :

- $b \geqslant 2$ est la base de représentation ;
- $s \in \{0, 1\}$ est le signe ;
- $\{a_{-m}, \ldots, a_n\}$ est l'ensemble de chiffres $0 \leqslant a_i \lneqq b$ pour tout i.

Remarque. m représente le nombre de chiffres suivant la virgule, et $n+1$ caractérise le nombre de chiffres précédant la virgule. On en déduit que la virgule est (implicitement) placée entre a_0 et a_{-1}.

On peut donc exprimer x par :

$$x = (-1)^s \sum_{k=-m}^{n} a_k b^k.$$

Remarque. La base choisie pour les ordinateurs est souvent 2, 8 ou 16 car 2 représente le binaire, et les bases octale et hexadécimale représentent chacune une écriture *compactée* du binaire par paquets de 3 ou 4 bits.

La base 10 est par fois également choisie.

Proposition 2.9 (Propriétés des nombres en virgule fixe)**.**

- *Les nombres représentés par virgule fixe sont équirépartis sur la droite des réels ;*
- *l'écart entre deux nombres consécutifs représentés en virgule flottante est* b^{-m}*, qui est le plus petit nombre représentable (en valeur absolue) ;*
- *et la plus grande valeur représentable est* $b^{n+1} - 1$*, où* $n+1$ *est le nombre de chiffres avant la virgule.*

Remarque. La représentation par virgule flottante limite fortement les valeurs extrêmes que peut prendre un nombre de par son équirépartition. Les nombres à virgule fixe (comme vu juste après) ont une densité de répartition différente, ce qui leur permet de prendre des valeurs maximum et minimum bien plus haute (en terme d'ordre de grandeur).

2.3.2 Représentation en virgule flottante

Soit $x \in \mathbb{F}$ un nombre. Sa représentation en virgule flottante est donnée par le quadruplet $(\{a_i\}_{1 \leqslant i \leqslant t}, b, s, e)$où :

- b et s représentent les mêmes quantités que pour la virgule fixe ;
- t est le nombre de *chiffres significatifs* ;
- $\{a_i\}_{1 \leqslant i \leqslant t}$ est l'ensemble des chiffres $0 \lneqq a_1 \lneqq b$ et $0 \leqslant a_i \lneqq b$ pour $i > 1$.

Remarque. On impose $a_1 \neq 0$ afin que la représentation d'un nombre soit unique.

Définition 2.10. La *mantisse* est la quantité définie par $\sum_{i=1}^{t} a_i b^{t-i}$. On la note $m \in \mathbb{N}$, ou encore $m[x] \in \mathbb{N}$ quand il est nécessaire de montrer que m est la mantisse de la valeur x.

Remarque. Dû à la normalisation des a_i, on sait borner m par :

$$b^{t-1} \leqslant m \leqslant b^t - 1.$$

On peut donc au final exprimer x comme :

$$x = (-1)^s b^e \sum_{i=1}^{t} a_i b^{-i} = (-1)^s b^e b^t b^{-t} \sum_{i=1}^{t} a_i b^{-i} = (-1)^s b^{e-t} \sum_{i=1}^{t} a_i b^{t-i} = (-1)^s b^{e-t} m.$$

Remarque. On note L et U les valeurs respectivement plus petite et plus grande que peut prendre la valeur e de l'exposant.

Proposition 2.11. *On paramétrise l'ensemble $\mathbb{F}$ par les quantités b, t, L, et U. Quand la paramétrisation doit être explicite, on note l'ensemble :*

$$\mathbb{F}(b, t, L, U).$$

Remarque. Dans une représentation normalisée, $0 \notin \mathbb{F}$.

Théorème 2.12. *Si x est un élément de $\mathbb{F}(b, t, L, U)$, alors on peut borner x tel que :*

$$b^{L-1} \leqslant |x| \leqslant b^U (1 - b^t).$$

Proposition 2.13. *Le cardinal de l'ensemble $\mathbb{F}(b, t, L, U)$ est donné par :*

$$\left|\mathbb{F}(b, t, L, U)\right| = 2(b-1)b^{t-1}(U - L + 1).$$

Remarque. Étant donné que l'exposant e est code sur un nombre n_e de bits (et représenté en complément à 2) dans une représentation machine de représentation flottante, on peut trouver que $L = -2^{n_e - 1} - 1 \leqslant e \leqslant 2^{n_e} = U$. On voit alors que les valeurs maximum prises par un nombre sont beaucoup plus grandes (en valeur absolue de l'ordre de grandeur) qu'en virgule fixe.

2.3.3 Distance relative dans $\mathbb{F}$

Prenons x_i et x_{i+1} dans $\mathbb{F}$. On note $\eta(x_i)$ la distance relative entre x_i et x_{i+1} que l'on définit :

$$\eta(x_i) : \left| \frac{x_{i+1} - x_i}{x_i} \right|.$$

Dans un système à virgule fixe, $\eta(x_i)$ est fixe pour tout i. Pour un système à virgule flottante, on trouve :

$$\eta(x_i) = \left| \frac{x_{i+1} - x_i}{x_i} \right| = \left| \frac{m[x_{i+1}]b^{e-t} - m[x_i]b^{e-t}}{m[x_i]b^{e-t}} \right| = \frac{1}{m[x_i]}.$$

Si x_{i+1} a un exposant plus grand que x_i, on a :

$$\eta(x_i) = \left| \frac{b^{t-1}b^{e-t+1} - (b^t - 1)b^{e-t}}{(b^t - 1)^{e-t}} \right| = \left| \frac{b^{e-t}}{(b^t - 1)b^{e-t}} \right| = \frac{1}{m[x_i]}.$$

On peut donc, en virgule flottante, également déterminer une distance générale (formule) mais qui n'est pas constante.

En sachant que :

$$b^{t-1} \leqslant m < b^t,$$

on peut trouver :

$$b^{-t} < \frac{1}{m} = \eta \leqslant b^{1-t}.$$

En divisant de part et d'autre par b^{1-t}, on trouve :

$$\frac{\epsilon}{b} < \eta(x_i) \leqslant \epsilon.$$

Définition 2.14. On appelle $\epsilon := b^{1-t}$ l'*epsilon machine* qui représente la distance maximale entre deux nombres consécutifs en virgule flottante.

Remarque. L'epsilon machine permet de déterminer $1 + \epsilon$ qui est le plus petit nombre $x \in \mathbb{F}$ tel que $x \gneqq 1$.
Remarque. Il ne faut pas confondre ϵ qui est la distance maximale entre deux nombres et b^{L-1} qui est le plus petit nombre que l'on peut représenter. En pratique, $\epsilon \simeq 10^{-16}$ alors que $b^{L-1} \simeq 10^{-308}$.

Définition 2.15. Le phénomène d'oscillation de $\eta(x_i)$ est appelé *wobbling precision* en anglais.

Remarque. Le phénomène de *wobbling precision* est d'autant plus grand que la base b est grande. C'est entre autres pour cela que les petites bases ($b = 2$) sont utilisées en pratique.

Définition 2.16. Soit $x \in \mathbb{R} \setminus \mathbb{F}$ un réel. x n'a pas de représentation exacte par l'ordinateur puisque $x \notin \mathbb{F}$.

- Si $|x| > b^U(1 - b^{-t})$, alors x est plus grand que le plus grand nombre que l'on peut représenter. On parle alors d'*overflow*. En cas d'overflow, habituellement, le système interrompt le programme.
- Si $|x| < b^{L-1}$, alors x est plus petit que le plus petit nombre possible à représenter. On parle alors d'*underflow*. En cas d'underflow, habituellement, le nombre x est arrondi à 0.
- Si x ne dépasse pas des bornes de représentation, on considère a_{t+1}, le chiffre suivant a_t dans la représentation théorique à précision infinie de x. On peut alors déterminer $\widehat{x}$ de deux manières :
  - par arrondi ;

– par troncature.

Un arrondi se fait par l'application $x \mapsto \widehat{x} = \mathrm{fl}(x)$, qui *remplace* a_t par a_t^* défini par :

$$a_t^* = \begin{cases} a_t & \text{si } a_{t+1} < \frac{b}{2}, \\ a_t + 1 & \text{si } a_{t+1} \geqslant \frac{b}{2}. \end{cases}$$

Si $a_{t+1} \geqslant \frac{b}{2}$ et $a_t = b - 1$, on réitère le processus pour a_{t-1} selon a_t, etc.

Une troncature se fait par l'application $x \mapsto \widehat{x} = \mathrm{tr}(x)$, qui conserve uniquement les t premiers chiffres de la notation théorique.

Définition 2.17. On définit ϵ_x l'erreur relative de x par :

$$\epsilon_x = \left| \frac{\mathrm{fl}(x) - x}{x} \right|.$$

On définit également la *précision machine* $u := \frac{\epsilon}{2}$.

Proposition 2.18. *Soit* $x \in \mathbb{R}$ *et* $\mathrm{fl}(x) \in \mathbb{F}$. *On a* $\epsilon_x \leqslant \frac{\epsilon}{2}$.

Preuve. $\left|\mathrm{fl}(x) - x\right|$ représente la distance entre x et son approximation. x est arrondi au nombre dans $\mathbb{F}$ le plus proche, et donc $\left|\mathrm{fl}(x) - x\right| \leqslant \frac{1}{2}|x_{i+1} - x_i|$ avec $x_i \leqslant x \leqslant x_{i+1}$. On trouve alors :

$$\epsilon_x = |\rho_x| \leqslant \frac{\epsilon}{2} = u.$$

□

Remarque. La précision machine u donne l'ordre de grandeur de la meilleure approximation possible pour un arrondi sur une machine donnée.

Remarque. On trouve alors :

$$\mathrm{fl}(x) = x(1 + \rho_x).$$

2.4 Erreurs de propagation et de génération

Les erreurs de propagation sont dues au conditionnement du problème. Les erreurs de génération quant à elles sont dues à la stabilité de l'algorithme mis en place pour la résolution du problème.

2.4.1 Erreurs de propagation

Soit F une fonction admettant une dérivée d'ordre 1 en d. Par Taylor (voir théorème 1.2), on sait que :

$$F(\widehat{d}) = F(d + d\rho_d) \simeq F(d) + F'(d)d\rho_d = F(d)\left(1 + \frac{F'(d)d}{F(d)}\rho_d\right) = x(1 + \kappa(d)\rho_d).$$

On appelle donc $\kappa(d)\rho_d$ l'*erreur de propagation*.

Il est également possible que $F(\widehat{d})$ subisse une approximation. On a alors :

$$\widehat{F(\widehat{d})} = x(1 + \kappa(d)\rho_d)(1 + \rho_{F(\widehat{d})}) = x(1 + \kappa(d)\rho_d + \rho_{F(\widehat{d})} + \kappa(d)\rho_d\rho_{F(\widehat{d})}).$$

On remarque cependant que $\rho_d \rho_{F(\widehat{d})}$ est négligeable face au reste (ordre de grandeur deux fois plus grand en valeur absolue) et donc,

$$\widehat{F(\widehat{d})} \simeq x(1 + \kappa(d)\rho_d + \rho_{F(\widehat{d})}).$$

Remarque. On dit que le conditionnement est lié aux erreurs de propagation car si $\kappa(d) > 1$, alors l'erreur d'arrondi initiale ρ_d va être amplifiée, ce qui est encore plus flagrant sur un algorithme itératif.

On ne peut pas changer le conditionnement d'un problème, et donc en cas de mauvais conditionnement, il est préférable de transformer le problème en un problème équivalent mais de plus faible conditionnement.

