M1 MIGS

UFR Bourgogne Franche-Comté

Parallélisation en temps pour la résolution numérique d'équations différentielles

Killian Vuillemot

Enseignant référent : Franz Chouly

Année 2019-2020

Remerciements

Je tiens à remercier M. Chouly pour l'aide apportée au long de la réalisation de ce projet. Je le remercie pour ses conseils, son encadrement et son attention.

Je remercie également M. Dupuis pour son encadrement au long de ce projet.

Table des matières

Introduction

La notion de résolution numérique d'équations différentielles a été introduite notamment par Euler à partir de 1768 dans le but de déterminer des solutions approchées d'équations différentielles difficiles à résoudre. De nombreuses méthodes ont alors été développées afin d'obtenir des solutions approchées de plus en plus précises.

Dès la naissance de l'informatique ces méthodes numériques ont été adaptées pour être utilisées informatiquement. On a alors pu gagner en rapidité. Cependant, au fil des années, le souhait a été de gagner également en précision sur les approximations. Il a alors fallu développer de nouvelles méthodes permettant d'obtenir une précision assez élevée en un temps raisonnable. De plus, certaines équations nécessitant une résolution sur des temps longs (trajectoires de planètes ou de satellites par exemple), il a fallu développer des méthodes permettant des résolutions à la fois précises et aussi rapides que possible. Dans ce but, des méthodes itératives de résolution ont été introduites : les méthodes parallèles en temps.

Nous allons donc au fil de ce rapport introduire la notion de résolution des équations différentielles avec des méthodes parallèles en temps. Pour cela, nous commencerons par une partie théorique axée essentiellement sur les équations différentielles ordinaires linéaires.

Une fois cette partie théorique des équations différentielles terminée, nous nous intéresserons à différentes méthodes numériques très connues. Ces méthodes nous seront très utiles pour la dernière partie qui sera consacrée à la découverte d'un algorithme introduit en 2001 par Lions, Maday et Turicini : l'algorithme Pararéel.

I Introduction aux équations différentielles

Débutons donc avec un peu de théorie. Nous allons ici introduire quelques notions et résultats afin notamment de garantir l'existence de solutions. Nous verrons alors la forme générale de ces solutions. Ce qui nous permettra ensuite d'introduire les méthodes numériques.

I. 1. EDO linéaires

I. 1. 1) Équations linéaires d'ordre 1

Définition I.1. Soient $I \subset \mathbb{R}$ et $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$.
Soient a, b et c trois fonctions de I dans $\mathbb{K}$, continues sur I. Soit :

$$a(t)\dot{y}(t) + b(t)y(t) = c(t). \tag{I.1}$$

L'équation I.1 est appelée équation différentielle linéaire d'ordre 1 et on la notera souvent sous la forme suivante :

$$a\dot{y} + by = c.$$

On appelle également équation homogène associée à I.1 (ou équation sans second membre), l'équation :

$$a(t)\dot{y}(t) + b(t)y(t) = 0. \tag{I.2}$$

Exemple :

Soient $I \subset \mathbb{R}$ et $t \in I$. Soit :

$$\dot{y}(t) = -\cos(t)y(t). \tag{I.3}$$

L'équation I.3 est une équation linéaire d'ordre 1.
Une représentation graphique sur $[0, 20]$ de la solution de cette équation, de condition initiale $y(0) = 1$, est disponible en annexe IV page 44.

Il est possible d'étendre cette notion à $\mathbb{K} = \mathbb{R}^n$ ou $\mathbb{C}^n$, avec $n > 1$. On parle alors de systèmes différentiels linéaires d'ordre 1. Dans cette section, nous allons nous restreindre aux équations et aux systèmes de la forme :

$$\dot{y} = ay + b \quad , \ a \in \mathbb{K} \ , \ b \in \mathbb{K}. \tag{I.4}$$

$$\dot{y} = Ay + b \quad , \ A \in \mathcal{M}_n(\mathbb{K}) \ , \ b \in \mathbb{K}^n. \tag{I.5}$$

Dans un premier temps, nous allons nous intéresser aux équations homogènes (sans second membre). Cela permettra d'introduire certaines notions qui seront complétées ensuite dans un cadre plus général.

Définition I.2. (*Problème de Cauchy*)
Soient $I \subset \mathbb{R}$, $\Omega \subset \mathbb{R}^n$ et $f : I \times \Omega \longrightarrow \mathbb{R}^n$ une application continue.
Soient $y : I \longrightarrow \Omega$ différentiable et $(t_0, y_0) \in I \times \Omega$. On appelle alors problème de Cauchy de condition initiale y_0 le système suivant :

$$\begin{cases} \dot{y}(t) = f(t, y(t)) \\ y(t_0) = y_0. \end{cases} \tag{P}$$

Nous allons maintenant énoncer un théorème important dans la résolution d'équations différentielles. Ce théorème ne sera pas entièrement démontré, on en donnera seulement une esquisse. En effet, la démonstration est longue et a été faite en licence.

Théorème I.1. (*Cauchy-Lipschitz*)
Soit (P) un problème de Cauchy avec f une fonction localement Lipschitzienne en la variable y.
Alors il existe une unique solution maximale au problème (P).

Démonstration. Soit (ε, r_0, k) tel que f soit localement k-Lipschitzienne.

Existence : on définit une suite de fonctions $y_n(t)$ par récurrence :

$$\begin{cases} y_0(t) = y_0 \\ y_{n+1}(t) = y_0 + \displaystyle\int_{t_0}^{t} f(s, y_n(s))\mathrm{d}s. \end{cases}$$

On montre alors que $\|y_n(t) - y_0\| \leqslant r_0$ pour tous $n \in \mathbb{N}$ et $t \in [t_0 - \varepsilon, t_0 + \varepsilon]$. On montre ensuite que les fonctions $y_n(t)$ convergent uniformément vers une fonction continue et que la suite des fonctions est uniformément de Cauchy et convergente. On passe alors à la limite pour y_{n+1} en faisant tendre n vers l'infini. Cette limite est une solution du problème de Cauchy.

Unicité : on prend deux solutions x et y du problème de Cauchy. On définit le maximum Q sur $[t_0 - \varepsilon, t_0 + \varepsilon]$ de $(x(t) - y(t))$. On montre alors que ce maximum est majoré par $k\varepsilon Q$ et donc que $Q = 0$.

□

Proposition I.2. Soit $t_0 \in I \subset \mathbb{R}$. Soit le problème suivant :

$$\begin{cases} \dot{y}(t) = a(t)y(t) \\ y(t_0) = y_0. \end{cases} \tag{I.6}$$

La solution sur I de I.6 est donnée par :

$$\forall t \in I,\ y(t) = y_0 \exp\left(\int_{t_0}^{t} a(u)\mathrm{d}u\right). \tag{I.7}$$

Remarque. Si $a \in \mathbb{K}$ alors la solution est donnée par : $y(t) = y_0 \exp(a(t - t_0))$.

Proposition I.3. Soit $I \subset \mathbb{R}$. On considère le problème avec second membre suivant :

$$\begin{cases} \dot{y}(t) = a(t)y(t) + b(t) \\ y(t_0) = y_0,\ t_0 \in I. \end{cases} \tag{I.8}$$

La solution de ce problème est alors la somme de la solution de l'équation homogène et d'une solution particulière. Elle est donc donnée par :

$$\forall t \in I, y(t) = y_0 \exp\left(\int_{t_0}^{t} a(u)\mathrm{d}u\right) + \int_{t_0}^{t} b(u) \exp\left(\int_{u}^{t} a(v)\mathrm{d}v\right) \mathrm{d}u.$$

On s'intéresse maintenant à la résolution des systèmes de la forme I.5, c'est-à-dire :

$$\dot{y}(t) = A(t)y(t) + b(t). \tag{I.9}$$

Avec $A : I \to \mathcal{M}_n(\mathbb{K})$ et $b \in \mathbb{K}^n$.
Pour cela, on va tout d'abord s'intéresser à quelques résultats qui nous mèneront à l'expression générale des solutions de tels systèmes.

Remarque. Dans le cas d'un système autonome, c'est-à-dire lorsque A ne dépend pas du temps, la proposition I.3 peut être étendue au cas des matrices et on obtient :

$$\forall t \in I, y(t) = y_0 \exp\left(A(t - t_0)\right) + \int_{t_0}^{t} b(u) \exp\left(A(t - u)\right) \mathrm{d}u.$$

Rappel. Soit $A \in \mathcal{M}_n(\mathbb{K})$. On rappelle que l'exponentielle de A est définie par :

$$\exp(A) = \sum_{k=0}^{+\infty} \frac{A^k}{k!}.$$

Cependant, lorsque le système n'est pas autonome, c'est-à-dire lorsque l'on a $A : I \to \mathcal{M}_n(\mathbb{K})$, cette formule n'est plus vérifiée. On doit donc définir plusieurs notions telles que la résolvante et le Wronskien.

Définition I.3. (*Matrice résolvante*)
Soient $t, t_0 \in I$. On appelle résolvante d'un système linéaire homogène l'unique solution du système :

$$\begin{cases} \dot{y}(t) = A(t)y(t) \\ y(t_0) = I_n. \end{cases} \tag{I.10}$$

Cette matrice est alors notée $R^t_{t_0}$.

Proposition I.4. La matrice résolvante associée à un système différentiel vérifie :

i) $\forall s, t, u \in I \quad , \quad R^u_t R^t_s = R^u_s$.

ii) $\forall s, t \in I$, R^t_s est inversible et $(R^t_s)^{-1} = R^s_t$.

Définition I.4. (*Wronskien*)
On appelle Wronskien associé à un système différentiel le déterminant de la matrice résolvante associée à ce système différentiel.

Théorème I.5. (*Formule de Liouville*)
Soit $I \subset \mathbb{R}$.
Soient $y_1, \ldots, y_n$, n solutions du système homogène associé à I.9 (i.e. $b \equiv 0$).
Soit $w(t) = \det(y_1(t), \ldots, y_n(t))$ leur Wronskien.
Alors :

$$\forall \ \ s, t \in I \ \ \ w(t) = w(s) \exp\left(\int_s^t \mathrm{Tr}(A(u)) \mathrm{d}u \right).$$

Pour démontrer le théorème I.5, nous avons besoin de rappeler quelques résultats :

Rappels. i) Soit $A \in \mathcal{M}_n(\mathbb{R})$. Soit $X = (x_1, \ldots, x_n) \in \mathcal{M}_n(\mathbb{R})$. On a alors la formule suivante :

$$\sum_{i=1}^{n} \det(x_1, \ldots, Ax_i, \ldots, x_n) = \mathrm{Tr}\,(A) \det(x_1, \ldots x_n).$$

ii) Dérivée d'une forme multilinéaire.
Soient ϕ une forme multilinéaire et $v_i(t) \in \mathbb{R}^n$, $i = 1, \ldots, n$ tels que :

$$\begin{array}{rccc} \phi : & \mathcal{M}_n(\mathbb{R}) & \longrightarrow & \mathbb{R} \\ & (v_1(t), \ldots, v_n(t)) & \longmapsto & \phi(v_1(t), \ldots, v_n(t)) \end{array}.$$

On a alors :

$$\frac{\mathrm{d}\phi}{\mathrm{dt}}(v_1(t),\dots,v_n(t)) = \sum_{j=1}^{n} \phi(v_1(t),\dots,\frac{\mathrm{d}}{\mathrm{dt}}v_j(t),\dots,v_n(t)).$$

Démonstration. On a $\dot{y} = Ay$ (équation sans second membre).
On pose $Y(t) = (y_1(t),\dots,y_n(t))$ et $A = (a_{ij})_{ij}$.
Soit :

$$w(t) = \det(y_1,\dots,y_n).$$

On a alors :

$$\begin{aligned}
\dot{w}(t) &= \frac{\mathrm{d}}{\mathrm{dt}}\det(y_1,\dots,y_n) \\
&= \sum_{i=1}^{n} \det(y_1,\dots,\frac{\mathrm{d}}{\mathrm{dt}}y_i,\dots,y_n) \\
&= \sum_{i=1}^{n} \det(y_1,\dots,Ay_i,\dots,y_n) \\
&= \mathrm{Tr}\,(A)\det(y_1,\dots,y_n).
\end{aligned}$$

On résout alors entre s et t l'équation différentielle :

$$\begin{cases} \dot{w}(t) = \mathrm{Tr}\,(A)w(t) \\ w(s) = w_s. \end{cases}$$

On obtient finalement par la proposition I.6 la solution :

$$w(t) = w(s)\exp\left(\int_s^t \mathrm{Tr}\,(A(u))\mathrm{d}u\right).$$

□

Théorème I.6. (*Solution générale avec second membre*)
Soit $I \subset \mathbb{R}$. Soit $(t_0, y_0) \in I \times \mathbb{K}^n$.
Il existe une unique solution sur I de l'équation I.9 telle que $y(t_0) = y_0$:

$$\forall t \in I\ ,\ y(t) = R_{t_0}^t y_0 + \int_{t_0}^t R_s^t b(s)\mathrm{d}s. \tag{I.11}$$

I. 1. 2) Équations linéaires d'ordre supérieur

Proposition I.7. Toute équation linéaire d'ordre n est équivalente à un système d'équations linéaires d'ordre 1, de dimension n.
C'est-à-dire que l'on a :

$$\sum_{i=1}^{n} a_{i-1}(t)y^{(i)}(t) = b(t) \Longleftrightarrow \dot{Y} = \begin{pmatrix} 0 & 1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ -a_0 & -a_1 & \dots & \dots & -a_{n-1} \end{pmatrix} Y + \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ b \end{pmatrix}.$$

Avec $Y = \begin{pmatrix} y & y' & y^{(2)} & \dots & y^{(n-1)} \end{pmatrix}^t$.

Un cas particulier est celui des équations à coefficients constants du type :

$$\ddot{y} + a\dot{y} + by = f(t). \tag{I.12}$$

Pour l'équation I.12, le plus simple sera de se ramener à l'équation caractéristique de l'équation homogène :

$$\lambda(t)^2 + a\lambda(t) + b = 0.$$

On cherche alors les solutions de cette équation et 3 cas se présentent :

i) Le discriminant Δ est positif : on a donc deux racines simples, réelles. Toute solution de l'équation homogène associée à I.12 est donnée par :

$$y(t) = \mu_1 \exp(\lambda_1 t) + \mu_2 \exp(\lambda_2 t),\ \mu_1, \mu_2 \in \mathbb{R}.$$

ii) Le discriminant Δ est nul : on a donc une racine réelle multiple. Alors :

$$y(t) = \mu_1 \exp(\lambda t) + \mu_2 t \exp(\lambda t),\ \mu_1, \mu_2 \in \mathbb{R}.$$

iii) Le discriminant Δ est négatif : on a donc deux racines complexes. On pose alors $\lambda_1 = a + ib$ et $\lambda_2 = a - ib$ et on a :

$$y(t) = \exp(at)(\mu_1 \cos(bt) + \mu_2 \sin(bt)),\ \mu_1, \mu_2 \in \mathbb{R}.$$

Il faut ensuite déterminer une solution particulière de l'équation I.12. La solution générale s'obtient en sommant la solution de l'équation homogène et la solution particulière.

