Université Mohamed Khider de Biskra
Faculté des Sciences et de la Technologie
Département de génie électrique

MÉMOIRE DE MASTER

Sciences et Technologies
Électronique
Électronique des systèmes embarqués

Réf. :

Présenté et soutenu par :
SAHRAOUI Meriem

Le : mardi 28 juin 2022

Multiplicateur de Lagrange augmenté chaotique Algorithme d'optimisation Définition et application

---

Jury :

Mme.	FEDIAS Meriem	MCB	Université de Biskra	Président
M.	SBAA Salim	Pr	Université de Biskra	Examinateur
Melle.	TOUMI Abida	Pr	Université de Biskra	Rapporteur
Mme.	NOUREDDINE Samia	Dr	Université de Biskra	Co-encadrant

Année universitaire : 2021-2022

Université Mohamed Khider de Biskra
Faculté des Sciences et de la Technologie
Département de génie électrique

MÉMOIRE DE MASTER

Sciences et Technologies
Électronique
Électronique des systèmes embarqués

Réf. :

Multiplicateur de Lagrange augmenté chaotique Algorithme d'optimisation Définition et application

Le : ……

Présenté par :
SAHRAOUI Meriem

Avis favorable de l'encadreur :
Mme. TOUMI Abida

Signature Avis favorable du Président du Jury

Mme. FEDIAS Meriem

Cachet et signature

Remerciements

Je remercie Allah le tout puissant de de m'avoir donné la volonté, le courage, la santé et la patience durant tous ces longues années d'études.

Je tiens à exprimer toute ma reconnaissance à mon encadrant, **Pr. TOUMI Abida**. Je la remercie de m'avoir encadré, orienté, aidé et conseillé.

Je remercie les professeurs de l'université de Biskra, qui m'ont fourni les outils nécessaires à la réussite de mes études universitaires.
Un grand merci pour : **Dr. NOUREDDINE Samia**.

Je désire aussi remercier les membres de jury : **Mme** la présidente **FEDIAS Meriem** et **Pr. SBAA Salim** pour la lecture et l'évaluation de mon mémoire.

Enfin, tous mes remerciements et mon gratitude vont à mes très chers parents, à ma précieuse famille et mes chers amies pour leurs soutien et encouragement, et toute personne ayant contribué à l'élaboration de ce travail, par un conseil ou même par un sourire.

Résumé

Dans le domaine de la résolution du problème d'optimisation il existe plusieurs méthodes inspirées des phénomènes naturels, appartenant à une famille qui travaille généralement pour offrir la solution la plus approximative et la plus optimale, par un ensemble de conditions sur l'optimisation.

L'objectif de ce projet est de présenter un nouvel algorithme d'optimisation sous contrainte basé sur le chaos CALM.

Pour démontrer l'efficacité de cette méthode, nous avons mené deux expériences, dans la première expérience nous avons appliqué l'algorithme à certaines fonctions de norme, dans la seconde expérience, nous avons appliqué l'algorithme à un vrai problème, la prédiction de séries temporelles qui est un espace ouvert de recherche.

Mots clés : *Optimisation, métaheuristiques, CALM, séries temporelles, prédiction*

Abstract

In the field of solving the optimization problem there are several methods inspired by natural phenomena, belonging to a family that generally works to offer the most approximate and optimal solution, by a set of conditions on optimization.

The objective of this project is to present a new chaos-based constrained optimization algorithm CALM.

To demonstrate the effectiveness of this method, we have conducted two experiments experiments, in the 1st experiment, we applied the algorithm to some norm functions, in the 2nd experiment, we applied the algorithm to a real problem, time series prediction, which is an open search space.

Key words: *Optimization, metaheuristics, CALM, time series, prediction*

Table des matières

Introduction Générale **1**

1 L'optimisation combinatoire **4**

1.1 Introduction 4

1.2 Définition 4

1.3 Formulation mathématique 4

1.4 Définitions et notions de base 5

1.4.1 Définition 1 5

1.4.2 Définition 2 5

1.4.3 Définition 3 6

1.4.4 Définition 4 6

1.4.5 Définition 5 6

1.5 Optimisation déterministe 7

1.5.1 Les méthodes d'interpolation 7

1.5.2 Les méthodes de recherche directe 8

1.5.3 Les méthodes de type gradients 8

1.5.4 L'algorithme de Newton 8

1.6 Optimisation stochastique 8

1.7 Les métaheuristiques 9

1.7.1 Présentation de quelques techniques métaheuristiques 10

1.7.1.1 Recuit simulé(Simulated Annealing) 10

1.7.1.2 La recherche taboue(Tabu Search) 10

1.7.1.3 L'algorithme génétique (Genetic Algorithm (algorithme génétique) (GA)) 10

1.7.1.4 L'algorithme de recherche d'harmonie (Harmony Search (recherche d'harmonie) (HS)) 12

1.8 Algorithme chaotique du multiplicateur de Lagrange augmenté (Chaotic augmented lagrange multiplier (Multiplicateur de Lagrange augmenté chaotique) (CALM)) 12

1.8.1 Contexte théorique 13

1.8.1.1 Principes fondamentaux de l'optimisation chaotique 13

1.8.1.2 Optimisation chaotique 14
1.8.1.3 Méthode du multiplicateur de Lagrange augmenté 15
1.8.2 Description de l'algorithme CALM 17
1.9 Conclusion 21

2 Les séries temporelles **24**
Introduction 24
2.1 Définition série temporelle (TS) 25
2.1.1 Exemples 25
2.2 Domaines d'application 26
2.3 Caractéristiques des séries temporelles 27
2.4 Analyse des séries temporelles 30
2.5 Processus stochastiques dans les séries temporelles 31
2.6 Méthodes de prévision des séries chronologiques 32
2.6.1 Méthodes statistiques 32
2.6.2 Technique des k plus proches voisins (k-NN) 33
2.6.3 Machines à vecteurs de support (SVM) 34
2.6.4 Réseaux neuronaux artificiels (NNs) 36
2.7 Travaux à base de la prédiction des séries temporelles 36
2.8 Conclusion 37

3 Évaluation et contribution de l'approche proposée **41**
3.1 Introduction 41
Introduction 41
3.2 La première expérimentation 41
3.2.1 Tableau des expressions des fonctions de test utilisées 43
3.2.2 Trace 3D de chaque fonction 45
3.2.3 Tableau comparatif des résultats 50
3.2.4 Courbes de Fitness 53
3.2.5 Analyse de variation ANOVA 56
3.3 La deuxième expérimentation 59
3.3.1 La première application 59
3.3.2 La deuxième application 66
3.4 Conclusion 68

Conclusion Générale **71**

Liste des figures

1.1 Les minima et maxima locaux et globaux d'une fonction 5
1.2 Courbe représentant les optimums locaux et les optimums globaux. 7
1.3 Optimum global par rapport à optimum local. [1] 7
1.4 Organigramme des principales étapes d'un GA. [1] 11
1.5 a Séries temporelles et b Espace d'état des séries temporelles. Séries temporelles d'un système chaotique et aléatoire, les séquences chaotiques peuvent être générées en utilisant la carte logistique $x_{n+1} = \lambda x_n(1 - x_n)$, avec $\lambda = 3.19$ et la condition initiale $x_0 = 0.01$. La série temporelle est vraiment similaire aux séries temporelles générées par des nombres normaux aléatoires de moyenne zéro et de variance 1. L'espace d'état des deux systèmes qui représente la différence entre x_n et x_{n+1} [2] 13
1.6 Exemples de cartes chaotiques 1-D [2] 14
1.7 Architecture générale de l'algorithme proposé . [2] 18

2.1 Taches solaires 25
2.2 Ventes de voitures 26
2.3 Trafic aérien 26
2.4 Tendance linéaire [3] 27
2.5 Variation saisonnière [3] 28
2.6 Variation cyclique [3] 28
2.7 Variation aléatoire [3] 29
2.8 Séparation des points de données par SVM [3] 35

3.1 Organigramme de l'algorithme CALM 42
3.3 Liste des figures des courbes 3D pour 23 fonctions [4] 50
3.14 Liste des figures des courbes de Fitness de 23 fonctions de test pour 7 méthodes appliquées. 56
3.25 Variation Anova pour 23 fonctions de test pour 7 méthodes appliquées. 59
3.26 Différences entre les inscriptions réelles et les inscriptions prévues pour l'UA sur la base de la méthode CALM. 62
3.27 Différences entre les inscriptions réelles et les inscriptions prévues pour l'UA sur la base de la méthode PSO. 65

3.28 Différence entre les erreurs de la prévision des inscriptions de l'UA obtenus par les algorithmes CALM et PSO. 65
3.29 l'organigramme du programme utilisée pour convertir la prédiction en optimisation 66
3.30 la convergence de la méthode utilisée vers les valeurs actuelles et la variation de l'erreur. 67

Liste des tableaux

2.1 Travaux à base de la prédiction des séries temporelles 37

3.1 Tableau des fonctions de test 44
3.2 Résultats d'évaluation des méthodes 51
3.3 Inscriptions réelles, Inscriptions prévues par CALM et l'erreur 60
3.4 Une comparaison des résultats de la prévision des inscriptions de l'UA pour les algorithmes CALM et PSO. 63

Liste des algorithmes

1 Mise à jour de $\overrightarrow{\mu} = (\mu_1, \mu_2, ..., \mu_m)$ [2] 18
2 FCW algorithme [5] 20

Liste des abréviations

ABC	Artificial Bee Colony Algorithm (L'algorithme de la colonie d'abeilles)
AR	autoregressive
ARCH	auto regressive conditional heteroscedasticity
ARIMA	Autoregressive Integrated Moving Average
ARMA	autoregressive–moving-average
CA	Cultural Algorithm (L'algorithme culturel
CALM	Chaotic augmented lagrange multiplier (Multiplicateur de Lagrange augmenté chaotique)
CI	computational intelligence
FCW	First carrier wave (première onde porteuse)
GA	Genetic Algorithm (algorithme génétique)
GARCH	generalized auto regressive conditional heteroscedasticity
GWO	Grey Wolf Optimisation (L'optimisation du loup gris)
HS	Harmony Search (recherche d'harmonie)
IAP	The International Airline Passengers (Les passagers des compagnies aériennes internationales)
KKT	Karush-Kuhn-Tucker
k-NN	k-Nearest Neighbours
MA	Moving Average
NAR	non-linear autoregressive model
NN	artificial neural network (Réseau neuronal artificiel)

PSO	Particle Swarm Optimization (optimisation par essaim de particules)
RNN	Recurrent Neural Network (Réseau neuronal récurrent)
SARIMA	Seasonal Autoregressive Integrated Moving Average
SCA	Sine Cosine Algorithm (L'algorithme de Sine Cosine)
SOS	Symbiotic Organisms Search (Algorithme de recherche par les organismes symbiotiques)
SVM	support vector machine
SVR	support vector regression
TAR	threshold autoregressive model
TS	time series (série temporelles

Introduction Générale

Contexte

De nos jours, les chercheurs et les ingénieurs sont souvent face à des problèmes complexes. L'optimisation combinatoire définit un cadre formel pour de nombreux problèmes de l'industrie, de la finance ou de la vie quotidienne. Les problèmes d'optimisation combinatoire sont habituellement définis comme une problématique de choix d'une meilleure alternative dans un ensemble très grand mais fini d'alternatives. En raison du très grand nombre d'alternatives pour ces problèmes, l'ensemble des alternatives, dit aussi ensemble de solutions réalisables, est défini en compréhension, en d'autres termes les solutions réalisables se distinguent par un ensemble de propriétés ou de conditions, dites aussi contraintes, qu'elles doivent toutes remplir. Une évaluation est associée à toute solution réalisable à l'aide d'une fonction dite fonction objectif. Résoudre un tel problème consiste donc à trouver une solution optimale, c'est-à-dire trouver une solution réalisable qui minimise ou maximise, selon le contexte, la fonction objectif.

La résolution d'un problème d'optimisation nécessite l'utilisation d'un procédé algorithmique permettant la maximisation ou la minimisation d'une ou de plusieurs fonctions « objectif » en respectant les contraintes posées par le problème. Le nombre de fonctions « objectif » du problème à traiter permet de le classer en problème mono objectif ou problème multi objectifs. En fait, un problème mono objectif consiste à optimiser une seule fonction « objectif ». Tandis qu'un problème multi objectifs consiste à optimiser deux ou plusieurs fonctions « objectif ».

Problématique

Pour certains problèmes, les méthodes de résolution de problèmes ont été classées en deux catégories : les méthodes exactes et les méthodes approchées. Une méthode exacte garantisse l'optimalité de la solution mais elle demande des coûts de recherche (temps de calcule et espace mémoire) souvent prohibitifs qui augmentent avec la taille de l'instance du problème traité. Une méthode approchée ne garantisse pas l'optimalité de la solution mais les coûts de recherche qu'elle nécessite sont raisonnables. La méthode la plus appropriée sera généralement une métaheuristique (une technique d'optimisation approximative, permet l'obtention des résultats d'optimisation sa-

tisfaisants dans un temps raisonnable).Les métaheuristiques se basent sur la minimisation d'une fonction objectif, et sur des mécanismes aléatoires pour explorer l'espace de recherche, afin de déterminer ou d'approcher un optimum global.

Objectif

Dans ce mémoire nous avons étudié la technique d'optimisation, puis nous l'avons implémenté sous logiciel MATLAB, ensuite nous avons évalué cette technique avec des fonctions de test populaires, une analyse d'influence des paramètres de CALM est procurée enfin nous avons appliqué cette méthode sur un problème.

Structure du mémoire

Le premier chapitre : on présente un aperçu sur l'optimisation, ensuite un regroupement des méthodes de résolution de problème d'optimisation est donné. Au final, un algorithme totalement unique, (CALM) est présenté pour résoudre les problèmes d'optimisation globale.

Le deuxième chapitre : Ce chapitre est la présentation des séries temporelles, les domaines d'application, les caractéristiques des séries temporelles, les modèles qui peut être utilisés pour calculer la probabilité de futur comportement entre deux limites spécifiées, et la méthodologie de construction des modèles.

Le troisième chapitre : Dans la première expérimentation, nous allons tester l'algorithme CALM sur des fonctions de norme et comparer les résultats avec d'autre méthodes.
Dans la deuxième expérimentation, on a présenté l'application de CALM dans le domaine de la prédiction des séries chronologiques en utilisant deux bases de données publiques de différentes complexités.

Le mémoire se termine par une conclusion et des perspectives.

Chapitre 1

L’optimisation combinatoire

Chapitre 1

L'optimisation combinatoire

1.1 Introduction

"Le désir humain de perfection trouve son expression dans la théorie de l'optimisation. Elle étudie comment décrire et atteindre ce qui est meilleur, une fois que l'on connaît comment mesurer et modifier ce qui est bon et ce qui est mauvais . . . la théorie de l'optimisation comprend l'étude quantitative des optimums et les méthodes pour les trouver " [6].

À partir de la citation ci-dessus, on peut comprendre la motivation du processus d'optimisation. En fait, l'optimisation tente d'améliorer les performances en se rapprochant d'un point optimal.

1.2 Définition

Un problème d'optimisation consiste à chercher une instanciation d'un ensemble de variables soumises à des contraintes ou pas, de façon à maximiser ou minimiser un critère.. [7]

Un problème d'optimisation, au sens général, consiste à déterminer une solution x qui appartient à un espace de recherche X . Cette solution minimise ou maximise une fonction objectif f . De plus, un problème d'optimisation peut présenter des contraintes d'égalité et/ou d'inégalité sur les solutions candidates ; généralement ce sont des conditions supplémentaires qui limitent l'espace de recherche [8].

1.3 Formulation mathématique

Un problème d'optimisation peut être décrit comme suit :
c'est rechercher $x \epsilon X \ et \ x = [x_1, x_2, x_3, ..., x_n]$ qui, selon le contexte, maximise ou minimise la fonction à optimiser $f(x)$.

Sous les contraintes :

$$g_j(x)\ j = 1, ..., J \tag{1.1}$$

Où les variables $x = [x_1, x_2, x_3, ..., x_n]$ Sont appelées "variables d'optimisation"

La fonction f est la fonction objectif (appelée aussi de fitness), elle est définie selon le problème posé, et J est le nombre de contraintes .

La résolution d'un problème d'optimisation revient alors à trouver les points de minimum ou maximum local (ou global) de la fonction . La figure 1.1 donne un exemple des minima et maxima locaux et globaux.

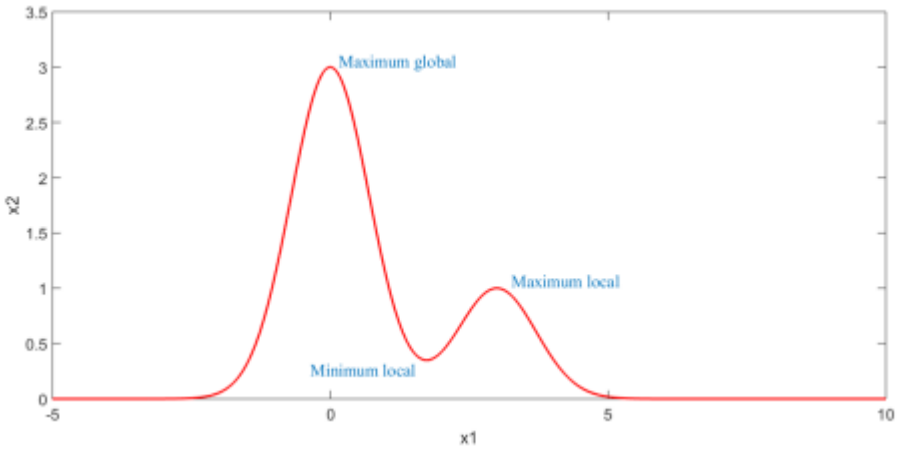


FIGURE 1.1 – Les minima et maxima locaux et globaux d'une fonction

1.4 Définitions et notions de base

Dans cette section nous décrirons les définitions les plus importantes .

1.4.1 Définition 1

Un problème d'optimisation combinatoire est défini par l'ensemble S des solutions possibles d'un problème. $X \subseteq S$ est l'ensemble des solutions réalisables. $f : X \to R$,une fonction appelée fonction «objectif». La résolution du problème consiste à minimiser ou maximiser $f(s)$ $(s \epsilon X)$.

1.4.2 Définition 2

L'espace de recherche S de problème est constitué d'un ensemble de valeurs,ceci peut être obtenu à partir des variables $(s_1, s_2, s_3, ..., s_n)$ qui construisent la solutions .

1.4.3 Définition 3

La solution à un problème d'optimisation est souvent un ensemble de quantités, à partir duquel la valeur est sélectionnée. Toutes ces valeurs sont généralement regroupés en vecteur qui représente la solution. Supposant un problème de taille n, le vecteur représentant la solution s est représenté par :

$$\overrightarrow{s}[s_1, s_2, s_3, ..., s_n] \tag{1.2}$$

1.4.4 Définition 4

Une contrainte d'un problème est une limitation imposée par la nature et les caractéristiques du problème sur les résultats proposées .

