RÉPUBLIQUE ALGEÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministre de l'Enseignement Supérieur et de la Recherche Scientifique

Centre universitaire Salhi Ahmed -Naâma
Institut des Sciences et technologies
Département de Mathématiques et Informatique

Mémoire de fin d'étude

Pour obtenir le diplôme de
Master en Mathématiques

Filière : Mathématiques

Spécialité : Analyse fonctionnelle et EDPs

Thème

SOLUTION NUMÉRIQUE DES ÉQUATIONS DIFFÉRENTIELLES ORDINAIRES

Presenté par :
NOUAL Soufiane

Devant le jury composé de :

Dr BELABED Zakaria	maître de conférences A	C-Univ Naâma	**Président**
Dr BELGUERNA Abderrahmane	maître de conférences A	C-Univ Naâma	**Encadreur**
Mme KHALAOUTI Hafida	maître assistant A	C-Univ Naâma	**Examinatrice**

Année universitaire 2018/2019

Dédicace

A mes chers parents :

Sources de mes joies, secrets de ma force
Vous serez toujours le modèle
Papa, dans ta détermination, ta force et ton honnêteté
Maman dans ta bonté, ta patience et ton dévouement pour nous.
Merci pour tous vos sacrifices pour que vos enfants
Grandissent et prospèrent
Merci de trimer sans relâche, malgré les péripéties de la vie
Au bien être de vos enfants
Merci d'être tout simplement mes parents
C'est à vous que je dois cette réussite
Et je suis fière de vous l'offrir.

A mes frères et ma sœur : Abdou, Yacine, Sanaa et Islam

En témoignage de l'attachement, de l'amour et de
L'affection que je porte pour vous.
Je vous dédie ce travail avec tous mes vœux de bonheur, santé et de réussite.

A mes chers ami(e)s : Hamza, Youcef, Imad, Ahmed, mes collègues de l'Ecole EPSECG TLEMCEN : Smail, Hmida, Hakim, Nouri, Abdou et Aissa.

Je ne peux trouver les mots justes et sincères pour vous
Exprimer mon affection et mes pensées, vous êtes pour moi des
Frères, sœurs et des amis sur qui je peux compter.
En témoignage de l'amitié qui nous uni et des souvenirs de
Tous les moments que nous avons passé ensemble, je vous dédie
Ce travail et je vous souhaite une vie pleine de santé et de
Bonheur.

A tous ceux qui me sont chers...

Remerciement

On remercie dieu le tout puissant de nous avoir donné la santé et la volonté d'entamer et de terminer ce mémoire.

Avant tout, nous tenons à présenter nos très sincères remerciements à notre encadrant M.BUELGUERNA pour son soutien, son encouragement, sa bienveillance et les critiques pertinentes qui nous ont été précieuses tout au long de ce travail.

Je tiens à remercier les membres du jury pour leur présence, pour leur lecture attentive de ma thèse ainsi que pour les remarques qu'ils m'adresseront lors de cette soutenance afin d'améliorer mon travail.

Mes remerciements particuliers à notre professeur , le Dr. MEKKI Slimane, qui était un mon meilleur conseilleur, sans son immense aide je ne pouvais pas terminer.

Un grand merci à vous, mes collègues, je vous souhaite à toutes et à tous beaucoup de bonheur et de réussite dans tous vos projets personnels et professionnels !
Bonne continuation à tous et meilleurs vœux de bonheur à chacun !

Résumé

Les manuels sur les équations différentielles ordinaires donnent souvent l'impression que la plupart des équations différentielles ordinaires (EDO) peuvent être résolues facilement, mais ça n'est pas toujours possible. Il reste vrai que les solutions de la grande majorité des EDO ne peuvent être résolues par des moyens analytiques. Par conséquent, il est important de pouvoir aborder le problème d'une autre manière. Il existe aujourd'hui de nombreuses méthodes produisant des approximations numériques pour résoudre des équations différentielles. Nous présentons brièvement des méthodes les plus populaires telles la méthode créée par Euler vers 1768, et de Runge-Kutta en 1901. Le but de ce travail est de montrer l'importance de l'outil informatique en résolvant des EDO. Matlab (2018) est utilisé pour intègrer presque tous ces problèmes car il s'agit d'un environnement de résolution des problèmes très pratique et largement utilisé avec des solveurs de qualité exceptionnellement faciles à utiliser. C'est également un langage de programmation de si haut niveau, ces programmes sont courts, ce qui permet de les répertorier des complets pour tous les exemples.

Mots-clés : Méthode d'Euler, méthode de Runge-Kutta, Matlab, solveurs de Matlab, pendule simple.

Abstract

Textbooks on differential equations often give the impression that most ordinary differential equations (ODE) can be solved easily, but experience does not bear this out. It remains true that solutions of the vast majority of first order initial value problems cannot be found by analytical means. Therefore, it is important to be able to approach the problem in other ways. Today there are numerous methods that produce numerical approximations to solutions of differential equations. We present briefly some of the most popular methods such as the method created by Euler around 1768, and Runge-Kutta in 1901. The purpose of this work is to show the importance of computer tools in solving ODE. Matlab (2018) is used to solve nearly all these problems because it is a very convenient and widely used problem-solving environment with quality solvers that are exceptionally easy to use. It is also such a high-level programming language that programs are short, making it practical to list complete programs for all the examples.

Keywords : Euler method, Runge-kutta method, Matlab, Matlab's solvers, simple pendulum.

Table des Matières

Liste des figures

Introduction générale

Le terme œquatio differentialis ou équation différentielle est apparu pour la première fois sous la plume de Leibniz en 1676 pour définir la relation entre les différentielles dx et dy des deux variables x et y. Dans la littérature anglo-saxonne, on appelle équation différentielle ordinaire une relation entre une variable indépendante (qui sera notée ici t) et une variable dépendante (qui sera notée ici y) ainsi que ses dérivées dans la variable indépendante. Ceci par opposition avec la notion d'équation aux dérivées partielles qui est une relation entre plusieurs variables indépendantes (par exemple la variable de temps t et une ou plusieurs variables d'espace) et une variable dépendante.

Les équations différentielles (ED) sont très importantes non seulement pour les mathématiques ou la physique, mais également pour les grands domaines de l'ingénierie et des technologies, de nombreux autres domaines scientifiques et il existe une myriade de problèmes d'ingénierie et scientifiques formulés et simulés à l'aide d'équations différentielles. Par exemple, les trajectoires et les équations du mouvement en chute ou en vol, la décroissance radioactive, la dynamique de processus, problèmes de logistique, équations de propagation d'ondes, problèmes de contrôle, pour ne nommer que ceux-là, sont formulés avec des calculs d'équations différentielles. Il existe de nombreuses définitions d'équations différentielles et l'une des plus simples est « Une équation différentielle est toute équation qui contient des dérivés, soit des dérivés ordinaires, soit des dérivés partielles. »

Bien que modéliser le comportement d'un système, d'un processus ou d'un phénomène à l'aide de ED puisse être relativement facile, mais résoudre nombreux ED analytiquement n'est pas plausible et la seule possibilité est alors de rechercher leurs solutions numériques. Il y a un certain nombre de méthodes pour résoudre les ED analytiquement et numériquement. Par exemple, pour définir la ou les solutions analytiques de la séparation des variables ou l'introduction de nouvelles variables ou en les exprimant sous forme d'équations linéaires ou en les multipliant par certains types de scalaires, sont utilisées en fonction du type de ED, par exemple homogène ou non homogène, ou linéaire ou non linéaire. Pour trouver des solutions numériques aux ED, il existe un certain nombre de méthodes à savoir : Euler, Runge-Kutta, la méthode d'Adams-Bashforth, la méthode d'Adams-Moulton, mise au point la règle du point milieu.

De nos jours, trouver des solutions numériques des équations différentielles utilisent

principalement des ordinateurs et des programmes informatiques. Il existe de nombreux logiciels commerciaux et ouverts pour trouver des solutions numériques d'équations différentielles. L'un de ces logiciels puissants est Matlab qui contient de nombreux outils faciles à utiliser et des fonctions intégrées permettant de résoudre ou de simuler des équations différentielles. Nous étudions quelques outils et fonctions du paquetage afin de montrer comment l'utiliser pour résoudre des problèmes de valeur initiale (IVP) d'équations différentielles ordinaires (EDO).

Décrivons maintenant l'organisation de notre mémoire :

Dans le premier chapitre nous avons collecté certains préliminaires, définitions, théorèmes et autres résultats auxiliaires utilisé dans ce mémoire.

Dans le deuxième chapitre nous avons présenté quelques techniques de résoudre analytiquement des EDO.

Dans le troisième chapitre on a passé à la résolution numérique des EDO, nous utilisons les méthodes les plus connues.

Au dernier chapitre nous introduisons Matlab comme un language moderne pour résoudre les EDO d'une manière plus simple et plus présis.

Chapitre 1

Généralitées

1.1 Équations différentielles ordinaires

Les équations différentielles décrivent l'évolution de nombreux phénomènes dans des domaines variés. Une équation différentielle est une équation impliquant une ou plusieurs dérivées d'une fonction inconnue. Si toutes les dérivées sont prises par rapport à une seule variable, on parle d'équation différentielle ordinaire (EDO).
Une EDO est une équation sous la forme d'une relation :

$$f(t, y(t), y'(t), y''(t), ..., y^{(n)}(t)) = g(t), \tag{1.1}$$

* dont les inconnues sont une fonction $y : I \subset \mathbb{R} \to \mathbb{R}$ et son intervalle de définition I,
$*$ dans laquelle cahabitent à la fois y et ses dérivées y', y'', ..., $y^{(n)}$(n est appelé l'ordre de l'équation).
Si la fonction g, appelée "second membre" de l'équation, est nulle, on dit que l'équation en question est **homogène**.

Résoudre une équation différentielle, c'est chercher toutes les fonctions définies sur un intervalle $I \subset \mathbb{R}$, qui satisfont l'équation (on dit aussi intégrer l'équation différentielle).

Équation différentielle normale : On appelle équation différentielle normale d'ordre n toute équation de la forme :

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

Équation différentielle autonome : On appelle équation différentielle autonome d'ordre n toute équation de la forme :

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

Autrement dit, f ne dépend pas explicitement de t.

1.1.1 Équations différentielles linéaires

Une EDO de type (1.1) d'ordre n est linéaire si elle est de la forme :

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

Solution:
On appelle solution (ou intégrale) d'une équation différentielle d'ordre n sur un certain intervalle I de $\mathbb{R}$, toute fonction y 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 (y, I). Si I contient sa borne inférieure 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.

Solution globale:
Soit $I \in \mathbb{R}$. Une solution (y, I) est dite globale dans I si elle est définie sur I tout entier.

1.1.2 Problème de Cauchy[1]

Etant donné un point $(t_0, y_0) \in U$, (avec U un ouvert de $\mathbb{R} \times \mathbb{R}^m$). Le problème de Cauchy consiste à trouver une solution $y : I \to \mathbb{R}^m$ de (1.1) sur un intervalle I contenant t_0, telle que $y(t_0) = y_0$.
Ce problème est formulé de la manière suivante :

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases} \tag{1.2}$$

Dans la pratique, la variable t représente le temps et y représente l'état d'un système matériel donné. L'équation (1.2) traduit physiquement la loi d'évolution du système considérée en fonction du temps.

[1]Augustin Louis, baron Cauchy (21 août 1789 [Paris]-23 mai 1857 [Sceaux]) : un mathématicien français. Les entrées du Dicomaths correspondant à Cauchy : Critère de Cauchy uniforme, déterminant de Cauchy, équations de Cauchy-Riemann, formule de Cauchy, inégalités de Cauchy, intégrale de Cauchy, introduction aux équations différentielles, lemme de Cauchy, loi de Cauchy, produit de Cauchy de deux séries, quelques inégalités célèbres : Cauchy-Schwarz, Hölder, Minkowski, Jensen, Règle de Cauchy, suite de Cauchy, théorème de Cauchy, théorème de Cauchy-Peano

Solution du problème de Cauchy : Supposons que $f : U \to \mathbb{R}^m$ continue. Soit $(t_0, y_0) \in U$ et y une fonction définie sur un intervalle ouvert I contenant t_0 et à valeurs dans $\mathbb{R}^m$.
Une fonction y est solution de (1.2) sur I si et seulement si :

i pour tout $t \in I, (t, y(t)) \in U$,

ii y est continue sur I,

iii pour tout $t \in I$, $\quad y(t) = y_0 + \int_{t_0}^{t} f(s, y(s))ds.$

1.1.3 Théorème d'existence et unicité locale

a Soit $f : U \to \mathbb{R}^m$ où U un ouvert de $\mathbb{R} \times \mathbb{R}^m$. On dit que f est lipschitzienne par rapport à sa variable y s'il existe une constante $K > 0$ telle que :

$$\forall (t, y) \in U, \forall (t, z) \in U \ : \ |f(t, y) - f(t, z)\| \leq K\|y - z\|.$$

b On dit que f est localement lipschitzienne par rapport à y si tout point $(t, y) \in U$, possède un voisinage appartenant à U et dans lequel f est lipschitzienne par rapport à y.

Le théorème de Cauchy (ou théorème de Cauchy-Lipschitz, ou encore théorème de Picard-Lindelöf chez les anglophones) affirme sous certaines conditions à préciser, que le problème de Cauchy (1.2) admet une unique solution. Plus précisément, on a :

Théorème 1.1.1. *(Existence et unicité locale). Soit*

$$f : U \to \mathbb{R}^m, \ \ (t, y) \to f(t, y),$$

où U un ouvert de $\mathbb{R} \times \mathbb{R}^m$. On suppose que f est continue en (t, y) et localement lipschitzienne par rapport à y. Alors on a les propriétés suivantes :

Existence : *Pour tout point $(t_0, y_0) \in U$, il existe un intervalle fermé I centré en t_0 et une solution locale $y : I \to \mathbb{R}^m$ de l'équation différentielle $y' = f(t, y)$ avec condition initiale $y(t_0) = y_0$.*

Unicité : *En outre $y \in C^1$ et cette solution est unique (c-à-d : si $z : J \to \mathbb{R}^m$, $J \subset I, t_0 \in J$ avec $z(t_0) = y_0$, alors $\forall t \in J, z(t) = y(t)$).*

Régularité : *Si de plus f est de classe C^r, $r \geq 1$, alors y est de classe C^{r+1} .*
Considérons l'équation différentielle (1.2)

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

La solution de cette équation sous les conditions initiales (t_0, y_0) est :

$$y(t) = y_0 + \int_{t_0}^{t} f(s, y(s))ds, t \in I \subset \mathbb{R}.$$

1.1.4 Théorème d'existence et d'unicité globale

Soit $f : U \to \mathbb{R}^m$, $(t, y) \to f(t, y)$, et $y' = f(t, y)$, une équation différentielle.

a Une solution $y : J \to \mathbb{R}^m$ est un prolongement d'une solution $\tilde{y} : I \to \mathbb{R}^m$ si $I \subseteq J$ et $\forall t \in I, \tilde{y}(t) = y(t)$.

b Une solution maximale est une solution qui n'a pas de prolongement strict. Autrement dit, une solution y définie sur I est maximale s'il n'existe pas de solution définie sur un intervalle J contenant strictement I (c-à-d : $I \subseteq J, I \neq J$) dont la restriction à I soit égale à y.

Le théorème global d'existence et d'unicité s'énonce comme suit :

Théorème 1.1.2. *Soit*

$$f : U \to \mathbb{R}^m, \ (t, y) \to f(t, y),$$

une fonction continue en (t, y) et localement lipschitzienne en y sur l'ouvert $U \subset \mathbb{R} \times \mathbb{R}^m$. Soit $(t_0, y_0) \in U$. Alors, il existe une unique solution maximale au problème de Cauchy (1.2).

Démonstration :
Pour la démonstration des théorèmes d'existence et unicité, vous pouvez consulter le livre 'An Introduction to Ordinary Differential Equations', de Ravi P. Agarwal et Donal O'Regan, les pages de 61 à 68.

Chapitre 2

Résolution exacte des EDO

Dans cette section, nous allons intéresser à différentes techniques pour intégrer, certains types d'équations différentielles. Il faut cependant garder à l'esprit que la résolution explicite des EDO n'est pas une chose aisée, et la plupart du temps ce sera trop difficile, voire impossible. Nous devrons alors nous contenter d'une analyse d'existence, unicité, positivité, etc... des solutions.

2.1 Équations de premier ordre

Commençons alors par un cas assez général, et nous irons vers les :

2.1.1 Équations à variables séparables

C'est une équation de la forme :

$$y' = \frac{f(t)}{g(y)},$$

ou sous forme symétrique :

$$g(y)dy - f(t)dt = 0.$$

On a :

$$\int g(y)dy - \int f(t)dt = 0,$$

d'où :

$$G(y) - F(t) = \text{constante}.$$

(G et F étant des primitives de g et f respectivement).