Cependant, lorsque les coefficients ne sont pas constants, cette méthode ne s'applique pas. Il faut alors utiliser la proposition I.7 et se ramener à un système d'ordre 1 que l'on résout comme nous l'avons vu précédemment.
Voyons maintenant, à travers un bref exemple, comment passer d'un système d'ordre 2 à un système d'ordre 1.

Exemple :

Prenons un exemple concret : le système solaire externe. Son évolution à travers le temps peut être modélisée par une équation différentielle vectorielle d'ordre 2 :
On pose $Y_i = (x_i, y_i, z_i)$ la position de la planète i dans le repère héliocentrique, m_i sa masse (relative au soleil) et G la constante gravitationnelle :

$$\ddot{Y}_i = G \sum_{j \neq i} m_j \frac{Y_j - Y_i}{\|Y_j - Y_i\|}.$$

En utilisant le fait que $Y_i = (x_i, y_i, z_i)$ puis en posant $V_i = (w_i, t_i, p_i) = (\dot{x}_i, \dot{y}_i, \dot{z}_i)$ on a alors :

$$\ddot{Y}_i = G \sum_{j \neq i} m_j \frac{Y_j - Y_i}{\|Y_j - Y_i\|} \iff \begin{cases} \dot{Y}_i = V_i \\ \dot{V}_i = G \sum_{j \neq i} m_j \frac{x_j - x_i}{\|Y_j - Y_i\|}. \end{cases}$$

Remarque. Il faut noter que pour résoudre une équation d'ordre 2, il est nécessaire de disposer de deux conditions initiales. En effet, on aura besoin d'une condition d'ordre 0 et d'une condition d'ordre 1 : la position initiale ainsi que la dérivée de la position initiale, c'est-à-dire la vitesse initiale.

Nous allons maintenant aborder très rapidement les notions d'équations non-linéaires et d'équations aux dérivées partielles. Le but est ici seulement d'introduire deux exemples : l'équation non-linéaire qui sera utilisée dans la partie programmation de ce rapport et l'équation aux dérivées partielles qui sera elle utilisée en II. 3. de ce rapport.

I. 2. Équations non-linéaires

Intéressons-nous maintenant brièvement à un second type d'équations différentielles : les équations (et systèmes) non linéaires. Dans cette partie, nous allons seulement présenter un exemple. En effet, le but principal de ce projet est la résolution numérique. Or, la résolution numérique de ces systèmes, par exemple avec les méthodes

d'Euler, ne diffère pas des méthodes de résolution des systèmes linéaires.
Tout d'abord, il est important de noter que la proposition I.7 et le théorème de Cauchy-Lipschitz énoncés dans le cadre linéaire restent vrais dans le cadre non-linéaire.

Exemple :

Modèle de Lotka-Volterra.
Le système de Lotka-Volterra (modèle proie-prédateur) est un système de deux équations non linéaires défini ainsi :
Soient $a, b, c, d \in \mathbb{R}$. Soit $t \in [0, T]$ avec $T \in \mathbb{R}^+$.
Soient $x_0, y_0 > 0$. On pose :

$$\begin{cases} \dot{x}(t) = ax(t) - bx(t)y(t), \\ \dot{y}(t) = -cy(t) + dx(t)y(t), \end{cases} \qquad (x(0), y(0)) = (x_0, y_0). \tag{I.13}$$

Le théorème I.1 s'applique également aux équations non-linéaires. On pose :

$$\begin{array}{rccc} f: & [0,T] \times \Omega \subset \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ & \left(t, \begin{pmatrix} x \\ y \end{pmatrix}\right) & \longmapsto & \begin{pmatrix} ax(t) - bx(t)y(t) \\ -cy(t) + dx(t)y(t) \end{pmatrix}. \end{array}$$

On obtient ainsi un problème de Cauchy dans le cadre d'une équation non-linéaire.

I. 3. Équations aux dérivées partielles

Le but de cette partie va être d'introduire brièvement la notion d'équations aux dérivées partielles. Pour cela, nous allons tout d'abord voir 2 définitions puis nous introduirons un exemple très concret et très important dans ce domaine : l'équation de la chaleur. Cependant, cet exemple ne sera pas étudié en détail, étant donné sa complexité. Il sera introduit dans cette partie avant d'être étudié numériquement en section II.

Définition I.5. Soit $u : (x_1, \ldots, x_n) \to u(x_1, \ldots, x_n)$. On appelle équation aux dérivées partielles d'ordre p toute relation :

$$F\left(u, x_1, x_2, \ldots, x_n, u(x_1, \ldots, x_n), \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \ldots, \frac{\partial^p u}{\partial x_n^p}\right) = 0. \tag{I.14}$$

Définition I.6. (*Formes générales des équations d'ordres 1 et 2*)
Soient (pour simplifier) $A, B, C, D, E, F, G \in \mathbb{R}$. Soit $u : (x, y) \to u(x, y)$. La forme générale d'une équation aux dérivées partielles (EDP ou PDE) d'ordre 1 est :

$$A\frac{\partial u}{\partial y} + B\frac{\partial u}{\partial x} + Cu = D.$$

La forme générale d'une équation d'ordre 2 est :

$$A\frac{\partial^2 u}{\partial y^2} + B\frac{\partial^2}{\partial x \partial y} + C\frac{\partial^2}{\partial x^2} + D\frac{\partial u}{\partial y} + E\frac{\partial u}{\partial x} + Fu = G.$$

Exemple :

L'équation de la chaleur unidimensionnelle :

$$\begin{cases} \frac{\partial u}{\partial t} - \alpha\frac{\partial^2 u}{\partial x^2} = 0, & (x, t) \in]0, X[\times]0, T[, \\ u(x, 0) = u_0(x), & x \in]0, X[, \\ u(0, t) = u(X, t) = 0, & t \in]0, T[. \end{cases} \tag{I.15}$$

Cette équation est un problème aux conditions limites. On peut par exemple prendre le cas d'une barre de longueur X avec à ses extrémités une température de 0° et partout ailleurs une température initiale $u_0(x)$. On cherche alors à étudier la température de la barre sur l'intervalle de temps $[0, T]$.
Cette équation est bien évidemment une équation aux dérivées partielles d'ordre 2. Nous verrons une méthode numérique de résolution pour cette équation en II.

II Méthodes de résolution numérique

Maintenant que nous avons abordé une partie de la théorie mathématique des équations différentielles, nous allons passer à l'aspect numérique de ce projet : les méthodes de résolution numérique des équations vues précédemment, EDO linéaires et non linéaires et EDP.

Pour cela, nous allons tout d'abord étudier un schéma de résolution général et extrêmement important : le θ-schéma. Par la suite, nous verrons quelques cas particuliers très utilisés du θ-schéma, notamment les méthodes d'Euler explicite et implicite.

Il est important de préciser l'intérêt des méthodes numériques. Ces méthodes nous permettront d'obtenir une approximation d'une solution particulière, c'est-à-dire la solution d'un problème de Cauchy de condition initiale $y_0 \in \Omega$.

Avant de commencer à cataloguer les différentes méthodes auxquelles nous nous intéresserons, nous allons définir quelques notions importantes et générales des méthodes numériques.
On se place dans le cadre suivant :
Soit $f : I \times \Omega \to \mathbb{R}$ telle que :

$$\begin{cases} \dot{y}(t) = f(t, y(t)) \\ y(t_0) = y_d. \end{cases} \tag{II.1}$$

Définition II.1. (*Schéma explicite*)
Soit Δt un pas de temps et soient $t_0, \ldots, t_n$ des temps tels que $t_i = t_0 + i\Delta t$ pour $i = 0, \ldots, n$. Soit $\phi : I \subset \mathbb{R} \times \Omega \subset \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}^d$.
On appelle schéma (ou méthode) explicite à un pas une relation de récurrence de la forme :

$$\begin{cases} y_{i+1} = y_i + \Delta t \phi(t_i, y_i, \Delta t) \\ y_0 = y_{d\Delta t}. \end{cases} \tag{II.2}$$

Une solution de ce schéma est donnée par un n-uplet $(y_i)_{i=0,\ldots,n}$ de vecteurs de Ω.

Exemple :

Parmi les différents exemples que nous verrons, on peut citer :

1. le schéma d'Euler explicite ;
2. les méthodes de Runge-Kutta explicites.

Ces différents exemples seront détaillés plus tard dans le rapport.

Définition II.2. (*Schéma implicite*)
Soit Δt un pas de temps et soient $t_0, \ldots, t_n$ des temps tels que $t_i = t_0 + i\Delta t$ pour $i = 0, \ldots, n$. Soit $\phi : I \subset \mathbb{R} \times \Omega \subset \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}^d$.
On appelle schéma (ou méthode) implicite à un pas une relation de récurrence de la forme :

$$\begin{cases} y_{i+1} = y_i + \Delta t \phi(t_{i+1}, y_{i+1}, \Delta t) \\ y_0 = y_{d\Delta t}. \end{cases} \tag{II.3}$$

Une solution de ce schéma est donnée par un n-uplet $(y_i)_{i=0,\ldots,n}$ de vecteurs de Ω.

Remarque. Ces schémas sont dits implicites car ils ne permettent pas directement de déterminer la valeur de y_{i+1}. Pour déterminer cette valeur, il sera nécessaire d'utiliser, par exemple, une méthode de Newton.

Exemple :

L'exemple le plus connu de schéma implicite est le schéma d'Euler implicite, aussi connu comme le schéma d'Euler rétrograde ou backward Euler. On peut également citer les schémas de Runge-Kutta implicites.

Définition II.3. (*Erreur locale et erreur globale*)
On définit l'erreur locale d'une méthode comme l'erreur commise par cette méthode à une itération. On a alors :

$$e_i = \|y_i - y(t_i)\| .$$

On peut alors définir l'erreur globale sur $[t_0, t_f]$ de cette méthode :

$$\varepsilon(\Delta t) = \max_{0 \leqslant i \leqslant \frac{t_f}{\Delta t}} e_i.$$

Définition II.4. (*Convergence d'une méthode*)
Soit $y(t)$ la solution de II.1 et soit $(y_i)_{i=0,\dots,n}$ la suite calculée à l'aide d'une méthode numérique de pas Δt.
Cette méthode est dite convergente sur $[t_0, t_n]$ si :

$$\forall\ y_0 \in \mathbb{R}^n\ ,\ \lim_{\Delta t \to 0} \max_{0 \leqslant i \leqslant n} \|y_i - y(t_i)\| = 0.$$

C'est-à-dire :

$$\forall y_0 \in \mathbb{R}^n,\ \lim_{\Delta t \to 0} \varepsilon(\Delta t) = 0.$$

Définition II.5. (*Erreur de consistance d'une méthode*)
On définit l'erreur de consistance d'une méthode par :

$$\mathcal{R}(t, y, \Delta t) = \frac{y(t + \Delta t) - y(t)}{\Delta t} - \phi(t, y(t), \Delta t).$$

Définition II.6. (*Consistance et ordre d'un schéma numérique*)
Un schéma numérique est consistant si pour toute solution $y \in \mathcal{C}^1([t_0, t_n], \Omega)$ d'une équation différentielle, pour tout $t \in [t_0, t_n]$, l'erreur de consistance de ce schéma converge uniformément vers 0 lorsque le pas de temps Δt tend vers 0 avec $t \in [t_0, t_n]$.
On peut alors définir la notion d'ordre d'un schéma numérique. On dit en effet

qu'un schéma est d'ordre $p \in \mathbb{N}^*$ (ou consistant à l'ordre p) si, en supposant $y \in \mathcal{C}^{n+1}([t_0, t_n], \Omega)$, il existe une constante C positive indépendante de t et de Δt telle que l'on ait :

$$\|\mathcal{R}(t, y, \Delta t)\| \leqslant C(\Delta t)^p.$$

On peut également dire qu'une méthode est d'ordre p lorsque l'on a le développement :

$$\mathcal{R}(t, y, \Delta t) = O(\Delta t^p).$$

Définition II.7. (*Stabilité d'une méthode*)
Une méthode numérique est dite stable s'il existe une constante $M > 0$ indépendante du pas Δt de la méthode telle que pour tous (x_i) et (y_i) vérifiant :

$$\begin{cases} x_{i+1} = x_i + \Delta t \phi(t_i, x_i, \Delta t), \ 0 \leqslant i \leqslant n-1 \\ y_{i+1} = y_i + \Delta t \phi(t_i, y_i, \Delta t) + \mathcal{R}(t_i, y, \Delta t), \ 0 \leqslant i \leqslant n-1 \end{cases} \tag{II.4}$$

on ait pour tout $i = 1, \ldots, n$:

$$\|x_i - y_i\| \leqslant M \left(\|x_0 - y_0\| + \sum_{k=0}^{i-1} \|\mathcal{R}(t_k, y, \Delta t)\| \right).$$

En d'autres termes : la méthode est stable si une petite perturbation sur la condition initiale ou sur ϕ entraîne une petite perturbation sur la solution calculée.

Proposition II.1. (*Condition suffisante de stabilité*)
Si la fonction ϕ associée à une méthode numérique est Lipschitzienne en y alors cette méthode numérique est stable. C'est-à-dire s'il existe $L \in \mathbb{R}^+$ tel que :

$$\forall t \in [0, t_n], \forall \Delta t \geqslant 0, \forall x, y \in \Omega, \|\phi(t, y, \Delta t) - \phi(t, x, \Delta t)\| \leqslant L \|y - x\|.$$

Proposition II.2. (*Condition nécessaire et suffisante de consistance*)
Un schéma numérique est consistant si et seulement si :

$$\phi(t, y, 0) = f(t, y).$$

Théorème II.3. Si une méthode numérique est stable et consistante alors cette méthode converge quelle que soit la condition initiale y_0.