1.4.5 Définition 5

un optimum local d'un problème d'optimisation est une solution qui est optimale (soit maximale ou minimale) dans un ensemble voisin $V(s)$ de la solution s . Une solution s' (appartenant à S) est un optimum local du voisinage $V(s)$ de la solution s si elle vérifie la condition suivante :

$$f(s') \leq f(s) \; \forall \epsilon \; V(s) \tag{1.3}$$

Pour un problème de maximisation la condition précédente est remplacée par la condition :

$$f(s') \geq f(s) \; \forall \epsilon \; V(s) \tag{1.4}$$

Cela contraste avec l'optimum global, qui est la solution optimale parmi toutes les solutions possibles, et pas seulement celles qui se trouvent dans un voisinage particulier de valeurs .
Une solution s^* est dite optimale (ou optimum global) si les deux contraintes suivantes sont vérifiées :

• Elle est réalisable : tirée de l'ensemble de solutions possibles (l'espace de recherche S) et respecte toutes les restrictions du problème posé.

• $\begin{cases} f(s^*) = max_{s \epsilon X}\{f(s)\} \; En \; cas \; de \; problème \; de \; maximisation \\ f(s^*) = min_{s \epsilon X}\{f(s)\} \; En \; cas \; de \; problème \; de \; minimisation \end{cases}$

Où X est l'ensemble de solutions réalisables.
Ainsi, une solution est dite optimum global si :

• $\begin{cases} f(s^*) \leq f(s) \; \forall \epsilon \; E(s) \; En \; cas \; de \; problème \; de \; maximisation \\ f(s^*) \geq f(s) \; \forall \epsilon \; E(s) \; En \; cas \; de \; problème \; de \; minimisation \end{cases}$

Où $E(s)$ est l'ensemble d'optimums locaux.Il est à noter que l'optimum global est aussi nommé maximum global au cas de problème de maximisation et minimum global au cas de problème de minimisation.
En d'autres termes,l'optimum global est le meilleur optimum local.

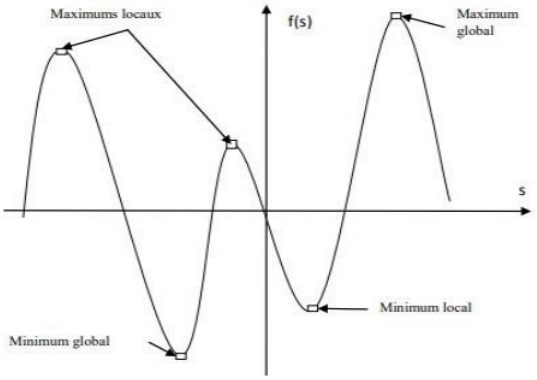


FIGURE 1.2 – Courbe représentant les optimums locaux et les optimums globaux.

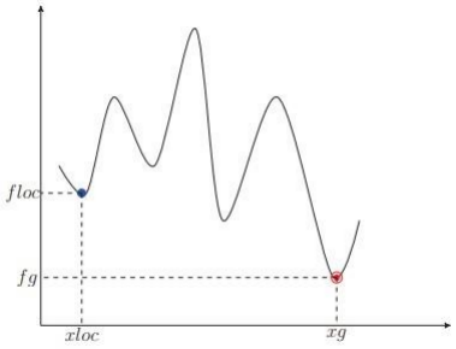


FIGURE 1.3 – Optimum global par rapport à optimum local. [1]

1.5 Optimisation déterministe

Ces méthodes ont d'abord été introduites pour résoudre avec précision des problèmes spécifiques, tels que les problèmes continus et linéaires sous contraintes linéaires (algorithme du simplexe de Danzig) ; ces méthodes ont également été étendues aux cas discrets et mixtes, mais uniquement dans les cas linéaires. Contrairement aux méthodes aléatoires, ces méthodes peuvent bien gérer les contraintes et peuvent être appliquées à des problèmes mixtes (nombres réels, entiers et variables catégorielles). Ils s'assurent d'obtenir une solution globale au problème. Cependant, il faut savoir que tant que le nombre de variables ne devient pas trop important, des méthodes déterministes globales sont toujours disponibles : [1]

1.5.1 Les méthodes d'interpolation

Elles sont utiles lorsque la fonction objective est non définie en certains points de l'espace. Cette famille est composée de méthodes tentant de construire un modèle de la fonction à optimiser. Ce modèle est souvent un polynôme interpolant f construit à partir d'un échantillonnage de points où f a déjà été évaluée. Le modèle est ensuite lui même optimisé. Il existe des méthodes qui tentent d'approcher les gradients de f et d'utiliser cette information afin de déterminer quels sont les points prometteurs où f est évaluée. [9]

1.5.2 Les méthodes de recherche directe

Le terme recherche directe est utilisé pour décrire une méthode qui utilise une variété d'essais de solutions possibles en combinaison avec des stratégies pour comparer chaque essai avec les meilleurs résultats obtenu jusqu'à présent et déterminer la solution à tester ensuite.

Cette famille de méthodes ne tente pas d'approcher les gradients ni de construire un modèle de f, des approximations de gradients sont utilisées pour ordonner les directions de recherche, elle se base principalement sur le calcul de la fonction objectif. [1]

1.5.3 Les méthodes de type gradients

Elles utilisent les gradients et/ou les Hessiens de f (ou leurs approximations) pour choisir des d_k et λ_k appropriés, de façon que la direction choisie soit une direction de descente.

Pour calculer d_k et λ_k ,l'algorithme peut utiliser des informations sur les valeurs du gradient (voire du Hessien) de f au point x_k courant. Le choix de la direction de descente conduit à attacher à l'algorithme général des noms particuliers. [1]

1.5.4 L'algorithme de Newton

Il s'agit d'une généralisation de la méthode de Newton-Raphson dans le cas monodimensionnel, c'est un algorithme efficace pour trouver des approximations d'un zéro (ou racine) d'une fonction d'une variable réelle à valeurs réelles. L'algorithme consiste à linéariser une fonction f en un point et à prendre le point d'annulation de cette linéarisation comme approximation du zéro recherché. Appliqué à la dérivée d'une fonction, cet algorithme permet d'obtenir une évaluation des points critiques. [1]

1.6 Optimisation stochastique

Les méthodes stochastiques, contrairement à la plupart des méthodes déterministes, ne nécessitent ni point de départ, ni à la connaissance du gradient de la fonction objectif pour atteindre la solution optimale. Elles s'appuient sur des mécanismes de transition probabilistes et aléatoires qui explorent efficacement l'espace de recherche et convergent vers l'optimum global. Leur nature aléatoire implique que plusieurs exécutions successives de ces méthodes conduisent à des résultats différents pour une même initialisation du problème d'optimisation. Cependant, elles demandent un nombre important d'évaluations de la fonction objectif en comparaison avec les méthodes déterministes exploitant la dérivée de la fonction objectif [10].

L'optimisation stochastique n'est pas une série de problèmes, de modèles et d'outils. Différent des autres domaines d'optimisation (programmation linéaire, non linéaire, dynamique) ; au contraire, lorsque la notion d'incertitude émerge, elle complète ces familles. Dans ce cas, On va discuter de la programmation stochastique linéaire et non linéaire ou la programmation aléatoire dynamique. En effet, le but Correction du problème dans l'optimisation stochastique était de prendre de meilleures décisions lorsque l'incertitude est impliquée. La littérature autour de la programmation aléatoire est de plus en plus abondante, et les travaux les plus importants dans ce domaine sont ceux. [1]

1.7 Les métaheuristiques

Le mot métaheuristique est dérivé de la composition de deux mots grecs : méta, signifiant « à un plus haut niveau »et heuristique. Ce terme a été mentionné pour la première fois par Fred Glover [11]. Les algorithmes métaheuristiques sont des algorithmes d'optimisation stochastique conçus pour résoudre des problèmes d'optimisation difficiles. En raison de leur simplicité et de leur robustesse, ils ont été utilisés avec succès dans diverses applications [**?**]. Généralement, toutes les méta-heuristiques ont des caractéristiques communes lequel : [12], [13], [1]

• Elles commencent par un tirage aléatoire d'un nombre de solutions candidates, ensuite, des processus de recherche aléatoire sont utilisés pour manipuler ces solutions et les faire passer de solutions de mauvaise qualité à la solution optimale.

• Les métaheuristiques sont des algorithmes itératifs, où le même schéma de recherche est répété plusieurs fois au cours de l'optimisation.

• Elles n'utilisent pas l'information du gradient de la fonction objectif.

• Elles sont capables de résoudre des problèmes d'optimisation difficiles à partir d'un nombre minimal d'information.

• Elles peuvent trouver une solution de très bonne qualité en un temps raisonnable sans garantie l'optimalité.

• La nature stochastique des métaheuristiques permet d'éviter les optima locaux.

• Elles sont adaptables à un grand nombre de problèmes d'optimisation.

• Souvent, elles sont inspirées par des phénomènes de la nature.

1.7.1 Présentation de quelques techniques métaheuristiques

Dans cette partie, nous donnons le principe général de quelques méthodes métaheuristiques.

1.7.1.1 Recuit simulé(Simulated Annealing)

L'origine de la méthode du recuit simulé provient de la métallurgie, où, pour atteindre les états de basse énergie d'un solide, on chauffe celui ci jusqu'à des températures très élevées, après on le laisse refroidir lentement. Ce processus est appelé le recuit. La méthode du recuit simulé a été développée simultanément par Kirk en 1983 et Cerny en 1985. Cette méthode repose sur l'algorithme de Metropolis en 1953.

Cet algorithme nous permet de sortir des minima locaux avec une probabilité élevée si la température T est élevée, et de conserver les états les plus probables pour des très basses températures. L'algorithme de Metropolis permet d'échantillonner la fonction objective par le biais d'une distribution de Boltzmann de paramètre T . Pour mieux comprendre la méthode du recuit simulé, il y a plusieurs notions à définir telles que la probabilité de Boltzmann et le critère de Metropolis. [12]

1.7.1.2 La recherche taboue(Tabu Search)

La méthode de recherche taboue a été proposée par Fred Glover. L'idée principale de cette technique est d'explorer le voisinage d'un lieu donné en effectuant un mouvement qui n'améliore pas forcément la solution. Afin d'éviter de revenir en arrière, il utilise le principe de la mémoire pour enregistrer les solutions interdites, ce qu'on appelle la « liste taboue ». Cette liste est mise à jour de manière itérative pour faire passer l'algorithme de l'exploration à l'utilisation [11].

Comme la méthode du recuit simulé, la méthode de recherche Tabou fonctionne avec une seule configuration courante, qui est actualisée au cours des itérations successives. La nouveauté ici est que, pour éviter le risque de retour à une configuration déjà visitée, on tient à jour une liste de mouvements interdits, appelée liste tabou. Cette liste contient m mouvements ($t \rightarrow s$) qui sont les inverses des m derniers mouvements ($s \rightarrow t$) effectués. La méthode modélise ainsi une forme primaire de mémoire à court terme. Dans sa forme de base, la méthode de recherche taboue présente l'avantage de comporter moins de paramètres que la méthode de recuit simulé. Cependant, la méthode n'étant pas toujours performante, il est souvent approprié de lui ajouter des processus d'intensification et/ou de diversification, qui introduisent de nouveaux paramètres de contrôle. Dans la recherche taboue avec des variables continues, deux concepts sont mis en jeu : le voisinage d'un point donné et un mouvement aléatoire dans le voisinage [11].

1.7.1.3 L'algorithme génétique (GA)

Les GA sont des algorithmes qui s'appuient sur des techniques dérivées de la génétique et de l'évolution naturelle : sélection, croisement,mutation [14].

Ces algorithmes disposent de trois opérations principales : [14]

• L'opérateur de sélection pour déterminer les meilleures solutions appelées aussi parents, qui sont utilisées pour engendrer la nouvelle génération. Il y a différentes stratégies de sélection comme la sélection par tirage à la roulette (roulette-wheel selection), la sélection par rang (ranking selection), la sélection par tournoi (tournament selection)...

• L'opérateur de croisement qui combine les caractéristiques des parents (préalablement sélectionnés) pour générer les enfants.

• L'opérateur de mutation qui modifie aléatoirement une partie de l'individu, ce qui permet de maintenir une certaine diversité dans la population.

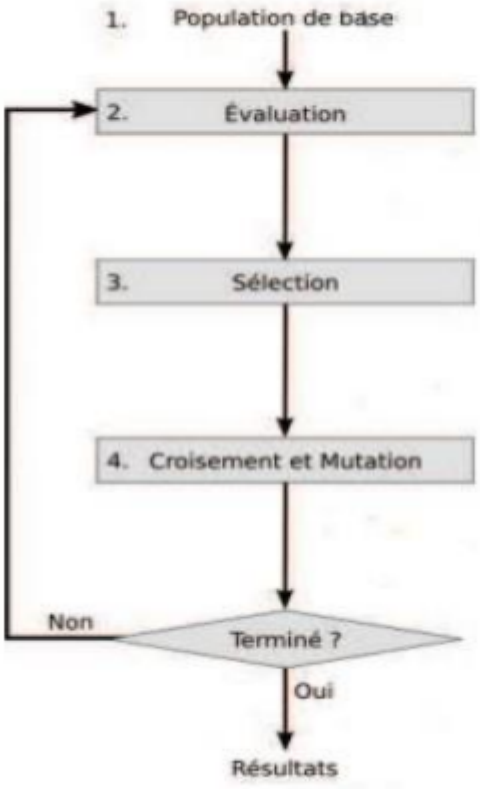


FIGURE 1.4 – Organigramme des principales étapes d'un GA. [1]

1.7.1.4 L'algorithme de recherche d'harmonie (HS)

L'algorithme de recherche harmonique a été développé par Geem et al. Ceci est basé sur le processus de performance musicale consistant à trouver l'harmonie parfaite dans l'orchestre en 2001, et chaque musicien joue une note pour trouver la meilleure harmonie. De même, chaque variable de décision dans le processus d'optimisation à la valeur trouver la meilleure solution.

1.8 Algorithme chaotique du multiplicateur de Lagrange augmenté (CALM)

Un problème d'optimisation sous contrainte peut être formulé de la manière suivante comme suit [2] :

$$min \;\; f(\overrightarrow{x}) Soumis\,à : g_j(\overrightarrow{x}) \geq 0 \;\; j = 1, ..., J h_k(\overrightarrow{x}) = 0 \;\; k = 1, ..., K x_i^l \leq x_i \leq x_i^u \;\; i = 1, ..., n. \tag{1.5}$$

Où $\overrightarrow{x} = (x_1, x_2, x_3, ..., x_n)$, $\overrightarrow{L} = (x_1^l, x_2^l, x_3^l, ..., x_n^l) and \; \overrightarrow{U} = (x_1^u, x_2^u, x_3^u, ..., x_n^u$ respectivement, sont le vecteur de conception, le vecteur de limite inférieure et le vecteur de limite supérieure pour les n variables d'optimisation,$f(\overrightarrow{x})$ est la fonction objective.$g_j(\overrightarrow{x}) \geq 0, (j = 1, \ldots, J)$ sont J contraintes d'inégalité et $h_k(\overrightarrow{x}) = 0, (k = 1, \ldots, K)$ sont K contraintes d'égalité.La valeur x_i^l définit la limite inférieure et la valeur x_i^u est la limite supérieure de x_i.L'espace de recherche S est défini par les limites supérieure et inférieure et la région réalisable Ω est définie par les contraintes d'inégalité et d'égalité [2].

Dans cette section, on va présenté une méthode CALM basée sur l'ergodicité des cartes chaotiques pour augmenter l'efficacité de l'étape de recherche locale et équilibrer la recherche d'exploration et d'exploitation pour résoudre les problèmes d'optimisation sous contraintes. Cette méthode utilise la nature ergodique de la carte du chaos pour réduire l'espace de recherche et obtenir les meilleurs paramètres pour résoudre les contraintes du problème.Ensuite, la méthode du First carrier wave (première onde porteuse) (FCW) peut être appliquée pour obtenir une solution optimale comme point initial de la méthode du simplexe de Nelder-Mead qui recherche finalement la solution optimale. L'algorithme du simplexe de Nelder-Mead est une méthode générique permettant de minimiser des fonctions non linéaires sans contraintes. La méthode peut être très sensible au point initial. L'algorithme simplex de Nelder-Mead a été mis en œuvre dans MATLAB sous le nom de Fminsearch.J. D'Errico a publié en 2006 la routine MATLAB Fminsearchbnd pour résoudre les problèmes d'optimisation sous contraintes par limites [2] .

1.8.1 Contexte théorique

Cette sous-section présente quelques concepts de base tels que les cartes de chaos, l'algorithme FCW et la méthode de base de multiplicateur de Lagrange augmenté pour mieux comprendre le nouveau algorithme.

1.8.1.1 Principes fondamentaux de l'optimisation chaotique

Les méthodes d'optimisation basées sur le chaos génèrent des séquences chaotiques en utilisant des systèmes dynamiques ou des cartes chaotiques, tandis que les algorithmes évolutionnaires utilisent généralement un processus aléatoire pour générer les solutions suivantes [2]. La figure 1.5 montre les principales différences entre les séquences chaotiques et aléatoires générées respectivement par la carte logistique et la distribution normale [2].

