UNIVERSITÉ CÔTE D'AZUR

Intégration projective d'équations différentielles raides

SINADINOVIC Marko et **BENHARRATS Nadir**

Mémoire sous la direction du **Dr. Pr. REY Thomas**

Table des matières

Préface

Ce mémoire représente une étape dans notre parcours académique et personnel. En explorant en profondeur les équations différentielles, les schémas numériques ainsi que l'intégration projective appliquée aux systèmes raides, on a pu confronter la rigueur théorique aux défis de l'implémentation numérique. Ce travail nous a permis d'acquérir un solide bagage en analyse numérique, tout en découvrant de nouvelles approches innovantes qui nous ont ouvert les portes d'une compréhension plus fine et nuancée des phénomènes étudiés.

Sur le plan pédagogique, ce mémoire nous a offert l'opportunité de consolider nos acquis théoriques étudié en cours notamment en équation différentielles, analyse numérique et en analyse tout en apprenant à développer et optimiser des algorithmes pour la résolution de problèmes raides. L'élaboration de ce travail nous a permis de passer d'une approche purement théorique à une application concrète, établir le lien entre les propositions mathématiques et leur mise en œuvre computationnelle, et ainsi d'acquérir une méthodologie de recherche qui nous servira tout au long de notre vie. En effet, ce travail de recherche nous a permis de développer d'acquérir certaine discipline, esprit critique sur nos erreurs et travaux mais aussi ce projet a confirmé notre envie de poursuivre les mathématiques dans le domaine de l'analyse numérique.

D'un point de vue intellectuel, ce mémoire fut un véritable défi. Nous avons dû jongler avec les cours, des recherches personnelles et un projet parallèle exigeant. Nous avons rencontré de nombreux obstacles, notamment des échecs répétés dans l'implémentation de notre code et des erreurs de calcul fréquentes qui nous ont contraints à revoir et à améliorer sans cesse nos méthodes. Par ailleurs, il a été difficile de condenser, introduire et d'expliquer de manière concise des notions et théorèmes complexes, nous avons essayé de notre mieux de présenter toutes les notions utiles liées aux équations différentielles et à l'analyse numérique. Pourtant, ces épreuves ont largement contribué à notre progression en mathématique et en programmation, mais aussi à notre progression personnelle, nous avons acquis des compétences de vulgarisation et des compétences philanthropes.

Je tiens à exprimer ma profonde reconnaissance envers toutes les personnes qui ont contribué à la réalisation de ce mémoire. Mes remerciements s'adressent tout particulièrement au Dr Pr. REY Thomas, ses conseils, ses explications, sa rigueur scientifique ainsi que sa disponibilité nous ont permis de surmonter les difficultés rencontrées durant la réalisation de ce mémoire. Je remercie également l'ensemble de nos professeurs, Dr. BERTHELIN Florent, Dr. SANGAM Afeintou, Dr. SCHEID Claire, pour leurs enseignements et leurs disponibilités. Nous tenons enfin à exprimer toute notre reconnaissance à nos familles et à nos proches, dont le soutien et les encouragements ont été une véritable source de motivation et de réconfort tout au long de ces années.

Ce mémoire de recherche est le reflet d'un parcours riche en découvertes, en remises en question et en apprentissages. Nous espérons qu'il saura transmettre la passion qui nous anime pour l'analyse numérique, et qu'il offrira aux lecteurs une contribution utile et inspirante dans le domaine des équations différentielles et de l'analyse numérique.

Chapitre 1

Rappels théoriques des équations différentielles

Les équations différentielles jouent un rôle fondamental dans de nombreux domaines scientifiques et techniques. Elles permettent de modéliser des phénomènes dynamiques où les variations d'une quantité dépendent de la quantité elle-même et de ses dérivées. Ce chapitre présente un rappel des concepts théoriques essentiels des équations différentielles, en se concentrant sur les équations linéaires et non linéaires, les méthodes de résolution explicites ainsi que les différents aspects théoriques liés à ces concepts.

1.1 Introduction au concept d'équation différentielle

1.1.1 Les équations différentielles

Afin d'introduire le cœur du sujet, nous devons expliciter les notions clés utilisées. Posons Ω un ouvert de $\mathbb{R} \times \mathbb{K}^n$ avec $\mathbb{K}$ un corps, $\mathbb{R}$ ou $\mathbb{C}$, $N \in \mathbb{N}^*$, et une fonction f définie sur Ω à valeurs dans $\mathbb{K}$. Une équation différentielle est une équation qui relie une fonction inconnue à ses dérivées. Elle peut être d'ordre 1, 2 ou supérieur, selon le nombre de dérivées impliquées.

Définition 1. Équation différentielle
Soient $\mathbb{O}$ un ouvert de $\mathbb{R} \times \mathbb{K}$, et une fonction $f : \mathbb{O} \to \mathbb{K}$. Alors l'équation suivante

$$\frac{dy}{dt} = f(t, y) \quad \textit{avec } (t, y) \in \mathbb{O}, t \in \mathbb{R} \textit{ et } y \in \mathbb{K} \tag{1.1}$$

est dite équation différentielle en y (la variable d'état) relativement à la variable réelle, dite variable de temps.

Remarque 1. *Lorsque le membre de gauche représente la dérivée d'ordre 1 de la fonction* $\mathbf{y}$*, on dit que l'équation différentielle est d'ordre 1. Si le membre de gauche représente la dérivée d'ordre 2 de la fonction* $\mathbf{y}$*, on dit que l'équation différentielle est d'ordre 2.*

De cette façon nous pouvons définir ce qu'est une équation différentielle d'odre n, avec $n \in \mathbb{N}^*$.

Définition 2. Équation différentielle d'odre n
Soient $N \in \mathbb{N}^*$, $\mathbb{O}$ un ouvert de $\mathbb{R} \times \mathbb{K}^N$, et une fonction $f : \mathbb{O} \to \mathbb{K}$. Alors l'équation suivante

$$\frac{dy^n}{dt^n} = f(t, y, \frac{dy}{dt}, \frac{dy^2}{dt^2}, ..., \frac{dy^{(n-1)}}{dt^{(n-1)}}) \quad \textit{avec } (t, y, \frac{dy}{dt}, \frac{dy^2}{dt^2}, ..., \frac{dy^{(n-1)}}{dt^{(n-1)}}) \in \mathbb{O}, t \in \mathbb{R} \textit{ et } y \in \mathbb{K}^N \tag{1.2}$$

est dite équation différentielle d'ordre n dans $\mathbb{K}^N$ en y, la variable d'état, relativement à la variable réelle t dites variable de temps.

1.1.2 Solutions

Le but est de trouver une solution à ces équations différentielles. Revenons à l'ordre 1, une solution est une fonction qui satisfait l'équation pour toutes les valeurs de la variable indépendante t. Prenons l'exemple le plus populaire pour illustrer nos propos : si nous avons l'équation différentielle

$$\frac{dy}{dt} = y$$

une solution possible est $y = e^t$, car la dérivée de e^t par rapport à t est e^t, ce qui satisfait l'équation sur l'ouvert $\mathbb{R} \times \mathbb{R}$. Ainsi une solution d'une équation différentielle d'ordre 1 est une fonction dérivable y définie sur un intervalle, non réduit à un point, telle que :

$$f(t, y, \frac{dy}{dt}) = 0$$

En généralisant à l'odre n nous obtenons la définition suivante.

Définition 3. Solution d'une équation différentielle d'ordre n
Soient $I \subset \mathbb{R}$ un intervalle de $\mathbb{R}$ et une fonction $y \in \mathrm{C}^n$ définie sur I et à valeur dans $\mathbb{K}^N$ telle que
a)

$$\forall t \in I, (t, y, \frac{dy}{dt}, \frac{dy^2}{dt^2}, ..., \frac{dy^{(n-1)}}{dt^{(n-1)}}) \in \mathbb{O}$$

b)

$$\forall t \in I, \frac{dy^n}{dt^n} = f(t, y, \frac{dy}{dt}, \frac{dy^2}{dt^2}, ..., \frac{dy^{(n-1)}}{dt^{(n-1)}})$$

Nous appellons solution de l'équation différentielle d'ordre n le couple (I, y).

Nous pouvons rajouter une précision sur la régularité des solutions des équations différentielles.

Proposition 1. Régularité des solutions
Soit $\mathbb{O}$ un ouvert de $\mathbb{R} \times \mathbb{K}^N$. Si $f : \mathbb{O} \to \mathbb{K}^N$ est de classe C^k avec $k \in \mathbb{N}$, alors toute solution de l'équation différentielle

$$\frac{dY}{dt} = f(t, Y), \quad (t, Y) \in \mathbb{O}, \quad t \in \mathbb{R}, \quad Y \in \mathbb{K}^N. \tag{1.3}$$

est de classe C^{k+1}.

D'ailleurs, nous pouvons donner des conditions pour la résolution d'une équation différentielle. En effet, pour résoudre une équation différentielle, il est souvent nécessaire de connaître certaines conditions, dites conditions initiales. Le problème de Cauchy est un exemple classique où l'on cherche à trouver une solution à une équation différentielle en connaissant la valeur de la solution à un instant donné ; de ce fait, nous pouvons utiliser cette information pour déterminer la solution sur un intervalle de temps. Cette condition initiale est cruciale pour garantir que la solution est unique et bien définie.

Définition 4. Problème de Cauchy
Soit $(t_0, y_0) \in \mathbb{O}$. Le problème de Cauchy avec donnée initiale (t_0, y_0) consiste à déterminer la ou les solutions y de l'équation différentielle sur un intervalle I tel que

$$\begin{cases} t_0 \in I \\ y(t_0) = y_0 \end{cases}$$

La condition $y(t_0) = y_0$ est appelée condition initiale ou condition de Cauchy.

Historiquement, le problème de Cauchy a été formulé pour étudier les solutions des équations différentielles ordinaires. Néanmoins, pour trouver les solutions à ces équations différentielles ordinaires, Cauchy a introduit la formulation intégrale. Celle-ci permet de reformuler le problème en termes d'intégrales. De ce fait, on peut utiliser des outils d'analyse, d'algèbre et de calcul numérique pour trouver la solution et étudier le comportement des solutions, notamment pour prouver leur existence et l'unicité de la solution sur un intervalle.

Proposition 2. Formulation intégrale du problème de Cauchy
Soient $\mathbb{O}$ un ouvert de $\mathbb{R} \times \mathbb{K}^N$ et $f : \mathbb{O} \to \mathbb{K}^N$, $f \in \mathrm{C}^0$. Soit $(t_0, y_0) \in \mathbb{O}$. Une fonction $y : I \to \mathbb{K}^N$ est une solution de

$$\frac{dY}{dt} = f(t, Y), \quad (t, Y) \in \mathbb{O}, \quad t \in \mathbb{R}, \quad Y \in \mathbb{K}^N. \tag{1.4}$$

telle que $y(t_0) = y_0$ si et seulement si

a) y est continue,
b) $\forall \mathrm{t} \in I$, $(t, y(t)) \in \mathbb{O}$,

c) $\forall t \in I$,

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

Cette formulation est particulièrement utile pour les méthodes numériques, tant pour l'approximation d'intégrales que pour les méthodes d'Euler explicite ou de Runge-Kutta, qui sont utilisées pour approximer les solutions des équations différentielles.

1.1.3 Prolongements de solution

À présent, nous connaissons certaines définitions et propositions sur les équations différentielles, particulièrement sur les solutions, mais nous pouvons apporter quelques notions cruciales pour la suite. Parlons des différents types de prolongements des solutions.

Considérons l'équation différentielle $\frac{dy}{dt} = y$ avec la solution générale $y(t) = Ce^t$. $(\mathcal{S})$
- Soit $y_1(t) = e^t$ définie sur l'intervalle $I_1 = [0, 1]$.
- Soit $y_2(t) = e^t$ définie sur l'intervalle $I_2 = [0, 2]$.

Introduisons le concept de prolongement d'une solution d'une équation différentielle.

Définition 5. Prolongement
Soient $y_1 : I_1 \to \mathbb{K}^N$ et $y_2 : I_2 \to \mathbb{K}^N$ deux solutions de (E). On dit que y_2 est un prolongement de y_1 si $I_1 \subset I_2$ et si la restriction de y_2 à I_1 est y_1.

Définition 6. Prolongement strict
Soient $y_1 : I_1 \to \mathbb{K}^N$ et $y_2 : I_2 \to \mathbb{K}^N$ deux solutions de (E). On dit que y_2 est un prolongement strict de y_1 si y_2 est un prolongement de y_1 avec $I_1 \neq I_2$. c'est-à-dire pas d'autres prolongements qu'elle-même

À l'exemple de $(\mathcal{S})$, y_2 est un prolongement de y_1 car $I_1 \subset I_2$ et la restriction de y_2 à I_1 est y_1. De plus, y_2 est un prolongement strict de y_1 car $I_1 \neq I_2$. Ces prolongements de solutions ont des appellations spécifiques.

Définition 7. Solution maximale
Une solution est dite maximale si elle n'admet pas de prolongements stricts.

Remarque 2. *En reprenant notre exemple $(\mathcal{S})$, la solution y_1 n'est pas une solution maximale de $\frac{dy}{dt} = y$ car elle est strictement prolongeable par y_2.*

En vue de qualifier une solution sur tout un intervalle I, la notion de solution globale existe.

Définition 8. Solution globale
Dans le cas où $\mathbb{O} = I \times \mathbb{O}'$ où I est un intervalle de $\mathbb{R}$ et $\mathbb{O}'$ un ouvert de $\mathbb{K}^N$, une solution globale est une solution définie sur I tout entier.

Remarque 3. *Reconsidérons $(\mathcal{S})$, et définissons notre exemple sur l'intervalle $I = \mathbb{R}$, cette solution est une solution globale car elle est définie sur tout l'intervalle I.*

Avant de se précipiter dans la résolution de différentes équations différentielles et dans l'analyse des solutions de ces dernières, nous nous devons de rappeler plusieurs lemmes utiles. Ce sont les différents lemmes de Gronwall.

1.1.4 Les lemmes de Gronwall

Lemme 1. Lemme de Gronwall différentielle
Soit I un intervalle de $\mathbb{R}$ et un point $t_0 \in \mathbb{R}$. Si nous avons l'inégalité suivante : $\forall\ t \in I$,

$$\frac{dw}{dt} \leq v(t)w(t)$$

avec $v \in C^1(I, \mathbb{R})$ et $v \in C^0(I, \mathbb{R})$, *alors* $\forall\ t \in I$:

$$w(t) \leq w(t_0) \exp\left(\int_{t_0}^{t} v(s)\, ds)\right)$$

Pour prouver ce lemme, il suffit de jouer avec le terme :

$$\beta(t) = \exp\left(-\int_{t_0}^{t} v(s)\,ds\right) w(t)$$

En calculant la dérivée de $\beta(t)$ par rapport à t :

$$\frac{d\beta}{dt} = \exp\left(-\int_{t_0}^{t} v(s)\,ds\right)\left(\frac{dw}{dt} - v(t)w(t)\right)$$

Or nous avons :

$$\frac{dw}{dt} \leq v(t)w(t)$$

Donc,

$$\frac{dw}{dt} - v(t)w(t) \leq 0$$

Ainsi,

$$\frac{d\beta}{dt} \leq \exp\left(-\int_{t_0}^{t} v(s)\,ds\right) 0 = 0$$

Cela signifie que β est décroissante. En évaluant β en $t = t_0$:

$$\beta(t_0) = \exp\left(-\int_{t_0}^{t_0} v(s)\,ds\right) w(t_0) = w(t_0)$$

Puisque β est décroissante et que $\beta(t_0) = w(t_0)$, alors pour tout $t \in I$:

$$\beta(t) \leq w(t_0)$$

En revenant à la définition de $\beta(t)$, nous obtenons :

$$\exp\left(-\int_{t_0}^{t} v(s)\,ds\right) w(t) \leq w(t_0)$$

Finalement,

$$w(t) \leq w(t_0) \exp\left(\int_{t_0}^{t} v(s)\,ds\right)$$

Un autre lemme de Gronwall que nous allons utiliser lorsque nous parlerons de consistance et de stabilité est le lemme de Gronwall discret. Ce dernier nous servira dans l'étude de l'accumulation des erreurs locales et de la convergence des méthodes d'intégration numérique.

Lemme 2. Lemme de Gronwall discret
Soit $(a_n)_{n\geq 0}$ une suite de réels positifs satisfaisant, pour tout $n \geq 0$,

$$a_n \leq C + \sum_{k=0}^{n-1} \gamma_k\, a_k,$$

où $C \geq 0$ et $(\gamma_k)_{k\geq 0}$ est une suite de réels non négatifs. Alors, pour tout $n \geq 0$, on a

$$a_n \leq C\, \exp\left(\sum_{k=0}^{n-1} \gamma_k\right).$$

La démonstration de ce lemme est triviale, nous procédons par récurrence.

— **Initialisation** : Pour $n = 0$ nous avons : $a_0 \leq$ C. Comme la somme est nul, nous obtenons $\exp(0) = 1$ et l'inégalité s'écrit : $a_0 \leq C\ \exp(0)$, ce qui est vrai.

— **Hypothèse de récurrence** : Supposons que pour tout k tel que $0 \leq k < n$, on ait

$$a_k \leq C\, \exp\left(\sum_{j=0}^{k-1} \gamma_j\right).$$

— **Héridité** : Pour $n \geq 1$, d'après l'hypothèse, on a

$$a_n \leq C + \sum_{k=0}^{n-1} \gamma_k \, a_k \leq C + \sum_{k=0}^{n-1} \gamma_k \left(C \, \exp\left(\sum_{j=0}^{k-1} \gamma_j \right) \right).$$

En factorisant C :

$$a_n \leq C \left[1 + \sum_{k=0}^{n-1} \gamma_k \, \exp\left(\sum_{j=0}^{k-1} \gamma_j \right) \right].$$

Or, on peut télescoper la somme $\forall k \geq 0$:

$$\exp\left(\sum_{j=0}^{k} \gamma_j \right) - \exp\left(\sum_{j=0}^{k-1} \gamma_j \right) = \exp\left(\sum_{j=0}^{k-1} \gamma_j \right) \Big(\exp(\gamma_k) - 1 \Big).$$

Puisque pour tout $\gamma_k \geq 0$ on a $\exp(\gamma_k) - 1 \geq \gamma_k$:

$$\gamma_k \, \exp\left(\sum_{j=0}^{k-1} \gamma_j \right) \leq \exp\left(\sum_{j=0}^{k} \gamma_j \right) - \exp\left(\sum_{j=0}^{k-1} \gamma_j \right).$$

En sommant cette inégalité de $k = 0$ à $n - 1$, on obtient :

$$\sum_{k=0}^{n-1} \gamma_k \, \exp\left(\sum_{j=0}^{k-1} \gamma_j \right) \leq \exp\left(\sum_{j=0}^{n-1} \gamma_j \right) - 1.$$

D'où :

$$a_n \leq C \left[1 + \exp\left(\sum_{j=0}^{n-1} \gamma_j \right) - 1 \right] = C \, \exp\left(\sum_{j=0}^{n-1} \gamma_j \right).$$

Étant donné nos connaissances actuelles sur les équations différentielles ordinaires et certaines inégalités, nous pouvons dès à présent explorer les divers théorèmes, méthodes et propositions applicables à la résolution d'équations différentielles linéaires.

1.2 Exploration des différentes équations différentielles

1.2.1 Équations différentielles linéaires

Qu'entend-on par équation différentielle linéaire ?

Définition 9. Équation différentielle linéaire
On dit que l'équation $\frac{dy}{dt} = f(t, y)$ est une équation différentielle linéaire si $f(t, y) = A(t)y + B(t)$ avec

$$\begin{cases} A \in \mathbb{M}_N(\mathbb{K}) \\ B \in \mathbb{K}^N \end{cases}$$

Remarque 4. *Nous appelons la fonction $B \in \mathbb{K}^N$ le second membre, mais dans certains cas, il se peut que la fonction $B = 0$. Alors, nous appelons $\frac{dy}{dt} = f(t, y)$ une équation différentielle linéaire homogène.*

Il est pertinent de s'interroger sur la résolution de ce type d'équations. Quels outils et théorèmes permettent d'obtenir une solution à ce genre d'équation ? Posons un problème classique. Soit :

$$\begin{cases} \frac{dy}{dt} = a(t)y \quad \text{avec a une fonction, continue, de I dans K} \\ y(t_0) = y_0 \end{cases}$$

Quelles sont les solutions à cette équation différentielle ? Essayons de manipuler les membres de gauche et de droite afin de trouver une solution.

Considérons l'équation différentielle $\frac{dy}{dt} = a(t)y$ avec la condition initiale $y(t_0) = y_0$. L'équation

différentielle peut être réécrite sous forme intégrale. En intégrant les deux côtés de l'équation de t_0 à t, nous obtenons :

$$\int_{t_0}^{t} \frac{dy}{y} = \int_{t_0}^{t} a(s)\,ds$$

L'intégrale de gauche est une intégrale de $\frac{1}{y}$, ce qui donne :

$$[\ln|y|]_{t_0}^{t} = \int_{t_0}^{t} a(s)\,ds$$

En évaluant cette intégrale, nous obtenons :

$$\ln|y(t)| - \ln|y(t_0)| = \int_{t_0}^{t} a(s)\,ds$$

En exponentiant les deux côtés de l'équation, nous obtenons :

$$|y(t)| = |y(t_0)| \exp\left(\int_{t_0}^{t} a(s)\,ds\right)$$

Comme $y(t_0) = y_0$, nous avons :

$$|y(t)| = |y_0| \exp\left(\int_{t_0}^{t} a(s)\,ds\right)$$

Puisque $y(t)$ peut être positif ou négatif, nous pouvons écrire la solution générale comme :

$$y(t) = y_0 \exp\left(\int_{t_0}^{t} a(s)\,ds\right)$$

Ainsi, nous avons montré que la solution maximale (et définie sur I donc globale) de l'équation différentielle $\frac{dy}{dt} = a(t)y$, avec la condition initiale $y(t_0) = y_0$, est :

$$y(t) = y_0 \exp\left(\int_{t_0}^{t} a(s)\,ds\right)$$

Proposition 3. Solution d'équation linéaire scalaire homogène d'odre 1
Soit I un intervalle de $\mathbb{R}$, a une fonction continue de I dans $\mathbb{K}$. Soient $t_0 \in I$ et $y_0 \in \mathbb{K}$. La solution maximale (et définie sur I donc globale) de l'équation différentielle $\frac{dy}{dt} = f(t, y)$, avec la condition initiale $y(t_0) = y_0$, est

$$y(t) = y_0 \exp\left(\int_{t_0}^{t} a(s)\,ds\right).$$

Cependant, il nous reste à démontrer l'unicité de la solution maximale.

Soit w une solution maximale définie sur $J \subset I$, $t_0 \in J$ et $w(t_0) = w_0$. Nous pouvons dériver le terme suivant :

$$\frac{d}{dt}[w(t)\exp\left(-\int_{t_0}^{t} a(s)\,ds\right)] = (\frac{dw}{dt} - a(t)w(t))\exp\left(-\int_{t_0}^{t} a(s)\,ds\right) = (\frac{dw}{dt} - \frac{dw}{dt})\exp\left(-\int_{t_0}^{t} a(s)\,ds\right) = 0$$

Comme la dérivée est nulle, $t \mapsto [w(t)\exp\left(-\int_{t_0}^{t} a(s)\,ds\right)]$ est une constante sur J. Ainsi sur J :

$$w(t)\exp\left(-\int_{t_0}^{t} a(s)\,ds\right) = w(t_0) = w_0$$

Donc w et y coïncident sur J, de plus comme y est une solution maximale sur I cela implique : $J = I$ et $y = w$. De cette manière nous avons prouvé l'unicité de la solution sur $I \subset \mathbb{R}$.

Toutefois les équations du type

$$\begin{cases} \frac{dy}{dt} = a(t)y & \text{avec a une fonction, continue, de I dans K} \\ y(t_0) = y_0 \end{cases}$$

ne sont pas les seuls qu'on rencontre. On peut complexifier cette dernière en lui rajoutant un second membre et nous obtenons :

$$\begin{cases} \frac{dy}{dt} = a(t)y + b(t) & \text{avec a et b des fonctions, continues, de I dans K} \\ y(t_0) = y_0 \end{cases}$$

Pour résoudre cette équation différentielle, Alembert a observé en 1762 que la solution générale est la somme de la solution générale de l'équation homogène et de la solution particulière de l'équation avec second membre. Pour trouver cette solution particulière, nous utilisons la méthode de la variation de la constante introduite par Lagrange. Cette méthode se résume à poser :

$$y(t) = C(t) \exp\left(\int_{t_0}^{t} a(s)\, ds\right)$$

et trouver le terme $C(t)$. Pour ce faire, nous allons dériver notre expression et calculer $C(t)$.

$$\frac{dy}{dt} = [\frac{dC}{dt} + a(t)C(t)] \exp\left(\int_{t_0}^{t} a(s)\, ds\right) \iff \frac{dy}{dt} - a(t)y(t) = \frac{dC}{dt} \exp\left(\int_{t_0}^{t} a(s)\, ds\right)$$

De ce fait y est solution de :

$$y(t) = C(t) \exp\left(\int_{t_0}^{t} a(s)\, ds\right)$$

si et seulement si

$$\frac{dC}{dt} \exp\left(\int_{t_0}^{t} a(s)\, ds\right) = b(t) \iff C(t) = C_0 + \int_{t_0}^{t} b(s) exp(\left(-\int_{t_0}^{s} a(\rho)\, d\rho\right) ds$$

Ainsi :

$$y(t) = C_0 \exp\left(\int_{t_0}^{t} a(s)\, ds\right) + \int_{t_0}^{t} \exp\left(-\int_{t_0}^{s} a(\rho)\, d\rho + \int_{t_0}^{t} a(\rho)\, d\rho\right) b(s)\, ds$$

Comme la solution générale est la somme de la solution générale de l'équation homogène et de la solution particulière de l'équation avec second membre et $t_0 = t$ implique $C_0 = y_0$, nous obtenons :

$$y(t) = y_0 \exp\left(\int_{t_0}^{t} a(s)\, ds\right) + \int_{t_0}^{t} \exp\left(\int_{s}^{t} a(\rho)\, d\rho\right) b(s)\, ds$$

Ce qui nous permet d'introduire la proposition suivante :

Proposition 4. Solution d'équation linéaire scalaire d'odre 1
Soient $a, b : I \to \mathbb{K}$ des fonctions continues avec I un intervalle de $\mathbb{R}$. Soient $t_0 \in I$ et $y_0 \in \mathbb{K}$. La solution maximale (et définie sur I donc globale) de l'équation différentielle $\frac{dy}{dt} = a(t)y + b(t)$, avec la condition initiale $y(t_0) = y_0$, est

$$y(t) = y_0 \exp\left(\int_{t_0}^{t} a(s)\, ds\right) + \int_{t_0}^{t} b(s) \exp\left(\int_{t_0}^{s} a(\rho)\, d\rho\right) ds.$$

Nous avons vu une forme de solution d'équation linéaire homogène d'ordre 1, c'est-à-dire de la forme : $\frac{dy}{dt} = a(t)y$ mais qu'en est-il des équations de la forme :

$$\frac{dy}{dt} = A(t)y + B(t), (t, y) \in I \times \mathbb{K}^N \quad (\mathcal{L})$$

et de la forme :

$$\frac{dy}{dt} = A(t)y, (t, y) \in I \times \mathbb{K}^N \quad (\mathcal{L}_H)$$

avec une condition initiale ? Et quelle théorème nous permet de trouver la structure de la solution d'une équation linéaire scalaire d'ordre 1 comme on l'a fait dans la preuve de la **Proposition 4** ? Pour répondre à ces questions, il existe un théorème qui permet d'assurer l'existence et l'unicité de solutions globales avec une condition de Cauchy et qui a de nombreuses conséquences.