Démonstration. Par la définition II.5 :

$$\begin{aligned} \mathcal{R}(t_j, y, \Delta t) &= \frac{y(t_{i+j}) - y(t_j)}{\Delta t} - \phi(t_j, y_j, \Delta t) \\ \Longrightarrow y(t_{j+1}) &= y(t_j) + \Delta t \left(\phi(t_j, y_j, \Delta t) + \mathcal{R}(t_j, y, \Delta t) \right). \end{aligned}$$

C'est-à-dire que (y_j) vérifie l'équation II.4 avec $\mathcal{R}(t_i, y, \Delta t) = \Delta t \mathcal{R}(t_j, y, \Delta t)$.
D'où, par la définition II.7 :

$$\begin{aligned}\exists C > 0,\ \|y(t_j) - y_j\| &\leqslant C \sum_{m=0}^{j-1} \|\mathcal{R}(t_m, y, \Delta t)\| \\ &\leqslant Cj\Delta t \max_{0 \leqslant m \leqslant j-1} \|\mathcal{R}(t_m, y, \Delta t)\|.\end{aligned}$$

Or :

$$\lim_{\Delta t \to 0} \left(Cj\Delta t \max_{0 \leqslant m \leqslant j-1} \|\mathcal{R}(t_m, y, \Delta t)\| \right) = 0.$$

Alors :

$$\lim_{\Delta t \to 0} \|y(t_j) - y_j\| = 0.$$

□

II. 1. Le θ-schéma dans le cadre ordinaire

Définition II.8. (*θ-schéma*)
Soient t_0, $t_n \in I$ le temps initial et le temps final de résolution de notre équation différentielle II.1. On se donne une discrétisation en temps de l'intervalle $[t_0, t_n]$ avec un pas Δt. On a donc $t_n = t_0 + n\Delta t$. De plus, on pose $y_n \approx y(t_n)$ qui est donc une approximation de la solution au temps t_n.
Soit maintenant $\theta \in [0, 1]$.
On peut définir le θ-schéma associé à l'équation différentielle II.1 par la relation de récurrence suivante :

$$y_{i+1} = y_i + \Delta t(\theta f(t_{i+1}, y_{i+1}) + (1 - \theta) f(t_i, y_i)). \tag{II.5}$$

On peut définir ce schéma numérique comme suit :
Soit $\phi : I \subset \mathbb{R} \times \Omega \subset \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}^d$

$$\begin{cases} y_{i+1} = y_i + \Delta t \phi(t_i, y_i, \Delta t) \\ y_0 = y(t_0). \end{cases} \tag{II.6}$$

Avec :

$$\phi(t_i, y_i, \Delta t) = \theta f(t_{i+1}, y_{i+1}) + (1 - \theta) f(t_i, y_i).$$

Remarque. Le principe du θ-schéma repose sur la méthode des différences finies. En effet, on réalise l'approximation :

$$\dot{y}(t) \approx \frac{y(t+\Delta t) - y(t)}{\Delta t}. \tag{II.7}$$

Proposition II.4. (*Erreur de consistance du θ-schéma*)
Soit $\mathcal{R}(t, y, \Delta t)$ l'erreur de consistance du θ-schéma. Cette erreur vaut alors :

$$\mathcal{R}(t, y, \Delta t) = \Delta t \left[\ddot{y}(t)\left(\frac{1}{2} - \theta\right)\right] + (\Delta t)^2 \left[\dddot{y}(t)\left(\frac{1}{6} - \frac{\theta}{2}\right)\right] + O(\Delta t^3).$$

De plus, le θ-schéma est consistant pour tout $\theta \in [0, 1]$.

Rappel. Pour la preuve suivante, nous aurons besoin du développement limité de la composition de fonctions régulières. Soient $y : t \mapsto y(t)$ et $f : (t, y(t)) \mapsto f(t, y(t))$ régulières. On a :

$$\begin{aligned} f(t+\Delta t, y(t+\Delta t)) =& f(t, y(t)) + \Delta t \dot{y}(t) f'(t, y(t)) + \frac{(\Delta t)^2}{2} \ddot{y}(t) f'(t, y(t)) \\ &+ \frac{1}{2}(\Delta t \dot{y}(t))^2 f''(t, y(t)) + O(\Delta t^3). \end{aligned}$$

Démonstration. D'après la définition II.6 page 14, on a :

$$\mathcal{R}(t, y, \Delta t) = \frac{y(t+\Delta t) - y(t)}{\Delta t} - \phi(t, y(t), \Delta t).$$

De plus, d'après la définition du θ-schéma II.8 on obtient :

$$\mathcal{R}(t, y, \Delta t) = \frac{y(t+\Delta t) - y(t)}{\Delta t} - \left[\theta f(t+\Delta t, y(t+\Delta t)) + (1-\theta) f(t, y(t))\right].$$

Puis, en utilisant un développement de Taylor, on obtient :

$$y(t+\Delta t) = y(t) + \Delta t \dot{y}(t) + \frac{\Delta t}{2} \ddot{y}(t) + O(\Delta t^3).$$

On effectue alors un développement limité de $\mathcal{R}$:

$$\begin{aligned}
\mathcal{R}(t,y,\Delta t) &= \frac{y(t)+\Delta t\dot{y}(t)+\frac{\Delta t}{2}\ddot{y}(t)+O(\Delta t^3)-y(t)}{\Delta t} \\
&\quad -[\theta f(t+\Delta t, y(t+\Delta t))+(1-\theta)f(t,y(t))] \\
&= \dot{y}(t)+\frac{\Delta t}{2}\ddot{y}(t)+\frac{(\Delta t)^2}{6}\dddot{y}(t)+O(\Delta t^3) \\
&\quad -\theta\left[f(t,y(t))+\Delta t\dot{y}(t)f'(t,y(t))+\frac{(\Delta t)^2}{2}\ddot{y}(t)f'(t,y(t))\right] \\
&\quad -\theta\left[\frac{1}{2}(\Delta t\dot{y}(t))^2 f''(t,y(t))+O(\Delta t^3)\right] \\
&\quad -(1-\theta)f(t,y(t)) \\
&= \dot{y}(t)-f(t,y(t))+\Delta t\left[\frac{\ddot{y}(t)}{2}-\theta\dot{y}(t)f'(t,y(t))\right] \\
&\quad +(\Delta t)^2\left[\frac{1}{6}\dddot{y}(t)-\frac{\theta}{2}\ddot{y}(t)f'(t,y(t))-\frac{\theta}{2}(\dot{y}(t))^2 f''(t,y(t))\right]+O(\Delta t^3).
\end{aligned}$$

Or :

$$\begin{aligned}
\dot{y}(t)=f(t,y(t)) &\Longrightarrow \ddot{y}(t)=\dot{y}(t)f'(t,y(t)) \\
&\Longrightarrow \dddot{y}(t)f''(t,y(t))(\dot{y}(t))^2+\ddot{y}(t)f'(t,y(t)).
\end{aligned}$$

Notre expression devient donc :

$$\begin{aligned}
\mathcal{R}(t,y,\Delta t) &= [\dot{y}(t)-\dot{y}(t)]+\left[\Delta t(\ddot{y}(t)\left(\frac{1}{2}-\theta\right)\right]+\left[(\Delta t)^2\left(\dddot{y}(t)\left(\frac{1}{6}-\frac{\theta}{2}\right)\right)\right]+O(\Delta t^3) \\
&= \left[\Delta t(\ddot{y}(t)\left(\frac{1}{2}-\theta\right)\right]+\left[(\Delta t)^2\left(\dddot{y}(t)\left(\frac{1}{6}-\frac{\theta}{2}\right)\right)\right]+O(\Delta t^3).
\end{aligned}$$

De plus, lorsque Δt tend vers 0, $\mathcal{R}$ converge uniformément vers 0. Donc ce schéma est consistant. □

On réalise une représentation graphique de l'ordre du θ-schéma en fonction de la valeur du paramètre θ. On obtient la figure 1 sur laquelle on voit que pour toutes les valeurs de $\theta \neq \frac{1}{2}$ le schéma est d'ordre 1 et d'ordre 2 pour $\theta = \frac{1}{2}$.

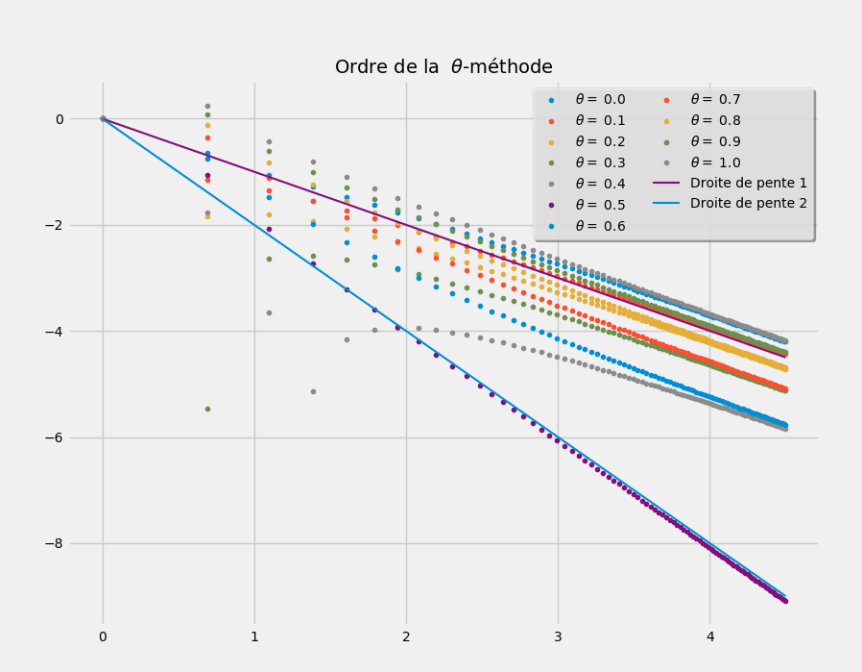


FIGURE 1 – *Ordre du θ-schéma.*

II. 1. 1) Euler Progressif : $\theta = 0$

Nous allons premièrement parler du schéma d'Euler progressif (ou Euler explicite ou encore forward Euler en anglais). Ce schéma nous mènera ensuite à notre première méthode de résolution numérique.
Soit $f : I \subset \mathbb{R}^2 \to \mathbb{R}$. On pose notre problème de Cauchy :

$$\begin{cases} \dot{y}(t) = f(t, y(t)) \\ y(t_0) = y_0. \end{cases} \tag{II.8}$$

Pour cette méthode, le principe est d'approcher localement la courbe de la fonction y par sa tangente. Soit Δt petit, par Taylor, on obtient :

$$y(t_0 + \Delta t) \sim y(t_0) + \Delta t \dot{y}(t_0) = y(t_0) + \Delta t f(y(t_0), t_0).$$

On utilise Δt comme "pas de découpage" de l'intervalle de temps : on découpe l'intervalle en segments de taille Δt. On a, en prenant t_0 le temps initial, t_f le temps final et n le nombre de sous-intervalles de temps que l'on veut obtenir (qui donnera

la "précision" de la méthode) :

$$\Delta t = \frac{t_f - t_0}{n}.$$

Définition II.9. (*Schéma d'Euler explicite*)

Le schéma d'Euler explicite est défini par la relation :

$$\begin{cases} y_{i+1} = y_i + \Delta t f(t_i, y_i) \\ y_0 = y(t_0). \end{cases}$$

Cette méthode est un cas particulier du θ-schéma défini par l'équation II.5 lorsque $\theta = 0$. En effet :

$$\begin{aligned} y_{i+1} &= y_i + \Delta t(\theta f(t_{i+1}, y_{i+1}) + (1-\theta) f(t_i, y_i)) \\ &= y_i + \Delta t(0 f(t_{i+1}, y_{i+1}) + (1-0) f(t_i, y_i)) \\ &= y_i + \Delta t f(t_i, y_i). \end{aligned}$$

Proposition II.5. (*Erreur de consistance et ordre de la méthode*)
L'erreur de consistance de la méthode d'Euler explicite est donnée par :

$$\mathcal{R}(t, y, \Delta t) = \frac{\Delta t}{2} \ddot{y}(t) + O(\Delta t^2).$$

Cette méthode est d'ordre 1.

Démonstration. Nous avons vu que la méthode d'Euler explicite était un cas particulier du θ-schéma donc d'après la proposition II.4 page 17 on a :

$$\mathcal{R}(t, y, \Delta t) = \left[\Delta t(\ddot{y}(t)\left(\frac{1}{2} - \theta\right)\right] + \left[(\Delta t)^2 \left(\dddot{y}(t)\left(\frac{1}{6} - \frac{\theta}{2}\right)\right)\right] + O(\Delta t^3)$$

On remplace alors $\theta = 0$:

$$\begin{aligned} &= \frac{\Delta t}{2} \ddot{y}(t) + \frac{(\Delta t)^2}{6} \dddot{y}(t) + O(\Delta t^3) \\ &= \frac{\Delta t}{2} \ddot{y}(t) + O(\Delta t^2). \end{aligned}$$

La méthode d'Euler explicite est donc d'ordre 1. En effet :

$$\mathcal{R}(t, y, \Delta t) = O(\Delta t).$$

□

Proposition II.6. (*Convergence de la méthode*)
Si la fonction f est Lipschitzienne par rapport à y alors la méthode d'Euler explicite est convergente.

Remarque. Le pseudo-code de la méthode d'Euler explicite est disponible en annexe page 45.

II. 1. 2) Euler Rétrograde : $\theta = 1$

Ce deuxième schéma d'Euler, le schéma rétrograde (Euler implicite ou encore backward Euler en anglais) nous amènera à une seconde méthode de résolution numérique. On se place toujours dans la cadre de l'équation II.8 page 19.
On peut définir la méthode d'Euler implicite de plusieurs façons, tout comme la méthode d'Euler explicite. On choisit ici d'utiliser la définition suivante :

Définition II.10. On appelle schéma d'Euler implicite de pas Δt le schéma suivant :

$$\begin{cases} y_{i+1} = y_i + \Delta t f(t_{i+1}, y_{i+1}) \\ y_0 = y(t_0). \end{cases}$$

Proposition II.7. (*Consistance, stabilité, convergence*)
La méthode d'Euler implicite a les propriétés suivantes :

i) Elle est consistante d'erreur :

$$\mathcal{R}(t, y, \Delta t) = -\frac{\Delta t}{2}\ddot{y}(t) + O(\Delta t^3).$$

ii) Elle est d'ordre 1 ;

iii) Elle est toujours stable ;

iv) Elle est convergente.

Démonstration. La preuve des deux premiers points est similaire aux démonstrations réalisées pour Euler explicite. Il suffit de remplacer $\theta = 0$ par $\theta = 1$. □

Remarque. Le pseudo-code de la méthode d'Euler implicite est disponible en annexe page 45.

