RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de L'enseignement Supérieur et de la Recherche Scientifique
UNIVERSITÉ IBN KHALDOUN TIARET
FACULTÉ DE MATHÉMATIQUES ET DE L'INFORMATIQUES
Département de Mathématiques

MÉMOIRE DE MASTER

Spécialité :
« Mathématiques »

Option :
« Analyse fonctionnelle et équations différentielles »

Présenté Par :
Bouchiba Ahmed et Boulebene Ahlam

Sous L'intitulé :

Etude numérique des équations différentielles

Soutenu publiquement le 01 / 06 / 2025 à Tiaret
Devant le jury composé de :

Mr Bekiri Mohamed	MCA	Université de MASCARA	Président
Mr Sebih Mohammed Elamine	MCB	Université de MASCARA	Examinateur
Mr Benkhaled Abdelkader	MCA	Université de MASCARA	Encadrant

Année universitaire : 2024/2025

❖ remerciement

Nous tenons tout d'abord à remercier Allah le tout puissant et miséricordieu, qui nous a donné la force et la patience d'accomplir et réaliser ce travail :Nous voudrions remercier tout particulièrementnotre encadreur Mr.Ben khaled Aek pour l'offre de son temps et sa patience . Ses conseils remarques et critiques ont toujours été une aide précieuse pour nous Nousavons beaucoups appris en travaillant avec lui, c'est un plaisir pour nous.

Nos respectueux remerciements à Mr.Bekiri Mohamed pour l'honneur qu'il nous fait en acceptant de présiderle juryde soutenance de cette mémoire. Nosremerciements s'adressent aussi a Mr.Sebih Mohammed Elamine pour l'honneur qu'il nous a fait en acceptant de faire partie du jury et pour avoir accepté d'examiner notre travail.

❖ Dédicace

Je dédie ce travail

A ma chère mère et ma chère femmeQuoi que je fasse ou que je dise, je ne saurais point vous remercier comme il se doit.Votre affection mecouvre, votre bienveillance me guide et votre présences à mes côtés a toujours été ma source de force pour affronter les différents obstacles.

A mes frères pour leur présence quotidienne.

A mes Enfants oussim ; ouanis , soltana. Je leur souhaite tout le meilleur dans leurs études.

Que Dieu les protège et leurs offre la chance et le bonheur.

❖BOUCHIBA.A

❖ Dédicace

C'est grâce à Dieu Le Tout-Puissant qui m'a donné la force et la patience et qui m'a aidé à surmonter les obstacles de ce parcours que j'ai pu atteindre ce succès pour commencer je dédit cette rèussite :

À celui qui est toujours là à mes côtés pour me soutenir ,et qui me donne assez de force et de courage pour que je puisse avancer , Mon mari «B.Ahmed ».

À celui qui sacrifie sa vie pour mon confort et mon bonheur, à mon seul soutien de qui je n'ai pas peur qu'il va me laisser tomber un jour cher (père + mére)

« Taher + Nadjia »

À mes soeurs Dr.amira et Pr.Asmahan et Pr.hannan qui ne me laissent pas seule face aux trahisons de la vie ,et à mes chers frères 'Amine + Bilal'

À tous ceux qui m'ont aidé tout au long de mon parcours, je vous remercie infiniment

❖ Boulebene.A

Table Des Matières

Introduction

Les équations différentielles jouent un rôle fondamental dans la modélisation des phénomènes physiques, biologiques, économiques et d'ingénierie. Leur résolution permet de prédire l'évolution dans le temps de systèmes dynamiques complexes. Cependant, dans la majorité des cas concrets, ces équations ne peuvent pas être résolues analytiquement (c'est-à-dire de manière exacte). Il devient alors indispensable d'utiliser des méthodes numériques.

Le mémoire s'inscrit dans cette optique, en mettant l'accent sur :

La compréhension théorique des équations différentielles ordinaires (EDO) : définitions, types d'EDO, existence et unicité des solutions (théorème de Cauchy-Lipschitz).

Le traitement des problèmes de Cauchy : transformation des EDO d'ordre supérieur en systèmes d'ordre 1, utilisation de modèles mathématiques pour des phénomènes concrets (ex : chute libre dans un champ de pesanteur).

La mise en œuvre de méthodes numériques : telles que la méthode de Picard, la méthode d'Euler, les méthodes de Taylor, et les célèbres méthodes de Runge-Kutta, pour approximer les solutions des EDO.

Chapitre 01

Equations différentielles

1.0.1 Équations différentielles ordinaires

.

Dans ce chapitre, on donne quelques définitions. Soient I un ouvert de $\mathbb{R}$, E un espace de Banach sur $\mathbb{R}$ et $U_1, U_2, \cdots, U_n$ des ouverts de E. Les équations

différentielle
ordinaires, seront notées en abrégé EDO dans la suite.

Définition 1.0.1 *Une équation différentielle sur l'espace de Banach E est une équation de la forme*

$$F(t, x, x', x'', \cdots, x^{(n)}) = 0, \tag{1.1}$$

où n est un entier non nul appelé l'ordre de l'équation, F est une fonction donnée de $(n+2)$ variables supposée régulière sur $I \times U_1 \times U_2 \times \cdots \times U_n$, x est la fonction inconnue de I dans l'espace de Banach E et $x, x', x'', \cdots, x^{(n)}$ sont ses dérivées successives. Plus précisément le problème est de trouver un intervalle ouvert I de $\mathbb{R}$ et une fonction $x : t \mapsto x(t)$ dérivable sur cet intervalle jusqu'à l'ordre n et vérifiant l'équation

$$\forall t \in I, \quad F(t, x, x', x'', \cdots, x^{(n)}) = 0, \tag{1.2}$$

cette équation est de forme très générale, en pratique, on préférera travailler avec des équations plus particulières dites du type explicite, pour lesquelles il existe une fonction H régulière sur $I \times U_1 \times U_2 \times \cdots \times U_{n-1}$ tel que

$$x^{(n)} = H(t, x, x', x'', \cdots, x^{(n-1)}). \tag{1.3}$$

Exemple 1.0.1

$$F(t, x, x', x'') = 0,$$

est d'ordre 2.

F(t, x, x', x'', ...,x(n-1))=0

est d'ordre n.

1.0.2 Equations différentielles linéaires

Définition 1.0.2. *Une EDO d'ordre n est linéaire si elle est de la forme*

$$a_n(t)x^{(n)}(t) + a_{n-1}(t)x^{(n-1)}(t) + \cdots + a_0(t)x(t) = g(t), \tag{1.4}$$

avec tous les $x^{(i)}$ de degré 1 et tous les coefficients dépendant au plus de t, si g est nulle, alors l'équation est dite homogène, ou sans second mombre. L'équation différentielle suivante

$$a_n(t)x^{(n)}(t) + a_{(n-1)}(t)x^{(n-1)}(t) + \cdots + a_0(t)x(t) = 0, \tag{1.5}$$

est appelée l'équation différentielle homogène associée a (1.4)
Si $a_j(t)$, $0 \leq j \leq n$, sont des constantes, on parle d'équation différentielle linéaire acoefficients constants.

Définition 1.0.3. *Soient $x : I \longrightarrow U$ et $\tilde{x} : \tilde{I} \longrightarrow \tilde{U}$ deux solutions de l'équation différentielle. On dit que $\tilde{x}$ est un prolongement de x si : $I \subset \tilde{I}$ et*

$$\tilde{x}(t) = x(t), \quad \textit{pour tout } t \in I.$$

Définition 1.0.4. *Soient I_1 et I_2 , deux intervalles sur $\mathbb{R}$, tels que $I_1 \subset I_2$.*
On dit qu'une solution (x, I) est maximale dans I_2 si et seulement si x n'admet pas de prolongement $(\tilde{x}, \tilde{I})$ solution de l'équation différentielle telle que $I_1 \subsetneq \tilde{I} \subset I_2$ (on verra mème plus tard que I_1 est nécessairement ouvert).

Définition 1.0.5. *Soit I un intervalle inclus dans $\mathbb{R}$. Une solution (x, I) est dite globale dans I si elle est définie sur l'intervalle I tout entier.*

Définition 1.0.6. • *On appelle solution générale de l'EDO, toute fonction $\Psi = \Psi(t, c)$ qui dépend d'une constante arbitraire c et tel que pour toute valeurs de c la fonction Ψ vérifie identiquement l'EDO, en tout points de $]t_0, t_1[$ qui est le domaine de définition de Ψ.*

• *On appelle solution particulière de l'EDO toute solutions déduite de la solution générale en donnant des valeurs concrètes a la constante arbitraire c, une solution*

particulière s'écrit :

$$x(t) = \Psi(t, c = c_0),$$

c est un constante, pour chaque valeur donner a c on a une solution particulière trouvé on a en chaque point de $]t_1, t_2[$ *l'unicité de la solution.*

1.0.3 Différents types d'équations

Définition 1.0.7. *Une équation différentielle ordinaire, également notée EDO , d'ordre n est une relation entre la variable réelle t, une fonction inconnue* $t \longmapsto x(t)$ *et ses dérivées* $x^{'}, x^{''}, \cdots, x^n$ *au point t définie par*

$$F(t, x, x^{''}, ..., x^n) = 0 \tag{1.6}$$

ou F n'est pas indépendante de sa dernière variable x^n*. On prendra t dans un intervalle I de* $\mathbb{R}$ *(I peut être* $\mathbb{R}$ *tout entier). La solution x en général sera a valeur dans* $\mathbb{R}^n$*,* $n \in \mathbb{N}^*$ *où n sera le plus souvent égale à* 1*, ou* 3*, On dit que cette équation est scalaire si F est à valeur dans* $\mathbb{R}$*.*

Exemple 1.0.2. *l'équation différentielle*

$$x^{'} + x^2 - t = 0,$$

est du premier ordre avec $V = \mathbb{R}^3$ *et* $F(t, x, x^{'}) = x^{'} + x^2 - t$*.*

Définition 1.0.8. *On appelle équation différentielle normale d'ordre n toute équation de la forme*

$$x^n = f(t, x, x^{''}, ..., x^{n-1}).$$

Définition 1.0.9. *On appelle équation différentielle autonome d'ordre n toute équations de la forme*

$$x^n = f(x, x^{''}, ..., x^{n-1}).$$

Autrement dit, f ne dépend pas explicitement de t.

Définition 1.0.10. *Une EDO de type* (1.6) *d'ordre n est linéaire si elle est de la forme*

$$a_n(t)x^n(t) + a_{n1}(t)x^{n-1}(t) + ... + a_1(t)x^{'}(t) + a_0(t)x(t) = g(t),$$

avec tous les x^i *de degré* 1 *et tous les coefficients dépendant au plus de t.*

Exemple 1.0.3. *l'équation* $x^{"} - 2x^{'}(t) + x = 0$ *est une équation ordinaire linéaire d'ordre deux à coefficient constantes.*

1.1 Solutions d'une équation différentielle

Définition 2.1.1. *On appelle solution d'une équation différentielle d'ordre n sur certain intervalle I de $\mathbb{R}$, toute fonction x définie sur cet intervalle I, n fois dérivable en tout point de I et qui vérifie cette équation différentielle sur I.*

On notera en général cette solution (x, I). Si I contient sa borne inférieur notée a (respectivement sa borne supérieure b), ce sont des dérivées à droite (respectivement à gauche) qui interviennent au point $t = a$ (respectivement $t = b$).

Intégrer une équation différentielle consiste à déterminer l'ensemble de ses solutions.

1.1.1 Solution maximale

Nous introduisons d'abord le concept de prolongement d'une solution. L'expression solution maximale est alors entendue implicitement au sens de la relation d'ordre fournie par le prolongement des solutions.

Définition 1.1.2. *On dit qu'une solution $x : I \to \mathbb{R}^n$ est maximale si x n'admet pas de prolongement $\tilde{x} : \tilde{I} \to \mathbb{R}^n$ avec $\tilde{I} \supsetneq I$.*

Théorème1.1.1. *Toute solution x se prolonge en une solution maximales $\tilde{x}$ (pas nécessairement unique).*

Preuve 1. • *Supposons que x soit définie sur un intervalle $I = |a, b|$ (cette notation désigne un intervalle ayant pour bornes a et b, incluses ou non dans I).*
Il suffira de montrer que x se prolonge en une solution $\tilde{x} : |a, \tilde{b}| \to \mathbb{R}^n (\tilde{b} \geq b)$ maximale à droite, c'est-à-dire qu'on ne pourra plus prolonger $\tilde{x}$ au delà de $\tilde{b}$. Le même raisonnement s'appliquera à gauche.
Pour cela, on construit par récurrence des prolongements successifs $x_{(1)}, x_{(2)}, \cdots$ de x avec $x_{(k)} : |a, b_k[\to \mathbb{R}^n$. On pose $x_{(1)} = x$, $b_1 = b$. Supposons $x_{(k-1)}$ déjà construite pour un indice $k \geq 1$. On pose alors

$$c_k = \sup\{c; x_{(k-1)};$$

se prolonge sur $|a, c[\}$.
On a $c_k \geq b_{k-1}$. Par définition de la borne supérieure, il existe b_k tel que $b_{k-1} \leq b_k \leq c_k$ et un prolongement $x_{(k)} : |a, b_k[\to R^n$ de $x_{(k-1)}$ avec b_k arbitrairement voisin de c_k, en particulier, on peut choisir

$$c_k - b_k < \frac{1}{k} \quad si \quad c_k < +\infty,$$

$$b_k > k \quad si \quad c_k = +\infty.$$

La suite (c_k) *est décroissante, car l'ensemble des prolongements de* $x_{(k-1)}$ *contient l'ensemble des prolongements de* $x_{(k)}$*, au niveau des bornes supérieures on a donc* $c_k \geq c_{k+1}$*. Si* $c_k < +\infty$ *à partir d'un certain rang, les suites*

$$b_1 \leq b_2 \leq \cdots \leq b_k \leq \cdots \leq c_k \leq c_{k-1} \leq \cdots \leq c_1;$$

sont adjacentes, tandis que si $c_k = +\infty$ *quel que soit* k*, on a* $b_k > k$*. Dans les deux cas, on voit que*

$$\tilde{b} = \lim_{k \to +\infty} b_k = \lim_{k \to +\infty} c_k.$$

Soit $\tilde{x} : |a, \tilde{b}| \to \mathbb{R}^n$ *le prolongement commun des solutions* $x_{(k)}$*, éventuellement prolongé au point* $\tilde{b}$ *si cela est possible. Soit* $z : |a, c| \to \mathbb{R}^n$ *un prolongement de* $\tilde{x}$*. Alors* z *prolonge* $x_{(k-1)}$ *et par définition de* c_k *il s'ensuit* $c \leq c_k$*. A la limite il vient* $c \leq \tilde{c}$*, ce qui montre que la solution* $\tilde{x}$ *est maximale à droite.*

1.2 Fonction lipschitzienne

Soit $\Omega \subset I \times \mathbb{R}^n$, f une application définie de Ω dans $\mathbb{R}^n$.