Théorème 1. Théorème de Cauchy-Lipschitz linéaire
Soient I un intervalle de $\mathbb{R}$, $A \in C(I, M_N(\mathbb{K}))$, $B \in C(I, \mathbb{K}^N)$, et $(t_0, y_0) \in I \times \mathbb{K}^N$. Alors il existe une unique solution globale de $(\mathcal{L})$ telle que :

$$\begin{cases} y : I \to \mathbb{K} \\ y(t_0) = y_0 \end{cases}$$

Nous connaissons l'existence d'une unique solution globale de $(\mathcal{L})$ mais une question naturelle qui doit nous venir à l'esprit est : quelle est la structure de cette solution globale ? En effet, le précédent théorème donne la structure de l'espace des solutions de $(\mathcal{L})$ et de $(\mathcal{L}_H)$. C'est l'une des conséquences du théorème de Cauchy-Lipschitz linéaire qui nous permet de trouver des solutions comme pour la **Proposition 4**.

Proposition 5. Structure de l'ensemble de S_H des solutions globales de $\mathcal{L}_H$
Soient I un intervalle de $\mathbb{R}$ et $A \in C(I, M_N(\mathbb{K})$. Alors

a) l'ensemble S_H des solutions maximales de $\mathcal{L}_H$ est un sous-espace vectoriel de $C(I, \mathbb{K}^N)$,

b) pour tout $t_0 \in I$, l'application

$$\Phi_{t_0} : S_H \to \mathbb{K}^N$$
$$y \mapsto y(t_0)$$

est un isomorphisme d'espaces vectoriels,

c) S_H est de dimension N.

En utilisant ces résultats, nous pouvons maintenant examiner comment l'ensemble des solutions S de l'équation différentielle $\mathcal{L}$ se structure en tant qu'espace affine de direction S_H.

Proposition 6. Structure de l'ensemble de S des solutions globales de $\mathcal{L}$
Soient I un intervalle de $\mathbb{R}$, $A \in C(I, M_N(\mathbb{K}))$ et $B \in C(I, \mathbb{K}^N)$. L'ensemble des solutions S de l'équation différentielle $\mathcal{L}$ est un $\mathbb{K}$-espace affine de direction S_H.

Avant de continuer la théorie sur les équations différentielles linéaires, nous devons poser quelques notions de vocabulaire utiles à la compréhension de certaines propositions et théorèmes.

Définition 10. Système fondamental
Un système fondamental de solutions de l'équation différentielle $\mathcal{L}_H$ est une famille $(y_1, \ldots, y_N)$ de N solutions indépendantes de $\mathcal{L}_H$

Définition 11. Matrice fondamentale
La matrice $\Phi(t) = (y_1(t) \cdots y_N(t))$, constituée de ces N solutions écrites comme des vecteurs colonnes, est appelée une matrice fondamentale de $\mathcal{L}_H$

Définition 12. Wronskien
Le wronskien est le déterminant de $\Phi(t)$.

De ce fait, nous pouvons utiliser ces définitions pour introduire une nouvelle proposition sur les formes des solutions de $(\mathcal{L}_H)$.

Proposition 7. Forme des solutions de $\mathcal{L}_H$
Soient I un intervalle de $\mathbb{R}$ et $A \in C(I, M_N(\mathbb{K}))$.

1. Soient $y_1, \ldots, y_N$ des solutions indépendantes de $\mathcal{L}_H$. Si y est une solution de (LH), alors il existe des constantes $C_1, \ldots, C_N \in \mathbb{K}$ telles que

$$y = C_1 y_1 + \cdots + C_N y_N.$$

2. Soit $\Phi(t)$ une matrice fondamentale du système $\mathcal{L}_H$. Alors les solutions s'écrivent

$$y(t) = \Phi(t)C, \quad t \in I,$$

où C est un vecteur quelconque constant de $\mathbb{K}^N$.

Si nous disposons d'un problème de Cauchy, d'après le théorème de Cauchy-Lipschitz linéaire, il existe une unique solution globale de $\mathcal{L}_H$ telle que $y(t_0) = y_0$ et les solutions s'écrivent :

$$y(t) = \Phi(t)C, \quad t \in I, C \in \mathbb{K}^N$$

Si $t = t_0$ alors $y(t_0) = y_0 = \Phi(t_0)C$ et $C = \Phi(t_0)^{-1} y_0$. Par conséquent nous obtenons le corollaire suivant.

Corollaire 1. Solutions de $\mathcal{L}_H$ avec condition de Cauchy
Considérons un intervalle I de $\mathbb{R}$, un vecteur $y_0 \in \mathbb{K}^N$ et une matrice $A \in C(I, M_N(\mathbb{K}))$. Supposons que $\Phi(t)$ soit une matrice fondamentale du système $\mathcal{L}_H$. La solution de l'équation différentielle $\mathcal{L}_H$ qui satisfait la condition initiale $y(t_0) = y_0$ est donnée par :

$$y(t) = \Phi(t)\Phi(t_0)^{-1}y_0, \quad t \in I.$$

Actuellement nous disposons de toutes les ressources nécessaires pour traiter les équations différentielle linéaire d'ordre n.

1.2.2 Équations différentielles d'ordre n

À présent nous traiteront des équations différentielles de la forme :

$$\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_1\frac{dy}{dt} + a_0 y = b(t), \quad \text{avec } (t, y) \in I \times \mathbb{K} \quad (\xi)$$

En ramenant à l'odre 1 :

$$Y = \begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ \vdots \\ Y_{n-1} \\ Y_n \end{pmatrix} = \begin{pmatrix} y \\ \frac{dy}{dt} \\ \frac{d^2y}{dt^2} \\ \vdots \\ \frac{d^{n-2}y}{dt^{n-2}} \\ \frac{d^{n-1}y}{dt^{n-1}} \end{pmatrix}$$

$$A(t) = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0(t) & -a_1(t) & -a_2(t) & \cdots & -a_{n-1}(t) \end{pmatrix},$$

$$B(t) = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b(t) \end{pmatrix},$$

Nous obtenons de cette manière l'équation différentielle avec second membre suivante :

$$\frac{dY}{dt} = A(t)Y + B(t) \quad \text{avec } (t, Y) \in I \times \mathbb{K}^N \quad (\mathcal{L})$$

Et si B = 0, nous avons une équation différentielle homogène :

$$\frac{dY}{dt} = A(t)Y \quad \text{avec } (t, Y) \in I \times \mathbb{K}^N \quad (\mathcal{L}_H)$$

En utilisant le théorème de Cauchy-Lipschitz pour les systèmes linéaires d'ordre n, nous pouvons établir un théorème qui s'applique aux systèmes d'équations différentielles linéaires d'ordre n avec des coefficients qui peuvent dépendre du temps t

Théorème 2. Cauchy-Lipschitz linéaire d'ordre n
Soient I un intervalle de $\mathbb{R}$, $A \in C(I, M_N(\mathbb{K}))$ et $B \in C(I, \mathbb{K}^N)$. Considérons le système d'équations différentielles d'ordre n suivant :

$$\frac{dY}{dt} = A(t)Y + B(t), \quad \text{avec } (t, Y) \in I \times \mathbb{K}^N$$

Alors, pour toute condition initiale $Y(t_0) = Y_0 \in \mathbb{K}^N$, il existe une unique solution globale de $(\mathcal{L})$ telle que :

$$\begin{cases} Y : I \to \mathbb{K}^N \\ Y(t_0) =_0 \end{cases}$$

Nous sommes en mesure d'obtenir un théorème plus spécifique, un théorème qui s'applique aux équations différentielles linéaires d'ordre n avec des coefficients constants.

Théorème 3. Théorème Cauchy-Lipschitz linéaire d'ordre n à coefficients constants
Soit I un intervalle de $\mathbb{R}$, et soient $a_0, \ldots, a_{n-1} \in \mathbb{K}$ et $b \in C(I, \mathbb{K})$. Considérons $t_0 \in I$ et $y_0, y_1, \ldots, y_{n-1} \in \mathbb{K}$. Alors, il existe une unique solution $y : I \to \mathbb{K}$ de l'équation différentielle $(\mathcal{L}_H)$ telle que :

$$\begin{cases} y(t_0) = y_0 \\ \frac{dy}{dt}(t_0) = y_1 \\ \frac{d^2y}{dt^2}(t_0) = y_2 \\ \vdots \\ \frac{d^{n-1}y}{dt^{n-1}}(t_0) = y_{n-1} \end{cases}$$

Dans un premier temps, trouvons les solutions de :

$$\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_1\frac{dy}{dt} + a_0 y = 0, \quad \text{avec } (t, y) \in I \times \mathbb{K} \quad (\xi_{\mathbf{H}})$$

Pour se faire nous devons trouver le polynome caractéristique de (ξ).

Définition 13. Polynôme caractéristique et équation caractéristique de (ξ)
Le polynôme caractéristique associé à l'équation différentielle (ξ) est le polynôme :

$$P(X) = X^n + a_{n-1}X^{n-1} + \cdots + a_1 X + a_0. \quad (\mathcal{P})$$

L'équation caractéristique associée à l'équation différentielle (ξ_H) est l'équation $P(X) = 0$.

Pour trouver les solutions de (ξ_H), nous devons trouver les racines complexes, $r_1, \ldots, r_p$ de $(\mathcal{P})$ et leurs multiplicités algébrique respectives $m_1, \ldots, m_p$. Alors, les solutions de l'équation différentielle linéaire homogène (ξ_H) sont de la forme :

$$y(t) = \left(\lambda_{1,1}t^{m_1-1} + \cdots + \lambda_{1,m_1-1}t + \lambda_{1,m_1}\right)e^{r_1 t} + \cdots + \left(\lambda_{p,1}t^{m_p-1} + \cdots + \lambda_{p,m_p-1}t + \lambda_{p,m_p}\right)e^{r_p t}$$

où $\lambda_{j,k} \in \mathbb{C}$ sont des constantes quelconques. En d'autres termes, une base de l'ensemble des solutions de l'équation différentielle (ξ_H) est formée par les fonctions de la forme :

$$t \mapsto t^l e^{r_k t},$$

avec $1 \leq k \leq p$ et $0 \leq l \leq m_k - 1$

Dans un second temps, si l'équation différentielle est de la forme (ξ_H) avec un second membre telle que :

$$\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_1\frac{dy}{dt} + a_0 y = b(t), \quad \text{avec } (t, y) \in I \times \mathbb{K} \quad (\xi)$$

Plusieurs choix s'offrent à nous : soit nous faisons varier la constante, soit nous cherchons une solution particulière déjà connue en reconnaissant la forme du second membre. Cependant, ce n'est pas le but de notre recherche ; nous vous conseillons la lecture du livre de Florent Berthelin, Équations différentielles. Avant de clôturer cette section, nous nous devons d'énoncer une proposition fondamentale en théorie des équations linéaires. Cette proposition concerne les solutions. Introduisons-la avec un exemple :

$$\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_1\frac{dy}{dt} + a_0 y = b(t), \quad \text{avec } (t, y) \in I \times \mathbb{K} \quad (\xi)$$

Posons $a_0, \ldots, a_{n-1} \in \mathbb{C}$, $c_1, \ldots, c_N \in \mathbb{C}$ et $b_1, \ldots, b_N \in C(I, \mathbb{C})$ avec I un intervalle de $\mathbb{R}$, Si $y_1, y_2, \cdots, y_{n-1}, y_n$ est une solution de (ξ). Montrons que la combinaison linéaire

$$y = \sum_{i=1}^{N} c_i y_i$$

est une solution de :

$$\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_1\frac{dy}{dt} + a_0 y = \sum_{i=1}^{N} c_i b_i(t)$$

La démonstration repose sur la réécriture de l'expression $\sum_{i=1}^{N} c_i b_i(t)$. Regroupons dans un premier temps tous les termes à l'ordre n, puis $n-1$, et ainsi de suite comme suit :

$$\sum_{i=1}^{N} \frac{d^n}{dt^n} c_i y_i + a_{n-1}\sum_{i=1}^{N} \frac{d^{n-1}}{dt^{n-1}} c_i y_i + \cdots + a_1 \sum_{i=1}^{N} \frac{d}{dt} c_i y_i + a_0 \sum_{i=1}^{N} c_i y_i$$

$$= c_1\left(\frac{d^n y_1}{dt^n} + \sum_{k=0}^{n-1} a_k \frac{d^k y_1}{dt^k}\right) + c_2\left(\frac{d^n y_2}{dt^n} + \sum_{k=0}^{n-1} a_k \frac{d^k y_2}{dt^k}\right) + \cdots c_N\left(\frac{d^n y_N}{dt^n} + \sum_{k=0}^{n-1} a_k \frac{d^k y_N}{dt^k}\right)$$

$$= \sum_{i=1}^{N} c_i b_i = \frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_1\frac{dy}{dt} + a_0 y$$

Ainsi nous pouvons introduire notre proposition.

Proposition 8. Principe de superposition des solutions
Soient $a_0, \ldots, a_{n-1} \in \mathbb{C}$, $c_1, \ldots, c_N \in \mathbb{C}$ et $b_1, \ldots, b_N \in C(I, \mathbb{C})$ avec I un intervalle de $\mathbb{R}$. Si pour $l = 1, \ldots, N$, y_l est une solution de

$$\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_1\frac{dy}{dt} + a_0 y = b_l(t),$$

alors $c_1 y_1 + c_2 y_2 + \cdots + c_N y_N$ est une solution de :

$$\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_1\frac{dy}{dt} + a_0 y = c_1 b_1(t) + c_2 b_2(t) + \cdots + c_N b_N(t).$$

Cette dernière stipule que lorsqu'on dispose de plusieurs solutions distinctes pour des équations différentielles linéaires non homogènes d'ordre $n \geq 1$, une combinaison linéaire de ces solutions constitue également une solution valide. Ce principe est particulièrement avantageux dans les domaines de la physique et de l'analyse numérique, où les systèmes peuvent souvent être divisés en sous-systèmes plus simples.

Pourtant, dans la vie quotidienne, les phénomènes physiques ou autres ne sont pas toujours représentés mathématiquement par des équations différentielles linéaires. Il existe une autre catégorie d'équations, à savoir les équations différentielles non linéaires de la forme $\frac{dy}{dt} = f(t, y)$, où y est une fonction vectorielle. L'inconvénient de ce type d'équations réside dans le fait qu'elles ne possèdent pas autant de propriétés que les équations différentielles linéaires scalaires.

1.2.3 Équations différentielles non linéaires

Définition 14. Équation différentielle non linéaire
On dit que l'équation $\frac{dy}{dt} = f(t, y)$ est une équation différentielle non linéaire si $f(t, y)$ ne peut pas être exprimée sous la forme $A(t)y + B(t)$, où

$$\begin{cases} A \in \mathbb{M}_N(\mathbb{K}) \\ B \in \mathbb{K}^N \end{cases}$$

Remarque 5. *Dans ce contexte, $f(t, y)$ peut inclure des termes non linéaires en y, tels que des produits, des puissances ou d'autres fonctions non linéaires de y. Les équations différentielles non linéaires sont souvent plus complexes à analyser et à résoudre que leurs homologues linéaires.*

Soient $N \in \mathbb{N}^*$, $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$, $\mathbb{O}$ un ouvert de $\mathbb{R} \times \mathbb{K}^N$ et $f : \mathbb{O} \mapsto \mathbb{K}^N$. Pour la suite, prenons en compte l'équation différentielle suivante :

$$\frac{dy}{dt} = f(t, y), (t, y) \in \mathbb{O} \quad (\Psi)$$

L'une des principales notions à retenir est que y s'appelle la variable d'état et t la variable réelle, dite variable de temps. Nous pouvons nous demander : comment résoudre ce genre d'équation ? Quels sont les caractères d'une solution de Ψ ? La continuité assure-t-elle l'existence et l'unicité de solutions globales ? Nous allons voir que l'on ne peut pas appliquer les mêmes propositions et

théorèmes que pour les équations différentielles linéaires. Un concept crucial dans cette section est le caractère lipschitzien d'une fonction. Ce caractère peut être local ou global selon les fonctions, mais pour le définir, il faut déterminer quelle variable en décide : est-ce la variable d'état y ou la variable de temps t ? Avant d'y répondre, introduisons ce que signifie une "fonction lipschitzienne".

Définition 15. Fonction lipschitzienne
Soit $I \subset \mathbb{K}^N$ un ensemble et $f : X \to \mathbb{K}^N$. La fonction f est dite lipschitzienne sur I s'il existe une constante $k \in [0, +\infty[$ telle que :

$$||f(y) - f(x)|| \leq k||y - x|| \quad \text{pour tous } x, y \in I.$$

On dit alors que f est k-lipschitzienne. Si $k \in [0, 1[$ alors f est k-contractante

Remarque 6. *Une conséquence intéressante de la k-lipschitziennité d'une fonction est que cette fonction est continue.*

Nous pouvons ainsi nous donner un cadre où il existe des solutions de Ψ. Ce dernier est constitué des fonctions localement lipschitziennes en la variable d'état. On ne choisit pas la variable de temps t, car elle est considérée comme une variable indépendante, une variable qui évolue de manière autonome et qui n'est pas affectée par les états du système, représenté par y. Pour illustrer nos propos, dans une équation différentielle décrivant le mouvement d'un objet, le temps avance indépendamment des positions et vitesses de l'objet. De ce fait, les conditions et propriétés des fonctions localement lipschitziennes (comme le fait que des perturbations minimes n'entraînent pas de changement important dans la solution) sont considérées et appliquées à la variable d'état y pour garantir la stabilité des solutions, mais aussi leur unicité. Dès lors, nous pouvons énoncer ce qu'est une fonction localement lipschitzienne.

Définition 16. Fonction localement lipschitzienne par rapport à la variable d'état y
Soit $\mathbb{O}$ un ouvert de $\mathbb{R} \times \mathbb{K}^N$. Une fonction $f : \mathbb{O} \mapsto \mathbb{K}^N$ est dite localement lipschitzienne par rapport à la variable d'état y si pour tout $(\tilde{t}, w) \in \mathbb{O}$, il existe $C_0 = [\tilde{t} - T, \tilde{t} + T] \times B(w, r) \subset \mathbb{O}$ avec $T, r > 0$ et $k \geq 0$ tels que pour tous $(t, y_1), (t, y_2) \in C_0$,

$$||f(t, y_1) - f(t, y_2)|| \leq k||y_1 - y_2||$$

Après avoir introduit la notion de fonction localement lipschitzienne par rapport à la variable d'état, il est essentiel de différencier cette propriété de celle d'une fonction globalement lipschitzienne. La différence principale réside dans la condition lipschitzienne : la condition locale est valable dans un voisinage restreint autour de chaque point, tandis que la condition globale s'applique sur l'ensemble du domaine de définition. Cette distinction est cruciale, car elle affecte la stabilité et l'unicité des solutions des équations différentielles de la forme Ψ.

Définition 17. Fonction globalement lipschitzienne par rapport à la variable d'état y
Soit $\mathbb{O} = I \times X$ où I est un intervalle de $\mathbb{R}$ et $X \subset \mathbb{K}^N$. Une fonction $f : \mathbb{O} \mapsto \mathbb{K}^N$ est dite globalement lipschitzienne par rapport à la variable d'état, uniformément par rapport à la variable de temps, si pour tout intervalle compact $\mathcal{K} \subset I$, il existe $k > 0$ tel que pour tous $(t, y_1), (t, y_2) \in \mathbb{K}^N \times X$,

$$||f(t, y_1) - f(t, y_2)|| \leq k||y_1 - y_2||$$

Cette condition garantit que les variations de f sont proportionnelles aux variations de la variable d'état, assurant ainsi une certaine régularité et stabilité des solutions de (Ψ).

Remarque 7. *Si une fonction f est de classe C^1 sur un ensemble $I \subset \mathbb{R}$, alors f est localement lipschitzienne sur I.*

En parlant de stabilité et régularité des solutions, il existe un théorème utilisé pour justifier l'existence d'une unique solution sur tout un intervalle :

Théorème 4. Théorème de Cauchy-Lipschitz cas globalement lipschitzien
Soient I un intervalle de $\mathbb{R}$ et $f : I \times \mathbb{K}^N \mapsto \mathbb{K}^N$ continue (dans toutes les variables) et globalement lipschitzienne par rapport à la variable d'état uniformément par rapport à la variable de temps. Soit $(t_0, y_0) \in I \times \mathbb{K}^N$. Alors il existe une unique solution globale de (Ψ) telle que :

$$\begin{cases} y : I \mapsto \mathbb{K}^N \\ y(t_0) = y_0 \end{cases}$$

Comme pour la continuité, on peut définir le caractère lipschitzien localement en un point ou sur un intervalle avec des propriétés précises, ou encore sur un ensemble, comme dans le **Corollaire2**. Pour assurer le caractère lipschitzien et ainsi obtenir un théorème local, nous devons définir ce qu'est "local" et ce qui garantit la stabilité des solutions. Pour ce faire, nous avons besoin d'une notion essentielle : le cylindre de sécurité. Il permet de garantir la continuité et la régularité des solutions dans un espace (intervalle). Ainsi, en établissant un cylindre de sécurité, on s'assure que les solutions restent dans une zone connue, ce qui facilite leur étude. Ainsi, nous obtenons la définition suivante.

Définition 18. Cylindre de sécurité
Soit $\mathbb{O}$ un ouvert de $\mathbb{R} \times \mathbb{K}^N$, $f : \mathbb{O} \mapsto \mathbb{K}^N$ continue et $y_0 \in \mathbb{K}^N$. On dit que $\mathcal{K} = [t_0 - \alpha_1, t_0 + \alpha_1] \times B(y_0, r_1) \subset \mathbb{O}$, avec $\alpha_1, r_1 > 0$, est un cylindre de sécurité pour (β) si pour toute fonction $y \in \tilde{X} = \mathcal{K}([t_0-\alpha_1, t_0+\alpha_1], B(y_0, r_1))$, la fonction $\Phi(y)$ à valeurs dans Kn définie sur $[t_0-\alpha_1, t_0+\alpha_1]$ par

$$\Phi(y)(t) = y_0 + \int_{t_0}^{t} f(s, y(s))\, ds$$

appartient à $\tilde{X}$.

Ce concept nous sert, et est cruciale, pour démontrer le théorème de Cauchy-Lipschitz surtout pour les points d'existence et d'unicité des solutions.

Théorème 5. Théorème de Cauchy-Lipschitz cas localement lipschitzien
Soit $\mathbb{O}$ un ouvert de $\mathbb{R} \times \mathbb{K}^N$ et $f : \mathbb{O} \mapsto \mathbb{K}^N$ continue et localement lipschitzienne par rapport à la variable d'état y. Soit $(t_0, y_0) \in \mathbb{O}$. Alors, il existe $\alpha > 0$ tel que le problème de Cauchy $y(t_0) = y_0$ pour (Ψ) ait une solution unique y définie sur $[t_0 - \alpha, t_0 + \alpha]$.

Après avoir établi le théorème de Cauchy-Lipschitz dans le cadre localement lipschitzien et que nous savons qu'il y a existence et unicité des solutions si des conditions sont remplies, il est pertinent de se pencher sur le prolongement et le raccordement de ces dernières. Cela permet d'étendre des solutions locales sur des intervalles plus grands sous certaines conditions d'unicité et de continuité.

Lemme 3. Raccordement de solutions si unicité
Sous les hypothèses du théorème de Cauchy-Lipschitz 2, supposons que y soit une solution de (Ψ) sur un intervalle $|\alpha, \gamma[$ et que z soit une solution de (Ψ) sur un intervalle $]b, d|$ avec $a < b < c < d$. Supposons que ces deux solutions coïncident en un point de $]b, c[$. Alors, en posant

$$\tilde{y}(t) = \begin{cases} y(t) & \text{pour } t \in |a, e[, \\ z(t) & \text{pour } t \in [e, d|, \end{cases}$$

où e est un point quelconque de $]b, c[$

Après avoir établi le théorème de Cauchy-Lipschitz dans le cadre localement lipschitzien et sachant qu'il y a existence et unicité des solutions sous certaines conditions, il est pertinent de se pencher sur le prolongement et le raccordement de ces dernières. Cela permet d'étendre des solutions locales sur des intervalles plus grands, sous certaines conditions d'unicité et de continuité.

Théorème 6. Théorème de Cauchy-Lipschitz non linéaire
Soit $\mathbb{O}$ un ouvert de $\mathbb{R} \times \mathbb{K}^N$ et $f : \mathbb{O} \mapsto \mathbb{K}^N$ continue dans toutes ses variables et localement lipschitzienne par rapport à la variable d'état. Soit $(t_0, y_0) \in \mathbb{O}$. Alors, il existe une unique solution maximale de (Ψ) telle que :

$$\begin{cases} y : I \mapsto \mathbb{K}^N, \text{I un ouvert} \\ y(t_0) = y_0 \end{cases}$$

Pour approfondir notre compréhension des dynamiques des systèmes différentiels, il est nécessaire d'examiner le comportement des solutions aux frontières de leurs intervalles de définition. Le théorème des extrémités nous aide à identifier les conditions dans lesquelles une solution maximale atteint une borne infinie.

