République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ MOHAMED KHIDER, BISKRA

FACULTÉ des SCIENCES EXACTES et des SCIENCES de la NATURE et de la VIE

DÉPARTEMENT DE MATHÉMATIQUES

Mémoire présenté en vue de l'obtention du Diplôme **:**

MASTER en Mathématiques

Option : **ANALYSE**

Par

HASNI MERIEM

Titre :

Résolution d'une Equation Différentielle Ordinaire par Les Algorithmes Méta-heuristiques

Membres du Comité d'Examen :

Dr.	**OUAAR FATIMA**	UMKB	Encadreur
Dr.	**BOUZIANE NADJETTE**	UMKB	Président
Dr.	**GHODJMIS FATIHA**	UMKB	Examinateur

Septembre 2020

Dédicace

Je didie ce humle travail à

Ma chère maman : **Noura Issani.**

Mon chèr papa : **Youcef.**

Mes chères sœurs :

Ines, Djouhaina, Nour El houda, et Aya.

Les petits **Anas** et **Mouadh.**

A toute la famille.

A tous mes amies : **Nissa, Naoual, Mery, Salma, Mani,** ma chère **Karima ...**

A tous mes **collégues**, mes **amies**

A toutes les personnes qui ont contribués de prés ou de loin pour la réalisation de ce travail.

HASNI MERIEM

REMERCIEMENTS

Avant tout, je tiens à remercie "$\mathcal{ALLAH}$" le tout puissant qui m'a donné la santé, le courage et la patience et la volonté pour réaliser ce travail malgré toutes ces conditions critiques.

Je tiens tout d'abord à remercier sincèrement mon encadreur : Dr. **OUAAR FATIMA** pour avoir m'encadrer et diriger, et pour l'effort fournit, ces conseils prodigués et sa patience bien voulu me proposer ce thème.

Je remercie les membres de jury : **BOUZIANE NADJETTE** et **GHODJMIS FATIHA** d'avoir accepté d'examiner et d'évaluer ce travail

Je remercie également, tous nos enseignants (es) du département de Mathématique à l'université de Mohamed khider, qui ont contribués à nos formations pendant les années de Licence et de Master.

Je souhaite particulièrement remercier mes parents qui m'ont stimulés et encouragés pendant mes études,
qui étaient toujours prêts à fournir tous les moyens physiques et morales pour la réussite de ce projet, et qui je n'ai jamais su dire toute l'affection que j'ai pour eux, mon cher **père**, ma chère **mère** et mes chères sœurs,

En fin je prie **Dieu** de nous retirer cette épidémie.

MERCI.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

L'optimisation mathématique ou la programmation mathématique est la sélection d'un élément meilleur (à l'égard de certains critères) de certain ensemble de solutions de rechange disponibles. L'optimisation des problèmes de toutes sortes se posent dans toutes les disciplines quantitatives de l'informatique et de l'ingénierie à la recherche opérationnelle et l'économie, et le développement des méthodes de solution a été d'intérêt pour les mathématiques pendant des siècles.

Dans le cas le plus simple, un problème d'optimisation consiste à maximiser ou minimiser une fonction réelle, en choisissant systématiquement l'entrée des valeurs à partir d'un ensemble autorisé et calculer la valeur de la fonction. La généralisation de la théorie de l' optimisation et les techniques à d'autres formulations constitue une grande partie des mathématiques appliquées. De manière plus générale, l'optimisation consiste notamment à trouver les "meilleures valeurs disponibles" d'une fonction objective donnée une domaine de définition (ou entrées), compris une variété de différents types de fonctions et différents types de domaines objectifs.

Notre choix de la fonction objective est dans le domaine économique, précisément l'équation de l'offre et de la demande qui est généralement résolue à l'aide des méthodes numériques classiques, nous en appliquons la méthode de Runge-Kutta (RK) comme méthode numérique de l'offre et de la demande. Nous l'avons comparé à des méthodes modernes et efficaces, que nous étudierons dans ce travail.

De nombreux modèles de systèmes financiers et économiques impliquent le taux de va-

riation d'une quantité. Il est donc nécessaire d'incorporer des dérivées dans le modèle mathématique. Ces modèles mathématiques sont des exemples d'équations différentielles produites par l'un des domaines les plus intéressants et utiles des mathématiques. Cela nous permet de modéliser de nombreux phénomènes naturels intéressants basés sur le changement en utilisant des équations différentielles.

Face à de nombreux problèmes durs et complexes qui existaient dans le monde réel, les techniques d'optimisation traditionnelles démontrent leur insuffisance, C'est ce qui pousse les chercheurs à exploiter la vaste inspiration de la nature qui résout et optimise leurs problèmes profonds en gardant l'équilibre entre leurs systèmes. Dans ce contexte, les algorithmes méta-heuristiques (MA) exploitent différents processus d'opérateurs inspirés de la nature. Récemment, les techniques de MA sont devenues largement utilisées dans diverses branches de la science et de l'économie.

Cette propagation de MA est due à plusieurs raisons : elle recherche les meilleurs paramètres aléatoires (en utilisant des opérateurs aléatoires). Cela permet de renoncer aux solutions optimales locales, en particulier dans le cas de problèmes réels, ce qui rend MA très flexible pour résoudre différents types de problèmes.

Au cours des dernières décennies, différentes MA ont été proposées pour résoudre des problèmes d'optimisation complexes. Quelques exemples sont : l'optimisation de l'essaim de particules(PSO) [8], les algorithmes génétiques (GA) [6] [1] [2] et le Evolution Differentiel (DE) [16], que l'on pourrait considérer comme MA classique. Dans le même contexte, il a également été proposé de nouvelles alternatives comme la Whale Optimization Algorithme (WOA) [11], l'algorithme de recherche de corbeau (CS) [3], recherche fractale stochastique (SFS) [13] et la colonie d'abeilles artificielle (ABC) [7].

L'importance de cette thèse réside dans la considération d'un Problème à Valeur Initiale (PVI) comme un problème d'optimisation mis en œuvre au moyen d' Algorithme d'Essaim de Salpe noté par la suite par (SSA - selon la traduction en anglais Salp Swarm Algorithm-) afin de trouver des solutions numériques à ce problème, d'où les résultats obtenus sont com-

parés à ceux de (GA) , Runge Kutta d'ordre 4 $(RK4)$ et les résultats exacts de l'exemple étudié. Des comparaisons sont faites en termes de qualité de la solution.

Le travail présenté dans ce mémoire est organisé en trois chapitres :

Chapitre 1 : Le premier chapitre est consacré à considérer les Problèmes à Valeur Initiale (PVI) dans les équations différentielles ordinaires (EDO) comme des problèmes d'optimisation, résolu en utilisant les méthodes de Runge-kutta.

Chapitre 2 : Le deuxième chapitre consiste à étudier deux types d'algorithmes métaheuristiques : l'algorithme d'essaim de salpe (SSA) et l'algorithme génétique (GA).

Chapitre 3 : Le troisième chapitre est consacré à la comparaison de l'efficacité de (SSA), (GA) et $(RK4)$ et c'est grâce à l'étude appliquée par le logiciel Matlab, qui a montrée que (SSA) nous donne presque précis avec des erreurs minimales pour résoudre l'équation de l'offre et de la demande.

Chapitre 1

Problèmes à Valeur Initiale

Les équations différentielles décrivent l'évolution de nombreux phénomènes dans des domaines variés. Le sujet des équations différentielles est l'un des domaines les plus intéressants et utiles des mathématiques. De nombreux phénomènes naturels intéressants qui impliquent des changements peuvent être décrits en utilisant différentes équations.

1.1 Equation Différentielle Ordinaire

Une équation différentielle est une équation impliquant une ou plusieures dérivées d'une fonction inconnue. Si toutes les dérivées sont prises par rapport à une seule variable, on parle d'équation différentielle ordinaire (EDO).

Définition 1.1.1 *Une (EDO) est une relation entre la variable réelle t, et une fonction inconnue $t \longmapsto y(t)$ est exprimée sous forme :*

$$F\left(t, y(t), \acute{y}(t), y^{"}(t), ..., y^{(p)}(t)\right) = 0, \tag{1.1}$$

on notera par :

$$F\left(t, y, \acute{y}, y^{"}, ..., y^{(p)}\right) = 0.$$

Dont les inconnues sont : une ***fonction*** $y : I \subset \mathbb{R} \to \mathbb{R}^n$ *et ses dérivés ; l'intervalle de définition de la fonction* y *est* I; *tel que* $t \in I$.

Dans laquelle cohabitent à la fois y et ses dérivées $\left(\acute{y}, y", ..., y^{(p)}\right)$, ($p$ est appelé l'ordre de l'équation), on pourra utilisé x de temps en temps au lieu de t i.e.($y(t)$ ou $y(x)$).
Si F à valeur dans $\mathbb{R}$, alors cette équation 1.1 est scalaire si non elle dit vectorielle.
Une (EDO) est d'ordre k si elle contient les dérivées de y jusqu'à l'ordre k.