Définition 1.2.1. *L'application* f *est dite lipschitzienne par rapport à la deuxième variable sur* Ω *s'il existe une constante* k*, appelée la constante de Lipschitz de* f*, telle que :*

$$\forall x \in I, \ \forall y, z \in \mathbb{R}^n, \quad \|f(x, y) - f(x, z)\| \leq k\|y - z\|.$$

la fonction $x \longmapsto f(t, x)$ *est localement lipschitzienne pour tout* $t \in I$*, c'est-à-dire que pour tout* $J \subset I$ *compact, pour tout* $O \subset \Omega$ *compact, il existe une constante* $\gamma = \gamma(J, O) > 0$ *telle que*

$$\|f(t, x) - f(t, y)\| \leq \gamma \|x - y\|, \quad \forall t \in J, \ \forall x, y \in O. \tag{1.7}$$

Remarque 1.2.1. *On dit que* f *est localement lipschitzienne par rapport à la deuxième variable si tout point de* $I \times \mathbb{R}^n$ *admet un voisinage* Ω *sur lequel* f *est lipschitzienne par rapport à la deuxième variable.*

1.3 Problème de Cauchy

Il arrive qu'on ne recherche pas toutes les solutions d'une EDO mais seulement celles qui vérifient certaines conditions, dites conditions initiales de Cauchy ou tout simplement conditions de Cauchy.
Un problème de Cauchy est un problème constitué d'une équation différentielle dont on recherche une solution vérifiant une certaine condition initiale. Cette condition peut prendre plusieurs formes selon la nature de l'équation différentielle.

Soit U un ouvert de $\mathbb{R} \times \mathbb{R}^n$ et $f : U \longrightarrow \mathbb{R}^n$ une fonction.

Définition 1.3.1. *Étant donnée une équation différentielle du premier ordre sous la forme suivante :*

$$x'(t) = f(t, x),$$

pour $(t, x(t)) \in U$*, et un point* $(t_0, x_0) \in U$ *le problème de cauchy correspondant consiste a chercher des solution* x*, tell que* $x(t_0) = x_0$*. On note le problème de cauchy de la forme suivante :*

$$\begin{cases} x'(t) = f(t, x(t)), \quad (t, x(t)) \in U \\ x(t_0) = x_0 \end{cases} . \tag{1.8}$$

Interprétation physique - Dans de nombreuses situations concrètes, la variable t représente le temps et $x = (x_1, x_2, \cdots, x_n)$ est une famille de paramètres décrivant l'état d'un système matériel donné. L'équation (1.8) traduit physiquement la loi d'évolution du système considéré en fonction du temps et de la valeur des paramètres. Résoudre le problème de Cauchy (1.8) revient à prévoir l'évolution du système au cours du temps, sachant qu'en $t = t_0$ le système est décrit par les paramètre $x_0 = (x_{0,1}, \cdots, x_{0,n})$. On dit que (t_0, x_0) sont les données initiales du problème de Cauchy (1.8).

Définition 1.3.2. *Une solution du problème de Cauchy* (1.8) *sur un intervalle ouvert* I *de* $\mathbb{R}$ *avec la condition initiale* $(t_0; x_0) \in U$ *et* $t_0 \in I$ *est une fonction dérivable* $x : \mathbb{R}_+ \longrightarrow \mathbb{R}$ *telle que :*

- *pour tout* $t \in I$, $(t, x(t)) \in U$,
- *pour tout* $t \in I$, $x'(t) = f(t, x(t))$,
- $x(t_0) = x_0$

Théorème1.3.1. *Soit* $f : I \times V \longrightarrow X$ *une application continue,* I *un ouvert de* $\mathbb{R}$ *et* V *un ouvert connexe non vide d'un* $\mathbb{R}$*-espace de Banach et* (t_0, x_0) *un point fixe de*

$\mathbb{R} \times V$ *et* x *une fonction définie sur un intervalle ouvert qui contient* t_0*, alors* x *est une solution du problème de Cauchy* (1.8) *sur* I *si et seulement si*

1. *pour tout* $t \in I$, $(t, x(t)) \in I \times V$,
2. x *est continue sur* I,
3. *pour tout* $t \in I$

$$x(t) = x_0 + \int_{t_0}^{t} f(s, x(s))\, ds.$$

Preuve 2. *Soit* $x : I \longrightarrow V$ *une fonction continue sur un intervalle ouvert* I *qui contient* t_0 *et que* $(t, x(t)) \in I \times V$.
Supposons que x *est une solution du problème de Cauchy* (1.8). *Alors* x *est dérivable sur* I *et vérifie :*

$$\begin{cases} x'(t) = f(t, x(t)) \\ x(t_0) = x_0 \end{cases} \tag{2.9}$$

En intégrant les deux membres de t_0 *à* t*, on obtient pour tout* $t \in I$

$$\int_{t_0}^{t} x'(t)\, ds = \int_{t_0}^{t} f(s, x(s))\, ds,$$

ce qui donne, en remplaçant $x(t_0)$ *par* x_0

$$x(t) = x_0 + \int_{t_0}^{t} f(s, x(s))\, ds, \quad \textit{pour tout} \quad t \in I.$$

Inversement, supposons que pour tout $t \in I$, x *vérifie :*

$$x(t) = x_0 + \int_{t_0}^{t} f(s, x(s))\, ds, \tag{1.10}$$

alors, d'aprés la continuité de x *et* f*, donc la derivvabilité de la fonction* $t \mapsto f(t, x(t))$*, on obtient*

$$x'(t) = f(t, x(t)), \quad \textit{pour tout} \quad t \in I, \tag{2.11}$$

de plus x *vérifie* $x(t_0) = x_0$ *ce qui signifie que* x *est solution du problème* (1.8).

1.3.1 Théorème de Cauchy Lipschitz

Théorème 1.3.2 (forme faible)**.** *Soit* I *un intervalle ouvert de* $\mathbb{R}^n$*. Si la fonction* $f : I \times \mathbb{R}^n \to \mathbb{R}^n$ *est de classe* $\mathcal{C}^1$ *alors pour toute donnée de Cauchy* $(t_0, x_0) \in I \times \mathbb{R}^n$*, il existe, au voisinage de* t_0*, une unique solution du problème de Cauchy associé.*
En particulier, pour toute telle donnée, il existe une unique solution maximale associée

et toute autre solution vérifiant la condition de Cauchy est une restriction de cette solution maximale.

Théorème 1.3.3. *Soient $I \subset \mathbb{R}$ et $f \in \mathcal{C}(I \times E, E)$. Soit le problèleme de Cauchy*

$$\begin{cases} x'(t) = f(t, x(t)) \\ x(t_0) = x_0. \end{cases} \tag{C}$$

avec $(t_0, x_0) \in I \times E$. Supposons que f est lipschitzienne par rapport à la deuxième variable uniformément sur I. Alors pour tout t_0 dans I et pour tout x_0 dans E le problème de Cauchy (C) admet une solution unique définie sur I.

1.4 l'existence et l'unicité

1.4.1 Solution locale

Soient I un intervalle de $\mathbb{R}$ et D un connexe de E.
Soit $f \in \mathcal{C}(I \times D, E)$ et Soit le problème de Cauchy (1.8). avec $(t_0, x_0) \in I \times D$. Supposons que f est localement lipschitzienne par rapport à la deuxième variable sur $I \times D$. Alors pour tout $(t_0, x_0) \in I \times D$, il existe un intervalle $J \subset I$ contenant t_0 tel que le problème de Cauchy (1.8) admet une unique solution définier sur J.

Définition 1.4.1. *La solution $x(t)$ définie sur J du problème de Cauchy* (1.8) *du théorème précédent s'appelle* ***solution locale*** *et on note $(x(t), J)$.*

Théorème 1.4.1. *Soient I un intervalle de $\mathbb{R}$ et D un connexe de E. Soit $f \in \mathcal{C}(I \times D, E)$. Soit le problème de Cauchy* (1.8)*, avec $(t_0, x_0) \in I \times D$. Supposons que f est localement lipschitzienne par rapport à la deuxiéme variable sur $I \times D$. Alors pour tout $(t_0, x_0) \in I \times D$, le problème de Cauchy* (1.8) *admet une unique solution définie sur un intervalle I_0 inclue dans I, tel que toute solution locale $(x(t), J)$ du problème* (1.8) *vérifie $J \subset I_0$.*

Preuve 3. *Comme f est localement lipschitzienne. Soient $(y_1(t), J_1)$ et $(y_2(t), J_2)$ deux solution locales. Par unicité $y_1(t) = y_2(t)$ pour tout $t \in J_1 \bigcap J_2$. Soit I_0 la réunion de tout les intervalles des solutions. Donc pour tout $t \in I_0$ il existe une solution locale $(y(t), J)$ telle que $t \in J$, posons alors $x(t) = y(t)$.*
Donc $x(t)$ est définie sur I_0 tout entier de plus par unicitè si $(y^*(t), J^*)$ est une solution locale alors $J^* \subset I_0$ et $y^*(t) = x(t)$ sur J^*.

Le théoreme précédent affirme l'existence d'une solution sur l'intervalle ouvert maximal I_0 inclue dans I. Intuitivement l'intervalle I_0 est le plus grand intervalle dans I contenant t_0 où la solution du (1.8) *est définie.*

Définition 1.4.2. *La solution $x(t)$ définie sur I_0 du problème de Cauchy* (1.8) *du théoréme précédent s'appelle* ***solution maximale*** *et I_0 s'appelle* ***intervalle maximale****.*

Remarque1.4.1. *Si $f : I \times E \longrightarrow E$ est localement lipschitzienne par rapport à la deuxième variable et si l'intervalle maximale $I_0 \subset I$ d'une solution $x(t)$ de l'équation est de la forme $]-\infty, b[$, $]a, +\infty[$ ou $]a, b[$ et $a, b \in I$ alors*

$$\lim_{t\to a} |x(t)| = +\infty \quad et \quad \lim_{t\to b} |x(t)| = +\infty.$$

Remarque 1.4.2. *Grâce au théorème des accroissements finis, on vérifie que toute fonction de classe $\mathcal{C}^1$ est localement lipschitzienne par rapport à sa deuxième variable. C'est pourquoi le théorème suivant est bien plus fort que le théorème 1.3.2.*

Théorème 1.4.2 (forme forte)**.** *Soient $I \subset \mathbb{R}$, $f \in \mathcal{C}(I \times \mathbb{R}^n, \mathbb{R}^n)$ et le problèleme de Cauchy*

$$\begin{cases} x'(t) = f(t, x(t)) \\ x(t_0) = x_0. \end{cases} \tag{PC}$$

Si la fonction $f : I \times \mathbb{R}^n \to \mathbb{R}^n$ est de classe $\mathcal{C}^1$ et localement lipschitzienne par rapport à sa deuxième variable, alors pour toute donnée de Cauchy $(t_0, x_0) \in I \times \mathbb{R}^n$, il existe, au voisinage de t_0, une unique solution du problème de Cauchy associé.
En particulier, pour toute telle donnée, il existe une unique solution maximale associée et toute autre solution vérifiant la condition de Cauchy est une restriction de cette solution maximale.

La démonstration de ce théorème peut se faire de plusieurs manières.
Elles partent toutes de la constatation que (J, x) est solution du problème de Cauchy si et seulement si $t_0 \in J$ et si x est une fonction continue sur J qui vérifie

$$x(t) = x_0 + \int_{t_0}^{t} f(s, x(s))\, ds. \tag{1.12}$$

Il s'agit donc de résoudre l'équation intégrale (1.12) dans l'espace fonctionnel $\mathcal{C}^0(J, \mathbb{R}^n)$, c'est pourquoi il est naturel que les preuves du théorème utilisent de façon fondamentale les grands théorèmes de l'analyse fonctionnelle : ou bien le théorème d'Ascoli (méthode de compacité) ou bien le théorème du point fixe de Banach.

Un outil central dans tous les problèmes d'équations différentielles est le lemme de Grönwall qui permet de déduire des bornes sur les solutions à partir d'inégalité intégrales qu'elles vérifient.