Cas particulier :
Les équations les plus simples sont de la forme :

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

avec $g \equiv 1$ pour tout $t \in I$. On suppose en outre que $y(t_0) = y_0$ pour un $t_0 \in I$.
Si on suppose que f est continue sur un intervalle $I \subset \mathbb{R}$ d'intérieur non vide. Les solutions de cette équation sont données par :

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

2.1.2 Équations linéaires

Nous restons toujours sur les EDO d'ordre 1. Nous nous intéressons ici aux équations différentielles ordinaires linéaires. Une équation différentielle linéaire du premier ordre est du type :

$$a(t)y'(t) + b(t)y(t) = d(t), \tag{2.1}$$

où les fonctions a et b sont données et s'appellent les coefficients de l'équation différentielle et la fonction d est donnée et s'appelle le second membre.

Une solution de (2.1) est une fonction y de classe C^1 sur un intervalle I vérifiant (2.1) pour tout $t \in I$.

EDO linéaire homogène :

On appelle équation différentielle homogène associée à (2.1) l'équation :

$$a(t)y'(t) + b(t)y(t) = 0, \tag{2.2}$$

avec a, b deux fonctions continues sur un intervalle. C'est une équation à variables séparables et sa solution générale est :

$$y_h(t) = Ce^{F(t)},$$

avec $C =$ constante et $F(t) = \displaystyle\int_{t_0}^{t} -\frac{b(t)}{a(t)}dt.$

EDO linéaire avec second membre :

Pour déterminer une solution particulière de l'équation non-homogène. On commence toujours par regarder s'il n'y a pas de solution évidente, sinon on peut appliquer l'une des méthodes suivantes :

1. **Méthode de la variation de la constante**
   On suppose que l'on a trouvé une solution de l'équation homogène de la forme : $y_h(t) = Ce^{F(t)}$.

   - On va chercher une solution particulière sous la forme :

   $$y_p(t) = C(t)e^{F(t)}.$$

   La constante C est maintenant considérée comme une fonction de t.

   - On injecte cette solution particulière dans l'équation (2.1) en calculant :

   $$y_p'(t) = C'(t)e^{F(t)} + c(t)F'(t)e^{F(t)}.$$

   La fonction C vérifie alors :

   $$C'(t) = \frac{d(t)}{a(t)}e^{-F(t)}.$$

   - On cherche une primitive quelconque de $\frac{d(t)}{a(t)}e^{-F(t)}$ afin de trouver $C(t)$ et obtenir $y_p(t)$.

   On a :

   $$C(t) = \int_{t_0}^{t} \frac{d(s)}{a(s)}e^{-F(s)}ds.$$

   Par conséquent,

   $$y_p(t) = e^{F(t)} \int_{t_0}^{t} \frac{d(s)}{a(s)}e^{-F(s)}ds,$$

   est la solution particulière de l'équation non-homogène.

Astuce : Pour résoudre l'EDO $a(t)y'(t) + b(t)y(t) = d(t)$, on peut retenir la formule :

$$y(t) = Ce^{F(t)} + K(t)e^{F(t)} = (C + K(t))e^{F(t)},$$

avec :

* $F(t)$ est une primitive de $-\dfrac{b(t)}{a(t)}$.

* $K(t)$ est une primitive de $\dfrac{d(t)}{a(t)}e^{-F(t)}$.

2. **Cas d'une EDO linéaire du premier ordre à coefficients constants**

Dans ce cas précis, on cherche une solution particulière de l'EDO du premier ordre à coefficients constants $ay'(t) + by(t) = d(t)$ sur l'intervalle I avec $a \neq 0$. Suivant la forme de la fonction d du second membre, différentes formes de solutions particulières peuvent être recherchées.

- Si le second membre est de la forme : $d(t) = \alpha\cos(t) + \beta\sin(t)$ alors on peut chercher une solution sous la forme : $y_p(t) = c_1\cos(t) + c_2\sin(t)$.

- Si le second membre est de la forme : $d(t) = e^{\lambda t}P_n(t)$, où P_n est un polynôme de degré n :

  1^{er} **cas :** si $\lambda \neq r = -\dfrac{b}{a}$, alors on cherche une solution sous la forme $y_p(t) = e^{\lambda t}Q_n(t)$, où Q_n est un polynôme de degré n.

  2^{eme} **cas :** si $\lambda = r = -\dfrac{b}{a}$, alors on cherche une solution sous la forme $y_p(t) = e^{\lambda t}tQ_n(t)$, où Q_n est un polynôme de degré n.

- Si le second membre est de la forme : $d_1(t) = \cos(\lambda t)P_n(t)$ ou de la forme : $d_2(t) = \sin(\lambda t)P_n(t)$, alors on cherche à l'aide de la méthode précédente une solution particulière complexe y_p^c de l'EDO $ay'+by = e^{i\lambda t}P_n(t)$. Une solution particulière de l'EDO avec d_1(resp.d_2) comme second membre est donnée par la partie réelle (resp. partie imaginaire) de y_p^c.

- Si le second membre est de la forme : $d_1(t) = ch(\lambda t)P_n$ ou de la forme : $d_2(t) = sh(\lambda t)P_n$, alors on cherche à l'aide de la méthode précédente une solution particulière y_p^+ de l'EDO $ay' + by = e^{\lambda t}P_n(t)$ et une solution particulière y_p^- de l'EDO $ay' + by = e^{-\lambda t}P_n(t)$.

  Une solution particulière de l'EDO avec d_1 (resp. d_2) comme second membre est donnée par : $y_p = \dfrac{y_p^+ + y_p^-}{2}$ $\left(\text{resp : } y_p = \dfrac{y_p^+ - y_p^-}{2}\right)$.

2.1.3 Équation de Bernoulli:

Ce type d'équations tient son nom du physicien et mathématicien suisse Bernoulli (1654-1705). Une équation de Bernoulli est une équation différentielle non linéaire de la forme

$$a(t)y'(t) + b(t)y(t) = c(t)y^r(t),$$

où $r \in \mathbb{R}$. Divisons l'équation par $y^r(t)$:

$$a(t)\frac{y'(t)}{y^r(t)} + b(t)\frac{1}{y^{r-1}(t)} = c(t).$$

Posons : $u = \dfrac{1}{y^{r-1}}$, d'où : $u' = \dfrac{du}{dt} = \dfrac{du}{dy}\dfrac{dy}{dt} = -(r-1)y^{-r}y' = -(r-1)\dfrac{y'}{y^r}$
Donc on obtient :

$$-\frac{a(t)}{r-1}u'(t) + b(t)u(t) = c(t).$$

C'est une équation linéaire.

Exemple 2.1.1. *Trouve toutes les solutions non nulles de l'EDO suivante :*

$$y' = y + 2y^5.$$

C'est une équation de Bernoulli avec $r = 5$. On dévise l'équation par y^5,

$$\frac{y'}{y^5} = \frac{1}{y^4} + 2.$$

Introduisont la fonction $u = 1/y^4$ et sa dérivée $u' = -4(y'/y^5)$, dans l'équation différentielle ci-dessus,

$$-\frac{u'}{4} = u + 2 \Rightarrow u' = -4u - 8 \Rightarrow u' + 4u = -8.$$

La dernière équation est linéaire par rapport à la fonction u. Elle peut être résolue par la méthode de facteur intégrant. On la multiplue par $v(t) = e^{4t}$, alors :

$$(e^{4t}u)' = -8e^{4t} \Rightarrow e^{4t}u = -\frac{8}{4}e^{4t} + c.$$

On obtient $u = ce^{-4t} - 2$. Puisque $u = 1/y^4$, donc :

$$\frac{1}{y^4} = ce^{-4t} - 2 \Rightarrow y(t) = \pm\frac{1}{(ce^{-4t} - 2)^{1/4}}.$$

2.1.4 Équation exacte

Une équation exacte pour y est de la forme :

$$M(t,y)y' + N(t,y) = 0, \tag{2.3}$$

avec les fonctions M et N sont de classe C^1 et satisfaitent :

$$\partial_t M(t,y) = \partial_y N(t,y).$$

L'équation exacte peut s'écrit comme une équation aux différentielles totales, nommée fonction potentielle qui est facile à résoudre.
Si la forme (2.3) est exacte (i.e: $\exists F$ tq $dF = Mdt + Ndy$ et $dF = 0$). Alors on peut écrire :

$$\frac{d\varphi}{dt}(t, y(t)) = 0,$$

où φ s'appelle fonction potentielle satisfait : $M = \partial_t\varphi, \;\; N = \partial_y\varphi$.
Donc, les solutions de l'équation extacte sont données sous la forme :

$$\varphi(t, y(t)) = C, \quad C \in \mathbb{R}.$$

Exemple 2.1.2. *Trouver toutes les solutions de l'EDO :*

$$2tyy' + 2t + y^2 = 0.$$

La première étape est de vérifier que cette équation différentielle est exacte, on a :

$$\left.\begin{array}{rcl} M(t,y) = 2ty & \Rightarrow & \partial_t M(t,y) = 2y, \\ N(t,y) = 2t + y^2 & \Rightarrow & \partial_y N(t,y) = 2y, \end{array}\right\} \Rightarrow \partial_t M(t,y) = \partial_y N(t,y).$$

Tant que l'EDO est exacte, alors il existe une fonction potentielle φ satisfait :

$$\left\{\begin{array}{l} \partial_t\varphi(t,y) = M(t,y) \\ \partial_y\varphi(t,y) = N(t,y) \end{array}\right. .$$

On intègre :

$$\partial_y\varphi(t,y) = 2ty \Rightarrow \varphi(t,y) = \int 2tydy + g(t),$$

où g est une constante dépent que de t. On obtient :

$$\varphi(t,y) = ty^2 + g(t).$$

De plus :

$$y^2 + g'(t) = \partial_t \varphi(t, y) = M(t, y) = 2t + y^2 \Rightarrow g'(t) = 2t \Rightarrow g(t) = t^2.$$

Donc, la fonction potentielle φ est de la forme :

$$\varphi(t, y) = ty^2 + t^2.$$

Par conséquent, ça implique que toutes les solutions y satisfont l'équation implicite :

$$ty^2(t) + t^2 = C,$$

pour toute $C \in \mathbb{R}$. Le choix $g(t) = t^2 + c_0$ ne modifie que la constante C.

2.2 Équation différentielle linéaire du second ordre

Une équation différentielle linéaire du second ordre est du type :

$$a(t)y''(t) + b(t)y'(t) + c(t)y(t) = g(t), \tag{2.4}$$

où a, b et c sont des fonctions données, appelées coefficients de l'équation différentielle et g est appelée seoncd membre de l'équation différentielle. Une solution de (2.4) est une fonction y de classe C^2 sur un intervalle I vérifiant (2.4) pour tout $t \in I$.

La solution générale de l'EDO de (2.4) s'écrivent :

$$y(t) = y_h(t) + y_p(t),$$

où y_h est solution de l'équation homogène associée et y_p une solution particulière de (2.4).

2.2.1 Solution de l'équation homogène

Soit:

$$a(t)y''(t) + b(t)y'(t) + c(t)y(t) = 0, \tag{2.5}$$

ontrairement aux EDO linéaires homogènes du premier ordre, on n'a pas d'expression explicite des solutions lorsque les coefficients sont non constants. Commençons par donner la structure des solutions d'une EDO linéaire homogène du second ordre :
Soit I un intervalle où les fonctions a, b et c sont définies et continues et tel que $a(t) \neq 0$, pour tout $t \in I$. Les solutions de l'équation homogène (2.5) sont de la forme :

$$y(t) = \lambda y_1(t) + \mu y_2(t),$$

où λ et μ sont des constantes arbitraires et y_1 , y_2 deux solutions linéairement indépendantes. Autrement dit, l'ensemble des solutions de (2.5) est un espace vectoriel de dimension 2.

La solution de l'équation (2.5) dépend ainsi de deux constantes arbitraires, soit 2 degrés de liberté. Il suffit donc de connaître deux solutions linéairement indépendantes de (2.5) pour les connaître toutes. En réalité, grâce à la méthode de réduction d'ordre décrite ci-après, il suffit de connaître une solution de l'équation homogène pour pouvoir en construire une deuxième, linéairement indépendante, et ainsi les obtenir toutes :

Méthode de réduction d'ordre. C'est une variante de la méthode de la variation de la constante. Le principe est le suivant :

Étape 1 "Deviner" une solution y_1 de l'équation (2.5): regarder la forme de l'équation, essayer par exemple des polynômes : t, t^2, e^t (si $a(t) + b(t) + c(t) = 0, y(t) = e^t$ est solution !), etc.

Étape 2 On cherche une solution y_2 de (2.5) sous la forme : $y_2(t) = C(t)y_1(t)$ avec C fonction non constante.

Étape 3 La solution générale de (2.5) s'écrit alors : $y(t) = \lambda y_1(t) + \mu y_2(t)$ avec $\lambda, \mu \in \mathbb{R}$ (ou $\mathbb{C}$ si on cherche des solutions complexes).

Exemple 2.2.1. *Soit à résoudre :*

$$\begin{cases} y'' = -2t(y')^2 \\ y(0) = 2 \ , y'(0) = -1. \end{cases}$$

Posons : $v = y' \Rightarrow v' = y''$.
Alors : $v' = -2tv^2 \Rightarrow \dfrac{v'}{v^2} \Rightarrow -\dfrac{1}{v} = -t^2 + c.$
D'où : $\dfrac{1}{y'} = t^2 - c \Leftrightarrow y' = \dfrac{1}{t^2 - c}.$

Les conditions initiales impliquent :

$$-1 = y'(0) = -\frac{1}{c} \Rightarrow c = 1 \Rightarrow y' = \frac{1}{t^2 - 1}.$$

On obtient donc : $y = \displaystyle\int \frac{dt}{t^2 - 1} + c.$
On intègre :

$$\frac{1}{t^2 - 1} = \frac{1}{(t-1)(t+1)} = \frac{a}{t-1} + \frac{b}{t+1}.$$

Par identification on aura : $a = \frac{1}{2}$, $b = -\frac{1}{2}$. *Et donc :*

$$\frac{1}{t^2-1} = \frac{1}{2}\left[\frac{1}{t-1} - \frac{1}{t+1}\right].$$

L'intégrale donc est facile, nous obtenons :

$$y = \frac{1}{2}\left(ln|t-1| - ln|t+1|\right) + c, \quad 2 = y(0) = \frac{1}{2}(0-0) + c.$$

D'où :

$$y = \frac{1}{2}\left(ln|t-1| - ln|t+1|\right) + 2.$$

Cas particulier des EDO homogènes à coefficients constants

Une équation différentielle linéaire du second ordre à coefficients constants est du type :

$$ay''(t) + by'(t) + cy(t) = 0, \tag{2.6}$$

où les réels a, b et c sont donnés dans $\mathbb{R}^* \times \mathbb{R} \times \mathbb{R}$.
On appelle équation caractéristique associée à (2.6) :

$$ar^2 + br + c = 0.$$

Notons : $\Delta = b^2 - 4ac$ son discriminant.

1. **Si** $\Delta > 0$ alors l'équation caractéristique admet deux solutions $r_1 \neq r_2$ réelles. La solution générale de (2.6) s'écrit :

   $y(t) = C_1 e^{r_1 t} + C_2 e^{r_2 t}$, avec : $C_1, C_2 \in \mathbb{R}$ constantes arbitraires.

2. **Si** $\Delta < 0$ alors l'équation caractéristique admet deux solutions complexes conjuguées $r = \alpha + i\beta$ et $\bar{r} = \alpha - i\beta$. Comme précédemment on a donc :

   - $y(t) = C_1 e^{rt} + C_2 e^{\bar{r}t}$, avec: $r, C_1, C_2 \in \mathbb{C}$, **solutions complexes**,
   - ou bien si on veut les solutions réelles de (2.6) :
     $y(t) = e^{\alpha t}(A\cos(\beta t) + B\sin(\beta t))$, avec $A, B \in \mathbb{R}$, **solutions réelles**.

3. **Si** $\Delta = 0$ alors l'équation caractéristique admet une unique solution $r = -\frac{b}{2a}$ (racine double) et les solutions de (2.6) s'écrivent :

   $y(t) = (A + Bt)e^{rt}$, où A et B sont deux constantes arbitraires réelles.

2.2.2 Recherche d'une solution particulière

Revenons à l'EDO complète :

$$a(t)y''(t) + b(t)y'(t) + c(t)y(t) = g(t),$$

dont on cherche une solution particulière. On commence toujours par regarder s'il n'y a pas de solution évidente, sinon on peut appliquer l'une des méthodes suivantes :

Cas d'une EDO à coefficients constants

• Si le second membre est de la forme $g(t) = \alpha\cos(t) + \beta\sin(t)$ alors on peut chercher une solution sous la forme : $y_p(t) = c_1\cos(t) + c_2\sin(t)$.

• Si le second membre est de la forme $g(t) = e^{\lambda t}P_n(t)$, avec P_n polynôme de degré n :

1. Si $a\lambda^2 + b\lambda + c \neq 0$ i.e: $\lambda \neq r_1$ et $\lambda \neq r_2$, on cherche une solution sous la forme $y_p(t) = e^{\lambda t}Q_n(t)$ où Q_n est un polynôme de degré n.
2. Si $a\lambda^2 + b\lambda + c = 0$ et $2a\lambda + b \neq 0$ i.e : si $\lambda = r_1$ ou $\lambda = r_2$ avec $r_1 \neq r_2$, on cherche une solution sous la forme $y_p(t) = e^{\lambda t}tQ_n(t)$ où Q_n est un polynôme de degré n.
3. Si $\lambda = r_1 = r_2$, on cherche une solution sous la forme $y_p(t) = e^{\lambda t}t^2Q_n(t)$ où Q_n est un polynôme de degré n.