2.4.2 Erreurs de génération

Supposons que le problème $x = F(d)$ soit décomposé en n sous-étapes $x_i = F_i(x_{i-1})$ où $x_0 = d$. Soit $\widehat{d} = d(1 + \rho_d)$ la valeur encodée en machine de d. On détermine :

$$\widehat{x_1} = d(1 + \kappa_1(d)\rho_d + \rho_i + \kappa_1(d)\rho_d\rho_i) = x(1 + \rho_{x_1}).$$

En réitérant ceci n fois, on obtient :

$$\widehat{x} = x\left(1 + \rho_n + \sum_{i=1}^{n-1} \rho_{n-i} \prod_{k=i+1}^{n} \kappa_k(d) + \prod_{k=1}^{n} \kappa_k(d)\rho_d\right).$$

On comprend de cela que ρ_n est l'erreur d'arrondi final et ne peut donc être évité. De même, $\rho_d \prod_{k=1}^{n} \kappa_k(d)$ est l'erreur de propagation et dépend du conditionnement, qui dès lors ne peut être évitée. La somme des produits sur κ_k et ρ_{n-i} quant à elle est l'erreur de génération et est dépendante de l'algorithme choisi pour résoudre le problème. Les ρ_{n-i} dépendent des fonctions F_i choisies lors des décisions d'implémentation. Ces erreurs sont donc évitables (ou du moins contrôlables par les choix d'implémentation).
Remarque. Si l'algorithme a besoin de réaliser un calcul dangereux – tel qu'une soustraction de deux nombres proches – il est préférable que cette opération soit réalisée le plus tôt possible afin de limiter l'erreur relative.

3 Résolution de systèmes linéaires

3.1 Définitions

Définition 3.1. Un système de N équations linéaires à n inconnues est donné par $(S) : \sum_{i=1}^{n} a_{ij}x_i = b_i$ pour $j \in \{1, \cdots, N\}$.

Un tel système se note également sous forme matricielle par $Ax = b$ où $A \in \mathbb{R}^{N\times n}$, $x \in \mathbb{R}^n$ et $b \in \mathbb{R}^N$. Si $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ est un n-uple satisfaisant ces N conditions, alors x est une solutions de (S).

Remarque. Dans cette section, seuls les systèmes linéaires carrés ($N = n$) seront étudiés.

Définition 3.2. Soit A une matrice carrée. Le rang de A est la dimension de l'espace vectoriel engendré par ses vecteurs colonnes.

Proposition 3.3. *La solution d'un système linéaire carré* $(S) : Ax = b$ *existe et est unique si et seulement si une des conditions équivalentes suivantes est remplie :*

1. A *est inversible ;*
2. A *est régulière (*$\det(A) \neq 0$*) ;*
3. *le rang de* A *vaut* n *;*
4. *Le système homogène* $(\bar{S}) : Ax = 0$ *admet* $x = (0, \ldots, 0)$ *pour seule solution.*

Théorème 3.4 (Formule de Cramer)**.** *Soit* $(S) : Ax = b$ *un système linéaire carré. La solution est donnée par le vecteur* x *tel que :*

$$x_j = \frac{1}{\det(A)}\Delta_j,$$

où Δ_j *est le déterminant de la matrice obtenue en remplaçant la* j*ème colonne de* A *par le vecteur* b*.*

Remarque. Le calcul du déterminant est en $\mathcal{O}(n!)$ et est donc impraticable pour des n même relativement petits (et impensables pour $n \sim 10^5$, ce qui est l'ordre de grandeur de certains systèmes de résolutions d'équations différentielles partielles). On opte donc pour une résolution numérique du système.

Définition 3.5. Une méthode de résolution est dite *directe* si elle fournit la solution exacte en un nombre fini d'étapes. Elle est dite itérative si elle fournit une approximation convergente vers la solution.

3.2 Systèmes triangulaires

Définition 3.6. Une matrice carrée A est dite *triangulaire inférieure* si $\forall j > i : A_{ij} = 0$.

Définition 3.7. Une matrice carrée A est dite *triangulaire supérieure* si A^T est triangulaire inférieure.

Définition 3.8. Un système $(S) : Ax = b$ est dit triangulaire inférieur (respectivement supérieur) si sa matrice A est triangulaire inférieure (respectivement supérieure).

Remarque. Il est fréquent de noter les matrices triangulaires inférieures L (pour *lower*) et les matrices triangulaires supérieures U (pour *upper*).

Proposition 3.9 (Résolution d'un système triangulaire inférieur)**.** *Soit* $(S) : Lx = b$ *un système triangulaire inférieur. On trouve (dans l'ordre croissant, à savoir pour* j *de* 1 *à* n*) les solutions de la manière suivante :*

$$x_j = \frac{(b_j - \sum_{k=1}^{j-1} x_k L_{jk})}{L_{kk}}. \tag{1}$$

Corollaire 3.10. *La résolution d'un système triangulaire supérieur* $(S) : Ux = b$ *se fait de la même manière mais dans l'ordre décroissant, c'est-à-dire pour* j *de* n *à* 1*. (Cela revient, théoriquement, à dire que la solution est la transposée de la solution de la transposée.)*

Définition 3.11. On définit le *flop* comme étant une unité de mesure des opérations à virgule flottante tel qu'une opération est 1 flop.

3.2.1 Analyse du coût calculatoire de la résolution d'un système triangulaire

Par l'équation (1), on voit que pour chaque x_j, il y a à opérer $(j-1)$ multiplications, $(j-1)$ additions, et une division. La résolution d'un système carré de dimension $n \times n$ nécessite donc, au total, $n + \frac{n(n-1)}{2} + \frac{n(n-1)}{2} = n^2$ opérations. On peut donc exprimer le coût calculatoire par n^2 flops.

Définition 3.12. Soient $(S_1) : A_1x = b_1$ et $(S_2) : A_2x = b_2$ deux systèmes linéaires d'équations. On dit qu'ils sont équivalents si leur ensemble de solution respectif est identique.

Proposition 3.13. *Soit* $(S) : Ax = b$ *un système. Il existe trois opérations fondamentales permettant de modifier un système en un autre système équivalent. Ces opérations sont :*

1. *échanger deux lignes ;*
2. *multiplier une ligne par une constante ;*
3. *ajouter à une ligne une combinaison linéaire des autres.*

Théorème 3.14 (Méthode de Gauss de diagonalisation de matrice)**.** *La méthode de gauss permet de diagonaliser une matrice carrée. Une fois la matrice carrée diagonalisée, la solution est immédiate.*

La méthode de diagonalisation est la suivante : tout d'abord, on ajoute *le vecteur* n *en* $n+1$*ème colonne de* A*. Ensuite, pour chaque ligne* i *(une par une), on prend l'élément* A_{ii}*. S'il est nul, on échange la* i*ème ligne avec la première ligne* $j > i$ *telle que* $A_{ji} \neq 0$*. Si un tel* j *n'existe pas, le déterminant est nul et donc la matrice n'est pas inversible, le système n'a donc pas de solution (du moins unique).Une fois qu'il est assuré que* $A_{ii} \neq 0$*, on remplace toutes les colonnes* $j \neq i$ *par* $A_j - \frac{A_{ii}}{A_{ji}} A_i$*. Une fois toutes les lignes traitées, la matrice est diagonalisée.*

3.2.2 Analyse du coût calculatoire de la méthode de Gauss

On peut limiter l'élimination de Gauss en triangulant la matrice et pas en la diagonalisant. Dans ce cas, à chaque colonne i à normaliser, il y a $(n - i)$ divisions à opérer. Le nombre de divisions est donc $\frac{n(n-1)}{2}$.

Il y a une multiplication à chaque élément normalisé (en considérant le coefficient $\frac{A_{ii}}{A_{ji}}$ calculé par la division comptée juste au-dessus). Il y a $(n-1)$ lignes, et pour la $i$ème d'entre elles, il y a $(n-i)(n-i+1)$ normalisations. Il en est de même pour les additions et soustractions. Le coût total est donc :

$$\begin{aligned}\sum_{k=1}^{n-1} 1 + 2\sum_{k=1}^{n-1}(n-k)(n-k+1) &= \frac{n(n-1)}{2} + 2\sum_{k=1}^{n-1} k(k+1) = \frac{n(n-1)}{2} + 2\sum_{k=1}^{n-1}(k^2+k)\\ &= \frac{n(n-1)}{2} + 2\sum_{k=1}^{n-1} k^2 + 2\sum_{k=1}^{n-1} k = \frac{3n(n-1)}{2} + 2\frac{n(n-1)(2n-1)}{6}\\ &= \frac{n(n-1)}{2}\big(9 + 2(2n-1)\big) = \frac{n(n-1)}{2}(4n+7) = \frac{2n^3}{3} + \frac{7n^2}{6} - \frac{4n^2}{6} - \frac{7n}{6}\\ &= \frac{2n^3}{3} + \frac{n^2}{2} - \frac{7n}{6}.\end{aligned}$$

En rajoutant à cela les n^2 flops de résolution de la matrice triangulaire, on obtient un coût total de résolution de $\frac{2n^3}{3} + \frac{3n^2}{2} + \frac{7n}{6}$.

3.3 Factorisation de matrices

3.3.1 Factorisation LU

Définition 3.15. On définit l'application δ sur un ensemble quelconque E telle que :

$$\delta : E \times E \to E : (x, y) \mapsto \delta_{x\,y} \coloneqq \begin{cases} 1 & \text{si } x = y,\\ 0 & \text{sinon.}\end{cases}$$

Définition 3.16. Soit A une matrice triangulaire (inférieure ou supérieure). Si pour tout i, on a : $A_{i\,i} = 1$, on dit que A est triangulaire *atomique*.

Proposition 3.17. *L'inverse A^{-1} d'une matrice triangulaire atomique A est également triangulaire atomique.*

Il peut être intéressant de décomposer la matrice carrée A d'un système par un produit de deux matrices L et U respectivement triangulaire inférieure et triangulaire supérieure, où $\forall i : L_{i\,i} = 1$.

Il existe plusieurs méthodes (dont certaines directes) pour procéder à cette factorisation, mais commençons par l'adaptation de la méthode de Gauss pour créer les matrices pendant la triangulation.

À l'étape de la renormalisation de la $i$ème colonne, il y a $(n - i)$ lignes à remplacer par une combinaison linéaire des autres. Cette renormalisation de la ligne peut s'écrire :

$$\forall j > i, k \geqslant i : A_{j\,k} \leftarrow A_{j\,k} - \frac{A_{j\,i}}{A_{i\,i}} A_{i\,k}.$$

On peut noter cette opération de manière matricielle par :

$$A \leftarrow M^{(i)} A,$$

où $M^{(i)}$ est la $i$ème matrice de diagonalisation et vaut :

$$M^{(i)} = \left[M^{(i)}_{k\,l}\right]_{k\,l} = \left[\delta_{k\,l} - \delta_{i\,l}\frac{A_{k\,i}}{A_{i\,i}}\right]_{k\,l}$$

Remarque. La matrice inverse de $M^{(i)}$ est la matrice $\left(M^{(i)}\right)^{-1}$ définie par :

$$\left(M^{(i)}\right)^{-1} = \left[\delta_{kl} + \delta_{il}\frac{A_{ki}}{A_{ii}}\right]_{kl} = (2I - M^{(i)}).$$

Cela est intuitivement vrai du fait que la matrice inverse représente l'application inverse. Et l'application inverse du fait de retirer α_{ki} fois la ligne i est de l'ajouter α_{ki}, où α_{ki} représente le coefficient $\frac{A_{ki}}{A_{ii}}$.