Lemme 1.4.1 (Lemme de Grönwall)**.** *Soient $I =]a, b[$, $u : I \to \mathbb{R}_+$ une fonction continue, $t_0 \in I$ ainsi que deux constantes α, $\beta \geq 0$, telles que*

$$0 \leq u(t) \leq \alpha + \beta \left| \int_{t_0}^{t} u(\tau) d\tau \right|, \quad \forall t \in I.$$

Alors

$$0 \leq u(t) \leq \alpha e^{\beta |t - t_0|}, \quad \forall t \in I.$$

Preuve 4. ***Etape 1*** *Supposons que $\alpha > 0$. Posons*

$$\nu(t) = \alpha + \beta \left| \int_{t_0}^{t} u(\tau) d\tau \right|.$$

On a que $\nu \in \mathcal{C}^1$ sauf en $t = t_0$:

$$\nu'(t) = \begin{cases} \beta\, u(t), & t > t_0 \\ -\beta\, u(t), & t < t_0. \end{cases}$$

De plus,

$$0 \leq u(t) \leq \nu(t), \quad \forall t \in I.$$

Et

$$0 \leq \nu(t_0) = \alpha \leq \nu(t), \quad \forall t \in I.$$

Distinguons trois cas :
$t > t_0$ Observer qu'alors,

$$0 \leq \frac{\nu'(t)}{\nu(t)} = \frac{\beta\, u(t)}{\nu(t)} \leq \beta;$$

que l'on intègre :

$$\int_{t_0}^{t} \frac{d}{d\tau} \left[\ln \left(\nu(\tau) \right) \right] \leq \beta (t - t_0);$$

ce qui implique que,

$$O \leq u(t) \leq \nu(t) \leq \alpha e^{\beta |t - t_0|}.$$

$t = t_0$ Le résultat est trivial.
$t < t_0$ Ressemble au premier cas.

Etape 2 *Supposons que $\alpha = 0$. On choisit une suite $(\alpha_n)_n \subset \mathbb{R}_+^\star$ telle que α_n converge*

vers α. Ainsi,

$$\begin{aligned} 0 \leq u(t) &\leq \alpha + \beta \left| \int_{t_0}^{t} u(\tau) d\tau \right| \\ &\leq \alpha_n + \beta \left| \int_{t_0}^{t} u(\tau) d\tau \right|. \end{aligned}$$

On applique ensuite l'étape 1 aux α_n, ce qui nous donne

$$0 \leq u(t) \leq \alpha_n \, e^{\beta |t - t_0|} \longrightarrow 0.$$

Preuve 5 (**Démonstration du théorème 1.4.2**
)**. *Unicité*** *Dans le cas où f est localement Lipschitzienne par rapport à sa seconde variable, on peut démontrer l'unicité d'un éventuelle solution en utilisant le Lemme de Grönwall. En effet, soient (J_1, x_1), (J_2, x_2) deux solutions du même problème de Cauchy en t_0 . On veut montrer que x_1 et x_2 sont égales sur $J_0 = J_1 \bigcap J_2$. Pour cela on introduit l'ensemble*

$$S = \{ t \in J_0, \ \textit{tel que } x_1(s) = x_2(s), \quad \forall s \in [t_0, t] \},$$

où $[t_0, t]$ est remplacé par $[t, t_0]$ si $t < t_0$.
Cet ensemble est non vide car il contient t_0 (x_1 et x_2 vérifient la même donnée de Cauchy à l'instant t_0). On va montrer que $S \bigcap [t_0, +\infty[= J_0 \bigcap [t_0, +\infty[$ (la même idée montrerait l'égalité de $S \bigcap]-\infty, t_0]$ et de $J_0 \bigcap]-\infty, t_0]$).
Supposons que $S \bigcap [t_0, +\infty[\neq J_0 \bigcap [t_0, +\infty[$. On pose alors $t^* = \sup(S)$. On a $t^* \geq t_0$ et $t_0 \in J_0$. En effet, si ce n'était pas le cas on aurait $t^* \in \partial J_0$ et alors $x_1 = x_2$ sur $[t_0, \sup J_0[$ et donc $x_1 = x_2$ sur $J_0 \bigcap [t_0, +\infty[$ par continuité de x_1 et x_2. Ceci contredit l'hypothèse. Par ailleurs, par continuité de x_1 et x_2, on sait que $x_1(t^*) = x_2(t^*) = \tilde{x}$.
Soit L une constante de Lipschitz de f sur le compact $K = [t^*, t^* + 1] \times \bar{B}(\tilde{x}, 1)$. Par continuité, il existe $\delta > 0$ tel que $t^* + \delta \in J_0$ et tel que

$$x_i(t) \in \bar{B}(\tilde{x}, 1), \ \forall t \in [t^*, t^* + \delta], \ \forall i = 1, 2.$$

Par ailleurs, comme x_1 et x_2 vérifient l'équation on a

$$x_i(t) = x_i(t^*) + \int_{t^*}^{t} f(s, x_i(s)) ds.$$

Par soustraction, on trouve

$$|x_1(t) - x_2(t)| \leq \int_{t^*}^{t} |f(s, x_1(s)) - f(s, x_2(s))| \, ds, \quad \forall t \in [t^*, t^* + \delta].$$

Comme x_1 et x_2 prennent leur valeurs dans K, on en déduit

$$|x_1(t) - x_2(t)| \leq L \int_{t^*}^{t} |x_1(s) - x_2(s)|\, ds, \quad \forall t \in [t^*, t^* + \delta].$$

Le lemme de Grönwall donne alors $x_1(t) = x_2(t)$ pour tout $t \in [t^*, t^* + \delta]$. Ceci montre que $t^* + \delta$ est dans S et contredit donc la définition de t^*.

Existence : *Soit $J = [t_0 - \alpha, t_0 + \alpha]$ un intervalle contenant t_0 dans son intérieur. On pose $x^0(t) = x_0$ pour tout $t \in J$ et on construit, par récurrence, la suite de fonctions*

$$x^{n+1}(t) = x_0 + \int_{t_0}^{t} f(s, x^n(s))ds, \quad \forall t \in j.$$

Ceci revient à définir $x^{n+1} = \varphi(x^n)$ où $\varphi : \mathcal{C}^0(J, \mathbb{R}^n) \to \mathcal{C}^0(J, \mathbb{R}^n)$ est l'application qui à x associe

$$\varphi(x)(t) = x_0 + \int_{t_0}^{t} f(s, x(s))ds, \quad \forall t \in J.$$

Résoudre l'équation (1.12) revient à trouver un point fixe de l'application φ. Comme J est compact on peut munir $E = \mathcal{C}^0(J, \mathbb{R}^n)$ de la norme infinie, ce qui en fait un espace complet. On peut donc espérer appliquer le théorème du point fixe de Banach à cette fonction. Pour cela, il faudrait montrer que φ est contractante. Comme on ne possède aucune information globale sur F, il se peut que $\|F(s, x)\|$ soit trés grand quand $\|x\|$ est grand et il y a donc aucune chance que nous arrivions à montrer que φ est contractante sur E.
On va donc essayer d'appliquer le théorème sur le sous-espace fermé $F = \mathcal{C}^0(J, \bar{B}(x_0, r))$ de E (qui est donc bien complet). Pour cela, on peut jouer sur les paramètres α et r pour faire en sorte que $\varphi(F) \subset F$ et que φ soit contractante.

- Fixons une valeur $\alpha_0 > 0$ et un nombre $r_0 > 0$ tels que le compact $K_0 = [t_0 - \alpha_0, t_0 + \alpha_0] \times \bar{B}(x_0, r_0)$ soit inclus dans l'ouvert U sur lequel (1.7) est vraie.

- On note maintenant $M = \sup_{[t_0-\alpha_0, t_0+\alpha_0] \times \bar{B}(x_0, r_0)} \|F\|$. Ainsi, pour toute fonction $x \in \mathcal{C}^0([t_0 - \alpha_0, t_0 + \alpha_0], \bar{B}(x_0, r_0))$ on a

$$\|\varphi(x)(t) - x_0\| \leq \left\| \int_{t_0}^{t} f(s, x(s))ds \right\| \leq |t - t_0|\, M.$$

Si on veut s'assurer que $\varphi(x)(t)$ reste dans la boule $\bar{B}(x_0, r_0))$, il faut se restreindre à un intervalle $[t_0 - \alpha_0, t_0 + \alpha_0]$ avec $0 < \alpha < \alpha_0$ choisi pour que

$$\alpha M \leq r_0. \tag{1.13}$$

Ainsi, l'espace $F_\alpha = \mathcal{C}^0([t_0-\alpha_0, t_0+\alpha_0], \bar{B}(x_0, r_0))$ *est laissé fixe par* φ *dés que* (1.13) *est vérifiée. – Essayons maintenant d'étudier le caractère contractant de* φ *sur un tel espace. Soient* $x, z \in F_\alpha$*, on a*

$$\|\varphi(x)(t) - \varphi(z)(t)\| \leq \left| \int_{t_0}^{t} \|f(s, x(s)) - f(s, z(s))\| \, ds \right| \leq C_{t_0,x_0} \, |t - t_0| \, \|x - z\|_\infty ,$$

et donc

$$\|\varphi(x) - \varphi(z)\| \leq C_{t_0,x_0} \alpha \, \|x - z\|_\infty .$$

En conclusion, φ *sera contractante dés que*

$$\alpha C_{t_0,x_0} < 1. \tag{1.14}$$

- En conclusion, on va choisir $0 < \alpha \leq \alpha_0$ *qui satisfait* (1.13) *et* (1.14)*, ce qui est bien entendu possible. La fonction* φ *laisse alors invariant le sous-espace fermé* $F_\alpha \subset E$ *et elle est contractante dans cet espace.*
D'aprés le théorème du point fixe de Banach, il existe donc une unique solution $x \in F_\alpha$ *à l'équation* (1.12) *et ainsi* $([t_0-\alpha, t_0+\alpha], x)$ *est une solution du problème de Cauchy considéré. C'est également l'unique solution sur cet intervalle qui prend ses valeurs dans la boule* $\bar{B}(x_0, r_0)$*.*

- Il reste à montrer que toute autre solution éventuelle z *du problème de Cauchy définie sur un intervalle de la forme* $[t_0 - \beta, T_0 + \beta]$ *avec* $\beta \leq \alpha$ *coincide avec* x*.*

· Si z *prend ses valeurs dans la boule* $\bar{B}(x_0, r_0)$*, alors la propositionriété d'unicité dans le théorème du point fixe donne le résultat.*

· Si z *ne prend pas ses valeurs dans cette boule, on note* $\tilde{B}$ *le plus grand nombre dans* $[0, \beta]$ *tel que* $z([t_0 - \beta, t_0 + \beta])$ *est contenu dans cette boule. On a* $\tilde{B} < \beta$ *par hypothèse et* $\tilde{B} > 0$ *car* $z(t_0) = x_0$ *est dans l'intérieur de la boule et que* z *est continue. On a alors*

$$\|z(t) - x_0\| \leq |t - t_0| \, M \leq \beta M \leq \alpha M \leq r_0, \ \forall t \in [t_0 - \tilde{B}, t_0 + \tilde{B}],$$

ce qui contredit la maximalité de $\tilde{B}$*.*

Chapitre 02

Méthodes résolution d'équations différentielles

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

$$F\left(t, x(t), x^{(1)}(t), \ldots, x^{(m)}(t)\right) = 0 \qquad \text{où} \qquad x^{(m)}(t) \equiv \frac{d^m x}{dt^m} \tag{1}$$

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

La solution du problème consiste à trouver une fonction $x(t)$ qui soit solution de (1) et dérivable sur un intervalle fini de $t \in [t_0, t_0 + T]$ de $\mathbb{R}$. Souvent dans les applications, la variable t représente le temps, et t_0 *est alors l'instant initial*. En général, il n'existe une solution unique à une équation différentielle qu'une fois certaines conditions limites imposées sur $x(t)$ et ses dérivées. Dans l'exemple de l'équation (1) les *conditions initiales* sont les valeurs de $x(t_0)$, $x^{(1)}(t_0), \ldots, x^{(m-1)}(t_0)$.

2 Problème de Cauchy :

La plupart des méthodes numériques pour résoudre les équations différentielles s'appliquent à des problèmes du type *problème de Cauchy* suivant le nom donné par les mathématiciens. Ce problème se formule de la manière suivante :

Trouver $y(t)$ définie et dérivable sur $[t_0, t_0 + T]$ et à valeurs dans $\mathbb{R}^m$ telle que :

$$\begin{cases} \frac{dy(t)}{dt} = f(t, y(t)) & \forall t \in [t_0, t_0 + T] \\ \\ y(t_0) = y_0 \end{cases} \tag{2}$$

où $f(t, y(t))$ est une fonction de $\mathbb{R}^{m+1}$ dans $\mathbb{R}^m$ et $y_0 \in \mathbb{R}^m$. Concrètement l'expression, "trouver $y(t)$ à valeurs dans $\mathbb{R}^m$ avec $y_0 \in \mathbb{R}^m$" consiste à dire pour des applications comme Matlab, que l'inconnue $y(t)$ est un vecteur de m fonctions inconnues avec pour condition limite le vecteur y_0 :

$$y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_m(t) \end{bmatrix} \qquad y_0 = y(t_0) = \begin{bmatrix} y_1(t_0) \\ y_2(t_0) \\ \vdots \\ y_m(t_0) \end{bmatrix} = \begin{bmatrix} y_{0,1} \\ y_{0,2} \\ \vdots \\ y_{0,m} \end{bmatrix} \tag{3}$$

De même, $f(t, y(t))$ est une fonction de t et du vecteur $y(t)$ et doit retourner un vecteur colonne :

$$\frac{dy(t)}{dt} \equiv \frac{d}{dt}\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = f(t, y(t)) \equiv \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \tag{4}$$

Pour la plupart des problèmes qui intéressent les scientifiques et les ingénieurs, des théorèmes mathématiques assurent l'existence et l'unicité d'une solution au problème de Cauchy. Néanmoins, souvent la solution ne peut être exprimée *analytiquement*. Pour de tels problèmes, on doit donc chercher à déterminer la fonction $y(t)$ par des méthodes *numériques*.