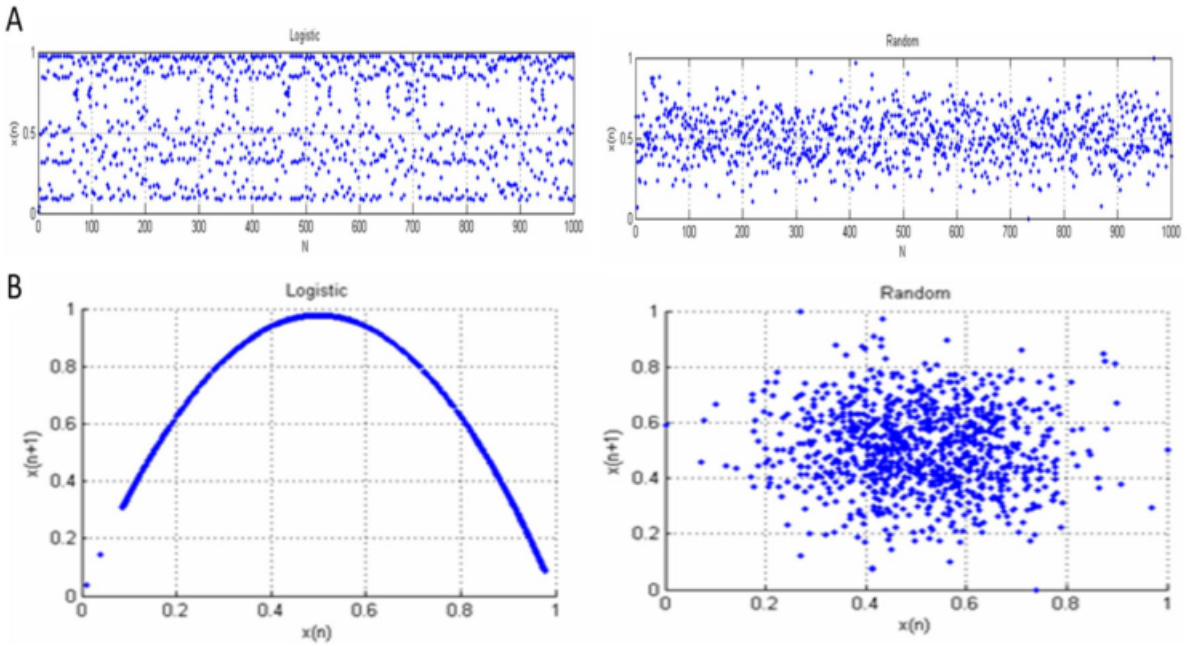


FIGURE 1.5 – a Séries temporelles et b Espace d'état des séries temporelles. Séries temporelles d'un système chaotique et aléatoire, les séquences chaotiques peuvent être générées en utilisant la carte logistique $x_{n+1} = \lambda x_n(1 - x_n)$, avec $\lambda = 3.19$ et la condition initiale $x_0 = 0.01$. La série temporelle est vraiment similaire aux séries temporelles générées par des nombres normaux aléatoires de moyenne zéro et de variance 1. L'espace d'état des deux systèmes qui représente la différence entre x_n et x_{n+1} [2]

Comme on peut le voir sur la figure 1.5, le modèle déterministe du chaos peut diminuer le coût de la complexité car ce modèle génère des solutions non répétitives. Une variable chaotique possède trois caractéristiques de base, dont le caractère pseudo-aléatoire, l'ergodicité et l'irrégularité. Par conséquent, la théorie du chaos exploite la compréhension de la dynamique non linéaire et peut être appliquée aux problèmes d'optimisation. [2]

1.8.1.2 Optimisation chaotique

En 1963, Edward Lorenz a présenté le premier attracteur chaotique à l'aide d'un modèle réduit. Il a présenté l'une des images les plus célèbres de la théorie du chaos, connue sous le nom d'effet papillon". Le chaos est une manière d'agir pseudo-aléatoirement dans un système déterministe et non linéaire qui dépend des conditions initiales. May (1976) [15] a montré que la dépendance aux conditions initiales est perçue dans les systèmes complexes. De la même manière, elle existe dans les cartes simples telles que la carte logistique. En général, une carte chaotique unidimensionnelle peut être exprimée comme suit [2] :

$$x(n+1) = f(\mu_1, \mu_2, ..., \mu_m, x(n)), \quad n = 0, 1, 2, 3, ... \tag{1.6}$$

Où μ_i,(i=1,...,m) sont les paramètres de contrôle et x est une variable. Le comportement de la carte (1.2) dépend des valeurs des paramètres de contrôle. En particulier, Eq.(1.2) est une carte chaotique pour certaines valeurs des paramètres de contrôle déterminés μ_i, (i = 1, . . . , m) chaque fois qu'un très petit changement dans la valeur initiale de la variable chaotique x, fera une différence considérable dans les prochaines valeurs de la variable chaotique x.Dans la figure 1.6, quelques exemples notables de cartes chaotiques unidimensionnelles sont présentés. [2]

Name	Equation	C^*	Source
Chebyshev	$x(n+1) = \cos\left(k\cos^{-1}(x(n))\right) x \in (-1, 1)$	k = 2	Tavazoei and Haeri (2007); He et al. (2009), Kohda et al. (1992)
Logistic	$x(n+1) = ax(n)(1-x(n))x(0) \in (0, 1)$, $x(0) \notin \{0, 0.25, 0.5, 0.75, 1\}$	a = 4	Arora et al. (1995)
Circle	$x(n+1) = x(n) + b - (a - 2\pi)\sin(2\pi x(n)) \bmod (1)$	a = 0.5, b = 0.2	Hilborn (2004)
Gaussian	$x(n+1) = \begin{cases} 0 x(n) = 0 \\ \frac{\mu}{x(n)} \bmod 1 x(n) \neq 0 \end{cases}$	μ = 1	Bucolo et al. (2002), Tavazoei and Haeri (2007)
ICMIC	$x(n+1) = \sin\left(\frac{a}{x(n)}\right) x \neq 0, x \in [-1, 1], a \in (0, \infty)$	a = 2	Tavazoei and Haeri (2007), He et al. (2009), He et al. (2001)
Sine	$x(n+1) = \left(\frac{a}{4}\right)\sin(\pi x(n)) 0 < a \leq 4$	a = 4	Devaney (1987)
Kent	$x(n+1) = \begin{cases} \frac{x(n)}{\beta} 0 < x(n) \leq \beta \\ \frac{(1-x(n))}{(1-\beta)} \beta < x(n) \leq 1 \end{cases}$	β = 0.4	Yang et al. (2007)

FIGURE 1.6 – Exemples de cartes chaotiques 1-D [2]

C^* valeur du paramètre de contrôle lorsqu'une séquence chaotique est générée par cette carte.

Les méthodes d'optimisation chaotiques utilisent des séquences numériques,créées par des cartes chaotiques. L'algorithme FCW est principalement une méthode d'optimisation chaotique de base développée pour l'optimisation globale. L'algorithme FCW est basé sur la carte logistique. [16] [17]. Le problème d'optimisation non linéaire est :

$$minf(x), \quad s.t. \quad x[L, U] \tag{1.7}$$

où f est la fonction objective, $x = (x_1, x_2, \ldots, x_n),\ L = (l_1, l_2, \ldots, l_n),\ U = (u_1, u_2, \ldots, u_n)$ et $x_i \epsilon [l_i, u_i], (i = 1, 2, \ldots., n)$. .

La logique de l'algorithme FCW est de générer des points chaotiques candidats x^c dans le domaine $[L, U]$ le vecteur de séquences chaotiques γ_1 . Afin de générer ces points, une carte linéaire est définie comme suit [2] :

$$\gamma_1(i) = \frac{(x_i^c - l_i)}{(u_i - l_i)}, \quad i = 1, 2, ..., n. \tag{1.8}$$

Donc, $0 \leq \gamma_1(i) \leq 1$ et $x_i^c = l_i + \gamma_1(i)[u_i - l_i]$, i = 1,2,...,n.

Ainsi, dans FCW, on suppose que les composantes initiales de γ_1 sont des valeurs aléatoires comprises dans l'intervalle (0, 1) et que dans les itérations suivantes, chaque composante de γ_1 , $\gamma_1(i)$, sera générée à l'aide d'une carte logistique. en utilisant la carte logistique. Le schéma de cet algorithme est maintenant fourni [2] :

Étape 1 : Initialisation : $\gamma_1(i), i = 1, ..., n.$
$\gamma_1(i)$ est une valeur aléatoire dans l'intervalle (0, 1).

Étape 2 : Générer des points chaotiques x_i^c dans le domaine $[l_i, u_i]$ par
$x_i^c = l_i + \gamma_1(i)[u_i - l_i]$, i = 1,2,...,n.

Étape 3 : x_1 est le point de conception avec la plus petite valeur de $f(x)$:
dans la première itération $x_1 \leftarrow x^c$ sinon,
dans les itérations suivantes, si $f(x^c) < f(x_1)$ alors $x_1 \leftarrow x^c$ sinon x_1 se fixe en place.

Étape 4 : Générer $\gamma_1(i)$ par carte logistique :
$\gamma_1(i + 1) = 4\gamma_1(i)[1 - \gamma_1(i)]$

Étape 5 : Répétez les étapes 2 à 4 de la recherche jusqu'à ce qu'un critère d'arrêt soit satisfait.

Puisque l'algorithme FCW a un certain problème de convergence,il a été combiné avec la recherche locale .

1.8.1.3 Méthode du multiplicateur de Lagrange augmenté

Puisque la technique de traitement des contraintes utilisée dans cette étude est liée à la méthode du multiplicateur de Lagrange augmenté, elle sera décrite dans cette section. La fonction de Lagrange augmentée pour résoudre le problème d'optimisation sous contrainte(1.5) est définie comme

suit : [2]

$$\mathcal{L}_A(\overrightarrow{x}, \overrightarrow{\lambda}; \overrightarrow{\mu}) = f(\overrightarrow{x}) + \sum_{i=1}^{J+K} \lambda_i c_i(\overrightarrow{x}) + \sum_{i=1}^{J+K} \mu_i c_i^2(\overrightarrow{x}), x_i^1 \leq x_i \leq x_i^u \quad i = 1, ..., n \tag{1.9}$$

Où

$$c_i(\overrightarrow{x}) = \begin{cases} h_i(\overrightarrow{x}) & i = 1, ..., K, \\ min[g_{i-k}(\overrightarrow{x}), \frac{\lambda_i}{2\mu_i}] & i = K+1, ..., K+J \end{cases} \tag{1.10}$$

$\overrightarrow{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_{J+K})$, $\overrightarrow{\mu} = (\mu_1, \mu_2, ..., \mu_{J+K})$ et λ_i, μ_i, $(i = 1, 2, ..., J+K)$ sont des nombres scalaires positifs.

Définition (condition de Karush-Kuhn-Tucker (KKT)) :
Supposons que $\overrightarrow{v} = (v_1, v_2, ..., v_n)$, $\overrightarrow{L} = (x_1^l, x_2^l, ..., x_n^l)$, and $\overrightarrow{U} = (x_1^u, x_2^u, ..., x_n^u)$. Soit $P(\overrightarrow{v}, \overrightarrow{L}, \overrightarrow{U})$ soit la projection du vecteur $\overrightarrow{v}$ sur la boîte rectangulaire $[\overrightarrow{L}, \overrightarrow{U}]$ qui est définie comme suit [2] :

$$P(\overrightarrow{v}, \overrightarrow{L}, \overrightarrow{U}) = \begin{cases} x_i^l & if \quad v_i \leq x_i^l \\ v_i & if \quad x_i^l \leq v_i \leq x_i^u \\ x_i^u & if \quad v_i \geq x_i^u \end{cases} \qquad for\ i = 1, 2, ..., n \tag{1.11}$$

On peut prouver que les conditions de KKT pour le problème sous contrainte de limite (1.9) correspondent à $\overrightarrow{x} - P\big(\overrightarrow{x} - \nabla_(\overrightarrow{x})\mathcal{L}_A\big(\overrightarrow{x}, \overrightarrow{\lambda}; \overrightarrow{\mu}\big), \overrightarrow{L}, \overrightarrow{U}\big) = 0$.

Soit $\overrightarrow{x}$ la solution optimale du problème d'optimisation sous contrainte (1.5) alors $\overrightarrow{x}^*$ est un point stationnaire de la fonction de Lagrange de l'équation (1.5) pour les multiplicateurs précis $\overrightarrow{\lambda}^*$. Par conséquent, $\overrightarrow{x}^*$ n'est pas une solution minimale de la fonction pseudo-objectif sans contrainte (ou de la fonction de Lagrange standard) en Eq.(1.12). Cependant, $\overrightarrow{x}^*$ est le point minimum de la fonction multiplicateur de Lagrange augmentée [2].

$$\mathcal{L}\big(\overrightarrow{x}, \overrightarrow{\lambda}\big) = f(\overrightarrow{x}) + \sum_{k=1}^{K} \lambda_k h_k(\overrightarrow{x}) - \sum_{j=1}^{J} \lambda_j g_j(\overrightarrow{x}) \tag{1.12}$$

$$x_i^l \leq x_i \leq x_i^u \quad i = 1, ..., n.$$

La fonction de pénalité quadratique du problème d'optimisation sous contrainte (1.5) est définie comme suit :

$$\mathcal{Q}\big(\overrightarrow{x}, \overrightarrow{\mu}\big) = f(\overrightarrow{x}) + \sum_{k=1}^{K} \big(\frac{\mu_k}{2}\big) h_k^2(\overrightarrow{x}) - \sum_{j=1}^{J} \big(\frac{\mu_j}{2}\big)([g_j(\overrightarrow{x}]^-)^2 \tag{1.13}$$

$$x_i^l \leq x_i \leq x_i^u \quad i = 1, ..., n.$$

avec $[g_j(\overrightarrow{x})]^- = max(-g_j(\overrightarrow{x}), 0)$.

La principale faiblesse de cette technique est le mauvais conditionnement lorsque μ_i s'approche de l'infini. Cependant, cette méthode a un certain nombre de limites et n'est pas recommandée dans de nombreux cas. Dans une analyse de la méthode du multiplicateur de Lagrange augmenté, il a été montré que la convergence de cet algorithme peut être assurée lorsque μ_i est finie. Par conséquent, le mauvais conditionnement dans cette technique est moins problématique que la méthode de pénalisation par rapport à la méthode de pénalité. Cependant, la fonction du multiplicateur de Lagrange augmenté est une combinaison de la fonction de Lagrange standard et de la fonction de pénalité quadratique avec le plus petit nombre d'erreurs. Selon les conditions d'optimalité pour la minimisation des problèmes non contraints, il existe une méthode numérique et pratique pour mettre à jour $\overrightarrow{\lambda}$,$\overrightarrow{\mu}$ par Eq.(1.14) [2].

$$\nabla_{(}\overrightarrow{x})\mathcal{L}_A\big(\overrightarrow{x}^k, \overrightarrow{\lambda}^k; \overrightarrow{\mu}^k\big) = \nabla f(\overrightarrow{x}^k) - \sum_{i=1}^{J+K}[\lambda_i^k - \mu_i^k c_i(\overrightarrow{x}^k)]\nabla_{ci}(\overrightarrow{x}^k) \approx 0 \tag{1.14}$$

k=1,2,...

Enfin, $\overrightarrow{\lambda}$ est calculé avec la formule récursive suivante :

$$\lambda_i^{k+1} \approx \lambda_i^k - \mu_i^k c_i(\overrightarrow{x}^k), \qquad i = 1, 2, ..., J + K, \quad k = 1, 2, ... \tag{1.15}$$

1.8.2 Description de l'algorithme CALM

Cette sous-section décrit l'algorithme CALM développé dans cette recherche. CALM intègre la technique du multiplicateur de Lagrange augmentée, des cartes chaotiques, l'algorithme d'optimisation chaotique et une méthode de recherche directe de Lagarias et al.(1998). Comme mentionné dans la section précédente, la méthode du multiplicateur de Lagrange augmentée est une combinaison de l'algorithme standard de Lagrange et de la technique de la pénalité quadratique pour traiter les contraintes du problème avec le moins d'inconvénients et d'inconditionnalités possibles. En fait, l'algorithme proposé améliore la méthode du multiplicateur de Lagrange augmenté par cartes chaotiques et incorpore la technique FCW avec afin de développer une méthode générique. Ainsi, la meilleure sous-région réalisable peut être obtenue par la carte logistique avec un effet d'inspiration de FCW. Ensuite, la méthode FCW est utilisée pour minimiser le Lagrangien augmenté,Eq.(1.8), par rapport à $\overrightarrow{x}$ et avant d'exécuter toute mise à jour des multiplicateurs de Lagrange $(\overrightarrow{\lambda})$ et des facteurs de pénalité $(\overrightarrow{\mu})$. Les facteurs de pénalité sont mis à jour par l'algorithme 1 [2].

Algorithm 1 Mise à jour de $\overrightarrow{\mu} = (\mu_1, \mu_2, ..., \mu_m)$ [2]

Début de l'algorithme
m= Nombre de contraintes,
m_equ = Nombre de contraintes d'égalité,
m_inequ = Nombre de contraintes d'inégalité
for $j = 1, 2, ..., m$ **do**
  if $(m\_equ \neq 0 \; and \; m\_equ \geq j)$ **then**

    if $(j - th \; contrainte \; \neq 0)$ **then**

      $\mu_j \leftarrow 100\mu_j$
    else
      $\mu_j \leftarrow \frac{\mu_j}{2}$
  else if$(m\_inequ \neq 0 \; and \; j > m\_equ)$ **then**

    if $(j - th \; contrainte \; > 0)$ **then**

      $\mu_j \leftarrow 100\mu_j$
    else
      $\mu_j \leftarrow \frac{\mu_j}{2}$
Fin de l'algorithme

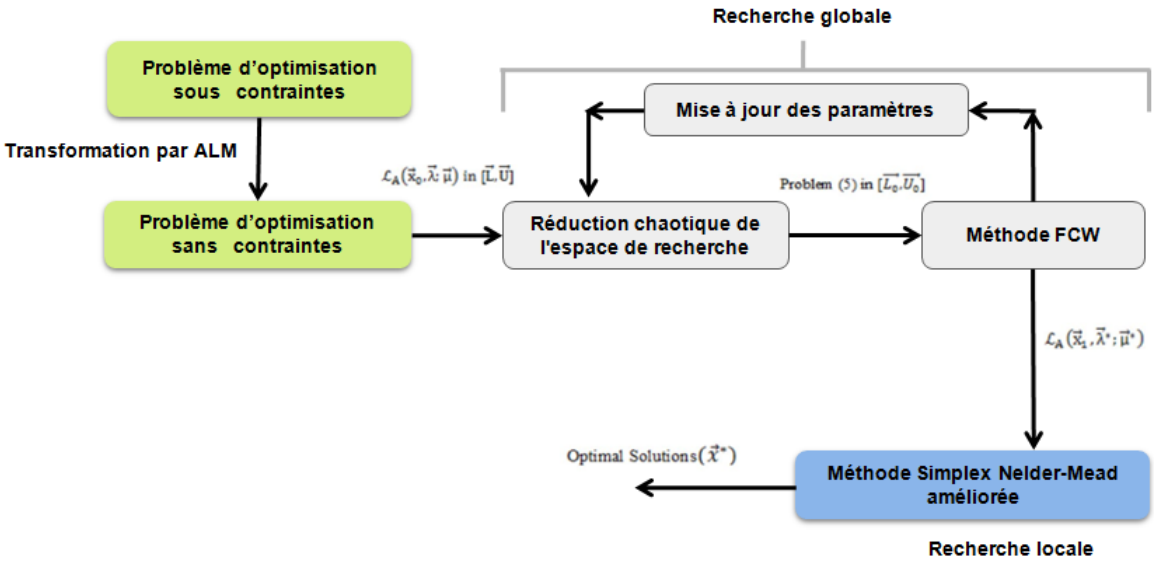


FIGURE 1.7 – Architecture générale de l'algorithme proposé . [2]

Pour cette raison, l'algorithme proposé comporte trois phases principales. Première phase, la carte chaotique améliore la technique du multiplicateur de Lagrange augmenté pour traiter les contraintes.L'algorithme proposé utilise l'ergodicité des cartes chaotiques pour réduire l'espace de recherche et trouver les multiplicateurs de Lagrange appropriés ($\overrightarrow{\lambda}$) et les facteurs de pénalité($\overrightarrow{\mu}$). Deuxième phase, la méthode FCW explore une valeur initiale appropriée en tant qu'étape de recherche globale.Troisième étape, la nouvelle version de l'algorithme simplex de Nelder-Mead peut être utilisée comme une recherche locale pour trouver la solution optimale [2]. La figure (1.7) illustre l'architecture générale de l'algorithme CALM.

Les principales étapes de calcul de l'algorithme proposé sont maintenant décrites [2] :

Étape 1 : Initialisation :
Mettre $k = 0,\ \overrightarrow{\lambda^0} = (\lambda_1^0, \lambda_2^0, ..., \lambda_{J+K}^0),\ \lambda_i^0 = \lambda 0_i,\ i = 1, 2, ..., J + K.$
$\overrightarrow{\mu^0} = (\mu_1^0, \mu_2^0, ..., \mu_{J+K}^0),\ \mu_i^0 = \mu 0_i,\ i = 1, 2, ..., J + K.$
$\omega_0 = \frac{1}{\|\overrightarrow{\mu^0}\|},\ \eta_0 = \frac{1}{\|\overrightarrow{\mu^0}\|^{0.1}}$

Choisir un point initial $\overrightarrow{x_0} = (x_1^0, x_2^0, ..., x_n^0)$, et des tolérances de convergence $\eta_* \ et \ \omega_*$.