On peut nommer les matrices intermédiaires $A^{(k)}$. On a donc $A^{(0)} = A$, $A^{(1)} = M^{(1)}A$, $A^{(2)} = M_2A^{(1)} = M_2M_1A$, etc. que l'on peut généraliser en :

$$A^{(k)} = M^{(k)}M^{(k-1)}\cdots M^{(2)}M^{(1)}A = \left(\prod_{\gamma=0}^{k} M^{(k-\gamma)}\right)A.$$

La matrice finale est une matrice triangulaire supérieure (algorithme de Gauss vu plus haut). On peut donc poser $U = A^{(n)}$ si n est la dimension de la matrice. On pose alors :

$$L \coloneqq (M^{(k)}M^{(k-1)}\cdots M^{(2)}M^{(1)})^{-1} = \left(\left(M^{(1)}\right)^{-1}\left(M^{(2)}\right)^{-1}\cdots\left(M^{(k-1)}\right)^{-1}\left(M^{(k)}\right)^{-1}\right).$$

Remarque. Étant donné que les M_i sont atomiques inférieures, leur inverse l'est également. Et la *composition* de ces matrices atomiques est également une matrice atomique inférieure. Dès lors, on sait que L est atomique inférieure et peut satisfaire la factorisation.

On a donc finalement bien $A = LU$ avec U triangulaire supérieure et L triangulaire inférieure atomique.
Remarque. L'intérêt de cette décomposition LU est qu'une fois la factorisation faite, la résolution du système $(S) : Ax = b$ se fait par la résolution de deux système triangulaires, à savoir :

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

Supposons qu'il y ait $p \in \mathbb{N}^*$ systèmes à résoudre ayant tous la même matrice caractéristique A. Cette suite de problèmes peut donc se noter :

$$Ax^{(k)} = b^{(k)}.$$

Ainsi, en appliquant p fois Gauss, asymptotiquement, le terme n^2 de Gauss devient négligeable, et on obtient : $p\mathcal{O}(n^3)$ flops. En appliquant une fois la décomposition et en utilisant ces deux matrices pour la résolution de sous-systèmes triangulaires, on obtient : $\mathcal{O}(n^3 + pn^2)$. Par définition du grand O, c'est équivalent à $\mathcal{O}(n^3)$. Asymptotiquement, la factorisation rend le programme p fois plus rapide (du moins, moins coûteux en opérations).

3.3.2 Pivots de la méthode de Gauss

Lors d'une élimination de Gauss, si le pivot A_{kk} vaut 0 à la kème étape, l'algorithme ne peut fonctionner. Numériquement, quand A_{kk} est proche de 0, l'algorithme risque de mal se dérouler.

Changement de pivot partiel Pour le changement *partiel* de pivot, il faut trouver, dans la $k$ème colonne (en dessous de $A_{k\,k}$) un élément $A_{p\,k} \neq 0$. Et ensuite, il faut échanger les lignes p et k dans la matrice A **et** dans le vecteur $b^{(k)}$[3].

Cela évite d'avoir un pivot nul. Cependant, on a dit que numériquement, un pivot *proche* de 0 était *dangereux*. Dès lors, on cherche le plus grand $A_{p\,k}$ (en valeur absolue).

Changement de pivot total Pour le changement *total*, le plus grand élément (en valeur absolue) ne doit pas être cherché dans la $k$ème colonne mais bien dans la sous-matrice $[N_{i\,j}]_{k\leqslant i\leqslant n, k\leqslant j\leqslant m}$. Une fois le plus grand élément $A_{p\,r}$ trouvé, à nouveau, on échange les lignes p et k, mais on échange également les colonnes k et r.
Remarque. Si la recherche d'un pivot est faite à toutes les étapes (et que la recherche est supposée en $\mathcal{O}(n)$), un pivot partiel amènerait une complexité en $\mathcal{O}(n^2)$ et un pivot total impliquerait une complexité en $\mathcal{O}(n^3)$ (toutes opérations confondues, tant pour l'un que pour l'autre).

Notons $P^{(i)}$ la matrice de permutations à la $i$ème étape. La matrice de permutation[4] est celle qui permet d'échanger les lignes (et éventuellement colonnes) de A pour le $i$ème pivot. On remarque que cette matrice est *auto-inverse*. En effet, $P^{(i)}\left(P^{(i)}\right)^{-1}$ représente un échange de deux lignes (et éventuellement deux colonnes) suivi de l'opération inverse. Or pour annuler une permutation, il faut la réeffectuer.

La matrice U se remplit lors de l'exécution de l'algorithme de Gauss. Dès lors, on pouvait écrire $A = LU$ s'il n'y avait pas de permutations car tous les pivots étaient non-nuls. Avec la notation $P^{(i)}$, on peut réécrire :

$$U = \left(\prod_{i=1}^{n} M^{(n+1-i)} P^{(n+1-i)}\right) A.$$

U est toujours une matrice triangulaire supérieure (par construction). Cependant, la matrice $L' \coloneqq AU^{-1}$ n'est plus triangulaire inférieure, mais en est une permutation. Dès lors, on note $P \coloneqq P^{(n)}P^{(n-1)}P^{(n-2)}\dots P^{(2)}P^{(1)}$. On définit alors la matrice L par :

$$L = PAU^{-1}.$$

On a alors l'égalité suivante :

$$PA = LU.$$

Remarque. Même s'il est assez coûteux d'effectuer les pivots à chaque étape, il est préférable lorsque le résultat doit être assez précis de tout de même les appliquer. Les deux raisons principales sont :

1. le conditionnement (erreur de propagation) car un petit pivot peut entraîner une très grande erreur relative, or le pivot est au centre de tout l'algorithme, ce qui risquerait de fausser toutes les valeurs ;
2. la stabilité (erreur de génération) car un petit pivot pourrait amener des produits à donner des résultats très bas qui pourraient être assimilés à 0 par absorption.

On sait que le déterminant d'une matrice triangulaire est le produit des éléments de sa diagonale. Donc :

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

Si on appelle D la matrice définie par :

$$D = [D_{i\,j}] = [\delta_{i\,i}U_{i\,i}].$$

[3] Et donc dans la matrice étendue utilisée par l'algorithme de Gauss.
[4] Qui est une permutation de la matrice identité.

Il existe des méthodes directes pour la factorisation LU (contrairement à la construction par l'algorithme de gauss qui est itérative). Ces méthodes doivent fixer n éléments afin de permettre la résolution. En effet, la factorisation $A = LU$ revient à résoudre un système de $n(n+1)$ variables et n^2 équations. En fixant n inconnues, on obtient un système de n^2 équations à n^2 inconnues qui possède une solution unique si ces équations forment une partie libre.

L'algorithme de Doolittle (voir juste ci-dessous) fixe les éléments de la diagonale L, mais il est possible également de fixer les éléments de la diagonale de U par exemple, c'est ce que fait la méthode de Crout.

3.3.3 Algorithme de Doolittle

En écrivant $LU = A$ avec $L_{ii} = 1$ pour $i = 1, \dots, n$, on peut résoudre toutes les équations dans l'ordre suivante :

- la 1ère ligne de U par $U_{1i} = A_{1i}$;
- la 1ère colonne de L par $L_{i1} = \frac{A_{i1}}{U_{11}}$;
- la 2ème ligne de U ;
- la 2ème colonne de L ;
- etc.

3.3.4 Algorithme de Crout

Le principe est exactement le même que pour l'algorithme de Doolittle, sauf que ce ne sont pas les L_{ii} qui sont imposés à 1 mais bien les U_{ii}. La méthode de résolution est similaire mais se fait dans l'ordre inverse :

- la 1ère colonne de L ;
- la 1ère ligne de U ;
- la 2ème colonnes de L ;
- la 2ème ligne de U ;
- etc.

3.3.5 Factorisation de Cholesky

Prenons $(S) : Ax = b$, un système linéaire tel que A est définie positive et symétrique. Ce genre de systèmes est fréquent en statistiques de simulation.

Théorème 3.18. *Soit $A \in \mathbb{R}^{n \times n}$ symétrique et définie positive. Alors il existe une unique matrice H triangulaire supérieure telle que sa diagonale est définie positive et $A = H^T H$.*

La méthode de Cholesky permet de déterminer cette matrice H symétrique en la déterminant ligne par ligne.
Remarque. Les équations sont données explicitement en écrivant $A = H^T H$.

Tout comme pour la factorisation LU, la factorisation de Cholesky permet de scinder le système (S) en deux systèmes triangulaire qui sont beaucoup plus simples à résoudre. On a effectivement :

$$Ax = b \iff H^T Hx = b \iff \begin{cases} H^T y & = b \\ Hx & = y \end{cases}.$$

Le coût numérique de cet algorithme est de $\mathcal{O}(\frac{n^3}{6})$ produits et additions, n racines carrées et $\mathcal{O}(\frac{n^2}{2})$ divisions. On peut donc dire que Cholesky se fait en $\mathcal{O}(\frac{n^3}{3})$, comparé à la méthode de Gauss qui se fait en $\mathcal{O}(\frac{2n^3}{3})$, sans compter les pivotages.
Remarque. De plus, la place de stockage nécessaire est réduite de moitié car il n'y a qu'une seule matrice à déterminer (H) et plus deux (L et U). De plus, A est supposée symétrique, et donc A et H peuvent toutes deux être stockées dans une matrice carrée $n \times n$.

3.4 Analyse des propriétés des algorithmes de factorisation matricielle

3.4.1 Analyse du conditionnement

Définition 3.19. Soit A une matrice réelle carrée $n \times n$ et soit p un naturel non nul. On définit la p-norme matricielle par :

$$\|A\|_p \coloneqq \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|Ax\|_p}{\|x\|_p}.$$

Quand $p = 2$, on note :

$$\|A\| \coloneqq \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|Ax\|}{\|x\|}.$$

Soient A et b les données d'un système linéaire (S) : $Ax = b$. On sait que si x est perturbé par δx, on a :

$$A(x + \delta x) = b + \delta b.$$

Or dans notre cas, c'est x que l'on cherche. On veut donc savoir comment se comporte δx. On sait que $Ax = b$, et donc $A\delta x = \delta b$, ou encore $\delta x = A^{-1}\delta b$. Par définition de la norme matricielle, on sait :

$$\|\delta x\| = \left\|A^{-1}\delta b\right\| \leqslant \left\|A^{-1}\right\| \|\delta b\|.$$

On sait également que $\|b\| \leqslant \|A\|\|x\|$ et donc $\frac{1}{\|x\|} \leqslant \frac{\|A\|}{\|b\|}$.

On peut alors définir le conditionnement, par définition :

$$\kappa(A) = \max_x \frac{\frac{\|\delta x\|}{\|x\|}}{\frac{\|\delta b\|}{\|b\|}} = \frac{1}{\frac{\|\delta b\|}{\|b\|}} \left\|A^{-1}\right\| \|\delta b\| \frac{\|A\|}{\|b\|} = \left\|A^{-1}\right\| \|A\|.$$

Définition 3.20. Le p-conditionnement κ_p de la matrice A est donné par :

$$\kappa_p(A) = \|A\|_p \left\|A^{-1}\right\|_p.$$

Remarque. On remarque que $1 = \|I\| = \left\|A^{-1}A\right\| \leqslant \|A\|\left\|A^{-1}\right\| = \kappa(A)$. Le conditionnement est donc toujours supérieur à 1. On remarque également que :

$$\kappa_p(A) = \left\|A^{-1}\right\|_p \|A\|_p = \|A\|_p \left\|A^{-1}\right\|_p = \kappa_p(A^{-1}).$$

De plus, on aimerait que multiplier la matrice A par une constante ne change pas le conditionnement car le problème reste intrinsèquement le même. En effet, $\kappa_p(\alpha A) = \|\alpha A\|_p \left\|(\alpha A)^{-1}\right\|_p = \alpha \|A\|_p \frac{1}{\alpha}\left\|A^{-1}\right\|_p = \kappa_p(A)$.