Cas général - Méthode de variation des constantes

Le principe est le suivant : on a trouvé une solution de l'équation homogène de la forme: $y_h(t) = Ay_1(t) + By_2(t)$. On va chercher une solution particulière sous la forme :

$$y_p(t) = A(t)y_1(t) + B(t)y_2(t).$$

En imposant aux fonctions $A(t)$ et $B(t)$ de vérifier la condition

$$A'(t)y_1(t) + B'(t)y_2(t) = 0.$$

On va être amenés à chercher des fonctions A et B vérifiant le système suivant :

$$(S)\begin{cases} y_1(t)A'(t) + y_2(t)B'(t) = 0 \\ y_1'(t)A'(t) + y_2'(t)B'(t) = \dfrac{g(t)}{a(t)} \end{cases}$$

Ce système est un système linéaire en $(A'(t), B'(t))$ de déterminant :

$$W(t) = \begin{vmatrix} y_1(t) & y_2(t) \\ y_1'(t) & y_2'(t) \end{vmatrix},$$

appelé Wronskien de y_1 et y_2 . Le système (S) a-t-il des solutions ? la réponse est oui car son déterminant $W(t)$ est non nul :

$$W(t) = y_1(t)y_2'(t) - y_1'(t)y_2(t) = y_1^2(t)\left(\frac{y_2}{y_1}\right)'(t) \neq 0.$$

Un calcul direct montre que le wronskien de deux solutions vérifie la relation

$$W'(t) = \frac{-b(t)}{a(t)}W(t).$$

Puisque y_1 et y_2 sont linéairement indépendants. On calcule donc $A'(t)$ et $B'(t)$ solutions du système (S) :

$$A'(t) = -\frac{g(t)y_2(t)}{a(t)W(y_1(t), y_2(t))} \text{ et } B'(t) = \frac{g(t)y_1(t)}{a(t)W(y_1(t), y_2(t))}.$$

Puis on intègre pour trouver $A(t)$ et $B(t)$ et en déduire $y_p(t)$.

Exemple 2.2.2. *Soit à résoudre l'EDO suivante :* $y''(t) + y(t) = \tan(t)$

La solution générale de l'EDO homogène est donnée par :

$$y_h(t) = c_1 \sin(t) + c_2 \cos(t).$$

On suppose donc que $y_p(t) = c_1(t)\sin(t) + c_2(t)\cos(t)$. *Alors,*

$$\begin{cases} y_p'(t) = c_1'(t)\sin(t) + c_2'(t)\cos(t) + c_1(t)\cos(t) - c_2(t)\sin(t) \\ y_p''(t) = c_1'(t)\cos(t) - c_2'\sin(t) - c_1(t)\sin(t) - c_2(t)\cos(t) \end{cases}.$$

On impose la condition : $c_1'(t)\sin(t) + c_2'(t)\cos(t) = 0$, *et en substituant dans l'EDO initiale, on obtient le système :*

$$(S)\begin{cases} c_1'(t)\sin(t) + c_2'(t)\cos(t) = 0 \\ c_1'(t)\cos(t) - c_2'(t)\sin(t) = \tan(t). \end{cases}$$

Le déterminant de ce système est le Wronskien de $(\sin(t), cos(t))$ *:*

$$W(\sin(t), cos(t)) = \begin{vmatrix} \sin(t) & \cos(t) \\ \cos(t) & -\sin(t) \end{vmatrix} = -1.$$

On obtient les deux solutions : $c_1'(t) = \sin(t)$ *et* $c_2'(t) = \cos(t) - \dfrac{1}{\cos(t)}$.
Il reste à intégrer ces deux fonctions :

$$c_1(t) = -\cos(t) + c_3 \text{ et } c_2(t) = \sin(t) - \ln\left|\frac{1}{\cos(t)} + \tan(t)\right| + c_4.$$

Une solution particulière de l'EDO complète est donc obtenue pour $c_3 = c_4 = 0$ *:*

$$\begin{aligned} y_p(t) &= c_1(t)\sin(t) + c_2(t)\cos(t) \\ &= -\cos(t)\sin(t) + \left(\sin(t) - \ln\left|\frac{1}{\cos(t)} + \tan(t)\right|\right)\cos(t) \\ &= \left(-\ln\left|\frac{1}{\cos(t)} + \tan(t)\right|\right)\cos(t). \end{aligned}$$

2.3 Systèmes différentiels linéaires

Dans cette section nous allons nous intéresser aux systèmes d'équations différentielles, que l'on peut obtenir directement par la modélisation d'un problème à plusieurs fonctions inconnues, mais également lorsque l'on passe d'une EDO d'ordre n à un système de plusieurs EDO d'ordre 1. Nous ne le faisons ici que pour le cas particulier des systèmes linéaires.

2.3.1 Théorie préliminaire

Soient I un intervalle de $\mathbb{R}$, $n \in \mathbb{N}^*$, $a_{i,j} : I \to \mathbb{R}, i, j = 1, ..., n$ et $f_i : I \to \mathbb{R}$ des fonctions continues. L'objectif est de trouver des fonctions $y_1, ..., y_n : I \to \mathbb{R}$, n fonctions de classe C^1 sur I telles que :

$$\begin{cases} y_1' = a_{11}(t)y_1 + ... + a_{1n}(t)y_n + f_1(t), \\ \vdots \\ y_n' = a_{n1}(t)y_1 + ... + a_{nn}(t)y_n + f_n(t). \end{cases} \tag{2.7}$$

On peut écrire ce système sous la forme matricielle :

$$Y'(t) = A(t)Y(t) + F(t),$$

où :

$$Y(t) = \begin{pmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{pmatrix}, A(t) = \begin{pmatrix} a_{11}(t) & \dots & a_{1n}(t) \\ \vdots & & \vdots \\ a_{n1}(t) & \dots & a_{nn}(t) \end{pmatrix} \quad et \quad F(t) = \begin{pmatrix} f_1(t) \\ \vdots \\ f_n(t) \end{pmatrix}.$$

En général il peut y avoir une infinité de solutions de cette équation.
Soient $t_0 \in I$ et $Y^0 \in \mathbb{R}^n$ données, avec :

$$Y^0 = \begin{pmatrix} y_1^0(t) \\ \vdots \\ y_n^0(t) \end{pmatrix}. \tag{2.8}$$

Le but est de trouver Y solution de l'équation (2.7) satisfaisant la condition initiale (2.8). Autrement dit, existe-t-il Y fonction dérivable définie sur I à valeurs dans $\mathbb{R}^n$ tel que

$$\begin{cases} Y'(t) = A(t)Y(t) + F(t), \\ Y(t_0) = Y^0. \end{cases} \tag{2.9}$$

pour tout $t \in I$?

Théorème 2.3.1. *Si $A : I \to \mathcal{M}_n(\mathbb{R})$ et $F : I \to \mathbb{R}^n$ sont continues, alors pour tout $t_0 \in I$ et pour tout $Y^0 \in \mathbb{R}^n$, il existe une solution unique au problème de Cauchy* (2.9)*.*

2.3.2 Systèmes homogènes

Le système (2.7) est dit homogène si $F \equiv 0$, c'est à dire :

$$Y'(t) = A(t)Y(t), \tag{2.10}$$

et nous avons l'existence et l'unicité des solutions de ce système dans le théorème suivant :

Théorème 2.3.2. *L'ensemble H des solutions d'un système homogène est un espace vectoriel de dimension n.*

2.3.3 Solution du système homogène

Soient $Y^1, ..., Y^n$ un ensemble fondamental de solutions de(2.10). Alors toute solution Y de (2.10) est de la forme :

$$Y(t) = \sum_{i=1}^{n} c_i Y^i(t),$$

avec $c_1, ..., c_n \in \mathbb{R}$.

Remarque 2.3.3. *Soient $Y^1, ..., Y^n : I \to \mathbb{R}$ solutions de* (2.10)*. Si les n fonctions sont indépendantes, on dit qu'ils forment un ensemble fondamental de solution de* (2.10)*. On notera alors :*

$$M(t) = \left(Y^1(t)|...|Y^n(t)\right) = \begin{pmatrix} y_{11}(t) & \dots & y_{1n}(t) \\ \vdots & & \vdots \\ y_{n1}(t) & \dots & y_{nn}(t) \end{pmatrix},$$

la matrice $n \times n$ qu'on appellera matrice fondamentale du système (2.10).
• *$M(t)$ est inversible, et de plus on a : $M'(t) = A(t)M(t)$.*

2.3.4 Systèmes non homogènes

Soient $Y^1, ..., Y^n$ un ensemble fondamental de solutions de (2.10) et Y_p une solution particulière de (2.7). Alors toute solution Y de (2.7) est de la forme :

$$Y(t) = Y_p + \sum_{i=1}^{n} c_i Y^i(t),$$

avec $c_1, ..., c_n \in \mathbb{R}$.

Comment trouver une solution particulière Y_p alors ? C'est par la :

Variation de la constante :

On va chercher un Y_p sous la forme :

$$Y_p(t) = \sum_{i=1}^{n} Y^i(t)\gamma_i(t),$$

où $\gamma_i : I \to \mathbb{R}$ est à trouver.
On obtient :

$$Y_p' = M\gamma' A + Y_p,$$

d'une part, et d'autre part on aimerait que Y_p satisfasse le système non-homogène :

$$Y_p' = MAY_p + F,$$

En identifiant, cela revient à chercher γ solution de :

$$M\gamma' = F,$$

et comme M est inversible, on doit donc trouver γ telle que :

$$\gamma' = M^{-1}F.$$

Par conséquent, un choix possible pour Y_p sera :

$$Y_p = M\gamma = M\int_{t_0}^{t} M^{-1}(s)F(s)ds,$$

pour un $t_0 \in I$ fixé.
On déduit alors du théorème que les solutions du problème non homogène sont de la

forme :

$$Y = Y_p + Y_F = M(t) \int_{t_0}^{t} M^{-1}(s)F(s)ds + M(t)C,$$

avec $C \in \mathbb{R}^n$.
Si en plus, on fixe $t_0 \in I$ et $Y^0 \in \mathbb{R}^n$ et on cherche la solution du problème de Cauchy, le vecteur $C \in \mathbb{R}^n$ est donné par :

$$C = M^{-1}(t_0)Y^0.$$

Alors l'unique solution du problème est donnée par :

$$Y(t) = M(t)M^{-1}(t_0)Y^0 + M(t) \int_{t_0}^{t} M^{-1}(s)F(s)ds.$$

2.3.5 Systèmes linéaires à coefficients constants

Nous allons dans cette section considérer un cas particulier de la section précédente. C'est le cas avec A est constante.

Solution du système homogène $Y'(t) = AY(t)$:

Le système différentielle linéaire homogène possède toujours n solutions linéairement indépendantes et toute solution Y peut s'exprimer comme combinaison linéaire de n solutions linéairement indépendantes $Y^1, Y^2, ..., Y^n$:

$$Y(t) = c_1Y^1(t) + c_2Y^2(t) + ... + c_nY^n(t),$$

où les c_i sont des constantes bien choisies. De plus, la solution générale peut s'écrire comme :

$$Y(t) = e^{At}Y^0,$$

où $Y^0 = Y(0) \in \mathbb{R}^n$ est arbitraire.

Exponentielle de A

Soit $A \in \mathcal{M}_{\backslash}(\mathbb{R})$ une matrice carrée de taille n, alors :

$$e^A = \sum_{n=0}^{+\infty} \frac{A^n}{n!} = 1 + A + \frac{A^2}{2} + \frac{A^3}{3!} + ...$$

L'exponentielle d'une matrice diagonalisable se calcule comme suit. Si la matrice A peut se diagonaliser comme :

$$A = P \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & \lambda_n \end{pmatrix} P^{-1},$$

avec P la matrice de passage (matrice des vecteurs propres), alors on peut écrire que :

$$e^A = \begin{pmatrix} e^{\lambda_1} & 0 & \dots & 0 \\ 0 & e^{\lambda_2} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & e^{\lambda_n} \end{pmatrix} P^{-1}.$$

On peut également obtenir de manière assez similaire une formule donnant l'exponentielle d'une matrice non diagonalisable. Pour cela il faut mettre la matrice sous sa forme de Jordan.

Propriétés de l'exponentielle d'une matrice :

- $e^0 = I$.
- Si les matrices A et B commutent ($AB = BA$), alors : $e^{A+B} = e^A e^B = e^B e^A$.
- $(e^A)^{-1} = e^{-A}$.
- $e^A A = A e^A$.
- $\dfrac{d}{dt} e^{At} = A e^{At}$.

Cas où A a n valeurs propres distinctes

Dans le cas où la matrice A est diagonalisable, alors la solution générale de $Y'(t) = AY(t)$ est de la forme :

$$Y(t) = c_1 e^{\lambda_1 t} V_1 + \cdots + c_1 e^{\lambda_n t} V_n = \sum_{i=1}^{n} c_i e^{\lambda_i t} V_i,$$

où :

- λ_i sont **les valeurs propres** de A,
- V_i sont **les vecteurs propres associés** ($(V_1, ...V_n)$ forme une base de vecteurs propres),
- c_i sont des **constantes arbitraires réelles** (ou **imaginaires** si $\lambda_i \in \mathbb{C}$).

Solution du système non homogène

Revenons au système avec second membre :

$$Y'(t) = AY(t) + F(t).$$

Si A est constante alors la solution est donnée par :

$$Y(t) = e^{A(t-t_0)}Y^0 + \int_{t_0}^{t} e^{A(t-s)}F(s)ds.$$

Exemple 2.3.4. *Cherchons à résoudre le problème de Cauchy linéaire de premier ordre non homogène :*

$$\begin{cases} y_1'(t) = y_1(t) + e^{-t}, & y_1(0) = 1 \\ y_2'(t) = -y_2(t) - 2e^{-2t}, & y_2(0) = 1 \end{cases}.$$

Si l'on pose $Y(t) = (y_1(t), y_2(t))$ *alors le problème de Cauchy s'écrit sous forme matricielle :*

$$\begin{cases} Y'(t) = AY(t) + F(t) \\ Y(0) = Y_0, \end{cases}$$

$$\textit{avec : } A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, Y_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \textit{ et } F(t) = \begin{pmatrix} e^{-t} \\ -2e^{-2t} \end{pmatrix}.$$

Dès que la matrice A *est diagonale, l'exponentielle* e^{At} *est simplement :*

$$e^{At} = \begin{pmatrix} e^{t} & 0 \\ 0 & e^{-t} \end{pmatrix},$$

ainsi la solution du problème homogène est :

$$Y_h(t) = e^{At}Y_0 = \begin{pmatrix} e^{t} \\ e^{-t} \end{pmatrix}.$$

Calculons maintenant le terme : $\int_0^t e^{A(t-s)}F(s)ds.$

$$e^{A(t-s)}F(s) = \begin{pmatrix} e^{t-s} & 0 \\ 0 & e^{-(t-s)} \end{pmatrix} \begin{pmatrix} e^{-s} \\ -2e^{-2s} \end{pmatrix} = \begin{pmatrix} e^{t-2s} \\ -2e^{-t-s} \end{pmatrix}.$$

$$\int_0^t e^{A(t-s)}F(s)ds = \begin{pmatrix} \int_0^t e^{t-2s}ds \\ \int_0^t -2e^{-t-s}ds \end{pmatrix} = \begin{pmatrix} -\frac{1}{2}e^{t}(e^{-2t}-1) \\ 2e^{-t}(e^{-t}-1) \end{pmatrix}.$$

La solution du problème de Cauchy est donc :

$$Y(t) = \begin{pmatrix} e^{t} - \frac{1}{2}e^{t}(e^{-2t}-1) \\ e^{-t} + 2e^{-t}(e^{-t}-1) \end{pmatrix}.$$

La solution graphique avec Matlab est donnée par le graphe suivant :

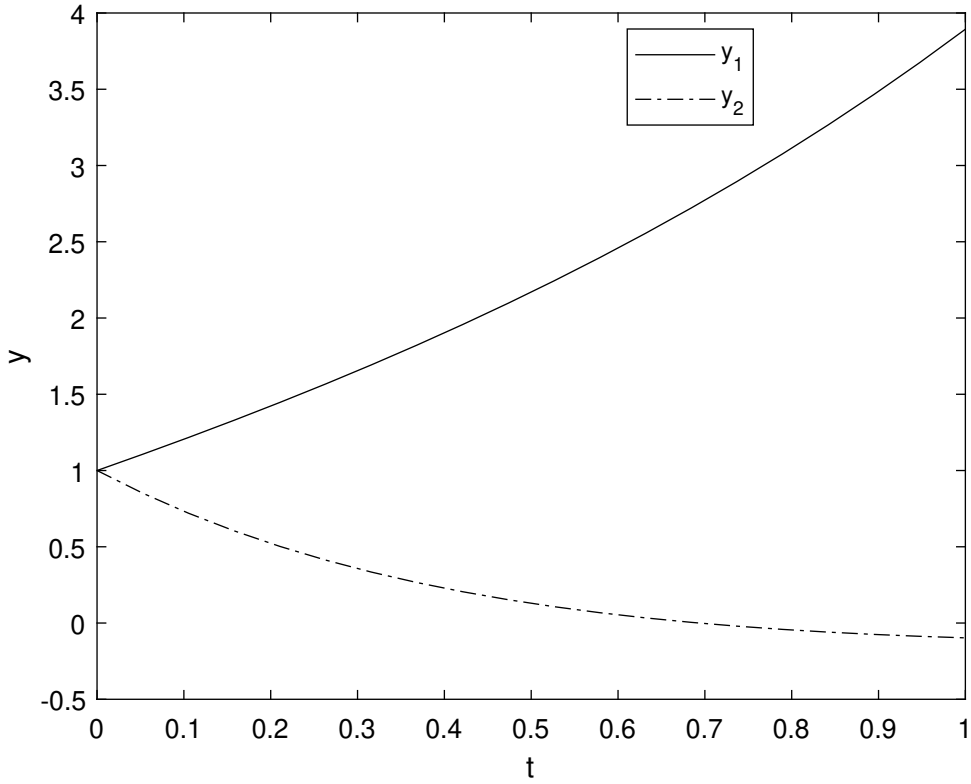


Figure 2.1: Solution graphique de l'exemple 2.3.5

Chapitre 3

Résolution numérique des EDO

Dans la grande majorité des cas, on ne connaît pas de solution exacte à une équation différentielle : les solutions analytiques sont des exceptions. On sait qu'une solution exacte existe, mais on ne peut pas en dire plus. À défaut d'une solution exacte, on peut résoudre numériquement une équation différentielle, c'est-à-dire obtenir une valeur numérique approchée $y(t)$ pour un ensemble de temps discrets. L'objectif est que lorsque le pas de temps diminue, la solution approchée converge vers la solution exacte.