Exemple 1.1.1 *les équations :*
$\acute{y}(t) - t = 0$,
$\acute{y}^2(t) - y(t) = 0$,
$\exp(\acute{y}^2(t)) - t^2 + y(t) = 0$.
Sont des équations différentielles ordinaires.

1.1.1 Equation Différentielle d'Ordre un

C'est une (EDO) tel que $n = 1$,

Définition 1.1.2 *Une équation différentielle* (ED) *du premier ordre est une équation de la forme :*

$$F(t, y, \acute{y}) = 0. \tag{1.2}$$

Où :
y *est la fonction inconnue de la variable* t *(ou* x *) à valeur dans* $\mathbb{R}^n$,*et* $\acute{y}$ *sa dérivée.*
F *est une fonction d'une partie de* $\mathbb{R} * \mathbb{R}^n * \mathbb{R}^n$ *vers* $\mathbb{R}^n$.

Remarque 1.1.1 *Le qualificatif ordinaire pour l'équation 1.2 signifie inconnue* y *dépend d'une seul variable* t. *L'orsqu'il* y *a plusieures dérivées* $\frac{dy}{dt}$ *on parle d'équation aux dérivées partielles* (EDP). *L'équation 1.2 est dite scalaire lorsque* $n = 1$, *si non elle est dite vectorielle.*

Considérons pour simplifier que la fonction $y(t)$ est à valeurs dans $\mathbb{R}$ (i.e $n = 1$), soit D un ouvert de $\mathbb{R} * \mathbb{R}$ et $f : D \to \mathbb{R}$ une fonctin continue, on dit qu'une équation 1.2 est sous forme normée (résolue), si elle s'écrit :

$$\acute{y} = f(t, y); \quad f : D \to \mathbb{R} \quad ; D \subset \mathbb{R} * \mathbb{R}. \tag{1.3}$$

Exemple 1.1.2 *$F(t, y, \acute{y}) = 3\acute{y} + t^2 y - 1 = 0$ une équation différentielle d'ordre un, car l'équation $\acute{y} = -\frac{1}{3}t^2 y + \frac{1}{3}$ est une équation sous forme résolue $\acute{y} = f(t, y) = -\frac{1}{3}t^2 y + \frac{1}{3}$ qui est continue.*

1.2 Equation Différentielle Linéaire de Premier Ordre

Définition 1.2.1 *Une équation différentielle du premier ordre est dit linéaire si elle peut se mettre sous forme :*

$$\acute{y} = f(t)\, y + g(t). \tag{1.4}$$

Où f et g sont des fonctions continues sur un intervalle I de $\mathbb{R}$. Cette équation 1.4 s'appelle une équation linéaire du premier ordre avec le seconde membre ; $g(t)$ est le seconde membre.

Remarque 1.2.1 *Toute équation de la forme 1.4 est dite homogène lorsque le seconde membre est nul $g(t) = 0$, dans le cas où $g(t) \neq 0$ on parle d'une équation non homogène.*

Exemple 1.2.1 *$\acute{y} - ty = 2t^3 + 9$ est une équation différentielle non homogène.*
$\acute{y} = yt^2 - (y - 3)$ est une équation différentielle non linéaire.

Equations différentielles à variables séparées :

Quand on peut isoler y et $\acute{y}$ dans un membre et le reste dans l'autre membre de l'équation différentielle, on a une équation différentielle à variables séparées. En utilise la notation

$\acute{y} = \dfrac{dy}{dt}$, alors on obtenu une équation qui peut se mettre sous la forme $f(t)\,dt = g(y)\,dy$ où f et g sont deux fonctions.

Exemple 1.2.2

$$t + \acute{y} = 3 \Rightarrow t + \frac{dy}{dt} = 3 \Rightarrow \frac{dy}{dt} = 3 - t \Rightarrow dy = (3-t)dt \text{ est séparée .}$$

Exemple 1.2.3

$$t + 3\acute{y} = 2 - y \Rightarrow t + 3\frac{dy}{dt} = 2 - y \Rightarrow 3\frac{dy}{dt} = 2 - y - t \text{ n'est pas séparée .}$$

1.2.1 Types d'une Equation Différentielle d'Ordre Un

On donne quelques types des équations différentielles d'ordre 1 :

Equation Différentielle de type ý=f(t)

Résoudre ce type d'équation différentielle équivant à chercher de primitive de f.

Exemple 1.2.4

$$\begin{aligned} &\acute{y} = 4\cos(2t) + \exp(2t) \Longrightarrow y(t) = \int (4\cos(2t) + \exp(2t))\,dt \ , \\ &\Rightarrow y(t) = 2\sin(2t) + \frac{1}{2}\exp(2t) + c \ ; \qquad c \in \mathbb{R}. \end{aligned}$$

Equation Différentielle de type ý=ay

Théorème 1.2.1 *Soit a un réel fini, soit l'équation différentielle :*

$$\acute{y} = ay. \tag{1.5}$$

L'équation 1.5 admette pour solution sur $\mathbb{R}$ *la famille des fonctions definies par :*

$$h_\lambda(t) = \lambda \exp(at), \qquad \lambda \in \mathbb{R}.$$

et ce sont les seules.

Preuve. On reécrit l'équation différentielle sous la forme :

$$\frac{\acute{y}}{y} = a,$$

et on intègre à gauche et à droite pour trouver :

$$\ln|y(t)| = at + b \quad ; b \in \mathbb{R},$$

on compose par l'exponentielle des deux côtés pour obtenir :

$$|y(t)| = e^{at+b},$$

autrement dit $y(x) = \pm e^b e^{at}$ en posant $\lambda = \pm e^b$ on obtient les solutions (non nulles) cherchées ; $y(t) = \lambda e^{at}$.

$$y(t) = \lambda e^{at}.$$

- L'équation différentielle (1.5) admet donc une infinité de solutions (puisque l'on a une infinité de choix de la constante λ).
- La constante λ peut être nulle. Dans ce cas, on obtient la "solution nulle" : $y = 0$ sur $\mathbb{R}$, qui est une solution évidente de l'équation différentielle.
- Ce théorème peut aussi s'interpréter ainsi : si y_0 est une solution non identiquement nulle de l'équation différentielle (1.5) alors toutes les autres solutions y sont des multiples de y_0. En termes plus savants, l'ensemble des solutions forme un espace vectoriel de dimension 1.

■

Exemple 1.2.5

$$\acute{y} = 2y \Longrightarrow y(t) = \lambda \exp(2t), \qquad \lambda \in \mathbb{R}.$$

$$3\acute{y} - 2y = 0 \Longrightarrow y(t) = \lambda \exp\left(\frac{2}{3}t\right), \qquad \lambda \in \mathbb{R}.$$

Equation Différentielle de type ý=f(t)y

C'est une équation différentielle homogéne à coefficient variable.

Théorème 1.2.2 *Soit f une fonction continue sur un intervalle I de $\mathbb{R}$, soit l'équation différentielle :*

$$\acute{y} = f(t)y. \tag{1.6}$$

Les solutions de l'équation 1.6 sont des fonctions définies sur I par :

$$h_\lambda(t) = \lambda \exp(F(t)), \qquad \lambda \in \mathbb{R}.$$

Où F est une primitive de f sur I et $\lambda \in \mathbb{R}$.

Preuve. On a :

$$\acute{y} = fy,$$

on obtient :

$$\frac{\acute{y}}{y} = f,$$

c'-à-dire

$$\frac{dy}{dt} = fy \Longrightarrow \frac{dy}{y} = f dt,$$

et on intègre l'égalitée pour trouver :

$$\int \frac{dy}{y} = \int f(t)\, dt \Longrightarrow \ln|y| = F(t) + c,$$

on compose par l'exponentielle des deux côtés :

$$|y| = \exp\left(F\left(t\right) + c\right) = \exp\left(F\left(t\right)\right)\exp\left(c\right),$$

on obtient

$$y\left(t\right) = \pm\exp\left(c\right)\exp\left(F\left(t\right)\right),$$

posons : $\lambda = \pm\exp\left(c\right)$.

Alors :

$$y\left(t\right) = \lambda\exp\left(F\left(t\right)\right).$$

■

Exemple 1.2.6

$$\acute{y} = 2t^2 y \Longrightarrow y\left(t\right) = \lambda\exp\left(\int 2t^2 dt\right) \Longrightarrow y\left(t\right) = \lambda\exp\left(\frac{2}{3}t^3\right).$$

Equation Différentielle de type ý=ay+b

C'est une équation différentielle avec seconde membre à coefficient constant.

Théorème 1.2.3 *L'équation différentielle :*

$$\acute{y} = ay + b, \qquad où\ a, b \in \mathbb{R}. \tag{1.7}$$

L'équation différentielle 1.7 admet pour solutions sur $\mathbb{R}$ *la famille des fonctions :*

$$h_\lambda\left(t\right) = \lambda\exp\left(at\right) - \frac{b}{a}, \qquad \lambda \in \mathbb{R}.$$

Et ce sont seules.