Théorème 7. Théorème des bouts
Soit $\mathbb{O} =]\alpha, \beta[\times \mathbb{K}^N$ et $f : \mathbb{O} \mapsto \mathbb{K}^N$ continue dans toutes ses variables et localement lipschitzienne par rapport à la variable d'état. Soit $y :]\gamma, \delta[\mapsto \mathbb{K}^N$ une solution maximale de $\frac{dy}{dt} = f(t, y)$.
Si $\delta < \beta$, alors

$$lim_{t \to \delta} \|y(t)\| = +\infty$$

Cela implique que si $t \mapsto y(t)$ est bornée, alors $\delta = \beta$.

Imaginons que nous avons une fonction f qui dépend du temps t et d'une autre variable y. Cette fonction est définie dans une région spécifique de l'espace, que nous appelons $\mathcal{E} =]\alpha, \beta[\times \mathbb{K}^N$. Maintenant, considérons une solution y de l'équation différentielle $\frac{dy}{dt} = f(t, y)$, définie sur un intervalle $]\gamma, \delta[$

Le théorème des extrémités nous dit que si $\delta < \beta$ alors la norme de $y(t)$ devient infinie lorsque t approche δ. En d'autres termes, $y(t)$ devient très grand à mesure que t s'approche de δ. Cela signifie également que si $y(t)$ reste bornée, alors l'intervalle $]\gamma, \delta[=]\gamma, \beta[$

Après avoir exploré les concepts clés des équations différentielles et des théorèmes associés, il devient impératif de se tourner vers des méthodes pratiques pour les résoudre. Bien que les solutions analytiques soient précieuses, comme les méthodes de variation de la constante, l'abaissement de l'ordre et bien d'autres encore, elles ne sont pas toujours accessibles pour des systèmes complexes ou non linéaires. C'est là qu'intervient l'approximation numérique.

Chapitre 2

Introduction à l'approximation numérique

Dans de nombreux cas, les solutions explicites des équations différentielles, obtenues par des méthodes analytiques, ne peuvent pas être déterminées. Cela peut être dû à la complexité des équations, à la nature des conditions initiales ou aux limitations imposées par les domaines d'étude. Ces limitations constituent un défi majeur dans l'application pratique des équations différentielles, ces dernières jouant un rôle central dans la compréhension du monde qui nous entoure.

Une solution à ce problème est la résolution numérique, qui repose sur des principes d'approximation numérique de polynômes, d'intégrales, mais aussi d'équations différentielles. Cette approche consiste à approximer la solution par des méthodes algorithmiques, qui produisent une solution approchée au lieu d'une solution exacte. Ces approximations permettent d'obtenir des résultats exploitables dans des contextes où l'analyse classique échoue.

2.1 Objectifs et limites des méthodes numériques

L'objectif principal des méthodes numériques est de fournir des solutions approchées à des problèmes mathématiques, en utilisant des calculs algorithmiques. Ces méthodes permettent de transformer des problèmes continus (décrits par des équations différentielles, intégrales ou aux dérivées partielles) en des problèmes discrets, qui peuvent être résolus à l'aide d'ordinateurs. Par exemple, la résolution numérique d'une équation différentielle consiste à approximer la solution en un ensemble fini de points, plutôt que de chercher une expression analytique globale.

2.1.1 Limites des méthodes numériques

Malgré leur capacité à fournir des solutions approchées pour des problèmes complexes, les méthodes numériques présentent certaines limitations qu'il est important de considérer dans notre étude.

1. **Erreurs numériques** : Les solutions sont approximatives et peuvent être affectées par des erreurs d'arrondi, de troncature ou de discrétisation.
2. **Coût computationnel** : Pour des problèmes de très grande taille, le temps de calcul peut devenir excessif.
3. **Stabilité** : Certaines méthodes peuvent devenir instables, produisant des solutions divergentes ou non physiques.
4. **Sensibilité aux conditions initiales** : Les solutions numériques peuvent être fortement influencées par des erreurs dans les données d'entrée, surtout pour les problèmes mal posés.

Pour tirer pleinement parti des méthodes numériques, il est donc essentiel de comprendre les concepts fondamentaux qui sous-tendent leur fonctionnement. Ces concepts incluent la représentation des nombres en machine, l'analyse des erreurs, ainsi que les propriétés de consistance, de stabilité et de convergence des schémas numériques.

2.2 Représentation des nombres et erreurs

2.2.1 Rappel sur les séries de Taylor

Les **séries de Taylor** sont un outil fondamental en analyse mathématique pour approximer des fonctions par des polynômes.

Définition 19. Série de Taylor
La série de Taylor d'une fonction $f(x)$ infiniment dérivable autour d'un point a est une série infinie qui représente $f(x)$ sous forme de polynôme. Elle est donnée par :

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

où $f^{(n)}(a)$ est la dérivée n-ième de f évaluée au point a,

Définition 20. Formule de Taylor-Lagrange
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 :

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

où $R_n(x)$ est le terme de reste, donné par :

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

avec ξ un point 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 $R_n = O(h^{n+1})$, où h représente $x - c$.

Ces formules sont primordiales en analyse numérique, car elles interviennent dans de nombreux théorèmes et propositions concernant diverses erreurs de quadrature élémentaire, mais aussi dans l'étude de la convergence des schémas numériques. Mais que signifie l'erreur ?

Définition 21. Erreur
L'erreur est la différence entre la valeur exacte et la valeur approximative obtenue par une méthode numérique. Elle peut être exprimée de la manière suivante :

$$Erreur = |Valeur\ exacte - Valeur\ approximative|$$

Avec
— La valeur exacte est la solution théorique précise.
— La valeur approximative est la solution obtenue par une méthode d'approximation.

Cependant, il existe plusieurs types d'erreurs.

2.2.2 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 d'une approximation de l'algorithme, d'erreurs dans les données ou, enfin, d'erreurs dues 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 22. Erreur absolue et relative
Soit $\tilde{a}$ la valeur approchée d'une quantité dont la valeur exacte est a. On appelle :
— **l'erreur absolue** la quantité $\tilde{a} - a$

— **l'erreur relative** la quantité $\frac{\tilde{a}-a}{a}$

De cette définition, on introduit la notion de **chiffre/nombre significatif**. Le nombre de chiffres significatifs est lié à l'erreur relative. Les chiffres corrects de $\tilde{a}$, à partir du premier chiffre ou de la première décimale non nulle, sont appelés **chiffres significatifs**.

2.3 Représentation des nombres en virgule flottante : limitations et arrondis

La première étape pour maîtriser les méthodes numériques consiste à comprendre comment les nombres sont représentés et manipulés par les ordinateurs, car cette représentation influence directement la précision et la fiabilité des calculs.

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 :

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

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. Cependant, nous ferons l'approximation que tout se passe réellement en base 10.

2.3.1 Limitations de la mantisse et de l'exposant

La représentation en virgule flottante utilise une **mantisse** m et un **exposant** q, stockés sur un nombre fini de bits. Cela implique :

- Un nombre fini de nombres réels peut être représenté.
- L'exposant q est limité par le nombre de bits alloués. Par exemple, avec 8 bits, $q \in [-127, 128]$.
- Si $q > 128$, on parle d'**overflow** (erreur critique).
- Si $q < -127$, on parle d'**underflow** (souvent remplacé par 0, mais pouvant causer des erreurs critique en cas de division).

La mantisse est également stockée sur un nombre fini de bits, ce qui limite la précision. Cela implique que la précision d'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 ses décimales, on choisit le nombre le plus proche pouvant être représenté comme valeur de x. C'est ce qu'on appelle l'**arrondi**. Il est donc naturel d'introduire la notion d'**epsilon machine**. L'epsilon machine représente la différence entre deux mantisses consécutives, c'est-à-dire le plus petit nombre machine tel que $1 + \epsilon > 1$ sur la machine.

2.3.2 Arrondi selon le standard IEEE

L'arrondi consiste à trouver le nombre représentable $\tilde{\alpha}$ le plus proche de α :

1. En base 10 avec p décimales, la mantisse de $\tilde{x}$ conserve les $p-1$ premiers chiffres de x.
2. Le p-ème chiffre est incrémenté si le $(p+1)$-ème chiffre est ≥ 5.

Maintenant que nous avons analysé comment les nombres sont représentés dans un ordinateur et comment l'arrondi, selon le standard IEEE, influe sur la précision des calculs numériques, il est temps de voir comment ces concepts se mettent en œuvre dans la résolution numérique concrète de problèmes. En effet, une fois que nous maîtrisons l'impact des erreurs d'arrondi et la représentation en virgule flottante, nous pouvons aborder la conception d'une *approximation d'intégrales*, où la maîtrise de la représentation des nombres en virgule flottante est primordiale pour obtenir une approximation juste des solutions d'intégrales.

Cependant, nous remarquons que le choix des différents points et intervalles est primordial pour garantir l'existence et l'unicité de nos solutions dans le cadre des équations différentielles, qui sont étroitement liées aux approximations d'intégrales. Nous nous devons d'introduire un mécanisme fondamental : la subdivision d'intervalle.

2.4 Subdivision d'un intervalle

2.4.1 Définition

Définition 23. Subdivision d'un intervalle
Soit un intervalle I = $[a, b]$ de la droite réelle, où $a < b$. Une subdivision de cet intervalle est un ensemble ordonné de points $\{p_0, p_1, p_2, \ldots, p_N\}$ tels que :

$$a = p_0 < p_1 < p_2 < \cdots < p_N = b.$$

Ces points divisent l'intervalle $[a, b]$ en N sous-intervalles jouxtant de la forme $I_i = [p_{i-1}, p_i]$, pour $i = 1, 2, \ldots, N$. Lorsque la longueur des sous-intervalles est constante et égale à h, on parle de subdivision uniforme, avec $h = \frac{b-a}{N}$. Autrement, la subdivision est dite non uniforme et $h_i = p_{i+1} - p_i$.

Dans notre cas des équations différentielles, nous dirons que l'on discrétise l'intervalle I. Pour être familier avec les notations qui suivent dans les chapitres suivants, initialisons les différents objets que nous utiliserons.

2.4.2 Notations

Soit N $\in \mathbb{N}^*$, on considère une subdivision de I = [t_{ini},t_{ini} + T] de (N+1 points) de I, ainsi :
— On note $(t_n)_{n\in\{0,\ldots,N\}}$ les (N+1) points de notre subdivision où : $t_0 = t_{ini}$ et $t_N = t_{ini} + T$
— On note $\Delta t_n = t_{n+1}$ - t_n, la longueur entre deux point
— On note le pas maximal de subdivision $\Delta t_{max,N} = max_{n\in\{0,\ldots,N-1\}} \Delta t_n$
— Dans le cas d'une subdivision uniforme $\Delta t = \frac{T}{N}$ et $\lim_{N\to+\infty} \Delta t = 0$

Maintenant, nous sommes prêts à découvrir le monde de l'approximation d'intégrales avec l'utilisation des discrétisations d'intervalles.

2.5 Approximation d'intégrales

Nous avons vu dans la partie théorique des équations différentielles que les intégrales sont au cœur des équations différentielles. Cependant, lorsqu'on souhaite calculer une intégrale numériquement, nous ne pouvons pas nous lancer sans méthodologie ni idée nous devons avoir un aperçu de ce que signifie l'approximation et comment approximer.

2.5.1 Introduction à l'approximation d'intégrale

Pour cela on se donne une fonction $f : [a, b] \to \mathbb{R}$ et on cherche à approcher la valeur

$$I(f) = \int_a^b f(x)\, dx.$$

Soit $n \in \mathbb{N}^*$ et $n + 1$ nœuds (points) distincts $(p_i)_{i\in\{0,\ldots,n\}}$. Pour $i \in \{0, \ldots, n\}$, $y_i = f(p_i)$. Nous souhaitons trouver un polynôme de degré au plus n qui passe exactement par les $n+1$ points distincts $(p_i)_{i\in\{0,\ldots,n\}}$, notons ce polynôme $\pi_n \in \mathbb{R}_n[X]$. Ainsi par unicité de ce polynôme π_n et par ses propriétés qui sont qu'il passe exactement par les points $(p_i)_{i\in\{0,\ldots,n\}}$, il s'exprime de la manière suivante grâce à la formule de Lagrange :

$$\pi_n(p) = \sum_{i=0}^{n} f(p_i)\, L_i(p) \quad \text{où} \quad L_i(p) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{p - p_j}{p_i - p_j}.$$

Par linéarité et comme le polynôme $\pi_n(p)$ passe exactement par les (n+1) points, nous obtenons :

$$I(f) = \int_a^b f(p)\, dp \iff I(\pi_n) = \sum_{i=0}^{n} f(p_i) \int_a^b \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{p - p_j}{p_i - p_j}\, dp$$

Nous ne nous attardons pas sur les différentes propriétés et preuves liées à ces résultats. Maintenant que nous savons comment approximer l'intégrale d'une fonction, nous allons introduire un exemple de formule de quadrature simple : la formule des rectangles à gauche.

2.5.2 Principe de la formule de quadrature élémentaire

Nous voulons approximer l'intégrale $I(f) = \int_a^b f(x)\,dx$, où $f : [a,b] \to \mathbb{R}$. Pour cela, nous allons utiliser les informations à notre disposition. Nous savons que $[a,b]$ est un intervalle réel, où $a < b$. On considère le polynôme $\pi_0 \in \mathbb{R}_0[X]$. Ainsi, pour notre intervalle $[a,b]$, il nous faut un seul point où π_0 passe ; notons ce point ω. Ainsi, nous approchons $I(f)$ de la manière suivante :

$$I(f) = \int_a^b f(p)\,dp = \int_a^b \pi_0(p)\,dp = \int_a^b f(\omega)\,dp = (b-a)f(\omega)$$

2.5.3 Principe de la formule de quadrature composé

Si nous subdivisons l'intervalle [a,b] en (n+1) points, donc en n intervalles tels que $I_i = [p_i, p_{i+1}]$ $\forall i \in \{0,\dots,n-1\}$, nous obtenons par linéarité de l'intégrale :

$$I(f) = \int_a^b f(p)\,dp = \sum_{i=0}^{n-1} \int_{p_i}^{p_{i+1}} f(t)\,dt$$

Et par le même procédé que pour le principe de la formule des rectangles élémentaires, nous souhaitons approcher f par $\pi_0^{[i]} \in \mathbb{R}_0[X]$ en un seul point dans chaque intervalle I_i. On note ce point ω_i, ainsi $\pi_0^{[i]}(p) = f(\omega_i)$, $\forall i \in \{0,\dots,n-1\}$, et :

$$I(f) = \int_a^b f(p)\,dp = \sum_{i=0}^{n-1} \int_{p_i}^{p_{i+1}} f(t)\,dt = \sum_{i=0}^{n-1} \int_{p_i}^{p_{i+1}} \pi_0^{[i]}(t)\,dt = \sum_{i=0}^{n-1} \int_{p_i}^{p_{i+1}} f(\omega_i)\,dt = \sum_{i=0}^{n-1} [p_i, p_{i+1}] f(\omega_i)$$

Maintenant que nous avons une méthode pour approcher $I(f)$, il ne nous reste plus qu'à déterminer ω_i et ω. Dans notre intérêt, nous nous intéressons uniquement à une seule formule de quadrature simple.

Formule des rectangles à gauche élémentaire

Comme son nom l'indique, nous allons prendre comme choix de ω la valeur à gauche de notre intervalle $[a,b]$, qui est a. Nous considérons le polynôme $\pi_0 \in \mathbb{R}_0[X]$ qui passe par le point a, tel que $\pi_0(p) = f(a)$, ainsi :

$$I(f) = \int_a^b f(p)\,dp = \int_a^b \pi_0(p)\,dp = \int_a^b f(a)\,dp = (b-a)f(a)$$

C'est sur cette méthode d'approximation d'intégrale que le fameux schéma numérique d'Euler explicite repose. Mais qu'est-ce qu'un schéma numérique ?

2.6 Théorie sur les schémas numériques

2.6.1 Schéma numérique pour les équations différentielles

En analyse numérique, un schéma numérique est une méthode discrète conçue pour approcher la solution d'un problème de Cauchy (**C**) comme celui-ci :

$$\begin{cases} \frac{dy}{dt} = f(t, y(t)), \quad t \in [t_0, T], \\ y(t_0) = y_0. \end{cases}$$

où :
— $y : [t_0, T] \to \mathbb{R}^d$ est la fonction inconnue à déterminer,
— $f : [t_0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ est une fonction donnée,
— $y_0 \in \mathbb{R}^d$ est la condition initiale.

En d'autres termes, l'objectif est d'obtenir une approximation numérique de la solution exacte $y(t)$ sur un maillage discret $(t_n)_{n\in\{0,\dots,N\}}$ de l'intervalle $[t_0, T]$, ainsi nous pouvons écrire la définition suivante.

Définition 24. Schéma numérique pour les équations différentielles
Un schéma numérique est un algorithme qui consiste à discrétiser l'intervalle $[t_0, T]$ en une suite de points $t_{ini} = t_0 < t_1 < \cdots < t_N = t_{ini} + T$ et à définir pour chaque sous-intervalle $[t_n, t_{n+1}]$ un pas de temps constant ou variable $\Delta t_n = t_{n+1} - t_n$. À l'étape $n \in \{0, \ldots, N-1\}$, on connaît $(y_k)_{k\in\{0,\ldots,n\}}$ et on peut calculer y_{n+1} à partir de $(y_k)_{k\in\{0,\ldots,n\}}$. Ainsi, on obtient une suite d'approximation $(y_n)_{n\in\{0,\ldots,N\}}$ telle que $y_n \approx y(t_n)$, où $y(t_n)$ est la solution exacte au temps t_n.

Parmi ces schémas numériques, puisqu'il en existe une infinité, une grande famille d'entre eux est celle des schémas numériques à un pas.

2.6.2 Schéma numérique à un pas

Ces schémas permettent de calculer à l'étape n $\in \{0, \ldots, N-1\}$ la valeur y_n à partir de y_{n-1}, ainsi on peut introduire le concept de fonction d'itération du schéma.

Définition 25. Fonction d'itération d'un schéma numérique
Soient $H > 0$ et $\Phi : \left[t_{\text{ini}}, t_{\text{ini}} + T\right] \times \mathbb{R}^d \times [0, H] \to \mathbb{R}^d$ continue avec d ≥ 1 et $y_0 \in \mathbb{R}^d$ tel que :

$$y_{n+1} = y_n + \Delta t_n \, \Phi(t_n, y_n, \Delta t_n) \quad n \in \{0, \ldots, N-1\}$$

On appelle fonction d'itération d'un schéma numérique la fonction Φ

De cette façon, nous pouvons définir un schéma numérique à un pas pour l'approximation de la solution du problème de Cauchy (**C**) de la manière suivante.

Définition 26. Schéma à un pas pour les équations différentielles
Un schéma numérique à un pas pour l'approximation de la solution de (**C**) est une relation de la forme :

$$(S) \quad \begin{cases} y_{n+1} = y_n + \Delta t_n \Phi(t_n, y_n, \Delta t_n), \quad n = 0, \ldots, N-1, \\ y_0 = y(t_0), \end{cases} \tag{2.1}$$

où $H > 0$ et $\Phi : \left[t_{\text{ini}}, t_{\text{ini}} + T\right] \times \mathbb{R}^d \times [0, H] \to \mathbb{R}^d$ est une fonction d'itération continue avec $d \geq 1$ et $y_0 \in \mathbb{R}^d$.

Après avoir défini la notion de schémas numériques qui permettent d'approcher la solution de (**C**), il est essentiel d'examiner les propriétés de ces méthodes. En effet, pour garantir que les approximations obtenues convergent vers la solution exacte, il faut que le schéma respecte certaines propriétés, comme la consistance et la stabilité, afin que les erreurs introduites à chaque étape ne se cumulent pas de manière incontrôlée. Détaillons donc ces notions fondamentales en analyse numérique.

2.6.3 Consistance

La consistance d'un schéma numérique à un pas peut être vue comme la rigidité d'un schéma. En d'autres termes, on peut voir la consistance comme la capacité de notre schéma à s'approcher de la solution de plus en plus à chaque étape. C'est-à-dire que la différence entre son estimation et la vraie solution devient de plus en plus petite quand le pas de temps diminue. Cela signifie que le calcul du schéma doit tendre vers l'équation d'origine quand Δt_n approche zéro. Ainsi, nous pouvons extraire une définition de notre explication :

Définition 27. Erreur locale de consistance
On appelle l'erreur locale de consistance la différence entre l'approximation donnée par la fonction d'itération et la vraie solution de y au temps t_{n+1}. Mathématiquement, nous obtenons l'expression suivante :

$$\varepsilon_n = y(t_{n+1}) - y(t_n) - h\Phi(t_n, y(t_n), \Delta t_n) \quad n \in \{0, \ldots, N-1\}.$$

$\Longleftrightarrow$

$$y(t_{n+1}) = y(t_n) - h\Phi(t_n, y(t_n), \Delta t_n) + \varepsilon_n \quad n \in \{0, \ldots, N-1\}.$$

Grâce à cette notion nous pouvons désormais caractériser la consistance.

Définition 28. Consistance
Soit $N \in \mathbb{N}^*$, on considère une suite de subdivisions $(S_N)_{N\in\mathbb{N}^*}$ de l'intervalle I, $\forall n \in \{0,\dots,N-1\}$. On dit que le schéma à un pas (S) est consistant pour l'équation différentielle $\frac{dy}{dt} = f(t,y)$ si pour toute solution de y, et pour toutes suites de subdivisions $(S_N)_{N\in\mathbb{N}^*}$, telles que $\Delta t_{max,N}$ et $\lim_{N\to+\infty} \Delta t_{max,N} = 0$, on a :

$$\lim_{N\to+\infty} \sum_{n=0}^{N-1} \|\varepsilon_n\| = 0.$$

où $\|.\|$ est la norme euclidienne de $\mathbb{R}^d$.

Proposition 9. Consistance et fonction d'itération
La méthode à un pas associée à Φ est consistante si et seulement si

$$\Phi(t,y,0) = f(t,y) \quad \text{pour tout } t \in [t_{\text{ini}}, t_{\text{ini}}+T] \text{ et pour tout } y \in \mathbb{R}^d.$$

Bien que la consistance assure que l'erreur locale tend vers zéro lorsque le pas de temps diminue, elle ne garantit pas à elle seule que l'erreur globale reste sous contrôle. En effet, même avec une erreur locale minime, de petites imprécisions peuvent s'accumuler à chaque pas de temps et provoquer une divergence de la solution numérique au bout d'un certain nombre d'itérations.

2.6.4 Stabilité

Afin de pouvoir manipuler cette incertitude, il existe une notion importante en analyse numérique, la stabilité. Elle permet de limiter l'amplification de ces erreurs à chaque étape de l'approximation.

Définition 29. Stabilité
On dit que le schéma explicite à un pas (S) est stable s'il existe un réel $M \geq 0$ tel que, étant données les perturbations $\varepsilon_1,\dots,\varepsilon_{N-1}$ et les suites finies $(y_n)_{n\in\{0,\dots,N\}}$ et $(z_n)_{n\in\{0,\dots,N\}}$ vérifiant :

$$(S) \quad \begin{cases} y_{n+1} = y_n + \Delta t_n \Phi(t_n, y_n, \Delta t_n), \quad n = 0,\dots,N-1, \\ y_0 = y(t_0), \end{cases} \tag{2.2}$$

et

$$z_{n+1} = z_n + \Delta t_n \Phi(t_n, z_n, \Delta t_n) + \varepsilon_n, \quad n \in \{0,\dots,N-1\}.$$

Alors, on a

$$\max_{0\leq n\leq N} \|y_n - z_n\| \leq M\Big(\|y_0 - z_0\| + \sum_{n=0}^{N-1} \|\varepsilon_n\|\Big).$$

Cela signifie que les erreurs d'approximation ne s'amplifient pas de manière incontrôlée et que M est indépendant du temps.

Proposition 10. Stabilité et lipschitzianité de la fonction d'itération
Soient $H > 0$ et $\Phi : [t_{\text{ini}}, t_{\text{ini}}+T] \times \mathbb{R}^d \times [0,H] \to \mathbb{R}^d$ une fonction continue et lipschitzienne par rapport à la variable d'état y sur $\mathbb{R}^d$, uniformément par rapport à la variable de temps $t \in I := [t_{\text{ini}}, t_{\text{ini}}+T]$ et au paramètre de discrétisation dans $[0,H]$ tel que :

$$\exists L > 0, \quad \text{telle que} \quad \forall t \in [t_{\text{ini}}, t_{\text{ini}}+T], \quad \forall \delta \in [0,H], \quad \forall (w,z) \in \mathbb{R}^d \times \mathbb{R}^d,$$

et

$$\|\Phi(t,w,\delta) - \Phi(t,z,\delta)\| \leq L\|w-z\|.$$

Alors la méthode est stable

Après avoir défini la consistance ainsi que la stabilité d'un schéma numérique, nous allons voir comment ces deux notions influent sur la convergence d'un schéma numérique à un pas.

2.6.5 Convergence

La consistance assure que l'erreur locale tend vers zéro lorsque le pas de temps décroît, tandis que la stabilité garantit que ces petites erreurs n'entraînent pas une amplification incontrôlée au fil des itérations. Nous commençons à nous douter que ces mécanismes ont un lien étroit avec la convergence, c'est ce que montre le théorème de Lax.

Théorème 8. Théorème de Lax
Si un schéma numérique est consistant et stable, alors il est convergent, c'est-à-dire que la solution approchée y_n tend vers la solution exacte $y(t_n)$ lorsque le pas de temps Δt tend vers zéro :

$$\lim_{\Delta t_n \to 0} \|y_n - y(t_n)\| = 0.$$

Corollaire 2. Conditions sur la fonction d'itération et convergence
La méthode à un pas converge si :

— $\Phi(t, y, 0) = f(t, y)$ *pour tout* $t \in [t_{ini}, t_{ini} + T]$ *et pour tout* $y \in \mathbb{R}^d$.

— $\Phi : [t_{ini}, t_{ini} + T] \times \mathbb{R}^d \times [0, H] \to \mathbb{R}^d$ *est une fonction continue et lipschitzienne par rapport à la variable d'état* y *sur* $\mathbb{R}^d$*, uniformément par rapport à la variable de temps* $t \in I := [t_{ini}, t_{ini} + T]$ *et au paramètre de discrétisation dans* $[0, H]$.

Une fois que nous savons ce qu'est la convergence d'un schéma numérique, nous pouvons maintenant quantifier cette convergence et étudier la vitesse à laquelle elle se produit.