Étape 2 : Convergence (condition KKT) :
Evaluer la fonction du multiplicateur de Lagrange augmenté en $\overrightarrow{x} = \overrightarrow{x_k} = ((x_1^k, x_2^k, ..., x_n^k)$ selon les équations (1.9) et (1.10), de sorte que

$$\|\overrightarrow{x_k} - P\Big(\overrightarrow{x_k} - \nabla_{(}\overrightarrow{x}_{)}\ \mathcal{L}_A(\overrightarrow{x_k},\ \overrightarrow{\lambda}^k; \overrightarrow{\mu}^k),\ \overrightarrow{L}\,,\ \overrightarrow{U}\Big)\| \leq \omega_k, \quad k = 0, 1, 2, ... \tag{1.16}$$

Étape 1 : Réduire l'espace de recherche :
N_1 est le nombre d'itérations pour obtenir un sous-espace faisable correct.
1. $\gamma_1(i)$ est une valeur aléatoire dans l'intervalle (0, 1).
2. Générer des points $x_1(i), x_u(i)$ dans le domaine $[x_i^l, x_i^u]$ par :

$$x_l(i) = x_i^l + \gamma_1(i)\big(x_i^u - x_i^l\big), \quad i =, ..., n.$$

$$x_u(i) = x_i^u + \gamma_1(i)\big(x_i^u - x_i^l\big), \quad i =, ..., n.$$

3. Si $x_u(i) < x_l(i)$ alors $ll \leftarrow x_u(i), x_u(i) \leftarrow x_l(i), x_l(i) \leftarrow ll\ \ i = 1, ..., n.$

$$\overrightarrow{x}_l^k = \big(x_l(1), x_l(2), ..., x_l(n)\big),\ \overrightarrow{x}_u^k = \big(x_u(1), x_u(2), ..., x_u(n)\big)$$

4. Si $\mathcal{L}_A(\overrightarrow{x}_l^k,\ \overrightarrow{\lambda}^k; \overrightarrow{\mu}^k) < \mathcal{L}_A(\overrightarrow{L}\,,\ \overrightarrow{\lambda}^k; \overrightarrow{\mu}^k)$alors $\overrightarrow{L} \leftarrow \overrightarrow{x}_l^k$ sinon, laisser $\overrightarrow{L}$inchangé.

Si $\mathcal{L}_A(\overrightarrow{x}_u^k, \overrightarrow{\lambda}^k; \overrightarrow{\mu}^k) < \mathcal{L}_A(\overrightarrow{U}, \overrightarrow{\lambda}^k; \overrightarrow{\mu}^k)$alors $\overrightarrow{U} \leftarrow \overrightarrow{x}_u^k$ sinon, laisser $\overrightarrow{U}$inchangé.

5. Générer $\gamma_1(i+1)$par carte logistique : $\gamma_1(i+1) = 4\gamma_1(i)[1-\gamma_1(i])$
6. Répéter les étapes 1-5 jusqu'à ce que le nombre maximal d'itérations N_1 soit atteint.

Étape 4 : Optimisation :
Utiliser FCW pour résoudre la fonction multiplicatrice de Lagrange augmentée indiquée par l'équation (1.9) et la sous-gamme de l'étape 3 pour le nombre limite d'itérations N_{max}.

L'algorithme dit de la première onde porteuse (first carrier wave)est la procédure la plus élémentaire pour générer des points candidats x_c à l'intérieur d'une région réalisable ; l'optimum, X_i, est le point candidat ayant la plus petite valeur pour(x_c). Le processus est schématisé dans l'algorithme 2.Les points candidats X_c (ligne 05) ont été générés dans le domaine $[L, U]$ au moyen du vecteur de séquence chaotique γ ; chaque composante de $\gamma, \gamma(i)$, a donc été cartographiée linéairement sur l'intervalle $[L(i), U(i)]$.On a supposé dans l'algorithme 1 que les composantes de γ étaient restreintes à l'intervalle [0, 1] comme cela se produit pour la carte logistique. Un nouveau vecteur de séquence chaotique a été généré à chaque itération en utilisant la carte chaotique H(ligne 12) ; par exemple, chaque composante a été générée en utilisant $\gamma(i) = 4\gamma(i)[1-\gamma(i)]$, si $H()$ était la carte logistique.L'optimum local actuel x_1 a été sauvegardé à la ligne 09.En outre, toutes les fCW.C2 itérations, l'optimum local actuel serait affiné au moyen d'un algorithme d'optimisation basé sur le gradient (ligne 14).Le processus complet a été répété quelques. C_i fois (de la ligne 02 à la ligne 15). [5]

Algorithm 2 FCW algorithme [5]

```
Début de l'algorithme
initialiser γ, fcw.C1, fcw.C2
for (c1 = 1, ..., fcw.c1) do
    new.min.found=FALSE
    for (c2 = 1, ..., fcw.c2) do
        xc = L + γ(U − L)
        if (c1 == 1) then
            xl = xc
            ∇f = f(xc) − f(xl)
        if (∇f < 0) then
            xl = xc
            new.min.found=TRUE
        γ = H(γ)
Fin de l'algorithme
```

Étape 5 : Mettre à jour les paramètres :

1. Mettre à jour $\overrightarrow{\lambda}^k = (\lambda_1^k, \lambda_2^k, ..., \lambda_{J+K}^k)$ par l'éq(1.15)

2. Mettre à jour $\overrightarrow{\mu}^k = (\mu_1^k, \mu_2^k, ..., \mu_{J+K}^k)$ avec l'algorithme (1).
3. Où $\overrightarrow{x}^k$ est la solution optimale du kième sous-problème obtenu à l'étape 3 et enfin,on fixe $k = k + 1$.

Étape 6 : Continuer avec l'étape 2.

Étape 7 : Utiliser la nouvelle version de l'algorithme simplex de Nelder-Mead pour résoudre la fonction du multiplicateur de Lagrange augmenté indiquée par l'Eq.(1.9) avec le point initial $\overrightarrow{x}^k$.

L'algorithme du simplexe de Nelder-Mead est une méthode générique permettant de minimiser des fonctions non linéaires sans contrainte. Cette méthode peut être très sensible au point initial. L'algorithme du simplexe de Nelder-Mead a été implémenté dans MATLAB sous le nom de Fminsearch. MATLAB sous le nom de Fminsearch. J. D'Errico a publié en 2006 la routine MATLAB Fminsearchbnd pour résoudre les problèmes d'optimisation sous contraintes [2].

1.9 Conclusion

Nous avons rappelé dans ce chapitre quelques notions sur l'optimisation. Puis, nous avons mis en clair les deux types des techniques d'optimisation : déterministes et stochastique, parmi ces dernières on trouve les métaheuristiques qui exploitent généralement des processus aléatoires dans l'exploration de l'espace de recherche.

En dernier, nous avons présenté un nouvel algorithme d'optimisation sous contraintes basé sur le chaos appelé (CALM). L'algorithme (CALM) est basé sur l'ergodicité des cartes chaotiques pour augmenter l'efficacité de l'étape de recherche locale et ééquilibrer la recherche d'exploration et d'exploitation pour résoudre les problèmes d'optimisation sous contraintes.Vous pouvez ensuite appliquer la méthode du première onde porteuse (FCW) pour obtenir la solution optimale comme point de départ de la méthode du simplexe de Nelder-Mead, qui trouve finalement la solution optimale. L'algorithme simplex de Nelder-Mead est un moyen courant de minimiser les fonctions non linéaires sans contraintes.

Dans le chapitre suivant, nous allons décrire un autre domaine qui fera l'objet de notre étude pratique, il s'agit des séries temporelles.

Chapitre 2

Les séries temporelles

Chapitre 2

Les séries temporelles

Introduction

Prédire l'évolution future d'un phénomène suscite depuis très longtemps un intérêt particulier ce qui a accéléré le développement des techniques prédictives et apportant diverses valeurs scientifiques entre ces techniques.

La prévision est le processus qui consiste à estimer ou à prédire les événements futurs (ou les valeurs) d'une séquence de données historiques basée sur le temps (appelée série chronologique) afin de faire face aux incertitudes futures [3].

La prédiction traditionnelle nécessite la connaissance d'informations antérieures sur le comportement du système à prédire. Une méthode de prédiction est complètement déterministe si le calcul du comportement futur est précis. Mais en réalité, en raison de plusieurs facteurs, il n'est pas possible de calculer avec précision le comportement futur. Cependant, il est possible de générer un modèle qui peut être utilisé pour calculer la probabilité d'un comportement futur entre deux limites spécifiées. Tel modèle est appelé modèle stochastique ou processus stochastique. Une importante classe de modèles stochastiques est utilisée pour décrire des séries temporelles stationnaires appelée la classe des modèles stochastiques stationnaires. Ces modèles supposent que les propriétés des séries chronologiques sont invariantes par la translation temporelle ; parmi ces modèles il y a : AR, MA, ARMA. Les processus utilisés pour la description des séries temporelles non stationnaires (en moyenne, en variance, cessionnaire) sont : ARIMA, SARIMA. De plus, pour la construction des modèles quelles que soient leurs classes, Box et Jenkins ont introduit une méthodologie utilisée pour obtenir le modèle linéaire optimal pour une série chronologique. Cette méthodologie se décompose en trois étapes : l'identification du modèle, l'estimation des paramètres et la validation du modèle.

2.1 Définition série temporelle (TS)

Une (TS) (ou chronologique) est une suite d'observations $(x_1, x_2, ..., x_n)$ indexées par le temps. L'indice temps peut être selon les cas : la minute, l'heure, le jour, l'année etc....
Le nombre n est appelé la longueur de la série.

Par conséquent, une série chronologique est un ensemble d'observations qui correspondent à la même variable. Données macroéconomiques (PIB national, inflation, exportations, etc.), données microéconomiques (chiffre d'affaires d'une entreprise particulière, nombre d'employés, revenus personnels, nombre d'enfants de sexe féminin, etc.), finances (prix de l'option d'achat ou de vente, actions, etc. .) Prix), météo (pluies, journées ensoleillées de l'année...), politique (nombre d'électeurs, suffrages reçus par les candidats...), démographie (taille moyenne des habitants, âge...). En pratique, tout ce qui est chiffrable et varie en fonction du temps. Ce qui est important ici, c'est la dimension temporelle car il s'agit de l'analyse d'une chronique historique : des variations d'une même variable au fil du temps, afin de pouvoir comprendre la dynamique. La périodicité de la série n'est pas importante : il peut s'agir de mesures quotidiennes, mensuelles, trimestrielles, annuelles... voire même sans périodicité [18].

En général, les séries temporelles sont tracées sur un graphique de valeurs (ordonnées) en fonction du temps (abscisses). Quand une série est stable autour de sa moyenne, elle est dite série stationnaire. Inversement, il existe également des séries non stationnaires. Quand la série croit sur l'ensemble de l'échantillon et donc possède une moyenne non constante, on parle de tendance.Enfin, lorsqu'on observe un phénomène qui se répète régulièrement, on parle de phénomènes saisonniers [18].

2.1.1 Exemples

• Les taches solaires. C'est le nombre annuel de taches observées à la surface du soleil entre 1700 et 1980. Il y a un enregistrement par an. donc 281 points reliés par des segments.

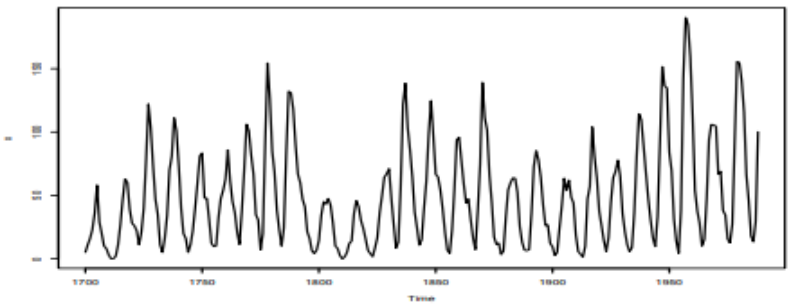


FIGURE 2.1 – Taches solaires

• Les ventes de voitures. Les données fonctionnent depuis 35 ans et il y en a une par mois. C'est-à-dire 420 points.

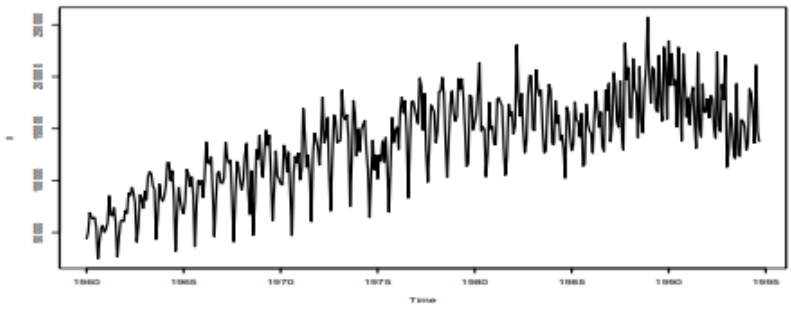


FIGURE 2.2 – Ventes de voitures

• Nombre de passagers pour le transport aérien (milliers). Une donnée par mois de 1949 à 1960.

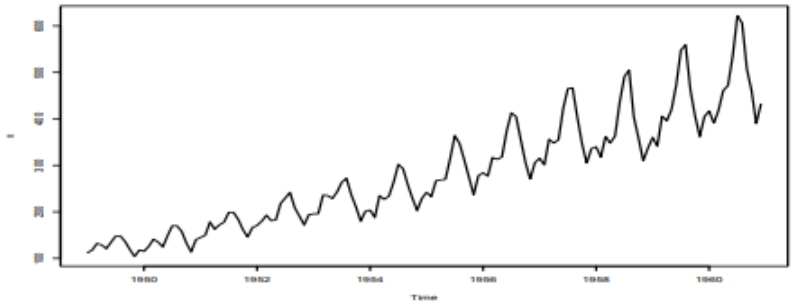


FIGURE 2.3 – Trafic aérien

2.2 Domaines d'application

On trouve des exemples de séries chronologiques dans de très nombreux domaines.
La liste suivante n'est qu'un échantillon :

- ☐ Médecine/biologie : suivi des évolutions des pathologies, analyse d'électroencéphalogrammes et d'électrocardiogrammes ;
- ☐ Traitement du signal : signaux de communications, de radars, de sonars, analyse de la parole ;
- ☐ Traitement des données : mesures successives de position ou de direction d'un objet mobile (trajectographie) ;
- ☐ Finance et économétrie : évolution des indices boursiers, des prix, des données économiques des entreprises, des ventes et achats de biens, des productions agricoles ou industrielles ;
- ☐ Météorologie : analyse de données climatiques. . .

2.3 Caractéristiques des séries temporelles

Pour traiter une série temporelle pour une application quelconque, il faut prendre en compte certains éléments (caractéristiques) qui caractérisent la série. Ces caractéristiques sont :

• **La tendance**, qui indique une augmentation ou une diminution moyenne des valeurs d'une série chronologique sur une longue période. La figure 2.4 montre un exemple de ce à quoi ressemble une tendance, où la ligne en pointillé indique une tendance linéaire à long terme. Une tendance non linéaire peut également être observée dans une série chronologique. Une tendance à la hausse est généralement observée dans les séries chronologiques relatives à la croissance démographique et à la consommation d'eau dans une communauté, tandis qu'une tendance à la baisse peut être observée dans les séries relatives aux épidémies et au taux d'analphabétisme [3].

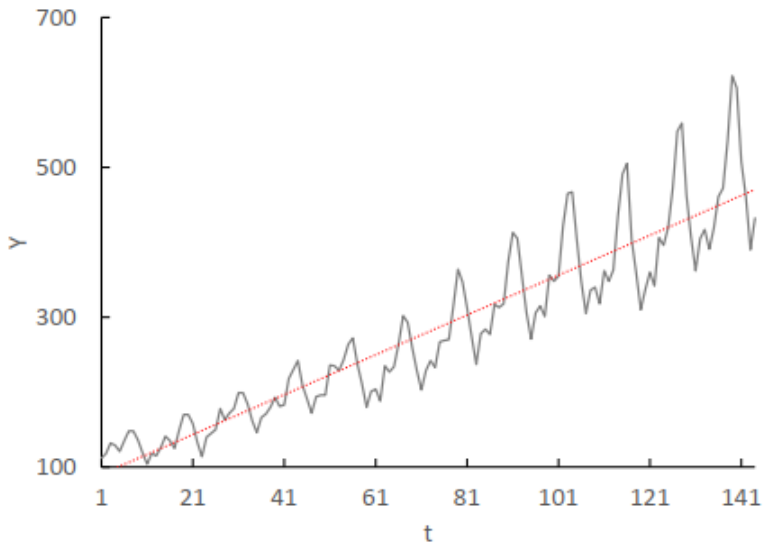


FIGURE 2.4 – Tendance linéaire [3]

• **Saisonnalité**, qui est une fluctuation régulière des valeurs d'une série chronologique sur des périodes fixes et connues. Le mouvement saisonnier peut être observé sur une base trimestrielle, mensuelle, hebdomadaire ou même quotidienne, et la saisonnalité est soit déterministe, soit stochastique. L'apparence de la variation saisonnière est illustrée par la figure 2.5. Les fluctuations saisonnières peuvent être dues aux coutumes, aux préférences, aux changements météorologiques et climatiques et aux technologies [3].

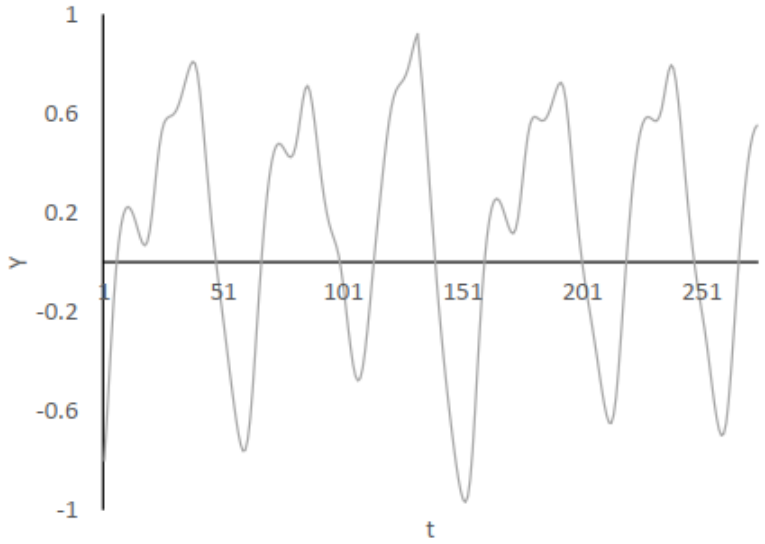


FIGURE 2.5 – Variation saisonnière [3]

• **Variations cycliques**, qui sont des changements réguliers des valeurs de séries temporelles sur une période variable inconnue (généralement plus d'une année). Les variations cycliques sont causées par des événements qui se produisent en cycles. La figure 2.6 montre un exemple de série temporelle avec des cycles d'environ 10 ans (certains durent huit, neuf ou dix ans). Contrairement à la saisonnalité, les variations cycliques couvrent une période plus longue et présentent un schéma systématique difficilement prévisible. Les variations cycliques peuvent être observées dans la plupart des séries financières et économiques [3].