Définition 3.21. Soit $A \in \mathbb{R}^{n\times n}$ une matrice carrée. On appelle *vecteur propre* de A tout vecteur x tel qu'il existe $\lambda \in \mathbb{R}$ tel que $Ax = \lambda x$.

Définition 3.22. On appelle valeur propre de A tout $\lambda \in \mathbb{R}$ tel qu'il existe $x \in \mathbb{R}^n$ tel que $Ax = \lambda x$.

Remarque. Même si le nombre de valeurs propres d'une matrice est fini, une valeur propre engendre un sous-espace vectoriel de vecteurs propres.

En effet, si x est un vecteur propre de A, alors il existe $\lambda \in \mathbb{R}$ tel que $Ax = \lambda x$. On en déduit donc que pour tout $\mu \in \mathbb{R}$, $A(\mu x) = \mu(Ax) = \mu\lambda x = \lambda(\mu x)$ et donc μx est également un vecteur propre de A.

Définition 3.23. Si λ est une valeur propre de $A \in \mathbb{R}^{n\times n}$ et $\lambda > 0$, alors $\sigma \coloneqq \sqrt{\lambda}$ est appelée une *valeur singulière* de A.

Définition 3.24. Si $A \in \mathbb{R}^{n\times n}$ est une matrice carrée de valeurs propres $\lambda_1, \ldots, \lambda_m$, on définit son *rayon spectral* par :

$$\rho(A) \coloneqq \max_k |\lambda_k|.$$

Remarque. On peut définir le conditionnement à l'aide de ces définitions :

$$\kappa(A) = \rho(A)\rho(A^{-1}) = \frac{\sigma_{max}}{\sigma_{min}}.$$

3.4.2 Analyse de la stabilité

À cause des erreurs d'arrondi, une méthode numérique fournit une solution approchée $\widehat{x} = x + \delta x$. Procédons à une analyse directe de la stabilité. Supposons que A et b soient représentés de manière approchée par $\widehat{A} = A + \delta A$ et $\widehat{b} = b + \delta b$.

On a alors :

$$(A + \delta A)(x + \delta x) = \widehat{A}\widehat{x} = \widehat{b} = b + \delta b.$$

Or on sait que $Ax = b$. Dès lors :

$$Ax + \delta Ax + A\delta x + \delta A\delta x = b + \delta b \qquad \Longleftrightarrow \qquad \delta Ax + (A + \delta A)\delta x = \delta b.$$

Théorème 3.25. *Soit $A \in \mathbb{R}^{n\times n}$ une matrice carrée régulière et δA une perturbation telles que :*

$$\left\|A^{-1}\right\|\|\delta A\| \lneqq 1.$$

Alors :

$$\frac{\|\delta x\|}{\|x\|} \leqslant \frac{\kappa(A)}{1 - \kappa(A)\frac{\|\delta A\|}{\|A\|}}\left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|}\right).$$

Remarque. La preuve de ce théorème est omise.

Corollaire 3.26. *Si les conditions du théorème précédent sont respectées et* $\|\delta A\| = 0$*, alors :*

$$\frac{1}{\kappa(A)}\frac{\|\delta b\|}{\|b\|} \leqslant \frac{\|\delta x\|}{\|x\|} \leqslant \kappa(A)\frac{\|\delta b\|}{\|b\|}.$$

Remarque. Ce corollaire dit que pour un grand conditionnement $\kappa(A)$, la borne supérieure pour $\frac{\|\delta x\|}{\|x\|}$ devient très grande. Cependant, il dit également que la borne inférieure devient très petite (et tend vers 0 pour $\kappa(A) \to +\infty$).

On en déduit qu'un conditionnement élevé n'implique *pas obligatoirement* une grande perturbation δx sur la solution déterminée.

On a pu remarquer que le calcul du p-conditionnement $\kappa_p(A)$ pour l'analyse du problème requiert le calcul de $\left\|A^{-1}\right\|_p$, et donc de A^{-1} qui est assez coûteux. L'analyse directe de stabilité vue ci-dessus requiert le conditionnement et donc par extrapolation nécessite également le calcul de A^{-1}.
Remarque. Il y a moyen, d'effectuer des estimations du conditionnement, par exemple à l'aide de la méthode LAPACK[5]. Ceci permet de déterminer une valeur approchée du conditionnement en $\mathcal{O}(n^2)$, ce qui est largement préférable à $\mathcal{O}(n^3)$ nécessaire pour inverser la matrice A.

On peut cependant procéder à une analyse à posteriori (contrairement à l'analyse à priori faite juste avant)[6].

Définition 3.27. On définit le *résidu* comme étant l'écart entre la valeur théorique b et le résultat de l'utilisation de la solution déterminée $A\widehat{x}$. On l'appelle r :

$$r \coloneqq b - A\widehat{x}.$$

Définition 3.28. On définit également l'erreur par $e \coloneqq x - \widehat{x}$.

L'objectif de l'analyse à posteriori est de lier la précision (et donc la fiabilité) de la solution finale et le résidu.

On peut en effet déterminer :

$$Ae = A(x - \widehat{x}) = b - A\widehat{x} = r.$$

On peut alors exprimer :

$$\frac{\|e\|}{\|x\|} \leqslant \left\|A^{-1}\right\| \|r\| \frac{\|A\|}{\|b\|} = \kappa(A)\frac{\|r\|}{\|b\|}.$$

Remarque. **Attention**, il est important de préciser qu'un petit résidu n'implique pas une petite erreur.

La donnée inconnue recherchée est e car b et A sont connus, et donc r est connu également. On procède alors à un *raffinement itératif*. On pose initialement $x^{(0)} = \widehat{x}$. On définit une borne supérieure pour l'écart admissible. On l'appelle ε. Ensuite, on procède au raffinement défini par :

$$\begin{cases} r^{(i)} & = b - Ax^{(i)} \\ Ae^{(i)} & = r^{(i)} \\ x^{(i+1)} & = x^{(i)} + e^{(i)} \end{cases}$$

[5] Algorithme implémenté dans MATLAB.

[6] L'analyse à priori permet d'évaluer, avant de lancer la résolution, quelle va être la sensibilité de la résolution aux problèmes d'arrondis, et donc possiblement d'améliorer l'implémentation afin d'en éviter les conséquences néfastes. Une analyse à posteriori est moins coûteuse de manière générale et se fait une fois la résolution terminée. Elle permet d'évaluer la pertinence de la solution suivant la sensibilité aux arrondis.

tant que $\left\|e^{(i)}\right\| > \varepsilon \left\|x^{(i)}\right\|$.

Remarque. Ce *raffinement itératif* tient son nom du fait qu'il permet de compenser les erreurs faites pendant la résolution du système linéaire. C'est ainsi que l'on peut noter $x^{(0)} = \widehat{x}$, il faut d'abord avoir une solution approchée par une méthode (telle que l'algorithme de Gauss, ou une factorisation quelconque permettant de résoudre des systèmes triangulaires) classique. Ensuite, l'objectif est d'obtenir une meilleure solution.

Si la matrice A n'est *pas trop mal* conditionnée, alors la convergence pour x va assez vite (quelques étapes seulement).

3.5 Méthodes itératives

Définition 3.29. Soit $(S) : Ax = b$ un système d'ordre n. La forme générale de résolution itérative d'ordre m est :

$$\begin{aligned} x^{(0)} &= f_0(A, b) \\ x^{(k)} &= f_k(x^{(k-1)}, x^{(k-2)}, \dots, x^{k+1-m}, A, b) \qquad \text{pour } k \geqslant m-1 \end{aligned}$$

Si les fonctions f_k ne dépendent pas de k, on dit que la méthode est *stationnaire*. Sinon, on la dit *non-stationnaire*.

Les procédés itératifs sont convergents en l'infini (par définition). En pratique, on fixe une limite ε en dessous de laquelle il faut abaisser $\left\|x^{(k)} - x\right\|$ afin d'accepter la solution. Plus ε est petit, plus le nombre d'itérations sera élevé (de manière générale).

Définition 3.30. Si $A \in \mathbb{R}^{n\times n}$ est une matrice avec un nombre d'éléments nuls proche de n^2, on dit que A est une matrice *creuse*.

Remarque. Les méthodes itératives sont particulièrement efficaces dans le cas de matrices creuses. En effet, lors d'une factorisation (par exemple $A = LU$), on perd la structure creuse intéressante.[7] On préfère donc utiliser une méthode itérative qui, quant à elle, ne modifiera pas la matrice A.

Définition 3.31. Soit $x^{(0)} \in \mathbb{R}^n$ un vecteur initial. On définit la méthode itérative linéaire du premier ordre par :

$$x^{(k)} \coloneqq Bx^{(k-1)} + f,$$

où $B \in \mathbb{R}^{n\times n}$ est appelée la *matrice d'itération*.

Définition 3.32. La méthode est dite *consistante* si pour x, solution de $Ax = b$, on a $x = Bx + f$, ou encore si :

$$f = (I - B)A^{-1}b.$$

Définition 3.33. La méthode est dite *convergente* si :

$$\lim_{k\to+\infty} e^{(k)} = \lim_{k\to+\infty} x^{(k)} - x = 0.$$

On remarque que si la méthode est consistante, elle n'est pas obligatoirement convergente.

[7] On parle alors de *fill-in* qui est le processus observé lorsqu'une matrice creuse perd sa structure en étant remplie aux endroits où les éléments étaient nuls.

Proposition 3.34. *Soit* $x^{(k)} = Bx + f$ *une méthode itérative linéaire. Si la méthode est consistante, alors pour tout* $k \geqslant 1$ *:*

$$e^{(k)} = B^{(k-1)}e^{(0)}.$$

Preuve. Prouvons-cela par récurrence sur k. Pour le cas initial, prenons $k = 1$. On a alors :

$$e^{(1)} = x^{(1)} - x = (Bx^{(0)} + f) - (Bx + f) = B(x^{(0)} - x) = B^1 e^{(0)}.$$

Supposons la formule vraie pour $k-1$ et montrons qu'elle est vraie pour k. On calcule :

$$e^{(k)} = x^{(k)} - x = (Bx^{(k-1)} + f) - (Bx + f) = Bx^{(k-1)} - Bx = Be^{(k-1)} = B(B^{k-1}e^{(0)}) = B^k e^{(0)}.$$

□

Théorème 3.35. *Soit* $B \in \mathbb{R}^{n\times n}$*, une matrice réelle carrée. Alors :*

$$\lim_{k\to+\infty} B^k = 0 \quad \iff \quad \rho(B) \lneqq 1.$$

Corollaire 3.36. *Une méthode consistante est convergente si et seulement si* $\rho(B) \lneqq 1$*.*

Remarque. $\rho(B)$ (le rayon spectral) est également appelé *facteur de convergence asymptotique*.
Remarque. On sait montrer que pour toute matrice B, son rayon spectral est inférieur à sa norme. On en déduit que si $\|B\| < 1$ dans une méthode consistante, alors la méthode est convergente.