2.1 Transformations vers un problème de Cauchy

Dans Matlab (Octave), de puissant programmes (fonctions) existent sous le nom générique de ODEs (Ordinary Differential Equation Solvers). Ils résolvent les systèmes de la forme de l'équation (2). Le travail principal d'un utilisateur de Matlab consiste donc le plus souvent à transformer son problème sous la forme de l'équation (2). Dans bien des domaines, surtout ceux des équations à dérivées partielles, les transformations d'un problème donné sous la forme d'un problème de Cauchy sont toujours d'actualité comme problèmes de recherche.

2.2 Traitement d'une équation différentielle d'ordre > 1

Nous ne regarderons que la transformation d'une équation différentielle d'ordresupérieur à 1, en problème de Cauchy. Considérons donc une équation différentielle d'ordre $m$de la forme suivante :

$$x^{(m)}(t) \equiv \frac{dx^{(m-1)}}{dt} = \varphi\left(t, x(t), x^{(1)}(t), \ldots, x^{(m-1)}(t)\right) \qquad \forall t \in [t_0, t_0 + T] \tag{5}$$

Posons de nouvelles fonctions $y_i(t)$ avec $i \in [1, 2, \ldots, m]$ définies telles que :

$$y_1(t) \equiv x(t), \quad y_2(t) \equiv x^{(1)}(t) \quad , \ldots, \quad y_m(t) \equiv x^{(m-1)}(t) \tag{6}$$

Grâce à ces définitions, l'équation (5) d'ordre m s'écrit comme un système de m équations d'ordre 1 :

$$\begin{cases} \frac{dy_1(t)}{dt} = y^{(2)}(t) \\ \vdots \\ \frac{dy_{m-1}(t)}{dt} = y^{(m)}(t) \\ \frac{dy_m(t)}{dt} = \varphi(t, y_1(t), y_2(t), \ldots, y_m(t)) \end{cases} \tag{7}$$

Ce système a donc la forme d'un problème de Cauchy en posant :

$$y(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_{m-1}(t) \\ y_m(t) \end{bmatrix} \qquad \text{et} \qquad f(t, y(t)) = \begin{bmatrix} y_2(t) \\ \vdots \\ y_m(t) \\ \varphi(t, y_1, \ldots, y_m) \end{bmatrix} \tag{8}$$

L'équation (5) s'écrira alors :

$$\frac{dy(t)}{dt} = f(t, y(t)) \qquad \forall t \in [t_0, t_0 + T] \tag{9}$$

Pour obtenir alors un problème de Cauchy, il faut spécifier les conditions initiales $(y_1(t_0), y_2(t_0), \ldots, y_m(t_0))$ ce qui revient à dire d'après l'équation (6), qu'il faut connaître $x(t)$ et ses dérivées jusqu'à l'ordre $m-1$ au 'temps' initial t_0 : $\left(x(t_0), x^{(1)}(t_0), \ldots, x^{(m-1)}(t_0)\right)$. On remarque qu'une équation différentielle d'ordre m d'une seule fonction inconnue, $x(t)$, se traduit par un problème de Cauchy avec m fonctions inconnues, $y_i(t)$, et m conditions initiales.

2.3 Equations différentielles à coefficients constants

En particulier, les équations différentielles à coefficients constants constituent une classe d'équations de la forme de l'éq.(5). Notamment quand φ est de la forme :

$$\varphi\left(t, x\left(t\right), x^{(1)}\left(t\right), \ldots, x^{(m-1)}\left(t\right)\right) = s\left(t\right) - a_1 x\left(t\right) - a_2 x^{(1)}\left(t\right) - \ldots - a_m x^{(m-1)}\left(t\right) \tag{10}$$

l'équation l'éq.(5) peut s'écrire comme une équation différentielle à coefficients constants :

$$a_1 x\left(t\right) + a_2 x^{(1)}\left(t\right) + \ldots + a_m x^{(m-1)}\left(t\right) + x^{(m)}\left(t\right) = s\left(t\right) \tag{11}$$

où la fonction $s\left(t\right)$ est communément appelée un terme source.

Pour des équations de la forme de l'éq.(11), les substitutions de l'éq.(6) amènent à un système d'équations de forme matricielle. Par exemple, une équation à coefficients constants d'ordre 4 s'écrit :

$$a_1 x\left(t\right) + a_2 x^{(1)}\left(t\right) + a_3 x^{(2)}\left(t\right) + a_4 x^{(3)}\left(t\right) + x^{(4)}\left(t\right) = s\left(t\right) \tag{12}$$

Après les substitutions de l'équation (6), cette équation s'écrit :

$$a_1 y_1\left(t\right) + a_2 y_2\left(t\right) + a_3 y_3\left(t\right) + a_4 y_4\left(t\right) + \frac{d}{dt} y_4\left(t\right) = s\left(t\right) \tag{13}$$

et l'équation (9) peut s'écrire sous une forme matricielle :

$$\frac{d}{dt}\begin{bmatrix} y_1\left(t\right) \\ y_2\left(t\right) \\ y_3\left(t\right) \\ y_4\left(t\right) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -a_1 & -a_2 & -a_3 & -a_4 \end{bmatrix} \begin{bmatrix} y_1\left(t\right) \\ y_2\left(t\right) \\ y_3\left(t\right) \\ y_4\left(t\right) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ s\left(t\right) \end{bmatrix} \tag{14}$$

Même s'il est intéressant de voir ce type de problème comme une équation matricielle, nous ne devons pas oublier que la formulation de l'équation (9) nous permet de traiter bien des problèmes qui ne prennent pas la forme d'une équation matricielle. On remarque aussi qu'il y a beaucoup de zéros dans l'équation (14) et donc une multiplication de matrice n'est pas la façon la plus éfficace de programmer $f\left(t, y\left(t\right)\right)$ (Voir la fonction (A) de la section 2.3 ci-dessous).

2.4 Exemple - Vol d'un point solide dans un champ de pesanteur.

Imaginons qu'on cherche à résoudre numériquement le problème du mouvement d'un point solide de masse m à la position $\overrightarrow{\boldsymbol{x}}\left(t\right) = x\widehat{\boldsymbol{x}} + y\widehat{\boldsymbol{y}} + z\widehat{\boldsymbol{z}}$ ayant une vitesse $\overrightarrow{\boldsymbol{v}} = \frac{d\overrightarrow{\boldsymbol{x}}}{dt}$ dans un champ de pesanteur $\overrightarrow{\boldsymbol{g}}$. (figure 1)

La mécanique du point nous dit qu'il suffit d'appliquer la relation fondamentale de la dynamique au point solide :

$$m\frac{d\overrightarrow{\boldsymbol{v}}}{dt} = \overrightarrow{\boldsymbol{P}} = m\overrightarrow{\boldsymbol{g}} \tag{15}$$

Puisqu'il s'agit d'une équation vectorielle, nous avons en principe trois équations scalaires à résoudre, mais nous savons que le vol du point s'effectue dans un plan parallèle au plan défini par (xOz). On arrive donc à un système de deux équations différentielles de deuxième ordre à résoudre :

$$\begin{cases} \frac{d^2x}{dt^2} = 0 \\ \frac{d^2z}{dt^2} = -g \end{cases} \tag{16}$$

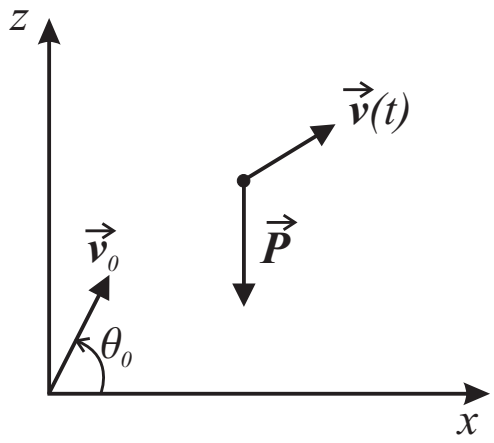

FIG. 1 – Mouvement d'un point de masse dans un champ de pesanteur

Avec les conditions limites

$$\begin{aligned} x\left(t_0\right) &= x_0 \qquad x^{(1)}\left(t_0\right) = v_{0,x} \\ z\left(t_0\right) &= z_0 \qquad z^{(1)}\left(t_0\right) = v_{0,z} \end{aligned} \tag{17}$$

nous connaissons la solution exacte de chacune de ces deux équations :

$$\begin{aligned} x\left(t\right) &= x_0 + v_{0,x}t \\ z\left(t\right) &= z_0 + v_{0,z}t - \frac{1}{2}gt^2 \end{aligned} \tag{18}$$

Nous voulons simplement tester notre capacité à trouver la solution de façon numérique. La connaissance d'une solution exacte nous permet de tester différentes méthodes de résolution numérique d'équations différentielles.

Pour résoudre les équations différentielles d'ordre 2 de l'éq.(16) on va définir des fonctions du système $u\left(t\right)$ (pour ne pas confondre avec la position $y\left(t\right)$) et invoquer les substitutions de l'éq.(6) :

$$\begin{aligned} u_1\left(t\right) &\equiv x\left(t\right) \\ u_2\left(t\right) &\equiv x^{(1)}\left(t\right) = v_x\left(t\right) \end{aligned} \tag{19}$$

L'équation $\frac{d^2x}{dt^2} = 0$ devient donc le système matriciel :

$$\frac{d}{dt}\begin{bmatrix} u_1\left(t\right) \\ u_2\left(t\right) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} u_1\left(t\right) \\ u_2\left(t\right) \end{bmatrix} \tag{20}$$

De même on peut définir

$$\begin{aligned} u_3\left(t\right) &\equiv z\left(t\right) \\ u_4\left(t\right) &\equiv z^{(1)}\left(t\right) = v_z\left(t\right) \end{aligned} \tag{21}$$

et le système $\frac{d^2z}{dt^2} = -g$ devient :

$$\frac{d}{dt}\begin{bmatrix} u_3\left(t\right) \\ u_4\left(t\right) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} u_3\left(t\right) \\ u_4\left(t\right) \end{bmatrix} + \begin{bmatrix} 0 \\ -g \end{bmatrix} \tag{22}$$

On peut regrouper ces deux équations sous la forme d'une seule grande équation matricielle :

$$\frac{du}{dt} \equiv \frac{d}{dt}\begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g \end{bmatrix} \tag{23}$$

ou de manière équivalente :

$$\frac{du}{dt} = f(t, u(t)) = \begin{bmatrix} u_2(t) \\ 0 \\ u_4(t) \\ -g \end{bmatrix} \tag{24}$$

La solution de cette équation va donc nous fournir les fonctions : $x(t) = u_1(t)$, $v_x(t) = u_2(t)$, $z(t) = u_3(t)$, et $v_z(t) = u_4(t)$. On peut facilement programmer une fonction en Octave/Matlab pour calculer $f(t, u(t))$:

```
function f = fprojectile(u,t)        % en Matlab - 'function f = fprojectile(t,u)'
% fonction f (t, u (t)) pour une particule dans un champ de pesanteur
  nc = length(u) ;
  f = zeros(nc,1) ;
  f(1) = u(2) ;
  f(2) = 0 ;
  f(3) = u(4) ;
  f(4) = -9.8 ;                  % valeur de g
end
```

(A)

Important : Il est très important que la fonction retourne un *vecteur colonne*. Les fonctions Matlab (Octave) pour résoudre une équation différentielle ne marchent pas si la fonction retourne un vecteur ligne. L'écriture de la fonction 'fprojectile' permet à l'argument u d'être un vecteur ligne ou un vecteur colonne. Il s'avère assez commode de prendre u comme un vecteur ligne.

Maintenant on peut utiliser la fonction 'lsode' d'Octave ('ODE45' de MatLab) pour résoudre numériquement l'équation différentielle. Cette fonction prend en argument le nom de la fonction $f(t, u(t))$, un vecteur contenant les valeurs de $t = [t_0, t_1, \ldots, t_N]$ pour lesquelles on veut connaître les valeurs de $u(t)$, $[u(t_0), u(t_1), \ldots, u(t_N)]$. Pour certaines applications, on ne s'intéressera qu'à une seule et unique valeur $u(t_N)$; dans ce cas, on donne simplement un vecteur $t = [t_0, t_N]$. Dans les cas où il faut connaître une trajectoire, il faut que $t = [t_0, t_1, \ldots, t_N]$ contienne suffisamment d'éléments pour que les courbes générées par 'plot' paraissent lisses.