Preuve. Nous allons démontrer cette théorie à travers la méthode suivante :

on a l'équation différentielle 1.7 imlique

$$\frac{dy}{dt} = ay + b \Longrightarrow \frac{dy}{ay+b} = dt,$$

on intégre les deux cotés :

$$\int \frac{dy}{ay+b} = \int dt,$$

multiple et dévisé par a :

$$\frac{1}{a}\int \frac{ady}{ay+b} = \int dt,$$

on obtient :

$$\ln|ay+b| = at + ac,$$

on compose par l'exponentielle des deux côtés :

$$|ay+b| = \exp(at+ac) = \exp(at)\exp(ac),$$

on obtient :

$$y(t) = \pm\frac{\exp(ac)}{a}\exp(at) - \frac{b}{a},$$

posons : $\lambda = \pm\frac{\exp(ac)}{a}$ alors :

$$y(t) = \lambda\exp(at) - \frac{b}{a}, \qquad \Big/ \begin{array}{c} \lambda \in \mathbb{R}, b \in \mathbb{R}, \\ a \in \mathbb{R}^* \end{array} .$$

■

Equation Différentielle de type y'=f(t)y+g(t)

Une équation différentielle linéaire du premier ordre avec le second membre (non homogène) est une équation de la forme :

$$y\prime = f(t)y + g(t). \tag{1.8}$$

Où f et g sont deux fonctions continues sur I dans $\mathbb{R}$, Pour résoudre une équation différentielle linéaire d'ordre 1 il n'y a pas de nouvelle formule à apprendre pour ce cas il suffit d'appliquer le principe de superposition : on cherche la solution générale de l'équation homogène y_h associée est :

$$\acute{y} = f(t)y. \tag{1.9}$$

Ensuite on cherche d'une solution particuliére y_p de 1.8.

Finalement :

$$y = y_h + y_p.$$

y est la solution générale de l'equation 1.8.

Recherche d'une solution particuliére y_p de l'équation 1.8 : on a vu que la solution homogène est :

$$y_h(t) = \lambda \exp(F(t)).$$

Où F est la primitive de la fonction f sur I et $\lambda \in \mathbb{R}$, pour trouver une solution particuliére y_p de l'équation 1.8 on utilise la méthode de **variance de la constante (méthode de Lagrange),** on fait comme λ est une fonction et on recherche la fonction définie par :

$$y_p(t) = \lambda(t) \exp(F(t)).$$

Solution de l'équation différentielle 1.8

$$\acute{y_p}(t) = f(t)\lambda(t)e^{F(t)} + \acute{\lambda}(t)e^{F(t)} = f(t)y_p(t) + \acute{\lambda}(t)e^{F(t)},$$

car $\acute{F} = f$

$$\acute{y_p}(t) - f(t)y_p(t) = \acute{\lambda}\,(t)\, e^{F(t)},$$

si on a :

$$\acute{\lambda}\,(t)\, e^{F(t)} = g(t) \Leftrightarrow \acute{\lambda}(t) = g(t)e^{-F(t)} \Longleftrightarrow \lambda(t) = \int g(t)e^{-F(t)},$$

alors y_p est une solution de (1.8).Donc on obtient une solution particulière sur I :

$$y_p(t) = \left(\int g(t)e^{-F(t)}\right) e^{F(t)}.$$

Ce qui ne donne la solution générale de 1.8 est donnée par :

$$y(t) = y_p(t) + \lambda e^{F(t)}; \quad \lambda \in \mathbb{R}.$$

1.3 Problèmes à Valeur Initiale (PVI)

En analyse, un **problème de Cauchy** est un problème constitué d'une équation différentielle dont on recherche une solution vérifiant une certaine condition initiale. Cette condition peut prendre plusieurs formes selon la nature de l'équation différentielle. Pour une condition initiale adaptée à la forme de l'équation différentielle, le théorème de **Cauchy-Lipschitz** assure l'existence et l'unicité d'une solution au **problème de Cauchy**[1]. Dans le cas d'une équation différentielle de premier ordre, de la forme $\acute{y} = f\,(t, y)$ 1.3, la condition initiale adaptée sera la donnée d'une valeur initiale pour la fonction inconnue y, et prendra la forme d'une équation $y(t_0) = y_0$. Les hypothèses du théorème de Cauchy-Lipschitz exigent une certaine régularité de la fonction f.

Voici l'énoncé du théorème de Cauchy-Lipschitz dans le cas des équations différentielles linéaires du premier ordre :

[1]Soit f une fonction de deux valeurs réelles a une valeur réelle $\begin{matrix} f : I_1 * I_2 \to \mathbb{R} \\ (t, y) \mapsto f\,(t, y) \end{matrix}$, ou I_1, I_2 sont deux intervalles de $\mathbb{R}$.

Théorème 1.3.1 (Théorème de Cauchy-Lipschitz) *Soit une équation différentielle linéaire du premier ordre* $y\prime = f(t)y + g(t)$*, où* f*,* $g : I \to \mathbb{R}$ *sont des fonctions continues sur un intervalle ouvert* I*. Alors, pour tout* $t_0 \in I$ *et pour tout* $y_0 \in \mathbb{R}$*, il existe une et une seule solution* y*, telle que* $y(t_0) = y_0$ *.D'après nos calculs précédents cette solution est :*

$$y(t) = \left(\int_{t_0}^{t} g(x) \exp\left(-F(x)\right) dx \right) \exp\left(F\left(x\right)\right) + y_0 \exp\left(F\left(t\right)\right) \Longrightarrow y\left(t_0\right) = y_0,$$

où F *est la primitive de* f *s'annulant en* t_0*, et cette solution vérifie bien* $y(t_0) = y_0$*.*

On appelle problème de Cauchy la donnée d'une équation différentielle et d'une condition initiale :

$$\begin{cases} \dfrac{dy}{dt} = f\left(t, y\right) \\ y\left(t_0\right) = y_0 \end{cases} . \tag{1.10}$$

C'est un problème de condition initiale, dont le théorème de Cauchy-Lipschitz garantit l'existence et l'unicité de la solution si f est lipschitzienne.

1.3.1 Résolution d'une (EDO) par la Méthode de Runge-Kutta

Les techniques de Runge-Kutta sont des schémas numériques à un pas qui permettent de résoudre les équations différentielles ordinaires. Elles font parties des méthodes les plus populaires de part leur facilité de mise en œuvre et leur précision. C'est **Carle Runge** et **Martin Kutta** qui, ont inventé ces méthodes en 1901. Ces méthodes reposent sur le principe de l'itération, c'est-à-dire qu'une première estimation de la solution est utilisée pour calculer une seconde estimation, plus précise, et ainsi de suite.

Le principe générale des méthodes de Runge-kutta : Considérons le problème

suivant :

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

Que l'on va chercher à résoudre en un ensemble discret $t_0 < t_1 < ... < t_n$ .Plutôt que de chercher une méthode directe, les méthodes de Runge-Kutta proposent d'introduire les points intermédiaires $\{(t_{n,i}, y_{n,i})\}_{1 \preceq i \preceq q}$ afin de calculer par récurrence les valeurs (t_n, y_n) avec

$$t_{n,i} = t_n + c_i h_n \quad ,$$

où $h_n = t_{n+1} - t_n$ est le pas de temps et c_i est dans l'intervalle $[0;1]$ Pour chaque point intermédiaire.

On note la pente correspondante :

$$p_{n,i} = f(t_{n,i}, y_{n,i}) \quad ,$$

ainsi, pour une solution exacte y du problème, on a :

$$y(t_{n,i}) = y(t_n) + \int_{t_n}^{t_{n,i}} f(t, y(t))\, dt = y(t_n) + h_n \int_0^{c_i} f(t_n + uh_n, y(t_n + uh_n))\, du \forall i = 1, ..., q.$$
$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))\, dt = y(t_n) + h_n \int_0^{1} f(t_n + uh_n, y(t_n + uh_n))\, du.$$

On calculera ces intégrales par une méthode de quadrature, qu'on peut choisir différentes pour deux valeurs distinctes de i :

$$\int_0^{c_i} g(u)\, du \approx \sum_{k=1}^{i-1} a_{ik} g(c_k) \quad , \int_0^{1} g(u)\, du \approx \sum_{k=1}^{q} b_k g(c_k) ,$$

calculées ici pour $g(u) = f(t_n + u\ h_n, y(t_n + uh_n))$.La méthode de Runge-Kutta d'ordre q sera donc donnée par :

$$\forall i = 1, ..., q, \begin{cases} t_{n,i} = t_n + c_i h_n \\ y_{n,i} = y_n + h_n \sum_{k=1}^{i-1} a_{ik} p_{n,k} \\ p_{n,i} = f\left(t_{n,i}, y_{n,i}\right) \end{cases},$$

$$y_{n+1} = y_n + h_n \sum_{k=1}^{q} b_k p_{n,k} \quad .$$

On résume la méthode souvent par le tableau des différents poids de quadrature, appelé tableau de Butcher :

$$\begin{array}{c|cccc} c_1 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ \vdots & \vdots & & \ddots & \\ c_q & a_{q1} & a_{q2} & \cdots & a_{q,q-1} \\ \hline & b_1 & b_2 & \cdots & b_{q-1} \quad b_q \end{array}$$

FIG. 1.1 – Tableau de Butcher

La méthode est consistante si :$\forall i = 2, ..., q, \sum_{k=1}^{i-1} a_{ik} = c_i$