Si le rayon spectral de la matrice d'itération est petit, l'algorithme convergera plus vite. De plus, la caractéristique de convergence d'une méthode itérative peut dépendre de certaines caractéristiques de matrices, et donc une méthode itérative peut converger pour une certaine famille de matrices et diverger pour une autre.

3.5.1 Méthode de Jacobi

La méthode de Jacobi consiste à réécrire le système (où l'on considère que la diagonale de A ne s'annule pas) :

$$\sum_{j=1}^{n} A_{ij}x_j = b_i \qquad \text{pour } i = 1, \dots, n$$

par une écriture identique :

$$x_i = \frac{1}{A_{ii}}\left(b_i - \sum_{j\neq i} A_{ij}x_j\right) \qquad \text{pour } i = 1, \dots, n.$$

On en fait une méthode itérative en la récrivant, avec $k \geqslant 1$:

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

L'idée est donc d'utiliser les valeurs approchées déterminées à l'étape $k-1$ pour les raffiner et en déterminer les valeurs approchées à l'étape k. Cela peut s'écrire de manière lus formelle à l'aide des matrices[8] :

$$x^{(k)} = D^{-1}\left((D - a)x^{(k-1)} + b\right),$$

[8]Les inversions de matrices sont de manière générale à éviter car $\mathcal{O}(n^3)$, mais dans le cas d'une matrice diagonale, la matrice inverse revient à prendre l'inverse multiplicatif des éléments de la diagonale. Il est donc plus aisé de la calculer, même numériquement (bien que l'inverse multiplicatif d'un très grand nombre sera très petit, et on risque les erreurs d'absorption).

où $D = \text{diag}(A)$ est diagonale. On en déduit que la matrice d'itération B_J (en indice, J pour Jacobi) vaut $B_J = D^{-1}(D - A) = I - D^{-1}A$.

3.5.2 Méthode de Gauss-Seidel

La méthode de Gauss-Seidel est très proche de la méthode de Jacobi, à la différence près que les éléments déjà calculés à l'étape k sont utilisés pendant l'étape k alors que dans la méthode de Jacobi, les éléments de l'étape k ne sont utilisés qu'à partir du début de l'étape $k+1$.

Plus formellement, la méthode de Gauss-Seidel peut s'exprimer comme suit :

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

De manière matricielle, cela se note :

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

où D est toujours la diagonale de A et où $D - A = E + F$. On trouve alors la matrice d'itération de Gauss-Seidel qui est donnée par :

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

Remarque. Ces deux méthodes sont suffisamment différentes pour que la convergence de la méthode Jacobi n'implique pas la convergence de la méthode Gauss-Seidel, et inversement. Cependant, de manière générale, sur une matrice pour laquelle les deux méthodes vont converger, typiquement Gauss-Seidel aura un taux de convergence supérieur.

Définition 3.37. Soit A une matrice $n \times n$. On dit que A est à *diagonale dominante* si tout élément de sa diagonale est supérieur (en valeur absolue) à la somme de tous les autres termes de sa ligne (en valeurs absolues toujours). Plus formellement :

$$\forall i \in \{1, \ldots, n\} : |A_{ii}| \geqslant < \sum_{j \neq i} |A_{ij}| .$$

Si A^T est à diagonale dominante, on dit que A est à diagonale dominante *par colonnes*.

On dit que A est à diagonale dominante *stricte* si les inégalités sont strictes.

Théorème 3.38. *Si $A \in \mathbb{R}^{n \times n}$ est une matrice à diagonale dominante stricte (par lignes ou par colonnes), alors les méthodes Jacobi* et *Gauss-Seidel sont convergentes pour A.*

Théorème 3.39. *Si A et $(2D - A)$ sont symétriques*[9] *et définies positives*[10]*, alors les méthodes Jacobi et Gauss-Seidel sont convergentes pour A.*

Théorème 3.40. *Si A est une matrice symétrique, alors la méthode Gauss-Seidel est convergente pour A.*

Remarque. On remarque que la méthode Jacobi requiert plus de conditions pour être convergente, ce qui peut partiellement expliquer se taux de convergence plus élevé de Gauss-Seidel.

[9] Une matrice carrée A est symétrique si $A^T = A$.

[10] Si pour tout vecteur x, on a $x^T A x > 0$, on dit que A est définie positive.

4 Résolution d'équations différentielles ordinaires

4.1 Introduction

Définition 4.1. Une *équation différentielle* est une équation faisant intervenir une fonction inconnue et une ou plusieurs de ses dérivées.

Remarque. Beaucoup de lois physiques peuvent s'exprimer en termes de *taux de variation*, ce qui est particulièrement adapté pour les équations différentielles (également appelées ED ou équadiffs).

Définition 4.2. Une EDO (*équadiff ordinaire*) du premier ordre en la fonction $y(t)$ est un équation sous la forme :

$$a_0(t)y(t) + a_1(t)y'(t) = p(t).$$

Si $p(x) = 0$ pour tout x, alors on parle d'équation homogène.

Remarque. On parle ici d'équation différentielles **ordinaires** du fait que la fonction inconnue $y(t)$ ne dépende que d'une seule variable t. Si la fonction y ne dépendait pas uniquement de t mais d'au moins deux variables, alors les dérivées apparaissant dans les équations ne seraient plus des dérivées ordinaires mais des dérivées partielles.

Dans ce cas-là, on parle d'équations différentielles partielles.

Une telle EDO se résout par ce que l'on appelle la *séparation des variables*. Cela revient à mettre tout ce qui concerne la fonction y d'un côté de l'égalité. On a alors :

$$\frac{d}{dy}\log(y(t)) = \frac{y'(t)}{y(t)} = -\frac{a_0(t)}{a_1(t)}.$$

En intégrant et puis en passant à l'exponentielle de part et d'autre, on trouve :

$$y(t) = \exp\left(-\int_{t_0}^{t} \frac{a_0(x)}{a_1(x)}\,dx\right),$$

où t_0 est une constante arbitraire déterminée par la condition initiale.

On pose la forme de la solution d'une équation homogène du premier ordre comme étant $y(t) = Ky_0(t)$. Si l'EDO n'est pas homogène (et donc $p(x)$ n'est pas la fonction constante nulle), alors la solution générale s'écrit sous la forme :

$$y(t) = K(t)y_0(t).$$

Cette méthode de résolution s'appelle *méthode de variation de la constante*. La fonction $K(t)$ doit satisfaire la propriété suivante :

$$a_0Ky_0 + a_1(K'y_0 + Ky_0') = p,$$

ce qui peut se réécrire comme :

$$K(t) = \int_{t_0}^{t} \frac{p(x)}{a_1(t)y_0(t)}\,dx.$$

Définition 4.3. On appelle D l'opérateur de différentiation : on pose $D\gamma(t) \coloneqq \gamma'(t)$, et donc $D^2\gamma(t) = D(D\gamma(t)) = D\gamma'(t) = \gamma''(t)$.

Définition 4.4. On note la forme générale d'une EDO *linéaire* d'ordre n à coefficients constants[11] comme suit :

$$\left(\sum_{k=0}^{n} a_k D^k\right) y(t) = p(t).$$

À nouveau, si $p(x)$ est la fonction constante nulle, on parle d'EDO linéaire homogène d'ordre n.

Théorème 4.5. *La solution générale d'une solution de l'EDO linéaire homogène d'ordre n est donnée par :*

$$y(t) = \sum_{k=1}^{n} \beta_k \exp(\alpha_k t),$$

où les β_k sont des constantes à déterminer à posteriori à l'aide des conditions initiales et où les α_k sont les solutions de l'équation caractéristique :

$$\pi(\alpha) = \sum_{k=0}^{n} a_k \alpha^k.$$

Si une racine α du polynôme caractéristique $\pi(\alpha)$ est de multiplicité $\mu(\alpha) \gneqq 1$, alors la forme générale de la solution est différente. En effet, les racines de multiplicité strictement supérieure à 1 se comportent de manière différente : elles apparaissent plusieurs fois avec un facteur t^k. Si γ est le nombre de racines distinctes de l'équation caractéristique, alors :

$$y(t) = \sum_{k=1}^{\gamma} \sum_{m=1}^{\mu(\alpha_k)} \beta_{k\,m} t^{m-1} \exp(\alpha_k t)$$

est la forme générale de la solution de l'EDO, où les β_{ij} sont les constantes à déterminer à posteriori à l'aide des conditions initiales.

4.1.1 EDO linéaires à coefficients constants non-homogènes

Dans le cas d'une EDO non-homogène d'ordre $n \gneqq 1$, on pourrait utiliser la méthode de variation de la constante mais cela amènerait alors un système d'équations différentielles. Une autre possibilité est de procéder à une *annihilation* du second membre (la fonction $p(t)$) afin de retrouver une équation homogène qui se résout *plus simplement*.

Pour annihiler le second membre, il faut rétablir des dérivations afin de faire disparaitre le membre. Par exemple si le second membre $p(t)$ est une fonction trigonométrique telle que $\sin(\lambda t)$, alors on sait que $D^2 p(t) = -\lambda^2 \sin(\lambda t)$. Dès lors, on peut *transformer* l'équation de départ :

$$\left(\sum_{k=0}^{n} D^k a_k\right) y(t) = p(t) = \sin(\lambda t)$$

par une autre *conservant l'ensemble des solutions*[12] donnée par :

$$(D^2 + \lambda^2)\left(\sum_{k=0}^{n} D^k a_k\right) y(t) = (D^2 + \lambda^2) p(t) = 0.$$

[11] La notion de *coefficients constants* vient du fait que les coefficients a_k sont des scalaires et ne sont pas des fonctions de t.

[12] Pour être précis, l'ensemble des solutions de la première équation est contenu dans l'ensemble des solutions de la seconde et donc une solution de la première est également une solution de la seconde. La réciproque n'est cependant pas vérifiée étant donné que le fait de rajouter une opération de dérivation rajoute des solutions.

Remarque. Dans cette section, seules les EDO du premier ordre seront analysées.

4.1.2 Importance des conditions initiales

De manière générale, ce n'est pas une unique solution qui est déterminée par la résolution d'une équation différentielle, mais bien une famille de fonctions. Pour réussir ensuite à tirer la fonction, il faut imposer des *conditions* dites *initiales*.

Ces conditions se retrouvent sous la forme $\frac{d^k}{dt^k}\big|_{t=\sigma_k} y(t) = \delta_k$. Ce sont ces conditions initiales (également appelées *problème de Cauchy*) qui permettent de donner une valeur numérique aux constantes β_j dans les formes générales des solutions.

Définition 4.6. Soit $f : I \times \mathbb{R} \to \mathbb{R} : (t, y) \mapsto f(t, y)$. On dit que f est une fonction *Lipschitzienne* (ou *de Lipschitz*) si :

$$\exists C \gneqq 0 \, t.\,q. \, \forall t \in I, y_1, y_2 \in \mathbb{R} : \left|f(t, y_1) - f(t, y_2)\right| \leqslant C|y_1 - y_2| \, .$$

C est appelé la *constante de Lipschitz*.