Le script suivant résout l'équation différentielle de l'équation (16) avec les conditions initiales $x_0 = z_0 = 0$, $v_0 \equiv \left\|\vec{\boldsymbol{v}}_0\right\| = 100\text{ms}^{-1}$, $\theta_0 \equiv (\widehat{\boldsymbol{x}}, \vec{\boldsymbol{v}}_0) = 30°$ et montre la position de la particule pour $N + 1 = 31$ temps compris entre 0 et 11s.

```
tmin = 0 ;          % temps initial t_0
tmax = 11 ;         % temps finale t_0 + T
v0 =100 ;           % vitesse intiale en mètres par seconde
thetdeg =30 ;       % angle initial en degrés
x0 = 0 ;            % position x_0 initiale
z0 = 0 ;            % position z_0 initiale
vx0 = v0*cos(thetdeg*pi/180) ;        % vitesse v_{x,0} initiale
vz0 = v0*sin(thetdeg*pi/180) ;        % vitesse v_{z,0} initiale
Nint = 30 ;         % nombre d'intervalles : N
```

```
h = (tmax-tmin)/Nint ;                  % taille du pas  h = t_{n+1} - t_n
t = linspace(tmin,tmax,Nint+1) ;        % vecteur des t = [t_0, t_1, ... t_{N-1}, t_N = t_0 + T]
u0 = [x0 vx0 z0 vz0] ;                  % conditions initiales u_0 = [x_0 v_{0,x} z_0 v_{0,z}]
usol =lsode("fprojectile",u0,t) ;       % résolution numérique par Octave
% [t,usol] = ODE45('fprojectile',t,u0) ;    % résolution numérique par Matlab
xpos = usol( :,1) ;                     % position des x  : première colonne de usol
zpos = usol( :,3) ;                     % position des z : troisième colonne de usol
plot(xpos,zpos,'o')                     % plot de z en fonction de x.
```

(B)

Il faut remarquer que la matrice 'usol' est une matrice avec $N+1$ lignes pour les $N+1$ valeurs de t_n et 4 colonnes correspondant respectivement à la position et à la vitesse suivant x et à la position et à la vitesse suivant z : $u(t_n) = [x(t_n), v_x(t_n), z(t_n), v_z(t_n)]$. On illustre sur la figure (2) une comparaison entre la solution générée par 'ODE45' de Matlab ('lsode' d'Octave) et la solution exacte. L'erreur absolue, $\sqrt{(u_1(t_N) - x(t_N))^2 - (u_3(t_N) - z(t_N))^2}$, sur la position du point solide après 11 secondes de vol est de l'ordre de 10^{-13}m sous Matlab.

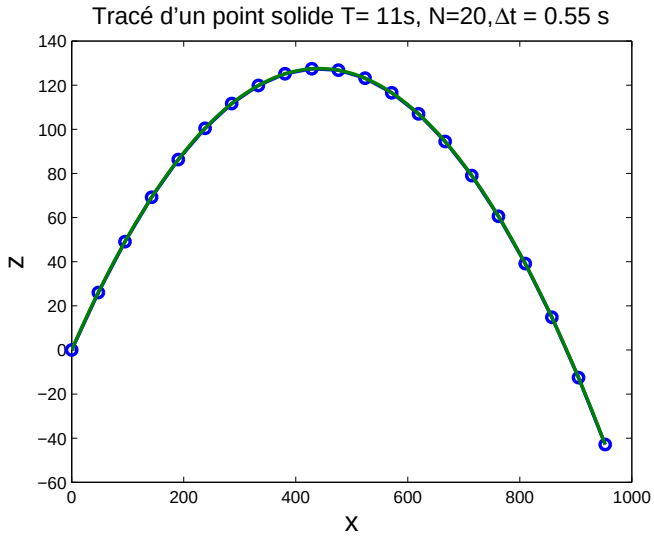

FIG. 2 – Mouvement d'un point de masse m, de vitesse $\boldsymbol{v}$ dans le champ de pesanteur $\boldsymbol{g}$. Solution de ODE45 avec $x_0 = z_0 = 0$, $v_0 = 100\text{ms}^{-1}$, et $\theta_0 = 30°$

2.5 Détermination des paramètres initiaux

Un des buts souvent recherchés lors des calculs différentiels est de déterminer où "d'optimiser" les paramètres initiaux afin d'obtenir un certain comportement désiré de la solution. Par exemple, dans le cas d'un projectile on pourrait s'intéresser au problème suivant : pour quel angle de lancement le projectile ira-t-il le plus loin ? Dans ce cas, on pourrait donc écrire une fonction simple qui trouve de façon approximative, le point d'atterrissage du point solide. Une telle fonction peut comporter d'abord une boucle qui permet de trouver les deux instants entre lesquels le projectile a dû toucher le sol. Plus précisément, la boucle chercherait l'instant t_n où $z(t_n) > 0$ et $z(t_{n+1} = t_n + h) < 0$. On peut ensuite faire une approximation linéaire entre ces deux points, i.e. on cherche à déterminer les coefficients a et b d'une fonction linéaire $z(t) = at+b$

tels que :

$$z_n \equiv z(t_n) = at_n + b \tag{25a}$$
$$z_{n+1} \equiv z(t_{n+1}) = at_{n+1} + b \tag{25b}$$

En soustrayant l'éq.(25a) de l'éq.(25b), on obtient :

$$z_{n+1} - z_n = a\,(t_{n+1} - t_n) \tag{26}$$

d'où on tire une expression pour a :

$$a = \frac{z_{n+1} - z_n}{(t_{n+1} - t_n)} \tag{27}$$

On remet cette expression pour a dans l'éq.(25a) et on trouve :

$$z_n = \frac{(z_{n+1} - z_n)\,t_n + b\,(t_{n+1} - t_n)}{(t_{n+1} - t_n)} \tag{28}$$

ce qui donne après simplification :

$$b = \frac{z_n t_{n+1} - t_n z_{n+1}}{t_{n+1} - t_n} \tag{29}$$

Le point d'atterrissage, t_a, est donc le zéro de la fonction linéaire $z\,(t_\mathrm{a}) = at_\mathrm{a} + b = 0$, i.e. t_a est donné par :

$$at_\mathrm{a} = -b$$
$$t_\mathrm{a} = \frac{z_n t_{n+1} - t_n z_{n+1}}{(z_n - z_{n+1})} \tag{30}$$

En général, la vitesse sur x n'est pas constante, et on peut faire une approximation linaire sur $x\,(t)$:

$$x_n \equiv x(t_n) = ct_n + d \tag{31}$$
$$x_{n+1} \equiv x(t_{n+1}) = ct_{n+1} + d \tag{32}$$

Une approximation à la position horizontale (des x) au point d'atterrissage est maintenant :

$$x\,(t_\mathrm{a}) = \frac{x_{n+1} - x_n}{(t_{n+1} - t_n)} t_\mathrm{a} + \frac{x_n t_{n+1} - t_n x_{n+1}}{t_{n+1} - t_n} \tag{33}$$

Et le point d'atterrissage est :

$$(t_\mathrm{a}, x\,(t_\mathrm{a})\,, z = 0) \tag{34}$$

Le fonction 'point_atterrissage' suivante écrite sous Matlab (Ocatve) permet d'effectuer ce calcul du point d'atterissage. Les arguments de cette fonction sont les trois vecteurs : $t = [t_0, t_1, \ldots, t_N]$, $x = [x(t_0), x(t_1), \ldots, x(t_N)]$, et $z = [z(t_0), z(t_1), \ldots, z(t_N)]$. Elle ne fonctionne de manière satisfaisante que si l'intervalle $\Delta t \equiv t_{i+1} - t_i$ est suffisamment petit.

```
function [ta,xa] = point_atterrissage(t,x,z)
% trouver le point d'atterissage d'un point solide
nc = length(t) ;
n = 1 ;      % boucle pour trouver l'interval [t_n,t_n+1] où z(t_n+1)< 0
```

```
while( z(n+1) > 0 && n<nc )
      n = n+1 ;
end  % while
 if (n ~= nc)
     ta=( z(n)*t(n+1)-t(n)*z(n+1) ) / ( z(n)-z(n+1)) ;
     Dt = t(n+1) - t(n) ;
     xa = ta*(x(n+1) - x(n))/Dt + (t(n+1)*x(n)-t(n)*x(n+1)) /Dt ;
  else
     disp('Pas de point atterissage dans les données')
      ta = -1 ;
      xa = -1 ;
  end  %end if
end
```

(C)

On voit en figure (3) les tracées des trajectoires et le point d'atterrissage de trois tirs avec différents angles de lancement avec $v_0 \equiv \left\|\overrightarrow{v}_0\right\| = 100\text{ms}^{-1}$. Bien entendu, en l'occurrence un calcul analytique simple nous permettrait de trouver l'angle 'optimal' de lancement. Néanmoins, des modélisations plus réalistes (ex. forces de frottement non-linéaires) peuvent facilement mener à des situations où un calcul analytique devient impossible.

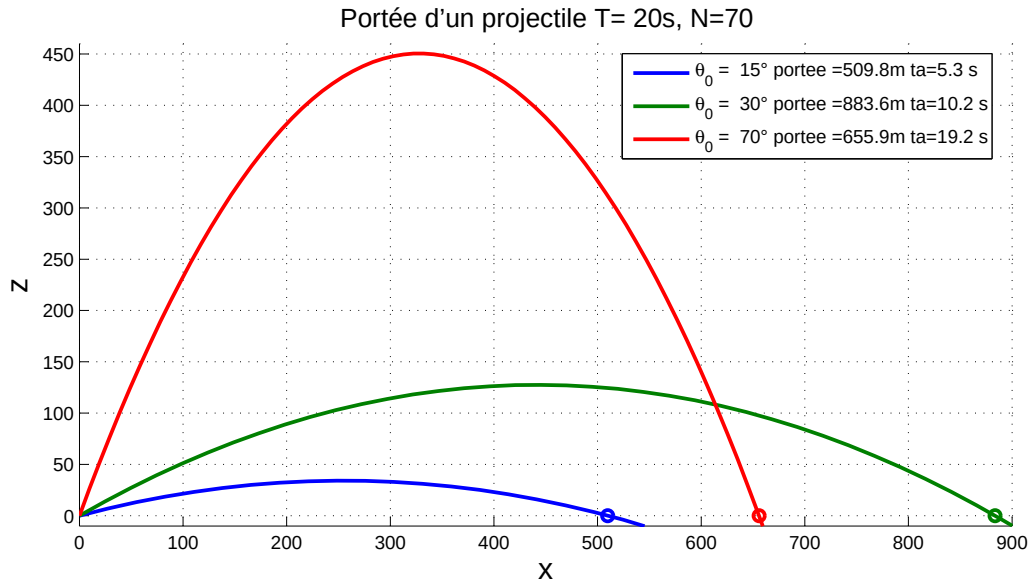

FIG. 3 – Tracées des trajectoires et le point d'atterrissage de trois tirs avec différents angles de lancement.

Nous avons vu dans cette section l'utilité d'avoir une résolution fiable et rapide d'équations différentielles. Dans la prochaine section nous étudierons des méthodes de résolution numérique d'équations différentielles.

Chapitre 03

Solutions numériques des équations différentielles

Il existe un grand nombre de façons de résoudre une équation différentielle et aucune méthode n'est clairement supérieure à toutes les autres dans toutes les circonstances. De nos jours, la technique Runge Kutta d'ordre 4 et à pas adaptatif est souvent utilisée. Nous verrons les raisonsde ce choix dans ce qui suit. Néanmoins, les recherches dans ce domaine ne sont pas encoreterminées et le développement des techniques numériques continue, surtout pour les situationsoù une très grande précision est demandée.

Dans tout ce qui suit, il faut garder à l'esprit que dans le cas le plus général, y est un vecteurcolonne, et $f(t, y(t))$ est un "vecteur" dont les "éléments" sont des fonctions. Néanmoins, afinde simplifier la discussion, nous ne considérons dans les exemples que des équations simples àune seule fonction inconnue. Par la suite, dans la section 3.5, nous verrons les généralisationsqui permettent de traiter multiples fonctions inconnues $y_i(t)$.

3.1 Formulation générale

On commence en remarquant qu'une solution formelle de l'équation

$$\frac{d}{dt}\, y(t) = f(t, y(t)) \tag{35}$$

est obtenue en intégrant de t_0 à t :

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

$$y(t) - y(t_0) = \int_{t_0}^{t} f(s, y(s))\, ds \tag{36}$$

donc la solution $y(t)$ s'écrit

$$y(t) = y_0 + \int_{t_0}^{t} f(s, y(s))\, ds \tag{37}$$

où $y_0 \equiv y(t_0)$. Le problème avec cette solution formelle est que l'inconnue $y(t)$ se trouve sous l'intégrale.

3.2 Méthode itérative de Picard

Dans la méthode de Picard, on peut itérer l'équation (37) afin de générer une série d'approximations à $y(t)$, dénotées ${}_1y(s)$, ${}_2y(s)$, ${}_ky(s)$,... . On initialise l'itération en prenant $y(t) \simeq y_0$ sous l'intégrale :

$$\begin{aligned}
{}_1y(t) &= y_0 + \int_{t_0}^{t} f(s,\ y_0)\, ds \\
{}_2y(t) &= y_0 + \int_{t_0}^{t} f(s,\ {}_1y(s))\, ds \\
&\vdots \\
{}_ky(t) &= y_0 + \int_{t_0}^{t} f(s,\ {}_{k-1}y(s))\, ds
\end{aligned} \tag{38}$$

La méthode de Picard est surtout utilisée dans les études mathématiques des équations différentielles. Néanmoins, cette méthode peut s'avérer intéressante quand il est possible de faire une intégration analytique de la suite $\int_{t_0}^{t} f(s,{}_k\, y(s))\, ds$. Les logiciels de calcul symbolique comme Mathématica et Maple sont capable de calculer ce type d'intégrale.