II. 1. 3) Crank-Nicolson : $\theta = \frac{1}{2}$

Définition II.11. On appelle schéma de Crank-Nicolson de pas Δt le schéma suivant :

$$\begin{cases} y_{i+1} = y_i + \frac{\Delta t}{2}\left(f(t_i, y_i) + f(t_{i+1}, y_{i+1})\right) \\ y_0 = y(t_0). \end{cases}$$

Proposition II.8. (*Erreur de consistance et ordre de la méthode*)
L'erreur de consistance de la méthode de Crank-Nicolson est donnée par :

$$\mathcal{R}(t, y, \Delta t) = (\Delta t)^2 \dddot{y}(t)\left(\frac{-1}{12}\right) + O(\Delta t^3).$$

Cette méthode est d'ordre 2.

Démonstration. En utilisant la proposition II.4 page 17 on obtient :

$$\mathcal{R}(t, y, \Delta t) = \left[\Delta t(\ddot{y}(t)\left(\frac{1}{2} - \theta\right)\right] + \left[(\Delta t)^2\left(\dddot{y}(t)\left(\frac{1}{6} - \frac{\theta}{2}\right)\right)\right] + O(\Delta t^3)$$

On remplace θ par $\frac{1}{2}$:

$$\begin{aligned} &= \Delta t\left(\ddot{y}(t)\left(\frac{1}{2} - \frac{1}{2}\right)\right) + (\Delta t)^2\left(\dddot{y}(t)\left(\frac{1}{6} - \frac{1}{4}\right)\right) + O(\Delta t^3) \\ &= (\Delta t)^2 \dddot{y}(t)\left(\frac{-1}{12}\right) + O(\Delta t^3). \end{aligned}$$

On a alors :

$$\mathcal{R}(t, y, \Delta t) = O(\Delta t^2).$$

Donc la méthode de Crank-Nicolson est d'ordre 2. □

Remarque. Le pseudo-code de cette méthode est disponible en annexe, page 46.

II. 2. Méthodes de Runge-Kutta

On s'intéresse maintenant aux méthodes de Runge-Kutta, les méthodes d'ordre 1 et d'ordre 4 étant les plus utilisées.
La méthode d'ordre 1 a déjà été définie dans le rapport puisqu'il s'agit de la méthode d'Euler explicite.

Définition II.12. (*Méthode d'ordre 2*)
On définit la méthode de Runge-Kutta d'ordre 2 comme suit :

$$\begin{cases} y_{i+1} = y_i + \Delta t \left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right), \\ y_0 = y(t_0). \end{cases} \quad \text{avec} \quad \begin{cases} k_1 = f(t_i, y_i), \\ k_2 = f(t_i + \Delta t, y_i + \Delta t k_1). \end{cases}$$

Définition II.13. (*Schéma de Runge-Kutta*)
Soit la discrétisation en temps suivante $t_i = t_0 + i\Delta t$ de l'intervalle $[t_0, t_n]$. On définit alors les temps intermédiaires suivants : $t_i^j = t_i + c_j \Delta t$ avec $j \in \{0, \dots, r\}$ et $c_j \in [0, 1]$. Soient $b_k \in [0, 1]$ tels que $\sum b_k = 1$ et $a_{j,k} \in [0, 1]$ tels que $\sum_k a_{j,k} = c_j$. On appelle schéma de Runge-Kutta à r étages le schéma suivant :

$$\begin{cases} y_i^j = y_i + \Delta t \sum_{k=1}^{r} a_{j,k} f(t_i^k, y_i^j), \\ y_{i+1} = y_i + \Delta t \sum_{k=1}^{r} b_k f(t_i^k, y_i^j). \end{cases} \tag{II.9}$$

La solution est alors le n-uplet de vecteurs y_i de Ω.

Remarques.
- Le schéma d'Euler est un schéma de Runge-Kutta à 1 étage.
- Le θ-schéma est lui un schéma de Runge-Kutta à deux étages.

Définition II.14. (*Méthode d'ordre 4*)
On définit la méthode de Runge-Kutta d'ordre 4 (méthode dite "RK4") comme suit :

$$\begin{cases} y_{i+1} = y_i + \Delta t \left(\frac{1}{6}(k_1 + k_4) + \frac{1}{3}(k_2 + k_3)\right), \\ y_0 = y(t_0). \end{cases} \tag{II.10}$$

Avec :

$$\begin{cases} k_1 = f(t_i, y_i), \\ k_2 = f(t_i + \frac{1}{2}\Delta t, y_i + \frac{1}{2}\Delta t k_1, \\ k_3 = f(t_i + \frac{1}{2}\Delta t, y_i + \frac{1}{2}\Delta t k_2), \\ k_4 = f(t_i + \Delta t, y_i + \Delta t k_3). \end{cases}$$

Proposition II.9. (*Stabilité, consistance et ordre de la méthode RK4*)
La méthode de Runge-Kutta d'ordre 4 est une méthode consistante d'ordre 4.

Remarque. Les algorithmes pour les méthodes de Runge-Kutta d'ordre 2 et d'ordre 4 sont disponibles en pseudo-code en annexes 4 page 46 et 5 page 47.

Remarque. Pour les méthodes vues précédemment, on peut également faire un lien avec les méthodes d'intégration numérique. En effet, on peut les retrouver en utilisant la formule suivante :

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t))\mathrm{d}t.$$

On a par exemple :

- Méthode d'Euler explicite : méthode des rectangles à gauche ;
- Méthode d'Euler implicite : méthode des rectangles à droite ;
- Méthodes de Crank-Nicolson et de Runge-Kutta d'ordre 2 : méthode des trapèzes ;
- Méthode de Runge-Kutta d'ordre 4 : méthode de Simpson.

Nous avons parlé de l'ordre de chacune des méthodes énoncées précédemment. On peut représenter graphiquement, avec une échelle logarithmique, ces différents ordres. Pour cela, on utilise un exemple simple d'équation différentielle dont on connaît la solution exacte :

$$\begin{cases} \dot{y}(t) = y(t), \\ y(0) = 1. \end{cases}$$

On représente le logarithme de la norme de la différence entre la solution exacte et la solution approchée, en $t = 1$, en fonction du logarithme du nombre de subdivisions de l'intervalle $[0, 1]$. On vérifie alors les différents ordres en comparant avec des droites de pentes correspondant aux ordres. On obtient la figure 2 page 25.

Remarque. Nous nous sommes ici concentrés sur les méthodes à un pas. Cependant, il existe également des méthodes numériques à pas multiples. On définit alors le pas $h_i = t_{i+1} - t_i$.

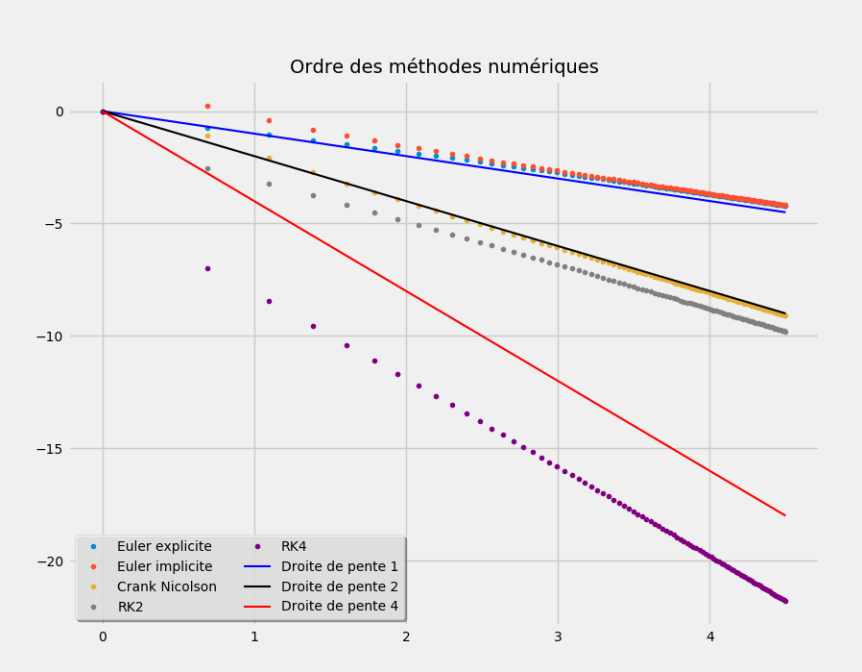


FIGURE 2 – *Ordre des différentes méthodes.*

II. 3. Pour les EDP : différences finies et θ-schéma

Maintenant que nous avons introduit le θ-schéma dans un cadre ordinaire, nous allons voir qu'il peut se généraliser au cadre des équations aux dérivées partielles. Les propositions et théorèmes énoncés dans le cadre ordinaire restent vrais sous certaines conditions dans ce cadre. Les démonstrations ne seront donc pas refaites.

Nous allons tout d'abord établir le cadre dans lequel nous travaillerons puis nous définirons le θ-schéma. Ensuite nous donnerons quelques propriétés.

Pour l'étude de schéma, nous allons nous placer dans le cadre d'équations paraboliques. On se place dans le cadre de l'équation de la chaleur unidimensionnelle, de condition initiale u_0 :

$$\begin{cases} \frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0, & (x,t) \in]0,1[\times]0,T[, \\ u(x,0) = u_0(x), & x \in]0,1[, \\ u(0,t) = u(1,t) = 0, & t \in]0,T[. \end{cases} \tag{II.11}$$

Il est admis qu'il existe une unique solution à cette équation.

Le but est ici de réaliser une approximation numérique de la solution cette équation, en supposant l'existence d'une solution connue. On va pour cela définir le θ-schéma dans le cadre de cette équation. Il s'agit, comme vu, en II. 1. d'un schéma aux différences finies. La méthode consiste en une discrétisation en temps de l'équation (comme dans le cadre ordinaire) combinée à une discrétisation par différences finies en espace.
Pour obtenir une bonne lisibilité, on introduit quelques notations :

- Pour la discrétisation en temps, on note le pas de temps Δt, $t_n = n\Delta t, n \in \{0, \dots, N\}$ tel que $t_N = N\Delta t = T$ avec $N \in \mathbb{N}$.
- Pour la discrétisation en espace, on note le pas Δx, $x_i = i\Delta x, i \in \{0, \dots, I+1\}$ tel que $(I+1)\Delta x = 1$ avec $I \in \mathbb{N}$.
- On note ensuite u_i^n l'approximation de $u(x_i, t_n)$.

On choisit de décomposer la définition du θ-schéma :

1) On introduit tout d'abord les notions de schéma explicite et implicite pour les équations aux dérivées partielles paraboliques ;
2) On définit ensuite le θ-schéma.

Définition II.15. (*Schéma explicite*)
On définit un schéma explicite comme suit :

$$\begin{cases} \frac{u_i^{n+1}-u_i^n}{\Delta t} - \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} = 0, & 1 \leqslant n \leqslant N, 1 \leqslant i \leqslant I, \\ u_i^0 = u_0(x_i), & 1 \leqslant i \leqslant I, \\ u_0^n = u_{I+1}^n = 0, & 0 \leqslant n \leqslant N. \end{cases} \tag{II.12}$$

Remarques.

- On peut écrire le schéma :

$$u_i^{n+1} = u_i^n + \delta(u_{i+1}^n - 2u_i^n + u_{i-1}^n), \text{ avec } \delta = \frac{\Delta t}{\Delta x^2}.$$

- Le terme $\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}$ s'obtient à partir d'une discrétisation par différences finies en espace. En effet, on a :

$$\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} = \frac{\frac{u_{i+1}^n - u_i^n}{\Delta x} - \frac{u_i^n - u_{i-1}^n}{\Delta x}}{\Delta x}.$$

Définition II.16. (*Schéma implicite*)
On définit un schéma implicite comme suit :

$$\begin{cases} \frac{u_i^{n+1}-u_i^n}{\Delta t} - \frac{u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}}{\Delta x^2} = 0, & 1 \leqslant n \leqslant N, 1 \leqslant i \leqslant I, \\ u_i^0 = u_0(x_i), & 1 \leqslant i \leqslant I, \\ u_0^n = u_{I+1}^n = 0, & 0 \leqslant n \leqslant N. \end{cases} \tag{II.13}$$

On peut alors écrire ce schéma :

$$u_i^n = u_i^{n+1} - \delta(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}), \text{ avec } \delta = \frac{\Delta t}{\Delta x^2}.$$

Définition II.17. (θ-*schéma*)
Soit $\theta \in [0,1]$. On définit alors le θ-schéma associé à l'équation II.11 par une combinaison convexe d'un schéma explicite et d'un schéma implicite. Le θ-schéma est en effet défini par :

$$\begin{cases} \frac{u_i^{n+1}-u_i^n}{\Delta t} = \alpha\frac{\theta}{\Delta x^2}(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}) + \alpha\frac{1-\theta}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) \\ u_i^0 = u_0(x_i), \ 1 \leqslant i \leqslant I, \\ u_0^n = u_{I+1}^n = 0, \ 0 \leqslant n \leqslant N. \end{cases} \tag{II.14}$$

Proposition II.10. (*Consistance du θ-schéma*)
Le θ-schéma défini en II.17 est consistant pour tout $\theta \in [0,1]$. Il est d'ordre 1 en temps et d'ordre 2 en espace si $\theta \neq \frac{1}{2}$. C'est-à-dire que l'erreur de troncature de ce schéma vaut $O((\Delta x)^2 + \Delta t)$.
Si $\theta = \frac{1}{2}$ alors on gagne un ordre en temps et le schéma devient d'ordre 2 en temps et en espace, son erreur de troncature est $O((\Delta x)^2 + \Delta t)$. On a alors le cas particulier du schéma de Crank-Nicolson.

Proposition II.11. (*Stabilité du θ-schéma*)
Le θ-schéma défini en II.17 est stable si $\theta < \frac{1}{2}$ à la condition que :

$$\delta \leqslant \frac{1}{2(1-2\theta)}.$$

Si $\theta \geqslant \frac{1}{2}$ alors le schéma est inconditionnellement stable.
Il est donc convergent sous les mêmes conditions.