2.6.6 Convergence à l'ordre p

C'est ici qu'intervient la notion d'**ordre de convergence**, qui mesure la rapidité avec laquelle l'erreur globale décroît lorsque le pas de discrétisation tend vers zéro. Un schéma d'ordre élevé permet d'obtenir une meilleure précision avec un pas de temps relativement grand, tandis qu'un schéma d'ordre faible nécessite une discrétisation plus fine pour garantir une erreur acceptable.

Définition 30. Consistance à l'ordre p
Soit $p \in \mathbb{N}^*$. Une méthode à un pas est dite consistante à l'ordre p pour l'équation différentielle $\frac{dy}{dt} = f(t, y)$ si, pour toute solution de cette dernière, il existe une constante C telle que, pour toute suite de subdivisions $(S_N)_{N \in \mathbb{N}}$, l'erreur de consistance ε_n vérifie pour tout $n \in \{0, \ldots, N-1\}$

$$\|\varepsilon_n\| \leq C \, (\Delta t_n)^{p+1}.$$

Ainsi en s'inspirant du Théorème de Lax, nous obtenons la définition suivante.

Définition 31. Convergence à l'ordre p
Soit $p \in \mathbb{N}^*$. Une méthode à un pas est dite convergente à l'ordre p pour l'équation différentielle $\frac{dy}{dt} = f(t, y)$ si, le schéma est stable et consistant à l'ordre p.

Désormais, nous avons les bases et les concepts théoriques de l'approximation numérique. Ces derniers nous offrent les outils indispensables pour analyser et comprendre le comportement des méthodes numériques lorsqu'elles doivent approcher la solution exacte d'un problème de Cauchy. Nous utiliserons ces outils et concepts pour analyser différents schémas numériques à un pas concrets et très utilisés qui sont la méthode d'Euler explicite, la méthode d'Euler implicite et le schéma d'Euler modifié, connu sous le nom de schéma à point milieu. Nous étudierons leurs formulations, leurs propriétés respectives ainsi que leurs performances en termes de convergence.

Chapitre 3

Analyse et comparaison de schémas numériques à un pas

Nous nous permettons de rappeler la définition d'un problème de Cauchy.

Définition 32. Problème de Cauchy
Soit $(t_0, y_0) \in \mathbb{O})$. Le problème de Cauchy avec donnée initiale (t_0, y_0) consiste à déterminer la ou les solutions y de l'équation différentielle sur un intervalle I tel que

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

où :
- $y : [t_0, T] \to \mathbb{R}^d$ est la fonction inconnue à déterminer,
- $f : [t_0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ est une fonction donnée,
- $y_0 \in \mathbb{R}^d$ est la condition initiale.

Il existe plusieurs manières de résoudre un problème de Cauchy. Ces méthodes utilisent différentes approches de résolution en manipulant l'intervalle sur lequel notre équation différentielle est définie. L'une des approches les plus simples est de subdiviser l'intervalle temporel $[t_0, T]$ en sous-intervalles de taille constante Δt_n. Afin de résoudre le problème de Cauchy mentionné, nous devons adopter une approche efficace et intuitive. Comme nous travaillons avec une équation différentielle, nous pouvons penser aux dérivées et aux pentes, et nous remémorer la définition des accroissements finis.

Définition 33. Accroissements finis
Soient une fonction f définie sur un intervalle $[a, b]$ et deux points x et $x + \alpha$ appartenant à cet intervalle. L'accroissement de la fonction f sur cet intervalle est donné par :

$$\Delta f = f(x + \alpha) - f(x).$$

Le rapport des accroissements finis est défini comme suit :

$$\frac{\Delta f}{\alpha} = \frac{f(x + h) - f(x)}{\alpha}.$$

Lorsque $\alpha \to 0$, ce rapport d'accroissements finis converge vers la dérivée de f au point x, soit :

$$\lim_{\alpha \to 0} \frac{\Delta f}{\alpha} = f'(x).$$

Ainsi, la dérivée de f au point x représente le taux de variation instantané de la fonction.

Les équations différentielles décrivent l'évolution de systèmes à travers des relations entre les dérivées d'une fonction inconnue et cette fonction elle-même. Pour résoudre numériquement une équation différentielle, il est courant de remplacer la dérivée par une approximation utilisant les accroissements finis. Cette approche donne naissance aux méthodes des différences finies.

Définition 34. Différences finies
Les différences finies sont une méthode numérique utilisée pour approximer les dérivées de fonctions à partir de valeurs discrètes. Plus précisément, pour une fonction $y : t \mapsto y(t)$ dérivable, on définit le quotient suivant :

$$\Delta_h y(t) = \frac{y(t+h) - y(t)}{h},$$

En se donnant une subdivision de l'intervalle défini dans le problème de Cauchy, et en utilisant les notations décrites dans le Chapitre 2, on remarque que :

$$\Delta_h y(t) = \frac{y(t_{k+1}) - y(t_k)}{h} \approx f(t_k, y(t_k))$$

Ainsi nous obtenons par approximation $y(t_k) \approx y_k$:

$$(\Xi) \quad y(t_{k+1}) \approx y_k + \Delta t_k f(t_k, y_k) = y_{k+1}$$

Puisque le problème de Cauchy nous donne directement la condition initiale y_0, il est possible, en appliquant la relation (Ξ), de déterminer successivement les valeurs $y(t_k)$. Cette approche, connue sous le nom de méthode d'Euler explicite à un pas, permet d'obtenir de manière itérative des approximations de y sur le maillage $(t_k)_{k\in\mathbb{N}^*}$, ce qui résout notre problème de Cauchy.

Cependant, une méthode plus intuitive pour retrouver le schéma d'Euler explicite à un pas est d'utiliser une formule de quadrature élémentaire, celle des rectangles à gauche. Comme nous savons que

$$\frac{dy}{dt} = f(t, y(t))$$

et par la formulation intégrale du problème de Cauchy $\forall t \in I$:

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

nous souhaitons trouver la solution sur un intervalle I donné par le problème de Cauchy, en utilisant nos outils d'approximation numérique comme la subdivision d'intervalle et l'approximation d'intégrales. Ainsi en subdivisant l'intervalle I = [t_0,t] en une suite (N+1) points $(t_k)_{k\in\{0,\dots,N\}}$, nous allons résoudre l'équation différentielle sur chaque intervalle de temps $[t_n, t_{n+1}]$ $\forall n \in \{0, ..., N-1\}$:

$$y(t) = y_0 + \int_{t_0}^{t} f(s, y(s))\, ds = y_0 + \sum_{n=0}^{N-1} \int_{t_n}^{t_{n+1}} f(t, y(t))\, dt$$

Ainsi :

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(x, y(x))\, dx$$

En approchant cette intégrale par la gauche, nous obtenons l'approximation de $y(t_{n+1})$:

$$y_{n+1} = y_n + (t_{n+1} - t_n) f(t_n, y_n)$$

où $\forall n \in \{0, ..., N-1\}$ y_n est une approximation de $y(t_n)$.

Maintenant que nous avons une idée de la manière d'approcher la solution d'un problème de Cauchy de façon algorithmique et itérative, comme la méthode d'Euler explicite, étudions quelques schémas.

3.1 Schémas d'Euler à un pas

3.1.1 Schéma d'Euler Explicite

Le schéma d'Euler explicite pour l'approximation de la solution du problème de Cauchy (C) est donné par la relation de récurrence :

$$(S_E) \quad \begin{cases} y_{n+1} = y_n + \Delta t_n f(t_n, x_n) \quad n \in \{0, ..., N-1\} \\ y_0 = y(t_0) \end{cases} \tag{3.1}$$

Implémentation Python :

```python
import numpy as np

def euler_explicit(f, t0, T, y0, N):
    dt = (T - t0) / N
    t_val = np.linspace(t0, T, N+1)
    y_val = np.zeros((N+1, len(x0)))
    y_val[0] = y0

    for n in range(N):
        y_val[n+1] = y_val[n] + dt * f(t_val[n], y_val[n])

    return t_val, y_val
```

Listing 3.1 – Schéma d'Euler explicite

Convergence

Pour étudier la convergence du schéma d'Euler explicite, considérons le schéma (S_E) appliquée au problème de Cauchy (C). Supposons que la solution exacte $y(t)$ est de classe C^2 sur l'intervalle $[t_0$,T] et que la fonction f est lipschitzienne en y. Un développement de Taylor-Lagrange au premier ordre autour de t_n donne, pour un certain $\xi_n \in [t_n, t_{n+1}]$,

$$y(t_{n+1}) = y(t_n) + \Delta t_n \, f\big(t_n, y(t_n)\big) + \frac{(\Delta t_n)^2}{2} \, y''(\xi_n).$$

L'erreur locale de troncature est alors définie par :

$$\varepsilon_n = y(t_{n+1}) - y(t_n) - \Delta t_n \, f\big(t_n, y(t_n)\big) = \frac{(\Delta t_n)^2}{2} \, y''(\xi_n) \implies \|\varepsilon_n\| = O((\Delta t_n)^2) \iff \|\varepsilon_n\| \leq C(\Delta t_n)^2$$

Avec une constante C ≥ 0 indépendante du temps. Cette erreur locale se cumule à chaque pas de temps, calculons la en posant $e_n = \|y(t_n) - y_n\|$:

$$\begin{aligned}
e_{n+1} &= y(t_{n+1}) - y_{n+1} \\
&= \Big[y(t_n) + \Delta t_n \, f\big(t_n, y(t_n)\big) + \varepsilon_n\Big] - \Big[y_n + \Delta t_n \, f\big(t_n, y_n\big)\Big] \\
&= \Big[y(t_n) - y_n\Big] + \Delta t_n \Big[f\big(t_n, y(t_n)\big) - f\big(t_n, y_n\big)\Big] + \varepsilon_n \\
&= e_n + \Delta t_n \Big[f\big(t_n, y(t_n)\big) - f\big(t_n, y_n\big)\Big] + \varepsilon_n. \\
&\leq \|e_n\| + \Delta t_n \, \|f(t_n, y(t_n)) - f(t_n, y_n)\| + \|\varepsilon_n\|
\end{aligned}$$

Puisque f est supposée Lipschitzienne par rapport à y, il existe une constante $L > 0$ telle que :

$$\|f\big(t_n, y(t_n)\big) - f\big(t_n, y_n\big)\| \leq L \, \|y(t_n) - y_n\| = L \, \|e_n\|.$$

Ainsi, on obtient :

$$\|e_{n+1}\| \leq \|e_n\| + \Delta t_n \, L \, \|e_n\| + \|\varepsilon_n\| = (1 + hL) \, \|e_n\| + \|\varepsilon_n\|.$$

Comme $\|\varepsilon_n\| \leq C \, (\Delta t_n)^2$, alors :

$$\|e_{n+1}\| \leq (1 + hL) \, \|e_n\| + C \, (\Delta t_n)^2.$$

Cette inégalité montre que l'erreur globale est constituée d'un terme d'amplification $(1+\Delta t_n L)$ de l'erreur au temps précédent et d'un terme d'erreur local d'ordre $(\Delta t_n)^2$. De plus, nous remarquons que notre inégalité est de la forme suivante :

$$E_{n+1} \leq (1 + hL) E_n + \beta$$

En itérant cette relation, et grâce à l'inégalité de Gronwall discret, on obtient $\forall n$:

$$E_n \leq C(\Delta t_n)^2 \sum_{k=0}^{n-1} (1 + hL)^{n-1-k}$$

En posant j = n-1-k, et en remarquant que c'est une somme géométrique :

$$E_n \leq C(\Delta t_n)^2 \frac{(1+\Delta t_n L)^n - 1}{\Delta t_n L}$$

Comme n ≤ N et $N\Delta t_n \leq T$:

$$(1+\Delta t_n L)^n - 1 \leq (1+\Delta t_n L)^N - 1 = \exp(Nln(1+\Delta t_n L)) \leq exp(LT)$$

En regroupant tous les termes nous obtenons :

$$\|e_n\| \leq \frac{C\,(\Delta t_n)^2}{\Delta t_n L}\left[(1+\Delta t_n L)^n - 1\right] = \frac{C\Delta t_n}{L}\left[(1+\Delta t_n L)^n - 1\right] \leq \frac{C\Delta t_n}{L}(exp(LT)-1)$$

En prenant le maximum sur tous les n tels que $t_n \in [t_0, T]$:

$$\max_{0\leq n\leq N} \|y(t_n) - y_n\| \leq K(\Delta t_n)^1 \quad K = \frac{C}{L}(exp(LT)-1)$$

Ainsi, le schéma d'Euler explicite converge vers la solution exacte avec un ordre de convergence global égal à 1.

3.1.2 Schéma d'Euler Implicite

Le schéma d'Euler implicite est défini par :

$$(S_I) \quad \begin{cases} y_{n+1} = y_n + \Delta t_n f(t_{n+1}, y_{n+1}) & n \in \{0, ..., N-1\} \\ y_0 = y(t_0) \end{cases} \tag{3.2}$$

Implémentation Python :

```python
from scipy.optimize import fsolve

def euler_implicit(f, t0, T, y0, N):
    dt = (T - t0) / N
    t_val = np.linspace(t0, T, N+1)
    y_val = np.zeros((N+1, len(y0)))
    x_val[0] = y0

    for n in range(N):
        g = lambda y_next: y_next - x_val[n] - dt * f(t_val[n+1], y_next)
        y_val[n+1] = fsolve(g, y_val[n])

    return t_val, y_val
```

Listing 3.2 – Schéma d'Euler implicite

3.1.3 Schéma d'Euler Modifié

Le schéma d'Euler modifié inspiré de la méthode de quadrature du point milieu est défini par :

$$(S_M) \quad \begin{cases} y_{n+1} = y_n + hf((\frac{t_n+t_{n+1}}{2}), y(\frac{t_n+t_{n+1}}{2}), & n \in \{0, ..., N-1\} \\ y_0 = y(t_0) \end{cases} \tag{3.3}$$

En utilisant le développement de Taylor, nous pouvons écrire :

$$y(\frac{t_n+t_{n+1}}{2}) = y(t_n) + \frac{\Delta t_n}{2} y'(t_n) + o(\frac{\Delta t_n}{2})$$

Ainsi :

$$(S_M) \quad \begin{cases} y_{n+1} = y_n + hf((\frac{t_n+t_{n+1}}{2}), y(t_n) + \frac{\Delta t_n}{2} f(t_n, y_n)) & n \in \{0, ..., N-1\} \\ y_0 = y(t_0) \end{cases} \tag{3.4}$$

Implémentation Python :

```python
import numpy as np

def PointMilieu(x0, f, T, N):
    dt = T / (N - 1)
    t = np.linspace(0, T, N)
    y = np.zeros(N)
    y[0] = y0
    for k in range(1, N):
        tk = t[k-1] + dt / 2
        yk = y[k-1] + (dt / 2) * f(t[k-1], y[k-1])
        y[k] = y[k-1] + dt * f(tk, yk)
        return t,y
```

Listing 3.3 – Schéma d'Euler Modifié

3.2 Comparaison des performances des méthodes

Il est naturel de se demander quelle méthode est la plus efficace et quelle est la moins coûteuse computationnellement. Comme ce sont des méthodes d'approximation, il existe une erreur, et c'est cette dernière qui nous aidera à déterminer les performances des méthodes étudiées.

L'importance de l'erreur dans les méthodes d'approximation des équations différentielles réside dans la précision et la stabilité des solutions obtenues. Une erreur plus faible indique une solution plus précise et une méthode plus fiable. Comparer les méthodes en fonction de leurs erreurs permet de mieux comprendre leurs avantages et inconvénients, ainsi que leur adéquation à différents types de problèmes. Considérons l'équation différentielle suivante :

$$f(t, y) = -4y$$

Nous allons comparer les différentes solutions données par nos trois méthodes d'approximation numérique. Pour calculer l'erreur, dont nous avons vu la définition précédemment, nous pouvons coder la fonction suivante :

```python
import numpy as np
def calculer_erreurs(approx, exacte):
    erreur_absolue = np.abs(approx - exacte)
    erreur_relative = erreur_absolue / np.abs(exacte)
    return erreur_absolue, erreur_relative
```

Listing 3.4 – Programme de calcul de l'erreur

Définissons nos paramètres :

```
itermax = 20
x0 = 1
T = 8
N = 20
epsabs = 10e-18
epsrel = math.sqrt(10e-18)
```

En utilisant nos trois codes des trois méthodes précédentes, comparons les résultats d'approximation de la solution de $f(t, y) = -4y$

```python
# On recupere les sorties de nos fonctions en fonction de nos parametres
(t, xEulerExplicite) = EulerExplicite(x0, f, T, N)
(t, xEulerImplicite) = EulerImplicite(x0, f, T, N)
(t, xPointMilieu) = PointMilieu(x0, f, T, N)
Exacte = x0 * np.exp(-4 * t)

# Calcul des erreurs
erreur_absolue_EulerExplicite, erreur_relative_EulerExplicite = calculer_erreurs(
    xEulerExplicite, Exacte)
erreur_absolue_EulerImplicite, erreur_relative_EulerImplicite = calculer_erreurs(
    xEulerImplicite, Exacte)
erreur_absolue_PointMilieu, erreur_relative_PointMilieu = calculer_erreurs(
    xPointMilieu, Exacte)
```

Listing 3.5 – Calcul de l'erreur

En mettant en forme les différents résultats obtenus (erreurs, stabilité et précisions) dans un tableau et dans des graphes, nous obtenons les représentations suivantes :

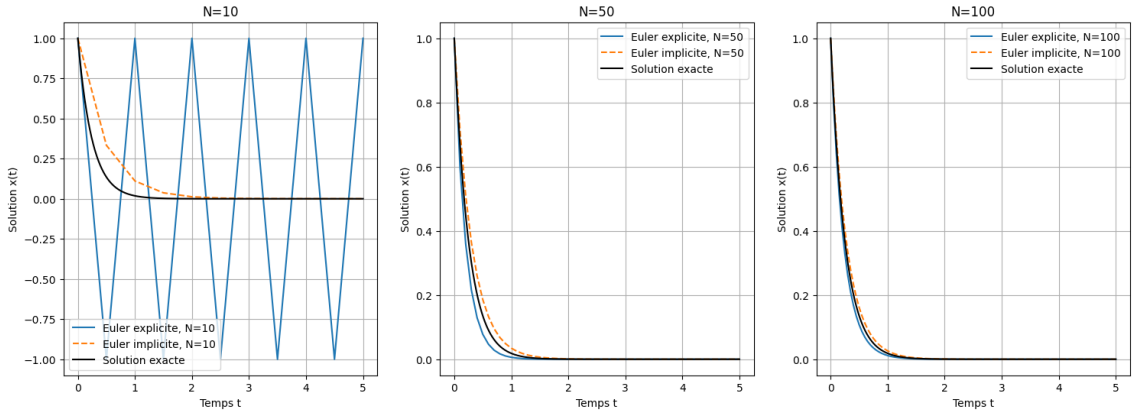


FIGURE 3.1 – Stabilité & précisions des solutions d'Euler Explicite & Implicite selon différents pas de temps

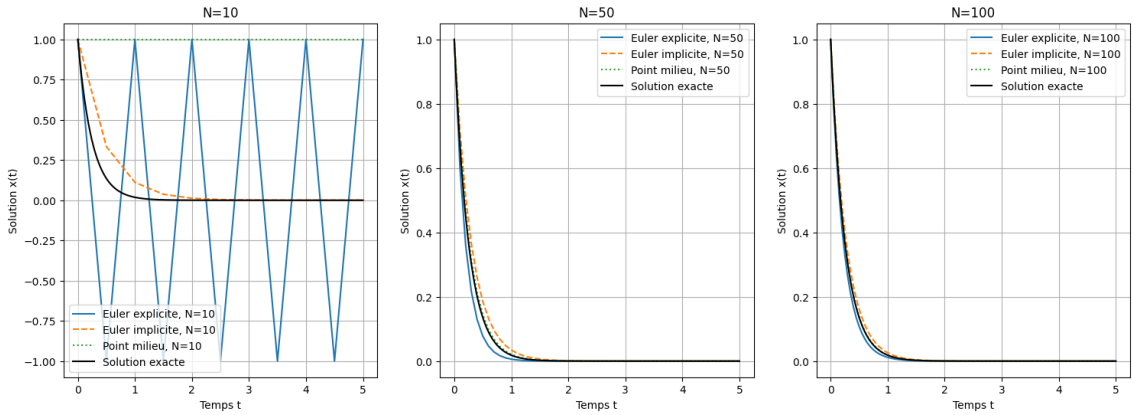


FIGURE 3.2 – Stabilité & précisions des solutions des 3 méthodes

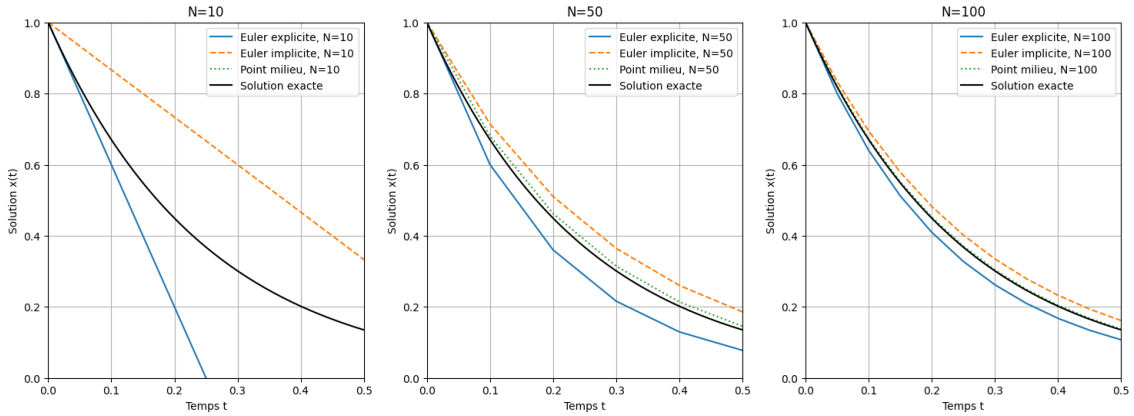


FIGURE 3.3 – Stabilité & précisions des solutions des 3 méthodes avec un zoom

Table 3.1 – Erreurs et rapports des méthodes d'Euler explicite, implicite, et point milieu.

Méthode	Ni	Erreur Abs.	Rapport	Rapport Base 2
Euler Explicite	320	1.93×10^{-2}		
Euler Implicite	320	1.77×10^{-2}		
Point Milieu	320	6.66×10^{-4}		
Euler Explicite	319	1.93×10^{-2}	0.997	-0.0046
Euler Implicite	319	1.78×10^{-2}	0.997	-0.0044
Point Milieu	319	6.70×10^{-4}	0.994	-0.0094
Euler Explicite	318	1.94×10^{-2}	0.997	-0.0046
Euler Implicite	318	1.78×10^{-2}	0.997	-0.0044
Point Milieu	318	6.75×10^{-4}	0.994	-0.0094

En examinant les résultats obtenus des trois méthodes numériques : Euler explicite, Euler implicite et Point milieu, nous pouvons conclure plusieurs choses sur leur efficacité, leur stabilité et leurs erreurs.

A) Stabilité et précision

La méthode d'Euler explicite est simple et peut être rapidement implémentée. Cependant, sa précision peut être limitée, surtout lorsqu'on utilise des pas de temps plus grands. Il peut être nécessaire de choisir des valeurs plus petites de N (nombre de points) pour obtenir des résultats satisfaisants, ce qui augmente le nombre de calculs et peut être lourd computationnellement.

En ce qui concerne la méthode d'Euler implicite, elle est plus stable et permet l'utilisation de pas de temps plus grands sans nuire à la précision. En effet, la valeur de la solution au prochain pas de temps est calculée en résolvant une équation impliquant la valeur de cette solution. En d'autres termes, la méthode d'Euler implicite prend en compte la dépendance de la solution future et assure ainsi une plus grande stabilité. En contrepartie, à chaque étape, la méthode résout des équations non linéaires, ce qui amplifie le temps de calcul.

Enfin, la méthode du point milieu peut être vue comme un bon compromis entre précision et stabilité. Elle est réputée pour sa facilité d'implémentation, plus simple que l'Euler implicite, et plus précise que l'Euler explicite. Cependant, cette méthode perd sa stabilité et sa précision pour les équations différentielles raides ou pour des pas de temps trop grands. Son ordre minimal, comparé aux méthodes d'ordre supérieur comme la méthode Runge-Kutta, lui fait aussi défaut en termes de précision. Cependant, nous n'allons pas nous attarder sur ces comparaisons qui ne sont pas le sujet de notre mémoire.

B) Erreurs

Les erreurs absolues obtenues avec les trois méthodes sont relativement faibles, ce qui indique une bonne précision globale. La méthode du point milieu génère les erreurs absolues les plus faibles. Cela s'explique par le fait que cette méthode utilise une approximation plus précise de la solution en prenant en compte la pente au point intermédiaire, ce qui améliore la précision du résultat final.

En ce qui concerne les erreurs relatives, elles suivent la même tendance que l'erreur absolue. La méthode du point milieu affiche les erreurs relatives les plus faibles. Cela confirme encore une fois sa meilleure précision, car elle minimise les écarts proportionnels par rapport à la solution réelle. En prenant en compte la pente intermédiaire, cette méthode réduit les erreurs accumulées à chaque pas de temps.

C) Conclusion

En conclusion, la méthode du point milieu semble être la plus efficace en termes de précision (erreurs absolues et relatives moindres) et offre une bonne stabilité. L'Euler explicite, bien que simple et rapide, peut nécessiter des pas de temps plus petits pour garantir précision et stabilité. Enfin, l'Euler implicite, bien que plus stable pour des pas de temps plus grands, peut complexifier le calcul en raison de la résolution des équations non linéaires à chaque étape.

La comparaison de ces méthodes montre que chaque méthode a ses propres avantages et inconvénients selon le contexte d'utilisation. Par exemple, l'Euler explicite peut être privilégié pour des calculs rapides avec des pas de temps adaptés, tandis que l'Euler implicite peut être choisi pour des problèmes nécessitant une grande stabilité.

Nous avons analysé et comparé de manière détaillée différentes méthodes à un pas classiques en évaluant leurs performances sur une équation différentielle ordinaire. Nous avons ainsi mis en évidence leurs comportements en termes d'erreur globale, de stabilité et de précision. Bien que ces méthodes soient simples et efficaces pour de nombreux problèmes, elles montrent parfois leurs limites, surtout lorsqu'il s'agit d'équations plus complexes ou raides. C'est dans ce contexte que le prochain chapitre introduit la théorie de l'intégration projective. Cette approche offre une alternative à l'intégration classique, en proposant une méthode qui s'adapte aux dynamiques spécifiques de l'équation.