Définition 3.0.1. *On appelle méthode à un pas, la méthode permettant de calculer y_{n+1} à partir de la seule variable antérieure y_n. Une méthode à r pas est une méthode qui utilise les r valeurs antérieures $y_n, ..., y_{n-r+1}$ afin de faire le calcul de y_{n+1}.*

Parmi ces méthodes on peut citer la :

3.1 Méthode d'Euler[1]

Considérons le problème :

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0, t \in [t_0, t_0 + T] \end{cases} \tag{3.1}$$

[1]Leonhard Euler (15 avril 1707 [Bâle] - 7 septembre 1783 [Saint-Pétersbourg]) : mathématicien et physicien suisse. Les entrées du Dicomaths correspondant à Euler : Constante d'Euler, les séries à l'hypothèse de Riemann, droite et cercle d'Euler, fonction Gamma, fonction homogène, formule d'Euler-pour les nombres complexes, formule d'Euler-MacLaurin, indicateur d'Euler, méthode d'Euler, opérateur différentiel, ponts de Königsberg et cycle eulérien.

Subdivisons l'intevalle $[t_0, t_0 + T]$ comme suit :

$$t_0 < t_1 < ... < t_n < t_{n+1} < ... < t_N = t_0 + T.$$

Posons :

$$h_n = t_{n+1} - t_n; \qquad h = \max_{0 \leq n \leq N} h_n,$$

pour $0 \leq n \leq N$, la solution du problème (3.1) est donnée par :

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(s, y(s))ds.$$

Cette solution est approchée sur l'intervalle $[t_n, t_{n+1}]$ par sa tangente au point t_n.
Et ainsi, on a :

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

d'où :

$$y(t_{n+1}) = y(t_n) + h_n f(t_n, y(t_n)).$$

Par récurrence, nous construisons une approximation y_n de $y(t_n)$ i.e: $(y_n \approx y(t_n))$ en remplaçant la relation précédente par :

$$y_{n+1} = y_n + h_n f(t_n, y_n), \quad n = 0, 1, ..., N - 1. \tag{3.2}$$

Ce qui revient à approcher, pour $s \in]t_n, t_{n+1}[$, $f(s, y(s))$ par $f(t_n, y_n)$. Le schéma défini par la relation (3.2) s'appelle le schéma d'Euler.

Précision de la méthode d'Euler :

La méthode d'Euler est une méthode du premier ordre , c'est-à-dire que l'erreur au point t_n s'exprime par l'inégalité :

$$|y_n - y(t_n)| \leq kh_n,$$

avec : y_n est la valeur approchée, $y(t_n)$ est la valeur exacte, et k une constante indépendante de n et de h_n.

Exemple 3.1.1. *Considérons le problème de Cauchy :*

$$\begin{cases} y' = t + y \\ y(0) = 1, \quad t \in [0, 1] \end{cases} \tag{3.3}$$

À l'aide de la méthode d'Euler, en subdivisant l'intervalle $[0, 1]$ *en dix parties égales. Selon l'algorithme d'Euler, on a :*

$$\begin{cases} y_{n+1} = y_n + h_n f(t_n, y_n), \quad 0 \leq n \leq 9 \ et \ h_n = 0.1 \\ h_n = t_{n+1} - t_n \end{cases}$$

Calculant par exemple y_1*, on a :* $y_1 = y_0 + h_n f(t_0, y_0) = 1 + 0.1 f(0, 1) = 1.1$.

Et delà on obtient les valeurs du tableau :

n	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
t_n	*0*	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*	*1.0*
y_n*appr*	*1*	*1.1*	*1.22*	*1.362*	*1.5282*	*1.7210*	*1.9431*	*2.1974*	*2.4871*	*2.8158*	*3.1874*
y_n*exc*	*1*	*1.1103*	*1.2428*	*1.3997*	*1.5836*	*1.7974*	*2.0442*	*2.3275*	*2.6510*	*3.0192*	*3.4365*
l'erreur	*0*	*0.0103*	*0.0228*	*0.0377*	*0.0554*	*0.0764*	*0.1011*	*0.1301*	*0.1639*	*0.2034*	*0.2491*

On trouve : $y(1) = y_{10} \simeq y(t_{10}) \simeq 3.187$.
La solution exacte de l'équation (3.3) est donnée par : $y(t) = 2e^t - t - 1$*, ce qui donne :*
$y(1) = 3.437$.
L'approximation calculée est donc trés grossière.

À l'aide de Matlab, on obtient :

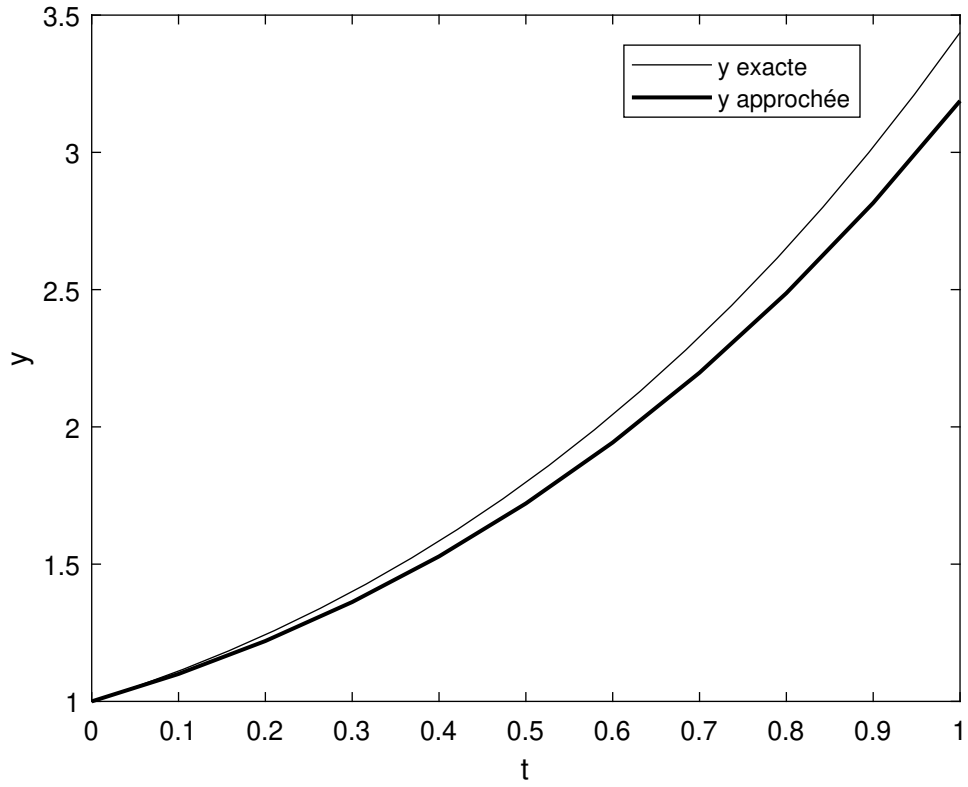


Figure 3.1: Solution graphique de l'exemple 3.1.1

3.2 Méthode de Taylor[2]

Supposons que f soit de classe C^1. Alors $y(t)$ est de classe C^2 , et le développement de Taylor d'ordre 2 implique :

$$y(t_{n+1}) = y(t_n + h_n) = y(t_n) + h_n y'(t_n) + \frac{h_n^2}{2!} y''(t_n) + O(h_n^2).$$

Comme :

$$y'(t_n) = f(t_n, y(t_n)),$$

et :

$$y''(t_n) = \frac{d}{dt}(y'(t))|_{t=t_n} = \frac{d}{dt}(f(t, y(t)))|_{t=t_n} = \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}.\frac{dy}{dt}\right)|_{t=t_n}$$

$$= \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}.y'\right)|_{t=t_n} = \frac{\partial f}{\partial t}(t_n, y(t_n)) + \frac{\partial f}{\partial y}(t_n, y(t_n)) f(t_n, y(t_n)).$$

Il vient :

$$y(t_{n+1}) = y(t_n) + h_n f(t_n, y(t_n)) + \frac{h_n^2}{2}\left[\frac{\partial f}{\partial t}(t_n, y(t_n)) + \frac{\partial f}{\partial y}(t_n, y(t_n)) f(t_n, y(t_n))\right] + O(h_n^2).$$

On est amené à considérer l'algorithme suivant , appelée : l'algorithme de Taylor d'ordre 2 :

$$\begin{cases} y(t_{n+1}) = y(t_n) + h_n f(t_n, y(t_n)) + \dfrac{h_n^2}{2}\left[\dfrac{\partial f}{\partial t}(t_n, y(t_n)) + \dfrac{\partial f}{\partial y}(t_n, y(t_n)) f(t_n, y(t_n))\right] \\ h_n = t_{n+1} - t_n. \end{cases}$$

Précision de la méthode de Taylor :

La méthode de Taylor d'ordre 2 est une méthode du second ordre. Autrement dit, l'erreur au point t_n vérifie :

$$|y_n - y(t_n)| \leq k h_n^2,$$

On peut réduire la marge d'erreur, en comptant plus de termes dans le développement de Taylor c'est-à-dire, on suppose $f \in C^p$, et donc $y \in C^{p+1}$ et sa dérivée k ème est :

$$y^{(k)}(t) = f^{[k-1]}(t, y(t)).$$

[2]Brook Taylor 18 août 1685 [Edmonton] - 29 décembre 1731 [Londres]. Renommé pour : Théorème de Taylor, Série de Taylor, Développement de Taylor.

On obtient donc l'algorithme de Taylor d'ordre p suivante :

$$\begin{cases} y_{n+1} = y_n + h_n f(t_n, y_n) + \sum_{k=1}^{p} \frac{h_n^k}{k} f^{(k-1)}(t_n, y_n), \\ t_{n+1} = t_n + h_n. \end{cases}$$

3.3 Méthode du point milieu

L'idée est que la corde de la fonction y sur $[t, t+h]$ a une pente voisine de $y'(t+\frac{h}{2})$, alors que dans la méthode d'Euler on approxime brutalement cette pente par $y'(t)$.
On écrit donc :

$$y'(t+\frac{h}{2}) = \frac{y(t+h) - y(t)}{h} + O(h^2).$$

Donc :

$$y(t+h) \simeq y(t) + hy'(t+\frac{h}{2}).$$

Puisque :

$$y'(t+\frac{h}{2}) = f(t+\frac{h}{2}, y(t+\frac{h}{2})).$$

Et comme la valeur de $y(t+\frac{h}{2})$ n'est pas connue, on l'approxime par :

$$y(t+\frac{h}{2}) \simeq y(t) + \frac{h}{2} f(t, y(t)),$$

d'où :

$$y(t+h) \simeq y(t) + hf\left(t+\frac{h}{2}, y(t) + \frac{h}{2} f(t, y(t))\right).$$

L'algorithme du point milieu est associé au choix :

$$\phi(t, y, h) = f\left(t+\frac{h}{2}, y + \frac{h}{2} f(t, y)\right).$$

On peut donc s'écrire :

$$\begin{cases} y_{n+\frac{1}{2}} = y_n + \frac{h_n}{2} f(t_n, y_n), \\ \\ p_n = f\left(t_n + \frac{h_n}{2}, y_{n+\frac{1}{2}}\right), \\ y_{n+1} = y_n + h_n p_n, \\ t_{n+1} = t_n + h_n. \end{cases}$$

3.4 Méthode de Runge Kutta[3]

Les méthodes de type Runge-Kutta permettent d'obtenir une plus grande précision que les autres méthodes, elles donnent en général des solutions numériques plus proches des solutions analytiques. Cette précision est obtenue par l'utilisation d'un pas de calcul intermédiare. Les deux méthodes de Runge-Kutta les plus employées sont l'algorithme dit (R-K2) à deux pas de calcul et l'algorithme dit (R-K4) à quatre pas de calcul.

Principe général de la méthode

On considère le problème de Cauchy :

$$\begin{cases} y'(t) = f(t, y(t)), \\ y(t_0) = y_0, t \in [t_0, t_0 + T]. \end{cases}$$

On reprend l'algorithme de Taylor en écrivant la seconde équation de la manière suivante

$$y_{n+1} = y_n + \frac{h_n}{2} f(t_n, y_n) + \frac{h_n}{2} \left[f(t_n, y_n) + h_n \frac{\partial f}{\partial t}(t_n, y_n) + h_n \frac{\partial f}{\partial y}(t_n, y_n) f(t_n, y_n) \right].$$

On a :

$$\left[f(t_n, y_n) + h_n \frac{\partial f}{\partial t}(t_n, y_n) + h_n \frac{\partial f}{\partial y}(t_n, y_n) f(t_n, y_n) \right] = \frac{2}{h_n} \left[y_{n+1} - y_n - \frac{h_n}{2} f(t_n, y_n) \right].$$

En utilisant les approximations : $\begin{cases} y_n \approx y(t_n), \\ \int_a^b f(t)dt \simeq \dfrac{b-a}{2} \left[f(a) + f(b) \right], \end{cases}$ (approximation de Trapèze).

On a :

$$y_{n+1} - y_n \simeq y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t))dt.$$

$$y_{n+1} - y_n = \frac{h_n}{2} \left[f(t_n, y_n) + f(t_{n+1}, y_{n+1}) \right].$$

D'où :

$$\frac{2}{h_n} \left[y_{n+1} - y_n - \frac{h_n}{2} f(t_n, y_n) \right] = f(t_{n+1}, y_{n+1}).$$

Donc on aura :

$$\left[f(t_n, y_n) + h_n \frac{\partial f}{\partial t}(t_n, y_n) + h_n \frac{\partial f}{\partial y}(t_n, y_n) f(t_n, y_n) \right] = f\left(t_n + h_n, y_n + h_n f(t_n, y_n) \right).$$

[3]Carl David Tolmé Runge 30 août 1856 et mort le 3 janvier 1927) : un mathématicien et physicien allemand. Martin Wilhelm Kutta : (3 novembre 1867 et mort le 25 décembre 1944) : un mathématicien allemand. Ils élaborèrent la méthode qui prends leurs noms "Runge Kutta" en 1901.

Finalement , on obtient :

$$y_{n+1} = y_n + \frac{h_n}{2}\left[f(t_n, y_n) + f\left(t_n + h_n, y_n + h_n f(t_n, y_n)\right)\right].$$

Ainsi, on obtient L'algorithme de Runge-kutta d'ordre 2 :

$$y_{n+1} = y_n + \frac{h_n}{2}(k_1 + k_2), \text{ avec : } \begin{cases} k_1 = f(t_n, y_n), \\ k_2 = f(t_{n+1}, y_n + h_n k_1), \\ t_{n+1} = t_n + h_n. \end{cases}$$

La méthode de Rung-kutta la plus utilisée est d'ordre 4 (on néglige les dérivées du 4 ème ordre dans le développement de Taylor). Il s'agit de l'algorithme suivant :

$$y_{n+1} = y_n + \frac{h_n}{6}(k_1 + 2k_2 + 2k_3 + k_4), \text{ avec : } \begin{cases} t_{n+1} = t_n + h_n, \\ k_1 = f(t_n, y_n), \\ k_2 = f(t_n + \frac{h_n}{2}, y_n + \frac{h_n}{2}k_1), \\ k_3 = f(t_n + \frac{h_n}{2}, y_n + \frac{h_n}{2}k_2), \\ k_4 = f(t_{n+1}, y_n + h_n k_3). \end{cases}$$

Ainsi, la méthode de Runge-kutta d'ordre 4 consiste à évaluer 4 valeurs intermédiaires k_1, k_2 et k_3 et k_4.