II. 4. Un exemple graphique

Avant d'introduire l'algorithme principal de ce rapport, l'algorithme Pararéel, il est important de présenter graphiquement une comparaison des méthodes introduites précédemment pour les équations différentielles ordinaires. Nous allons pour cela prendre l'exemple des équations de Lorenz :

$$\begin{cases} \dot{x}(t) = \sigma(y(t) - x(t)) \\ \dot{y}(t) = \rho x(t) - y(t) - x(t)z(t) \\ \dot{z}(t) = x(t)y(t) - \beta z(t). \end{cases} \tag{II.15}$$

La figure 3 représente les points $(x(t), y(t), z(t))$ calculés avec avec les méthodes d'Euler explicite et implicite, la méthode Crank-Nicolson et les méthodes de Runge-Kutta d'ordre 2 et 4.
Sur la figure 4 on a représenté x, y et z en fonction de t pour chacune des méthodes. On a choisi les paramètres $\sigma = 10$, $\rho = 28$ et $b = 8/3$. On travaille sur l'intervalle de temps $[0, 4]$ avec 4000 points et le vecteur initial $y_0 = (5, -5, 20)$.

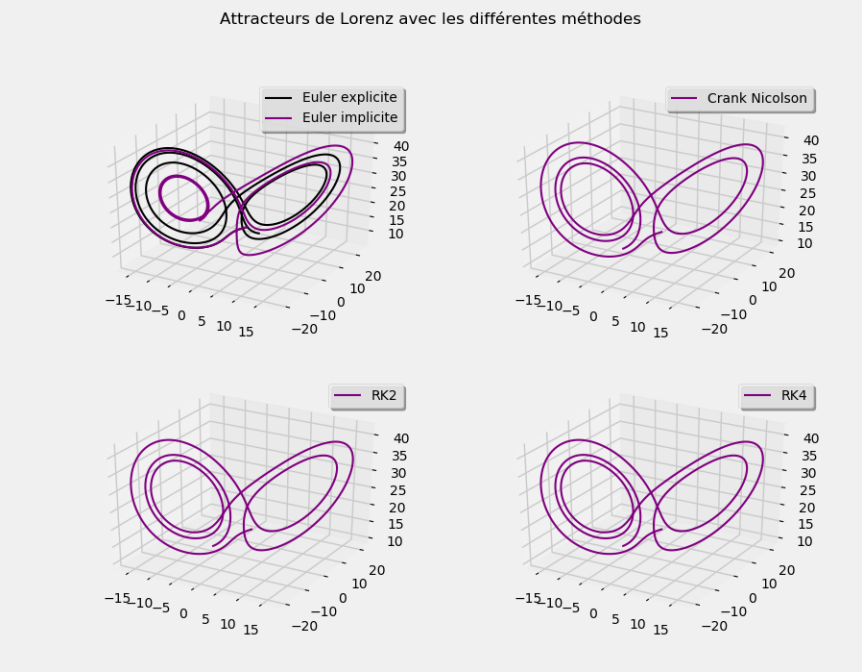


FIGURE 3 – *Représentation 3D des attracteurs de Lorenz.*

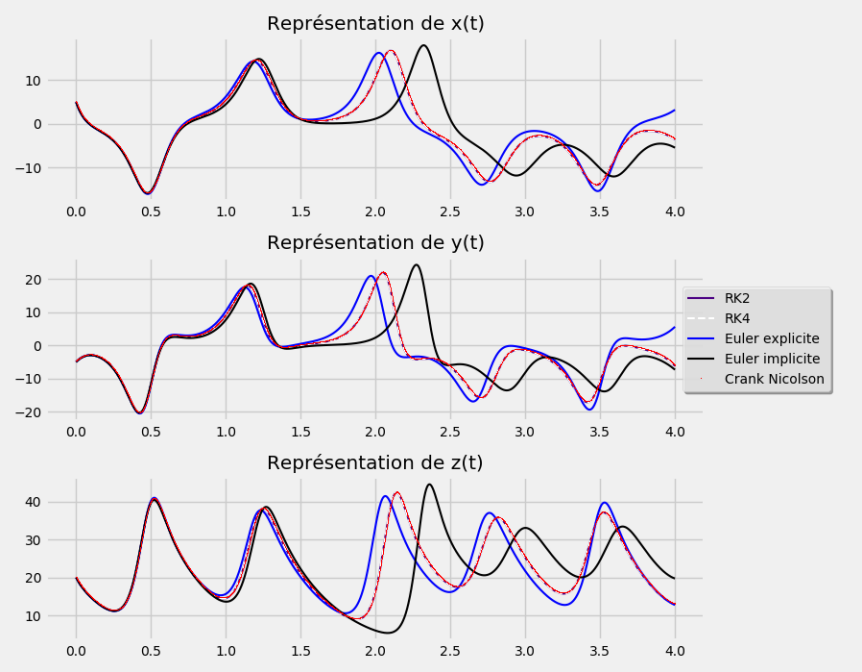


FIGURE 4 – *Représentation des coordonnées en fonction du temps.*

Remarque. On voit bien sur ces représentations l'importance d'utiliser des méthodes de résolution adaptées au problème étudié. Ici, les méthodes d'ordre 1 sont très rapidement imprécises alors que les solutions des méthodes d'ordres 2 et 4 semblent confondues.

III Des méthodes de tirs multiples à l'algorithme Pararéel

III. 1. Introduction aux méthodes de tirs multiples

Le principe des méthodes de tirs multiples est le suivant :
On utilise une décomposition de notre espace-temps et on veut résoudre le problème sur chaque sous-intervalle en utilisant seulement les valeurs de ce sous-intervalle, ce qui permet des résolutions en parallèle. Le but est d'obtenir des approximations sur chaque sous-domaine, pour finalement obtenir une approximation de la solution sur tout l'espace-temps.

En 1964, Jörg Nievergelt a eu l'idée d'essayer de paralléliser des méthodes d'intégration de systèmes différentiels linéaires en utilisant le principe suivant :

- On décompose l'espace-temps en sous-domaines $[t_i, t_{i+1}]$;
- On réalise une approximation (grossière) de la solution sur chaque sous-domaine ;
- On prend plusieurs points initiaux dans un voisinage de chaque approximation faite ;
- On détermine précisément, pour chacun des points initiaux, la trajectoire de la solution sur chaque sous-intervalle ;
- On réalise les calculs de l'approximation réelle de la solution pour chaque sous-intervalle avec un principe de correction. On utilise pour cela 2 trajectoires déterminées sur chaque sous-intervalle telles que l'approximation déterminée avec le solveur grossier soit comprise entre les valeurs finales de ces deux trajectoires.

Jörg Nievergelt a ensuite introduit une méthode pour les équations linéaires ordinaires scalaires.

Par la suite, les méthodes ont été généralisées et ce principe a été adapté à des méthodes itératives afin d'obtenir des résultats plus précis. Comme pour les méthodes de Nievergelt, on utilise le principe de discrétisation de l'intervalle de temps, afin de paralléliser les calculs en utilisant (dans le meilleur des cas) un processeur par intervalle de temps.

III. 2. Les méthodes de tirs multiples

On se place ici dans le cadre d'une équation aux conditions limites. On pourra ensuite arriver au cas particulier des problèmes de Cauchy. On pose le problème aux conditions limites suivant :
Soient $\Omega \subset \mathbb{R}^d$, $I \subset \mathbb{R}^+$ et $f : \quad I \times \Omega \quad \longrightarrow \quad \Omega$ et :

$$\begin{cases} \dot{y}(t) = f(t, y(t)), \\ r(y(t_0), y(T)) = 0. \end{cases} \tag{III.1}$$

Avec r la fonction définissant les conditions aux limites.
On réalise une discrétisation de l'intervalle $[t_0, T] = \bigcup_{i=0}^{n-1} [t_i, t_{i+1}]$ avec $t_n = T$.

On note $\lambda_i = y(t_i)$. On peut maintenant définir n nouveaux problèmes de Cauchy afin de déterminer les trajectoires sur chaque sous-intervalle :

$$\begin{cases} \dot{y} = f(t, y(t)),\ t \in [t_i, t_{i+1}], \\ y(t_i) = \lambda_i. \end{cases} \tag{III.2}$$

On appelle trajectoire sur $[t_i, t_{i+1}]$ de condition initiale λ_i la solution de chaque problème de Cauchy. On la note $y(t_i, t_{i+1}, \lambda_i)$.
Le problème III.1 se transforme donc en n problèmes de Cauchy. Cependant, on ne connaît pas précisément les λ_i. En effet, leur détermination exacte nécessiterait de connaître la solution exacte en tout point t_i, ce qui n'est pas le cas.
On veut donc essayer de déterminer les vecteurs $\lambda_i \in \Omega$ tels que y soit une application continue et vérifie :

$$\begin{cases} y(t_i, t_{i+1}, \lambda_i) = \lambda_{i+1},\ i = 0, \dots n-1, \\ r(\lambda_0, \lambda_n) = 0. \end{cases} \tag{III.3}$$

Le but va être d'annuler les erreurs que l'on aura entre la valeur approchée λ_{i+1} et la trajectoire $y(t_i, t_{i+1}, \lambda_i)$, afin d'avoir une approximation la plus précise possible.
L'équation III.3 nous donne un système de $n+1$ équations de dimension $\dim(\Omega)$, n équations pour les erreurs et une équation pour les conditions aux limites :

$$\begin{cases} y(t_0, t_1, \lambda_0) = \lambda_1 \\ y(t_1, t_2, \lambda_1) = \lambda_2 \\ \vdots \\ y(t_{n-1}, t_n, \lambda_{n-1}) = \lambda_n \\ r(\lambda_0, \lambda_n) = 0. \end{cases} \tag{III.4}$$

On pose ce système sous forme matricielle et on obtient :

$$\begin{bmatrix} r(\lambda_0, \lambda_n) \\ y(t_0, t_1, \lambda_0) - \lambda_1 \\ y(t_1, t_2, \lambda_1) - \lambda_2 \\ \vdots \\ y(t_{n-1}, t_n, \lambda_{n-1}) - \lambda_n \end{bmatrix} = 0 \Longleftrightarrow \begin{bmatrix} F_n(\lambda_0, \lambda_n) \\ F_0(\lambda_0, \lambda_1) \\ F_1(\lambda_1, \lambda_2) \\ \vdots \\ F_{n-1}(\lambda_{n-1}, \lambda_n) \end{bmatrix} = 0.$$

On peut alors poser $\Lambda = (\lambda_0 \dots \lambda_n)^t$ ce qui donne l'équation :

$$F(\Lambda) = \begin{bmatrix} F_n(\lambda_0, \lambda_n) \\ F_0(\lambda_0, \lambda_1) \\ F_1(\lambda_1, \lambda_2) \\ \vdots \\ F_{n-1}(\lambda_{n-1}, \lambda_n) \end{bmatrix} = 0. \tag{III.5}$$

Pour résoudre ce problème, on utilisera généralement des itérations de Newton. On note l'itération k de la méthode de Newton comme suit :

$$\Lambda^{(k+1)} = \Lambda^{(k)} - \left(JF(\Lambda^{(k)})\right)^{-1} F(\Lambda^{(k)}).$$

Il est alors possible d'expliciter la matrice Jacobienne JF en définissant différentes matrices carrées de dimension $\omega = \dim(\Omega)$:

$$\begin{cases} A = \frac{\partial F_n}{\partial \lambda_0}(\Lambda) = \frac{\partial r(\lambda_0, \lambda_n)}{\partial \lambda_0}, \\ B = \frac{\partial F_n}{\partial \lambda_n}(\Lambda) = \frac{\partial r(\lambda_0, \lambda_n)}{\partial \lambda_n}, \\ G_i = \frac{\partial F_i(\Lambda)}{\partial \lambda_i} = \frac{\partial y(t_i, t_{i+1}, \lambda_i)}{\partial \lambda_i}, i = 0, \dots, n-1. \end{cases}$$

En notant I la matrice identité de dimension ω, on a :

$$JF = \begin{pmatrix} A & & & & B \\ G_0 & -I & & & \\ & G_1 & -I & & \\ & & \ddots & -I & \\ & & & G_{n-1} & -I \end{pmatrix}$$

Remarque. Pour le calcul des G_i on utilise la base canonique $(e_j)_j$ et on calcule la colonne j des G_i par différences finies :

$$\frac{\partial F_i(\Lambda)}{\partial \lambda_i}|_j \approx \frac{y(t_i, t_{i+1}, \lambda_i + \Delta t e_j) - y(t_i, t_{i+1}, \lambda_i)}{\Delta t}.$$

On introduit les $\Delta\lambda_i^{(k)}$ représentant la différence entre la composante i de $\Lambda^{(k+1)}$ et celle de $\Lambda^{(k)}$.
On définit :

$$\begin{aligned} \Lambda^{(k+1)} - \Lambda^{(k)} &= \left(\Delta\lambda_0^{(k)} \dots \Delta\lambda_n^{(k)}\right)^t \\ &= -JF(\Lambda^{(k)})^{-1} F(\Lambda^{(k)}). \end{aligned}$$

Alors :

$$F(\Lambda^{(i)}) = -JF(\Lambda^{(k)})(\Lambda^{(k+1)} - \Lambda^{(k)})$$

$$\iff \begin{bmatrix} F_n(\lambda_0^{(k)}, \lambda_n^{(k)}) \\ F_0(\lambda_0^{(k)}, \lambda_1^{(k)}) \\ \vdots \\ F_{n-1}(\lambda_{n-1}^{(k)}, \lambda_n^{(k)}) \end{bmatrix} = - \begin{pmatrix} A & & & & B \\ G_0 & -I & & & \\ & G_1 & -I & & \\ & & \ddots & -I & \\ & & & G_{n-1} & -I \end{pmatrix} \begin{bmatrix} \Delta\lambda_0^{(k)} \\ \Delta\lambda_1^{(k)} \\ \vdots \\ \Delta\lambda_n^{(k)} \end{bmatrix}.$$

On note maintenant $F_i = F_i(\lambda_i^{(k)}, \lambda_{i+1}^{(k)})$. On a donc :

$$\begin{cases} F_n = -A\Delta\lambda_0^{(k)} - B\Delta\lambda_n^{(k)} \\ F_0 = -G_0\Delta\lambda_0^{(k)} + \Delta\lambda_1^{(k)} \\ F_1 = -G_1\Delta\lambda_1^{(k)} + \Delta\lambda_2^{(k)} \\ \vdots \\ F_{n-1} = -G_{n-1}\Delta\lambda_{n-1}^{(k)} + \Delta\lambda_n^{(k)} \end{cases} \iff \begin{cases} A\Delta\lambda_0^{(k)} = -B\Delta\lambda_n^{(k)} - F_n \\ \Delta\lambda_1^{(k)} = G_0\Delta\lambda_0^{(k)} + F_0 \\ \Delta\lambda_2^{(k)} = G_1\Delta\lambda_1^{(k)} + F_1 = G_1(G_0\Delta\lambda_0^{(k)} + F_0) + F_1 \\ \vdots \\ \Delta\lambda_n^{(k)} = G_{n-1}\Delta\lambda_{n-1}^{(k)} + F_{n-1}. \end{cases}$$

On obtient ainsi une relation pour calculer les $\Delta\lambda_i^{(k)}$:

$$\Delta\lambda_i^{(k)} = \left(\prod_{j=0}^{i-1} G_j\right) \Delta\lambda_0^{(k)} + \sum_{j=0}^{i-1} \left(\prod_{m=0}^{j-1} G_{i-1-m}\right) F_{i-1-j}. \tag{III.6}$$

Une fois $\Delta\lambda_0$ calculé, on peut déterminer $\Lambda^{(k)}$ puis l'itération suivante, $\Lambda^{(k+1)}$.