3.2.1 Exemple : méthode de Picard pour résoudre l'équation $\frac{d}{dt}y(t) = t - y(t)$

Dans ce problème, il n'y a qu'une seule fonction inconnue $y(t)$. Ce problème est suffisamment simple que l'on puisse trouver une solution analytique. Prenons la condition limite $(0, 1)$ i.e. $y(t_0 = 0) = 1$; la solution analytique de l'équation

$$\frac{d}{dt}y(t) = t - y(t) \tag{39}$$

est obtenue par les techniques habituelles d'équations différentielles à coefficients constants. Vous pouvez vérifier que la solution de (39) est :

$$y(t) = (y_0 + 1)\, e^{-t} + t - 1 \tag{40}$$

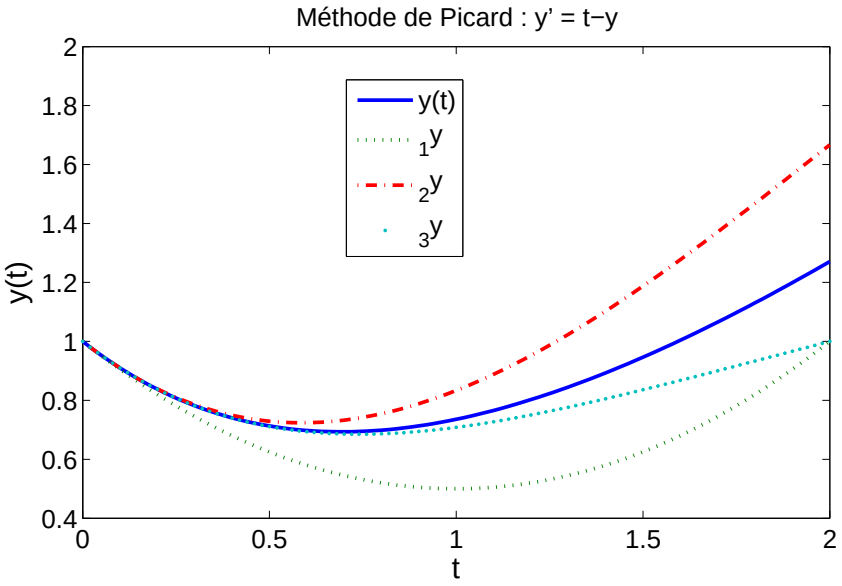

FIG. 4 – Méthode de Picard. Comparaison des solutions de $\frac{d}{dt}y(t) = t - y(t)$ avec les approximations de Picard ${}_1y$, ${}_2y$, ${}_3y$.

Avec la condition limite $(0, 1)$ i.e. $y(t_0 = 0) = 1$, la méthode de Picard donne la suite :

$$\begin{aligned}
{}_1y(t) &= 1 + \int_0^t (s-1)\, ds = 1 - t + \frac{t^2}{2} \\
{}_2y(t) &= 1 + \int_0^t \left[s - \left(1 - s + \frac{s^2}{2} \right) \right] ds = 1 - t + t^2 - \frac{t^3}{6} \\
{}_3y(t) &= 1 + \int_0^t \left[s - \left(1 - t + t^2 - \frac{t^3}{6} \right) \right] ds = 1 - t + t^2 - \frac{t^3}{3} + \frac{t^4}{24} \\
&\vdots
\end{aligned} \tag{41}$$

En dépit de son efficacité, la méthode de Picard n'est intéressante que pour un nombre limité de problèmes où nous sommes capables d'effectuer analytiquement les intégrations. On préfère

en général des techniques qui sont de nature plus numérique mais de domaine d'application plus vaste.

Sur la figure (4), on représente une comparaison des solutions de la suite de Picard de l'équation (41) à la solution exacte. Ce graphique est généré par le code scripte, (D), ci-dessous. La série de solutions de Picard illustre une propriété souvent rencontrée par des solutions numériques aux équations différentielles, notamment que plus l'intervalle T sur lequel on veut trouver la solution est grande, plus la solution numérique risque de s'éloigner de la solution exacte quand $t \to T$. Néanmoins on doit garder à l'esprit que l'ampleur de ce phénomène dépend de la nature de l'équation différentielle et de la méthode employée pour résoudre l'équation.

La Figure (4) est générée par le code suivant :

```
function Picard(tmax,Nint)
% Compare les 3 premiers termes dans la suite de Picard ...
% avec la solution exacte de y' = t - y conditions limites (0,1)
% ici t0 = 0 , et y0 = 1
  t = linspace(0,tmax,Nint+1);                                        (D)
  yanal = 2*exp(-t) + t - 1;                % solution analytique
  y1 = 1 - t + t.^2/2;                      % Approximation de Picard : 1y
  y2 = 1 - t + t.^2 - t.^3/6;               % Approximation de Picard : 2y
  y3 = 1 - t + t.^2 - t.^3/3 + t.^4/24;     % Approximation de Picard : 3y
  axes('FontSize',14);
  plot(t,yanal,t,y1,' :',t,y2,'-.',t,y3,'.','LineWidth',2);
  xlabel(' t ','FontSize',16);
  ylabel(' y(t) ','FontSize',16);
  legend('y(t)','_1y','_2y','_3y')
end
```

On doit remarquer que nous avons simplifié notre traitement de la suite de Picard par un choix judicieux des conditions initiales. Les choses se compliquent un peu si l'on veut utiliser la suite de Picard pour y_0 et t_0 arbitraires :

$$
\begin{aligned}
{}_1y(t) &= y_0 + \int_{t_0}^{t} (s-1)\,ds = y_0 + y_0\,(t_0 - t) + \frac{1}{2}\left(t^2 - t_0^2\right) \\
{}_2y(t) &= y_0 + \int_{t_0}^{t} \left\{ s - \left[y_0 + y_0\,(t_0 - s) + \frac{1}{2}\left(s^2 - t_0^2\right) \right] \right\} ds \\
&= y_0 + \frac{1}{6}\left\{ t^2\,(3-t) + 3\,\left(t^2 - 2\,t + 2\right)\,y_0 + 6\,(1-t)\,t_0\,y_0 + 3\,t_0^2\,(t + y_0 - 1) - 2\,t_0^3 \right\} \\
&\vdots
\end{aligned}
\tag{42}
$$

3.3 Méthodes basées sur la série de Taylor

Les méthodes basées sur la série de Taylor se prêtent bien à des traitements numériques. Commençons par subdiviser l'intervalle $[t_0, t_0 + T]$ en N sous intervalles de même longueur h et appelons t_n les points de subdivision. Nous avons donc :

$$
h = \frac{T}{N} \qquad \text{et} \qquad t_n = t_0 + nh \qquad \text{pour} \qquad n = 0, \ldots, N \tag{43}
$$

Par conséquent en MatLab, t est un vecteur avec $N + 1$ éléments :

$$
t = [t_0, \ldots, t_N] \tag{44}
$$

Le développement de la série de Taylor de $y(t_{n+1})$ jusqu'à l'ordre m autour du point t_n s'écrit :

$$y(t_{n+1}) = y(t_n) + hy^{(1)}(t_n) + \frac{h^2}{2!}y^{(2)}(t_n) + \ldots + \frac{h^m}{m!}y^{(m)}(t_n) + O\left(h^{m+1}\right) \tag{45}$$

3.3.1 Méthode d'Euler

Si on arrête la série de Taylor à l'ordre 1 on obtient :

$$y(t_{n+1}) \simeq y(t_n) + hy^{(1)}(t_n) \tag{46}$$

L'application de cette formule pour calculer les $y(t_n)$ est connue comme *méthode d'Euler*. On remarque que l'erreur commise à chaque étape est de l'ordre de $O\left(h^2\right)$. Il est clair qu'on peut augmenter la précision de cette méthode en diminuant la taille de h, mais un h petit implique qu'on doit évaluer la fonction un grand nombre de fois ce qui est coûteux en temps d'exécution. Qui plus est, un grand nombre d'évaluations de la fonction peut provoquer un cumul des erreurs numériques commises par la machine.

En Matlab, on peut facilement programmer la méthode d'Euler avec la fonction suivante :

```
function [t,y] = Euler(f,tmin,tmax,Nint,y0)   % Méthode d'Euler
 % Nint - nombre de sous intervalles N
 % tmin - temps t0
 % tmax - temps t0 + T
 % f est une fonction avec comme arguments t et y(t) : f(t,y(t))
 % y0 à l'entrée est composé des valeurs des conditions limites y0 = y(t0)
  h = (tmax-tmin)/Nint;                  % valeur du pas
  t = linspace(tmin,tmax,Nint+1);        % vecteur de t discrétisé t=[tmin,tmax]
  y(1) = y0;                             % initialisation : y(1)=y(t0) = y0
  for n = 2 : Nint+1
     y(n) =y(n-1) + h*feval(f,t(n-1),y(n-1));        % Calcul d'Euler
  end    % for n
 end     % fonction Euler
```

(E)

La fonction 'Euler' retourne le vecteur $t = [t_0, \ldots, t_N]$ et un vecteur y contenant les $y(t_n)$ à chacun des temps t_n de l'intervalle $t \in [t_0, t_0 + T]$.

A noter : nous avons utilisé la fonction MatLab (Octave) 'feval'. La fonction 'feval("fcnt",x)' est une commande de Octave (MATLAB) qui évalue la fonction de nom 'fcnt' et d'argument x . **Exemple :** y=feval("sin",x) est équivalente à y=sin(x). L'utilisation de 'feval' nous permet d'écrire une fois pour toutes la méthode d'Euler et de passer le nom de la fonction à évaluer comme un argument de la fonction 'Euler'.

Exemple numérique :

Utilisons la même équation teste que dans l'exemple de la méthode de Picard : $\frac{dy}{dt} = t - y(t)$. Essayons d'abord avec $N = 20$ intervalles. En MatLab (Octave) entrez :

```
>>  f = inline('t - y','t','y');
>>  [t,yEul20] = Euler( f, 0, 2, 20, 1);
```

où la fonction 'inline' est une façon rapide de créer une fonction simple comme $f = t - y$ sans avoir à créer un fichier 'f.m' de cette fonction.

On peut générer les valeurs exactes de la solution de cette équation par les commandes suivantes :

```
>> yt = inline('(y0+1)*exp(-t) + t - 1','t','y0');
```

```
>> yexact = yt(t,y0) ;
```

Finalement, on peut comparer graphiquement la solution générée par la méthode d'Euler avec la solution exacte grâce aux commandes :

```
>> plot(t,yexact,t,yEul20,' :') ;
>> legend('y(t)',' Euler ') ;
```

Le vecteur t est donc ici $t = [t_0 = 0, t_1, \ldots, t_N = 2]$ et 'yEul20' est la solution par la méthode d'Euler avec 20 intervalles.

Bien entendu, on peut améliorer la solution d'Euler en diminuant h (i.e. en augmentant le nombre d'intervalles). On voit sur la figure (5) une comparaison des solutions de $y' = t - y$ par la méthode d'Euler pour $N = 20$ et $N = 50$ pour $t \in [0, 2]$.

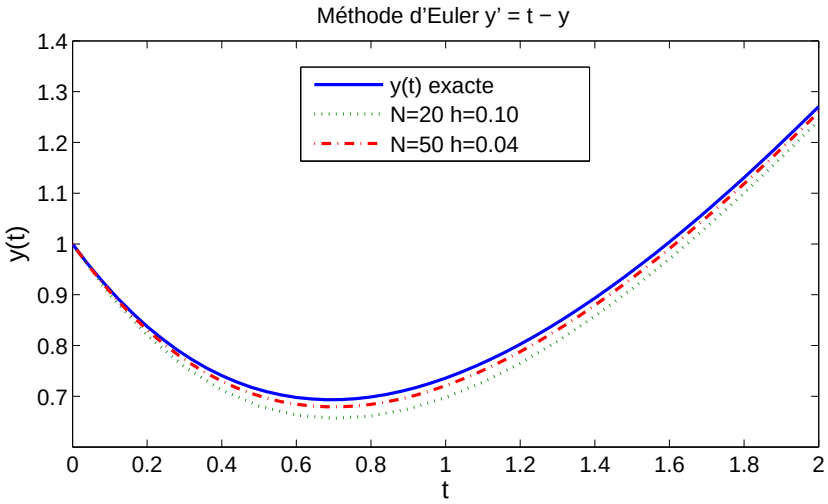

FIG. 5 – Méthode d'Euler. Comparaison des solutions de $\frac{d}{dt}y(t) = t - y(t)$ avec les approximations par méthode d'Euler.

Vous avez peut-être remarqué qu'ici la solution par Euler s'approche de la solution exacte pour les grands t. Ce phénomène est une propriété de l'équation différentielle particulièrement simple que nous étudions ici, et non une propriété générale des solutions par méthode d'Euler.

3.3.2 Méthodes de Taylor d'ordre plus élevés

En considérant l'équation (45), et en augmentant le nombre de termes inclus dan la série de Taylor on s'attend à une amélioration de la solution numérique. Pour ceci, nous avons besoin des dérivées d'ordre supérieur à 1. On peut les obtenir en dérivant l'équation différentielle de

base :

$$\begin{aligned}
\frac{dy}{dt} &\equiv y^{(1)} = f\left(t, y\left(t\right)\right) \\
\frac{d^2y}{dt^2} &\equiv y^{(2)} = \frac{d}{dt} f\left(t, y\left(t\right)\right) \\
y^{(3)} &= \frac{d^2}{dt^2} f\left(t, y\left(t\right)\right) \\
&\vdots \\
y^{(m)} &= \frac{d^{m-1}}{dt^{m-1}} f\left(t, y\left(t\right)\right)
\end{aligned} \tag{47}$$

Le développement de la série de Taylor (45), nous montre que l'erreur commise par une méthode de Taylor d'ordre m est $O\left(h^{m+1}\right)$. Néanmoins, il est incorrect de penser que plus m est grand mieux c'est, puisque les calculs à chaque étape deviennent plus lourds au fur et à mesure que m augmente (voir (F) ci dessous). Il est communément admis qu'il n'est pas profitable de prendre $m > 4$.