Exemple 3.4.1. *1. La méthode de Runge-Kutta classique, définie par le tableau de Butcher suivant :*

0	0	0	0	0
1/2	*1/2*	0	0	0
1/2	0	*1/2*	0	0
1	0	0	1	0
	1/6	*2/6*	*2/6*	*1/6*

Ce qui donne le schéma de Runge-Kutta d'ordre 4 :

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + \frac{h_n}{2}, y_n + \frac{h_n}{2}k_1), \\ k_3 &= f(t_n + \frac{h_n}{2}, y_n + \frac{h_n}{2}k_2), \\ k_4 &= f(t_n + h_n, y_n + h_n k_3). \\ \hline y_{n+1} &= y_n + \frac{h_n}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

Exemple 3.4.2. *Soit le problème de Cauchy :*

$$\begin{cases} y'(t) = y(t) - \dfrac{2t}{y(t)} \quad t \in [0, 0.2] \\ y(0) = 1 \end{cases}$$

La solution exacte de ce problème est : $y(t) = \sqrt{2t+1}$.

A l'aide de l'algorithme de Runge-Kutta d'ordre 2 :

$$\begin{cases} h_n = 0.2 - 0 = 0.2, \\ k_1 = f(t_0, y_0) = f(0, 1) = 1, \\ k_2 = f(t_1, y_0 + h_n k_1) = f(0.2, 1.2) = 0.866667, \\ y_1 = y_0 + \dfrac{h_n}{2}(k_1 + k_2) = 1.186667. \end{cases}$$

Ainsi : $y(0.2) \simeq y_1 = 1.186667$.

La valeur exacte étant : $y(0.2) = \sqrt{1.4} = 1.183216$.

L'erreur commise est : $|y(0.2) - y_1| = |1.186667 - 1.183216| = 0.003451$.
En utilisant la méthode de Runge-Kutta d'ordre 4 :

$$\begin{cases} h_n = 0.2 - 0 = 0.2, \\ k_1 = f(t_0, y_0) = f(0, 1) = 1, \\ k_2 = (t_0 + \dfrac{h_n}{2}, y_0 + \dfrac{h_n}{2} k_1) = f(0.2, 1.2) = 0.866667, \\ k_3 = (t_0 + \dfrac{h_n}{2}, y_0 + \dfrac{h_n}{2} k_2) = f(0.1, 1.0918182) = 0.0908637, \\ k_4 = f(t_1, y_0 + h_n k_3) = f(0.2, 1.181727) = 0.843239, \\ y_1 = y_0 + \dfrac{h_n}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 1.183229. \end{cases}$$

Ainsi : $y(0.2) \simeq y_1 = 1.183229$.
L'erreur est : $|y(0.2) - y_1| = |1.183216 - 1.183229| = 0.000013$.
La solution graphique donnée par Matlab est la suivante :

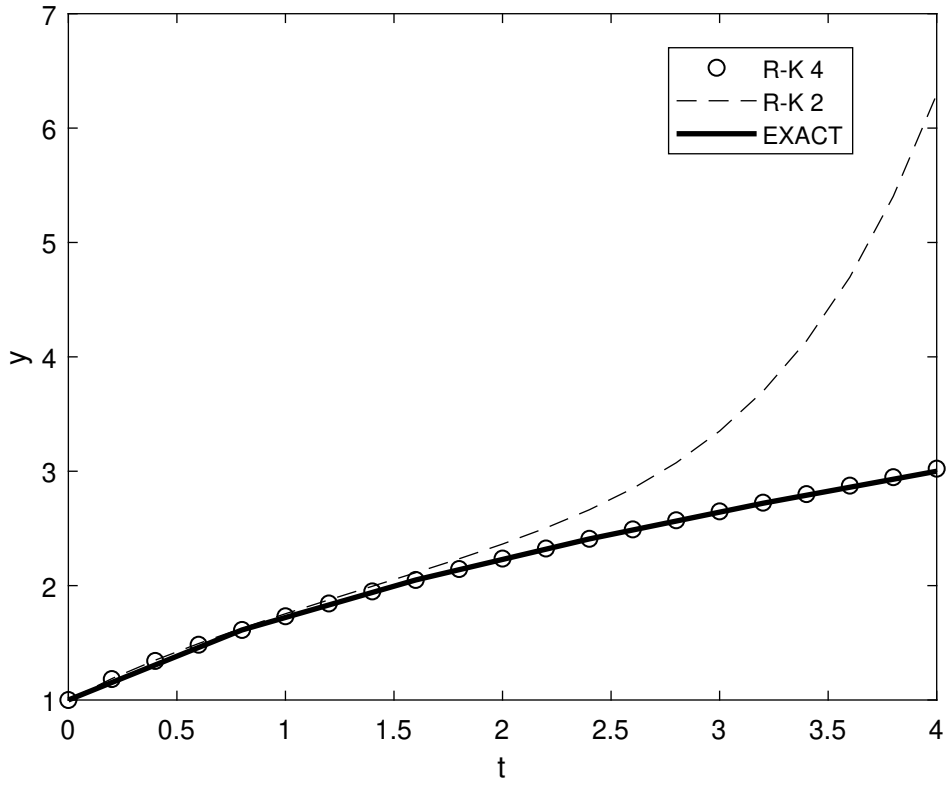


Figure 3.2: Solution graphique de l'exemple 3.4.1

3.5 Méthode numériques à pas multiples

Les méthodes d'Euler et Runge – Kutta sont connues comme des méthodes à pas, car typiquement, y_{n+1} est déterminé uniquement à partir de y_n. Dans cette section, nous considérons des méthodes à plusieurs pas dans lesquelles le calcul de la solution numérique y_{n+1} utilise les valeurs de la solution sur plusieurs nœuds précédents.
Notre but est d'approcher $f(t, y(t))$ en utilisant une interpolation polynômiale, puis intégrons cette interpolation polynômiale. De nombreuses méthodes on va considérer seules les plus populaires, les méthodes Adams – Bashforth (AB) et Adams – Moulton (AM). Ces méthodes sont la base de certains des codes informatiques les plus largement utilisés. Ils sont généralement plus efficaces que les méthodes de RK, surtout si l'on souhaite trouver la solution avec un degré élevé de précision.

3.5.1 Méthode d'Adams-Bashforth

Supposons que $h_n = t_{n+1} - t_n = cst$. Soit $y(t)$ est une solution exacte associée au problème de Cauchy. Supposons qu'on ait déja calculé les points $y(t_{n-1})$ et $y(t)$ et les pentes

$f_{n-1} = f(t_{n-1}, y(t_{n-1}))$ et $f_n = f(t_n, y(t_n))$.

La méthode d'Adams consiste à approximer la fonction $f(t, y(t))$ par son polynôme d'interpolantion aux points t_n, t_{n-1}. Soit $P(t)$ ce polynôme, $P(t)$ est la fonction définie par :

$$P(t) = f_n + \frac{f_n - f_{n-1}}{t_n - t_{n-1}}(t - t_n)$$

On écrit alors ,

$$\begin{aligned} y_{t_{n+1}} &= y(t_n) + \int_{t_n}^{t_{n+1}} f(t_n, y(t_n))dt \\ &\simeq y(t_n) + \int_{t_n}^{t_{n+1}} P(t)dt \\ &= y(t_n) + \int_{t_n}^{t_{n+1}} \left[f_n + \frac{f_n - f_{n-1}}{t_n - t_{n-1}}(t - t_n) \right] dt \\ &= y(t_n) + h_n f_n + \left(\frac{f_n - f_{n-1}}{h} \right) \left[\frac{(t - t_n)^2}{2} \right]_{t_n}^{t_{n+1}} \\ &= y(t_n) + \frac{h_n}{2}(3f_n - f_{n-1}). \end{aligned}$$

Donc l'algorithme d'Adams-Bashforth à 2 pas s'écrit :

$$\begin{cases} y_0, y_1 \text{ donnés} \\ y_{n+1} = y_n + h_n \left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1}) \right), \\ t_{n+1} = t_n + h_n, \\ f_{n+1} = f(t_n, y_n). \end{cases}$$

3.5.2 Méthode d'Adams-Moulton

Soit $y(t)$ une solution exacte. Le développement de Taylor de $y(t)$ est donné par :

$$\begin{aligned} y(t_n) &= y(t_{n+1} - h_n) \\ &= y(t_{n+1}) - h_n y'(t_{n+1}) + O(h_n^2) \\ &= y(t_{n+1}) - h_n f\left(t_{n+1}, y(t_{n+1})\right) + O(h_n^2). \end{aligned}$$

En négligeant les termes en h_n^2, l'algorithme d'Adams-Moulton d'ordre 1 s'en déduit alors :

$$\begin{cases} y_0 \text{ donnée}, \\ y_{n+1} = y_n + h_n f(t_{n+1}, y(t_{n+1})), \\ t_{n+1} = t_n + h_n. \end{cases}$$

On augmente la précision de la méthode d'Adams-moulton en négligeant les termes en h_n^3 dans le développement de Taylor. On aboutit alors à :

L'algorithme d'Adams-moulton d'ordre 2, dite aussi méthode de Crank-Nicolson :

$$\begin{cases} y_0 \text{ donnée,} \\ y_{n+1} = y_n + \dfrac{h_n}{2}(f_{n+1} + f_n), \\ t_{n+1} = t_n + h_n, \end{cases}$$

où

$$f_{n+1} = f(t_{n+1}, y_{n+1}) \text{ et } f_n = f(t_n, y_n).$$

On fait de même pour obtenir **l'algorithme d'Adams-Moulton d'ordre 3** :

$$\begin{cases} y_0, y_1 \text{ données,} \\ y_{n+1} = y_n + h_n \left(\dfrac{5}{12} f_{n+1} + \dfrac{8}{12} f_n - \dfrac{1}{12} f_{n-1} \right), \\ t_{n+1} = t_n + h_n, \end{cases}$$

où

$$f_{n+1} = f(t_{n+1}, y_{n+1}),\ f_n = f(t_n, y_n), \text{ et } f_{n-1} = f(t_{n-1}, y_{n-1}).$$

Chapitre 4

Résolution des EDO avec Matlab

4.1 Pourquoi Matlab ?

Matlab est un langage de programmation interprété, très puissant dans le calcul matriciel. Il contient une multitude de fonctions d'analyse numérique et statistique, d'optimisation, de représentation graphique... Sa richesse en toolboxes lui permet d'exécuter diverses applications dans différents domaines de la science(contrôle, traitement de signal et d'images, finance, électrotechnique, ...). Matlab réalise des calculs numériques énormes permettant d'avoir des graphiques en 2D et en 3D, de résoudre des équations différentielles linéaires et non linéaires dont on est incapable de les résoudre analytiquement, ... Les différents solveurs qu'il contient, les fonctions qui lui sont intégrées ainsi que la possibilité de lui introduire des algorithmes nous permet de faire tourner avec souplesse de gros calculs et graphiques. C'est un logiciel puissant pour explorer les problèmes des sciences et des mathématiques et pour résoudre une large quantité d'informations avec facilité et rapidité. Pour accomplir l'une de ses importantes fonctionnalités qui est la résolution des équations différentielles, Matlab est doté des solveurs ayant une forme bien particulière.

4.1.1 Solveurs

Nous pouvons distinguer deux types de solveurs :

1. **Solveurs classiques :**

   Ode45 : C'est la plus populaire et le premier choix des utilisateurs. Elle se base sur la formule de Runge-Kutta explicite d'ordre 4 et 5 simultanément, en ajustant le pas en fonction de l'erreur estimée et qui est la différence entre les deux solutions données par les deux différents ordres.

Ode23 : Elle est plus riche qu'ode45 dans certains cas, malgré qu'elle utilise à son tour la méthode de Runge Kutta explicite à un petit pas, mais d'ordre 2 et 3 simultanément pour estimer l'erreur .

Ode113 : Cet algorithme représente un solveur multi-pas explicite, qui se base sur une méthode d'Adams-Bashforth-Moulton et qui peut aller jusqu'au 13ème ordre. Ode113 est utilisée pour des problèmes qui commutent intensivement, ou lorsqu'ils exigent une tolérance rigoureuse et fournit des résultats assez cohérents.

2. **Solveurs raides :**

Ode15s : basé sur les formules implicites de différenciation numérique (NDF: Numerical Differenciation Formulas.), ou les BDF(Backward Differenciation Formulas), d'ordre élevé variant de 1 à 5. Elle est conseillée en cas de ralentissement de ode45, ou de variation rapide de la solution.

ode23t : c'est une implémentation de la règle du trapèze pour les problèmes modérément raides .

ode23tb : il s'agit d'une méthode implicite, qui implémente des formules de différenciation arriérées et la règle de trapèzes TR-BDF2(Trapezoidal rule and the backward differentiation formula of order 2). Elle est plus efficace que ode15s dans le coté précision).

ode23s : c'est la troisième version modifiée de la méthode implicite à un pas de Rosenbrock d'ordre 2 et 3. Considérée plus efficace que ode15s, cette méthode évolue de y n à y n+1 à un ordre 2 et contrôle l'erreur, en comparant les solutions ultérieures avec celles trouvées à l'ordre 3.

4.1.2 Utilisation et implémentation des solveurs sous Matlab

la syntaxe d'appel du solveur est :

$$[t, Y] = \text{odexx}(@\text{edofct}, [t0\ t1], Y0, \text{options}).$$

Avec :

Les **entrées** du solveur sont :

- **Odexx** : le type de solveur choisi pour la résolution (ode45, ode23s,...).

- **@edofct (ou 'edofct')** : la fonction Matlab contenant les équations différentielles.
- **[t0 t1]** : l'intervalle de calcul des solutions.
- **Y0** : vecteur des conditions initiales.
- **options** : les paramètres d'options qui peuvent être : NonNégative qui veille sur la positivité des solutions, Max step, initial step pour pouvoir fixer la taille maximale du pas ou le pas initial, Relative Tolérance et Absolute Tolérance pour contrôler la précision...

Les **sorties** du solveur sont :

- **t** : vecteur colonne comportant des instants de $[t0\,t1]$ où les solutions étaient calculées. L'utilisateur est libre dans le choix de nombre de points où les solutions sont affichées.
- **Y** : c'est le vecteur des solutions numériques correspondantes aux instants t.

4.2 Trouver des solutions explicites

4.2.1 Équations de premier ordre

Supposons, par exemple, que nous voulions résoudre l'équation différentielle de premier ordre :

$$y'(t) = yt. \tag{4.1}$$

Nous pouvons utiliser la fonction **dsolve ( )** de Matlab, l'entrée et la sortie pour résoudre ce problème dans Matlab est donné ci-dessous.

```
>> y = dsolve('Dy = y*t','t')
y =
C1*exp(t^2/2)
```

Notez en particulier que Matlab utilise le capital D pour indiquer la dérivé et exige que toute l'équation apparaît entre guillemets simples. Matlab prend t pour être la variable indépendante.

Pour résoudre le problème (4.1) avec la valeur initiale $y(1) = 1$, utilisons :

```
>> dsolve('Dy=t*y','t')
ans =
C1*exp(t^2/2)
>> y = dsolve('Dy=t*y','y(1)=1','t')
y =
exp(-1/2)*exp(t^2/2)
```

Maintenant nous avons résolu l'EDO, supposons que nous voulions tracer la solution pour avoir une idée approximative de son comportement. Par conséquent, nous pouvons utiliser le script :

```
>> t = linspace (0,1,20);
>> z = eval (vectorize(y));
>> plot(t,z)
```

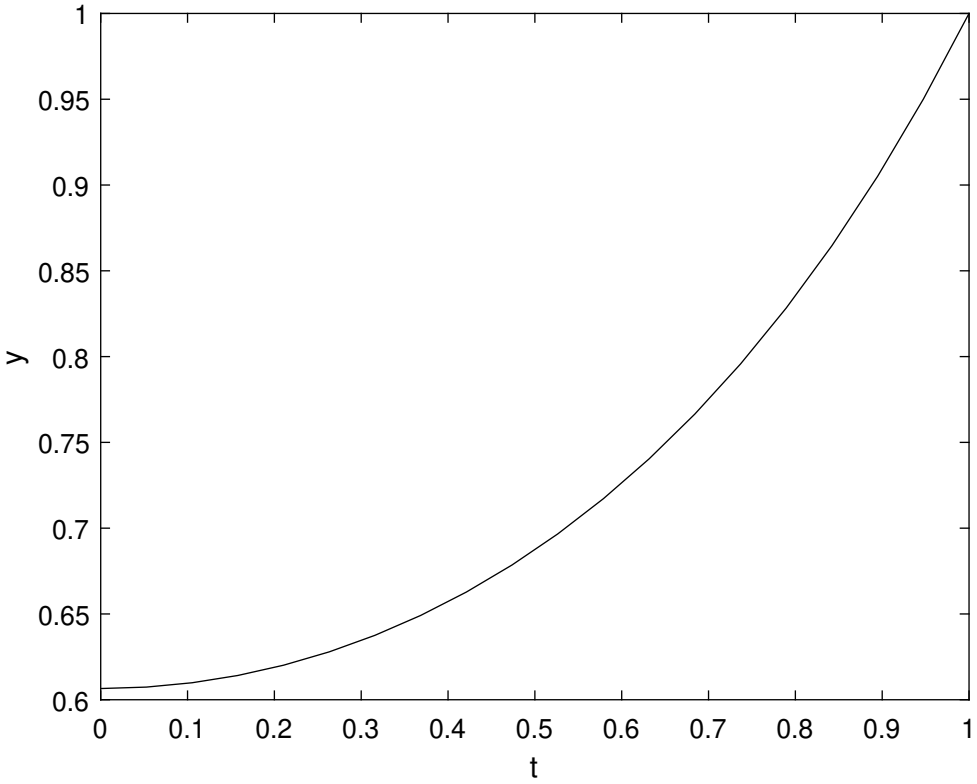


Figure 4.1: Solution de l'équation du premier ordre : $y'(t) = yt$.