Remarque. Prenons le cas d'un problème de Cauchy, de condition initiale $y_0 \in \Omega$:

$$\begin{cases} \dot{y}(t) = f(t, y(t)), t \in [t_0, T], \\ y(t_0) = y_0. \end{cases} \tag{III.7}$$

On peut obtenir directement $\lambda_0 = y_0$ et on "gagne" donc une contrainte. Cela se traduit par le fait que l'on gagne une dimension dans chaque système et finalement, on supprime la première ligne et la première colonne de la matrice jacobienne $JF(\Lambda^{(k)})$. On obtient donc une nouvelle relation, simplifiée, donnée par :

$$\begin{cases} F_0 = \Delta\lambda_1^{(k)} \\ F_1 = -G_1\Delta\lambda_1^{(k)} + \Delta\lambda_2^{(k)} \\ \vdots \\ F_{n-1} = -G_{n-1}\Delta\lambda_{n-1}^{(k)} + \Delta\lambda_n^{(k)}. \end{cases} \tag{III.8}$$

On obtient finalement la relation :

$$\Delta\lambda_{i+1}^{(k)} = G_i \Delta\lambda_i^{(k)} + F_i,\ i = 0, \dots, n-1. \tag{III.9}$$

On peut aller encore un peu plus loin et déterminer la relation de récurrence pour le calcul des $\lambda_{i+1}^{(k+1)}$. On reprend l'équation III.9 et on utilise tout ce dont on dispose :

$$\begin{cases} \Delta\lambda_{i+1}^{(k)} = \lambda_{i+1}^{(k+1)} - \lambda_{i+1}^{(k)}, \\ \Delta\lambda_i^{(k)} = \lambda_i^{(k+1)} - \lambda_i^{(k)}, \\ F_i = y(t_i, t_{i+1}, \lambda_i^{(k)}) - \lambda_{i+1}^{(k)}. \end{cases} \tag{III.10}$$

L'équation III.9 s'écrit :

$$\begin{aligned} &\Delta\lambda_{i+1}^{(k)} = G_i \Delta\lambda_i^{(k)} + F_i \\ \Longrightarrow\ &\lambda_{i+1}^{(k+1)} - \lambda_{i+1}^{(k)} = G_i(\lambda_i^{(k+1)} - \lambda_i^{(k)}) + y(t_i, t_{i+1}, \lambda_i^{(k)}) - \lambda_{i+1}^{(k)}. \end{aligned}$$

On obtient finalement la relation générale dans le cas d'un problème de Cauchy :

$$\begin{cases} \lambda_0^{(k+1)} = y_0, \\ \lambda_{i+1}^{(k+1)} = G_i(\lambda_i^{(k+1)} - \lambda_i^{(k)}) + y(t_i, t_{i+1}, \lambda_i^{(k)}). \end{cases} \tag{III.11}$$

Théorème III.1. (*Convergence des méthodes de tirs multiples*)

i) Si la fonction f du problème III.1 page 30 est de classe $\mathcal{C}^2$ en y alors la méthode III.11 converge localement de manière quadratique.

ii) Si $\lambda_0^{(0)} = y_0$ alors lorsque $k \geqslant i$, on a :

$$\lambda_i^{(k)} = y(t_i).$$

III. 3. Un cas particulier : l'algorithme Pararéel

Le but de l'algorithme Pararéel (Parareal en anglais) est d'obtenir une approximation relativement précise en un temps relativement faible.
Dans cette partie, on se place dans le cadre d'un problème non linéaire :

$$\begin{cases} \dot{y}(t) = f(t, y(t)),\ t \in [0, T], \\ y(0) = y_0. \end{cases} \tag{III.12}$$

On part d'une discrétisation en temps de l'intervalle $[t_0, t_n]$ avec un pas Δt.
On applique ensuite le principe général suivant :

- Utiliser un solveur grossier (rapide) pour approcher les solutions ($y_i^{(0)}$ de y) aux temps t_i ;
- Utiliser ensuite un solveur fin pour déterminer la solution y sur chaque sous-intervalle $[t_i, t_{i+1}]$ en prenant en condition initiale la solution obtenue avec le solveur grossier sur ces intervalles, c'est-à-dire $y_i^{(0)}$;
- Faire une "correction" et itérer : "$y_{i+1}^{(k+1)} = \mathcal{G}(y_i^{(k+1)}) + \mathcal{F}(y_i^{(k)}) - \mathcal{G}(y_i^{(k)})$" ($\mathcal{G}$= solveur grossier, $\mathcal{F}$= solveur fin).

L'avantage de cet algorithme se verra notamment sur des supercalculateurs. On peut par exemple citer Sunway TaihuLight, un supercalculateur Chinois possédant 10 649 600 cœurs ou bien Behold Summit, un supercalculateur américain, considéré comme le plus puissant en 2019. En effet, le point fort de cet algorithme est de pouvoir réaliser un grand nombre de calculs en parallèle. La seconde étape peut être réalisée en attribuant à chaque processeur un sous-intervalle de temps : chaque processeur réalise donc une approximation, ce qui améliore considérablement le temps de calcul global de l'algorithme.
Soyons maintenant un peu plus précis dans les notations et dans les explications de cet algorithme.
On se donne la discrétisation suivante de l'intervalle $[0, T]$: $[0, T] = \bigcup[t_{i-1}, t_i]$ avec $i = 1, \ldots, N$. On a donc $0 = t_0$ et $T = t_N$.
Comme expliqué précédemment, pour appliquer cet algorithme il est maintenant nécessaire de définir deux solveurs :

- Tout d'abord un solveur grossier : $\mathcal{G}(t_i, t_{i+1}, y_i)$ qui nous donnera une approximation grossière de la trajectoire $y(t_i, t_{i+1}, y_i)$ avec la condition initiale $y(t_i) = y_i$.
- Puis un solveur fin : $\mathcal{F}(t_i, t_{i+1}, y_i)$ qui nous donnera une approximation plus précise de la trajectoire $y(t_i, t_{i+1}, y_i)$ avec la condition initiale t_i.

Remarques. i) Il existe beaucoup de combinaisons différentes de solveurs.

ii) Étant donné que nous définissons deux solveurs différents, il faudra également définir deux pas de temps différents : un pas fin, δt et un pas grossier, Δt.

On commence par initialiser la "solution" à l'itération 0 avec une approximation grossière, on a donc :

$$\begin{cases} y_0^{(0)} := y_0 \\ y_{i+1}^{(0)} := \mathcal{G}(t_i, t_{i+1}, y_i^{(0)}). \end{cases}$$

On obtient la solution "initiale" : $Y^{(0)} = (y_0^{(0)}, \dots, y_N^{(0)})$.
On itère ensuite cela, jusqu'au nombre maximal d'itérations souhaité, K.
On a la définition suivante :

Définition III.1. (*Algorithme Pararéel*)
L'algorithme Pararéel de solveurs fin $\mathcal{F}$ et grossier $\mathcal{G}$ est défini pour $k = 0, \dots, K-1$ par la relation :

$$\begin{cases} y_0^{(k+1)} = y_0 \\ y_{i+1}^{(k+1)} = \mathcal{F}(t_i, t_{i+1}, y_i^{(k)}) + \mathcal{G}(t_i, t_{i+1}, y_i^{(k+1)}) - \mathcal{G}(t_i, t_{i+1}, y_i^{(k)}),\ i \in [\![0, N-1]\!]. \end{cases} \tag{III.13}$$

On obtient à chaque itération un vecteur approchant la solution.
Cet algorithme est un algorithme de parallélisation à travers le temps. En effet, une fois la première itération obtenue, on dissocie les calculs sur chaque intervalle de temps, on fait donc des calculs en parallèle, ce qui améliore le temps de calcul.
Étudions maintenant la théorie mathématique de cet algorithme.

Théorème III.2. Avec les notations précédemment définies, si $k \geqslant i$ alors :

$$y_i^{(k)} = \mathcal{F}(0, t_i, y_0).$$

Autrement dit, lorsque k est supérieur ou égal à i, l'approximation $y_i^{(k)}$ faite avec l'algorithme coïncide avec l'approximation faite avec la méthode fine.

Démonstration. On va prouver ce théorème par une induction sur i.
Pour $i = 0$: $y_0^{(k)} = y_0$.
Supposons maintenant que l'on a $y_i^{(k)} = \mathcal{F}(0, t_i, y_0)$ pour tout $k \geqslant i$. Alors, au rang $i+1$, on applique l'algorithme Pararéel et on a :

$$y_{i+1}^{(k+1)} = \mathcal{F}(t_i, t_{i+1}, y_i^{(k)}) + \mathcal{G}(t_i, t_{i+1}, y_i^{(k+1)}) - \mathcal{G}(t_i, t_{i+1}, y_i^{(k)}).$$

Avec l'hypothèse d'induction on a :

$$y_i^{(k+1)} = \mathcal{F}(0, t_i, y_0) = y_i^{(k)}.$$

Alors :

$$\mathcal{G}(t_i, t_{i+1}, y_i^{(k+1)}) - \mathcal{G}(t_i, t_{i+1}, y_i^{(k)}) = 0.$$

D'où :

$$y_{i+1}^{(k+1)} = \mathcal{F}(t_i, t_{i+1}, y_i^{(k)}).$$

Finalement :

$$\begin{aligned} y_{i+1}^{(k+1)} &= \mathcal{F}(t_i, t_{i+1}, \mathcal{F}(0, t_i, y_0)) \\ &= \mathcal{F}(0, t_{i+1}, y_0). \end{aligned}$$

□

Théorème III.3. (*Lien entre algorithme Pararéel et tirs multiples*)
L'algorithme Pararéel est un cas particulier des méthodes de tirs multiples.

Démonstration. Rappelons tout d'abord que l'algorithme Pararéel est utilisé dans le cadre d'un problème de Cauchy.
On est donc dans le cadre de la remarque III. 2. page 33 que l'on peut alors utiliser.
En reprenant l'équation III.11 on note notre méthode :

$$\begin{cases} \lambda_0^{(k+1)} = y_0, \\ \lambda_{i+1}^{(k+1)} = G_i(\lambda_i^{(k+1)} - \lambda_i^{(k)}) + y(t_i, t_{i+1}, \lambda_i^{(k)}). \end{cases} \tag{III.14}$$

En reprenant les notations de l'algorithme Pararéel $\mathcal{F}$ et $\mathcal{G}$, on pose :

$$\begin{cases} y(t_i, t_{i+1}, \lambda_i^{(k)}) = \mathcal{F}(t_i, t_{i+1}, \lambda_i^{(k)}) \\ G_i\lambda_i^{(k+1)} = \mathcal{G}(t_i, t_{i+1}, \lambda_i^{(k+1)}) \\ G_i\lambda_i^{(k)} = \mathcal{G}(t_i, t_{i+1}, \lambda_i^{(k)}). \end{cases}$$

On peut alors réécrire l'équation III.14 et on retrouve immédiatement la formule définissant l'algorithme Pararéel :

$$\begin{cases} \lambda_0^{(k+1)} = y_0 \\ \lambda_{i+1}^{(k+1)} = \mathcal{F}(t_i, t_{i+1}, \lambda_i^{(k)}) + \mathcal{G}(t_i, t_{i+1}, \lambda_i^{(k+1)}) - \mathcal{G}(t_i, t_{i+1}, \lambda_i^{(k)}). \end{cases} \tag{III.15}$$

□

Théorème III.4. (*Convergence de l'algorithme Pararéel*)
Soient $\mathcal{F}$ et $\mathcal{G}$ respectivement un solveur fin et un solveur grossier tels que :

- Le solveur fin est suffisamment précis pour qu'on puisse le considérer exact ;
- L'erreur locale de troncature du solveur grossier est majorée par $C_3\Delta t^{p+1}$, c'est-à-dire que la méthode associée au solveur grossier $\mathcal{G}$ est d'ordre $p+1$;
- Il vérifie la condition de Lipschitz :

$$\|\mathcal{G}(t, t+\Delta t, x) - \mathcal{G}(t, t+\Delta t, y)\| \leqslant (1 + C_2\Delta t)\,\|x - y\|\,;$$

- Soient c_m continûment différentiables avec $m = p+1, p+2, \ldots$, alors :

$$\mathcal{F}(t_{i-1}, t_i, x) - \mathcal{G}(t_{i-1}, t_i, x) = c_{p+1}(x)\Delta t^{p+1} + c_{p+2}(x)\Delta t^{p+2} + \ldots.$$

Sous ces conditions, à l'itération k de l'algorithme, on a :

$$\begin{aligned}\left\|y(t_i) - Y_i^k\right\| &\leqslant \frac{C_3}{C_1}\frac{(C_1\Delta t^{p+1})^{k+1}}{(k+1)!}(1 + C_2\Delta t)^{i-k-1}\prod_{j=0}^{k}(i-j) \\ &\leqslant \frac{C_3}{C_1}\frac{(C_1 t_i)^{k+1}}{(k+1)!}\exp[C_2(t_i - t_{k+1})]\Delta t^{p(k+1)}.\end{aligned}$$

En d'autres termes : si $\mathcal{G}$ vérifie la condition de Lipschitz et est d'ordre $p+1$, si l'approximation fine est d'une précision telle que l'on peut la considérer exacte alors, à l'itération k, l'algorithme est consistant et se comporte comme une méthode d'ordre $(k+1)p$ pour le problème III.12 page 34 avec un temps maximal T fini.

Proposition III.5. (*Temps de convergence*)
Soient $\mathcal{F}$ et $\mathcal{G}$ de pas δt et Δt, convergents et stables pour ces pas. Alors, la différence entre la solution séquentielle et la solution parallèle atteint la précision machine au plus en n itérations avec $n = \frac{T-t_0}{\Delta t}$.

Remarque. L'algorithme Pararéel est présenté en pseudo-code en annexe 6 page 48.