Chapitre 4

Introduction théorique à l'intégration projective numérique

Les systèmes d'équations différentielles jouent un rôle essentiel dans les mathématiques appliquées et la physique, en permettant de modéliser de nombreux phénomènes naturels et technologiques, qu'il s'agisse de la dynamique des fluides, des circuits électriques ou des modèles économiques. Certains de ces modèles se caractérisent par des équations différentielles raides et des comportements évoluant sur plusieurs échelles temporelles, rendant leur résolution plus complexe avec des méthodes numériques traditionnelles.

Dans ce contexte, la méthode d'intégration projective, qui repose sur une reformulation des approximations intégrales par projection, permet une meilleure stabilité et gestion des erreurs dans certains cas difficiles. Cette méthode en deux temps se révèle particulièrement avantageuse pour traiter des équations pour lesquelles les schémas à un pas classiques peinent à offrir une précision suffisante, s'imposant ainsi comme une solution efficace et complémentaire aux méthodes classiques étudiées au chapitre précédent.

Définition 35. Système raide
Un système est dit raide lorsqu'il contient des échelles de temps très différentes, c'est-à-dire des termes qui évoluent beaucoup plus rapidement que d'autres. De ce fait, on observe deux comportements distincts : d'un côté, une phase de transition très rapide qui disparaît presque immédiatement, et de l'autre, une évolution plus lente qui joue un rôle déterminant dans la progression globale du système sur le long terme.

Pour illustrer ce concept que l'on appellera "multiscale", on peut prendre un exemple très courant en mécanique des fluides : la turbulence. Les turbulences peuvent contenir des vortex de tailles très différentes ; certains disparaissent immédiatement en appliquant un impact considérable à un moment donné, tandis que d'autres persistent et influencent le comportement global sur la longue durée.

4.1 L'intégration projective numérique

L'intégration projective est une méthode numérique conçue pour traiter efficacement des systèmes présentant une raideur ou, plus généralement, des comportements multiscales. Afin d'obtenir une approximation de la solution de ces systèmes, on utilise des méthodes numériques comme celles vues précédemment. Cependant, les méthodes classiques d'intégration, qu'elles soient explicites ou implicites, doivent utiliser des pas de temps très réduits pour capturer ces comportements immédiats. Cela entraîne un coût computationnel énorme lorsqu'il s'agit d'étudier la dynamique sur de longues durées. L'intégration projective répond à ce défi en exploitant la séparation des échelles pour combiner deux phases d'intégration, chacune adaptée à une échelle temporelle spécifique.

4.1.1 Idée de l'intégration projective

L'idée principale de l'intégration projective est de séparer les composants rapides et lents du système et de les traiter différemment. Dans de nombreux problèmes raides, les phénomènes qui se produisent sur un laps de temps rapide convergent rapidement vers un phénomène qui dure sur un laps de temps considérable. De ce fait, après une courte phase d'intégration à l'aide de pas de

temps extrêmement petits, ce qu'on appellera **micro**, la solution atteint l'état lent du système. Ensuite, nous extrapolons la solution avec une projection afin de parcourir des intervalles de temps plus grands, tout en nous basant sur les observations du comportement en pas micro. En somme, l'idée est donc de décomposer l'intégration en deux phases distinctes :

1. **Phase micro** : des pas de temps très courts sont utilisés pour résoudre le système avec une méthode explicite (par exemple Euler explicite), dans le but de capturer précisément les effets des dynamiques rapides.
2. **Phase macro (projection)** : l'information obtenue pendant la phase micro est ensuite exploitée pour effectuer une extrapolation de la solution sur un intervalle de temps plus large.

Maintenant que nous avons le concept en tête, nous pouvons entrer dans les détails en expliquant l'intérêt de chaque étape de l'intégration projective.

4.1.2 L'étape micro

Cette phase consiste à intégrer le système sur une courte durée avec un pas de temps très réduit, noté h_{micro}, anciennement $\Delta t_{micro,n}$. Durant cette période, l'approximation numérique doit être suffisamment précise pour capturer la décroissance rapide des modes instables ou non pertinents à l'échelle longue. On suppose que la résolution de l'équation différentielle du problème s'écrit sous la forme :

$$\frac{dy}{dt} = f(y,t).$$

À partir d'une condition initiale $y(t_0) = y_0$, un schéma d'intégration explicite, ici la méthode d'Euler explicite, est appliqué de manière itérative pour obtenir une suite de points $y(t_0 + j\, h_{\text{micro}})$ pour $j = 1, 2, \ldots, K$, où K désigne le nombre de pas micro effectués avant de réaliser la projection. Ainsi, pour chaque pas, on écrit approximativement :

$$y_{j+1} \approx y_j + h_{\text{micro}}\, f(y_j, t_j).$$

L'objectif principal durant cette phase est d'exprimer la dynamique rapide afin d'obtenir une estimation correcte de la tendance de la solution. De plus, dans certains algorithmes, un pas supplémentaire peut être réalisé pour affiner l'estimation de la dérivée (la pente), avec une démarche qui imite une différence finie. De plus, le choix du nombre K et de h_{micro} est crucial, ils doivent être suffisamment petits pour assurer la stabilité locale de l'intégrateur explicite, mais aussi assez grands pour observer la disparition rapide des dynamiques superflues.

4.1.3 L'étape macro

Une fois l'étape micro terminée, la solution a convergé vers une dynamique lente. Maintenant une estimation de la dérivée moyenne est utilisée pour extrapoler la solution sur un intervalle de temps plus long, noté T_{macro}. Supposons que l'on dispose de deux estimations successives obtenues pendant la phase micro, notées $y_{\text{micro},K}$ et $y_{\text{micro},K+1}$, de sorte que l'on puisse définir une pente approximative par :

$$\text{pente} \approx \frac{y_{\text{micro},K+1} - y_{\text{micro},K}}{h_{\text{micro}}}.$$

La solution extrapolée est alors obtenue par :

$$y_{\text{proj}} = y_{\text{micro},K+1} + \Big(\Delta t - (K+1)h_{\text{micro}}\Big) \times \text{pente} \quad \text{où } \Delta t_n = t_{n+1} - t_n$$

Nous arrivons à cette formule car la dérivée estimée est représentative de l'évolution de la solution sur l'intervalle macro, ce que l'on peut se permettre car les dynamiques rapides sont déjà observées et utilisées. Ainsi, nous pouvons extrapoler en avant tout en réduisant le temps de calcul global. La réussite de cette projection dépend à la fois de la justesse de l'estimation de la pente et de la cohérence entre les différentes dynamiques.

4.2 Consistance

Pour cette méthode hybride, cette erreur est composée de deux parties.

1. **L'erreur de l'étape micro :**
   Pendant cette phase, on utilise un schéma explicite d'ordre faible, Euler explicite, avec un pas de temps h_{micro}. Pour l'équation différentielle

   $$\frac{dy}{dt} = f(y,t),$$

   la méthode d'Euler explicite donne, pour un pas de temps h_{micro} :

   $$y(t + h_{\text{micro}}) = y(t) + h_{\text{micro}}\, f(y(t),t) + O\big(h_{\text{micro}}^2\big).$$

   Ainsi, d'après les calculs faits dans le chapitre précédent, l'erreur locale dans cette phase est de l'ordre de h_{micro}, .
2. **L'erreur de la projection, la phase macro :**
   La solution est extrapolée à partir d'une estimation de la pente obtenue lors de la phase micro. Si l'estimation de la dérivée est prise sur une durée h_{micro} par différence finie et que l'on projette ensuite sur une durée $\Delta t - (K+1)h_{\text{micro}}$, l'erreur reste de l'ordre de

   $$O\big(\Delta t - (K+1)h_{\text{micro}}\big),$$

En combinant les deux erreurs, la méthode est dite consistante si, lorsque $h_{\text{micro}} \to 0$ et $\Delta t \to 0$, en respectant une relation de proportionnalité avec h_{micro}, l'erreur locale tend vers zéro. Autrement dit, le schéma approchera alors la solution exacte de l'équation différentielle.

4.3 Stabilité

Pour étudier la stabilité, il est courant d'examiner le comportement de la méthode sur l'équation de test linéaire

$$f(t,y) = \frac{dy}{dt} = \lambda y,$$

où $\lambda \in \mathbb{C}$ avec $\Re(\lambda) < 0$ pour les modes raides.

4.3.1 Stabilité de la phase Micro avec l'intégration par Euler explicite

En utilisant le schéma d'Euler, la solution après un pas h_{micro} est donnée par :

$$y_{j+1} = y_j + h_{\text{micro}}\,\lambda\, y_j = y_j\Big(1 + h_{\text{micro}}\lambda\Big).$$

Pour que l'intégration par Euler soit stable pour la phase micro, il faut que le facteur multiplicatif ait un module inférieur à 1, nous assurons que toute perturbation ou erreur d'arrondi diminue au fil des étapes :

$$\Big|1 + h_{\text{micro}}\lambda\Big| < 1.$$

Dans notre cas où $\lambda \in \mathbb{C}$ avec $\Re(\lambda) < 0$:

$$h_{\text{micro}} < \frac{2}{|\lambda|}.$$

Cette condition permet de maintenir la solution approximative proche de la solution exacte et d'éviter que les erreurs ne s'accumulent de manière incontrôlée, surtout dans le cas des systèmes raides. Dans les cas raides, on choisit généralement h_{micro} encore plus petit afin de garantir un amortissement rapide des modes instables.

4.3.2 Stabilité de la phase d'extrapolation, la phase macro

Après avoir effectué K pas micro, on obtient par récurrence directe :

$$y_K = y_0\Big(1 + h_{\text{micro}}\lambda\Big)^K.$$

Pour améliorer l'estimation de la dérivée, un pas micro supplémentaire est réalisé :

$$y_{K+1} = y_K + h_{\text{micro}}\,\lambda\,y_K = y_K\Big(1 + h_{\text{micro}}\lambda\Big).$$

Ansi on obtient par les différences divisés la pente :

$$\text{pente} \approx \frac{y_{K+1} - y_K}{h_{\text{micro}}} = \lambda y_K.$$

Par projection :

$$y_{\text{proj}} = y_{K+1} + \Big(\Delta t - (K+1)h_{\text{micro}}\Big)\,\lambda\,y_K.$$

En substituant $y_{K+1} = y_K\Big(1 + h_{\text{micro}}\lambda\Big)$, on obtient :

$$y_{\text{proj}} = y_K\Big[1 + h_{\text{micro}}\lambda + \lambda\Big(\Delta t - (K+1)h_{\text{micro}}\Big)\Big] = y_K\Big[1 + \lambda\Big(\Delta t - Kh_{\text{micro}}\Big)\Big].$$

Puisque $y_K = y_0\Big(1 + h_{\text{micro}}\lambda\Big)^K$ on obtient alors le gain :

$$G = \Big(1 + h_{\text{micro}}\lambda\Big)^K\Big[1 + \lambda\Big(\Delta t - Kh_{\text{micro}}\Big)\Big].$$

où G est le facteur par lequel la solution est multipliée lors d'une phase macro complète.

4.3.3 Conditions de stabilité globale

La stabilité de l'intégration projective repose donc sur :

1. **Stabilité locale phase micro :**
   Le choix de h_{micro} doit être suffisamment petit afin que le critère suivant soit respecté :

$$\Big|1 + h_{\text{micro}}\lambda\Big| < 1 \iff h_{\text{micro}} < \frac{2}{|\lambda|}.$$

2. **Stabilité de la projection :**
   Il est nécessaire que K soit choisi de manière que :

$$\Big|\Big(1 + h_{\text{micro}}\lambda\Big)^K\Big| < 1,$$

   De sorte que les erreurs potentielles ne soit pas amplifiées même après une valeur trop grande de T_{macro} :

$$|G| = |\Big(1 + h_{\text{micro}}\lambda\Big)^K\Big[1 + \lambda\Big(\Delta t - Kh_{\text{micro}}\Big)\Big]| < 1.$$

La stabilité et la consistance de la méthode d'intégration projective reposent sur le choix judicieux des paramètres h_{micro}, K et Δt. Pour que la phase de projection se réalise sans provoquer une amplification excessive des erreurs, il est important de choisir un pas micro qui reste entièrement dans la zone de stabilité de l'intégrateur explicite. De plus, en sélectionnant judicieusement le nombre de sous-pas K de façon à rendre les dynamiques rapides insignifiantes toutes proportions gardées, on obtient une transition fluide et progressive vers la phase de projection.

D'un autre côté, la consistance de la méthode est assurée puisque les erreurs locales, qu'elles proviennent de la phase micro ou de l'extrapolation, tendent vers zéro lorsque h_{micro} et Δt deviennent très petits. En conclusion, la fiabilité de la méthode est garantie dans un contexte où il existe une nette séparation entre les dynamiques rapides et lentes, ce qui est une condition essentielle pour traiter efficacement des systèmes raides.

Maintenant que nous connaissons le principe de l'intégration projective et les différentes conditions à respecter pour assurer la fiabilité de la méthode, nous devons l'implémenter numériquement.

Chapitre 5

Implémentation de l'intégration projective numérique

L'implémentation de cette méthode consiste à traiter les différentes dynamiques séparément en s'inspirant de la méthode d'Euler explicite. Dans un premier temps, on souhaite résoudre numériquement un système d'équations différentielles ordinaires sous la forme :

$$\frac{dY}{dt} = f(y,t), \quad y(t_0) = y_0 \quad y \in \mathbb{R}^n$$

Comme dit précédemment, nous devons traiter les différentes dynamiques séparément, ainsi notre algorithme reposera sur deux étapes :

1. Un **intégrateur interne** de type Euler explicite avec un petit pas h_{micro}.
2. Une **extrapolation projective** permettant de faire un grand saut temporel T_{macro}.

5.1 Acheminement

5.1.1 Initialisation

Débutons par l'initialisation de nos variables et objets. Comme nous disposons d'une condition initiale y_0 de dimension $n \geq 1$, nous devons nous assurer de manipuler correctement cette variable, qui peut être de dimension 1 ou plus grande. De ce fait, nous la convertissons en tableau NumPy avec des valeurs flottantes afin de faciliter les opérations vectorielles, et nous stockons la dimension de notre problème.

Ensuite, nous devons initialiser les différentes variables temporelles cruciales dans notre méthode. On définit `dt_global` comme la somme du temps total passé dans $[t_i,\ t_{i+1}]$ soit une itération projective

$$\Delta t = \texttt{dt\_global} = K \times h_{\text{micro}} + \texttt{T\_macro}$$

Ce pas de temps global sera utilisé pour mettre à jour l'instant dans chaque itération projective. Ensuite, nous devons calculer le nombre d'itérations tel que :

$$N = \left\lfloor \frac{t_f - t_0}{\Delta t} \right\rfloor$$

Cette variable représente le nombre de fois que nous allons effectuer l'itération projective.

Enfin, il ne nous reste plus qu'à créer des tableaux qui stockeront nos solutions à chaque itération. Nous initialisons `y_f`, qui est une matrice de dimension $N \times n$, où chaque ligne représente la solution à un instant précis, et `t_f`, un vecteur de longueur N contenant les instants. Ainsi, le premier élément de `t_f` est t_0, correspondant à la condition initiale, et la première ligne de `y_f` contient les valeurs de la condition initiale y_0.

5.1.2 Initialisation de l'itérations projectives

Maintenant que nous disposons de toutes nos variables et de nos différents typages, nous pouvons commencer l'implémentation de notre méthode. Dans un premier temps, pour chaque itération, nous allons récupérer la solution de l'itération précédente, notée $y_f[i-1]$, et créer une copie de

cette dernière afin de pouvoir utiliser cette valeur sans la modifier dans notre phase micro. Ainsi, $\forall i \in [1, \ldots, N-1]$.

$$\texttt{yt} = y_f[i-1]$$

5.1.3 Phase micro

Dans un premier temps on effectue K micro-pas en utilisant le schéma d'Euler explicite

$$y_{j+1} = y_j + h_{\text{micro}} f(y_j, t_j), \quad j = 0, \ldots, K-1$$

Comme mentionné dans la partie théorique, afin d'affiner l'estimation de la dérivée (la pente), on applique un $(K+1)$-ième pas d'Euler explicite :

$$y_{K+1} = y_K + h_{\text{micro}} f(y_K, t_K)$$

De ce fait à chaque itération la solution `yt` est mise à jour.

5.1.4 Phase macro, l'intégration projective

Maintenant qu'on a obtenu des estimations de solutions grâce à Euler explicite, nous pouvons extrapoler notre solution. Dans un premier temps on estime la pente avec une démarche qui imite une différence finie.

$$\text{pente} = \frac{y_{K+1} - y_K}{h_{\text{micro}}}$$

Ensuite nous extrapolons sur un pas de temps plus grand en ajoutant le produit de la pente avec le temps restant :

$$y_{\text{proj}} = y_{\text{micro},K+1} + \Big(\Delta t - (K+1)h_{\text{micro}}\Big) \times \text{pente} \quad \text{où } \Delta t_n = t_{n+1} - t_n$$

Ainsi, la solution extrapolée `y_proj` devient la solution de l'itération projective courante et est stockée dans le tableau `y_f`[i].

Ensuite nous mettons à jour le temps global en ajoutant notre pas de temps au vecteur `t_f` tel que :

$$\texttt{t\_f[i]} = \texttt{t\_f[i-1]} + \texttt{dt\_global}$$

5.2 Implémentation de la méthode

```
    def pfe(f, y0, t0, tf, h_micro, K, T_macro):
        y0 = np.array(y0, dtype=float)
        n = len(y0)
        dt_global = (K+1) * h_micro + T_macro
        N = int(np.floor((tf - t0) / dt_global))
        y_f = np.zeros((N, n))
        t_f = np.zeros(N)
        y_f[0] = y0
        t_f[0] = t0
        for i in range(1, N):
            yt = y_f[i - 1].copy()
            for j in range(K):
                yt = yt + h_micro * f(yt, t_f[i - 1] + j * h_micro)
            yt_2 = yt + h_micro * f(yt, t_f[i - 1] + K * h_micro)
            pente = (yt_2 - yt) / h_micro
            y_proj = yt_2 + (dt_global - (K + 1) * h_micro) * pente
            y_f[i] = y_proj
            t_f[i] = t_f[i - 1] + dt_global
        return t_f, y_f
```

Listing 5.1 – Schéma d'Intégration projective sur la base d'Euler explicite

Chapitre 6

Application et comparaison de performances de l'intégration projective

6.1 Résolution d'équations différentielles et systèmes raides

6.1.1 Équation linéaire raide

Considérons l'équation différentielle suivante

$$\frac{dy}{dt} = -1000\,(y - t) + 1 \qquad y(0) = 0.5,$$

Pour résoudre cette dernière en posant :

$$y(t) = t + Ce^{-1000t},$$

on a alors

$$y'(t) = 1 - 1000\,C\,e^{-1000t}.$$

En injectant nous obtenons

$$1 - 1000\,C\,e^{-1000t} = -1000\Big[\big(t + Ce^{-1000t}\big) - t\Big] + 1 = -1000\,C\,e^{-1000t} + 1,$$

ce qui est vérifié pour tout t. La condition initiale $y(0) = 0.5$ donne alors :

$$y(0) = 0 + C = 0.5 \quad \Longrightarrow \quad C = 0.5.$$

La solution exacte est donc

$$y(t) = t + 0.5\,e^{-1000t}.$$

Nous remarquons que le terme $0.5\,e^{-1000t}$ décroît rapidement tel que pour $t \geq 10^{-3}$ la solution est très proche de dynamique lente $y(t) \approx t$. Maintenant nous devons trouver nos paramètres pour respecter les conditions de stabilité. En effet, nous voulons :

$$\Big|1 + h_{\text{micro}}\,\lambda_{\text{test}}\Big| < 1 \quad \Longleftrightarrow \quad h_{\text{micro}} < \frac{2}{|\lambda_{\text{test}}|}.$$

Or, ici, la partie raide de l'équation est $-1000\,(y - t)$, donc $|\lambda_{\text{test}}| = 1000$ et

$$\frac{2}{|\lambda_{\text{test}}|} = \frac{2}{1000} = 0.002.$$

Ainsi, en choisissant

$$h_{\text{micro}} = 10^{-5},$$

nous avons $10^{-5} < 0.002$.

De plus, une autre condition est que :

$$|G| = \left|\left(1 + h_{\text{micro}}\lambda_{\text{test}}\right)^K \left[1 + \lambda_{\text{test}}\left(\Delta t - K\, h_{\text{micro}}\right)\right]\right| < 1.$$

Or,

$$1 + h_{\text{micro}}\lambda_{\text{test}} = 1 - 1000 \times 10^{-5} = 1 - 0.01 = 0.99,$$

donc,

$$\left(1 + h_{\text{micro}}\lambda_{\text{test}}\right)^K \approx (0.99)^{10} \approx 0.90438.$$

Et,

$$\lambda_{\text{test}}\left(\Delta t - K\, h_{\text{micro}}\right) = -1000\left(0.002 - 10\times 10^{-5}\right) = -1000\left(0.002 - 0.0001\right) = -1000\times 0.0019 = -1.9.$$

Ce qui donne :

$$1 + \lambda_{\text{test}}\left(\Delta t - K\, h_{\text{micro}}\right) = 1 - 1.9 = -0.9.$$

En prenant la valeur absolue, nous avons

$$\left|1 + \lambda_{\text{test}}\left(\Delta t - K\, h_{\text{micro}}\right)\right| = 0.9.$$

En regroupant ces résultats, nous obtenons :

$$|G| \approx 0.90438 \times 0.9 \approx 0.81394 < 1$$

Ce qui satisfait la condition de stabilité de la projection, et donc nos paramètres sont justes. En résumé, on effectue une courte phase de micro-intégration à l'aide d'un pas $h_{\text{micro}} = 10^{-5}$ pendant 10 micro-pas, puis on projette la solution sur un intervalle plus long T_{macro}. Nos paramètres sont donc :

— $h_{\text{micro}} = 10^{-5}$
— $K = 10$
— $T_{\text{macro}} = 0.002$

Ainsi la phase micro dure $10 \times 10^{-5} = 10^{-4}$ et le pas global de la méthode est

$$\Delta t = K\, h_{\text{micro}} + T_{\text{macro}} = 10^{-4} + 0.002 = 0.0021$$

Ainsi, pour simuler sur 1 secondes, le nombre d'itérations de la méthode PFE sera d'environ

$$N \approx \frac{1}{0.0021} \approx 476$$

Passons à sa simulation en implémentant l'équation différentielle raide :

```
def f(y, t):
    return -1000*(y - t) + 1

def solution_f(t):
    return t + 0.5 * np.exp(-1000*t)
```

Listing 6.1 – Implémentation de l'ED raide

Ensuite nous définissons nos paramètres :

```
#Conditions initiales et duree de simulation
y0 = 0.5 #condition initiale
t0 = 0.0 #temps de depart
tf = 1.0 #temps max de simulation

#Parametres
h_micro = 1e-5
K = 10
T_macro = 0.002  #projection sur 0.002 secondes
```

Listing 6.2 – Paramètres de notre méthode

Ainsi nous obtenons :

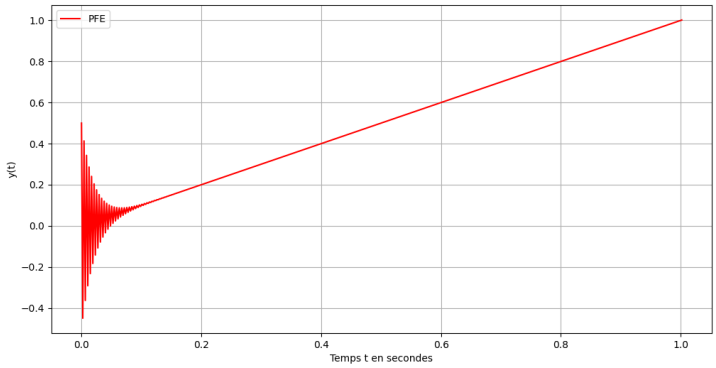


FIGURE 6.1 – Simulation de notre équation linéaire raide

6.1.2 Brusselator

Le Brusselator est un système d'équations différentielles ordinaires utilisé pour modéliser des réactions chimiques oscillantes. Il constitue un exemple de système dynamique présentant un comportement raide pour certains paramètres. Ce modèle, introduit dans le contexte de la chimie non linéaire, repose sur les équations suivantes :

$$(B)\begin{cases}\frac{dX}{dt} = A + X^2Y - BX - X \\ \frac{dY}{dt} = BX - X^2Y\end{cases}$$

où A et B sont des paramètres positifs.

Afin d'appliquer notre méthode numérique, nous devons nous assurer que les conditions de stabilité sont respectées. Cependant, nous ne pouvons pas appliquer notre test à ce système : nous disposons de deux variables A et B et nous devons choisir nos paramètres de l'intégration projective selon ces deux variables positives. Cependant, on ne peut pas choisir des valeurs anodines pour A et B, nous devons étudier le système pour obtenir son comportement et connaître pour quelles valeurs il présente une dynamique raide et rapide, en d'autres termes, oscillatoire.
Nous allons étudier le système (B) autour de son point d'équilibre qui est :

$$X^* = A, \quad Y^* = \frac{B}{A}.$$

En étudiant la Jacobienne du système (B) nous obtenons :

$$J = \begin{pmatrix}\frac{\partial f}{\partial X} & \frac{\partial f}{\partial Y} \\ \\ \frac{\partial g}{\partial X} & \frac{\partial g}{\partial Y}\end{pmatrix} = \begin{pmatrix}2XY - (B+1) & X^2 \\ \\ B - 2XY & -X^2\end{pmatrix}$$

où $f(X,Y) = A - (B+1)X + X^2Y$ et $g(X,Y) = BX - X^2Y$,
En calculant le polynôme caractéristique, nous trouvons :

$$det(J - \alpha Id_2) = det\begin{pmatrix}2XY - (B+1) - \alpha & X^2 \\ \\ B - 2XY & -X^2 - \alpha\end{pmatrix}$$

En évaluant au point d'équilibre :

$$X^* = A, \quad Y^* = \frac{B}{A}.$$

$$det(J - \alpha Id_2) = det\begin{pmatrix}2B - B - 1 - \alpha & A^2 \\ \\ B - 2B & -A^2 - \alpha\end{pmatrix} = det\begin{pmatrix}B - 1 - \alpha & A^2 \\ \\ -B & -A^2 - \alpha\end{pmatrix} = \alpha^2 + \alpha[A^2 - (B-1)] + A^2$$

Comme dit précédemment, nous souhaitons que le système (B) présente des dynamiques rapides et raides. Pour cela, il faut que les valeurs propres soient complexes conjuguées, ce qui implique que le discriminant de ce polynôme soit négatif.

$$\Delta = [A^2 - (B-1)]^2 - 4A^2 < 0 \iff [A^2 - (B-1)]^2 < 4A^2$$

En résolvant cette inégalité, nous obtenons que les valeurs propres sont complexes conjuguées si et seulement si :

$$(A-1)^2 < B < (A+1)^2$$

Ainsi en choisissant A = 1 et B = 3, nous remplissons la condition ci-dessus et le point d'équilibre s'écrit : $(X^*, Y^*) = (1,3)$. Il nous reste plus qu'à étudier la stabilité de notre modèle pour ces paramètres.