Méthode de Runge-kutta d'ordre 1 $(RK1)$ **:** Cette méthode est équivalente à la méthode d'Euler, une méthode simple de résolution d'équations différentielles du 1 ordre. Considérons le problème suivant :

$$\begin{cases} \acute{y} = f\left(t, y\right) \\ y\left(t_0\right) = y_0 \end{cases}.$$

La méthode $(RK1)$ utilise l'équation

$$y_{n+1} = y_n + hf\ \left(t_n, y_n\right).$$

Où h est le pas de l'itération. Le problème s'écrit donc :

0	0
	1

FIG. 1.2 – RK-1

Méthode de Runge-kutta d'ordre 2 $(RK2)$ **:** La méthode $(RK2)$ du point milieu est une composition de la méthode d'**Euler** :

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

Où h est le pas de l'itération ; Elle consiste à estimer la dérivée au milieu du pas d'intégration :

$$y_{n+\frac{1}{2}} = y_n + \frac{h}{2} f\left(t_n, y_n\right)$$
$$\acute{y}_{n+\frac{1}{2}} = f\left(t_n + \frac{h}{2}, y_{n+\frac{1}{2}}\right)$$

et à refaire le pas d'intégration complet à partir de cette estimation : $y_{n+1} = y_n = h\acute{y}_{n+\frac{1}{2}}$. Ce schéma est couramment appelé **schéma prédicteur-correcteur explicite**. C'est le cas particulier pour $\alpha = 1/2$ de la méthode plus générale :

0	0	0
α	α	0
	$1 - \frac{1}{2\alpha}$	$\frac{1}{2\alpha}$

FIG. 1.3 – Schéma prédicteur-correcteur explicite

On reconnait ainsi que la méthode de quadrature utilisée pour les temps intermédiaires est celle du point milieu.C'est une méthode d'ordre 2 car l'erreur est de l'ordre de h^3.Un autre cas courant est la méthode de **Heun**, correspondant au cas $\alpha = 1$. La méthode de quadrature repose sur la méthode des **trapèzes**.

La méthode de Runge-Kutta classique d'ordre quatre ($RK4$) : C'est un cas particulier d'usage très fréquent, noté $RK4$. Considérons le problème suivant :

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

La méthode $RK4$ est donnée par l'équation :

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

Où :

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\ k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\ k_4 = f(t_n + h, y_n + hk_3) \end{cases}.$$

L'idée est que la valeur suivante (y_{n+1}) est approchée par la somme de la valeur actuelle (y_n) et du produit de la taille de l'intervalle (h) par la pente estimée. La pente est obtenue par une moyenne pondérée de pentes :

- k_1 est la pente au début de l'intervalle ;
- k_2 est la pente au milieu de l'intervalle, en utilisant la pente k_1 pour calculer la valeur de y au point $t_n + \frac{h}{2}$ par le biais de la méthode d'Euler ;

– k_3 est de nouveau la pente au milieu de l'intervalle, mais obtenue cette fois en utilisant la pente k_2 pour calculer y ;

– k_4 est la pente à la fin de l'intervalle, avec la valeur de y calculée en utilisant k_3.

Dans la moyenne des quatre pentes, un poids plus grand est donné aux pentes au point milieu.

0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
$\frac{1}{2}$	0	$\frac{1}{2}$	0	0
1	0	0	1	0
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

FIG. 1.4 – Schéma de RK-4

La méthode $RK4$ est une méthode d'ordre 4, ce qui signifie que l'erreur commise à chaque étape est de l'ordre de h^5, alors que l'erreur totale accumulée est de l'ordre de h^4. Ces formules sont aussi valables pour des fonctions à valeurs vectorielles.

Chapitre 2

L'Optimisation par les Algorithmes Méta-heuristiques

Dans ce chapitre, nous allons présenter les algorithmes méta-heuristiques qui sont des algorithmes visant à résoudre des problèmes d'optimisation difficiles pour lesquels on ne connaît pas de méthode classique plus efficace, et là nous étudierons deux algorithmes ; le premièr est l'algorithme d'essaim de salpe (SSA), le second est l'algorithme génétique avec ses descriptions formelles.

2.1 Les Algorithmes Méta-heuristiques

2.1.1 Définition d'une Méta-heuristique

Une méta-heuristique est une méthode algorithmique capable de guider et d'orienter le processus de recherche dans un espace de solution, souvent très grand à des régions riches en solutions optimales. Le fait de rendre cette méthode abstraite et plus générique conduite à une vaste utilisation pour des champs d'application différentes.
A ces applications, les méta-heuristiques permettent de trouver des solutions peut-être pas toujours optimales, en tout cas très proches de l'optimum et en un temps raisonnable.

En d'autres termes, une méta-heuristique peut être considérée comme un cadre algorithmique général qui peut être appliqué à différents problèmes d'optimisation avec relativement peu de modifications pour les adapter à un problème spécifique. Il n'existe pas de définition qui fasse l'unanimité, mais tous s'accordent sur les points suivants :

1. Les méta-heuristiques sont des stratégies permettant de guider la recherche d'une solution optimale.
2. Le but visé par les méta-heuristiques est d'explorer l'espace de recherche efficacement afin de déterminer des solutions (presque) optimales.
3. Les méta-heuristiques sont en général non-déterministes et ne donnent aucune garantie d'optimalité.
4. Les méta-heuristiques peuvent contenir des mécanismes qui permettent d'éviter d'être bloqué dans des régions de l'espace de recherche.
5. Les concepts de base des méta-heuristiques peuvent être décrit de manière abstraite, sans faire appel à un problème spécifique.

2.1.2 Catégories des Algorithmes Méta-heuristiques

Les algorithmes méta-heuristiques sont classés en deux classes dominantes : **techniques évolutives** et **intelligence de l'essaim**.

Les algorithmes évolutionnaires : imitent les concepts d'évolution dans la nature. L'algorithme le meilleur et le plus apprécié de cette classe est l'Algorithme Génétique (GA). Cet algorithme simule les concepts de la théorie darwinienne de l'évolution. En GA, l'optimisation est initiée avec un ensemble de solutions aléatoires pour un problème particulier. Après avoir évalué les solutions par la fonction objective, il modifie les variables des solutions basées sur leur valeur de fitness, les meilleures personnes ont une plus grande probabilité d'impliquer dans l'amélioration d'autres solutions, les solutions initiales sont très probables être amélioré.

Les techniques d'intelligence en essaim : ces algorithmes sont fondés sur le comportement collectif d'un groupe de créatures. Par exemple, il est possible de garantir collectivement la survie d'une colonie sans avoir une unité de contrôle centralisée.
Au cours des dernières décennies, différents MA ont été proposées pour résoudre des problèmes d'optimisation complexes. Quelques exemples sont cités selon le tableau 2.1.

Algorithmes	**Inspiration**
Algorithme de chauve-souris	Echolocalisation caractéristique des microbes
Recherche de coucou	Comportement parasite des espèces de coucous
Algorithme Firefly	Clignotant caractéristique des lucioles
Recherche d'harmonie	Performance musicale
Tabou recherche	Mémoire humaine
Optimisation des essaims de particules	Comportement en essaim d'oiseaux et de poissons

TAB. 2.1 – Quelques exemples des algorithmes et son inspiration

2.1.3 Avantages d'Utiliser les Algorithmes Méta-heuristiques

Les méta-heuristiques présentent certains avantages biologiques pour la survie de l'espèce dans son évolution naturelle :

- Flexible : car les agents peuvent être facilement ajoutés ou retirés sans influencer la structure.
- Robuste : les systèmes sont robustes car les agents sont de conception simple et peuvent exécuter des tâches en cas de défaillance de certains agents. La défaillance d'un seul agent a peu d'impact sur les performances du système.
- Evolutif : les systèmes sont évolutifs car la même architecture de contrôle peut être appliquée à quelques agents ou à des millions d'agents.
- Décentralisé : Il n'y a pas de contrôle central dans la colonie.

- Auto-organisé : Les solutions sont émergentes plutôt que prédéfinies.
- Adaptation : Ils sont capables de s'adapter facilement à de nouvelles situations.
- Vitesse : les changements dans le système peuvent être engendrés rapidement.
- Stabilité et adaptabilité : Ils sont utilisés pour ajuster les oscillations écologiques sans changer rapidement de mode, car le changement de mode coûte de la vitalité.
- Qualité : en plus de la capacité de calcul essentielle, un essaim devrait avoir la capacité de réagir à des composants de qualité, par exemple, la nourriture et la sécurité.

2.1.4 Limitations des Méta-heuristiques

La capacité des méta-heuristiques se développe en réalité rapidement et est étendue. Ils offrent une option pour résoudre des problèmes complexes. Cela étant dit, les méta-heuristiques ont encore quelques limitations. Ceux-ci sont :

- Comportement : Ils sont difficiles à prédire le comportement à partir des règles individuelles.
- Connaissances : Les fonctions de la colonie ne pouvaient pas être comprises avec les informations de travail d'un agent.
- Sensibilité : Même un petit changement dans les résultats des principes de base dans la conduite à différents niveaux de rassemblement.
- Action : le comportement de l'agent ressemble à une clameur car l'activité de décision est stochastique.