III. 4. La parallélisation : un avantage en termes de temps

Un des points importants de l'algorithme Pararéel est qu'il peut être utilisé avec plusieurs méthodes numériques. Généralement, nous utiliserons la même méthode fine pour obtenir une précision satisfaisante : la méthode RK4 avec un pas suffisamment fin. Cependant, il peut être intéressant de modifier la méthode grossière afin de visualiser les différences obtenues. Certaines illustrations et comparaisons sont donc disponibles dans le Notebook.

Nous allons pour terminer étudier le point essentiel de la parallélisation : le gain de temps. Pour cela, nous allons énoncer une proposition permettant d'expliciter le *Speedup* de l'algorithme Pararéel. Puis, nous ferons trois représentations graphiques permettant de visualiser le gain de temps réalisé.

Proposition III.6. (*Speedup*)
Le Speedup de l'algorithme est donné par :

$$S := \frac{\tau_s}{\tau_p}.$$

Avec τ_s le temps d'exécution de la méthode fine appliquée de manière séquentielle et τ_p le temps d'exécution de l'algorithme Pararéel appliqué en parallèle sur P processeurs.
Soient $T_{\mathcal{F}}$ et $T_{\mathcal{G}}$ les temps d'exécution de la méthode fine et de la méthode grossière sur un intervalle de temps $[0, t_n]$ avec des pas de temps respectifs δt et Δt.

Pour K itérations de l'algorithme, on applique une première fois la méthode grossière à tout l'intervalle puis K fois la méthode fine et la méthode grossière. Le temps d'exécution de l'algorithme parallèle avec K itérations est alors :

$$\tau_p = T_{\mathcal{G}} + K\left(\frac{T_{\mathcal{F}}}{P} + T_{\mathcal{G}}\right).$$

Le speedup théorique de l'algorithme Pararéel est donc :

$$S = \frac{\tau_s}{\tau_p} = \frac{T_{\mathcal{F}}}{T_{\mathcal{G}} + K\left(\frac{T_{\mathcal{F}}}{P} + T_{\mathcal{G}}\right)}.$$

Remarque. Le speedup de l'algorithme vérifie :

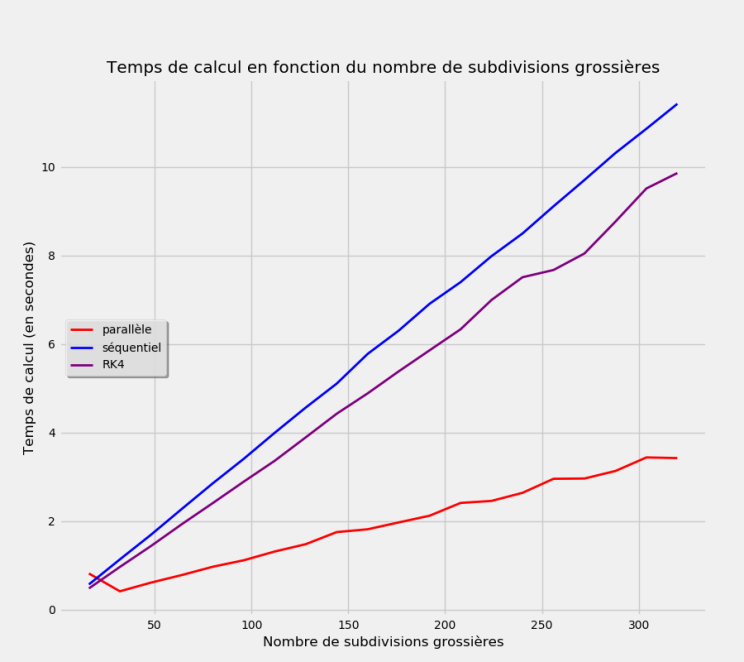
$$S \leqslant \min\left\{\frac{T_{\mathcal{F}}}{T_{\mathcal{G}}}, \frac{P}{K}\right\}.$$

Pour les deux premières représentations graphiques, on choisit de comparer le temps mis entre l'algorithme Pararéel appliqué en parallèle et la méthode fine : la méthode de Runge-Kutta d'ordre 4, appliquée de manière séquentielle. On comparera également avec le temps mis par la version séquentielle de l'algorithme Pararéel. On réalise pour cela une seule itération de l'algorithme Pararéel et on applique la méthode RK4 sur un même exemple : les équations de Lorenz. On choisit pour l'algorithme Pararéel les paramètres suivants : la méthode RK4 en méthode fine et la méthode d'Euler explicite en méthode grossière. On se place sur l'intervalle de temps $[0, 4]$ et on fait varier les pas de temps. On sépare notre intervalle de base en $NG = 16$ sous-intervalles puis, sur chaque sous-intervalle, on applique notre méthode fine qui calculera 10^3 points à partir de chaque point obtenu avec la méthode grossière.

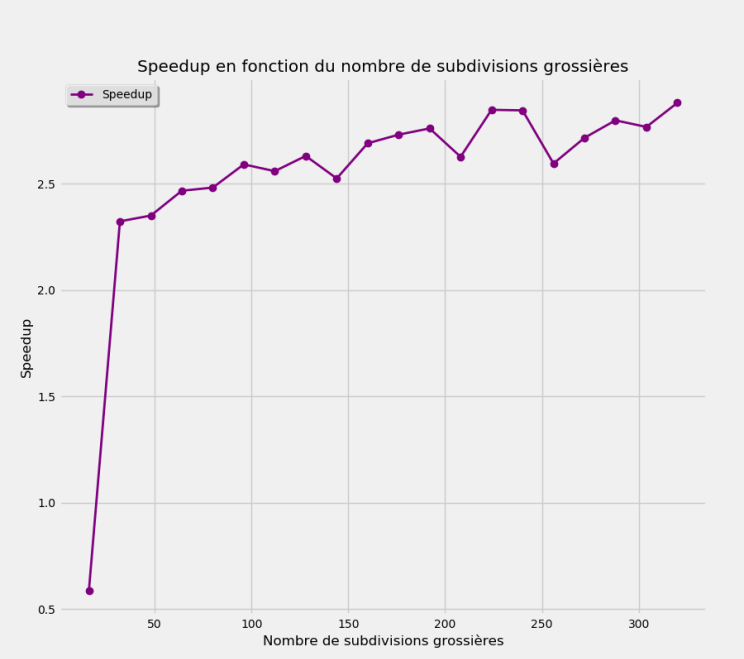
On fait ensuite varier le nombre de sous-intervalles grossiers NG de 16 en 16 jusqu'à 320. On applique la méthode fine RK4 avec $NG \times 10^3$ sous-intervalles. Les tests sont réalisés sur un ordinateur possédant 8 cœurs physiques (16 processeurs logiques). On obtient alors les résultats présentés sur la figure 5a page 41. On représente ensuite le speedup de l'algorithme Pararéel sur la figure 5b page 41.

Remarques.
- D'autres représentations similaires à la figure 5a seront disponibles en annexe IV.
- Les figures 11a et 11b en page 51 sont des représentations obtenues pour 3 itérations.

On se propose de conclure avec une dernière représentation graphique : le speedup en fonction du nombre de cœurs utilisés, au maximum 16 processeurs logiques ici. On reprend l'exemple des attracteurs de Lorenz sur l'intervalle $[0, 4]$ avec un pas de temps grossier $\Delta t = \frac{4}{64}$ pour la méthode d'Euler explicite et un pas fin $\delta t = \frac{\Delta t}{10^3}$ pour RK4. On obtient la figure 6 page 42.



(a) *Temps de calcul.*



(b) *Speedup de l'algorithme Pararéel.*

FIGURE 5 – *Représentations avec* $NF = 1000$ *et* NG *variant.*

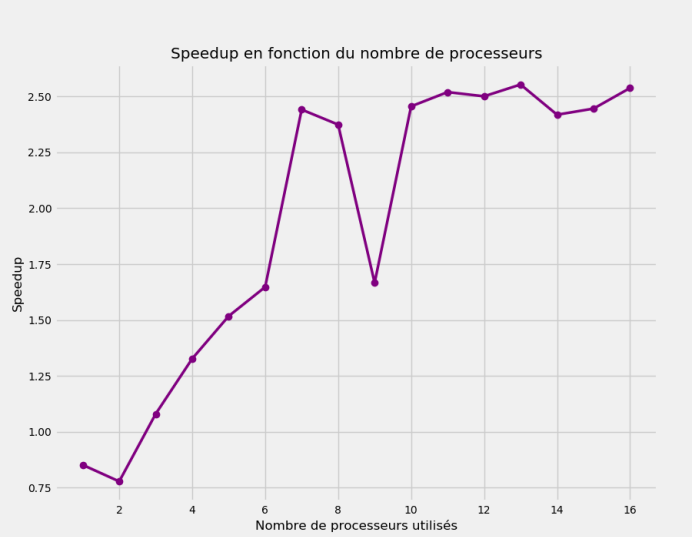


FIGURE 6 – *Représentation du speedup en fonction du nombre de processeurs.*

Conclusion

Nous avons, à travers ces quelques pages, abordé la notion d'équations différentielles, en particulier les équations différentielles ordinaires. Cela nous a mené à la théorie des méthodes de résolution numérique de ces équations différentielles. Nous avons alors vu de façon plus détaillée certaines de ces méthodes ainsi que leurs propriétés. Ces méthodes nécessaires à l'élaboration de l'algorithme Pararéel nous ont permis d'introduire la notion de parallélisation en temps. Nous avons alors traité l'algorithme Pararéel, introduit en tant que méthode de tirs multiples. Après avoir vu certaines propriétés de cet algorithme, nous avons sur un exemple réalisé des comparaisons mettant en valeur les avantages de la parallélisation en temps.

Un enrichissement intéressant, suite à ce projet, serait d'étendre la notion de parallélisation aux équations aux dérivées partielles. En effet, nous avons introduit le θ-schéma dans le cadre de ces équations, notamment l'équation de la chaleur. Il serait alors intéressant de voir comment paralléliser la résolution de ce type d'équations.

Bibliographie

[1] Philippe Chartier. Chapitre 3 : solution des équations différentielles linéaires. http://www.irisa.fr/ipso/fichiers/cours3.pdf.

[2] Jacques Rappaz et Marco Picasso. *Introduction à l'analyse numérique*. 2003.

[3] Martin J. Gander. Time parallel time integration. https://www.lpp.polytechnique.fr/IMG/pdf_EquaDiffS4.pdf.

[4] David Guibert. *Analysis of parallel methods for solving stiff ODE and DAE*. Theses, Université Claude Bernard - Lyon I, September 2009.

[5] Vincent Le Gallic Hadrien Batmalle, Pierre-Elliott Bécue. Accélération des méthodes de parallélisation en temps par approches de type quasi-newton. http://dev.ipol.im/~morel/M%E9moires_Stage_Licence_2011/Batmalle_B%E9cue_Le_Gallic.pdf.

[6] Raphaèle Herbin. Analyse numérique des équations aux dérivées partielles. Lecture, October 2011.

[7] Aline LEFEBVRE-LEPOT. Fiche de cours 5 : Equations différentielles linéaires. http://www.cmap.polytechnique.fr/~lefebvre/SEMESTRE_EV2/Cours5.pdf.

[8] Guillaume Legendre. Introduction à l'analyse numérique et au calcul scientifique. https://www.ceremade.dauphine.fr/~lissy/TD/AnNum1819/cours-legendre-2018.pdf.

[9] Yvon Maday. The parareal in time algorithm. https://www.ljll.math.upmc.fr/publications/2008/R08030.pdf.

[10] Yvon Maday and Olga Mula. An Adaptive Parareal Algorithm. working paper or preprint, October 2019.

[11] Mathieu Mansuy. Rappels : Equations différentielles d'ordre 1 et 2. https://webusers.imj-prg.fr/~mathieu.mansuy/pdf/EDO-revisions-edl12.pdf.

[12] Jean-Luc Raimbault. Equations differentielles cours et exercices. https://www.lpp.polytechnique.fr/IMG/pdf_EquaDiffS4.pdf.

[13] Julien Vovelle. Equations différentielles - cours no 3 equations différentielles linéaires. http://math.univ-lyon1.fr/~vovelle/3Cours2.pdf.

IV Annexes

Représentation graphique d'une solution de I.3

On détermine la solution exacte avec la formule I.7 page 6. On trouve alors :

$$y(t) = \exp(-\sin(t)).$$

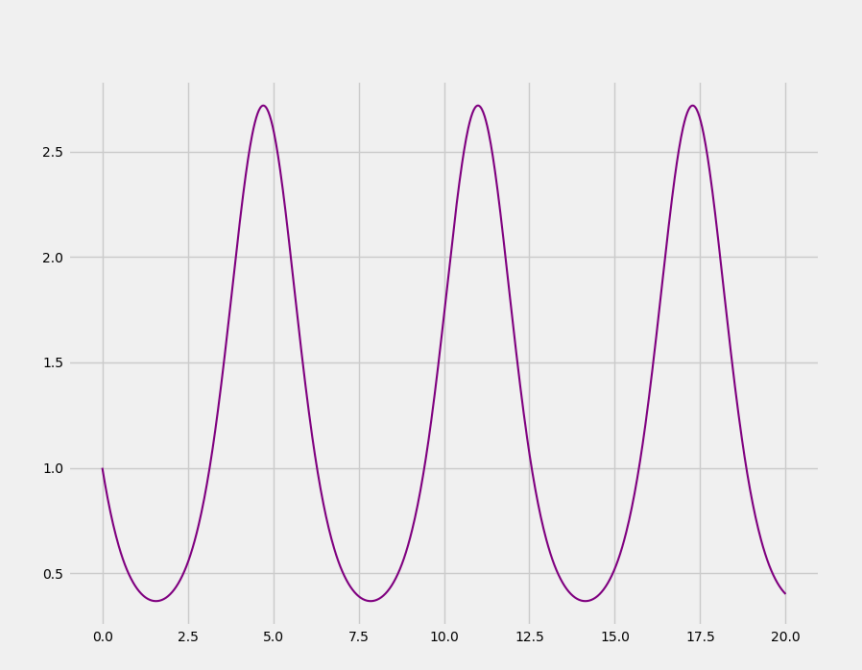


FIGURE 7 – *Représentation d'une solution de I.3.*

Méthode d'Euler explicite

Algorithme 1 : Algorithme d'Euler explicite

Résultat : Vecteur $(y_0, \ldots, y_n)$.
Entrées : f, t_0, t_f, y_0, n;
Initialisation :
Initialiser le pas de temps $\Delta t = \dfrac{T}{n}$;
Discrétiser l'intervalle de temps en $n+1$ points t_i tels que : $t_i - t_{i-1} = \Delta t$;
Traitement :
pour $i = 0, \ldots, n-1$ **faire**
  Calcul de y_{i+1} :