Nous devons dans un premier temps trouver notre λ_{test} qui nous permettra d'étudier la stabilité. Il correspond à la valeur absolue des valeurs propres du polynôme caractéristique de la Jacobienne suivante :

$$J = \begin{pmatrix} \frac{\partial f}{\partial X} & \frac{\partial f}{\partial Y} \\ \frac{\partial g}{\partial X} & \frac{\partial g}{\partial Y} \end{pmatrix} = \begin{pmatrix} 2XY - (B+1) & X^2 \\ B - 2XY & -X^2 \end{pmatrix}$$

Par les mêmes calculs que précédemment et en remplaçant A et B par leurs valeurs nous obtenons :

$$det(J - \alpha Id_2) = det \begin{pmatrix} 2B - B - 1 - \alpha & A^2 \\ B - 2B & -A^2 - \alpha \end{pmatrix} = det \begin{pmatrix} B - 1 - \alpha & A^2 \\ -B & -A^2 - \alpha \end{pmatrix} = \alpha^2 + \alpha + 1$$

Ansi les valeurs propres sont :

$$\alpha = \frac{1 \pm i\sqrt{3}}{2}$$

Cependant, nous devons introduire un théorème utile pour le choix de notre λ_{test} et qui nous permet de comprendre pourquoi nous faisons tous ces calculs.

Théorème 9. Théorème de la bifurcation de Hopf
Soit un système dynamique non linéaire donné par

$$\dot{x} = F(x, \mu), \quad x \in \mathbb{R}^n, \ \mu \in \mathbb{R},$$

avec F suffisamment régulier. Supposons que x^* est un point fixe pour $\mu = \mu_0$ et que la matrice jacobienne $J(x^*, \mu_0)$ possède une paire de valeurs propres simples complexes conjuguées

$$\lambda(\mu_0) = \alpha(\mu_0) \pm i\beta(\mu_0)$$

telles que

$$\alpha(\mu_0) = 0, \quad \beta(\mu_0) \neq 0,$$

et que la dérivée de $\alpha(\mu)$ par rapport à μ en μ_0 est non nulle. Alors, localement autour de x^* et pour μ proche de μ_0, le système subit une bifurcation de Hopf : le point fixe x^* perd sa stabilité lorsque $\alpha(\mu)$ passe de négatif à positif ou inversement et une solution périodique apparaît ou disparaît.

Ici, la partie réelle vaut $+\frac{1}{2}$, indiquant que le point fixe est instable. Toutefois, dans le cadre de notre méthode à deux temps, nous ciblons la capture de la dynamique rapide dans la direction stable qui est la dynamique lente. Comme dans notre contexte la valeur absolue de λ_{test} est : $|\alpha| = 1$, par le théorème de la bifucrcation de Hopf, on utilise souvent une valeur $\lambda_{\text{test}} = -1$ pour la phase micro, afin d'avoir un schéma stable qui annule rapidement les transitoires non désirées.

Cela rejoint ce que nous avons énoncé dans notre Chapitre 4 sur la stabilité de l'intégration projective numérique : Pour étudier la stabilité, il est courant d'examiner le comportement de la méthode sur l'équation de test linéaire $f(t, y) = \frac{dy}{dt} = \lambda y$ où $\lambda \in \mathbb{C}$ avec $\Re(\lambda) < 0$ pour les modes raides. Nous pouvons désormais étudier les paramètres pour garantir une stabilité et consistance de notre méthode. Commençons par étudier la stabilité locale de la phase micro avec $\lambda_{\text{test}} = -1$

$$|1 + h_{micro}\lambda_{\text{test}}| = |1 - h_{micro}| < 1 \implies h_{micro} < \frac{2}{|\lambda_{\text{test}}|} = 2$$

Ainsi, pour assurer une bonne approximation et respecter la condition ci-dessus nous pouvons aisement choisir $h_{micro} = 0.01$. Ensuite étudions la stabilité de la phase macro avec $\lambda_{\text{test}} = -1$ et $h_{micro} = 0.01$ et K = 10, nous devons satisfaire :

$$|G| = \left|\left(1 + h_{\text{micro}}\lambda_{\text{test}}\right)^K \left[1 + \lambda_{\text{test}}\left(\Delta t - K\, h_{\text{micro}}\right)\right]\right| < 1.$$

Or

$$1 + h_{\text{micro}}\lambda_{\text{test}} = 1 - 1 \times 10^{-2} = 0.99$$

Donc

$$\left(1 + h_{\text{micro}}\lambda_{\text{test}}\right)^K \approx (0.99)^{10} \approx 0.904.$$

Or, comme nous nous situons dans une zone délicate due aux oscillations, nous devons faire attention à ce que l'extrapolation soit positive. De ce fait, nous imposons :

$$T_{\text{macro}} > K h_{\text{micro}} = 0.1$$

Ainsi on peut choisir $T_{\text{macro}} = 0.11$. De ce fait :

$$1 + \lambda_{\text{test}}\left(\Delta t - K\, h_{\text{micro}}\right) = 1 - 0.01 = 0.99$$

En regroupant ces résultats, nous obtenons :

$$|G| \approx 0.904 \times 0.99 \approx 0,8949 < 1$$

Ce qui satisfait la condition de stabilité de la projection, et donc nos paramètres sont justes. En résumé, on effectue une courte phase de micro-intégration à l'aide d'un pas $h_{\text{micro}} = 10^{-2}$ pendant 10 micro-pas, puis on projette la solution sur un intervalle plus long T_{macro}. Nos paramètres sont donc :

— $h_{\text{micro}} = 10^{-2}$
— $K = 10$
— $T_{\text{macro}} = 0.11$

Ainsi la phase micro dure $10 \times 10^{-5} = 10^{-4}$ et le pas global de la méthode est

$$\Delta t = K\, h_{\text{micro}} + T_{\text{macro}} \approx 0.22$$

Ainsi, pour simuler sur 500 secondes, le nombre d'itérations de la méthode PFE sera d'environ

$$N \approx \frac{500}{0.22} \approx 2174$$

Passons maintenant à la simulation de notre système (B) :

```
def brusselator(y, t):
    A = 1
    B = 3
    X, Y = y
    dXdt = A - (B+1)*X + X**2 * Y
    dYdt = B * X - X**2 * Y
    return np.array([dXdt, dYdt])
```

Listing 6.3 – Implémentation du Brusselator

Ensuite nous définissons nos paramètres :

```
#Parametres de simulation
y0 = [1.0, 1.0]
t0 = 0.0
tf = 500.0

#Parametres de la methode
h_micro = 0.01
K = 10
T_macro = 0.11
```

Listing 6.4 – Paramètres de notre méthode pour le Brusselator

Ainsi nous obtenons :

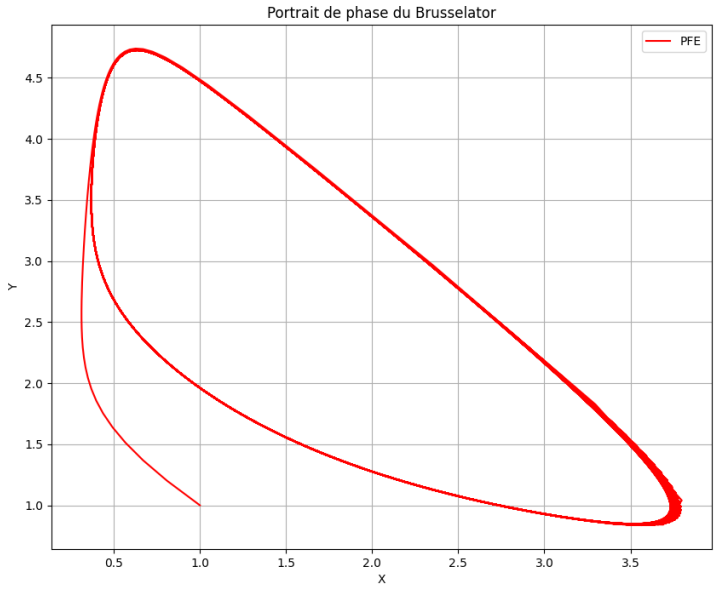


FIGURE 6.2 – Simulation du portrait de phase du Brusselator pour A = 1 et B = 3

6.1.3 Système de Lorenz

Le système de Lorenz est l'un des modèles mathématiques les plus célèbres pour illustrer la complexité et la sensibilité liées aux conditions initiales dans des systèmes dynamiques non linéaires. Ce système de trois équations différentielles ordinaires a révolutionné notre compréhension du chaos déterministe. Le modèle s'écrit de la manière suivante :

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases}$$

où x, y et z représentent des paramètres précis en physique. Les valeurs classiques souvent étudiées sont

$$\sigma = 10, \quad \rho = 28, \quad \beta = \frac{8}{3}.$$

Pour cet exemple, nous n'allons pas réaliser l'étude complète du portrait de phase de ce système. Nous utiliserons les paramètres obtenus dans la section précédente, qui satisfont les conditions de stabilité et de consistance de la méthode d'intégration projective numérique. Nos paramètres sont donc :

— $h_{\text{micro}} = 10^{-2}$
— $K = 10$
— $T_{\text{macro}} = 0.11$

En implémentant le système de Lorenz en python :

```python
def lorenz_systeme(state, t, sigma=10, rho=28, beta=8/3):
    x, y, z = state
    dx = sigma * (y - x)
    dy = x * (rho - z) - y
    dz = x * y - beta * z
    return np.array([dx, dy, dz])
```

Listing 6.5 – Implémentation du système de Lorenz

Ensuite nous définissons nos paramètres :

```
#Parametres de simulation
y0 = [1.0, 1.0, 1.0]
t0 = 0.0
tf = 50.0

#Parametres de la methode
h_micro = 0.01
K = 10
T_macro = 0.11
```

Listing 6.6 – Paramètres de notre système de Lorenz

Ainsi nous obtenons :

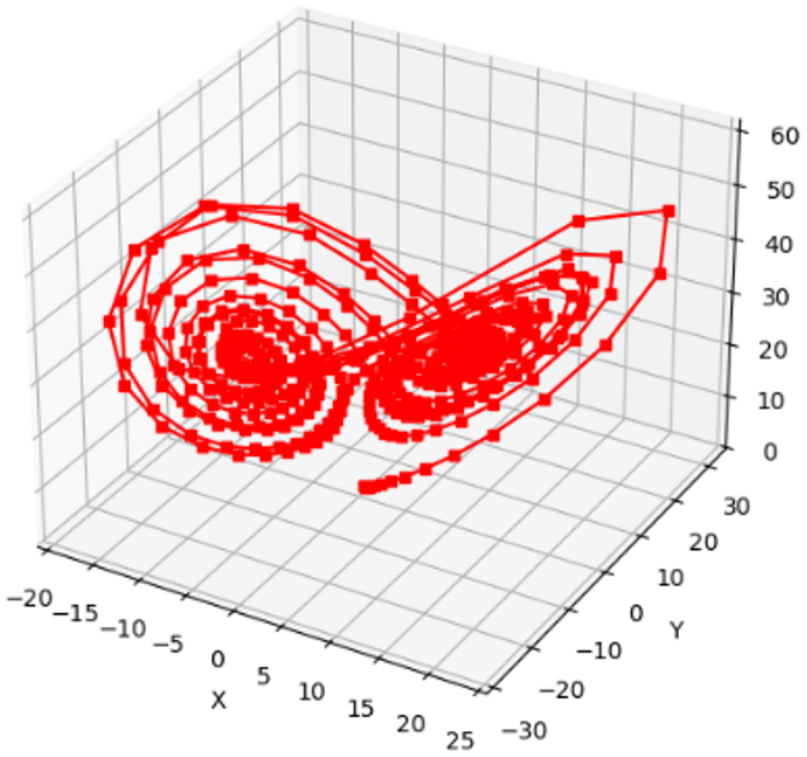


FIGURE 6.3 – Simulation du portrait de phase du système de Lorenz

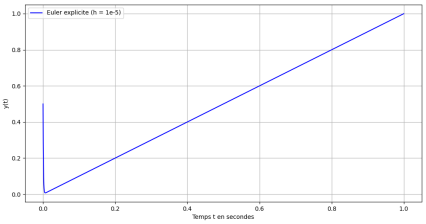
Après avoir illustré l'application de l'intégration projective sur divers exemples présentant des dynamiques rapides et raides, il est désormais pertinent de procéder à une analyse comparative. Nous allons confronter les performances de notre méthode à celles du schéma numérique d'Euler explicite sur les mêmes exemples.

6.2 Comparaisons

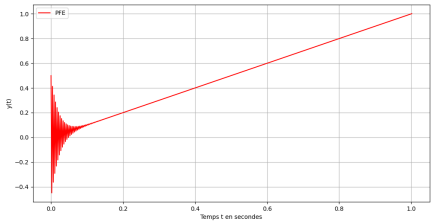
Ces comparaisons nous permettront de quantifier les gains, s'il y en a, en temps de calcul et en précision offerts par l'intégration projective par rapport à une méthode élémentaire. Pour ce faire, nous allons utiliser les mêmes mécanismes qu'au Chapitre 3 : nous calculerons l'erreur moyenne entre l'approximation et la solution, l'erreur absolue, qui correspond à l'erreur maximale, et enfin l'erreur L^2.

6.2.1 Équation linéaire raide

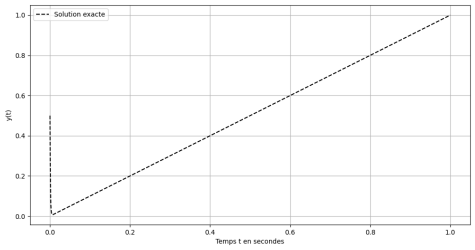
$$\frac{dy}{dt} = -1000\,(y - t) + 1 \qquad y(0) = 0.5,$$



Euler explicite



PFE



Solution exacte

FIGURE 6.4 – Simulation sur l'équation linéaire raide

TABLE 6.1 – Erreurs pour notre équation différentielle

Méthode	**Temps de calcul (s)**	**Erreur Moyenne**	**Erreur Abs**	**Erreur norme 2**
Euler Explicite	0.059	2.5041×10^{-6}	9.2355×10^{-4}	1.2537×10^{-2}
Intégration Projective	0.001	1.0×10^{-2}	5.134×10^{-1}	1.085

Ainsi nous remarquons les différences entre les erreurs suivantes :

TABLE 6.2 – Différences des erreurs

Méthode	**ΔErreur Moyenne**	**ΔErreur Abs**	**ΔErreur L2**
|EE - PFE|	1.000558×10^{-2}	5.124967×10^{-1}	1.072169

De plus nous remarquons que dans notre simulation, la phase micro de l'intégration projective dure

$$10 \times 10^{-5} = 10^{-4} \text{ secondes,}$$

et que le pas global est défini par

$$\Delta t = K\, h_{\text{micro}} + T_{\text{macro}} = 10^{-4} + 0.002 = 0.0021 \text{ secondes.}$$

Ainsi, pour simuler sur 1 seconde, on effectue environ

$$N_{PFE} \approx \frac{1}{0.0021} \approx 476 \text{ itérations.}$$

En revanche, pour la méthode Euler explicite avec un pas de $h = 1 \times 10^{-5}$ secondes, le nombre d'itérations nécessaire est

$$N_{\text{EE}} \approx \frac{2}{1 \times 10^{-5}} = 200000$$

ce qui est nettement supérieur, d'où la différence de temps de calcul.

Les tableaux mettent en évidence que la méthode Euler explicite donne de bonnes approximations. Cependant, cette grande précision est obtenue au prix d'un temps de calcul plus élevé et d'un très grand nombre d'itérations, environ 200 000, pour simuler l'intervalle étudié.

À l'inverse, la méthode d'intégration projective (noté PFE) présente un temps de calcul négligeable et demande environ 476 itérations pour la simulation sur 1 seconde, grâce à un pas global de 0.0021 secondes.

Nous pouvons donc observer que, malgré un avantage en temps de calcul considérable, la méthode PFE souffre d'une erreur d'extrapolation élevée dans ce problème raide due au grand pas global Δt imposé pour exploiter les gains en rapidité, la phase micro de 10^{-4} secondes ne suffit pas à compenser l'erreur accumulée lors de la projection sur un intervalle de 0.002 secondes. Cependant si nous appliquons la méthode d'Euler explicite sur l'intervalle [0,1] avec seulement 476 itérations et si nous augmentons le nombre d'itérations à 200 000 de notre méthode d'intégration projective de la manière suivante :

```
#Conditions initiales et duree de simulation
y0 = 0.5
t0 = 0.0
tf = 1

#Parametres pour Euler explicite pour avoir 472 iterations
h_euler = 0.00211 #(1/472)

#Parametres PFE pour avoir 200 000 iterations
h_micro = 1e-7
K = 10
T_macro = 0.000009
```

Listing 6.7 – Modification de paramètres

Ainsi nous obtenons les erreurs suivantes :

Table 6.3 – Erreurs à 472 itérations

Méthode	Temps de calcul	Erreur Moyenne	Erreur Abs	Erreur norme 2
Euler explicite	0.007	4.9183×10^{40}	4.6254×10^{42}	1.0657×10^{43}
Intégration Projective	0.001	1.0×10^{-2}	5.134×10^{-1}	1.085

Table 6.4 – Erreurs à 200 000 itérations

Méthode	Temps de calcul	Erreur Moyenne	Erreur Abs	Erreur norme 2
Euler Explicite	0.059	2.5041×10^{-6}	9.2355×10^{-4}	1.2537×10^{-2}
Intégration Projective	0.452	1.0255×10^{-6}	7.4884×10^{-4}	1.016741×10^{-2}

De ce fait, nous remarquons qu'à itérations égales, la méthode d'intégration projective sera toujours plus précise que le schéma numérique d'Euler explicite. Nous allons appliquer le même procédé aux autres exemples.

6.2.2 Brusselator

$$(B) \begin{cases} \frac{dX}{dt} = A + X^2Y - BX - X \\ \frac{dY}{dt} = BX - X^2Y \end{cases}$$

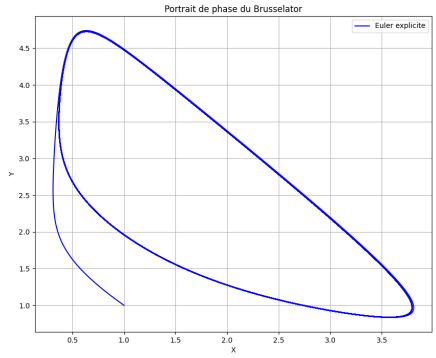
où $A = 1$ et $B = 3$

Pour les paramètres suivants :

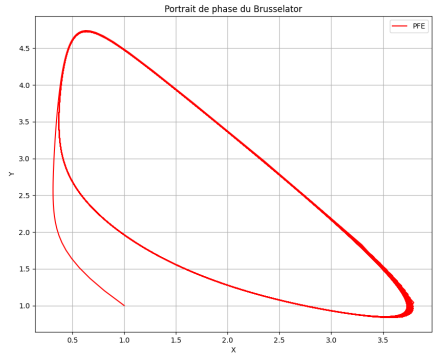
```
#Parametres de simulation
y0 = [1.0, 1.0]
t0 = 0.0
tf = 500.0
#Parametres pour Euler explicite
h_euler = 0.01
#Parametres pour PFE
h_micro = 0.01
K = 10
T_macro = 0.11
```

Listing 6.8 – Paramètres pour le Brusselator

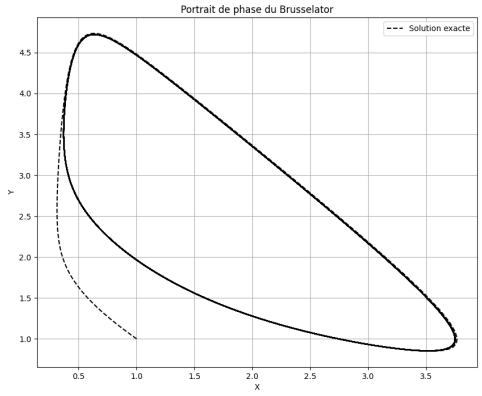
Nous obtenons les simulations suivantes :



Euler explicite



PFE



Solution exacte

FIGURE 6.5 – Simulation sur le Brusselator

TABLE 6.5 – Erreurs pour le Brusselator

Méthode	**Temps de calcul (s)**	**Erreur Moyenne**	**Erreur Abs**	**Erreur norme 2**
Euler Explicite	0.209	6.5194×10^{-1}	4.806	2.439×10^{-2}
Intégration Projective	0.088	1.4273	4.356	7.9393×10^{-1}

De la même manière, nous remarquons que le nombre d'itérations diffère entre les deux méthodes. Nous avons environ 2174 itérations pour PFE et environ 50 000 itérations pour Euler explicite. Si nous modifions nos paramètres afin de comparer les deux méthodes avec un nombre d'itérations égal, nous obtenons :

```
#Parametres de simulation
y0 = [1.0, 1.0]
t0 = 0.0
tf = 500.0

#Parametres pour Euler explicite
h_euler = 0.23 #500/0.23 = 2174 iterations environs

#Parametres pour PFE
h_micro = 1e-5
K = 10
T_macro = 0.0099
```

Listing 6.9 – Paramètres modifiés pour le Brusselator

Ainsi nous obtenons les résultats suivant :

Table 6.6 – Erreurs à 2174 itérations

Méthode	Temps de calcul	Erreur Moyenne	Erreur Abs	Erreur norme 2
Euler Explicite	20.321 s	NA	NA	NA
Intégration Projective	0.088	1.4273	4.356	7.9393×10^{-1}

On remarque que l'avantage de la méthode PFE réside dans son efficacité en termes de temps de calcul, quelle que soit la configuration. Cependant, cette efficacité s'accompagne d'une perte de précision, comme le montrent les erreurs. Lorsque l'on égalise le nombre d'itérations, par exemple avec 2174 itérations pour chaque méthode, Euler explicite devient instable et génère des erreurs. En ce qui concerne les deux méthodes avec 50000 itérations, nous obtenons :

Table 6.7 – Erreurs à 50 000 itérations

Méthode	Temps de calcul	Erreur Moyenne	Erreur Abs	Erreur norme 2
Euler Explicite	0.209	6.5194×10^{-1}	4.806	2.439×10^{-2}
Intégration Projective	0.092	1.298	4.5326	7.506×10^{-1}

Table 6.8 – Comparaison PFE et Euler explicite (En moyenne)

Méthode	Temps de calcul	Erreur Moyenne	Erreur Abs	Erreur norme 2
PFE	**x2.3 plus rapide**	**x1.6 plus d'erreur**	**x1.06 moins d'erreur**	**x30 plus d'erreur**

En somme, le choix entre Euler explicite et la méthode d'intégration projective dépend fortement de l'objectif. Si la précision locale et la bonne approximation sont prioritaires, Euler explicite est à privilégier. En revanche, si l'on cherche à réduire le coût de calcul global, la méthode PFE offre un temps de calcul très avantageux, au prix d'une perte en précision due à l'erreur d'extrapolation, qu'on peut essayer de contrôler et diminuer.

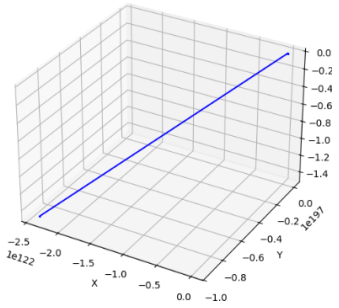
6.2.3 Système de Lorenz

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases}$$

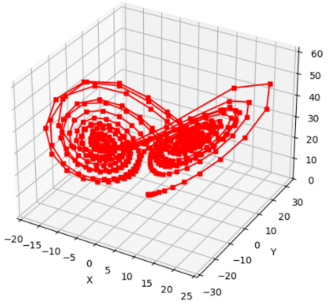
où x, y et z et $\sigma = 10, \quad \rho = 28, \quad \beta = \frac{8}{3}$. Pour les paramètres suivants :

```
#Parametres de simulation
y0 = [1.0, 1.0, 1.0]
t0 = 0.0
tf = 50.0

#Parametres Euler explicite
h_euler = 0.21

#Parametres PFE
h_micro = 0.01
K = 10
T_macro = 0.11
```

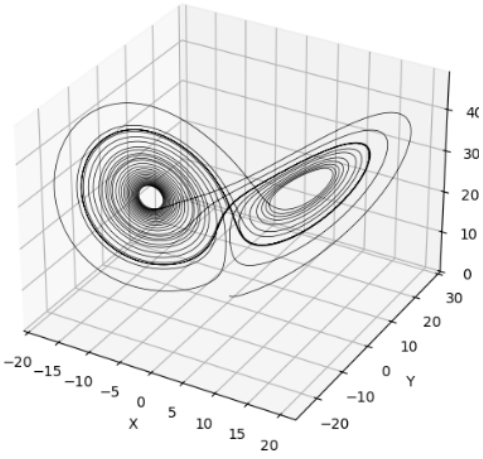
Listing 6.10 – Paramètres pour le système de Lorenz



Euler explicite



PFE



Solution exacte

TABLE 6.9 – Erreurs pour le système de Lorenz

Méthode	**Temps de calcul**	**Erreur Moyenne**	**Erreur Abs**	**Erreur norme 2**
Euler Explicite	0.003 s	NA	NA	NA
Intégration Projective	0.012	1.86848×10^1	4.691×10^1	3.1905×10^2

Les résultats montrent que, pour ces paramètres, la méthode PFE génère des erreurs élevées, traduisant une imprécision de l'approximation projective par rapport à la solution exacte du système de Lorenz. La méthode PFE effectue ici environ 227 itérations avec un pas global de $\Delta t = 0.22$, tandis qu'Euler explicite, avec le même intervalle et le même nombre d'itérations, ne parvient pas à fournir une approximation en raison d'un pas de temps trop grand. En réduisant le pas de temps d'Euler explicite à 0.01 seconde, cette méthode réalise environ 5 000 itérations, nous permettant d'obtenir le portrait de phase suivant :

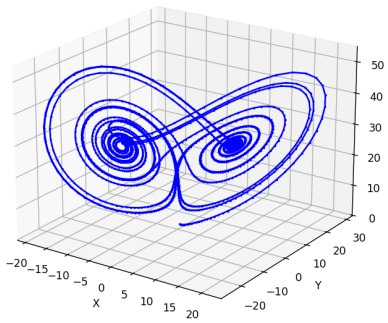


FIGURE 6.6 – Approximation du portrait de phase du système de Lorenz par Euler explicite

Et les résultats suivants :

TABLE 6.10 – Erreurs pour 5 000 itérations d'Euler explicite et 227 itérations pour PFE

Méthode	**Temps de calcul**	**Erreur Moyenne**	**Erreur Abs**	**Erreur norme 2**
Euler Explicite	0.023	1.87160×10^1	4.7261×10^1	1.4619×10^3
Intégration Projective	0.012	1.86848×10^1	4.691×10^1	3.1905×10^2

TABLE 6.11 – Comparaison PFE et Euler explicite à 5 000 itérations

Méthode	**Temps de calcul**	**Erreur Moyenne**	**Erreur Abs**	**Erreur norme 2**
PFE	x2 plus rapide	x1.1 moins d'erreur	x1.007 moins d'erreur	x4.6 moins d'erreur

Nous remarquons à travers ces résultats que la méthode PFE est nettement plus rapide en termes de calcul, mais aussi en termes d'erreur. En effet, l'intégration projective fournit une approximation de la solution qui est globalement plus précise, même en utilisant moins d'itérations qu'Euler explicite. Dans cet exemple, nous exploitons pleinement les avantages de notre méthode, qui consiste à capturer la dynamique rapide au niveau microscopique, puis à extrapoler sur une dynamique plus lente au niveau macroscopique. Cela permet d'obtenir un compromis favorable entre une faible complexité computationnelle et une erreur globale réduite, en particulier en norme L^2.