2.2 Algorithme d'Essaim de Salpe (SSA)

L'(SSA) est un algorithme d'intelligence d'essaim récent basé sur le mécanisme des essaims de salpes et leur interaction sociale lorsqu'ils vivent dans des océans profonds ; il imite leur

comportement qui se sont organisés comme essaims appelés chaînes de salpes. Ce comportement amélioré leur mouvement à travers les forces de l'eau pour chasser leurs proies et de trouver leur nourriture. (SSA) a démontré sa capacité dans différentes applications depuis sa création.

2.2.1 Description de Processus d'une Essaim de Salpe

Algorithme de "salp swarm" (SSA) est un algorithme d'intelligence récemment développé en 2017 par [11]. Ils vivent dans des mers profondes et se déplacent par les courants d'eau pour trouver leur nourriture, les essaims de salpes sont structurés comme des chaînes composées par le chef situé aux chaînes de salpe de tête et les autres salpes de suiveurs. Jusqu'à présent, la conduite d'essaim de salpes n'est pas bien comprise ; ainsi, les analystes pensent que leur conduite est de mise à niveau leur style de vie à la recherche de nourriture.

C'est un tunicier planctonique qui a un corps transparent sous la forme de tonneau et de tissus similaires structure de la gelée, comme les méduses, et ça ce qui est clair dans 2.1.

FIG. 2.1 – L'individu de salpe.

Se déplaçant au fur et à mesure qu'elle se contracte, l'eau pompée à travers son corps gélatineux, qui est l'un des exemples les plus efficaces de propulsion par jet dans le règne animal. Ce salpe pompe l'eau à travers les filtres alimentaires internes et se nourrit de

phytoplancton. Ils vivent dans les profondeurs de l'océan et se déplacent par les forces de l'eau pour trouver leur nourriture, et cela se fait en les organisant en un essaim appelé les chaînes salpides comme il apparaît 2.2. La chaîne des salpes divisées en deux groupes : la tête salpée est un leader et d'autres sont des adeptes, comme ce qui est clair dans 2.3. Jusqu'à présent, le comportement de l'essaim des salpes n'est pas bien exprimé, par conséquent, les chercheurs considèrent le comportement de celui-ci a été fait pour améliorer leur mouvement dans la recherche de nourriture [14].

FIG. 2.2 – L'Essaim de salpes

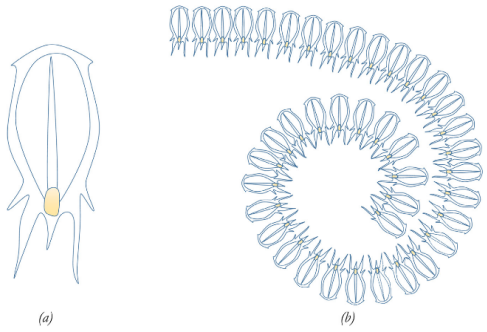

FIG. 2.3 – (a) Individu de Salpe ,(b) Essaim de Salpes (Chaine de Salpes).

2.2.2 Formulation de (SSA)

Le modèle mathématique des chaînes mathématiques des chaînes de salpes, proposé d'abord par [14], consiste à diviser la population en deux groupes : les dirigeants et les adeptes.

La salpe à l'avant de la chaîne est le leader tandis que les salpes attachées restantes sont les suiveurs. Il ressort de l'appellation que les salpes suiveuses sont dirigées directement ou indirectement par le salpe leader (chef).

Chaque salpe est fournie par un vecteur de dimension D, représentant les coordonnées de la salpe position dans un espace de recherche en dimension D. Chaque chaîne est composée d'une population de n salpes cibler une source de nourriture dans l'espace de recherche, indiqué par X^g.

Les positions du groupe le chef de la chaîne de salpes sont mises à jour via l'équation suivante :

$$x_j^1 = \begin{cases} g_j + \omega\left(\left(ub_j - lb_j\right)\epsilon_1 + lb_j\right) & \epsilon_2 \geq 0 \\ g_j - \omega\left(\left(ub_j - lb_j\right)\epsilon_1 + lb_j\right) & \epsilon_2 \prec 0 \end{cases} \quad , j = 1...D, \tag{2.1}$$

où X_j^1 indique le j^{eme} coordonnée de la position de tête de salpe, g_j la coordonnée j^{eme} de la position de la source de nourriture, ub_j et lb_j sont les j^{eme} borne supérieure et borne inférieure respectivement, les $\epsilon_1, \epsilon_2 \in [0, 1]$ sont des nombres aléatoires. Le g_j terme représente la position de la source de nourriture au j^{eme} dimension.

En regardant l'équation 2.1, il est clair que les principales mises à jour de la position de salpe sont basées principalement sur la source de nourriture. Le paramètre ω est utilisé pour contrôler l'équilibre entre l'exploration et stratégie d'exploitation de l'(SSA). Il est mis à jour à chaque itération comme suit :

$$\omega = 2e^{-(4t/L)^2}, \tag{2.2}$$

où t est l'itération actuelle, et L est le nombre maximal d'itérations. Sur la base d'une loi de Newton sur le mouvement, les positions de salpe des suiveurs sont mises à jour à chaque itération en utilisant la formule suivante :

$$x_j^i = \frac{\left(x_j^i - x_j^{i-1}\right)}{2} \qquad , j = 1, ..., D, \tag{2.3}$$

avec $i \geq 2$ et x_j^i indiquant la position du i^{eme} salpe suiveurs en j^{eme} dimension.
Pour que le modèle d'essaim (SSA) converge vers l'optimum global, qui est remplacé par la source de nourriture, le suiveur salpe se déplace vers le chef des salpes tandis que le dernier est attiré par la source de nourriture. Il convient de noter que généralement l'optimum global est inconnu, d'où remplacé par la meilleure solution obtenue jusqu'à présent comme source de nourriture comme illustré dans le pseudo-code dans le tableau 2.2.

N :le nombre d'individus (salpes) dans une même population ;

L :le nombre maximum d'itérations ;

$t = 0$,

initialiser la population$x_i (i = 1, ..., N)$ à partir d'un espace de recherche prédéfini

Calculer $F_i = f(x_i)$ f : fonction fitness

$g = x_i$, $G = opt_{i=1,...,N} Fi$ $opt = \{f \min; \max g\}$, G :meilleure valeur de fitness globale

Etant que $t < L$ Faire

Mettre à jourω en utilise (Eq w)

Pour $i = 1, ..., N$ Faire Calculer leader et les suiveures salpes

Si $i == 1$ ensiute

Autre Mettre à jour la position des leader salpes utilisant (Eq Leader)

Autre

Mettre à jour la position des suivres salpes utilisant (Eq suiveures)

Fin si

Vérifier la faisabilité des nouveaux individus x_i

Fin pour

Calculer $F_i = f(x_i)$

Pour $i = 1, ..., N$ Fair

Si $Bi < G$ Alors

$g = x_i$

$G = B_i$

Fin Si

Fin Pour

$t = t + 1$;

Fin Tant que

TAB. 2.2 – Pseudo code de (SSA)

Commence par générer une population initiale de N positions sélectionnées au hasard dans l'espace de recherche. La fonction de fitness de chaque individu est calculée et la meilleure solution est définie comme source de nourriture. La population est ensuite mise à jour à chaque itération sur la base des équations leader et suiveurs fournies respectivement dans l'équation 2.1 et équation de **??**. Chaque nouvelle position x_i pour tout $i = 1, \cdots, N$ est vérifiée pour la faisabilité en utilisant les limites supérieure et inférieure fournies dans chaque dimension $j = 1, \cdots, D$. Les fonctions objectives des solutions réalisables nouvellement générées sont calculées et comparées à la source de nourriture. Le meilleur minimum obtenu jusqu'à présent est mis à jour en conséquence. Le processus itératif ci-dessus est répété jusqu'à ce qu'un critère d'arrêt soit satisfait ou qu'un nombre maximal d'itérations soit épuisé. Le tableau (2.2) donne le Pseudo-Code de SSA

2.3 Algorithme Génétique

Appelés aussi algorithmes évolutionnaires, les algorithmes génétiques sont des méthodes évolutives qui s'inspirent fortement des mécanismes biologiques liés aux principes de sélection et d'évolution naturelle. Développés initialement par [5] (Holland, 1975) pour répondre à des besoins spécifiques en biologie, les algorithmes génétiques ont rapidement été adaptés à des contextes très variés ils utilisent la notion de sélection naturelle développée par le scientifique **Darwin** et l'appliquent à une population de solutions potentielles au problème donné.

Les Algorithmes Génétiques se basent sur les opérateurs de sélection, de croisement et de mutation. La population initiale est générée aléatoirement. On attribue à chacune des solutions une note qui correspond à son adaptation au problème. Ensuite, une re-sélection est effectuée au sein de cette population, voici un fragment de code 2.3 récapitulant le principe des Algorithmes Génétiques :

Début

Générer une population initiale P_0 (de taille Taille-Population).