Pour notre équation $y^{(1)} = t - y$ traitée dans les exemples précédents, ce procédé pour déterminer les $y^{(m)}$ avec $m > 1$ est facile :

$$\begin{aligned}
y^{(1)} &= t - y \\
y^{(2)} &= 1 - \frac{d}{dt} y\left(t\right) = 1 - t + y\left(t\right) \\
y^{(3)} &= \frac{d}{dt}\left(1 - t + y\left(t\right)\right) = -1 + t - y \\
y^{(4)} &= \frac{d}{dt}\left(-1 + t - y\left(t\right)\right) = 1 - t + y \\
&\vdots
\end{aligned} \tag{48}$$

La fonction Matlab suivante calcule les approximations à solution de $y^{(1)} = t - y$ faites avec la série de Taylor jusqu'à l'ordre 4.

```
function Taylor(tmin,tmax,Nint,y0)          % Approximations par série de Taylor
% Nint - nombre de sous intervalles N
% tmin - temps t0   ;    tmax - temps t0 + T
% On trait ici l'équation y' = t - y(t)
 h = (tmax-tmin)/Nint ;                      % valeur du pas
 t = linspace(tmin,tmax,Nint+1) ;            % vecteur de t discrétisé t=[tmin,tmax]
 yt = inline('(y0+1)*exp(-t) + t - 1','t','y0') ; % solution exact                    (F)
 texa = linspace(tmin,tmax,101) ;     % discrétisation de t pour l'affichage de la solution exacte
  yexact = yt(texa,y0) ;
 y1 = inline('t - y','t','y') ;                % y^(1)
 y2 = inline('1 - t + y','t','y') ;            % y^(2)
 y3 = inline('-1 + t - y','t','y') ;           % y^(3)
 y4 = inline('1 - t + y','t','y') ;            % y^(4)
 yT1(1) = y0 ;          % yT1 contient les solutions de y - Approximation d'Euler
 yT2(1) = y0 ;          % yT2 contient les solutions de y Approximation de Taylor ordre 2
 yT3(1) = y0 ;          % yT3 contient les solutions de y Approximation de Taylor ordre 3
```

```
 yT4(1) = y0;              % yT4 contient les solutions de y Approximation de Taylor ordre 4
  for n = 2 :Nint+1        % Calcul approximation d'Euler
       yT1(n) = yT1(n-1) + h*y1(t(n-1),yT1(n-1));
  end % for n
  for n = 2 :Nint+1          % Calcul de l'approximation de Taylor ordre 2
       yT2(n) = yT2(n-1) + h*y1(t(n-1),yT2(n-1)) + (h^2/2)*y2(t(n-1),yT2(n-1));
  end % for n
 for n = 2 :Nint+1           % Calcul de l'approximation de Taylor ordre 3
      yT3(n) = yT3(n-1) + h*y1(t(n-1),yT3(n-1)) + ...
      +(h^2/2)*y2(t(n-1),yT3(n-1))+(h^ 3/6)*y3(t(n-1),yT3(n-1));
   end % for n
for n = 2 :Nint+1           % Calcul de l'approximation de Taylor ordre 4
  yT4(n) = yT4(n-1) + h*y1(t(n-1),yT4(n-1)) + ...
   +(h^2/2)*y2(t(n-1),yT4(n-1))+(h^ 3/6)*y3(t(n-1),yT4(n-1)) + ...
   + (h^4/24)*y4(t(n-1),yT4(n-1));
end % for n
% plot : créer un graphique avec les résultats des 4 courbes
axes('FontSize',14);
plot(texa,yexact,t,yT1,t,yT2,t,yT3,t,yT4,'+','LineWidth',2)
xlabel(' t ','FontSize',16);
ylabel(' y(t) ','FontSize',16);
legend('y(t) exacte','Euler (ordre 1)','Taylor ordre 2','Taylor ordre 3','Taylor ordre 4');
tit = sprintf('Méthodes de Taylor y"=t-y : N=%d, h=%.2f',Nint,h);
title(tit);
end
```

On peut maintenant trouver la solution de $y^{(1)} = t - y$ avec $t \in [0, 2]$ et $N = 20$ avec la commande :

```
>> Taylor(0,2,20,1);
```

Le résultat apparaît en figure (6). On voit qu'à partir de l'ordre 2 de Taylor, on ne voit plus de différence entre les approximations basées sur la méthode de Taylor et la solution exacte à l'échelle de la figure, alors que la différence entre l'approximation d'Euler et la solution exacte est considérable. On peut essayer de faire apparaître les différences entre les calculs à différents ordres de la série de Taylor en augmentant le pas de $h = 0.1$ à $h = 0.4$ avec la commande :

```
>> Taylor(0,2,5,1);
```

Le résultat de cette commande apparaît sur la figure (7). On constate que même avec ce pas assez grand, les 5 valeurs calculées par la méthode de Taylor à l'ordre 4 reste sur la courbe de la solution exacte.

3.4 Runge Kutta

Ayant constaté les améliorations offertes par le développement de Taylor, on doit remarquer un problème important. Pour se servir de la méthode de Taylor, il faut calculer les fonctions $y^{(m)}(t)$ d'ordre $m > 1$ à la main en prenant les dérivées de l'équation de base $y' = f(t, y(t))$. Par conséquent, avec les méthodes de Taylor on doit recommencer cette étape chaque fois qu'on change la fonction f. Cette opération est lourde et loin d'être pratique.

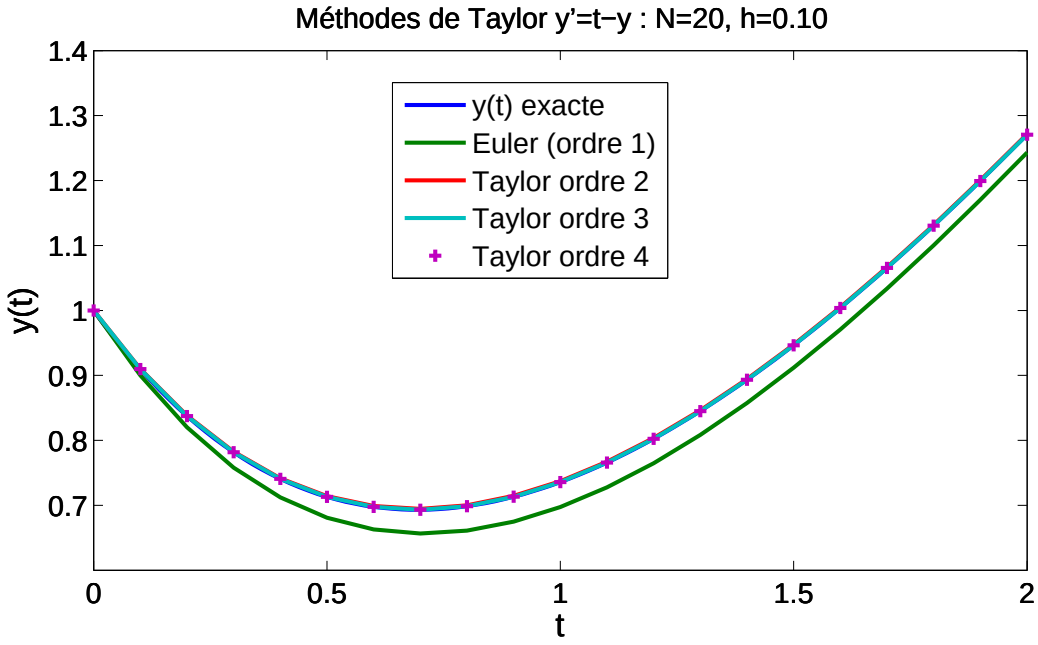

FIG. 6 – Méthodes basées sur la série de Taylor avec $h = 0.1$.

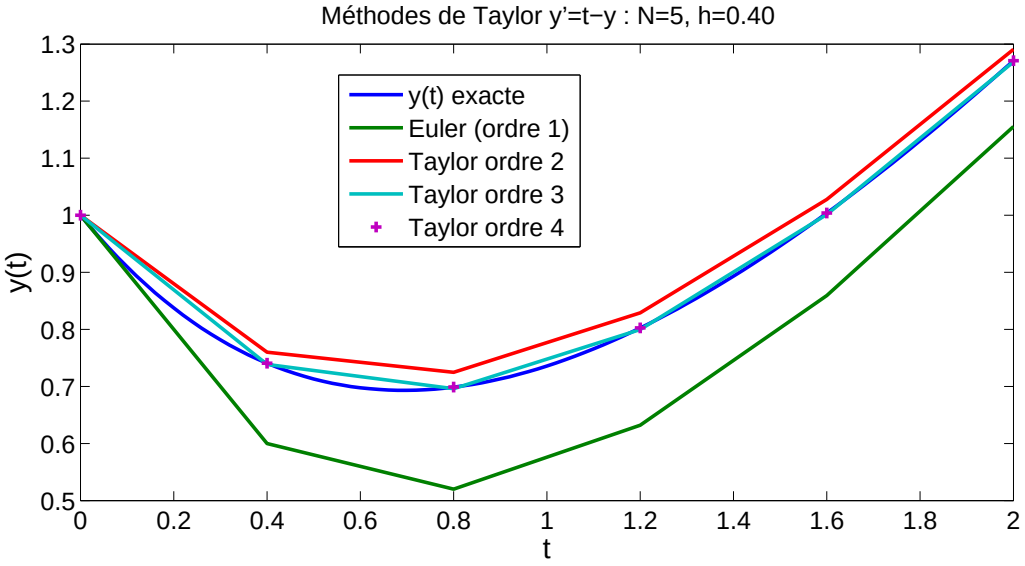

FIG. 7 – Méthodes basées sur la série de Taylor avec $h = 0.4$.

La méthode Runge Kutta tire les avantages des méthodes de Taylor tout en gardant une simplicité d'exécution de la méthode d'Euler. En pratique, Runge Kutta remplace l'évaluation analytique des ordres $y^{(m)}$, $m > 1$ par des dérivées numériques obtenues en évaluant la fonction $f(t, y(t))$ à différents endroits afin d'obtenir presque les mêmes résultats que ceux obtenu avec la méthode de Taylor.

3.4.1 Runge Kutta d'ordre 2

Rappelons la méthode d'Euler

$$y(t_{n+1}) = y(t_n) + hf(t_n, y(t_n)) \tag{49}$$

La méthode d'Euler permet ainsi de calculer $y(t_n + h)$ en fonction de $y(t_n)$ et la dérivée en $y(t_n)$. Cette méthode n'est pas symétrique par rapport à l'intervalle puisqu'il ne fait pas intervenir l'information sur la dérivée en fin d'intervalle, i.e. $f(t_n, y(t_{n+1}))$ n'intervient pas.

La méthode Runge Kutta est plus symétrique puisqu'elle revient à évaluer numériquement la dérivée au centre de l'intervalle. Les équations de cette méthode sont :

$$\begin{aligned} k_1 &\equiv hf(t_n, y(t_n)) \\ k_2 &\equiv hf\left(t_n + \frac{1}{2}h, y(t_n) + \frac{1}{2}k_1\right) \\ y(t_{n+1}) &= y(t_n) + k_2 + O\left(h^3\right) \end{aligned} \tag{50}$$

Une façon de programmer la méthode Runge Kutta d'ordre 2 est la suivante :

```
function [t,y] = RK2(f,tmin,tmax,Nint,y0)   % Méthode de Runge Kutta d'ordre 2
% Nint - nombre de sous intervalles
% tmin - temps t0
% tmax - temps t0 + T
% f est une fonction avec comme arguments t et y : f(t,y(t))
% y0 contient les valeurs des conditions limites
 h = (tmax-tmin)/Nint ;                % valeur du pas
 t = linspace(tmin,tmax,Nint+1) ;      % vecteur de t discrétisé t=[tmin,tmax]
 y(1) = y0 ;                           % y contient les solutions de y(t_n)n = 1, ..., Nint + 1
 for n = 2 :Nint+1
    k1 = h*feval(f,t(n-1),y(n-1)) ;
    k2 = h*feval(f,t(n-1)+h/2,y(n-1)+k1/2) ;
    y(n) = y(n-1) + k2 ;
 end % for n
end
```

(G)

Comme la méthode d'Euler, les méthodes de Runge Kutta peuvent être appliquées à une fonction arbitraire. La quasi-équivalence de la méthode Runge Kutta d'ordre 2 avec la méthode de Taylor d'ordre 2 n'est pas évidente sans quelques manipulations mathématiques (voir ci-dessous). Néanmoins, on peut rapidement vérifier que leur comportement au niveau numérique est similaire. Pour notre exemple habituel, $y' = t - y$, nous comparons en figure (8) les résultats des deux méthodes en utilisant un grand pas, $h = 0.4$.

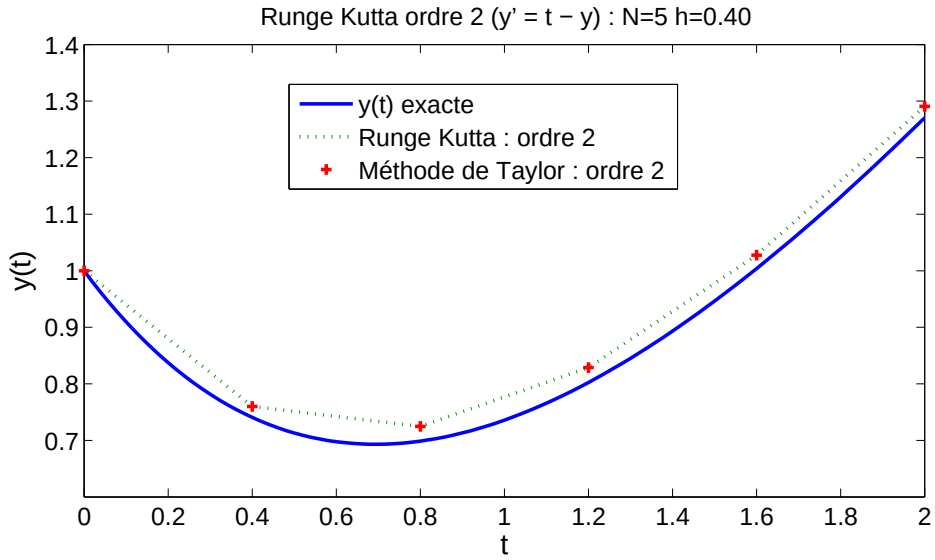

FIG. 8 – Comparaison de la méthode de Runge Kutta d'ordre 2 avec la série de Taylor d'ordre 2 avec $h = 0.4$.