6.3 Conclusion

Les résultats de nos simulations montrent clairement un compromis entre Euler explicite et l'intégration projective (PFE). D'une part, Euler explicite, avec environ 200000 itérations en utilisant un pas très petit, offre une précision considérable, mais au prix d'un coût de calcul élevé. D'autre part, la méthode PFE atteint des résultats comparables en beaucoup moins d'itérations et avec un temps de calcul nettement réduit.

En ajustant les paramètres de la PFE, on peut réduire l'erreur d'extrapolation et obtenir une approximation de la solution souvent plus précise, notamment en norme L^2, qu'avec Euler explicite pour un nombre d'itérations équivalent.

Ainsi, l'intégration projective apparaît comme une méthode efficace pour simuler efficacement des systèmes complexes et raides, tout en assurant une bonne précision avec un coût computationnel réduit.

Néanmoins les phénomènes qui nous entourent dépendent rarement d'un seul paramètre comme le temps, mais de plusieurs paramètres comme la gravité, la température. De ce fait nous allons nous intéresser à une notion mathématique qui permet de modéliser ces phénomènes, les équations différentielles aux dérivés partielles.

Chapitre 7

Introduction équations aux dérivées partielles et schémas numériques

Les équations aux dérivées partielles, notées EDP, jouent un rôle central dans la modélisation de phénomènes complexes où plusieurs variables indépendantes interagissent. Alors qu'une équation différentielle ordinaire, EDO, peut décrire l'évolution d'une grandeur en fonction d'une seule variable souvent le temps alors qu'une EDP permet de tenir compte simultanément divers paramètres. Ces dimensions multiples sont essentielles pour modéliser par exemple la propagation d'ondes, la diffusion de la chaleur, ou encore la dynamique des fluides. L'étude des EDP est donc à la croisée des chemins entre théorie mathématique, modélisation physique et applications numériques.

7.1 Introduction à la théorie des équations aux dérivées partielles

7.1.1 Définitions et exemples fondamentaux

Définition 36. Équation aux dérivées partielles
On appelle équation aux dérivées partielles une équation mettant en relation une fonction inconnue $u = u(x_1, x_2, \ldots, x_n)$ et ses dérivées partielles d'ordre un ou supérieur. On la présente généralement sous la forme :

$$F\left(x_1, \ldots, x_n,\, u,\, \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial^2 u}{\partial x_i \partial x_j}, \ldots\right) = 0.$$

L'un des exemples les plus populaires qui représente les EDP est l'équation de la chaleur en dimension 1 :

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2},$$

où $D > 0$ désigne le coefficient de diffusion. Parmi les autres exemples notables, on trouve l'équation des ondes et l'équation de Laplace, cette dernière étant au cœur de la théorie du potentiel pour les problèmes stationnaires.

Nous avons mentionné précédemment la notion d'ordre, mais qu'est-ce que cela signifie ?

Définition 37. Ordre d'une équation aux dérivées partielles
L'ordre d'une EDP correspond au degré maximum des dérivées partielles apparentes dans l'équation. Par exemple, l'équation de la chaleur est d'ordre 2 puisque la dérivée spatiale la plus élevée est de deuxième ordre.

7.1.2 Classification des EDP linéaires du second ordre

Considérons une EDP linéaire du second ordre à deux variables :

$$A(x,y)\frac{\partial^2 u}{\partial x^2} + 2B(x,y)\frac{\partial^2 u}{\partial x \partial y} + C(x,y)\frac{\partial^2 u}{\partial y^2} + \cdots = 0.$$

où $A(x, y)$ est le coefficient qui pondère l'effet de la dérivée seconde par rapport à x reflétant ainsi la variabilité locale des propriétés physiques du milieu. La classification d'une équation différentielle partielle repose sur le discriminant :

$$\Delta = B^2 - AC.$$

— Si $\Delta > 0$, l'équation est dite **hyperbolique** (par exemple l'équation des ondes) : elle possède des caractéristiques réelles distinctes et modélise la propagation d'informations.
— Si $\Delta = 0$, l'équation est dite **parabolique** (comme l'équation de la chaleur) : elle représente un comportement intermédiaire, avec une seule famille de caractéristiques réelles.
— Si $\Delta < 0$, l'équation est dite **elliptique** (comme l'équation de Laplace) : elle n'admet pas de caractéristiques réelles et décrit souvent des situations stationnaires ou en équilibre.

Les caractéristiques d'une EDP correspondent aux courbes le long desquelles l'équation se simplifie en une équation différentielle ordinaire. L'analyse de ces dernières permet d'interpréter la direction et la vitesse de propagation des perturbations dans le support du phénomène modélisé.

7.1.3 Existence, unicité et régularité des solutions

La théorie des EDP ne se limite pas à leur formulation : il est indispensable d'établir des résultats garantissant l'existence et l'unicité des solutions, ainsi que leur régularité. Avant d'énoncer ces théorèmes, expliquons ce qu'est une fonction analytique.

Définition 38. Fonction analytique
Soit $\mathbb{O}$ un ouvert de $\mathbb{R}$ (ou de $\mathbb{C}$). Une fonction $f : \mathbb{O} \to \mathbb{R}$ (ou $\mathbb{C}$) est dite analytique sur $\mathbb{O}$ si, pour tout point $x_0 \in U$, il existe un voisinage $V \subset \mathbb{O}$ de x_0 et une suite de coefficients $(a_n)_{n\in\mathbb{N}}$ tels que, pour tout $x \in V$,

$$f(x) = \sum_{n=0}^{\infty} a_n (x - x_0)^n,$$

avec la série entière convergente et égale à $f(x)$ sur V.

Ainsi nous pouvons énoncer un théorème primordiale en analyse des EDP.

Théorème 10. Théorème de Cauchy–Kowalevski
Soit $m \geq 1$ un entier et considérons le problème de Cauchy suivant pour une fonction inconnue $u = u(t, x_1, \ldots, x_n)$ dans un voisinage de $(0, x_0) \in \mathbb{R}^{n+1}$:

$$\frac{\partial^m u}{\partial t^m} = F\Big(t, x_1, \ldots, x_n,\, u,\, \frac{\partial u}{\partial t},\, \frac{\partial u}{\partial x_1},\, \ldots,\, \frac{\partial^k u}{\partial t^{k_0} \partial x_1^{k_1} \cdots \partial x_n^{k_n}},\, \ldots\Big),$$

avec les conditions initiales :

$$\frac{\partial^k u}{\partial t^k}(0, x) = \phi_k(x), \quad \textit{pour } k = 0, 1, \ldots, m-1.$$

Si la fonction F ainsi que les données initiales $\phi_0, \phi_1, \ldots, \phi_{m-1}$ sont analytiques par rapport à toutes leurs variables dans un voisinage de $(0, x_0)$, alors il existe un voisinage de $(0, x_0)$ dans lequel le problème admet une solution unique $u(t, x)$ elle-même analytique.

Maintenant que nous avons l'unicité et l'existence de la solution sur un domaine, nous pouvons apporter plus de précision sur ce dernier.

Théorème 11. Principe du maximum fort
Soit $\Omega \subset \mathbb{R}^n$ un domaine ouvert, connexe et borné, et soit $u \in C^2(\Omega) \cap C^0(\overline{\Omega})$ une solution d'une équation elliptique de la forme

$$Lu = 0 \quad \textit{dans } \Omega,$$

où l'opérateur L est elliptique (par exemple, le laplacien $\Delta u = 0$). Le principe du maximum fort affirme que si u atteint un maximum (ou un minimum) en un point intérieur de Ω, alors u est constante dans Ω. En conséquence, pour toute solution non constante, le maximum (et le minimum) de u est nécessairement atteint sur le bord $\partial\Omega$.

Remarque 8. *Ce principe fournit des estimations essentielles sur le comportement des solutions et joue un rôle fondamental dans l'analyse qualitative des problèmes elliptiques.*

Après avoir établi que le principe du maximum fort offre un contrôle qualitatif des solutions et que le théorème de Cauchy–Kowalevski garantit, dans un cadre analytique, leur existence locale et unicité, nous nous orientons vers un cadre variationnel où le théorème de Lax–Milgram assure l'existence et l'unicité des solutions faibles.

Théorème 12. Théorème de Lax–Milgram
Soient :

1. V un espace de Hilbert muni de la norme $\|\cdot\|$
2. $a(\cdot,\cdot) : V \times V \to \mathbb{K}$ (avec $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$) une forme bilinéaire (ou sesquilinéraire en cas complexe) continue, c'est-à-dire qu'il existe une constante $M > 0$ telle que

$$|a(u,v)| \leq M \|u\| \|v\|, \quad \forall\, u, v \in V$$

3. $a(\cdot,\cdot)$ soit coercive, c'est-à-dire qu'il existe une constante $\alpha > 0$ telle que

$$a(v,v) \geq \alpha \|v\|^2, \quad \forall\, v \in V,$$

4. $L : V \to \mathbb{K}$ une forme linéaire continue, c'est-à-dire qu'il existe une constante $C_L > 0$ telle que

$$|L(v)| \leq C_L \|v\|, \quad \forall\, v \in V.$$

Alors, il existe un unique $u \in V$ tel que

$$a(u,v) = L(v), \quad \forall\, v \in V.$$

De plus, la solution u satisfait l'inégalité suivante :

$$\|u\| \leq \frac{1}{\alpha} \|L\|_{V'},$$

où la norme $\|L\|_{V'}$ du dual V' est définie par

$$\|L\|_{V'} = \sup_{\substack{v \in V \\ \|v\|=1}} |L(v)|.$$

Après avoir montré que le principe du maximum fort fournit un contrôle qualitatif essentiel sur le comportement des solutions, que le théorème de Cauchy–Kowalevski garantit dans un cadre analytique l'existence locale et l'unicité, et que le théorème de Lax–Milgram assure, via une approche variationnelle, l'existence et l'unicité des solutions faibles, nous nous tournons désormais vers l'approche numérique afin d'élaborer des schémas de discrétisation capables de concrétiser ces résultats théoriques.

7.2 Approche numérique et théorèmes des schémas de discrétisation

Lorsque les solutions analytiques sont difficiles à obtenir ou inexistantes, l'approche numérique devient primordiale. Les méthodes de discrétisation traduisent l'EDP en un système d'équations algébriques, permettant ainsi la simulation sur ordinateur.

7.2.1 Méthodes de discrétisation

Les principales stratégies de discrétisation comprennent :
— **Les différences finies** : ces méthodes remplacent les dérivées par des approximations discrètes sur un maillage régulier comme vu dans le cadre des schémas numériques à un pas.
— **Les éléments finis** : cette approche repose sur une décomposition du domaine en sous-domaines nommés éléments et sur l'utilisation de fonctions d'interpollation. Elle se prête particulièrement bien aux domaines complexes.
— **Les volumes finis** : orientées vers la conservation des flux, ces méthodes sont privilégiées dans la résolution des équations de conservation, notamment en mécanique des fluides.

Chaque méthode présente des compromis entre simplicité de mise en œuvre, précision, et coût de calcul.

7.2.2 Théorèmes de convergence et de stabilité

La validité d'un schéma numérique repose sur trois propriétés essentielles comme vu dans les chapitres précédents : la consistance, la stabilité et la convergence. Un résultat fondamental dans ce domaine est le théorème d'équivalence de Lax-Richtmyer :

Théorème 13. Théorème d'équivalence de Lax-Richtmyer
Pour un schéma numérique linéaire consistant approchant une EDP bien posée, la stabilité du schéma est une condition nécessaire et suffisante pour assurer la convergence vers la solution exacte.

Autrement dit, si le schéma est consistant et stable, alors il converge vers la solution continue du problème. Ainsi il existe une condition pour les schémas explicites appliqués aux EDP hyperboliques.

Théorème 14. Condition de Courant–Friedrichs–Lewy (CFL)
Soit un problème aux valeurs initiales pour une équation aux dérivées partielles linéaire hyperbolique bien posé et considérons un schéma numérique explicite de différences finies consistant approchant ce problème. Pour que le schéma soit convergent, il est nécessaire que le domaine numérique de dépendance recouvre le domaine analytique de dépendance de l'équation continue. Concrètement, si l'on note Δt le pas de temps, Δx le pas spatial et a la vitesse maximale de propagation de l'information dans le système, alors il faut que

$$\frac{|a|\Delta t}{\Delta x} \leq 1.$$

Cette inégalité exprime que, durant le pas de temps Δt, l'information ne doit pas parcourir une distance supérieure à la taille de la maille spatiale Δx, condition indispensable pour assurer la stabilité du schéma.

Pour comprendre ce théorème, utilisons un exemple.

7.2.3 Exemple numérique : l'équation des ondes

Prenons l'équation des ondes en une dimension :

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}.$$

Un schéma explicite classique pour cette équation est donné par :

$$\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{\Delta t^2} = c^2 \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}.$$

La stabilité de ce schéma requiert que le rapport

$$\frac{c\Delta t}{\Delta x}$$

respecte la condition CFL.

Nous abordons à présent la discrétisation numérique et l'étude de la stabilité, où l'approximation des dérivées par des différences finies telle que celle utilisée dans la méthode d'Euler joue un rôle essentiel.

7.3 Discrétisation numérique et stabilité

Dans un schéma explicite, comme Euler, on approxime les dérivées à l'aide de différences finies. Pour les EDP, cela se traduit par des expressions du type :

$$\frac{\partial u}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t}, \quad \frac{\partial u}{\partial x} \approx \frac{u_i^n - u_{i-1}^n}{\Delta x}$$

La stabilité d'un tel schéma dépend du choix des pas de temps et d'espace, ce que formalise le critère de Courant–Friedrichs–Lewy comme vu précédemment

7.3.1 Étude de cas : convection non linéaire 1D et mise en evidence du critère CFL

Un autre type d'équation aux dérivées partielles fondamental en physique est l'équation de convection non linéaire. Elle permet de modéliser la propagation d'une quantité transportée par son propre flux, par exemple un fluide ou une onde de choc. Elle s'écrit sous la forme :

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$$

Cette équation est une EDP du premier ordre de type hyperbolique.

Résolution numérique

Pour résoudre numériquement cette équation, on utilise une discrétisation par différences finies en avant dans le temps et en arrière pour l'espace ce qui donne :

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + u_i^n \frac{u_i^n - u_{i-1}^n}{\Delta x} = 0$$

Ce schéma est explicite. Ensuite, en isolant le terme inconnu u_i^{n+1}, on obtient :

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

Cette équation va nous permettre d'étudier concrètement l'influence du critère CFL sur la stabilité de ce schéma.

7.3.2 Implémentation

Le code est divisé en trois parties :

1. Définition du maillage
2. Calcul du nombre de courant CFL
3. Résolution

On initialise nos paramètres et définit le maillage.

```
#Parametres du maillage spatial
nx = 61                              #nombre de points de grille
dx = 2 / (nx - 1)                    #pas d'espace

#Parametres temporels
nt = 20                              #nombre de pas de temps
dt =  0.015                          #pas de temps
c = 1                                #vitesse fictive de propagation

#Condition initiale :
u = np.ones(nx)
u[int(0.5 / dx):int(1 / dx + 1)] = 2
u_initial = u.copy()
```

Listing 7.1 – Initialisation des paramètres du maillage et de la condition initiale

Ensuite nous on calcule

$$CFL = \frac{|a|\Delta t}{\Delta x}$$

```
CFL = np.max(np.abs(u)) * dt / dx
print(CFL)
```

Listing 7.2 – Calcul et vérification du nombre de Courant (CFL)

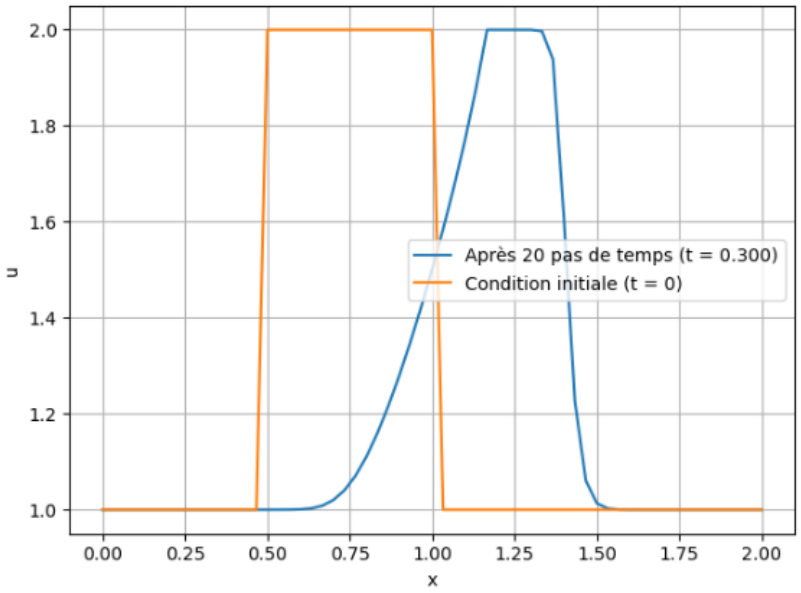
Enfin on résout :

```
for n in range(nt):
    u_n = u.copy()
    for i in range(1, nx - 1):
        if u_n[i] > 0:
            u[i] = u_n[i] - u_n[i] * dt/dx * (u_n[i] - u_n[i-1])
        else:
            u[i] = u_n[i] - u_n[i] * dt/dx * (u_n[i+1] - u_n[i])
```

Listing 7.3 – Résolution du schéma de convection non linéaire

7.3.3 Solutions

Solution stable avec CFL < 1



[h !]

FIGURE 7.1 – Solution stable (CFL = 0915)

Instabilité quand CFL > 1

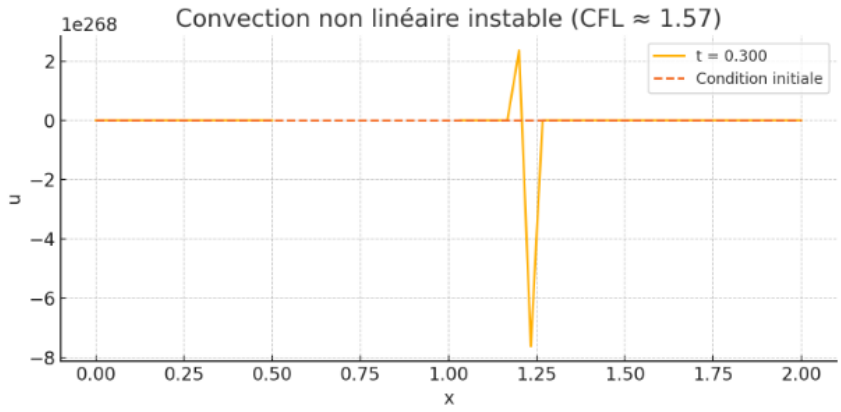


FIGURE 7.2 – Solution instable (CFL = 1.57)

7.3.4 Conclusion

Pour comprendre les phénomènes d'instabilité observés dans nos simulations, il faut revenir à la logique même de ce que nous programmons.

À chaque itération de la boucle temporelle, le schéma numérique utilise les informations disponibles sur l'état de l'onde pour estimer sa forme à l'instant suivant. Tant que le maillage est suffisamment fin, cette estimation est relativement fiable : la solution reste stable, et l'onde conserve ses caractéristiques, notamment sa forme de "marche carrée" dans notre exemple initial.

Mais si, au cours d'un pas de temps Δt, l'onde se déplace d'une distance plus grande que celle séparant deux points du maillage Δx, le schéma perd le fil de l'évolution correcte. Autrement dit, l'information "saute" des points de grille, et la simulation devient instable.

Ce comportement est précisément ce que décrit la condition de stabilité dite de Courant–Friedrichs–Lewy

(CFL). Elle impose que la vitesse maximale de propagation $u_{\max}$, multipliée par le pas de temps Δt, ne dépasse pas l'espacement spatial Δx :

$$\text{CFL} = \frac{u_{\max}\,\Delta t}{\Delta x}$$

Si cette condition n'est pas respectée (i.e. si le CFL > seuil critique), alors la solution se dégrade brutalement : apparition d'oscillations parasites, perte de forme, voire explosion numérique.

Ainsi, le nombre de Courant est un outil simple mais fondamental pour garantir la stabilité des schémas explicites. Il relie intimement le temps, l'espace et la physique du problème simulé, et doit être soigneusement pris en compte dès la conception du maillage.

En ayant connaissance des différentes notions, théorèmes, mécanismes d'implémentation et de discrétisation, nous pouvons désormais appliquer l'intégration projective.

7.4 Implémentation de la méthode d'intégration projective

Le code débute par une phase de micro-intégration, où l'on résout le problème sur de petits pas de temps afin de capturer les dynamiques rapides. Ensuite, une projection linéaire est réalisée pour extrapoler la solution sur une échelle temporelle plus grande.

```
while t < t_final:
    #Microintegration
    for i in range(k+1):
        u_n = u.copy()
        for j in range(1, nx - 1):
            if u_n[j] > 0:
                u[j] = u_n[j] - u_n[j] * dt / dx * (u_n[j] - u_n[j - 1])
            else:
                u[j] = u_n[j] - u_n[j] * dt / dx * (u_n[j + 1] - u_n[j])
        u_hist[i, :] = u.copy()

    #Projection lineaire
    u_dot = (u_hist[-1] - u_hist[-2]) / dt

    u = u_hist[-1] + T_macro * u_dot

    #Mise a jour du temps
    t += dt_project
```

Listing 7.4 – Résolution du schéma de convection non linéaire

Ensuite nous devons initialiser nos paramètres de la manière suivante :

1. **Maillage :**
   - $nx = 61$, le nombre de points.
   - $dx = \frac{2}{nx-1}$, le pas d'espace.
2. **Paramètres temporels :**
   - $nt = 300$, le nombre de pas de temps.
   - $\Delta t = 0.001$, le pas de temps.
   - $T_{\text{macro}} = 0.005$, la durée de l'étape de projection.
   - $k = 3$, le nombre de pas effectués dans la phase micro.

Les deux figures présentent la solution du problème de convection non linéaire après un temps $t = 0.3$. On observe que les deux méthodes produisent des résultats qualitativement similaires, avec une diffusion numérique modérée. Toutefois, l'intégration projective permet d'atteindre ce résultat avec un temps de calcul réduit de moitié. Cela démontre son efficacité pour les systèmes raides, en réduisant le nombre total de pas tout en conservant une précision acceptable.

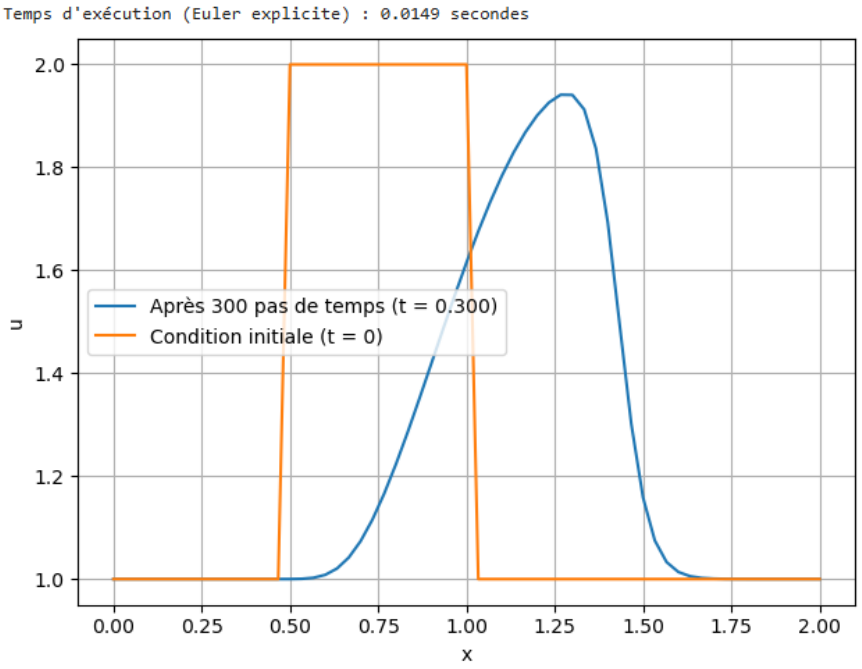


FIGURE 7.3 – Résultat obtenu avec le schéma d'Euler explicite

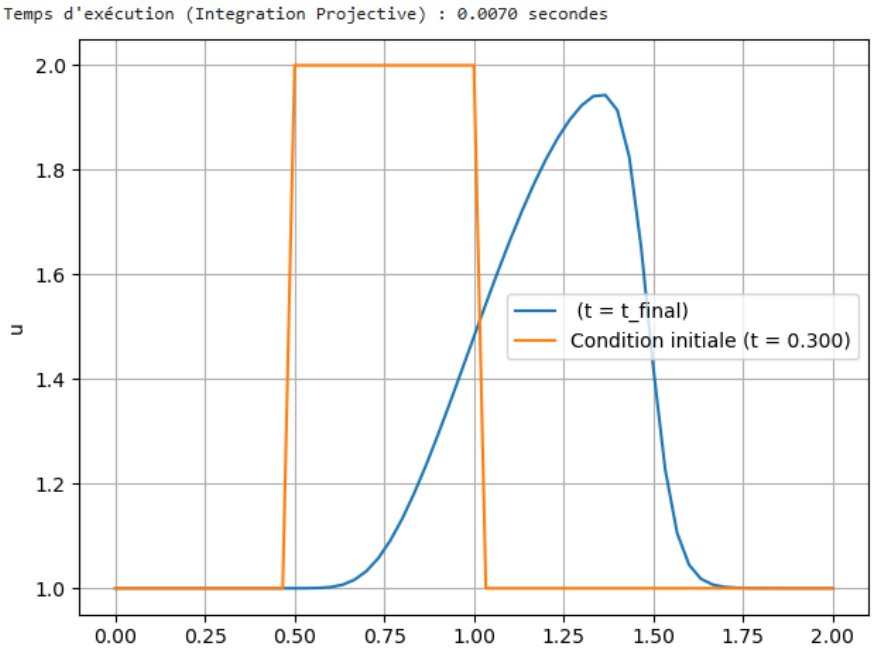


FIGURE 7.4 – Résultat obtenu avec l'intégration projective

Chapitre 8

Étude en 2D des équations de Navier-Stokes incompressibles

Les équations de Navier–Stokes occupent une place centrale dans la modélisation mathématique des phénomènes fluides. Elles décrivent l'évolution temporelle du mouvement d'un fluide newtonien soumis à des forces internes comme la viscosité et externes comme la gravité.