4.2.2 Équations d'ordre deux et d'ordre supérieur

Supposons que nous voulions résoudre et tracer la solution de l'équation de second ordre suivante :

$$\begin{cases} y''(t) + 8y'(t) + 2y(t) = cos(t) \\ y(0) = 0, \;\; y'(0) = 1. \end{cases} \tag{4.2}$$

Le code Matlab suivant suffit :

```
>> Eqn2 = 'D2y+8*Dy+2*y=cos(t)';
>> Inits = 'y(0)=0, Dy(0)=1';
>> y = dsolve(Eqn2,Inits,'t');
>> t = linspace(0,1,20);
>> z = eval(vectorize(y));
```

```
>> plot(t,z,'b','linewidth',1.5)
>> xlabel('t')
>> ylabel('y')
```

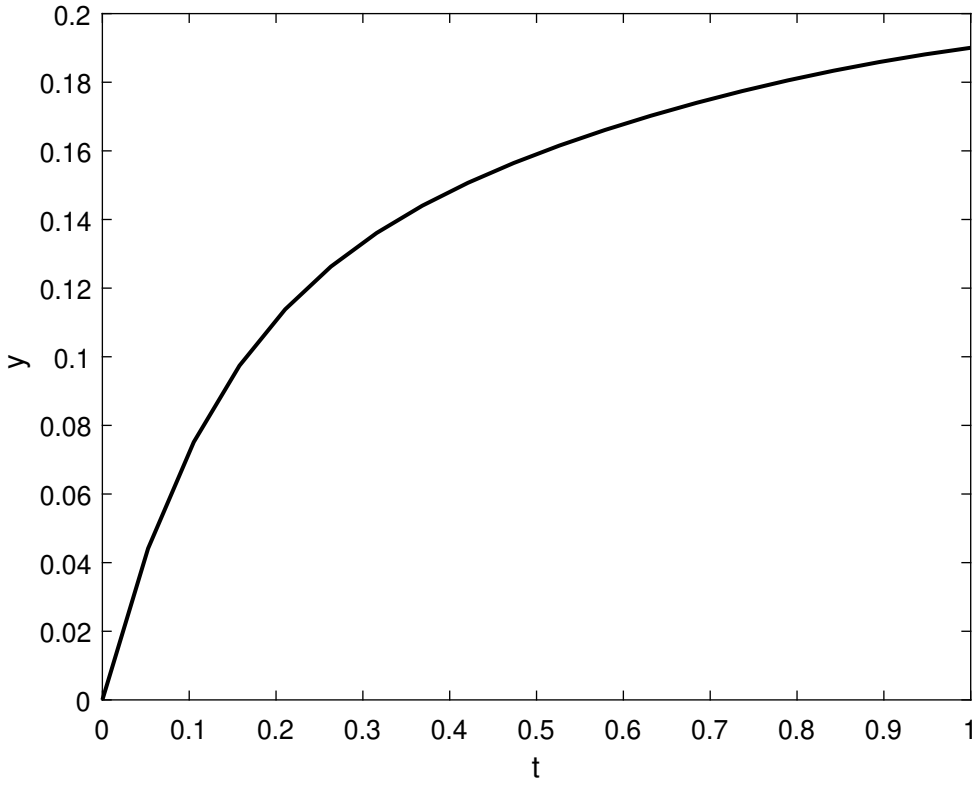


Figure 4.2: Solution de l'équation du second ordre $y''(t) + 8y'(t) + 2y(t) = cos(t)$.

Systèmes :

Supposons que nous voulions résoudre et tracer des solutions du système de trois équations différentielles ordinaires :

$$\begin{cases} \dfrac{dx}{dt} = x + 2y - z \\ \dfrac{dy}{dt} = x + z \\ \dfrac{dz}{dt} = 4x - 4y + 5z. \end{cases} \tag{4.3}$$

Supposons qu'on a comme valeurs initiales : $x(0) = 1$, $y(0) = 2$ et $z(0) = 3$. Nous avons, alors la syntexe :

```
>> Inits = 'x(0)=1, y(0)=2, z(0)=3';
>> [x,y,z] = dsolve('Dx=x+2*y-z','Dy=x+z','Dz=4*x-4*y+5*z',Inits)
x =
6*exp(2*t) - (5*exp(3*t))/2 - (5*exp(t))/2
y =
(5*exp(3*t))/2 - 3*exp(2*t) + (5*exp(t))/2
z =
10*exp(3*t) - 12*exp(2*t) + 5*exp(t)
```

Pour tracer les solutions, tappant :

```
>> t=linspace(0,0.5,25);
>> xx=eval(vectorize(x));
>> yy=eval(vectorize(y));
>> zz=eval(vectorize(z));
>> plot(t,xx,t,yy,t,zz)
```

Le résultat affiché est la figure :

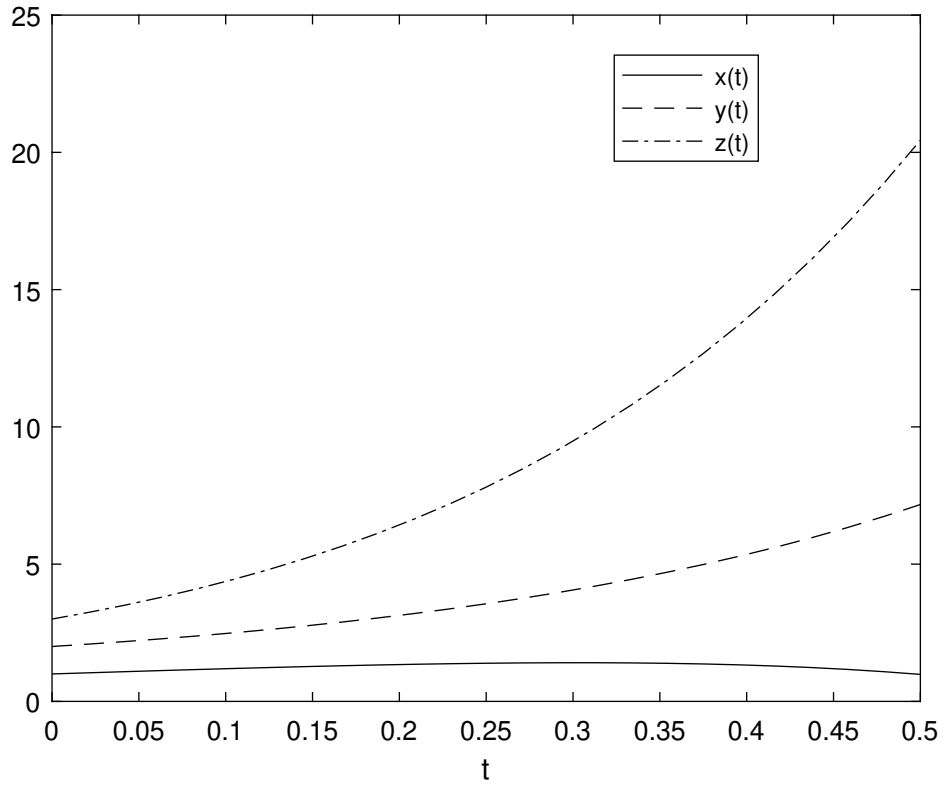


Figure 4.3: Solution graphique du système (4.3).

4.3 Trouver des solutions numériques sous Matlab

Matlab dispose d'un certain nombre d'outils permettant de résoudre numériquement les équations différentielles ordinaires. On se concentrera sur les deux principales, les fonctions intégrées ode23 et ode45, qui implémentent les versions Runge–Kutta du 2ème/ 3ème ordre et Runge–Kutta 4ème / 5ème ordre, respectivement.

4.3.1 Équations de premier ordre avec la fonction "anonyme"

Exemple 4.3.1. *Approximer numériquement la solution de l'équation différentielle de premier ordre suivante :*

$$\begin{cases} \dfrac{dy}{dt} = ty^2 + y \ ; \ \ t \in [0, 0.5]. \\ y(0) = 1. \end{cases} \tag{4.4}$$

Pour toute équation différentielle sous la forme $y' = f(t, y)$, nous commençons par définir la fonction $f(t, y)$. Pour les équations simples, nous pouvons définir $f(t, y)$ comme une fonction anonyne. Ici :

```
>> f= @(t,y) t*y^2+y
f =
function_handle with value:
@(t,y)t*y^2+y
```

L'utilisation de base du solveur ode45 est :

ode45(fonction, domaine, condition initiale).

C'est-à-dire que nous utilisons :

```
>> [t,y] = ode45(f,[0  0.5],1)
```

ou

```
>> [t,y]=ode45(f, [0,.1,.2,.3,.4,.5],1)
```

si nous pourrions seulement approcher : $y(.1), y(.2), ..., y(.5)$.

```
>> [t,y] = ode45(f,[0,.1,.2,.3,.4,.5],1)

t =

         0
    0.1000
    0.2000
    0.3000
    0.4000
    0.5000


y =
```

```
1.0000
1.1111
1.2500
1.4286
1.6667
2.0000
```

Matlab renvoie deux vecteurs colonnes, le premier avec les valeurs de t et le second avec les valeurs de y. Puisque t et y sont des vecteurs avec les composants correspondants, nous pouvons tracer les valeurs avec :

```
plot(t,y,'b','linewidth',1.5)
 xlabel('t')
 ylabel('y')
```

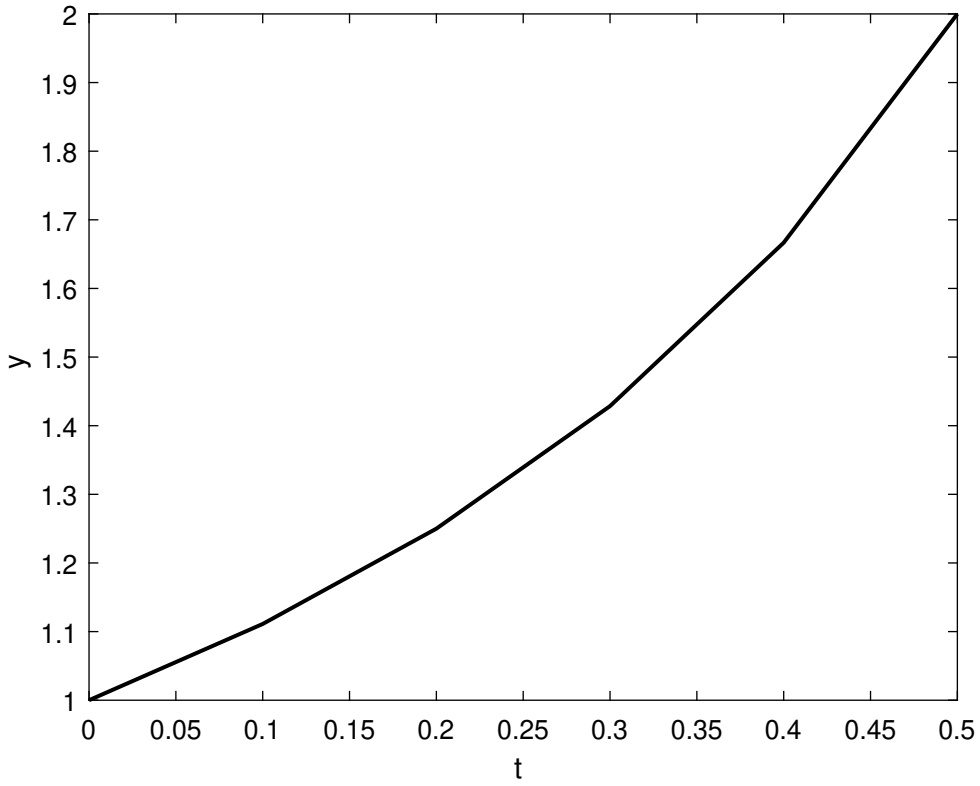


Figure 4.4: Solution graphique du problème (4.4)

Remarque 4.3.2. *Plusieurs options sont disponibles pour le résolveur ode45, offrant à l'utilisateur des limites pour contrôler l'algorithme. Deux options importantes sont la tolérance relative et absolue, respectivement RelTol et AbsTol. À chaque étape de l'algorithme ode45, une erreur est approximée. Si y_k est l'approximation de $y(t_k)$ à l'étape k, et e_k est l'erreur approximative à cette étape, Matlab choisit sa partition pour*

assurer :

$$e_k \leq max(RelTol * y_k, AbsTol),$$

où les valeurs par défaut sont $RelTol = .001$ *et* $AbsTol = .000001$.

4.3.2 Équations de premier ordre avec M-file

Alternativement, nous pouvons résoudre le même EDO précédente par définir $f(t, y)$ comme un M-file :

```
% Cette fonction définit l'équation de l'exemple 4.3.1.
function dy = firstode(t,y)
dy = t* y^2 + y ;
```

Dans ce cas, nous n'avons besoin que d'une seule modification dans la commande ode45 : nous devons utiliser un pointeur @ pour indiquer le M-file. C'est-à-dire que nous utilisons les commandes suivantes :

```
>> tspan=[0,0.5];
>> y0=1;
>> [t,y]=ode45(@firstode,tspan,y0);
```

4.3.3 Systèmes d'EDO

La résolution d'un système d'EDO dans Matlab est assez similaire à la résolution d'une seule équation, bien que puisqu'un système d'équations ne peut être défini comme une fonction inline, nous devons le définir comme un M-file.

Système de Lorenz :

On va résoudre le système d'équations de Lorenz :

$$\begin{cases} \dfrac{dx}{dt} = -\sigma x + \sigma y \\ \dfrac{dy}{dt} = \rho x - y - xz \\ \dfrac{dz}{dt} = -\beta z + xy, \end{cases}$$

où pour les besoins de cet exemple, nous allons prendre $\sigma = 10$, $\beta = 8/3$ et $\rho = 28$, ainsi que $x(0) = -8$, $y(0) = 8$ et $z(0) = 27$. Le M-file contenant les équations de Lorenz apparaît ci-dessous :

```
function dx = lorenz(t,x);
% Paramètres
sigma = 10 ;
beta = 8/3 ;
rho = 28 ;
% Équation différentielle
dx = zeros(3,1);
dx(1) = sigma * (x(2)-x(1));
dx(2) = rho * x(1)-x(2)-x(1) * x(3);
dx(3) = x(1) * x(2)-beta  x(3);
```

Observez que x est stocké sous $x(1)$, y sous $x(2)$ et z sous $x(3)$. Aditionellement, x prime est un vecteur colonne, comme le montre le point-virgule suivant la première apparition de $x(2)$. Si dans la fenêtre de commande, tapant :

```
>> x0=[-8 8 27];
>> tspan=[0,20];
>> [t,x]=ode45(@lorenz,tspan,x0)
```

La sortie de la dernière commande consiste en une colonne de temps suivie par une matrice avec trois colonnes, la première correspondant aux valeurs de x aux instants associés, et de même, les deuxième et troisième colonnes correspondent aux valeurs de y et z, respectivement. La matrice a été notée x dans l'appel standard ode45, et en général les coordonnées de la matrice peut être spécifiée sous la forme x (m, n), où m désigne la ligne et n la colonne. Ce qui nous intéressera le plus, ce sont les colonnes de x, qui correspondent aux colonnes composantes du système. Le long de ces lignes, nous pouvons désigner toutes les lignes ou toutes les colonnes par un deux points ":". Par exemple, x (:, 1) fait référence à toutes les lignes de la première colonne de la matrice x. C'est ça fait référence à toutes les valeurs de notre composant x d'origine. En utilisant cette information, nous pouvons facilement tracer l'attracteur étrange de Lorenz.

```
plot(x(:,1),x(:,3),'b','linewidth',1.5)
xlabel('x')
ylabel('y')
```

Le résultat obtenu est la figure suivante :

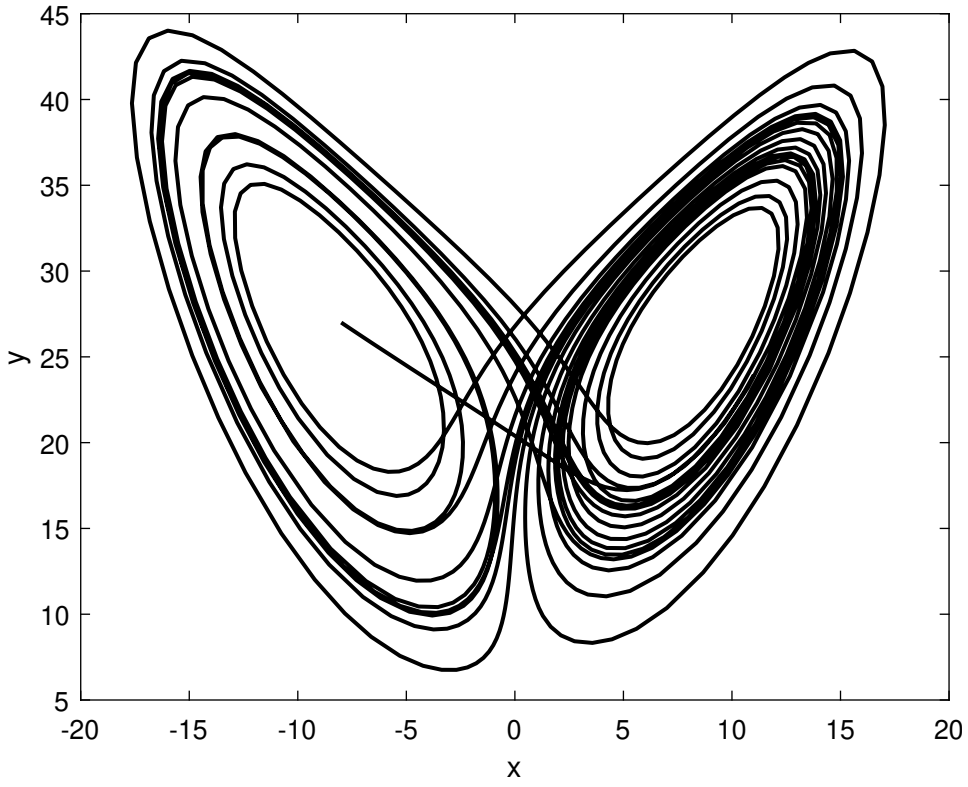


Figure 4.5: Solution graphique du système de Lorenz

4.3.4 Équations d'ordre 2

Matlab ne sait résoudre que des systèmes différentiels du premier ordre. Une étape préliminaire à la résolution par Matlab de l'équation différentielle est donc la mise de celle-ci sous forme d'un système différentiel du premier ordre. Par exemple, revenons à l'équation (4.2). En prenant $y_1(t) = y(t)$ et $y_2(t) = y'(t)$, nous avons le système :

$$\begin{cases} y_1'(t) = y_2(t) \\ y_2'(t) = -8y_2(t) - 2y_1(t) + \cos(t). \end{cases}$$

Nous pouvons maintenant procéder comme dans la section précédente.