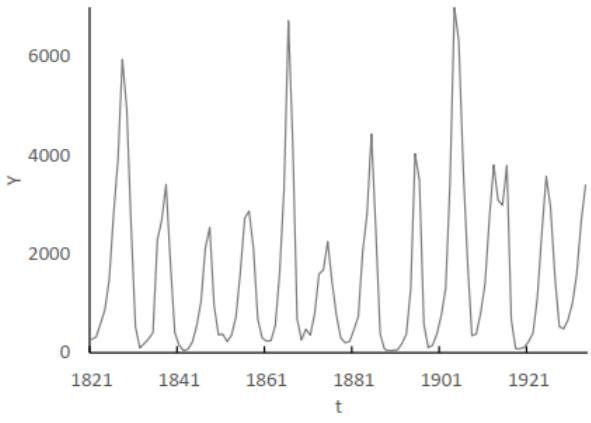


FIGURE 2.6 – Variation cyclique [3]

• **Variations aléatoires**, également appelées erreurs résiduelles, sont des mouvements irréguliers dans une série chronologique causés par des circonstances imprévisibles qui ne sont pas régulières

et ne se répètent pas selon un schéma particulier. Elles sont causées par des événements tels que l'apparition d'une maladie, une révolution, une guerre ou des grèves. La figure 2.7 illustre un exemple de série à variations aléatoires. aléatoires. Les variations aléatoires ne sont pas prévisibles [3].

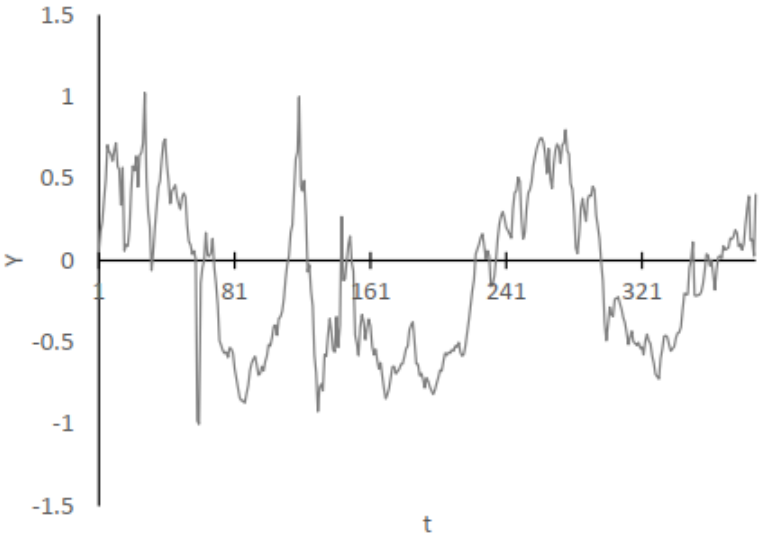


FIGURE 2.7 – Variation aléatoire [3]

D'autres caractéristiques telles que les valeurs aberrantes et les observations manquantes sont souvent observées dans une série chronologique. Les observations aberrantes sont des observations de données qui s'écartent considérablement de la distribution des données, provoquant des erreurs importantes sur les modèles construits à partir des données. Les observations aberrantes sont normalement éliminées à l'aide de techniques statistiques lors de la préparation des données avant la construction du modèle. Même si cette action atténue le problème des valeurs aberrantes, d'autres informations importantes peuvent être perdues en même temps. Chatfield [19] a noté que le traitement de certaines caractéristiques spécifiques telles que les valeurs aberrantes, les observations manquantes ou les erreurs possibles peut être plus important que le choix de la méthode de prévision [3].

Le tracé d'une série temporelle révèle la présence de différentes caractéristiques contenues dans la série, et est généralement utilisé pour étudier et évaluer les propriétés d'une série chronologique [3].

Étant donné que les séries chronologiques présentent généralement différentes combinaisons de composantes caractéristiques, la décomposition est employée dans l'analyse classique des séries

chronologiques pour décrire séparément ces composantes. Sur la base de la manière dont ces composantes sont combinées, la modélisation linéaire peut être classée en compositions additives, multiplicatives et pseudo-additives [20].

Un modèle additif,

$$Y_t = T + S + R \tag{2.1}$$

où T, S et R sont respectivement la tendance, les variations saisonnières et les variations aléatoires, suppose que les composantes caractéristiques sont indépendantes les unes des autres [3].

Un modèle multiplicatif,

$$Y_t = T \times S \times R \tag{2.2}$$

d'autre part, suppose que les composantes caractéristiques ne sont pas nécessairement indépendantes les unes des autres. La décomposition pseudo-additive présente des caractéristiques à la fois du modèle additif et du modèle multiplicatif [3].

2.4 Analyse des séries temporelles

L'analyse des séries chronologiques consiste à utiliser un modèle mathématique pour extraire des statistiques significatives ou d'autres caractéristiques d'une série chronologique afin de comprendre sa structure sous-jacente et les fonctions qui la produisent. Quelques applications typiques de l'analyse des séries temporelles sont [3] :

• **prévision**, qui est le processus d'estimation des valeurs futures d'une série temporelle sur la base des valeurs précédentes ;

• **la classification**, qui est le processus de regroupement des séries temporelles dans un certain nombre de classes ;

• **l'explication**, qui est le processus de description d'une série chronologique en termes de paramètres d'un modèle ; et ‘

• **la transformation**, qui est le processus de mise en correspondance d'une série temporelle avec une autre.

Les applications de prévision sont les plus répandues et les plus imminentes dans la littérature en raison de leur grande importance dans de nombreux domaines. La plupart des décisions stratégiques sont souvent basées sur les résultats des prévisions. Par conséquent, l'ajustement d'un modèle qui peut modéliser adéquatement une série chronologique est très important [3].

Dans la modélisation d'un prévisionniste de série temporelle, les valeurs de la série sont analysées pour développer un modèle mathématique qui capture le processus sous-jacent de génération de données de la série. Le modèle mathématique développé est ensuite utilisé pour prédire les valeurs futures de la série. L'analyse est basée sur le fait que les observations successives ne sont pas indépendantes et que leur ordre chronologique doit être pris en compte [3].

Comme décrit dans [21], la prévision des séries temporelles est le problème de trouver une fonction $f : \mathbb{R}^n \to \mathbb{R}$ de manière à obtenir une estimation $y(t+d)$ de y au temps $(t+d)$, étant donné n observations de y jusqu'au temps t [3] :

$$y(t+d) \approx f(y(t); y(t1); :::; y(tn)) \tag{2.3}$$

où d est le délai de prédiction (également appelé horizon de prévision). Cela implique que la est un problème d'approximation de fonctions [3].

2.5 Processus stochastiques dans les séries temporelles

Une série temporelle est dite déterministe si les valeurs futures exactes peuvent être déterminées. Cependant, dans la plupart des applications de séries temporelles, les valeurs futures ne peuvent pas être prédites avec certitude en raison de l'erreur résiduelle, ϵ, qui est le résultat des processus de bruit (produits aléatoirement par des facteurs d'influence incontrôlables). Les valeurs futures d'une série ont une distribution de probabilité qui est conditionnée par la connaissance des valeurs passées [3],

$$y(t+d) = f(y(t); y(t1);) + \epsilon; \quad \epsilon \sim N(0; \alpha^2) \tag{2.4}$$

où N est une distribution normale de moyenne nulle avec une variance de α^2.

Une expression mathématique qui décrit la structure de probabilité d'une série chronologique est appelée processus stochastique [22]. Par conséquent, une série chronologique est en fait un échantillon de réalisation d'un processus stochastique qui produit la série [19]. Les observations d'une série temporelle sont généralement supposées être indépendantes et identiquement distribuées (i.i.d), suivant une distribution normale. Cependant, cette hypothèse n'est pas tout à fait correcte, car une série temporelle présente un modèle plus ou moins régulier sur le long terme [19], à l'exception d'une série chronologique chaotique .

2.6 Méthodes de prévision des séries chronologiques

Au cours de plus d'un demi-siècle de recherche active dans le domaine de l'analyse des séries temporelles, de nombreuses méthodes de prévision des séries chronologiques ont été développées. Ces méthodes vont de techniques simples, telles que la prévision naïve (qui consiste à fixer la prévision à la dernière observation de la série temporelle) à des méthodes plus complexes d'intelligence informatique (CI) telles que les réseaux neuronaux artificiels (**NNs!**) [3].

2.6.1 Méthodes statistiques

Il existe plusieurs techniques statistiques utilisées dans la modélisation des séries temporelles. Les plus importantes de ces techniques sont la moyenne mobile (MA), le lissage exponentiel , l'autorégression (AR), la moyenne mobile autorégressive (ARMA) et la moyenne mobile intégrée autorégressive (ARIMA). Chartfield [19] a observé que les méthodes statistiques étaient les outils de prévision des séries temporelles les plus utilisés. Parmi toutes les techniques statistiques, l'ARIMA est la plus populaire et la plus utilisée.

Le modèle ARIMA a été développé par Box et Jenkins en 1970, et est devenu très populaire et important dans la prévision en raison de l'approche pratique (appelée méthodologie de Box-Jenkin) développée pour construire le modèle. Dans le modèle ARIMA, la valeur future d'une série temporelles est supposée être une fonction linéaire des observations passées de la série plus une erreur aléatoire. Par conséquent, la fonction sous-jacente qui produit la série chronologique est exprimée comme suit [3] :

$$\hat{y}_t = \phi_0 + _t + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_j \epsilon_{tj} \tag{2.5}$$

où $\hat{y}_t$ est la valeur prévue, t est l'erreur aléatoire avec la propriété de moyenne nulle et de variance constante. et de variance constante ; ϕ_i et θ_j sont les paramètres du modèle [3].

D'après l'équation (2.5), si la valeur de $q = 0$, le modèle se réduit à un modèle AR pur d'ordre p. Si $p = 0$, le modèle devient un modèle MA d'ordre q. Par conséquent, lors de la construction d'un modèle ARIMA, la tâche principale consiste à trouver l'ordre approprié, $(p; q)$, du modèle.La méthodologie de Box-Jenkins décrit trois étapes itératives afin de sélectionner un modèle ARIMA satisfaisant [3] :

• **L'étape d'identification du modèle**, qui implique le prétraitement des séries de données pour une meilleure modélisation et la détermination de l'ordre (p ; q) du modèle ARIMA. Si la série temporelle est non stationnaire, elle est réduite à une série stationnaire par simple différentiation de degré g, jusqu'à ce qu'il n'y ait plus de modèle évident tel qu'une tendance ou une saisonnalité

dans la série. Différencier signifie prendre la différence entre des observations consécutives. Une fois que les données sont stationnaires, l'ordre (p ; q) du modèle est déterminé .

• **L'étape d'estimation des paramètres**, qui consiste à optimiser les paramètres du modèle, ϕ_i et θ_j, de sorte que la mesure globale de l'erreur soit minimisée à l'aide de techniques d'optimisation non linéaires telles que l'estimation du maximum de vraisemblance et les méthodes des moindres carrés non linéaires .

• L'étape de vérification du diagnostic, qui consiste à vérifier la qualité de l'ajustement du modèle à la série. Le modèle est ensuite utilisé pour la prévision si le modèle est adéquat. Si le modèle s'avère inadéquat, les trois étapes de Box-Jenkins sont répétées jusqu'à ce qu'un meilleur modèle alternatif soit identifié.

Bien que les modèles ARIMA soient très populaires, ils sont limités à la gestion de séries temporelles linéaires. Pour surmonter ce problème, de nombreux modèles statistiques non linéaires tels que le modèle autorégressif à seuil (TAR), le modèle ARCH, le modèle GARCH et NAR ont été proposés. Ces modèles sont capables de traiter des séries temporelles non linéaires, mais leur complexité mathématique est élevée et ils dépendent aussi largement d'une connaissance spécifique de la manière dont la série temporelle est générée [[23], [24]].

2.6.2 Technique des k plus proches voisins (k-NN)

L'algorithme (k-NN) est une méthode d'apprentissage non paramétrique simple utilisée pour la classification et la prévision. L'algorithme k-NN utilise les voisinages locaux pour prendre une décision, en partant de l'hypothèse que les objets qui sont proches les uns des autres partagent des caractéristiques similaires [25]. Par conséquent, les données d'apprentissage sont considérées comme un espace métrique de caractéristiques. Dans l'algorithme k-NN, l'ensemble d'apprentissage est mémorisé et aucun modèle n'est associé au concept d'apprentissage [3].

Pour prévoir une série temporelle à l'aide de la technique k-NN, la série est organisée en k sous-séquences qui sont similaires à la sous-séquence la plus récente avant le point de prévision. Ainsi, les histoires avec un comportement dynamique similaire sont détectées dans les séries passées et utilisées plus tard dans la prévision du point suivant à la fin de la série [3].

Trois paramètres prédéterminés sont nécessaires dans la prévision k-NN, à savoir la dimension d'intégration, m, la fréquence d'échantillonnage (c'est-à-dire le temps de retard), τ, et le nombre de voisins les plus proches, k. Une fois ces paramètres déterminés, la série temporelle donnée, disons

$(y_1; y_2; :::; y_t; :::; y_T)$, est transformée en sous-séquences de longueurs égales sous forme de vecteurs, contenant m observations échantillonnées de la série à des intervalles de τ, c'est-à-dire [3]

$$y_t^{m;T} = (y_t; y_{t-\tau}; ::::; y_{t-(m-1)\tau}; \quad t = m; m+1; ::::; T \tag{2.6}$$

Ces vecteurs sont souvent appelés vecteurs m-historiques. Le k-NN utilise k vecteurs m-histoire qui ont un comportement dynamique similaire à celui du vecteur de retard, $y_T^{m;T}$, pour prédire la valeur de la série temporelle au prochain pas de temps, $t = T + 1$, c'est-à-dire [3]

$$y_t^{m;T} = (y_T; y_{T-\tau}; y_{T-2\tau}; ::::; y_{T-(m-1)\tau}) \tag{2.7}$$

La sélection des k m-histoires similaires au vecteur de retard est basée sur une mesure de similarité, généralement une fonction de distance telle que la distance euclidienne. Récemment, Durbin et Rumelhart ont montré que la distance de Mahalanobis est plus appropriée en raison de la corrélation possible entre les vecteurs à différentes trames temporelles, par rapport aux distances basées sur des vecteurs déterministes, comme la distance euclidienne. Ainsi, les vecteurs k historiques les plus proches qui minimisent la fonction de distance, $\Omega(y_t^{m;T}; y_t^{m;T})$, sont sélectionnés [3].

La valeur à $t = T + 1$ est alors prédite comme la moyenne (s'il n'y a pas de valeurs aberrantes, sinon la médiane) des valeurs de sortie cibles sur les k vecteurs d'historique les plus proches. D'autres méthodes simples, telles que la moyenne pondérée simple [26], et la régression linéaire locale [27], sont également utilisées [3].

Le coût d'apprentissage dans le k-NN est nul et aucune hypothèse n'est requise sur les caractéristiques des concepts à apprendre. Le k-NN utilise également des procédures simples d'approximation locale pour apprendre des concepts complexes. Cependant, les concepts appris n'ont pas de description (ce qui signifie que le modèle ne peut pas être interprété). Un autre inconvénient du k-NN est que la recherche des k plus proches voisins devient coûteuse en termes de calcul pour les grands ensembles de données. De plus, la technique souffre de la malédiction de la dimensionnalité (c'est-à-dire les propriétés non intuitives des données observées lorsqu'on travaille dans un espace à haute dimension) [3].

2.6.3 Machines à vecteurs de support (SVM)

Une machine à vecteurs de support [[28], [29]] est une méthode d'apprentissage supervisée utilisée à des fins de classification. Lorsque le SVM est utilisé pour la prévision, il est appelé régression à vecteur de support (SVR) [30]. Les SVM sont basés sur l'idée de construire un hyperplan qui sépare au mieux un ensemble de données en deux classes distinctes, comme le montre la figure 2.8 [3].
Les points de données les plus proches des deux côtés de l'hyperplan de séparation sont appelés

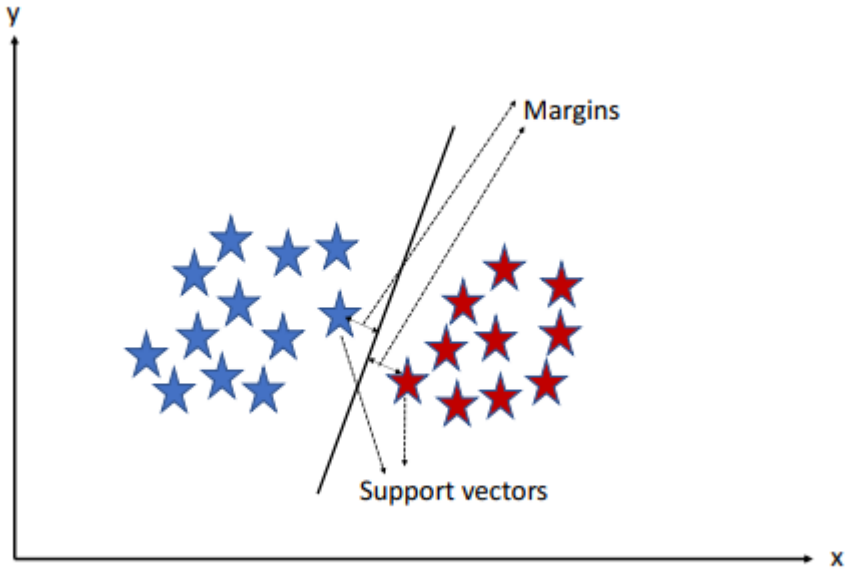


FIGURE 2.8 – Séparation des points de données par SVM [3]

vecteurs de support et sont considérés comme des éléments critiques de l'ensemble de données. Pour avoir une plus grande chances de classer correctement de nouvelles données, on choisit l'hyperplan ayant la plus grande marge possible (c'est-à-dire la distance entre les vecteurs de support des classes opposées). Ce type de classificateur linéaire avec un hyperplan à marge maximale est appelé classificateur à marge maximale [3].

Afin de classer des données linéairement non séparables, l'astuce du noyau est appliquée à l'hyperplan à marge maximale. Le kernel permet de cartographier les données dans un espace de dimension supérieure où les données sont linéairement séparables. Ainsi, pour classer des données non linéaires, un hyperespace à marge maximale est ajusté dans l'espace caractéristique transformé [3].

Un processus similaire est suivi dans le SVR. Les séries de données sont transposées dans un espace de caractéristiques de haute dimension par le biais d'un mappage non linéaire, et une régression linéaire est effectuée dans cet espace [31]. L'exécution d'une régression linéaire dans un espace de caractéristiques de haute dimension correspond à la régression non linéaire dans l'espace d'entrée de faible dimension [3].

Les SVR ont été utilisés pour prévoir les données de séries temporelles lorsque les processus sous-jacents sont typiquement non linéaires, non stationnaires et non définis a-priori [32]. Le principal avantage des SVM est qu'ils fournissent une solution unique et globalement optimale, contrairement à d'autres méthodes comme les NNs [33]. Les SVM fournissent également un

modèle précis même avec un petit ensemble de données. Cependant, le modèle a une complexité algorithmique élevée et nécessite un processus de sélection de modèle approfondi [3].

2.6.4 Réseaux neuronaux artificiels (NNs)

Les NNs sont les modèles d'intelligence artificielle les plus performants et les plus fréquemment utilisés pour la prévision des séries chronologiques [34]. Un NN traditionnel a une architecture composée de neurones artificiels interconnectés (nœuds) organisés en trois couches, c'est-à-dire des couches d'entrée, cachées et de sortie. Les NNs sont des modèles basés sur des données qui peuvent représenter n'importe quel type de fonction non linéaire [35]. Fondamentalement, un NN est une fonction complexe représentant une correspondance non linéaire entre un espace d'entrée donné et un espace de sortie. Pour une approximation précise du mappage, le NN doit être entraîné à apprendre les modèles contenus dans un ensemble de données. Même si les NNs ont donné de bons résultats dans de nombreux problèmes de prévision de séries temporelles, leurs performances dépendent de la sélection de la bonne architecture de NN et de l'algorithme d'apprentissage approprié [36], [37].