8.1 Introduction aux modèles incompressibles

Lorsqu'on considère un fluide incompressible, c'est-à-dire dont la densité est constante, le système d'équations se simplifie tout en conservant ses propriétés. Les modèles incompressibles repose sur deux équations fondamentales :

1. L'équation de conservation de la masse, l'incompressibilité :

$$\nabla \cdot \mathbf{u} = 0$$

qui exprime que le fluide ne subit ni dilatation ni compression avec $\mathbf{u} = \mathbf{u}(x,t) \in \mathbb{R}^d$ qui est le champ de vitesse du fluide, c'est-à-dire une fonction vectorielle qui donne à chaque point x de l'espace et à chaque instant t, la vitesse locale du fluide. En dimension 3, il s'écrit de la façon suivante où chaque composante représente la vitesse dans une direction donnée :

$$\mathbf{u}(x,t) = \begin{bmatrix} u_1(x,t) \\ u_2(x,t) \\ u_3(x,t) \end{bmatrix}$$

2. L'équation de conservation de la quantité de mouvement loi de Navier–Stokes :

$$\rho \left(\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} \right) = -\nabla p + \mu \Delta \vec{u} + \vec{f},$$

où

1. ρ est la masse volumique constante du fluide
2. p est la pression exercée localement
3. μ est la viscosité dynamique
4. $\vec{f}$ désigne les forces volumiques externes comme la gravité
5. $\Delta \vec{u}$ modélise la diffusion visqueuse via le laplacien

Cette équation, souvent appelée équation vectorielle des fluides newtoniens incompressibles, met en évidence la complexité des phénomènes régis par la non-linéarité du terme $(\vec{u} \cdot \nabla)\vec{u}$ et le couplage entre la pression et la vitesse. Sa résolution analytique reste difficile dans la plupart des cas, en particulier en dimension trois mais en dimension deux les équations de Navier-Stokes sont bien comprises.

8.2 Les équations de Navier-Stokes bidimensionnel

On travaille dans le plan (x, y) et on suppose que $w = 0$ et toutes les dérivées par rapport à z sont nulles. Ainsi, on a :

$$\vec{u} = (u(x, y, t), v(x, y, t))$$

et l'équation devient composante par composante :

1. Équation en x, le momentum horizontal

$$\rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) = -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

2. Équation en y, le momentum vertical

$$\rho\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) = -\frac{\partial p}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$$

et nous obtenons la condition d'incompressibilité suivante :

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

Maintenant que nous avons mis en évidence la manière dont la dynamique du fluide est décrite ses composantes de vitesse et la contrainte de divergence nulle. Nous voulons formuler une équation qui décrit la pression pour ce faire nous allons utiliser la célèbre équation de Poisson.

8.2.1 Équation de Poisson pour la pression

L'idée consiste à prendre la divergence des deux côtés de l'équation de Navier–Stokes :

$$\nabla \cdot \left(\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u}\right) = -\nabla^2 p/\rho + \nu \nabla \cdot \nabla^2 \vec{u}$$

En utilisant la condition d'incompressibilité :

$$\nabla \cdot \vec{u} = 0 \Rightarrow \nabla \cdot \left(\frac{\partial \vec{u}}{\partial t}\right) = 0, \quad \nabla \cdot \nabla^2 \vec{u} = \nabla^2(\nabla \cdot \vec{u}) = 0$$

Nous obtenons une équation de Poisson pour la pression :

$$\nabla^2 p = -\rho \nabla \cdot (\vec{u} \cdot \nabla \vec{u})$$

En 2D, cette divergence se développe explicitement de la manière suivante :

$$\nabla^2 p = -\rho\left[2\frac{\partial u}{\partial x}\frac{\partial u}{\partial x} + 2\frac{\partial v}{\partial y}\frac{\partial v}{\partial y} + \frac{\partial v}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial u}{\partial y}\frac{\partial v}{\partial x} + \frac{\partial v}{\partial x}^2 + \frac{\partial u}{\partial y}^2\right]$$

Cette équation permet de retrouver le champ de pression à chaque instant à partir du champ de vitesse. De cette manière, ces équations couplées forment un système non linéaire qui nécessite des méthodes numériques pour être résolu. Dans les chapitres suivants, nous les discrétiserons pour simuler l'écoulement dans une cavité.

8.3 Application à un problème : l'écoulement de cavité entraînée

Le problème de l'écoulement de cavité est un cas test fondamental en mécanique des fluides numérique. On remplit une cavité carrée (de côté L) d'un fluide visqueux incompressible, les trois parois inférieure, gauche et droite sont fixes (vitesse nulle), tandis que la paroi supérieure se déplace à vitesse constante U_0 dans la direction horizontale.

8.3.1 Modélisation du problème et hypothèses

Ce mouvement impose un cisaillement au fluide qui génère un écoulement complexe caractérisé par un vortex principal au centre, ainsi que des tourbillons secondaires aux coins. Maintenant que nous avons une idée de ce problème physique, nous imposons des hypothèses pour mener à bien notre application :

— Fluide incompressible ($\nabla \cdot \vec{u} = 0$)
— Écoulement bidimensionnel ($\vec{u} = (u(x, y, t), v(x, y, t))$)
— Pas de forces volumiques (gravité négligée)
— Viscosité constante (μ)

8.3.2 Système d'équations avant discrétisation

En ayant nos hypothèses nous devons traduire mathématiquement notre cas pour cela nous allons utilisé des équations différentielles aux dérivées partielles en coordonnées cartésiennes (x, y) :

$$\begin{cases} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) & \text{la quantité de mouvement selon } x \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) & \text{la qantité de mouvement selon } y \\ \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = \rho\left(\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) - \left(\frac{\partial u}{\partial x}\right)^2 - 2\frac{\partial u}{\partial y}\frac{\partial v}{\partial x} - \left(\frac{\partial v}{\partial y}\right)^2\right) & \text{l'équation de Poisson pour la pression} \end{cases}$$

Ce système est maintenant prêt à être discrétisé dans le temps et l'espace, afin d'être résolu numériquement sur une grille cartésienne.

8.3.3 Discrétisation des équations

Pour résoudre numériquement le système, nous utilisons une grille cartésienne régulière (x_i, y_j) de pas Δx, Δy et un pas de temps Δt. Chaque variable dépendante comme la vitesse u, v et pression p est approchée aux points de la grille et aux instants discrets : $u^n_{i,j} \approx u(x_i, y_j, t^n)$.

Discrétisation de l'équation du moment selon x

Pour discrétiser l'équation du moment dans la direction x, nous allons séparer et approximer chaque terme à l'aide de schémas numériques adaptés.

1. **Temps** : Pour approximer la dérivée temporelle, nous utilisons un schéma d'Euler explicite. Cela nous permet d'exprimer le taux de variation de u à un point (i, j) en fonction des valeurs aux instants n et $n + 1$. Ainsi la formule employée est :
$$\left.\frac{\partial u}{\partial t}\right|^n_{i,j} \approx \frac{u^{n+1}_{i,j} - u^n_{i,j}}{\Delta t}$$
2. **Espace - Convection** : Pour modéliser la convection, qui traduit le transport de la quantité de mouvement par le fluide, nous utilisons une approximation de première ordre décentrée à gauche. L'idée est de prendre l'information provenant du point d'avant pour estimer la dérivée spatiale. Nous écrivons ainsi :
$$\left.\frac{\partial u}{\partial x}\right|^n_{i,j} \approx \frac{u^n_{i,j} - u^n_{i-1,j}}{\Delta x}, \quad \left.\frac{\partial u}{\partial y}\right|^n_{i,j} \approx \frac{u^n_{i,j} - u^n_{i,j-1}}{\Delta y}$$
3. **Diffusion - le Laplacien** : Le terme de diffusion est caractérisé par des dérivées secondes, que nous discrétisons à l'aide d'une différence finie centrée en trois points pour garantir une précision d'ordre 2. Pour la dérivée seconde par rapport à x et y, nous avons :
$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} \approx \frac{u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j}}{\Delta x^2}, \quad \left.\frac{\partial^2 u}{\partial y^2}\right|_{i,j} \approx \frac{u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}}{\Delta y^2}$$
4. **Gradient de pression** : L'approximation du gradient de pression en x est également réalisée via une différence finie centrée. Cette méthode d'ordre 2 permet d'obtenir une estimation précise de la variation spatiale de la pression :
$$\left.\frac{\partial p}{\partial x}\right|_{i,j} \approx \frac{p^n_{i+1,j} - p^n_{i-1,j}}{2\Delta x}$$

En substituant ces approximations dans l'équation du moment selon x, on obtient la relation discrète suivante :

$$\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} + u_{i,j}^n \frac{u_{i,j}^n - u_{i-1,j}^n}{\Delta x} + v_{i,j}^n \frac{u_{i,j}^n - u_{i,j-1}^n}{\Delta y}$$

$\Longleftrightarrow$

$$-\frac{1}{\rho}\frac{p_{i+1,j}^n - p_{i-1,j}^n}{2\Delta x} + \nu\left(\frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} + \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2}\right)$$

Discrétisation de l'équation du moment selon y

De la même manière nous obtenons :

$$\frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta t} + u_{i,j}^n \frac{v_{i,j}^n - v_{i-1,j}^n}{\Delta x} + v_{i,j}^n \frac{v_{i,j}^n - v_{i,j-1}^n}{\Delta y}$$

$\Longleftrightarrow$

$$-\frac{1}{\rho}\frac{p_{i,j+1}^n - p_{i,j-1}^n}{2\Delta y} + \nu\left(\frac{v_{i+1,j}^n - 2v_{i,j}^n + v_{i-1,j}^n}{\Delta x^2} + \frac{v_{i,j+1}^n - 2v_{i,j}^n + v_{i,j-1}^n}{\Delta y^2}\right)$$

Enfin, il nous reste à discrétiser l'équation de Poisson.

Discrétisation de l'équation de Poisson (pression)

Pour approximativement calculer le Laplacien de la pression, nous utilisons des différences finies centrées sur trois points. Ainsi, la dérivée seconde par rapport à x est discrétisée par

$$\frac{p_{i+1,j}^n - 2p_{i,j}^n + p_{i-1,j}^n}{\Delta x^2},$$

et celle par rapport à y par

$$\frac{p_{i,j+1}^n - 2p_{i,j}^n + p_{i,j-1}^n}{\Delta y^2}.$$

La somme de ces deux termes représente l'opérateur Laplacien complet appliqué à p et fournit une approximation d'ordre 2, garantissant ainsi une symétrie autour du point central, ce qui est crucial pour la stabilité et la précision globale du schéma, ainsi le second membre s'écrit :

$$\frac{p_{i+1,j}^n - 2p_{i,j}^n + p_{i-1,j}^n}{\Delta x^2} + \frac{p_{i,j+1}^n - 2p_{i,j}^n + p_{i,j-1}^n}{\Delta y^2}$$

Maintenant nous devons le discrétiser. Dans un premier temps nous nous occupons de la variation temporelle de la divergence de la vitesse. Elle est approximée par une différence finie centrée pour les dérivées spatiales et un schéma d'Euler explicite pour le temps :

$$\frac{1}{\Delta t}\left(\frac{u_{i+1,j}^n - u_{i-1,j}^n}{2\Delta x} + \frac{v_{i,j+1}^n - v_{i,j-1}^n}{2\Delta y}\right).$$

Ensuite les termes non linéaires dues à la variation spatiale des vitesses sont discrétisées à l'aide de différences centrées pour garantir une précision d'ordre 2. On obtient ainsi pour le carré de la dérivée de u en x :

$$-\left(\frac{u_{i+1,j}^n - u_{i-1,j}^n}{2\Delta x}\right)^2,$$

pour le terme de produit croisé entre u et v :

$$-2\left(\frac{u_{i,j+1}^n - u_{i,j-1}^n}{2\Delta y}\right)\left(\frac{v_{i+1,j}^n - v_{i-1,j}^n}{2\Delta x}\right),$$

et enfin pour le carré de la dérivée de v en y :

$$-\left(\frac{v_{i,j+1}^n - v_{i,j-1}^n}{2\Delta y}\right)^2.$$

La combinaison de ces deux contributions donne la forme complète du second membre :

$$\rho\left[\frac{1}{\Delta t}\left(\frac{u^n_{i+1,j}-u^n_{i-1,j}}{2\Delta x}+\frac{v^n_{i,j+1}-v^n_{i,j-1}}{2\Delta y}\right)-\left(\frac{u^n_{i+1,j}-u^n_{i-1,j}}{2\Delta x}\right)^2-2\left(\frac{u^n_{i,j+1}-u^n_{i,j-1}}{2\Delta y}\right)\left(\frac{v^n_{i+1,j}-v^n_{i-1,j}}{2\Delta x}\right)-\left(\frac{v^n_{i,j+1}-v^n_{i,j-1}}{2\Delta y}\right)^2\right]$$

Après avoir établi en détail la discrétisation de l'équation de Poisson et des autres équations du modèle, nous disposons désormais d'un cadre théorique pour simuler l'écoulement d'un fluide incompressible. Nous allons désormais implémenter en Python l'ensemble des schémas numériques développés précédemment.

8.3.4 Implémentation numérique

Avant d'implémenter kes équations de Navier–Stokes en 2D dans le cas d'un fluide incompressible nous devons nous assurer de traiter chaque composante et de bien initialiser leurs mises à jours.

Mise à jour de la composante horizontale de la vitesse (u)

On implémente une fonction qui met à jour le champ de vitesse u à chaque itération temporelle. Elle utilise un schéma explicite de type décentré à droite pour les termes d'espaces, et un schéma centré à trois points pour les termes diffusifs.

```
def velocite_u(u, dx, dy, dt, rho, p, un, vn):
    u[1:-1, 1:-1] = (un[1:-1, 1:-1]-
                     un[1:-1, 1:-1] * dt / dx *
                     (un[1:-1, 1:-1] - un[1:-1, 0:-2]) -
                     vn[1:-1, 1:-1] * dt / dy *
                     (un[1:-1, 1:-1] - un[0:-2, 1:-1]) -
                     dt / (2 * rho * dx) * (p[1:-1, 2:] - p[1:-1, 0:-2]) +
                     nu * (dt / dx**2 *
                     (un[1:-1, 2:] - 2 * un[1:-1, 1:-1] + un[1:-1, 0:-2]) +
                     dt / dy**2 *
                     (un[2:, 1:-1] - 2 * un[1:-1, 1:-1] + un[0:-2, 1:-1])))
    return u
```

Listing 8.1 – Implémentation de l'équation pour la vitesse u

Mise à jour de la composante composante verticale de la vitesse (v)

De manière similaire, on met à jour la composante verticale v :

```
def velocite_v(v, dx, dy, dt, rho, p, un, vn):
    v[1:-1,1:-1] = (vn[1:-1, 1:-1] -
                     un[1:-1, 1:-1] * dt / dx *
                     (vn[1:-1, 1:-1] - vn[1:-1, 0:-2]) -
                     vn[1:-1, 1:-1] * dt / dy *
                     (vn[1:-1, 1:-1] - vn[0:-2, 1:-1]) -
                     dt / (2 * rho * dy) * (p[2:, 1:-1] - p[0:-2, 1:-1]) +
                     nu * (dt / dx**2 *
                     (vn[1:-1, 2:] - 2 * vn[1:-1, 1:-1] + vn[1:-1, 0:-2]) +
                     dt / dy**2 *
                     (vn[2:, 1:-1] - 2 * vn[1:-1, 1:-1] + vn[0:-2, 1:-1])))
    return v
```

Listing 8.2 – Implémentation de l'équation pour la vitesse v

Maintenant nous devons implémenter et construire numériquement le second membre de l'équation de Poisson qui découle de la contrainte d'incompressibilité $\nabla \cdot \vec{u} = 0$ et permet le couplage pression-vitesse.

Construction du second membre de l'équation de Poisson pour la pression

En prenant notre discrétisation de l'équation de Poisson :

```
def Poisson(b, rho, dt, u, v, dx, dy):
    b[1:-1, 1:-1] = (rho * (1 / dt *
                     ((u[1:-1, 2:] - u[1:-1, 0:-2]) /
                      (2 * dx) + (v[2:, 1:-1] - v[0:-2, 1:-1]) / (2 * dy)) -
                     ((u[1:-1, 2:] - u[1:-1, 0:-2]) / (2 * dx))**2 -
                       2 * ((u[2:, 1:-1] - u[0:-2, 1:-1]) / (2 * dy) *
                            (v[1:-1, 2:] - v[1:-1, 0:-2]) / (2 * dx))-
                           ((v[2:, 1:-1] - v[0:-2, 1:-1]) / (2 * dy))**2))
    return b
```

Listing 8.3 – Implémentation du second membre de l'équation de Poisson

Nous devons résoudre cette équation, pour cela nous nous réferons aux tutoriels de CFD en Python 12 Steps to Navier–Stokes développé par Lorena Barba et son équipe.

Résolution de l'équation de Poisson pour la pression

La pression est obtenue via une méthode itérative de relaxation de type Jacobi, avec un critère d'arrêt basé sur la norme L^1. Ainsi le code associé, est tiré du tutoriel "CFD Python : 12 Steps to Navier–Stokes" développé par Lorena Barba :

```
def pression_poiss(p, dx, dy, b, l1norm_target):
    pn = np.empty_like(p)
    pn = p.copy()
    l1norm = 1
    small = 1e-8
    while l1norm > l1norm_target:
        pn = p.copy()
        p[1:-1, 1:-1] = (((pn[1:-1, 2:] + pn[1:-1, 0:-2]) * dy**2 +
                          (pn[2:, 1:-1] + pn[0:-2, 1:-1]) * dx**2) /
                          (2 * (dx**2 + dy**2)) -
                          dx**2 * dy**2 / (2 * (dx**2 + dy**2)) *
                          b[1:-1,1:-1])
        p[:, -1] = p[:, -2] # dp/dx = 0 at x = 2
        p[0, :] = p[1, :]   # dp/dy = 0 at y = 0
        p[:, 0] = p[:, 1]   # dp/dx = 0 at x = 0
        p[-1, :] = 0        # p = 0 at y = 2
        l1norm = (np.sum(np.abs(p[:]-pn[:])) / (np.sum(np.abs(pn[:]))+small))
    return p
```

Listing 8.4 – Résolution de l'équation de Poisson pour p

Enfin, il nous reste à initialiser nos paramètres.

Initialisation des paramètres

Nous utilisons une discrétisation spatiale uniforme sur un domaine carré de taille unitaire. Les paramètres du maillage sont les suivants :

— $n_x = n_y = 129$, le nombre de points de discrétisation dans les directions x et y
— $L_x = L_y = 1$, la longueur du domaine dans chaque direction (géométrie carrée)
— $\Delta x = \frac{L_x}{n_x - 1}$ et $\Delta y = \frac{L_y}{n_y - 1}$, le pas d'espace en x et y

Les autres paramètres physiques sont :

— $\rho = 1$, la densité du fluide
— $\nu = 0{,}01$, la viscosité cinématique (fluide modérément visqueux)
— $c = 1$, la vitesse du couvercle supérieur (paroi entraînante)
— $\Delta t = 0{,}001$, le pas de temps

Le nombre de Reynolds, qui nous dit si l'écoulement sera laminaire ou turbulent est donné par :

$$\mathrm{Re} = \frac{c \cdot L}{\nu} = \frac{1 \cdot 1}{0{,}01} = 100 \quad \text{ce qui correspond à un écoulement laminaire dans la cavité}$$

8.3.5 Simulation et résultats

En compilant nos différentes parties de code, nous obtenons la visualisation du champ d'écoulement laminaire (Re= 100) dans la cavité entraînée. Nous voyons que la figure (a) présente le champ de pression codé en couleurs, allant du bleu (basse pression) au rouge (haute pression), accompagné des vecteurs de vitesse qui indiquent la direction et l'intensité du flux dans la cavité. Alors que la figure (b) superpose les lignes de courant sur la distribution de la vitesse horizontale, ce qui met en évidence la formation de vortex ce qui est commun pour un écoulement laminaire.

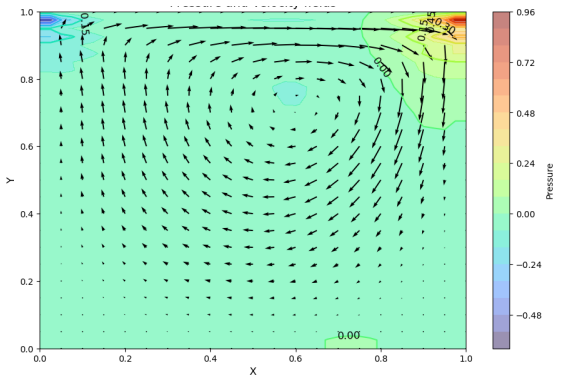


FIGURE 8.1 – Champ de pression et champ de vitesse (flèches)

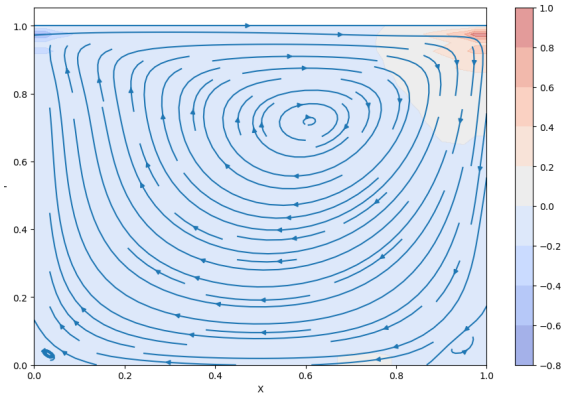


FIGURE 8.2 – Lignes de courant superposées à la vitesse horizontale

Après avoir obtenu des résultats satisfaisant avec la simulation classique (cf. figures ci-dessus), on constate que le schéma d'Euler explicite nécessite un nombre très élevé d'itérations et donc un temps de calcul conséquent.

Pour pallier ces problèmes, nous allons d'implémenter la méthode d'intégration projective.

8.4 Implémentation de la méthode d'intégration projective

To be continued...

Bibliographie

[1] Thomas Rey,
Cours sur les Équations Différentielles et EDP.
Université Nice Côte d'Azur - Département de mathématiques - M1 IM

[2] Thomas Rey, R. Bailo, W. Melis, G. Samaey
Projective integration methods for multiscale collisional kinetic equations.
Université Nice Côte d'Azur

[3] Florent Berthelin,
Équations Différentielles.
Cassini Ed, 2017 ISBN 2842252292

[4] Afeintou Sangam,
Cours sur les Schémas Numériques des Équations Différentielles.
Université Nice Côte d'Azur - Département de mathématiques - L3 Mathématiques

[5] Claire Scheid,
Cours d'Analyse Numérique 2.
Université Nice Côte d'Azur - Département de mathématiques - L3 Mathématiques

[6] C. W. Gear and Ioannis G. Kevrekidis,
Projective Methods for Stiff Differential Equations : Problems with Gaps in Their Eigenvalue Spectrum, 2002.

[7] Ward Melis,
Projective Integration for Hyperbolic Conservation Laws and Multiscale Kinetic Equations.
Thèse : Doctor in Engineering Science – PhD en Informatique),
2017 KU Leuven – Faculty of Engineering Science.

[8] SIAM Journal on Scientific Computing,
Projective Integration : A Fast and Efficient Numerical Method for Stiff Differential Equations.
SIAM Journal on Scientific Computing, **24**(1) : 16–40

[9] E. Hairer, S. P. Nørsett, and G. Wanner,
Solving Ordinary Differential Equations I : Nonstiff Problems (2^{nd} ed.).
Springer-Verlag, 2009.
ISBN 978-3-540-92251-4.

[10] *Multiscale Computational Methods for Stiff Problems*,
in *Computational Methods in Applied Sciences*, Vol. 4, Springer-Verlag, pp. 99–116,
ISBN 978-3-540-78864-9.

[11] Jean-Pierre Demailly,
Analyse Numérique et Équations Différentielles ($4^{\text{ème}}$ éd.).
Edp Sciences, 2016 ISBN 2759819264

[12] Barba, Lorena A., and Forsyth, Gilbert F. (2018)
CFD Python : the 12 steps to Navier-Stokes equations
Journal of Open Source Education, 1(9),21 et https ://github.com/barbagroup/cfd

[13] Thomas Giarrizzi, Yuran Wang
Non-uniqueness of Turbulent Solutions of the Navier-Stokes Equation in Dimension N = 3
ENS PSL,2024

[14] Jacques-Louis Lions and Giovanni Prodi
Un théorème d'existence et unicité dans les équations de Navier- Stokes en dimension 2.
C. R.Acad. Sci., Paris, 248 :3519–3521, 1959.

[15] Alain Ciffréo , Francois Peters
Cours sur la mécanique des milieux continus
Université Nice Côte d'Azur - Département de physique - L3 Physique