Tant que Non-critère d'arrêt

Faire

Début

{Produire une nouvelle génération P_{t+1} à partir de la génération P_t}

Sélectionner les individus selon une stratégie de sélection

Recombiner les individus de la population P_s (avec une probabilité P_s)

Muter la population des enfants (avec une probabilitéP_m) pour donner P_m

$P_{t+1} = P_m$ {la population P_{t+1} remplace la population P_t}

Calculer la Fitness des individus de la population courante

Fin

Fin

TAB. 2.3 – Pseudo code d'(AG)

Les algorithmes génétiques s'inspirent de la théorie de l'évolution et des règles de la génétique qui expliquent la capacité des espèces vivantes à s'adapter à leur environnement par la combinaison des mécanismes suivants :

1 La sélection naturelle fait que les individus les mieux adaptés à l'environnement tendent à survivre plus longtemps et ont donc une plus grande probabilité de se reproduire.

2 La reproduction par croisement fait qu'un individu hérite ses caractéristiques de ses parents, de sorte que le croisement de deux individus bien adaptés à leur environnement aura tendance à créer un nouvel individu bien adapté à l'environnement.

3 La mutation fait que certaines caractéristiques peuvent apparaître ou disparaître de façon aléatoire, permettant ainsi d'introduire de nouvelles capacités d'adaptation à l'environnement, capacités qui pourront se propager grâce aux mécanismes de sélection et de croisement.

4 Les algorithmes génétiques reprennent ces mécanismes pour définir une méta-heuristique. L'idée est de faire évoluer une population de combinaisons, par sélection, croisement et mutation, la capacité d'adaptation d'une combinaison étant ici évaluée par la fonction objective à optimiser. Pseudo-code décrit ce principe général, dont les principales étapes sont détaillées.

Chapitre 3

Optimisation d'un (PVI) par L'Algorithme de (SSA)

Dans ce chapitre on utilise le logiciel matlab pour tester l'efficacité de (SSA) et (GA) à travers une étude de simulation telles que la comparaison entre les résultats exacts et ceux obtenus par la méthode Runge-kutta d'ordre 4 qui est considérée comme un outil classique de résolution numérique d'un (PVI). Les résultats affichés montrent une performance de(SSA) meilleur qu'effectuer de(GA) ça qui est à son tour meilleure que la méthode du Runge-kutta $(RK4)$ en termes de la qualité de la solution. Le (SSA) nous donne une approximation précise et satisfaisante pour la solution avec un minimum des erreurs.

3.1 Problème d'Equilibre Offre-Demande

L'offre est la quantité d'articles vendus ou offerts par le vendeur dans les différents niveaux de prix et de temps. La demande est la quantité du produit souhaité et peut être achetée par les consommateurs à différents niveaux de prix et une certaine période de temps. La loi adoptée en matière d'approvisionnement est lorsque le prix du biens ou du service augmente, la quantité de biens ou de services offerts augmentera, et si le les prix baissent,

la quantité du biens ou services offerts va baisser. Alors que la loi adoptée la demande est lorsque le prix d'un biens ou d'un service augmente, la quantité de biens ou de services demandée diminuera, et lorsque le prix d'un biens ou d'un service baissera, le montant du biens ou services demandés augmenteront.

3.1.1 Le Problème Principal

Il existe des problèmes économétriques fondamentaux qui conduisent à des équations différentielles. Par exemple, compte tenu de l'équilibre offre-demande, on peut considérer la variation de prix comme étant un rapport de la différence entre la demande (D) et l'offre (S) comme suit :

$$\frac{\Delta P}{\Delta t} = a\left(D\left(t\right) - S\left(t\right)\right),$$

où a est une constante positive.

Pour t assez petit, nous obtenons le modèle différentiel :

$$\acute{P}\left(t\right) = a\left[D\left(t\right) - S\left(t\right)\right].$$

Lorsque la demande dépasse l'offre, la partie droite représente une valeur positive, ce qui implique que P est une fonction croissante. Formellement, nous pouvons considérer le problème général de Cauchy comme suit :

$$\begin{cases} \acute{y} = f\left(x, y\right) \\ y\left(x_0\right) = y_0 \end{cases},$$

où x est la variable indépendante et $y = y\left(x\right)$ est la variable dépendante. En utilisant l'hypothèse classique :

$$f : [x_0 - X, x_0 + X] \times [y_0 - Y, y_0 + y] \to \mathbb{R},$$

est continue et satisfait à la condition de Lipschitz :

$$|f(x, y_1) - f(x, y_2)| \leq L |y_1 - y_2| .$$

Il en résulte qu'il existe une seule solution y. Il existe de nombreuses méthodes utilisées pour trouver la solution, mais, dans la pratique, nous résolvons toujours le problème en utilisant des méthodes numériques, comme Runge-Kutta ou les méthodes d'Euler mais ces outils mathématiques classiques ne sont pas très précis. L'objectif principal de ce chapitre vise à souligner la possibilité d'utiliser une méthode différente, basée sur les algorithmes méta-heuristiques comme (SSA).

3.1.2 La Fonction Objective

La formule approximative suivante est utilisée pour le dérivé : où $x_0 = a < x_1 < ... < x_n = b$, $x_i = a + i * h$, et $h = \dfrac{b-a}{n}$

$$\acute{y}(x_i) \approx \frac{y_i - y_{i-1}}{h}$$

$$\left| \acute{y}(x_i) - \frac{y_i - y_{i-1}}{h} \right| \leq const.h$$

Par conséquent, la forme discrète du problème de Cauchy sera :

$$\frac{y_i - y_{i-1}}{h} = f(x_i, y_i), \qquad i = 1, ..., n. \tag{3.1}$$

Le système ci-dessus est généralement non linéaire. Trouver le vecteur $(y_1, y_2, ...y_n)$ qui satisfait la condition ci-dessus est le but de ce chapitre. Bien entendu, pour une solution admissible, nous n'avons pas l'égalité, et par conséquent, la formule d'erreur est considérée :

$$\left(\frac{y_i - y_{i-1}}{h} - f(x_i, y_i) \right)^2 .$$

La fonction objective, associée a un individu $y = (y_1, y_2, ...y_n)$, sera :

$$F(y) = \sum_{i=1}^{n} \left(\frac{y_i - y_{i-1}}{h} - f(x_i, y_i) \right)^2.$$

3.1.3 La Convergence

L'individu le mieux adapté de la population, à l'instance t est désigné par u_t, c'est-à-dire individu dans la population $P(t)$ qui a la valeur minimale de la fonction F. Dans [8] c'est déjà indiqué que la séquence $(u_t)_{t\geq 0}$ converge, sa limite étant la solution de l'optimisation du problème inf F. Alors que la solution est la limite d'une séquence convergente, en appliquant le (SSA) et l'algorithme (GA), l'assertion suivante est vraie :
Pour $\varepsilon > 0$, il y a $(y_1, y_2, ..., y_n)$ tel que :

$$F(y) = \sum_{i=1}^{n} \left(\frac{y_i - y_{i-1}}{h} - f(x_i, y_i) \right)^2 < \varepsilon,$$

il en résulte qu'il y a $y = (y_1, y_2, ..., y_n)$ tel que :

$$\left| \frac{y_i - y_{i-1}}{h} - f(x_i, y_i) \right| < h,$$

en tenant compte d'approximation de la dérivée :

$$\begin{aligned} |\acute{y}(x_i) - f(x_i, y_i)| &\leq \left| \acute{y}(x_i) - \frac{y_i - y_{i-1}}{h} \right| + \left| \frac{y_i - y_{i-1}}{h} - f(x_i, y_i) \right| \\ &< C.h \end{aligned}$$

La dernière relation montre que la valeur finale $y = (y_1, y_2, .., y_n)$ est une solution approximative de problème de Cauchy, pour les petites valeurs de h.

Paramètres	Valeur
Dimension des variables de recherche (D)	10
Nombre total d'itérations	1000
Nombre d'agents de recherche (taille de la population) N	30
Intervalle d'espace de recherche	$[-5, 5]^D$

TAB. 3.1 – Paramètres adoptés par (SSA).

3.2 Etude Expérimentale

Pour illustrer la méthode traitée et pour démontrer son efficacité, le problème économique Demande-Offre est considéré en prenant une taille de pas uniforme h. Dans le tableau (3.2), les résultats de (SSA), (GA), et Range-Kutta d'ordre 4 (notée $RK4$)par rapport aux résultats exacts pour le problème étudié sont présentés bien que le tableau (3.3) présente l'erreur absolue des méthodes utilisés. Pour plus de commodité, les paramètres nécessaires pour générer le (SSA) sont présentés dans le tableau (3.1). Tous les calculs ont été effectués sur un système d'exploitation professionnel MSWindow 2007 dans le compilateur de la version R2007a de l'environnement Matlab sur PC Intel Duo Core 2.20 Ghz.

3.2.1 Paramètres Adoptés par (SSA)

Le tableau (3.1) présente les paramètres necessaires pour générer le (SSA).