4.4 Transformations de Laplace

Un des outils les plus utiles en mathématiques est la transformation de Laplace. Matlab a construit dans les routines de calcul des transformations de Laplace et des transformations inverses de Laplace. Par exemple, pour calculer la transformation de Laplace de $f(t) = t^2$, tapez simplement :

```
>> syms t
>> laplace(t^2)
ans =
2/s^3
```

Pour inverser, disons, $F(s) = \dfrac{1}{1+s}$, tapez :

```
>> syms s
>> ilaplace(1/(1+s))
ans =
exp(-t)
```

4.5 Problèmes de valeurs limites

La plupart des cours d'initiation aux équations différentielles ordinaires se concentrent principalement sur les problèmes de valeur initiale (IVP). Une autre classe d'EDO qui surgissent souvent dans les applications sont des problèmes de valeur limite (BVP). Considérons, par exemple, l'EDO :

$$\begin{cases} y'' - 3y' + 2y = 0 \\ y(0) = 0 \\ y(1) = 10, \end{cases}$$

où nos conditions $y(0) = 0$ et $y(1) = 10$ sont spécifiées sur l'intervalle $[0, 1]$. La première étape de la résolution de ce type d'équation consiste à l'écrire en tant que un système du premier ordre avec : $y_1 = y$ et $y_2 = y'$, pour lesquels on a :

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= -2y_1 + 3y_2. \end{aligned}$$

Nous enregistrons ce système dans le M-file bvp.m :

```
% Cette fonction définit les conditions d'un système d'EDO de premier ordre
function dy = bvp(t,y)
dy = zeros(2,1) ;
dy(1) = y(2) ;
dy(2) = -2 * y(1) + 3 * y(2) ;
```

Ensuite, nous spécifions les conditions aux limites dans une autre fonction M-file, qui enregistre les limites résidus.

```
% Cette fonction définit les conditions aux limites / résidus.
function res = bc(y0,y1)
res = [y0(1),y1(1)-10];
```

Par résidu, nous entendons le côté gauche de la condition limite une fois qu'elle a été définie sur 0. Dans ce cas, la seconde condition aux limites est $y(1) = 10$, son résidu est donc $y(1) - 10$, ce qui est enregistré dans la deuxième composante du vecteur renvoyé par

bc.m. Les variables $y0$ et $y1$ représentent la solution à $t = 0$ et à $t = 1$ respectivement, tandis que le 1 entre parenthèses indique le premier composant du vecteur. Dans le cas où la deuxième condition aux limites était $y'(1) = 10$, nous remplacerions $y1(1) - 10$ par $y1(2) - 10$.
Maintenant on commence à résoudre notre problème. Dans le code ce qui suit, nous spécifions d'abord une grille de valeurs de t à résoudre par Matlab et une hypothèse initiale pour le vecteur qui serait donné pour le problème de valeur initiale $[y(0), y'(0)]$. (Bien sûr, $y(0)$ est connu, mais $y'(0)$ doit être une hypothèse. Nous résolvons le problème de valeur limite avec le solveur intégré bvp4c de Matlab.
La syntaxe est la suivante :

```
>> sol=bvpinit(linspace(0,1,25),[0  1]);
>> sol=bvp4c(@bvp,@bc,sol);
>> sol.x
>> sol.y
```

Notez que dans ce cas, Matlab renvoie la solution sous la forme d'une structure dont le premier sol.x contient simplement les valeurs de x spécifiées. Le deuxième composant de la structure sol est sol.y qui est une matrice contenant comme première ligne les valeurs de $y(x)$ aux x points de la grille nous avons spécifié, et comme deuxième rangée, les valeurs correspondantes de $y'(x)$.

4.6 Méthodes numériques

4.6.1 Méthode d'Euler

Rappelons tout d'abord par le shéma d'Euler

$$y_{n+1} = y_n + h_n f(t_n, y_n), \quad n = 0, 1, ..., N-1$$

Considérons le problème de Cauchy :

$$\begin{cases} y' = \sin(ty) \\ y(0) = \pi \end{cases} \tag{4.5}$$

On veut résoudre ce problème, à l'aide de la méthode d'Euler, en subdivisant l'intervalle [0,1] en dix parties égales.
Nous allons effectuer les premières itérations en détail, puis nous écrirons un M-file sous Matlab pour le réaliser dans son intégralité.
La valeur initiale $y(0) = \pi$ nous donne les valeurs $t_0 = 0$ et $y_0 = \pi$. Et on a par suite :

$$y_1 = y_0 + h_n \sin(t_0 y_0) = \pi + 0.1 \sin(0) = \pi.$$

$$y_2 = y_1 + h_n \sin(t_1 y_1) = \pi + 0.1 \sin(0.1\pi) = 3.1725.$$

$$y_3 = y_2 + h_n \sin(t_2 y_2) = 3.1725 + 0.1 \sin(0.2(3.1725)) = 3.2318.$$

Plus généralement, nous pouvons utiliser le M-file suivant euler.m.(ici $h_n = dt$)

```
function [tvaleurs,yvaleurs] = euler(f,t0,tn,y0,n)
% Taille de pas
dt = (tn-t0)/n ;
% Conditions initiales
t(1) = t0 ;
y(1) = y0 ;
% Méthode d'Euler
for k = 1:n
t(k+1) = t(k) + dt ;
y(k+1) = y(k) + f(t(k),y(k)) * dt ;
end
tvaleurs = t' ;
yvaleurs = y' ;
```

Nous pouvons implémenter ce fichier avec le code suivant :

```
>> f=inline('sin(t*y)')
 [t,y] = euler(f,0,1,pi,10)
 plot(t,y,'b','linewidth',1.5)
 xlabel('t')
 ylabel('y')

f =

     Inline function:
     f(t,y) = sin(t*y)


t =

         0
    0.1000
    0.2000
    0.3000
    0.4000
    0.5000
    0.6000
    0.7000
    0.8000
    0.9000
    1.0000


y =

    3.1416
    3.1416
    3.1725
    3.2318
    3.3142
    3.4112
    3.5103
    3.5963
    3.6548
    3.6764
    3.6598
```

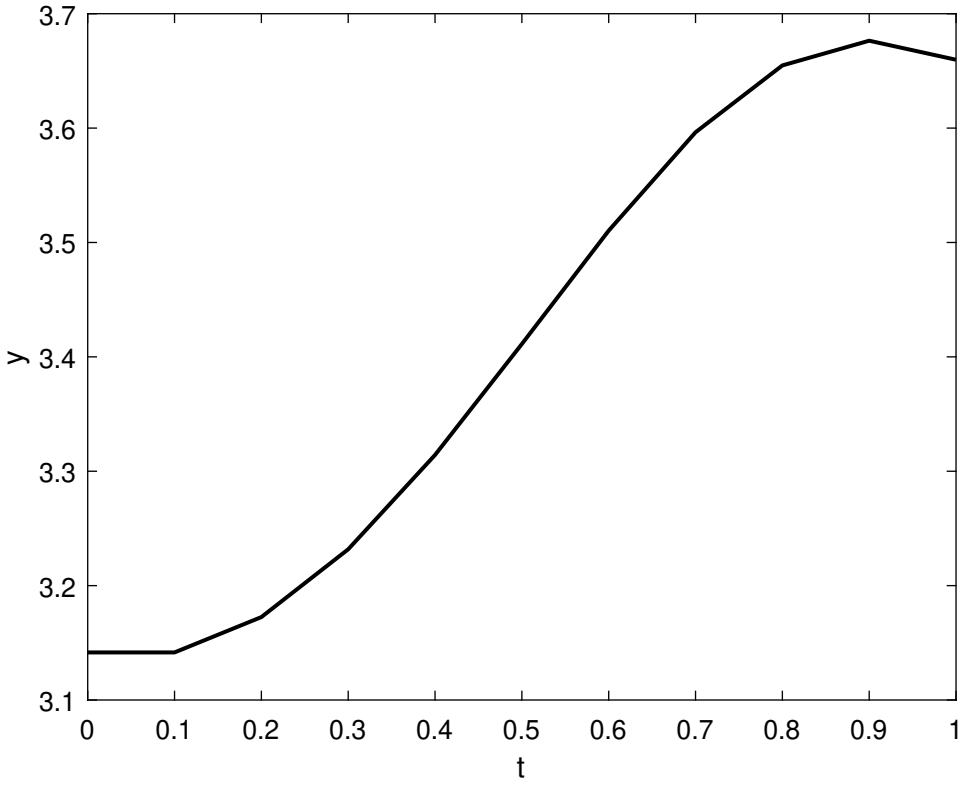
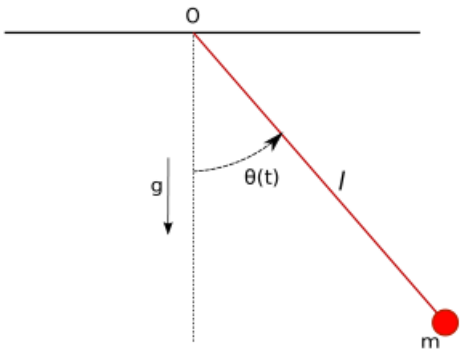


Figure 4.6: Solution graphique du problème (4.5) avec la méthode d'Euler.

4.7 Pendule simple



Considérons un pendule de masse m, de longueur l, oscillant sous l'accélération de la gravité g, se déplace selon la deuxième loi du mouvement de Newton :

$$ml(\theta)'' = -mg\sin(\theta),$$

où l'angle θ dépend du temps t. Si nous réorganiser les termes nous obtenons une équation scalaire de second ordre :

$$\theta'' + \frac{g}{l}\sin(\theta) = 0.$$

Le moyen facile pour résoudre cette équation est de limiter la solution aux cas où l'angle θ est petite. Dans ce cas, nous pouvons faire l'approximation linéaire $\sin\theta \approx \theta$.
D'où :

$$\theta'' + \frac{g}{l}\theta = 0,$$

est une EDO d'ordre 2 homogène à coefficients constants dont la solution est :

$$\theta(t) = \theta_0 \cos(\sqrt{\frac{g}{l}} + \phi).$$

Ici θ_0 et ϕ sont des constantes arbitraires qui dépendent des conditions initiales. L'angle θ_0 est appelé l'amplitude du mouvement et correspond au déplacement maximal du pendule par rapport au verticale. La constante ϕ s'appelle la constante de phase, elle nous permet de connaître la position de pendule, lorsque t égal 0.

L'équation différentielle de second ordre, peut s'écrire comme un système d'équations de premier ordre. Si on introduit $y_1 = \theta$ et $y_2 = \theta'$, alors :

$$\begin{cases} y_1' = y_2 \\ y_2' = -\dfrac{g}{l}\sin(y_1). \end{cases}$$

En utilisant les valeurs des paramètres : $m = 0.05\,kg$, $l = 0.5\,m$ et $g = 9.8\,m/s^2$.
Sous Matlab, la description du système doit se faire dans un m-file décrivant la fonction qui fournit les n dérivées sous la forme d'un vecteur colonne. Dans notre cas, cela donne

```
% Paramètres

l=0.5,g=9.8;

% EDO de pendule

f= @(t,y)[y(2);-g/l*sin(y(1))];

% Conditions initiales :

init=[pi/2;0];

% Résoudre l'EDO sur l'intervalle 0 à 10 seconds :

[t,y]=ode45(f,[0,10],init)

% ploting
plot(t,y(:,1));
grid on;hold on;
```

```
plot(t,y(:,2),'linewidth',2);
xlabel('Temps(s)');
ylabel('Amplitude')
legend('angle(rad)','vitesse angulaire(rad/s)')
title('mouvement du pendule simple entre 0 et 10 seconds')
```

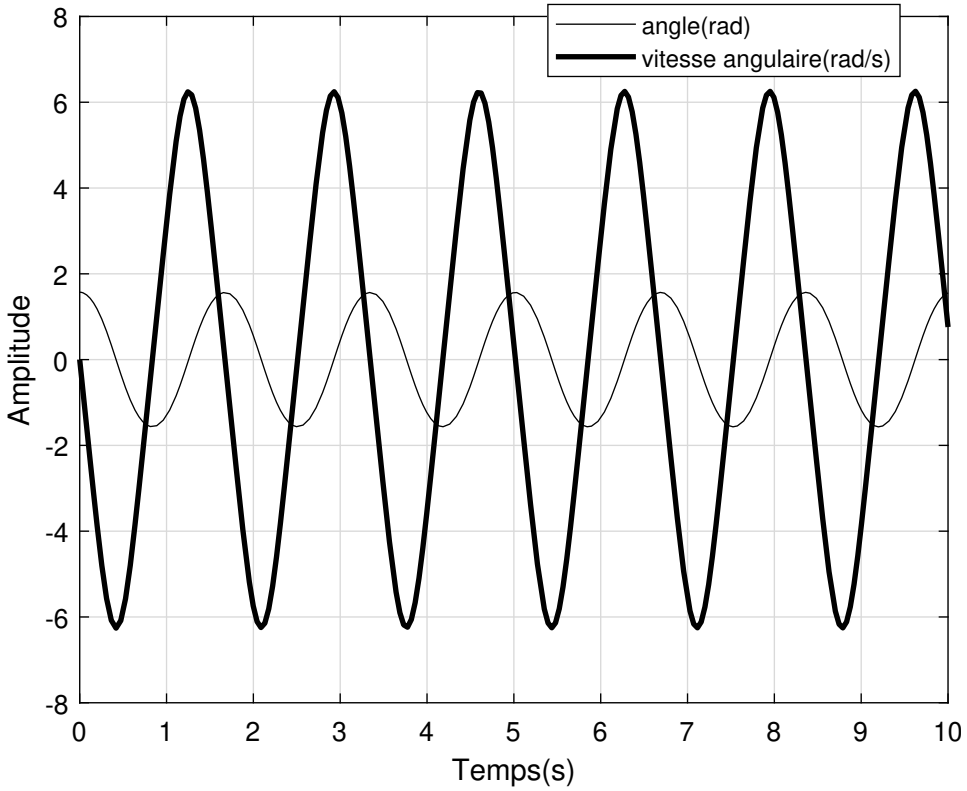


Figure 4.7: mouvement du pendule simple entre 0 et 10 seconds

Décrivons ici le code de la solution numérique des équations de mouvement pour un pendule simple en utilisant la méthode d'Euler. Il est important cela, nous notons que ces oscillations grandir avec le temps. Le code Matlab satisfait est le suivant :

```
% Paramètres

l= 0.5;          %longueur du pendule en mètres
g =9.8;          %l'accélération de la gravité

% Méthode d'Euler

n = 500;         %nombre de pas
t0=0;            %temps initial
tf=10;           %temps final
dt=(tf-t0)/n;    %temps entre pas
y2= zeros ( n , 1 ) ;   %initialise y2 , un vecteur de dimension n X 1, à zéro
```

```
y1= zeros ( n, 1 ) ;     % initialise y1 , un vecteur de dimension n X 1, à zéro
t = zeros ( n, 1 ) ;     % ceci initialise le temps vecteur à être tout zéros
y1(1) =0.2;      % avoir un déplacement initial, sinon le pendule ne sera pas basculé
for k= 1: n-1      % boucle sur les pas de temps
y2(k+1) = y2(k)-(g / l ) * sin(y1(k)) * dt ;
y1(k+1) = y1(k)+y2(k) * dt
t (k+1) = t (k) + dt ;
end

% Tracer la solution numérique en rouge

plot ( t , y1 ,'r');
xlabel ('Temps( s )');
ylabel ('Angle ( rad )');
```