Dérivation de Runge Kutta d'ordre 2 Avec un peu d'effort, on peut dériver les formules de Runge Kutta à partir des formules de Taylor et les formules des dérivées par différences finies. Prenons un développement en série de Taylor autour du point $(\widetilde{t},\widetilde{y})$ au centre de l'intervalle t_n, t_{n+1}.

$$\widetilde{t} \equiv t_n + \frac{h}{2} \qquad \widetilde{y} \equiv y\left(t_n + \frac{h}{2}\right) \tag{51}$$

La série de Taylor développée autour de ce point s'écrit en remplaçant $t_n \to t_n + \frac{h}{2}$, et $h \to \frac{h}{2}$ dans le développement de Taylor de l'éq.(45) :

$$\begin{aligned} y\left(t_n + h\right) &= y\left(t_n + \frac{h}{2}\right) + \frac{h}{2}y^{(1)}\left(t_n + \frac{h}{2}\right) + \frac{h^2}{8}y^{(2)}\left(t_n + \frac{h}{2}\right) + O(h^3) \\ &= \widetilde{y} + \frac{h}{2}f\left(t_n + \frac{h}{2}, \widetilde{y}\right) + \frac{h^2}{8}y^{(2)}\left(t_n + \frac{h}{2}\right) + O(h^3) \end{aligned} \tag{52}$$

Maintenant, on remplace $y^{(2)}\left(t_n + \frac{h}{2}\right)$ par sa dérivée par différences discrètes :

$$\begin{aligned} y^{(2)}\left(t_n + \frac{h}{2}\right) &= \frac{4}{h^2}\left[y\left(t_n + h\right) - 2y\left(t_n + \frac{h}{2}\right) + y\left(t_n\right)\right] + O\left(h\right) \\ &= \frac{4}{h^2}\left[y\left(t_n + h\right) - 2\widetilde{y} + y\left(t_n\right)\right] + O\left(h\right) \end{aligned} \tag{53}$$

Insérant ce résultat dans l'équation (52), on obtient :

$$\begin{aligned} y\left(t_n + h\right) &= \widetilde{y} + \frac{h}{2}f\left(t_n + \frac{h}{2}, \widetilde{y}\right) + \frac{1}{2}\left[y\left(t_n + h\right) - 2\widetilde{y} + y\left(t_n\right)\right] + O(h^3) \\ &= \frac{h}{2}f\left(t_n + \frac{h}{2}, \widetilde{y}\right) + \frac{1}{2}y\left(t_n + h\right) + \frac{1}{2}y\left(t_n\right) + O(h^3) \end{aligned} \tag{54}$$

ce qui peut s'écrire :

$$y\left(t_n+h\right)=y\left(t_n\right)+hf\left(t_n+\frac{h}{2},\widetilde{y}\right)+O(h^3) \tag{55}$$

mais la série de Taylor d'ordre un nous donne le résultat suivant :

$$\begin{aligned}\widetilde{y}\equiv y\left(t_n+\frac{h}{2}\right)&=y\left(t_n\right)+\frac{h}{2}f\left(t_n,y\left(t_n\right)\right)+O(h^2)\\&\equiv y\left(t_n\right)+\frac{h}{2}k_1+O(h^2)\end{aligned} \tag{56}$$

où nous avons utilisé la définition de k_1 (voir l'éq.(50)). Avec ce résultat, on peut faire l'approximation :

$$\begin{aligned}hf\left(t_n+\frac{h}{2},\widetilde{y}\right)&=hf\left(t_n+\frac{h}{2},y\left(t_n\right)+\frac{h}{2}k_1\right)+O(h^3)\\&\equiv hk_2+O(h^3)\end{aligned} \tag{57}$$

et l'équation (55) devient finalement la formule de Runge Kutta d'ordre 2 :

$$y\left(t_n+h\right)=y\left(t_n\right)+hk_2+O(h^3). \tag{58}$$

3.4.2 Runge Kutta : ordres 3 et 4

Les dérivations de Runge Kutta aux ordres 3 et 4 sont fastidieuses. Heureusement leurs formules sont faciles à programmer et nous nous contentons d'utiliser les résultats.

La formule Runge-Kutta à l'ordre 3 est :

$$\begin{aligned}k_1&=hf\left(t_n,y\left(t_n\right)\right)\\k_2&=hf\left(t_n+\frac{1}{2}h\ ,\ y\left(t_n\right)+\frac{1}{2}k_1\right)\\k_3&=hf\left(t_n+h\ ,\ y\left(t_n\right)+2k_2-k_1\right)\\y\left(t_{n+1}\right)&=y\left(t_n\right)+\frac{1}{6}\left(k_1+4k_2+k_3\right)+O\left(h^4\right)\end{aligned} \tag{59}$$

La formule Runge-Kutta à l'ordre 4 est de loin la plus utilisée. Elle a une forme assez symétrique :

$$\begin{aligned}k_1&=hf\left(t_n,y\left(t_n\right)\right)\\k_2&=hf\left(t_n+\frac{1}{2}h\ ,\ y\left(t_n\right)+\frac{1}{2}k_1\right)\\k_3&=hf\left(t_n+\frac{1}{2}h\ ,\ y\left(t_n\right)+\frac{1}{2}k_2\right)\\k_4&=hf\left(t_n+h\ ,\ y\left(t_n\right)+k_3\right)\\y\left(t_{n+1}\right)&=y\left(t_n\right)+\frac{1}{6}\left(k_1+2k_2+2k_3+k_4\right)+O\left(h^5\right)\end{aligned} \tag{60}$$

La technique Runge-Kutta à l'ordre 4 intervient dans la plupart des programmes ODE (Ordinary Differential Equations) comme ceux utilisés par Matlab et Octave.

On montre maintenant en figure (9) qu'en pratique Runge-Kutta à l'ordre 4 donne presque les mêmes résultats que la technique de Taylor d'ordre 4.

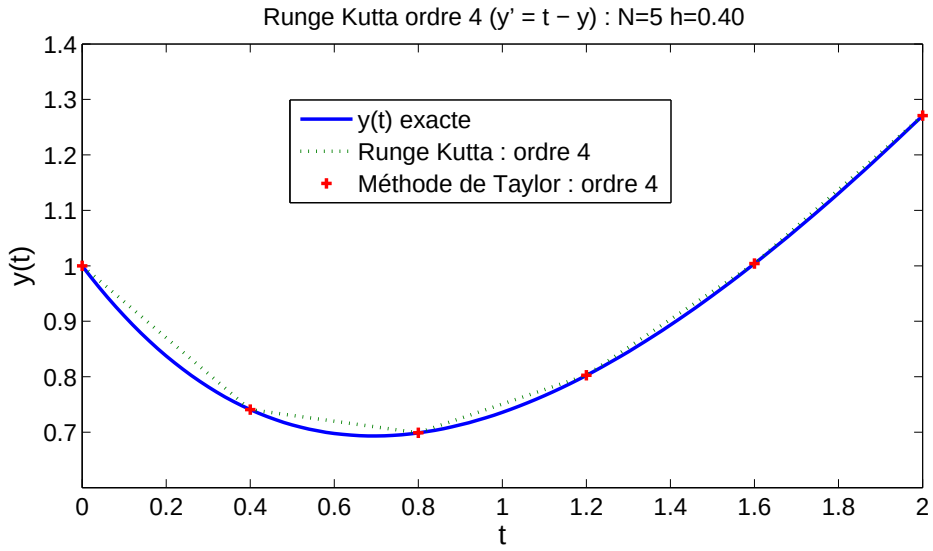

FIG. 9 – Comparaison de la méthode Runge Kutta d'ordre 4 avec la série de Taylor ordre 4 avec $h = 0.4$.

On remarque que Runge-Kutta d'ordre 4 fait deux fois plus d'évaluations de la fonction f que Runge-Kutta d'ordre 2. Par conséquent RK4 n'est vraiment meilleur que RK2 que si RK4 donne de meilleurs résultats que RK2 avec un pas deux fois plus grand. Ceci est souvent le cas, mais pas toujours.

3.4.3 Runge Kutta à pas adaptatif et méthodes prédiction correction

Même si l'erreur produite par la méthode Runge Kutta est de l'ordre de $O\left(h^5\right)$, il reste la question du choix de h afin d'avoir le meilleur compromis entre précision et temps de calcul. Il est intuitivement clair que l'on peut choisir un h relativement grand dans une région où $y\left(t\right)$ varie plutôt lentement, mais un petit h est nécessaire quand il y a de fortes variations de $y\left(t\right)$. Matlab utilise une méthode Runge Kutta d'ordre 4 à pas adaptatif dans ODE45 pour résoudre les équations différentielles. Dans les situations où une grande précision est nécessaire, des méthode dites prédiction correction sont souvent plus efficace que même Runge Kutta d'ordre 4. Néanmoins, les méthodes prédiction correction ont besoin d'une première solution approximative pour démarrer leurs algorithmes. Souvent donc, Runge Kutta est invoqué par les algorithmes prédiction correction pour fournir une première solution approximative.

3.5 Fonctions Euler et Runge Kutta adaptée à $y \in \mathbb{R}^m$

On peut facilement écrire les fonctions d'Euler et de Runge-Kutta afin qu'elles marchent pour un nombre de fonctions inconnues, $y_i\left(t\right)$, arbitraires. Pour faciliter les comparaisons, on écrira ces fonctions afin qu'elles retournent des valeurs dans le même format que 'ODE45' (de MatLab), ou ('lsode' d'Octave) c'est-à-dire dans une matrice composée de $N+1$ lignes et une colonne pour chaque fonction inconnue $y_i\left(t\right)$, $i \in [1, m]$.

```
function [t,y] = Euler(f,tmin,tmax,Nint,y0)   % Méthode d'Euler
% Nint - nombre de sous intervalles
% tmin - temps t_0
```

```
% tmax - temps t_0 + T
% f est une fonction avec comme arguments t et un vecteur y : f(t, y(t))
% y0 contient les valeurs des conditions initiales
 h = (tmax-tmin)/Nint;                                                    (H)
 t =linspace(tmin,tmax,Nint+1);          % vecteur de t discrétisé t=[tmin,tmax]
 t = t';
 nc = length(y0);
 y = zeros(Nint+1,nc);     % initialisation d'une matrice de solution
 y(1, :) = y0;       % la ligne y(1, :) contient les conditions initiales y_0
 for n = 2 :Nint+1
        y(n, :) = y(n-1, :) + h*feval(f,t(n-1),y(n-1, :))'; % Calcul d'Euler
 end    % for n
end
```

On peut facilement modifier les fonctions Runge Kutta de la même manière afin d'obtenir des fonctions qui acceptent y_0 comme un vecteur et donnent les solutions pour $y_i\,(t)$, $i \in [1, m]$.

→

Conclusion

Dans ce mémoire, nous avons étudié les équations différentielles ordinaires (EDO) sous deux aspects : **théorique** et **numérique**.

Sur le plan **théorique**, nous avons abordé les fondements mathématiques des EDO :

- Les définitions formelles des équations différentielles et des problèmes de Cauchy.
- Les conditions d'existence et d'unicité des solutions, notamment à travers les **théorèmes de Cauchy-Lipschitz** et de **Picard**.
- Le rôle des fonctions lipschitziennes dans la stabilité des solutions.

Sur le plan **numérique**, nous avons exploré différentes méthodes d'approximation des solutions :

- **Méthode de Picard** : utile pour les démonstrations théoriques et la convergence.
- **Méthode d'Euler** : simple à implémenter, mais peu précise.
- **Méthodes de Taylor d'ordre supérieur** : permettent une meilleure précision mais exigent des dérivées successives.
- **Méthodes de Runge-Kutta** (ordres 2, 3 et 4) : elles offrent un bon compromis entre précision et complexité, et sont largement utilisées dans les logiciels modernes comme MATLAB.

Ainsi, nous avons démontré que lorsque les solutions exactes ne sont pas accessibles, les méthodes numériques deviennent essentielles pour obtenir des approximations utiles dans la pratique.

BIBLIOGRAPHIE

[1] J.F.Alzaidy, The (G'/G)-Expansion Method for Finding Traveling Wave Solutions of Some Nonlinear Pdes in Mathematical Physics, Mathematics Departement, Faculty of science, Taif University, Kingdom of Saudi Arabia.

[2] A. bekit, Application of the (G'/G) expansion method for nonlinear evolution equations, Dumlupinar Univercity, Art-Science Faculty,Department of Mathematics, Kutahya, Turkey,Reeived 5 December 200 ; received in revised from 22 January 2008 ; accepted 30 January 2008.

[3] A. kelleche, Équation de la physique mathématique, Université Djilali Bounaama, October 10,2020.

[4] A. lesfari, Équations différentielles ordinaires et équations aux dérivées partielles, Ellipses Édition Marketing S.A., Avril 2015 97-286.

[5] A. majid wazwaz, Partial Differential Equations and solitary waves theory, Department of Mathematics, Saint Xavier University Chicago, IL 60655, USA 2009.

[6] S. nicaise, Analyse numérique et équations aux dérivées partielles Cours et problèmes résolus, Achevé d'imprimer sur les presses de la SNEL S.A. Mars 2000 10-145.

[7] R. Hedli, Quelques méthodes de résolution des équations aux dérivées partielles non linéaire, Université Ferhat Abbas - Sétif 1, Thèse Pour l'obtention du diplôme de doctorat en sciences, $19/09/2020$.