3.2.2 Application Numérique

L'algorithme ci-dessus est utilisé pour l'équation :

$$\begin{cases} \acute{y}(t) = 2(t^3 - t^2) \\ y(1) = 0 \end{cases},$$

i	x_i	Résultats Exacts	Résultats de (SSA)	Resultats de (GA)	Résultats de $(RK4)$
0	1.0000	0.0000	0.0000	0.0000	0.0000
1	1.1000	0.0114	0.0115	0.0115	0.0117
2	1.2000	0.0515	0.0518	0.0520	0.0522
3	1.3000	0.1301	0.1310	0.1314	0.1319
4	1.4000	0.2581	0.2592	0.2600	0.2602
5	1.5000	0.4479	0.4493	0.4500	0.4505
6	1.6000	0.7128	0.7147	0.7153	0.7159
7	1.7000	1.0674	1.0696	1.0700	1.0710
8	1.8000	1.5275	1.5302	1.5310	1.5319
9	1.9000	2.1100	2.1132	2.1141	2.1147
10	2.0000	2.8333	2.8369	2.8381	2.8386

TAB. 3.2 – Résultats obtenus par les méthodes de (SSA), (GA) et (RK4) pour d=10.

afin de trouver une solution approximative $y : [1, 2] \to \mathbb{R}$ et $h = \frac{1}{10}$. Les résultats obtenus et la comparaison entre les performances de (SSA), (GA) et Range-Kutta d'ordre 4 $(RK4)$ aux résultats exacts sont présentés dans le tableau suivant :

Les résultats numériques du tableau (3.2) sont illustrés sur la figure (3.1).

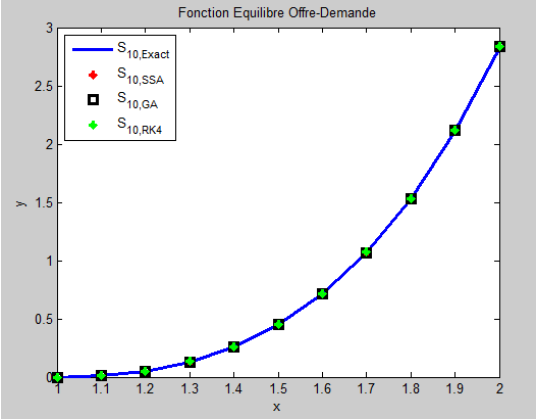

FIG. 3.1 – Plot des solutions numériques des méthodes étudiées pour d=10.

La comparaison entre les performances de (SSA), (GA) et Rnage-Kutta d'ordre 4 face aux résultats exacts confirme que (SSA) est meilleure que (GA) et $(RK4)$ car elle a une courbe très proche de la courbe exacte contrairement aux méthodes (GA) et $(RK4)$.

i	x_i	Err.Abs de (SSA)	Err.Abs de (GA)	Err.Abs de $(RK4)$
0	1.0000	0.0000	0.0000	0.0000
1	1.1000	0.0001	0.0001	0.0003
2	1.2000	0.0003	0.0005	0.0007
3	1.3000	0.0009	0.0013	0.0018
4	1.4000	0.0011	0.0019	0.0021
5	1.5000	0.0014	0.0021	0.0026
6	1.6000	0.0019	0.0025	0.0031
7	1.7000	0.0022	0.0026	0.0036
8	1.8000	0.0027	0.0035	0.0044
9	1.9000	0.0032	0.0041	0.0047
10	2.0000	0.0036	0.0048	0.0053

TAB. 3.3 – L'erreurs absolus obtenus par les méthodes SSA, GA et Range-Kutta pour d=10.

Le tableau (Table 3.3) présente l'erreur absolue entre les résultats exacts et les méthodes étudiées :

La figure (3.2) montre la représentation graphique de l'erreur absolue. Dans les deux représentations (tabulaire et graphique) de l'erreur absolue, la méthode (SSA) offre une erreur absolue très négligeable par rapport à la méthode (GA) et $(RK4)$.

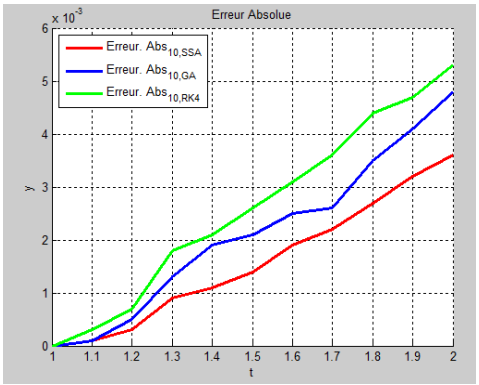

FIG. 3.2 – Plot d'erreur absolue des méthodes étudiées.

3.2.3 Discussions

Les résultats de simulation indiquent que le (SSA) est simple, réduit le temps, flexible et exponentiellement meilleur pour résoudre le modèle différentiel de l'équilibre Offre-Demande, nous avons utilisé le (SSA) en prenant en compte l'équation de l'équilibre Offre-Demande définie par $\acute{y}(t) = 2(t^3 - t^2)$, $y(1) = 0$ pour trouver une solution approximative sur l'intervalle $[1, 2]$ comme exemple d'application qui considère la variation du prix comme étant un rapport de la différence entre la demande et l'offre. Nous avons sélectionné une population de 30 individus, classés par fonction de performance. Pour $h = \frac{1}{10}$, le vecteur final y est : $(0.000, 0.0115, 0.0518, 0.1310, 0.2592, 0.4493, 0.7147, 1.0696, 1.5302, 2.1132, 2.8369)$

Conclusion Générale

Les techniques numériques pour résoudre les équations différentielles telles que les méthodes de Range-Kutta sont entourées par la substitution de la dérivée qui se produit dans l'équation différentielle par des extensions approximatives de la fonction près des points d'intérêt. Evidemment, cette approximation n'est valable que si l'intervalle de temps dans lequel la fonction est suffisamment linéaire, si non la solution devient imprécise.

L'inexactitude de la résolution des problèmes complexes et des situations non linéaires nécessite une augmentation significative des tentatives de simulation des fonctions informatiques basées sur des méthodes plus efficaces.

Parce que les algorithmes méta-heuristiques donnent des résultats prometteurs apparaissant dans de nombreux domaines, notre étude adaptera l'un des algorithmes les plus efficaces et inspirés de la nature pour améliorer (optimiser) un problème mathématique. Lorsque nous choisissons SSA pour optimiser l'PVI dans le domaine économique, exactement l'équation de l'offre et de la demande qui est généralement résolue à l'aide des méthodes numériques classiques, dans ces notes, nous appliquons Runge-kutta ($RK4$) comme méthode numérique pour l'équation de l'offre et de la demande.

Notre motivation vient de la nécessité de trouver des solutions plus précises à ce type de problème en évitant les difficultés de calcul et sans avoir besoin d'outils mathématiques avancés lors de la résolution des problèmes mathématiques complexes, pour ces raisons, le (SSA) semble appropriée. Nous considérons (SSA) comme une alternative à la méthode numérique comparée à GA pour trouver des résultats numériques et optimiser (minimiser)

l'erreur.

En raison du fait que les algorithmes ou méthodes traditionnels ne s'adaptent pas bien pour résoudre de nombreux problèmes dans la pratique, tandis que l'algorithme SSA est très fort et peut produire des solutions robustes à de petits problèmes.

Grâce à l'exemple choisi et après avoir comparé les solutions exactes, les résultats de l'algorithme (SSA), les résultats de (GA), et les résultats de Runge-Kutta d'ordre ($RK4$), nous avons constaté que le SSA était plus en mesure de fournir des solutions précises avec une erreur minimale. Il est important de noter que les résultats actuels concernent principalement l'algorithme SSA standard. Il serait utile que la recherche supplémentaire se concentre sur l'élargissement de la méthodologie proposée pour améliorer d'autres problèmes économiques par le biais d'autres variables de l'ASS.

Bibliographie

[1] Albu L.L., Macroeconomie non-lineara şi prognoză, Academia Română, 2002.

[2] Askarzadeh Alireza., A novel metaheuristic method for solving constrained engineering optimization problems : Crow search algorithm, Computers and Structures, 169(Complete) :1{12, 2016}.

[3] Banzhaf W., Nordon P., Keller R.E., Francone F.D., Genetic Programming – An introduction, Morgan Kaufmann Publishiers, San Francisco 1998.

[4] Baykasoglu Adil., Ozabakir Lale., Tapkan Pinar., "Artificial Bee Colony and its Application to Generalised Assignment Problem", I-Tech Education and Publication, 2007.

[5] Holland,. J. H., Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

[6] Goldberg.David.E., Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[7] Karaboga., An idea based on Honey Bee Swarm for Numerical Optimization, Technical Report TR06, Erciyes University, October 2005.

[8] Kennedy.J and R. Eberhart., Particle swarm optimization, In Neural Networks, 1995. Proceedings, IEEE International Conference on, volume 4, pages 1942{1948 vol.4, Nov 1995}.

[9] Mateescu G.D., Optimization by using evolutionary algorithms with genetic acquisitions, Romanian Journal of Economic Forecasting, No. 2/2005.

[10] Mateescu G.D., On the applications of genetic algorithms to differential equations, Romanian Journal of Economic Forecasting,2/2006.

[11] Mirjalili.at.al Seyedali et Andrew Lewis. The whale optimization algorithm, Adv. Eng. Softw, 95(C) :51{67, May 2016 }.