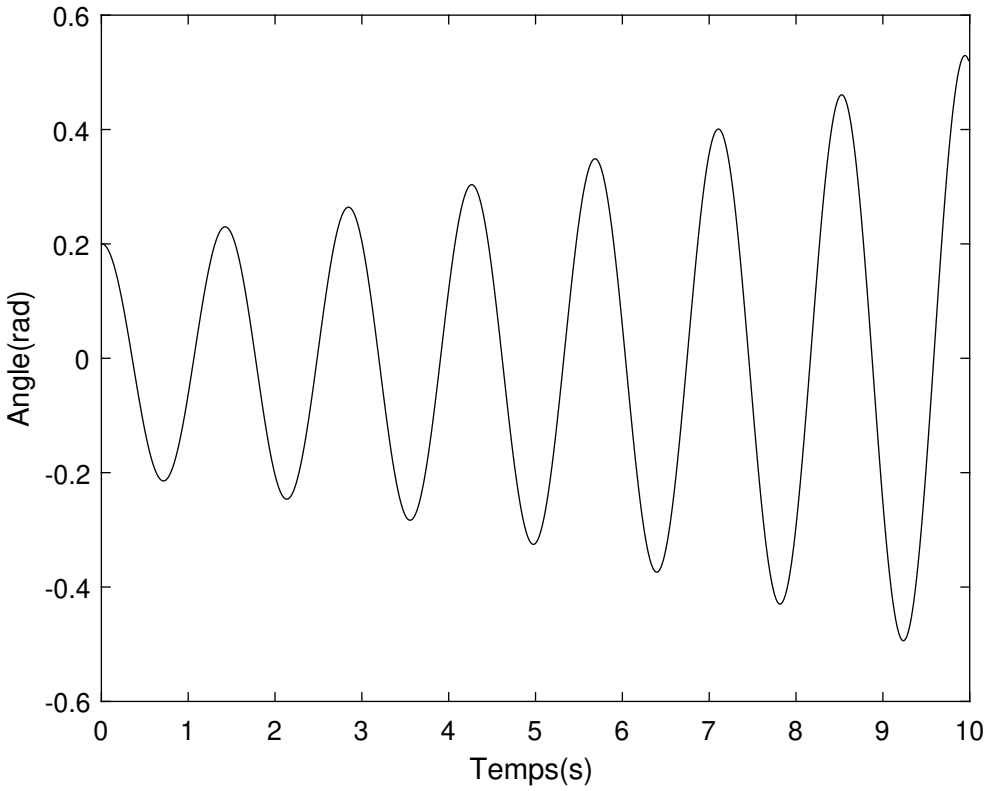


Figure 4.8: Résolution de l'EDO du pendule simple avec la méthode d'Euler

Nous pouvons faire une camparaison entre le cas linéaire (l'angle θ est petite, ie : $\sin\theta \approx \theta$) et le cas non linéaire (l'angle θ est assez grande). Pour cela tappant le script :

```
% Paramètres

l=0.5,g=9.8;
```

```
% EDO de pendule

f= @(t,y)[y(2);-g/l*sin(y(1))];
g= @(t,y)[y(2);(-g/l)*(y(1))];

% Conditions initiales :

init=[pi/2;0];

% Résoudre l'EDO sur l'intervalle 0 à 10 seconds :

[t,y]=ode45(f,[0,10],init)

% ploting
plot(t,y(:,1),'- -');
grid on;hold on;
[t,y]=ode45(g,[0,10],init)
plot(t,y(:,1),'r');
xlabel('Temps(s)');
ylabel('Angle (rad)')
```

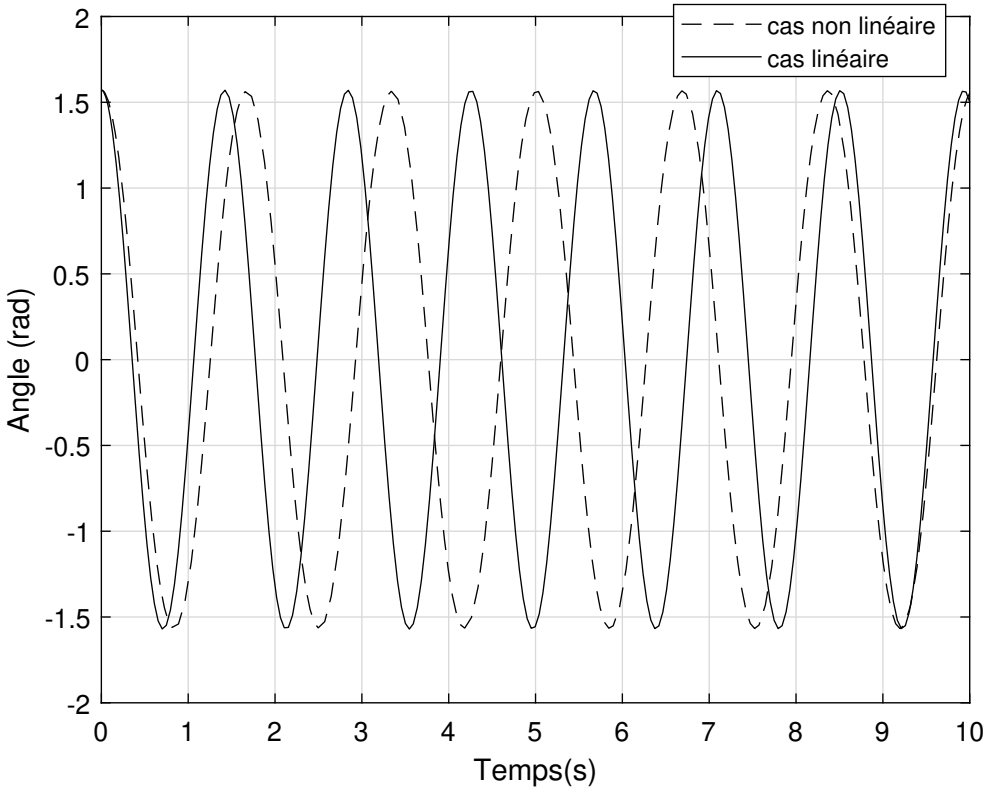


Figure 4.9: Comparaison entre le cas linéaire et non linéaire

Localisation d'un événement :

En règle générale, les solveurs d'EDO dans Matlab se terminent après la résolution de l'EDO sur un domaine spécifié. domaine de la variable indépendante (que nous avons appelée ci-dessus tspan).
Dans les applications, cependant, nous aimerions souvent arrêter la solution à une valeur particulière de la variable dépendante (par exemple, lorsqu'un objet lancé depuis le sol atteint son maximum taille ou lorsqu'une population dépasse une certaine valeur). Dans notre exemple,
supposons que nous voudrait déterminer la période de pendule. Puisque nous ne connaissons pas l'intervalle de temps (en fait, c'est ce que nous essayons de déterminer), nous aimerions préciser que Matlab résoudre l'équation jusqu'à ce que le pendule oscille à travers une fraction spécifiée de son cycle complet et de donner le temps que cela a pris. Dans notre cas, nous allons enregistrer le temps il faut le pendule pour atteindre le bas de son arc, et le multiplier par 4 pour arriver à la période du pendule. (De cette manière, l'événement est indépendant de la position initiale du pendule.)
Le système est défini dans une fonction M-file avec $l = 0.5m$.

```
function dy = pendode(t,y)
% Paramètres:
g=9.81;
l=0.5;
% Équations differentielles :
dy=zeros(2,1)
dy(1)=y(2);
dy(2)=-(g/l)*sin(y(1));
```

En plus de cette fonction M-file, nous écrivons une autre d'événements M-file $pendevent.m$ qui spécifie l'événement que nous recherchons.

```
% Cette fonction définit l'événement ou notre pendule atteint son point central.
function [lookfor stop direction] = pendevent(t,y)
% Recherche pour cette expression définie sur 0.
lookfor = y(1);
% Arrêt lorsque l'événement est localisé.
stop = 1;
% Spécifie la direction du mouvement lors de l'événement.
direction = -1;
```

Dans pendevent.m, la ligne lookfor = y (1) spécifie que Matlab doit rechercher l'événement $y(1) = 0$ (c'est-à-dire, $\theta(t) = 0$). Si nous voulions rechercher l'événement $\theta(t) = 1$, on utilise lookfor = y (1) -1. La ligne stop = 1 indique à Matlab d'arrêter la résolution lorsque l'événement est localisé, et la commande direction = -1 indique à Matlab de n'accepter que les événements pour $y(2)$ (c'est-à-dire, θ') est négatif (si le pendule commence à droite de son centre, il sera se déplacer dans le sens négatif la première fois qu'il atteint le point central).

Nous pouvons maintenant résoudre l'EDO jusqu'au moment où notre pendule atteint le point central avec les commandes suivantes dans la fenêtre de commande:

```
>> options = odeset('Events',@pendevent);
>> y0=[pi/4 0];
>> [t, y, te, ye, ie] = ode45(@pendode,[0,10],y0,options);
>> te
te =
0.5215
>> ye
ye =
-0.0000 -2.3981
```

Ici, $y0$ est un vecteur de données initiales, pour lequel nous avons choisi que le pendule commence par l'angle $\pi/4$ et sans vitesse initiale. La commande $ode45$ renvoie un vecteur de temps t, une matrice des variables dépendantes y, l'heure à laquelle l'événement s'est produit, te, et les valeurs de y quand l'événement s'est produit, ye. Dans le cas où un vecteur d'événements est spécifié, un vecteur d'index ie décrit quel événement s'est produit dans chaque instance. Ici, un seul événement a été spécifié c'est-à-dire $ie = 1$. Dans ce cas, nous voyons que l'événement s'est produit à l'instant $t = 0.5215$, et par conséquent, la période est $P = 2,086$ (avec des erreurs numériques). Bien que la période exacte du pendule soit difficile à analyser numériquement, il n'est pas difficile de la montrer à travers la petite approximation angulaire $\sin\theta \approx \theta$ que pour petite θ la période du pendule est approximativement $P = 2\pi\sqrt{\frac{l}{g}}$, ce qui donne dans notre cas $P = 2.001$. (Alors que l'approximation aux petites angles donne une période indépendante de θ, la période d'un pendule dépend de θ.)
Afin de mieux comprendre cet indice $'ie'$, vérifions que le temps est le même pour chaque quart de balançoire. C'est-à-dire, enregistrons les temps auxquelles $\theta = 0$ et aussi les temps auxquelles $\theta' = 0$ et regardons les temps entre eux. Dans ce cas, $pendevent.m$ est remplacé par $pendevent1.m$.

```
% Cette fonction définie l'évenement quand notre pendule retourne à sa
% position originale pi/4.
function [lookfor stop direction] = pendevent1(t,y)
% Recherches pour cette expression définie sur 0.
lookfor=[y(1); y(2)];
% N'arrête pas lorsque l'événement est localisé.
stop=[0; 0];
% N'importe quelle direction acceptée.
direction=[0; 0]
```

Dans ce cas, nous recherchons deux événements différents, et donc les variables dans $pendevent1.m$ sont des vecteurs avec deux composantes, chacune correspondant à un événement. Dans ce cas, nous ne arrêtons après chaque événement, et nous ne spécifions pas de direction. Dans la fenêtre de commande, nous avons le suivant :

```
>> options=odeset('Events',@pendevent);
>> y0=[pi/4 0];
```

```
>> [t, y, te, ye, ie] = ode45(@pendode,[0,10],y0,options);
>> te
te =
    0.5215
>> ye
ye =
   -0.0000   -2.3981
>> options=odeset('Events',@pendevent1);
>> y0=[pi/4  0];
>> [t, y, te, ye, ie] = ode45(@pendode,[0  2],y0,options);
>> te
te =
    0.0000
    0.5216
    1.0431
    1.5646
>> ye
ye =
    0.7854   -0.0000
   -0.0000   -2.3972
   -0.7853    0.0000
    0.0000    2.3970
>> ie
ie =
     2
     1
     2
     1
```

On voit que sur un intervalle [0,2], les temps d'événement sont approximativement 0, 0.5216, 1.0431, et 1.5646. En regardant la matrice ye, pour laquelle la première valeur dans chaque ligne est une position angulaire et la seconde est une vitesse angulaire, on voit que le premier événement correspond à la position de départ, le deuxième événement correspond à la pendaison droite du pendule, le troisième événement correspond au pendule qui a basculé complètement vers le côté opposé, et le quatrième événement correspond avec le pendule suspendu tout droit sur son voyage de retour. Il est maintenant clair comment ie fonctionne. Il égale à 2 lorsque le deuxième événement spécifié se produit et 1 lorsque le premier événement spécifié se produit.

Conclusion

La question principale de ce mémoire, était comment résoudre une EDO ?

La plupart des équations différentielles ordinaires apparaissant dans des applications réalistes telles que les mathématiques appliquées, la physique, la métrologie et l'ingénierie. Toutes ces disciplines concernent les propriétés des équations différentielles de différents types.

Dans une première démarche, nous avons présenté quelques techniques de résoudre analytiquement des EDO. Une EDO ne peut pas toujours être résolu exactement. Pour ces problèmes, on fait une analyse qualitative pour avoir une idée approximative du comportement de la solution.

Ensuite, une procédure numérique a été utilisée, telle que les méthodes d'Euler, de Runge Kutta.., il faut toujours garder à l'esprit la question de savoir si les résultats sont suffisamment précis pour être utiles. La précision des résultats numériques peut être vérifiée directement par comparaison avec la solution obtenue de manière analytique.

En raison des gros efforts nécessaires pour résoudre les EDO, certaines méthodes avec Matlab ont été étudiées, cet géant logiciel qui est riche en toolboxes (solveurs) permet à la fois de résoudre des équations différentielles linéaires et non linéaires dont on est incapable de les résoudre analytiquement, de réaliser des calculs numériques énormes avec grande précision, permettant d'avoir des graphiques en 2D et en 3D, et alors faire directement une comparaison avec la solution obtenue de manière analytique.

Perspectives :

En guise de perspectives d'avenir, on souhaite aborder d'autre type d'équations différentielles à titre d'exemple, les équations différentielles fractionnaires à retard fini ou infini ainsi que les équations semi-linéaires d'ordre entier ou fractionnaire. On souhaite aussi d'utiliser Matlab pour simuler les processus stochastiques, et comparer sa efficacité avec d'autres logiciels.

Bibliographie :

[1] L. Halpern, *Équations différentielles Ètude mathématique et numérique*. Institut GALILEE, 99 avenue Jean-Baptiste-Clément 93430 VILLETANEUSE, janvier 2016.

[2] Demailly, Jean-Pierre, *Analyse numérique et équations différentielles*, EDP sciences17, avenue du Hoggar, 2006.

[3] L. Pujo-Menjouet, *Équations Différentielles Ordinaires et Partielles*. Université Claude Bernard, Lyon I 43, boulevard 11 novembre 1918 69622 Villeurbanne cedex, France, 05-Dec-2016.

[4] A. Lesfari, *Équations différentielles*. Département de Mathématiques Faculté des Sciences Université Chouaïb Doukkali B.P. 20, El-Jadida, Maroc. SMA5, 2012-2013.

[5] Abdalftah Elbori, Ltfei Abdalsmd, *Simulation of Simple Pendulum*. International Journal of Engineering Science Invention ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726 www.ijesi.org ||Volume 6 Issue 4|| April 2017 || PP. 33-38.

[6] GABRIEL NAGY, *ORDINARY DIFFERENTIAL EQUATIONS*. Mathematics Department, Michigan State University, East Lansing, MI, 48824. NOVEMBER 29, 2017.

[7] Aude Rondepierre et Adeline Rouchon, *Notes du cours de Mathématiques de l'UF Outils mathématiques pour l'ingénieur*. INSA Département STPI Année 2016/2017.

[8] Gloria Faccanoni, *Équations différentielles ordinaires Recueil d'exercices corrigés et aide-mémoire*. IMATH Bâtiment Université de Toulon Avenue de l'université 83957 LA GARDE - FRANCE.

[9] Glenn Lahodny Jr, *Solving Ordinary Differential Equations in MATLAB*. Spring 2015.

[10] BELLOUFI MOHAMMED, *Cours d'Analyse Numérique 2*. Université Mohamed Chérif Messaadia de Souk-Ahras Faculté des Sciences et Technologie Département des Mathématiques et Informatique, Avril 2015.

[11] D.G. Simpson, Ph.D. *The Nonlinear Pendulum*. Department of PhysicalSciences and Engineering, Prince George's Community College. December 31, 2010.

[12] Parasuram Harihara And Dara W. Childs, *Solving Problems in Dynamics and Vibrations Using MATLAB*. Dept of Mechanical Engineering Texas A & M University College Station.

[13] Kevin Berwick, *Computational Physics using MATLAB* ®. West Lafayette, Indiana, USA, September 2012.

[14] Ravi P. Agarwal, Donal O'Regan, *An Introduction to Ordinary Differential Equations*. Springer Science & Business Media. 2008.

[15] Kendall Atkinson, Weimin Han, David Stewart, *NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS*. University of Iowa Iowa City, Iowa. A JOHN WILEY SONS, INC., PUBLICATION. 2009.