Dans les problèmes de prévision, un NN effectue la cartographie suivante :

$$y_{t+1} = f(y_1; y_2; :::; y_t) \tag{2.8}$$

où y_{t+1} est l'observation prévue produite en tant que sortie du NN et $y_1; y_2; :::; y_t$ sont les valeurs historiques de la série introduite dans le NN comme entrée. Par exemple, un NN avec neuf nœuds d'entrée et une sortie utilise les valeurs de série $y_1; y_2; :::; y_9$ pour prévoir la valeur y_{10}. La fenêtre d'apprentissage se déplace alors d'un pas pour extraire $m - 9$ exemples d'apprentissage de la série avec m observations. Le NN apprend à prévoir y_t en utilisant $y_{t-9}; :::; y_{t-1}$ comme entrée. l'entrée, et le processus se poursuit pendant plusieurs étapes d'apprentissage jusqu'à ce que la précision de la prédiction soit satisfaisante [3].

Pour une gestion explicite de l'ordre entre les observations lors de l'apprentissage d'une fonction de mise en correspondance entre l'entrée et la sortie, on utilise des NNs récurrents RNNs [38]. L'ajout de la composante temporelle de la série ajoute une nouvelle dimension à la fonction à approximer. Donc, le réseau acquiert la capacité d'apprendre la fonction de mappage entrée-sortie au fil du temps, plutôt que de simplement mapper les entrées et les sorties. Cette capacité permet de débloquer les séries temporelles pour les réseaux neuronaux [3].

2.7 Travaux à base de la prédiction des séries temporelles

Dans cette section nous allons établir une présentation des quelques travaux exploitant la prédiction des time series (série temporelles (TS) .

TABLE 2.1 – Travaux à base de la prédiction des séries temporelles

Références	Objectif	Bases de données	Méthodes utilisées
Coelho, Igor M and Coelho, Vitor N and Luz, Eduardo J da S and Ochi, Luiz S and Guimarães, Frederico G and Rios, Eyder [39]	Conception d'une unité de traitement graphique (GPU) à utiliser pour prévoir les différentes parties de la série chronologique de la charge d'un micro-réseau.	The Reference Energy Disaggregation Data Set (REDD)	modèle métaheuristique hybride , l'apprentissage en profondeur(Deep learning)
Adhikari, Ratnadip and Agrawal, Rajiv Kumar [36]	Dans cet article, deux variantes de PSO sont utilisées pour entraîner des réseaux neuronaux multicouches à action directe pour la prévision de séries chronologiques saisonnières.	la série chronologique mensuelle des passagers des compagnies aériennes, la série chronologique trimestrielle des ventes et la série chronologique trimestrielle de la production de bière aux États-Unis.	Réseaux neuronaux basés sur PSO
Al-Douri, Yamur K and Al-Chalabi, Hussan and Lundberg, Ja [40]	évaluer les performances de la programmation génétique dans la prévision des réponses des patients atteints de PR (polyarthrite rhumatoïde) à ces traitements biologique.	mackey glass time series	algorithme de réseau neuronal à rétropropagation

2.8 Conclusion

Ce chapitre a donné un bref aperçu des séries temporelles. Nous avons défini une série temporelle, en donnant quelques exemples, et discuté les caractéristiques d'une série temporelle. Ces caractéristiques sont la tendance, la saisonnalité, les variations cycliques et aléatoires. Ces composantes peuvent être combinées de différentes manières dans une série chronologique.

L'analyse des séries temporelles est réalisée afin d'ajuster un modèle qui décrit adéquatement une série temporelle. Certaines applications de l'analyse des séries chronologiques comprennent la prévision, la classification, l'explication et la transformation. Dans le cas de la prévision, les valeurs historiques des séries temporelles sont analysées pour développer un modèle mathématique qui capture le processus sous-jacent de génération de données de la série, qui est ensuite utilisé pour prédire les valeurs futures de la série. Cependant, les valeurs futures ne peuvent être prédites avec certitude dans la plupart des séries chronologiques en raison d'erreurs résiduelles.

Enfin, nous avons discuté les méthodes de prévision des séries temporelles les plus importantes et les plus utilisées, notamment ARIMA, k-NN, SVM et NNs.

Nous avons présenté aussi quelques travaux exploitant la prédiction des TS .

Dans le chapitre prochain nous allons tester notre algorithme CALM et présenter son application.

Chapitre 3

Évaluation et contribution de l'approche proposée

Chapitre 3

Évaluation et contribution de l'approche proposée

3.1 Introduction

Dans ce chapitre, nous allons décrire deux contributions, dans la première nous allons tester l'algorithme CALM sur des fonctions de norme, ensuite, comparer les résultats obtenues avec les résultats d'autres méthodes, la seconde contribution consiste à l'expérimentation des séries temporelles pour la régression des situations futures.

3.2 La première expérimentation

Dans ce qui suit, nous présentons l'organigramme fonctionnel de la méthode CALM.

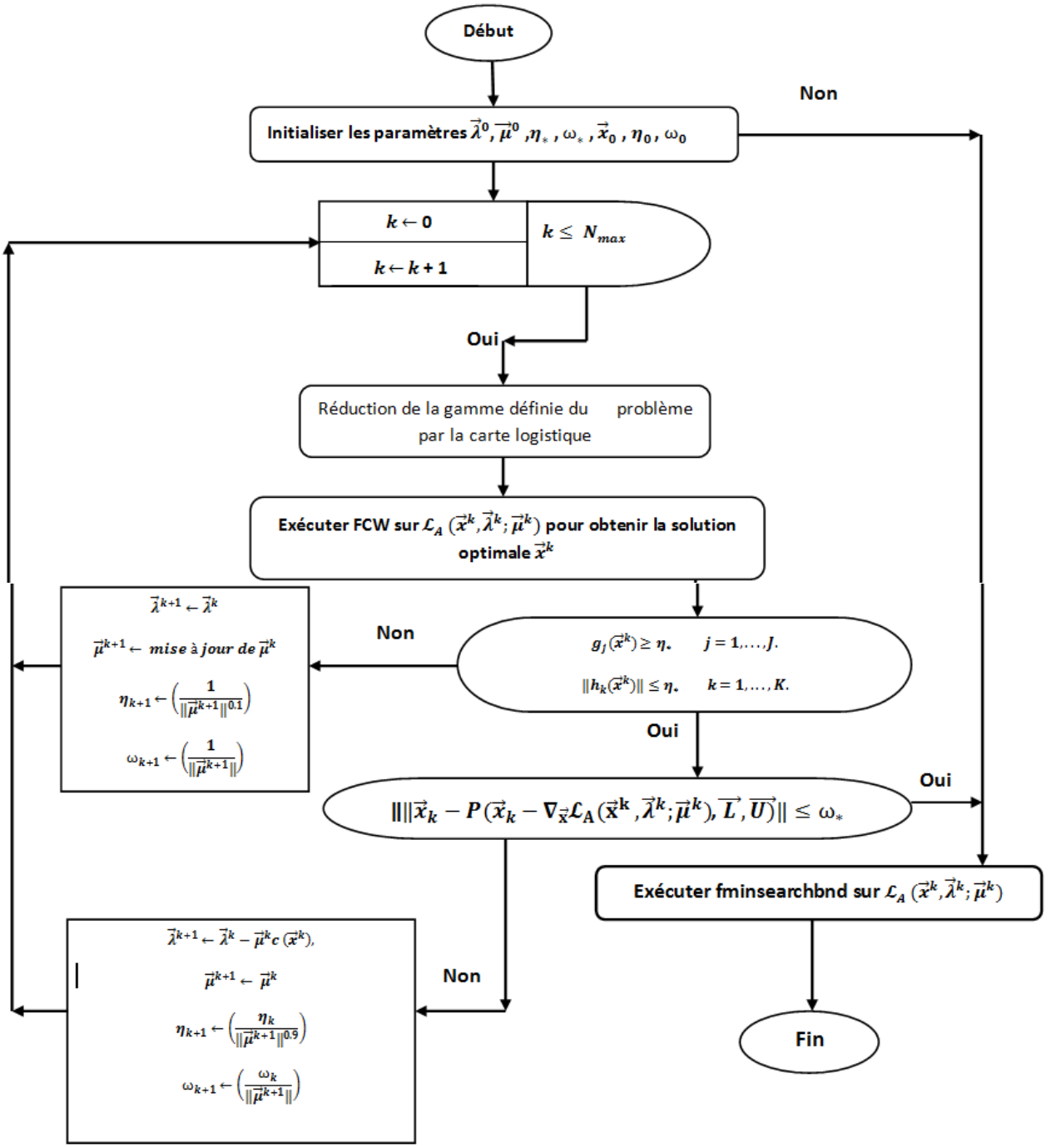


FIGURE 3.1 – Organigramme de l'algorithme CALM

Chaque nouvelle technique métaheuristique doit être validée par des fonctions de test où l'optimum global est connu a priori et le nombre d'itérations est fixé pour comparer les résultats avec d'autres techniques. Pour les métaheuristiques à base de population, la taille de la population est toujours fixée à 100 solutions candidate pour toutes les 23 fonctions. Ces fonctions de test sont classées en trois groupes . [**?**]

3.2.1 Tableau des expressions des fonctions de test utilisées

Dans le tableau 3.1 nous dressons une liste des différentes fonctions utilisées pour le test de fonctionnement.

Cette liste de fonction nous a permis de vérifier la robustesse et la performance de la solution algorithmique proposée concernant la métaheuristique CALM.

L'ensemble des fonctions est divisée en 3 catégories à savoir :

• Fonctions unimodales à haute dimension : F1 au F7 permet de tester l'exploitation des méthodes pour l'obtention des meilleures solutions ;

• Fonctions multimodales à haute dimension : F8 au F13

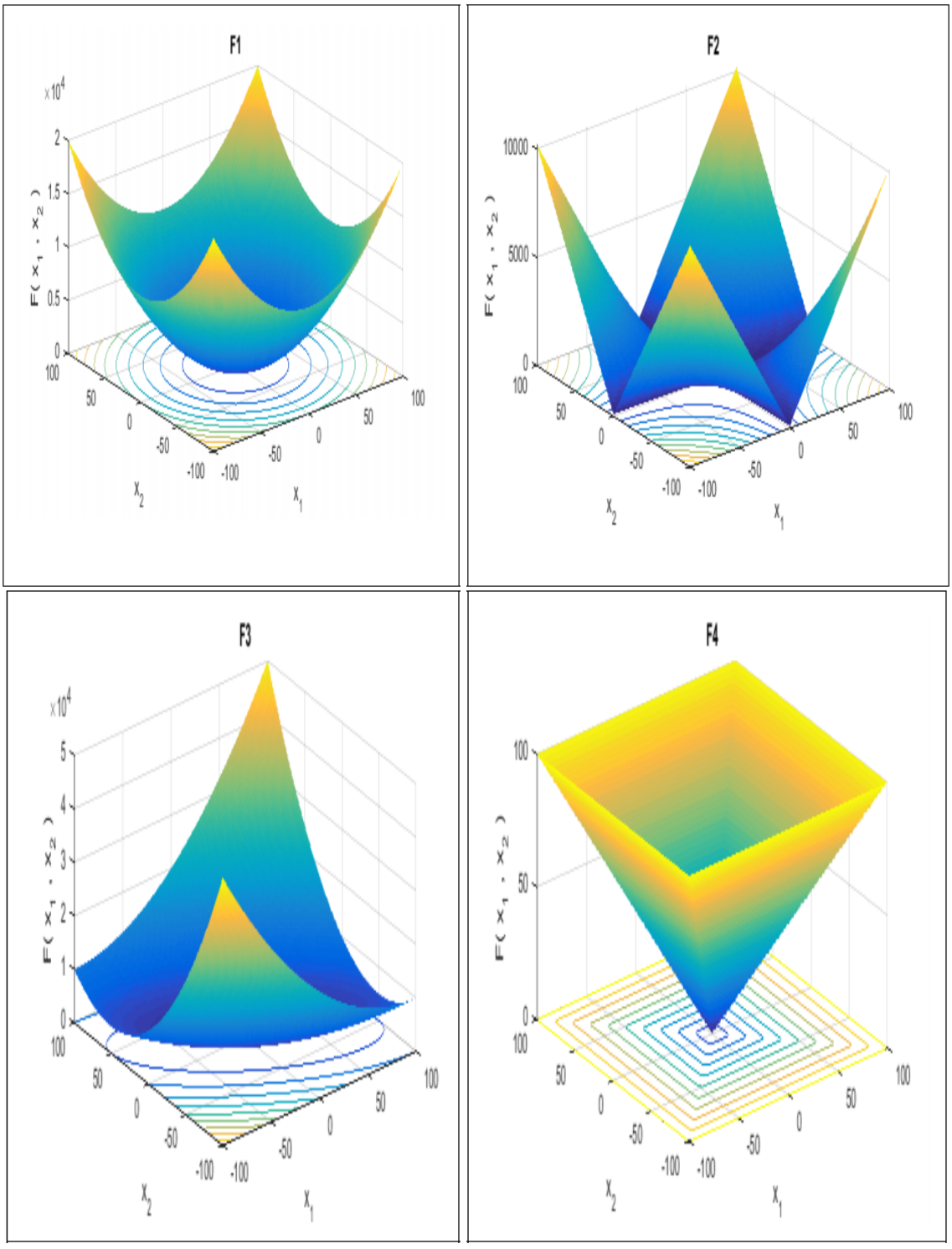
• Fonctions multimodales à faible dimension : F14 au F23

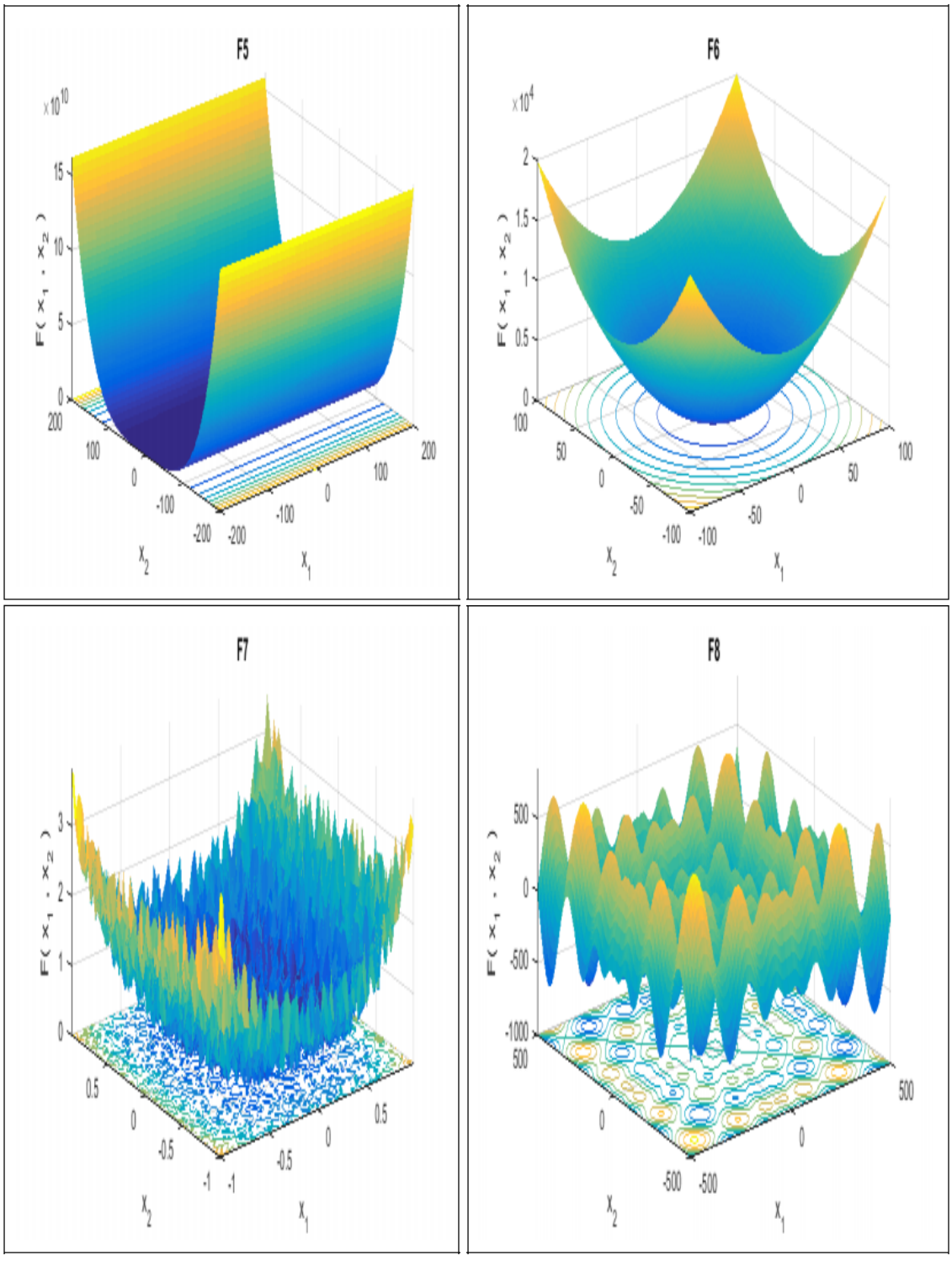
TABLE 3.1 – Tableau des fonctions de test