Théorème 4.7. *Soit* $f : I \times \mathbb{R} \to \mathbb{R} : (t, y) \mapsto f(t, y)$. *Si pour tout* $t \in I$, *la dérivée partielle de* f *selon* y *est continue, alors* f *est une fonction de Lipschitz.*

Preuve. Par le théorème de la moyenne, on sait qu'il existe $\xi \in (y_1, y_2)$ tel que :

$$\left|f(t, y_1) - f(t, y_2)\right| = \left|\tfrac{\partial}{\partial y}\big|_{y=\xi} f(t, y)\right| |y_1 - y_2| \, .$$

On trouve donc que f est Lipschitzienne de constante $C \coloneqq \max_y \left|\frac{\partial}{\partial y}\big|_{y=\xi} f(t, y)\right|$. □

Théorème 4.8. *Soit* $f(t, y)$ *une fonction Lipschitzienne et continue. Alors le problème de Cauchy suivante :*

$$\begin{cases} \frac{dy}{dt}(t) & = f(t, y) \\ y'(t_0) & = y_0 \end{cases}$$

a une solution $y(t) = y(t, y_0) \in C^1$ *pour tout* $t \in I$.

Remarque. Ce théorème signifie qu'une EDO du premier ordre représente un problème bien posé (unicité de la solution et dépendance continue).

On remarque en plus que $\left\|y(t, y_1) - y(t, y_0)\right\| \leqslant \exp(C|t - t_0|)|y_1 - y_0|$. Dès lors, pour un petit C (constante de Lipschitz), de petites variations sur la condition initiale impliquent des petites variations sur la solution finale.

4.2 Résolution d'EDO

La solution dite *analytique*[13] d'une EDO du premier ordre est donnée par :

$$y(t) = y_0 + \int_{t_0}^{t} f(\tau, y(\tau)) \, d\tau.$$

[13] C'est-à-dire la solution théorique.

Cependant, seules très peu de ces EDO non-linéaires peuvent être résolues directement. De plus, pour certaines que l'on sait résoudre, on ne sait pas exprimer de manière formelle la solution obtenue. On opte alors dans ces cas pour une résolution numérique.

Une telle résolution ne donne pas, au final, une fonction $y(t)$ à proprement parler, mais plus précisément un tableau de valeurs en un ensemble de points $\{t_0, \dots, t_n\}$ choisis à l'avance. La résolution numérique est dès lors donnée par $\{\widehat{y}(t_0), \dots, \widehat{y}(t_n)\}$.

La résolution numérique se fait alors par le théorème de Taylor (théorème 1.2). On sait alors que l'on peut exprimer une fonction $g(x)$ comme un polynôme avec comme coefficients $\frac{g^{(k)}}{k!}$. Dès lors, considérons le problème de Cauchy suivant :

$$\begin{cases} \frac{dy}{dt}(t) & = f(t, y) \\ y(t_0) & = y_0 \end{cases}.$$

On sait que $y(t_0) = y_0$, ensuite on détermine $y'(t_0)$ par $f(t, y_0)$. En dérivant encore cette fonction $f(t, y)$, on peut déterminer plusieurs dérivées successives. On a alors $\{y^{(k)}(t_0)\}_{0 \leqslant k \leqslant p}$. On pose ensuite $h \coloneqq t - t_0$, et par Taylor, on peut exprimer :

$$y(t) = \left(\sum_{k=0}^{p} \frac{y^{(k)}(t_0)}{k!} h^k \right) + \varepsilon,$$

où ε représente l'erreur qui est bornée par :

$$\varepsilon \leqslant \frac{y^{(p+1)}(\zeta)|h|^{p+1}}{(p+1)!},$$

où $\zeta \in (t_0 - l, t_0 + l)$ où $[t_0 - l, t_0 + l]$ est le voisinage de t_0 sur lequel la fonction opère. Cette fonction est donc une approximation dans un voisinage de t_0. On l'évalue donc en $t = t_i$ pour $0 \leqslant i \leqslant n$, et on obtient $\{\widehat{y}(t_i)\}$, l'ensemble des valeurs approchées. Ces valeurs peuvent ensuite, par exemple, être mises sur un graphique afin de déterminer l'allure de la solution.
Remarque. Cependant, les dérivées ne sont pas toujours simples à évaluer, ni à déterminer (surtout numériquement). De plus, le calcul de l'erreur dépend de ζ qui est un paramètre que l'on ne connait pas avec précision (on ne connait que l'intervalle dans lequel il se trouve). Cela rend l'erreur impossible à établir précisément. Également, ce résultat demande beaucoup de termes pour des grandes valeurs de h (compensations). Il est donc plus facile à appliquer pour des évaluations proches de l'origine[14].

Tout ces éléments font qu'une méthode itérative sera, de manière générale, plus facilement choisie.

4.2.1 Méthode d'Euler

Taylor donne une petite erreur lorsque h est petit. L'idée derrière la méthode d'Euler est donc de *découper* I, l'intervalle $(t_0, t_0 + T)$ d'intégration en N sous-intervalles $I_k = [t_k, t_{k+1})$ où $t_k = t_0 + kh$. On appelle $h \gtrless 0$ le *pas de discrétisation*. Pour chacun de ces sous-intervalles, la méthode d'Euler consiste à appliquer la méthode de Taylor (d'ordre 1[15]) afin de déterminer les $\widehat{y}(t_k)$. Formellement, pour tout k, on définit :

$$\begin{cases} \widehat{y}(t_k) & = \widehat{y}(t_{k-1}) + h\widehat{y}'(t_{k-1}) \\ \widehat{y}'(t_k) & = f(t_k, \widehat{y}(t_k)) \end{cases}.$$

Le principe est donc d'utiliser la pente de la fonction au début de l'intervalle I_n afin de calculer l'incrément de l'étape $n + 1$.

[14] Avoir h proche de l'origine ne veut pas obligatoirement dire que t doit également être proche de l'origine. Si h est petit (et donc proche de l'origine) c'est que t est proche de t_0 qui est la valeur dont on connait l'évaluation de y par le problème de Cauchy.
[15] Et donc se basant sur les deux premiers termes de la série de Taylor.

Remarque. Il y a 4 propriétés à vérifier concernant la méthode d'Euler :

1. consistance ;
2. zéro-stabilité ;
3. convergence ;
4. stabilité absolue.

Consistance La consistance concerne les erreurs de troncature. On note $e_k \coloneqq y(t_k) - \widehat{y}(t_k)$ l'*erreur globale* au nœud t_k de la kème étape. On note $\widehat{y^*}(t_k)$ la solution obtenue après un pas de la méthode d'Euler basée sur la solution exacte $y(t_k)$. On peut alors exprimer l'erreur globale comme :

$$e_k = y(t_k) - \widehat{y^*}(t_k) + \widehat{y^*}(t_k) - \widehat{y}(t_k).$$

En notant $\epsilon_k(h) \coloneqq y(t_k) - \widehat{y^*}(t_k)$, on peut définir l'*erreur de troncature locale* par :

$$\tau_k(h) \coloneqq \frac{\epsilon_k(h)}{h}.$$

On définit ensuite l'*erreur de troncature globale* par :

$$\tau(h) \coloneqq \max_{1 \leqslant k \leqslant \frac{T}{h}} \left|\tau_k(h)\right|,$$

où T représente la longueur de l'intervalle I d'intégration.

Définition 4.9. On dit qu'une méthode numérique est *consistante* si son erreur de troncature globale $\tau_k(h) \to 0$ pour $h \to 0$.

Remarque. Cette définition est équivalent à dire que la méthode est consistante si $\tau_k(h) = \mathcal{O}(h)$.

Définition 4.10. On dit qu'une méthode numérique est p-consistante[16] pour $p \in \mathbb{N}^*$ si pour tout $t \in I$, on a $\tau_k(h) = \mathcal{O}(h^p)$ quand $h \to 0$.

Par Taylor, on peut écrire plus formellement la solution exacte à l'étape k comme[17] :

$$y(t_k) = y(t_k) + hf(t_k, y(t_k)) + h^2 \frac{y''(\zeta)}{2}.$$

On en déduit donc l'erreur de troncature locale :

$$\tau_k(h) = \frac{y(t_k) - \widehat{y^*}(t_k)}{2} = \frac{y''(\zeta)}{2h} h^2 = \frac{y''(t\zeta)h}{2}.$$

On peut donc exprimer $\tau(h) = \mathcal{O}(h)$

Stabilité La stabilité d'une méthode à pas (telle que la méthode d'Euler) se définit par deux composantes :

- la *zéro-stabilité* ;
- et la *stabilité absolue*.

[16] Ou consistante d'ordre p.

[17] À nouveau avec $\zeta \in (t_k, t_{k+1})$ qui est le point en lequel il faut évaluer les dérivées afin de déterminer l'erreur.

En considérant la méthode à pas générale :

$$\widehat{y}(t_k) = \widehat{y}(t_{k-1}) + h\Phi(t, \widehat{y}(t_{k-1})),$$

la zéro-stabilité concerne le contrôle des erreurs de générations induites dans le calcul du pas $\Phi(t, \widehat{y}(t_k))$. Ces erreurs peuvent être évitées en prenant un pas h très petit. La stabilité absolue concerne les erreurs de génération dans $\widehat{y}(t_{k-1}) + h\Phi(t, \widehat{y}(t_{k-1}))$.

Définition 4.11. Soient $\widehat{y_1}$ et $\widehat{y}y_2$, deux solutions d'une méthode numérique données par :

$$\begin{cases} \widehat{y_1}(t_k) & = \widehat{y_1}(t_{k-1}) + h\Phi(t_{k-1}, \widehat{y_1}(t_{k-1})) \\ \widehat{y_1}(t_0) & = y_0, \end{cases}$$

et

$$\begin{cases} \widehat{y_2}(t_k) & = \widehat{y_2}(t_{k-1}) + h\left[\Phi(t_{k-1}, \widehat{y_2}(t_{k-1})) + \delta(t_k)\right] \\ \widehat{y_2}(t_0) & = y_0 + \delta(t_0). \end{cases}$$

Cette méthode est dite zéro-stable si :

$$\exists C \gneqq 0 \, t.\,q. \, \forall \epsilon \gneqq 0 : \exists h_0 \, t.\,q. \, \forall h < h_0 : \left(\forall 1 \leqslant k \leqslant \frac{T}{h} : \delta(t_k) \leqslant \epsilon\right) \Rightarrow \left|\widehat{y_1}(t) - \widehat{y_2}(t)\right| < C\epsilon.$$

Théorème 4.12. *Soit une méthode numérique générique explicite à pas donnée par :*

$$\widehat{y}(t_k) = \widehat{y}(t_{k-1}) + h\Phi(t_{k-1}, \widehat{y}(t_{k-1})).$$

Si la fonction Φ d'incrément est Lipschitzienne par rapport à sa seconde variable, et si la constante de Lipschitz C ne dépend pas de h ni des nœuds t_k, alors la méthode est zéro-stable.

Corollaire 4.13. *Il suit directement de ce théorème que la méthode d'Euler est zéro-stable.*

Convergence La consistance de la méthode permet de déterminer la propriété intrinsèque à la méthode concernant l'approximation de la solution exacte, alors que la zéro-stabilité permet de déterminer son comportement (à la méthode) par rapport aux erreurs numériques engendrées.

Définition 4.14. La méthode est dite convergente si :

$$\forall n \in \left\{0, \ldots, \frac{T}{h}\right\} : e_n(h) \coloneqq \left|y(t_n) - \widehat{y}(t_n)\right| = \mathcal{O}(h).$$

Définition 4.15. Si $e_n = \mathcal{O}(h^p)$ pour un $p \in \mathbb{N}^*$, alors on dit que la méthode est p-convergente (ou convergente d'ordre p).