[12] Munnier.A., Théorie des équations différentielles ordinaires, Institut élie cartan 2006− 2007.

[13] Salimi.Hamid., Stochastic fractal search., Know-Based Syst, 75(C) :1{18, February 2015 }.

[14] Seyedali Mirjalili., Amir H., Gandomi., Seyedeh Zahra Mirjalili., Shahrzad Saremi., Hossam Faris, and Seyed Mohammad Mirjalili., Salp swarm algorithm : A bio-inspired optimizer for engineering design problems, Advances in Engineering Software, 114 : 163 { 191, 2017. ISSN 0965 − 9978. doi : https ://doi.org/10.1016/j.advengsoft.2017.07.002.

[15] Stoer J., Bulirsch R., Introduction to Numerical Analysis, Springer-Verlag, 1992.

[16] Storn Rainer and Kenneth Price., Differential evolution &ndash, a simple and efficient heuristic for global optimization over continuous spaces, Global Optimization, 11(4) : 341{359, December 1997}.

Annexe A : Le logiciel Matlab

Le langage **MATLAB** a été conçu par Cleve Moler à la fin des années 1970 à partir des bibliothèques Fortran,LINPACK et EISPACK. Alors professeur de mathématiques à l'université du Nouveau-Mexique, il souhaitait permettre à ses étudiants de pouvoir utiliser ces deux bibliothèques sans connaitre le Fortran. Cleve Moler l'utilisa ensuite pour des cours donnés à l'université Stanford où il reçut un accueil mitigé de la part des étudiants en mathématiques (habitués au Fortran). Par contre, les étudiants en technologie, en particulier en traitement du signal, furent beaucoup plus intéressés. Un ingénieur, Jack Little en comprend rapidement les capacités et entreprend avec un collègue, Steve Bangert, de le recoder en langage C. Jack Little, Cleve Moler et Steve Bangert créèrent la société The MathWorks en 1984 afin de commercialiser la version 1.0 de MATLAB 5,6.

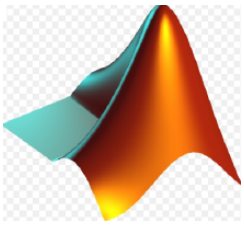

FIG. 3.3 – Icône du logiciel Matlab.

MATLAB a ensuite évolué, en intégrant par exemple la bibliothèque LAPACK en 2000, en se dotant de nombreuses boîtes à outils (Toolbox) et en incluant les possibilités données par d'autres langages de programmation comme $C++$ ou ***Java***.

MATLAB "matrix laboratory" est un langage de script2 émulé par un environnement de développement du même nom ; il est utilisé à des fins de calcul numérique. Développé par la société The MathWorks, MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, $C++$, Java, et Fortran. Les utilisateurs de **MATLAB** (environ 4 millions en 2019) sont de milieux très différents comme l'ingénierie, les sciences et l'économie dans un contexte aussi bien industriel que pour la recherche. Matlab peut s'utiliser seul ou bien avec des toolboxes (« boîte à outils »).

Les toolboxes MATLAB sont développées par des professionnels, rigoureusement testées et entièrement documentées. Les applications MATLAB permet de tester différents algorithmes sur les données. Itéré jusqu'à obtenir les résultats attendus, puis générer automatiquement un programme MATLAB pour reproduire ou automatiser le travail.

Fonction	Le role
Plot	permet de tracer une ou plusieure courbe sur le méme graphe
Legend	sont un moyen utile pour étiqueter les série de donnée tracées sur un graphique
hold on	permet de trecer plusieure courbe sue la figure courante
grid on	met la grille sur le graphe trecé pae plot

TAB. 3.4 – Les foctions les plus utilisées dans ce travail

Principales fonctionnalités :

- Langage de haut niveau pour le calcul scientifique et technique ;
- Environnement bureau pensé pour l'exploration itérative, la conception et la résolution de problèmes ;
- Graphiques destinés à la visualisation de données et outils conçus pour créer des tracés personnalisés ;

– Applications dédiées à l'ajustement de courbes, la classification de données, l'analyse de signaux et bien d'autres tâches spécialisées ;
– Boîtes à outils additionnelles conçues pour répondre à de nombreux besoins spécifiques aux ingénieurs et aux scientifiques.

Annexe B : Abréviations et Notations

Les différentes abbréviations et notations utilisées durant cette thèse est représentées selon ce tableau :

Abréviations	Notations
PVI	Problèmes à valeur initiale.
SSA	Salpe Swarm Algorithme.
PSO	Particle Swarm Optimization.
AG	Algorithme génétique.
HS	Harmony Search.
EDO	Equation Différentielle Ordinaire.
$RK-4$	Range Kutta d'ordre 4.
AM	Algorithmes Méta-heuristiques.
ED	Evolution Différentiel.
WOA	Whale Optimization Algorithme.
SFS	Recherche (Search) fractale stochastique.
SC	Recherche(Search) de carbeau.
CAB	Colonie d'Abeilles (Bee) Artificielle.
$\mathbb{R}$	Ensemble des Nombres Réelles
at al.	Les Autres

Annexe C : Code Matlab des Méthdes Etudiées

3.3 Code Matlab de la Méthode de (RK4)

```
dydt=inline('2.*(y^3-y^2))','t','y');
a=0;
b=1;
n=10;
h=(b-a)/n;
t=a:h: b;
epsilon=0.0001;
u(1)=1+epsilon;
for i=1:n-1
u1(i)=u(i);
u2(i)=u(i)+h/2*dydt(t(i),u1(i));
u3(i)=u(i)+h/2*dydt(t(i)+h/2,u2(i));
u4(i)=u(i)+h*dydt(t(i)+h/2,u3(i));
u(i+1)=u(i)+h/6*(dydt(t(i),u1(i)+2*dydt(t(i)+h/2,u2(i)) +2*dydt(t(i) +h/2,u3(i))
 +dydt(t(i+1),u4(i))))
end
```

```
plot(t(1: end-1),u, ' o-g' )
```

3.4 Code Matlabe de (SSA)

```
%_____________________________________________________________________________
% Salp Swarm Algorithm (SSA) source codes version 1.0
% Developed in MATLAB R2016a
% Author and programmer: Seyedali Mirjalili
% e-Mail: ali.mirjalili@gmail.com
% seyedali.mirjalili@griffithuni.edu.au
% Homepage: http://www.alimirjalili.com
% Main paper:
% S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili,
% Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems
% Advances in Engineering Software
% DOI: http://dx.doi.org/10.1016/j.advengsoft.2017.07.002
%_____________________________________________________________________________
 function [FoodFitness,FoodPosition,Convergence_curve]=SSA(N,Max_iter,lb,ub,dim,fobj)
 if size(ub,1)==1
 ub=ones(dim,1)*ub;
 lb=ones(dim,1)*lb;
 end
 Convergence_curve = zeros(1,Max_iter);
 %Initialize the positions of salps
 SalpPositions=initialization(N,dim,ub,lb);
 FoodPosition=zeros(1,dim);
 FoodFitness=inf;
```

```
%calculate the fitness of initial salps
for i=1:size(SalpPositions,1)
SalpFitness(1,i)=fobj(SalpPositions(i,:));
end
[sorted_salps_fitness,sorted_indexes]=sort(SalpFitness);
for newindex=1:N
Sorted_salps(newindex,:)=SalpPositions(sorted_indexes(newindex),:);
end
FoodPosition=Sorted_salps(1,:);
FoodFitness=sorted_salps_fitness(1);
%Main loop
l=2;
while l<Max_iter+1
c1 = 2*exp(-(4*l/Max_iter)^2); % Eq. (3.2) in the paper
for i=1:size(SalpPositions,1)
SalpPositions= SalpPositions';
if i<=N/2
for j=1:1:dim
c2=rand();
c3=rand();
%%%%%%%%%%%% % Eq. (3.1) in the paper %%%%%%%%%%%%
if c3<0.5
SalpPositions(j,i)=FoodPosition(j)+c1*((ub(j)-lb(j))*c2+lb(j));
else
SalpPositions(j,i)=FoodPosition(j)-c1*((ub(j)-lb(j))*c2+lb(j));
end
%_____________________________________________________________________________
```

```
 end
 elseif i>N/2 && i<N+1
 point1=SalpPositions(:,i-1);
 point2=SalpPositions(:,i);
 SalpPositions(:,i)=(point2+point1)/2; % % Eq. (3.4) in the paper
 end
 SalpPositions= SalpPositions';
 end
 for i=1:size(SalpPositions,1)
 Tp=SalpPositions(i,:)>ub';Tm=SalpPositions(i,:)<lb';SalpPositions(i,:)
=(SalpPositions(i,:).*(~(Tp+Tm)))+ub'.*Tp+lb'.*Tm;
 SalpFitness(1,i)=fobj(SalpPositions(i,:));
 if SalpFitness(1,i)<FoodFitness
 FoodPosition=SalpPositions(i,:);
 FoodFitness=SalpFitness(1,i);
 end
 end
 Convergence_curve(l)=FoodFitness;
 l = l + 1;
end
```