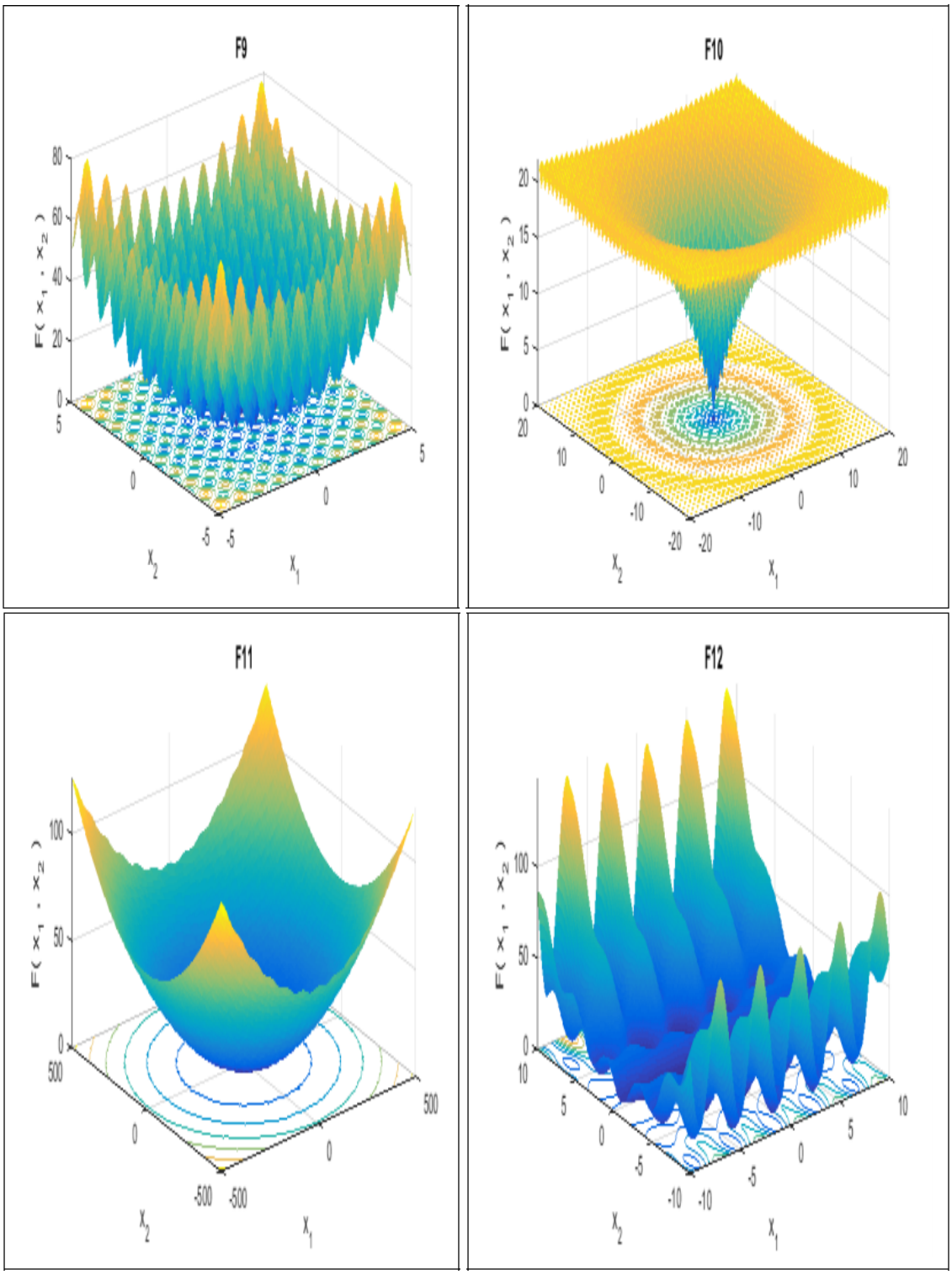
fonctions de test	D	L'intervalle	Optimum
$f_{01} = \sum_{i=1}^{n} x_i^2$	30	$[-100, +100]$	0
$f_{02} = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, +10]$	0
$f_{03} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	$[-100, +100]$	0
$f_{04} = max_i \{ \lvert x_i \rvert, 1 \leq i \leq D \}$	30	$[-100, +100]$	0
$f_{05} = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, +30]$	0
$f_{06} = \sum_{i=1}^{D} ([x_i + 0.5])^2$	30	$[-100, +100]$	0
$f_{07} = \sum_{i=1}^{D} i x_i^4 + random\ [0,1]$	30	$[-1.28, +1.28]$	0
$f_{08} = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	$[-500, +500]$	-418.9829 * n
$f_{09} = \sum_{i=1}^{D} [x_i^2 - 10 \cos(2\pi x_i) + 10]$	30	$[-5.12, +5.12]$	0
$f_{10} = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right) - \exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + e$	30	$[-32, +32]$	8.8818e-16
$f_{11} = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, +600]$	0
$f_{12} = \frac{\pi}{D} \left\{ 10 \sin^2(\pi y_i) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_i + 1)] + (yD - 1)^2 + \sum_{i=1}^{D} u(x_i, 10, 100, 4) \right\}$ $y_i = 1 + \frac{x_i + 1}{4}, u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i > a \\ 0 \quad -a < x_i < a \\ k(-x_i - a)^m x_i < -a \end{cases}$	30	$[-50, +50]$	1.5705e-032
$f_{13} = 0.1 \left\{ 10 \sin^2(\pi y_i) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_i + 1)] + (yD - 1)^2 + \sum_{i=1}^{D} u(x_i, 10, 100, 4) \right\}$	30	$[-50, +50]$	0
$f_{14} = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i-1}^{2} (x_i - a_{ij})^6} \right]^{-1}$	2	$[-65.53, +65.53]$	0.998004
$f_{15} = \sum_{i=1}^{11} \left[a_i - \frac{x_1 (b_i^2 + b_i x_i)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	$[-5, +5]$	0.0003075
$f_{16} = 4x_1^2 - 2.1x_i^4 + \frac{1}{3} x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, +5]$	-1.0316285
$f_{17} = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) cos x_1 + 10$	2	$[5, +10] * [0, +15]$	0.398
$f_{18} = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2)]$	2	$[-5, +5]$	3
$f_{19} = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij} (x_j - p_{ij})^2 \right)$	3	$[0, +1]$	-3.86
$f_{20} = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij} (x_j - p_{ij})^2 \right)$	6	$[0, +1]$	-3.32
$f_{21} = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, +10]$	-10.1532
$f_{22} = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, +10]$	-10.4029
$f_{23} = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, +10]$	-10.5364

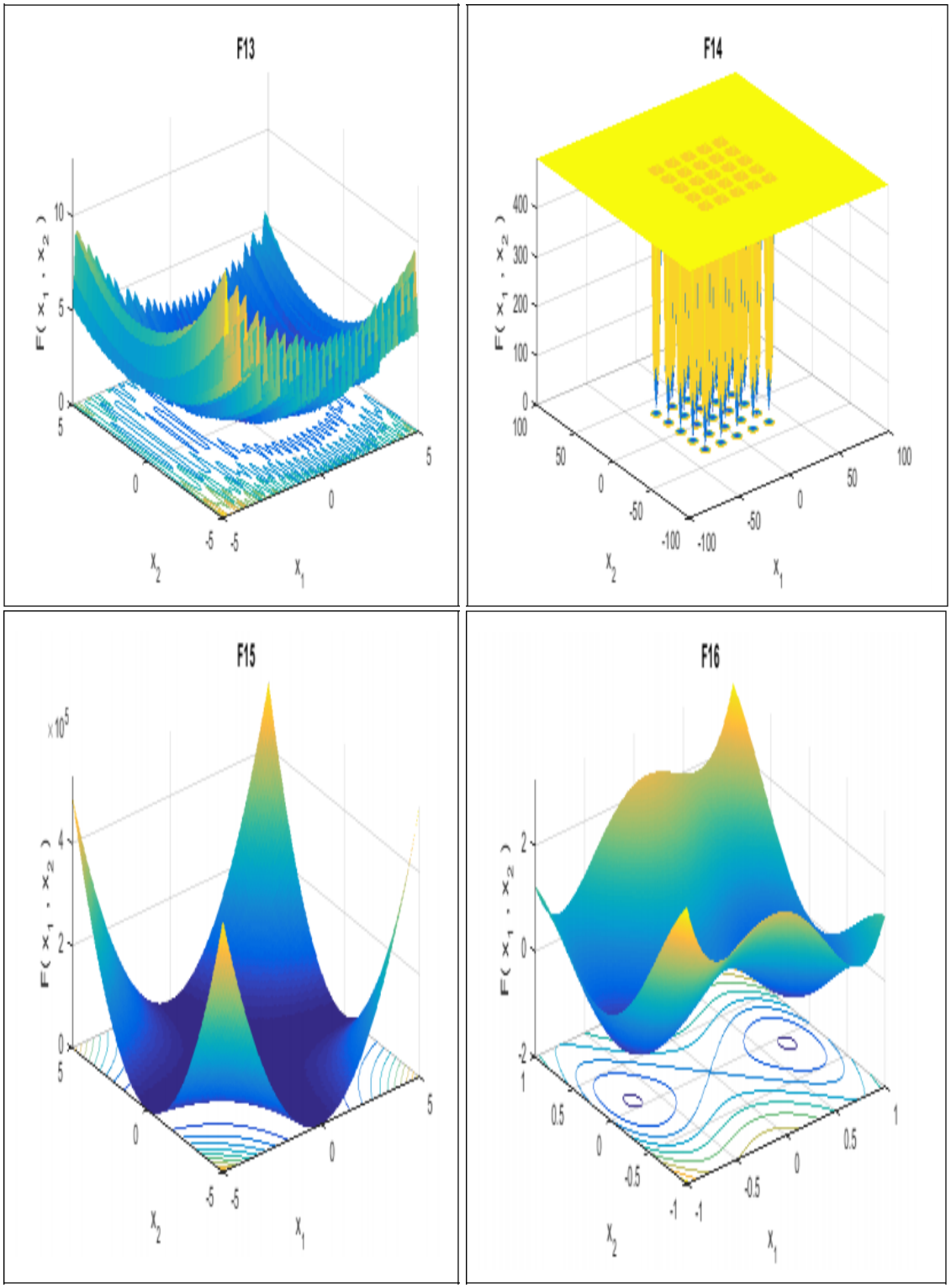
3.2.2 Trace 3D de chaque fonction

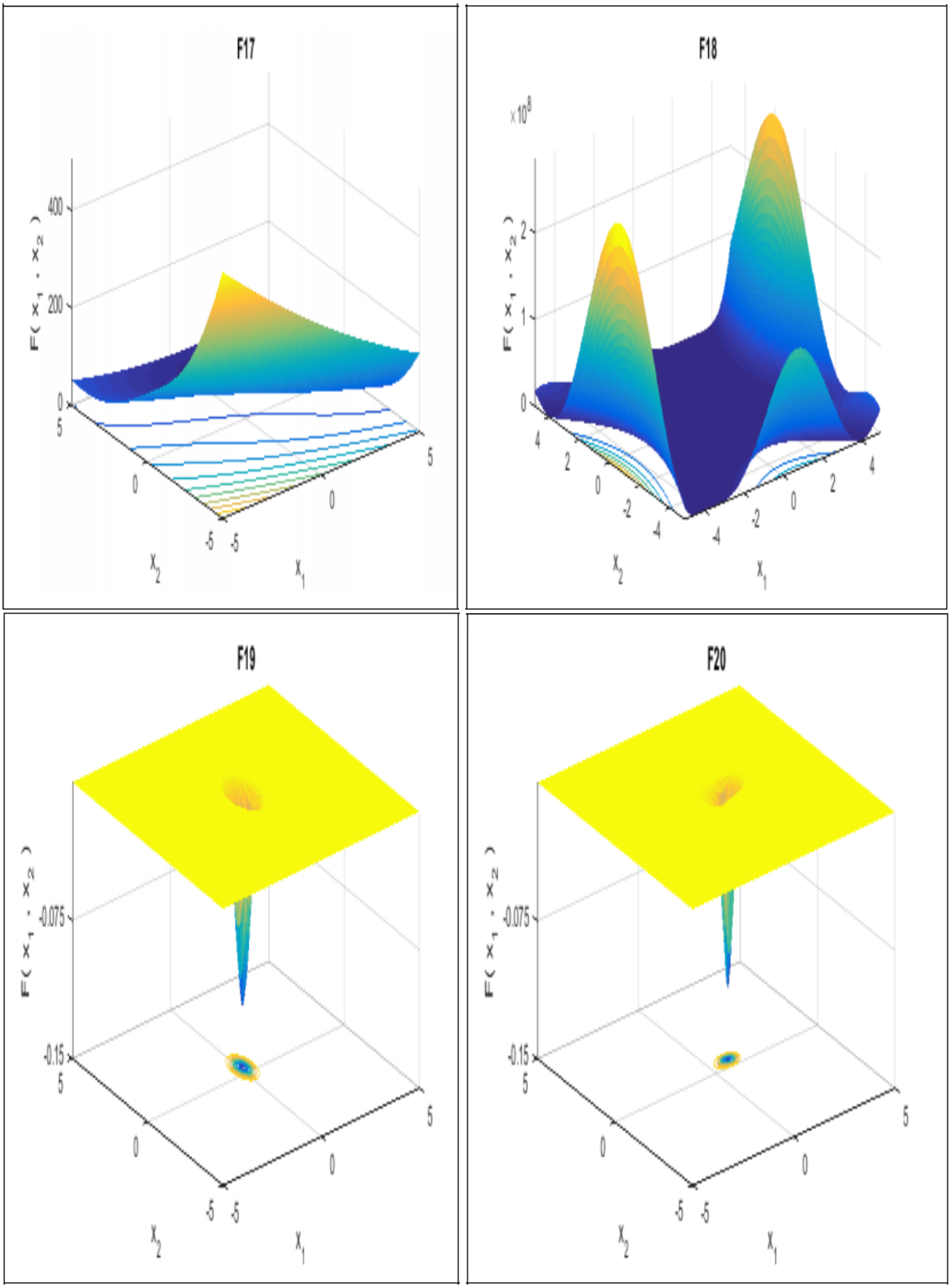
Pour bien présenter notre évaluation, nous dressons dans cette partie les aspects des fonctions en 3D, Montrent le nombre d'optimum locaux et globaux de chaque fonction.




F5
F6
F7
F8


F9
F10
F11
F12
F(x1 , x2)
X1
X2


F13
F14
F15
F16
F(x1 , x2)
X1
X2


F17
F18
F19
F20
F(x1 , x2)
x10^8
X1
X2

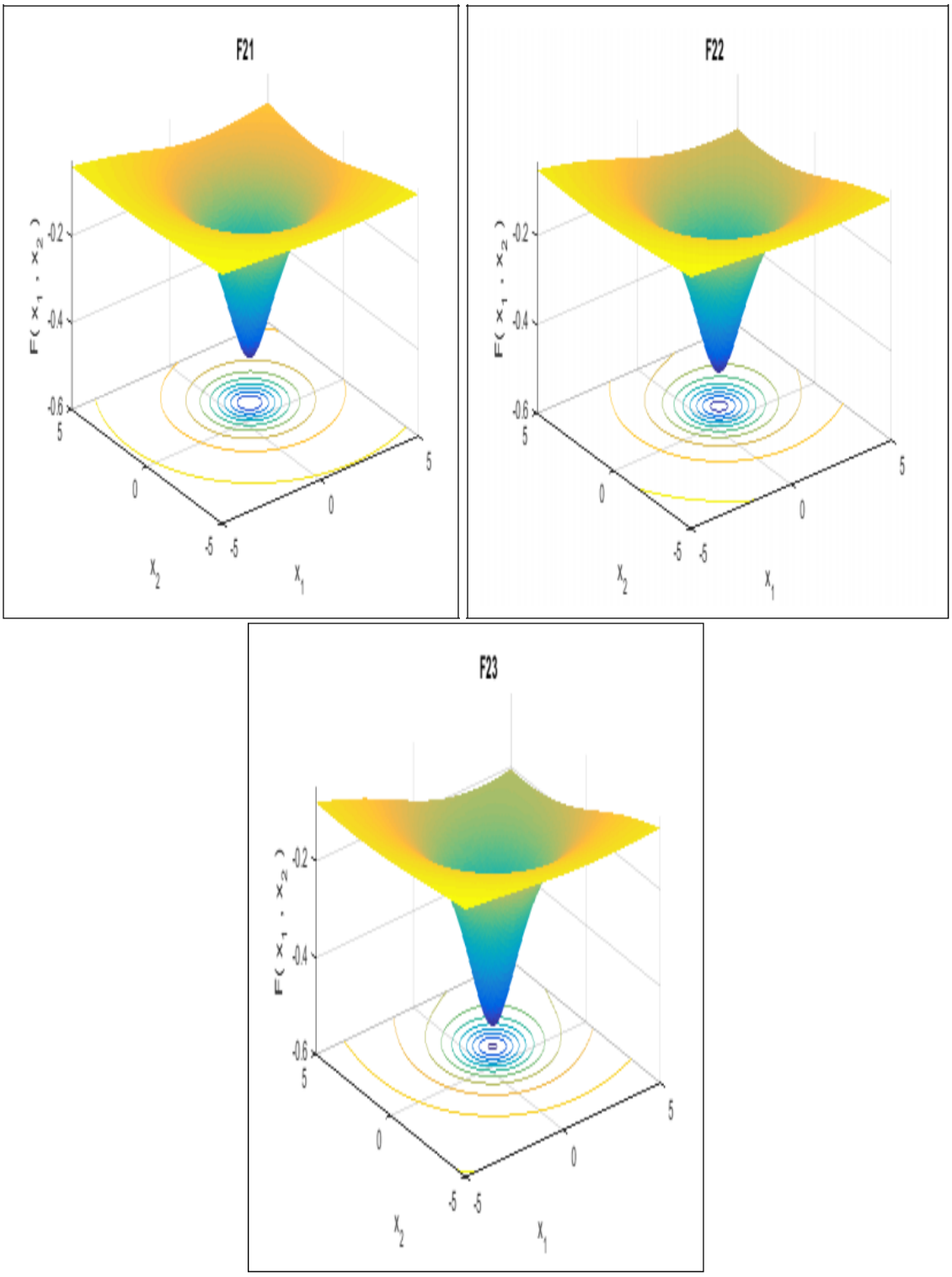


FIGURE 3.3 – Liste des figures des courbes 3D pour 23 fonctions [4]

3.2.3 Tableau comparatif des résultats

Pour avoir une bonne appréciation de notre solution, nous avons comparé la méthode CALM avec d'autres méthodes : PSO, CA, ABC, SCA, GWO,SOS. Ces résultats sont montrés dans le Tableau 3.2. Pour l'évaluation de chaque fonction, nous avons utilisés trois variables qui sont la moyenne, l'écart type et le classement de chaque méthode. Ce classement est obtenu sur la base de la meilleure moyenne des valeurs des fonctions objectives et les meilleurs écarts type par rapport à la valeur optimale de la fonction. Dans la catégorie unidimensionnelle la méthode CALM donne

du F1 au F3 de très bons résultats par rapport aux autres méthodes sauf pour la fonction F5 pour laquelle CALM a donné une solution différente de l'optimum de même pour les autres méthodes. Pour la catégorie des multidimensionnelles (petite et haute), notre méthode a donné pour certaines valeurs des résultats exactes (optimales), pour d'autres, le résultat le plus proche et quelque uns des solutions différentes.Nous avons constaté que notre méthode est efficace dans l'exploitation c'est-à-dire elle génère les meilleures solutions, mais pour l'exploration (F13…F22) donne des solutions entre bonnes et acceptables.