Théorème 4.16 (Théorème fondamental de l'analyse numérique)**.** *Si une méthode numérique de résolution d'EDO est zéro-stable* ***et*** *consistante, alors elle est convergente.*

Preuve. Soit une méthode générique (et explicite) à pas :

$$\begin{cases} \widehat{y}(t_n) & = \widehat{y}(t_{n-1}) + h\phi(t_{n-1}, \widehat{y}(t_{n-1})) \\ \widehat{y^*}(t_n) & = y(t_{n-1} + h\phi(t_{n-1}, y(t_{n-1})). \end{cases}$$

La différence entre les deux est que $\widehat{y}$ représente une solution approchée, et la méthode itérative se base sur la solution approchée de l'étape précédente alors que $\widehat{y^*}$ représente une solution approchée calculée selon la solution théorique exacte de l'étape précédente.

$$\begin{aligned}
\widehat{y}(t_n) - \widehat{y^*}(t_n) &= \widehat{y}(t_{n-1}) + h\phi(t_{n-1}, \widehat{y}(t_{n-1})) - y(t_n) \\
&= \widehat{y}(t_{n-1}) + h\phi(t_{n-1}, \widehat{y}(t_{n-1})) - y(t_n) + \widehat{y^*}(t_n) - \widehat{y^*}(t_n) \\
&= \widehat{y}(t_{n-1}) + h\phi(t_{n-1}, \widehat{y}(t_{n-1})) - y(t_n) + \widehat{y^*}(t_n) - \big(y(t_{n-1}) + h\phi(t_{n-1}, y(t_{n-1}))\big) \\
&= \widehat{y}(t_{n-1}) - y(t_{n-1}) + h\left[\phi(t_{n-1}, \widehat{y}(t_{-1})) - \phi(t_{n-1}, y(t_{n-1})) + \frac{\widehat{y^*}(t_n) - y(t_n)}{h}\right] \\
&= \big(\widehat{y}(t_{n-1}) - y(t_{n-1})\big) + h\left(\phi(t_{n-1}, \widehat{y}(t_{n-1})) - \phi(t_{n-1}, y(t_{n-1}) + \tau_n(h)\right).
\end{aligned}$$

Par définition de méthode zéro-stable, on sait qu'il existe C tel que :

$$\left|\widehat{y}(t_n) - y(t_n)\right| \leqslant C \max_n \left|\tau_n(h)\right| = C\tau(h).$$

Si une méthode est consistante, alors pour $h \to 0$, on a $\tau(h) \to 0$. On trouve donc bien que :

$$\left|\widehat{y}(t_n) - y(t_n)\right| = \mathcal{O}(h).$$

□

Remarque. le calcul de la convergence de la méthode d'Euler fait l'hypothèse que tous les calculs sont en arithmétique exacte. On peut, pour être plus rigoureux, introduire les erreurs d'arrondi :

$$\left|\widehat{y}(t_n) - y(t_n)\right| = \widehat{y}(t_{n-1}) - y(t_{n-1}) + h\left[f(t_{n-1}, \widehat{y}(t_{n-1})) - f(t_{n-1}, y(t_{n-1})) + \tau_n(h) + \epsilon_n(h)\right] + \rho_{n-1}.$$

Pour tout n et avec $h \leqslant h_0$, en supposant $\epsilon_n \leqslant \epsilon$, et $\rho_n \leqslant \rho$ et en sachant que $\tau_n \leqslant Kh$, on trouve que :

$$\left|\widehat{y}(t_n) - y(t_n)\right| \leqslant C\left(\frac{K}{C}h + \epsilon + \frac{\rho}{h}\right).$$

On y voit donc clairement qu'en prenant h trop petit, la bore de l'erreur devient trop grande.

À cause des erreurs d'arrondi, on ne peut pas dire que l'erreur générale tend vers 0 pour $h \to 0$, maison peut cependant dire qu'il existe une valeur optimale (que l'on note ici h_{opt}) qui minimise l'erreur. Pour $h \lneqq h_{opt}$, l'erreur d'arrondi prend le dessus sur l'erreur de troncature et donc l'erreur globale augmente.

Stabilité absolue la stabilité absolue concerne le comportement de la méthode numérique pour $t_n \to +\infty$. Étudions donc le problème de Cauchy linéaire suivant :

$$\begin{cases} y'(t) &= \lambda y(t) \\ y(0) &= 1. \end{cases},$$

avec $\lambda \in \mathbb{R}_0^-$. La solution analytique est :

$$y(t) = \exp(\lambda t),$$

et donc :

$$\lim_{t \to +\infty} \left|y(t)\right| = 0.$$

Définition 4.17. Une méthode numérique pour l'approximation du problème linéaire de Cauchy est dite *absolument stable* si :

$$\lim_{t_n \to +\infty} \left|\widehat{y}(t_n)\right| = 0.$$

Définition 4.18. L'espace de stabilité absolue de la méthode numérique est donné par :

$$\mathcal{A}_\lambda \coloneqq \left\{ h\lambda \,\mathrm{t.q.}\, h \in \mathbb{R} \wedge \lim_{t_n \to +\infty} \left|\widehat{y}(t_n)\right| = 0 \right\}.$$

Dans le problème de Cauchy actuel, on a la fonction $f(t, y) = \lambda y$. En y appliquant Euler, on trouve :

$$\widehat{y}(t_n) = \widehat{y}(t_{n-1}) + hf(t_{n-1}, y(t_{n-1})) = \widehat{y}(t_{n-1})(1 + h\lambda).$$

Puisque $y(0) = 1$, on trouve (par récurrence) $y(t_n) = (1 + \lambda h)^n$. Dès lors, on peut définir que la méthode est absolument stable si et seulement si $|1 + \lambda h| \lneqq 1$, ce que l'on peut réécrire :

$$\frac{-2}{\lambda} \gneqq h \gneqq 0.$$

Si cette condition n'est pas respectée, on dit que la méthode est instable.

Définition 4.19. Une méthode numérique de résolution d'EDO est dite à *un pas* quand pour tout n, l'élément $\widehat{y}_{n+1}$ ne dépend que de $\widehat{y}_n$ et pas de $\widehat{y}_k$ pour $k \lneqq n$. Sinon, on dit que la méthode est à *pas multiples*.

Définition 4.20. Une méthode numérique de résolution d'EDO est dite *explicite* si pour tout n, la valeur $\widehat{y}_n$ peut se calculer directement avec les valeurs $\widehat{y}_k$ pour $k \lneqq n$. Sinon, on dit que la méthode est *implicite*.

Remarque. Les méthode d'Euler dites respectivement *retrograde* et *du trapèze* sont absolument stable pour tout h et sont respectivement d'ordre 1 et 2.

4.3 Runge-Kutta

Les méthodes de Runge-Kutta sont des méthodes itératives à un pas qui augmentent leur précision en augmentant le nombre d'évaluations de la fonction f par pas. L'idée générale derrière un R-K d'ordre p est d'avoir les p premiers termes de la série de Taylor équivalents entre la solution exacte $y(t_n)$ et la solution approchée $\widehat{y}(t_n)$). En particulier, un R-K d'ordre 1 est la méthode d'Euler.

Définition 4.21. La forme générale d'un R-K est la suivante :

$$\widehat{y}(t_n) = \widehat{y}(t_{n-1}) + hF\left(t_{n-1}, \widehat{y}(t_{-ni1}), h, f\right),$$

où F est définie par :

$$F\left(t_n, \widehat{y}(t_n), h, f\right) \coloneqq \sum_{k=1}^{p} b_k K_k,$$

avec :

$$K_i \coloneqq f\left(t_n + c_i h, \widehat{y}(t_n) + h \sum_{j=1}^{p} (a_{ij} K_j)\right).$$

Les valeurs a_{ij}, b_i et c_i sont les paramètres de la méthode. On suppose que la condition suivante est remplie :

$$\forall 1 \leqslant i \leqslant s : c_i = \sum_{k=1}^{p} a_{ij}.$$

Remarque. Un R-K est explicite si et seulement si $\forall j \geqslant i : a_{ij} = 0$.

La méthode Runge-Kutta d'ordre p se note RKp ou R-Kp.

4.3.1 Runge-Kutta explicite d'ordre 2

Par la remarque précédente, on sait que pour que la méthode soit explicite, il faut que $a_{11} = a_{12} = a_{22} = 0$.
On détermine alors les valeurs c_i par :

$$\begin{cases} c_1 &= a_{11} + a_{12} = 0, \\ c_2 &= a_{21} + a_{22} = a_{21}. \end{cases}$$

La forme explicite de la méthode itérative est alors :

$$\widehat{y}(t_n) = \widehat{y}(t_{n-1}) + h\left(b_1 K_1 + b_2 K_2\right),$$

où les K_j valent[18] :

$$K_1 = f\left(t_{n-1} + c_1 h, \widehat{y}(t_{n-1}) + h a_{11} K_1 + h a_{12} K_2\right) = f\left(t_{n-1}, \widehat{y}(t_{n-1})\right),$$
$$K_2 = f\left(t_{n-1} + c_2 h, \widehat{y}(t_{n-1}) + h a_{21} K_1 + h a_{22} K_2\right) = f\left(t_{n-1} + c_2 h, \widehat{y}(t_{n-1}) + h a_{21} K_1\right) = f\left(t_{n-1} + c_2 h, \widehat{y}(t_{n-1}) + h c_2 K_1\right).$$

Le développement de Taylor d'une fonction à deux variables est donné par :

$$f(x_0 + \delta x_0, y_0 + \delta y_0) \simeq f(x_0, y_0) + \frac{\partial f}{\partial x}(x_0, y_0)\delta x_0 + \frac{\partial f}{\partial y}(x_0, y_0)\delta y_0.$$

On peut dès lors appliquer Taylor sur K_2, ce qui donne :

$$\begin{aligned} K_2 &= f(t_{n-1} + c_2 h, y(t_{n-1}) + c_2 h K_1) = f(t_{n-1}, y(t_{n-1})) + c_2 h \tfrac{\partial f}{\partial t}(t_{n-1}, y(t_{n-1})) + c_2 h K_1 \tfrac{\partial f}{\partial y}(t_{n-1}, y(t_{n-1})) + \mathcal{O}(h^2) \\ &= K_1 + c_2 h \left(\tfrac{\partial f}{\partial t}(t_{-1n}, y(t_{n-1})) + K_1 \tfrac{\partial f}{\partial y}(t_{n-1}, y(t_{n-1}))\right) + \mathcal{O}(h^2). \end{aligned}$$

En réinsufflant cette donnée de K_2 dans le calcul de $\widehat{y}(t_n)$, on trouve :

$$\begin{aligned} \widehat{y}(t_n) = y(t_{n-1}) &\quad + (b_1 K_1 + b_2 K_2) = y(t_{n-1}) + h b_1 f(t_{n-1}, y(t_{n-1})) + h b_2 (K_2) \\ = y(t_{n-1}) &\quad + h b_1 f(t_{n-1}, y(t_{n-1})) + h b_2 f(t_{n-1}, y(t_{n-1})) + h^2 b_2 c_2 \tfrac{\partial f}{\partial t}(t_{n-1}, y(t_{n-1})) \\ &\quad + h^2 b_2 c_2 f(t_{n-1}, y(t_{n-1})) \tfrac{\partial f}{\partial y}(t_{n-1}, y(t_{n-1})) + \mathcal{O}(h^3). \end{aligned}$$

En considérant que $y'(t_n) = f(t_{n-1}, y(t_{n-1}))$, on trouve[19] :

$$\begin{aligned} y''(t_n) = \left(y'\right)'(t_n) = \left(f'\right)(t_{n-1}, y(t_{n-1})) &= \tfrac{\partial f}{\partial t}(t_{n-1}, y(t_{n-1})) + \tfrac{\partial f}{\partial y}(t_{n-1}, y(t_{n-1})) \tfrac{dy}{dt}(t_{n-1}) \\ &= \tfrac{\partial f}{\partial t}(t_{n-1}, y(t_{n-1})) + \tfrac{\partial f}{\partial y}(t_{n-1}, y(t_{n-1})) f(t_{n-1}, y(t_{n-1})) \end{aligned}$$

On peut également effectuer le même développement de Taylor sur la solution exacte, ce qui donne :

$$\begin{aligned} y(t_n) &= y(t_{n-1}) + h \tfrac{dy}{dt}(t_{n-1}) + \frac{h^2}{2} \tfrac{d^2 y}{dt^2}(tn - 1) + \mathcal{O}(h^3) \\ &= y(t_{n-1}) + h f(t_{n-1}, y(t_{n-1})) + \frac{h^2}{2}\left(\tfrac{\partial f}{\partial t}(t_{n-1}, y(t_{n-1})) + f(t_{n-1}, y(t_{n-1}) \tfrac{\partial f}{\partial y}(t_{n-1}, y(t_{n-1}))\right) + \mathcal{O}(h^3). \end{aligned}$$

[18] On suppose que l'étape $n-1$ a été déterminée avec précision infinie et donc que $\widehat{y}(t_{n-1}) = y(t_{n-1})$.
[19] En utilisant la formule de dérivation d'une composition.