$$y_{i+1} = y_i + \Delta t f(y_i, t_i).$$

fin

Méthode d'Euler implicite

Algorithme 2 : Algorithme d'Euler implicite

Résultat : Vecteur $(y_0, \ldots, y_n)$.
Entrées : f, t_0, t_f, y_0, n;
Initialisation :
Initialiser le pas de temps : $\Delta t = \dfrac{T}{n}$;
Discrétiser l'intervalle de temps en $n+1$ points t_i tels que : $t_i - t_{i-1} = \Delta t$;
Traitement :
pour $i = 0, \ldots, n-1$ **faire**
  Calcul de y_{i+1} :

$$y_{i+1} = y_i + \Delta t f(y_{i+1}, t_{i+1}).$$

fin

Méthode de Crank-Nicolson

Algorithme 3 : Algorithme de Crank-Nicolson

Résultat : Vecteur $(y_0, \ldots, y_n)$.
Entrées : f, t_0, t_f, y_0, n;
Initialisation :
Initialiser le pas de temps : $\Delta t = \dfrac{T}{n}$;
Discrétiser l'intervalle de temps en $n+1$ points t_i tels que : $t_i - t_{i-1} = \Delta t$;
Traitement :
pour $i = 0, \ldots, n-1$ **faire**
  Calcul de y_{i+1} :

$$y_{i+1} = y_i + \frac{1}{2}\Delta t f(y_{i+1}, t_{i+1}) + \frac{1}{2}\Delta t f(y_i, t_i).$$

fin

Méthode de Runge-Kutta d'ordre 2

Algorithme 4 : Algorithme RK2

Résultat : Vecteur $(y_0, \ldots, y_n)$.
Entrées : f, t_0, t_f, y_0, n;
Initialisation :
Initialiser le pas de temps : $\Delta t = \dfrac{T}{n}$;
Discrétiser l'intervalle de temps en $n+1$ points t_i tels que : $t_i - t_{i-1} = \Delta t$;
Traitement :
pour $i = 0, \ldots, n-1$ **faire**
  Calcul des coefficients :

$$k_1 = f(y_i, t_i)$$
$$k_2 = f(y_i + \Delta t k_1, t_i + \Delta t).$$

  Calcul de y_{i+1} :

$$y_{i+1} = y_i + \frac{\Delta t}{2}(k_1 + k_2).$$

fin

Méthode de Runge-Kutta d'ordre 4

Algorithme 5 : Algorithme RK4

Résultat : Vecteur $(y_0, \ldots, y_n)$.
Entrées : f, t_0, t_f, y_0, n;
Initialisation :
Initialiser le pas de temps : $\Delta t = \dfrac{T}{n}$;
Discrétiser l'intervalle de temps en $n+1$ points t_i tels que : $t_i - t_{i-1} = \Delta t$;
Traitement :
pour $i = 0, \ldots, n-1$ **faire**
Calcul des coefficients :

$$\begin{aligned} k_1 &= f(y_i, t_i) \\ k_2 &= f(y_i + \frac{\Delta t}{2} k_1, t_i + \frac{\Delta t}{2}) \\ k_3 &= f(y_i + \frac{\Delta t}{2} k_2, t_i + \frac{\Delta t}{2}) \\ k_4 &= f(y_i + \Delta t k_3, t_i + \Delta t). \end{aligned}$$

Calcul de y_{i+1} :

$$y_{i+1} = y_i + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

fin

Algorithme Pararéel

Pour des raisons de lisibilité, on notera $\lambda^{(k)}$ le vecteur solution à l'itération k, vecteur composé des approximations $\lambda_i^{(k)}$ aux temps t_i. On a alors : $\lambda^{(k)} = (\lambda_0^{(k)}, \ldots, \lambda_n^{(k)})$.
On supposera ici que l'on dispose d'un supercalculateur disposant de NG processeurs.

Algorithme 6 : Algorithme Pararéel

Résultat : Vecteur λ contenant les approximations à chaque itération.
Entrées : $\mathcal{F}, \mathcal{G}, T, y_0, NF, NG, K$;
Initialisation : Initialiser le pas de temps grossier : $\Delta t = \dfrac{T}{NG}$;
Discrétiser l'intervalle de temps en $NG+1$ points t_i tels que : $t_i - t_{i-1} = \Delta t$;
Initialiser le pas de temps fin : $\delta t = \frac{\Delta t}{NF}$;
Discrétiser chaque intervalle $[t_i, t_{i+1}]$ en $NF+1$ points t_j tels que $t_j - t_{j+1} = \delta t$;
Initialiser le vecteur des solutions : $\lambda_0^{(0)} = y_0$;
Traitement :
pour $i = 0, \ldots, NG-1$ **faire**
  Calcul des composantes de $\lambda^{(0)}$:

$$G_{i+1}^{(0)} = \mathcal{G}(t_i, t_{i+1}, \lambda_i^{(0)})$$
$$\lambda_{i+1}^{(0)} = G_{i+1}^{(0)}.$$

fin
pour $k = 0, \ldots, K-1$ **faire**
  Approximation fine en parallèle :
  pour *chaque processeur* $i = 0, \ldots, NG-1$ **faire**
    pour *j=0,..., NF-1* **faire**

$$F_{i+1,j+1}^{(k)} = \mathcal{F}(t_j, t_{j+1}, \lambda_i^{(k)});$$

    fin
  fin
  On donne la condition initiale pour $\lambda^{(k+1)}$: $\lambda_0^{(k+1)} = y_0$;
  pour $i = 0, \ldots, NG-1$, *en séquentiel* **faire**
    Calcul de $\lambda_{i+1}^{(k+1)}$:

$$\lambda_{i+1}^{(k+1)} = F_{i+1,NF}^{(k)} + \mathcal{G}(t_i, t_{i+1}, \lambda_i^{(k+1)}) - \mathcal{G}(t_i, t_{i+1}, \lambda_i^{(k)}).$$

  fin
fin

Représentations du Speedup de l'algorithme Pararéel

On présente ici quelques résultats obtenus avec un nombre plus élevé de points, ce qui prend plus de temps.

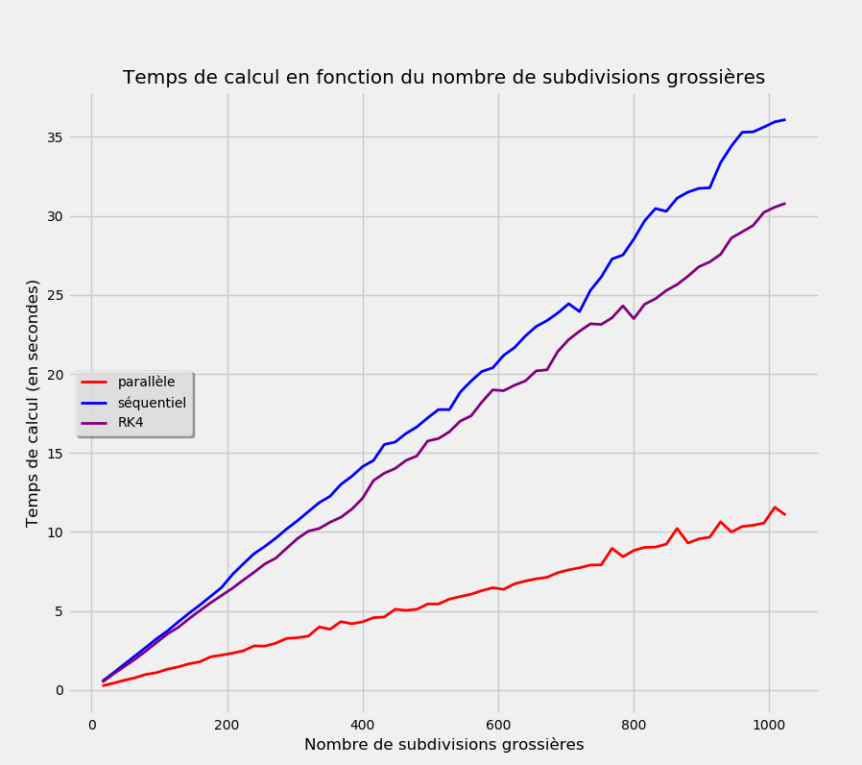


FIGURE 8 – *Comparaison avec des pas* $NF = 1000$ *et* NG *variant de 16 à 1024.*

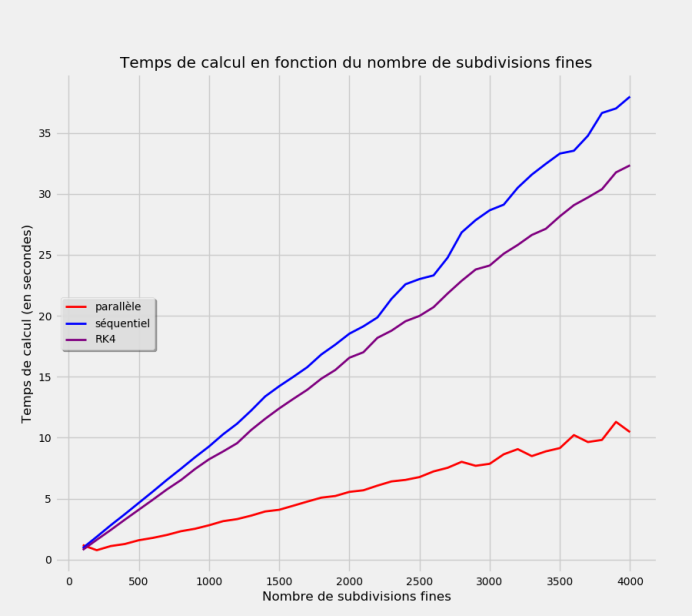


FIGURE 9 – *Représentation avec* $NG = 256$ *et* NF *variant.*

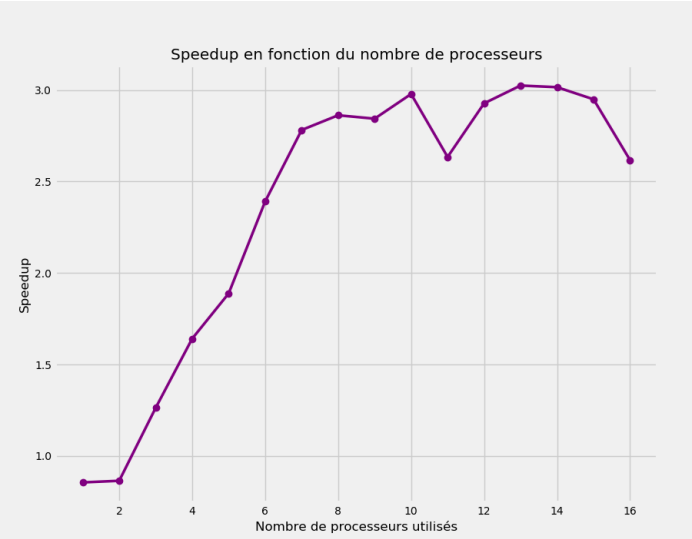
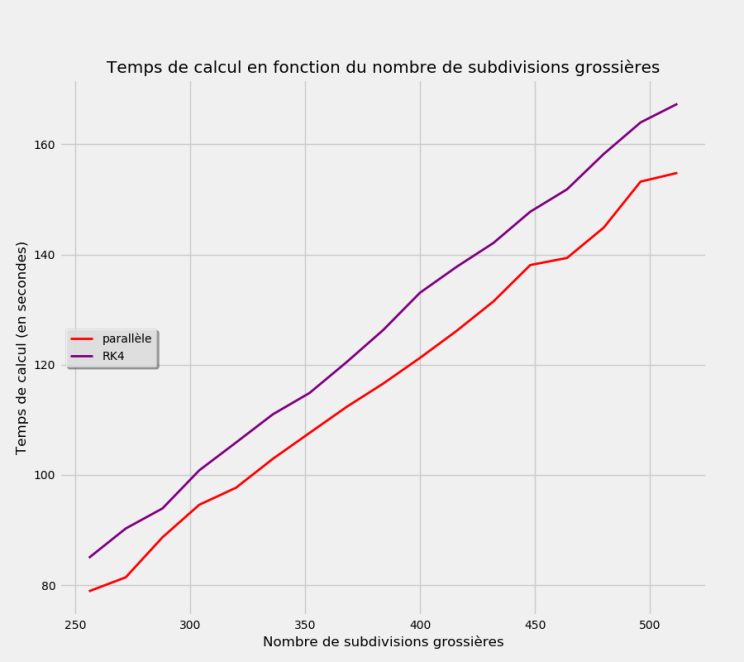
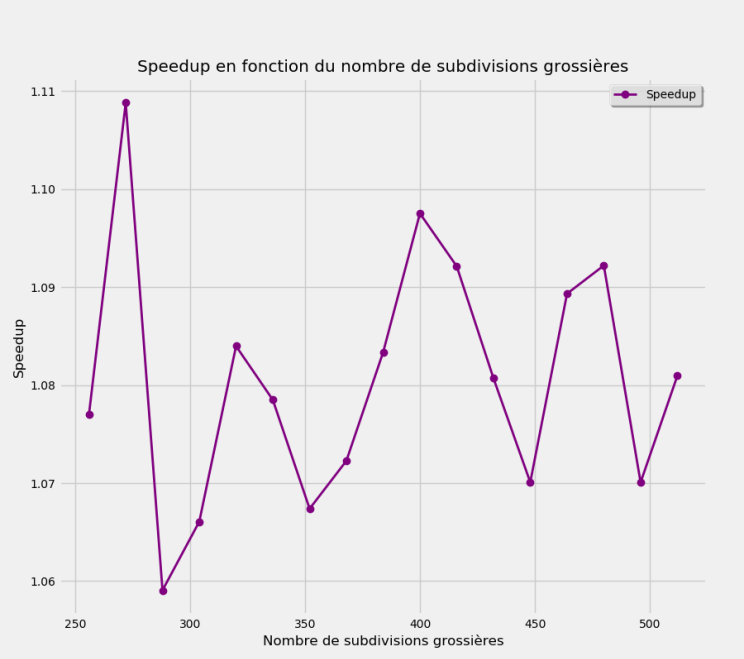


FIGURE 10 – *Représentation du Speedup avec* $NG = 128$ *et* $NF = 10^4$.



(a) *Temps de calcul.*



(b) *Speedup.*

FIGURE 11 – *Représentations pour 3 itérations avec* $NF = 10000$ *et* NG *variant.*