En identifiant respectivement les termes de $\widehat{y}(t_n)$ et de $y(t_n)$, on peut déterminer les égalités suivantes :

$$\begin{cases} b_1 + b_2 & = 1, \\ c_2 b_2 & = \frac{1}{2}. \end{cases}$$

Il y a trois paramètres à déterminer et uniquement deux équations, ce qui nous laisse choisir un des paramètres arbitrairement pour réduire le degré de liberté. Par exemple, en posant $b_1 = 0$, on trouve directement $b_2 = 1$ et donc $c_2 = \frac{1}{2}$. La méthode peut donc s'écrire :

$$\widehat{y}(t_n) = \widehat{y}(t_{n-1}) + hK_2,$$

étant donné que :

$$K_2 \coloneqq f\left(t_{n-1} + c_2 h, \widehat{y}(t_{n-1}) + h c_2 K_1\right) = f\left(t_{n-1} + \frac{h}{2}, \widehat{y}(t_{n-1}) + \frac{h}{2} f(t_{n-1}, y(t_{n-1}))\right).$$

Une interprétation de cette méthode est qu'un *demi-pas* est d'abord effectué, et puis que la dérivée est *instanciée* en ce milieu de pas. C'est cette dérivée-là qui est utilisée à la place de la dérivée en t_{n-1}.

Théorème 4.22. *La stabilité absolue d'un Runge-Kutta d'ordre* p *est présente si et seulement si* $\left|R(h\lambda)\right| \lneqq 1$, *où on définit :*

$$R(h\lambda) \coloneqq \sum_{k=0}^{p} (h\lambda)^k \frac{1}{k!}.$$

5 Interpolation

Introduction

Lorsqu'une fonction a une forme analytique compliquée à évaluer, différentier ou encore intégrer, il est confortable de pouvoir l'approcher par une fonction beaucoup plus simple à analyser, tel qu'un polynôme. Pour ce faire, on évalue la fonction analytique en un certain nombre de points, que l'on appelle les *nœuds d'interpolation*.

Soit f une fonction dont on connait qu'un nombre fini d'évaluations. Deux cas se distinguent :

1. si les données sont supposées exactes, on parle de situation *déterministe* et on résout cela par interpolation ;
2. si les données sont perturbées, on parle de situation *stochastique*[20] et on résout cela par lissage.

Dans cette section, seule l'interpolation est discutée, le lissage concerne la section suivante.

L'idée est d'avoir un ensemble de $n+1$ couples (x_i, y_i) tels que $f(x_i) = y_i$ pour tout i, et on recherche une autre fonction ϕ telle que $\phi(x_i) = y_i$ afin d'obtenir une approximation de f. Les exemples d'interpolation les plus connus sont les suivants :

- interpolation *polynômiale* (ϕ est un polynôme) ;
- interpolation polynômiale *par morceaux* (également appelée *interpolation par splines*) ;
- approximation trigonométrique (pas discutée ici).

5.1 Interpolation polynômiale

Théorème 5.1. *Soient $x_0, \dots, x_n$, $n+1$ points distincts, et soient $y_0, \dots, y_n$ leur valeur associée. Alors il existe une unique polynôme π_n tel que :*

$$\forall 0 \leqslant i \leqslant n : \pi_n(x_i) = y_i.$$

Remarque. La preuve de ce théorème (omise ici) ne fournit pas un algorithme afin de déterminer les coefficients du polynôme. Il y a donc deux moyens de déterminer le polynôme avec précision :

- utiliser une méthode générale[21] qui consiste à créer un système linéaire en remplaçant les valeurs de x_i et y_i dans l'équation du polynôme. Cependant, cette méthode implique une résolution de système, ce qui est assez coûteux ($\mathcal{O}(n^3)$ comme vu dans la section 3) et peut, en plus, donner un système mal conditionné.
- Utiliser une méthode *ad hoc*. Les méthodes de Lagrange et de Newton présentent l'avantage d'être en complexité $\mathcal{O}(n^2)$.

5.1.1 Polynôme de Lagrange

Définition 5.2. On définit le ième polynôme de Lagrange par :

$$l_i(x) \coloneqq \prod_{j \neq i} \frac{x - x_j}{x_i - x_j},$$

[20] Terme relativement équivalent à *probabiliste*.
[21] Que l'on peut également qualifier de *naïve*.

avec $0 \leqslant i \leqslant n$.

Théorème 5.3. *Soient $\{(x_i, y_i\}_{0\leqslant i\leqslant n}$ des couples de données. Le polynôme de Lagrange est donné par :*

$$\pi_n(x) \coloneqq \sum_{i=0}^{n} y_i l_i(x).$$

On remarque que les polynômes de Lagrange ont une propriété très agréable qui est que :

$$l_i(x_j) = \delta_{ij}.$$

En effet, si $i = j$, alors tous les facteurs du produit seront 1, et le polynôme s'évalue donc par 1 ; et si $i \neq j$, alors un des facteurs du produit donnera 0, ce qui absorbera tout le résultat et donc évaluera le polynôme par 0. De plus, chaque polynôme de Lagrange est de degré n. C'est assez clair du fait que ces polynômes sont définis par un produit où le dénominateur est constant (pour chaque facteur) et où le numérateur est de degré 1 en x.

Définition 5.4. On définit le polynôme nodal de degré $n+1$ par :

$$\omega_{n+1} \coloneqq \prod_{i=0}^{n}(x - x_i).$$

Lemme 5.5. *Si ω_{n+1} est le polynôme nodal de degré $n+1$ lié à un ensemble $\{(x_i, y_i)\}$ de points, alors :*

$$\forall 0 \leqslant i \leqslant n : \omega'_{n+1}(x_i) = \prod_{j\neq i}(x_i - x_j).$$

On remarque à nouveau que le polynôme nodal de degré $n+1$ s'annule en tout x_i.

Théorème 5.6. *Le polynôme π_n peut s'écrire de manière équivalente comme :*

$$\pi_n(x) = \sum_{i=0}^{n} \frac{\omega_{n+1}(x)}{(x - x_i)\omega_{n+1'}(x_i)} y_i.$$

Preuve. Il suffit de montrer que le polynôme de Lagrange l_i est équivalent à :

$$\frac{\omega_{n+1}(x)}{(x - x_i)\omega'_{n+1}(x_i)}.$$

On trouve donc :

$$l_i(x) = \prod_{j\neq i} \frac{x - x_j}{x_i - x_j} = \left(\prod_{j\neq i}(x - x_j)\right)\left(\prod_{j\neq i} \frac{1}{x_i - x_j}\right) = \frac{\omega_{n+1}(x)}{(x - x_i)} \frac{1}{\omega'_{n+1}(x_i)}.$$

□

Théorème 5.7. *Soient $x_0, \ldots, x_n$, $n+1$ points distincts et soit $x \in \operatorname{dom} f$. On suppose f de classe C^{n+1} sur I_x, le plus petit ensemble continue contenant tous les x_i[22]. Alors l'erreur d'interpolation au point x est donnée par :*

$$E_n(x) = f(x) - \pi_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x),$$

où ω_{n+1} est le polynôme nodal de degré $n+1$ et où $\xi \in I_x$.

[22]On peut dès lors exprimer I_x comme étant l'intervalle $[m_x, M_x]$ où $m_x \coloneqq \min_i x_i$ et $M_x \coloneqq \max_i x_i$ en posant $x_{n+1} \coloneqq x$.

Remarque. Ce théorème est assez proche du théorème de Taylor : il permet de borner l'erreur d'une approximation (ici polynômiale).

Définition 5.8. Soit f une fonction continue sur $[a, b]$. On définit sa norme infinie par :

$$\|f\|_\infty \coloneqq \max_{x\in[a,b]} \left|f(x)\right|.$$

La norme infinie d'une fonction est donc la plus grande valeur (en valeur absolue) que peut prendre la fonction sur son domaine de définition. Pour tout $x \in [a, b]$, on peut alors affirmer :

$$f(x) \leqslant \|f\|_\infty,$$

par définition. On peut donc revenir au théorème et borner l'erreur par :

$$E_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(x) \leqslant \frac{\left\|f^{(n+1)}\right\|_\infty}{(n+1)!}\|\omega_{n+1}\|_\infty = \frac{\left\|f^{(n+1)}\right\|_\infty (b-a)^{n+1}}{(n+1)!}.$$

On peut remarquer un certain nombre de propriétés sur la méthode d'interpolation de Lagrange. Premièrement, on peut borner l'erreur d'interpolation, mais il faut pour cela connaitre la fonction f, ce qui est loin d'être systématique. On peut également observer que l'erreur d'interpolation est faible près des valeurs x_i, et l'est d'autant plus que la fonction f est lisse.

Une propriété des polynômes nodaux (et donc par extension applicable à Lagrange) est le fait que le maximum de $x \mapsto \left|\omega_{n+1}(x)\right|$ est toujours atteint dans un des deux intervalles extrêmes (à savoir $[x_0, x_1]$ ou $[x_{n-1}, x_n]$). Instinctivement, c'est lié au fait que dans ces intervalles, ce sont les plus grands exposants qui prennent le dessus sur les autres (ils deviennent dominant, et on qualifie les plus petits exposants de *négligeables*), ce qui implique qu'on a une valeur qui sera plus grande dans es intervalles (en valeur absolue tout du moins).

Par définition, si la fonction f est un polynôme de degré n, alors l'erreur est nulle (ce qui est logique car $n+1$ points permettent de caractériser un unique polynôme de degré n, comme énoncé par le théorème 5.1).

On peut également remarquer (ce qui est lié avec la remarque localisant le maximum de la fonction $x \mapsto \left|\omega_{n+1}(x)\right|$) que l'erreur d'extrapolation est typiquement supérieure à l'erreur d'interpolation. L'erreur d'extrapolation étant définie comme l'erreur d'évaluation d'un point x n'appartenant **pas** à l'intervalle I_x, contrairement à l'erreur d'interpolation. En effet, un polynôme de degré n (typiquement π_n) va osciller entre sa première et sa dernière racine. Après cela, le polynôme se dirige (avec une croissance définie par le degré) vers les infinis, et donc le polynôme d'interpolation π_n n'est pas bon à utiliser pour approcher f en dehors de l'ensemble de définition.