TABLE 3.2 – Résultats d'évaluation des méthodes

Fonctions		Optimum	PSO	ABC	CA	SCA	GWO	SOS	CALM
F1	Moy	0	0.3970	0.1336	1.2946e-04	1.2728	8.1334e-41	0	0
	EcTy		0.3209	0.0324	1.3350e-04	3.3294	1.1758e-40	0	0
	Class		5	4	3	6	2	1	1
F2	Moy	0	0.0352	0.1752	59.2268	2.3977e-06	8.0399e-47	0	0
	EcTy		0.1624	0.0990	20.9776	6.6258e-06	8.3453e-47	0	0
	Class		4	5	6	3	2	1	1
F3	Moy	0	1.6452e+03	8.4519e+03	2.6431e+04	3.8001e+03	2.8663e-11	0	4.1433e-49
	EcTy		640.9846	1.7992e+03	4.6212e+03	3.0736e+03	7.8706e-11	0	2.0717e-48
	Class		4	6	7	5	3	1	2
F4	Moy	0	2.1999	12.3781	3.1395	5.3804	1.3831e-21	0	2.5275
	EcTy		1.7124	1.6148	3.2499	5.9093	2.8568e-21	0	12.6375
	Class		3	6	4	5	2	1	7
F5	Moy	0	33.0197	27.8865	48.5678	27.0538	25.8985	28.8209	43.0704
	EcTy		31.9313	0.4019	115.9528	0.7684	0.7736	0.0334	215.3519
	Class		5	3	6	2	1	4	7
F6	Moy	0	2.7682e+04	3.3316e+04	1.5645e+04	3.1148e+04	1.2398e+03	2.9951	0.0317
	EcTy		4.9476e+03	5.1635e+03	3.3031e+03	5.9109e+03	330.2118	0.7141	0.1587
	Class		5	7	4	6	3	2	1
F7	Moy	0	0.0251	0.0311	0.0701	0.0086	1.9404e-04	1.6e-03	3.4590e-04
	EcTy		0.0170	0.0079	0.0177	0.0085	9.6654e-05	2.2e-03	0.0017
	Class		5	6	7	4	1	3	2
F8	Moy	-4180.9829*n	-9.9460e+03	-9.1745e+32	-1.0400e+04	-4.3341e+03	-6.3427e+03	-6.5835e+03	-0.3227e+03
	EcTy		514.9218	2.1229e+33	669.8460	285.6626	818.9499	666.3787	1.6136e+03
	Class		3	1	2	6	5	4	7
F9	Moy	0	275.2287	308.3371	316.9494	213.5441	47.6089	0	0
	EcTy		17.2402	20.9358	18.6367	47.0853	15.1017	0	0
	Class		4	5	6	3	2	1	1
F10	Moy	8.8818e-16	13.4428	16.4168	11.1855	16.7991	0.0844	8.8818e-16	0.0355e-15
	EcTy		0.7387	0.7822	0.8800	3.8739	0.0292	0	0.1776e-15
	Class		5	6	4	7	3	1	2
F11	Moy	0	43.5010	57.5828	18.1817	36.2411	0.0848	0	0

	EcTy		11.9431	9.5749	4.6308	15.6611	0.0495	0	0
	Class		5	6	3	4	2	1	1
F12	Moy	1.5705e-032	0.0456	0.5025	4.1986	0.3651	0.0120	0.1628	0.0035
	EcTy		0.0902	0.1092	5.9838	0.0549	0.0119	0.0616	0.0174
	Class		3	6	7	5	1	4	2
F13	Moy	0	0.0040	0.1013	0.0062	2.0343	0.2046	2.6935	0.0391
	EcTy		0.0095	0.0132	0.0096	0.1064	0.1377	0.6036	0.1957
	Class		1	3	2	6	5	7	4
F14	Moy	0.998004	0.9980	1.0297	1.1561	1.7920	3.3878	0.9980	0.2875
	EcTy		4.3578e-13	0.1571	0.7905	0.9918	3.8337	0	1.4373
	Class		2	3	5	6	7	1	4
F15	Moy	0.0003075	0.0036	0.0010	0.0044	9.5601e-04	0.0019	3.4411e-04	6.4115e-04
	EcTy		0.0075	3.3967e-05	0.0122	3.9461e-04	0.0056	1.8314e-04	0.0032
	Class		6	3	7	2	5	1	4
F16	Moy	-1.0316285	-1.0316	-1.0315	-1.0316	-1.0316	-1.0316	-1.0316	-0.0412
	EcTy		7.0682e-13	7.6299e-05	6.2476e-16	5.2826e-05	1.6969e-05	6.7987e-16	0.2062
	Class		3	6	2	5	4	1	7
F17	Moy	3	3.0000	3.0001	3.000	3.0000	3.0000	3.0000	8,1712
	EcTy		1.8965e-12	1.8513e-04	1.3628e-15	9.3354e-05	2.1140e-05	0.5368e-16	6,2150
	Class		3	6	2	5	4	1	7
F18	Moy	-3.86	-3.8628	-3.8625	-3.8628	-3.8537	-3.8612	-3.8628	-3.7320
	EcTy		1.0538e-07	2.3838e-04	1.8467e-15	0.0037	0.0025	2.2662e-15	0.0974
	Class		3	4	1	6	5	2	7
F19	Moy	-3.32	-3.2602	-3.2515	-3.2792	-3.0135	-3.2608	-3.2839	-2.5889
	EcTy		0.0606	0.0403	0.0582	0.2057	0.0732	0.0566	0.3707
	Class		4	5	2	6	3	1	7
F20	Moy	-3.32	-6.9258	-10.1018	-7.7627	-3.2472	-9.9496	-8.7258	-3.1823
	EcTy		3.0358	0.1039	3.5566	1.6226	1.0105	2.3362	1.6710
	Class		5	1	4	6	2	3	7
F21	Moy	-10.4029	-7.8961	-10.3587	-7.5721	-4.5168	-10.4009	-8.2768	-2.7180
	EcTy		3.4918	0.0345	3.5908	1.4180	0.0011	2.6576	0.7956
	Class		4	2	5	6	1	3	7
f22	Moy	-10.5364	-8.5370	-10.4930	-7.7253	-4.4472	-10.3185	-9.0222	-4.6398
	EcTy		3.3054	0.0230	3.6025	0.9653	1.0813	2.4782	2.3421
	Class		3	5	4	6	2	1	7

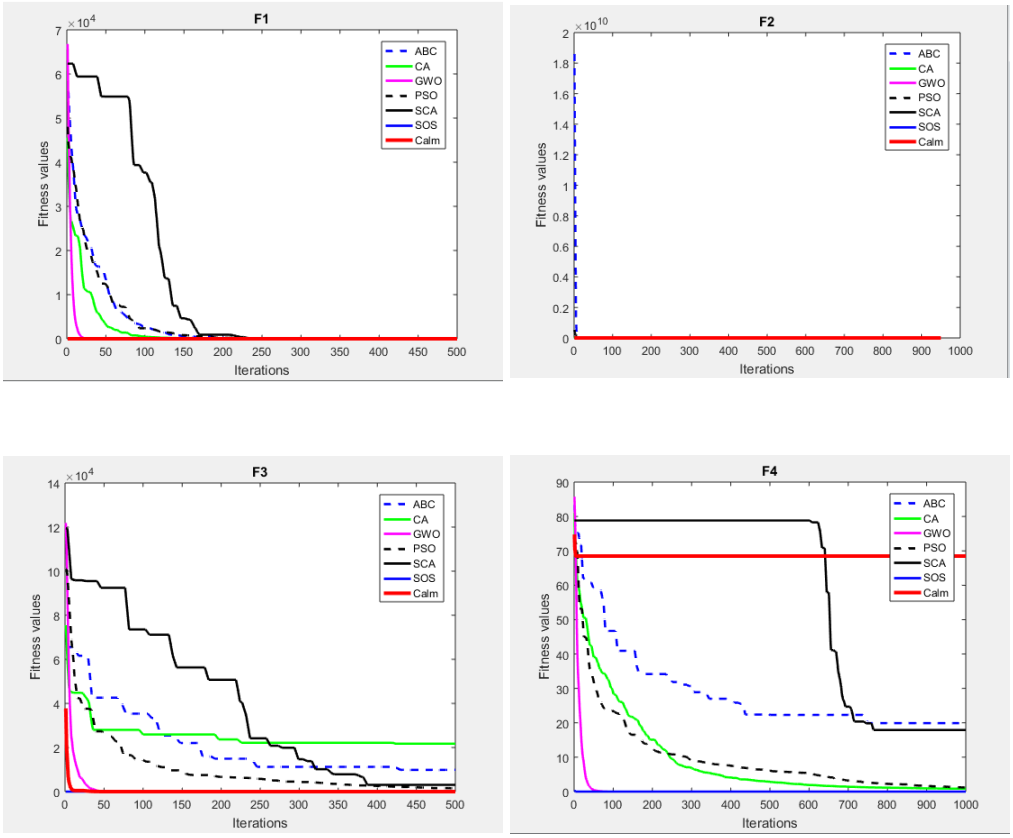
3.2.4 Courbes de Fitness

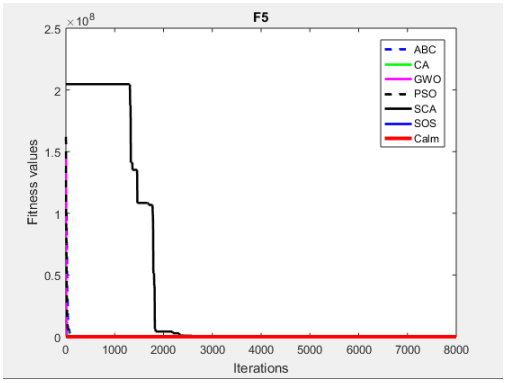
Pour plus de performance de cette métaheuristique, on présente les tracés des courbes des fonctions Fitness de test. La figure 3.20 montre les courbes de convergence obtenues par l'algorithme CALM sur 22 fonctions.

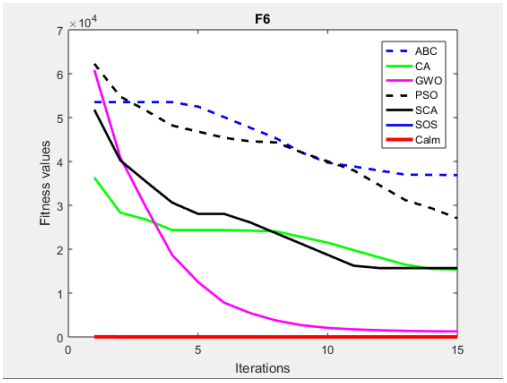
Les valeurs indiquées sur ces figurent représentent les résultats de fitness des 22 fonctions selon CALM, SOS, PSO, ABC, CA, GWO et SCA.

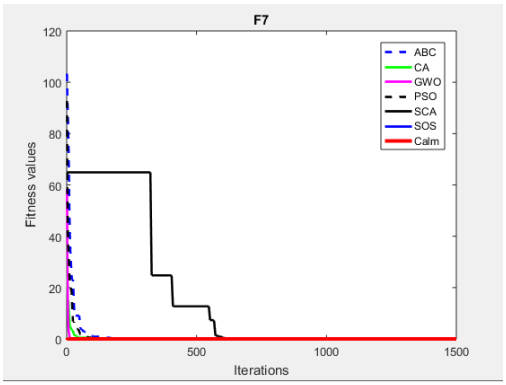
A partir de ces figures, il est clair que notre méthode a une vitesse de convergence plus rapide que les six autres méthodes.

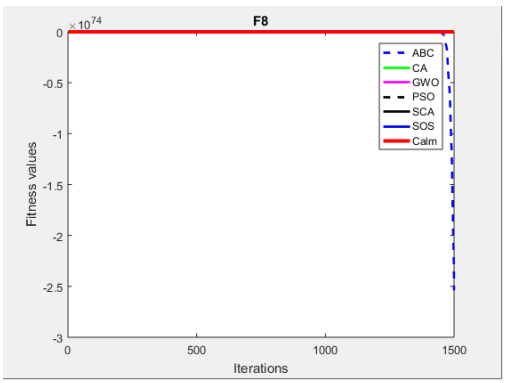
Pour les fonctions des deux types unimodale et multimodale à haute dimension notre algorithme CALM possède une convergence rapide et une approximation idéale, mais pour les fonctions du type multimodale à faible dimension, le comportement de CALM était faible parce que l'optimum global est lié à la dimension de chaque fonction donc c'est difficile de contrôler la méthode pour trouver l'optimum global .

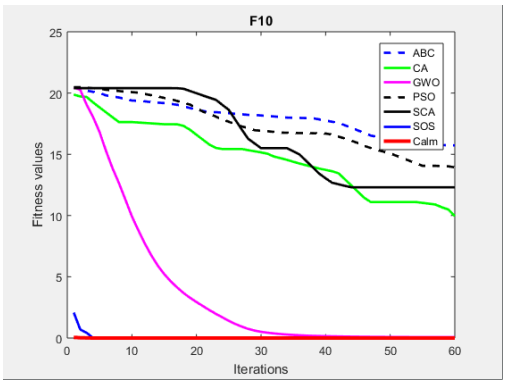


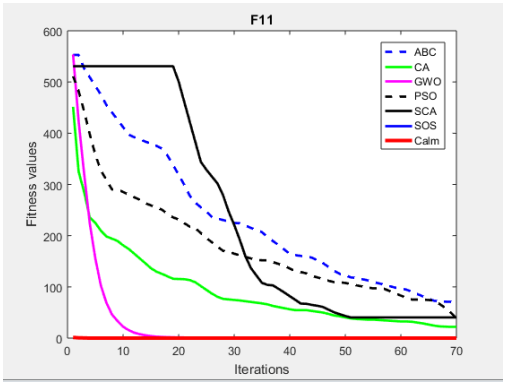

F5
×10^8
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm

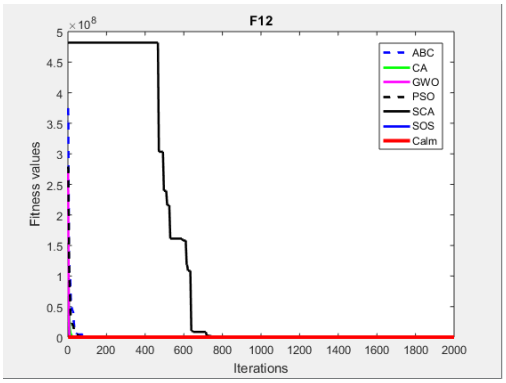

F6
×10^4
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm

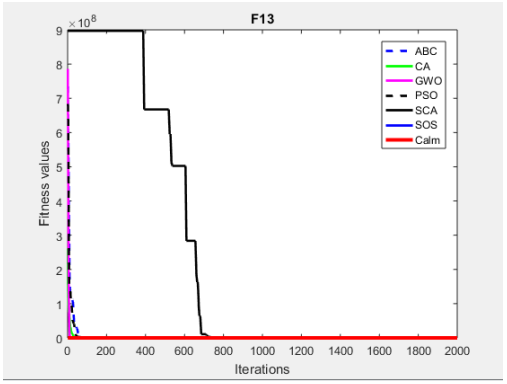

F7
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm

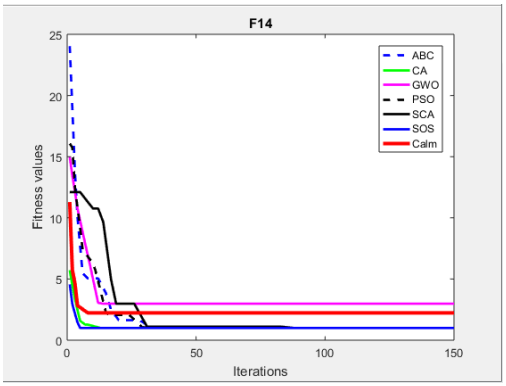

F8
×10^74
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm

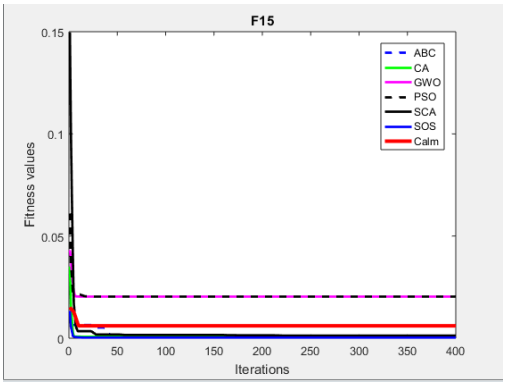

F10
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm

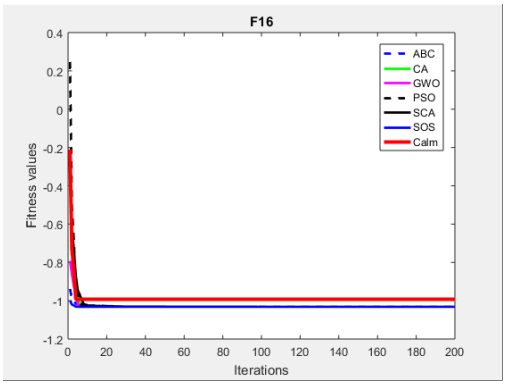

F11
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm

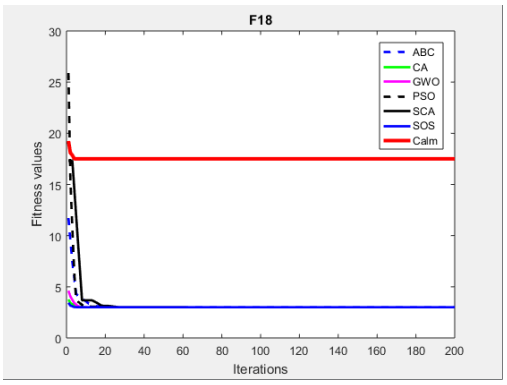

F12
×10^8
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm

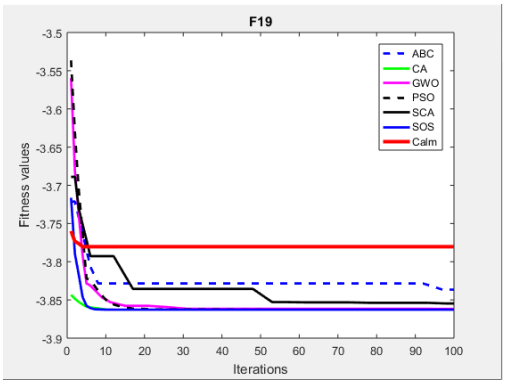

F13
×10^8
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm

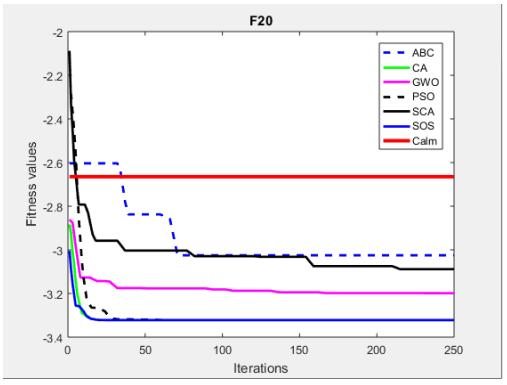

F14
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm

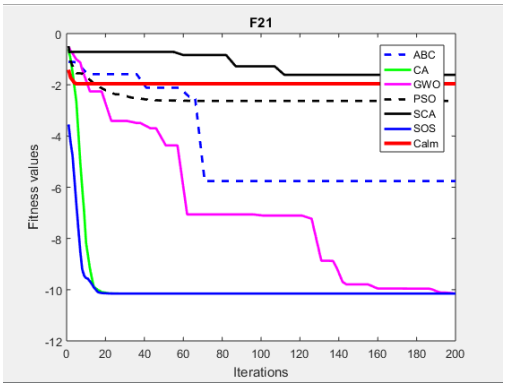

F15
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm

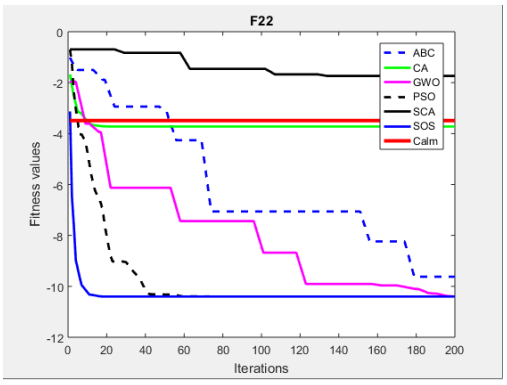

F16
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm


F18
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm


F19
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm


F20
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm


F21
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm


F22
Fitness values
Iterations
ABC
CA
GWO
PSO
SCA
SOS
Calm

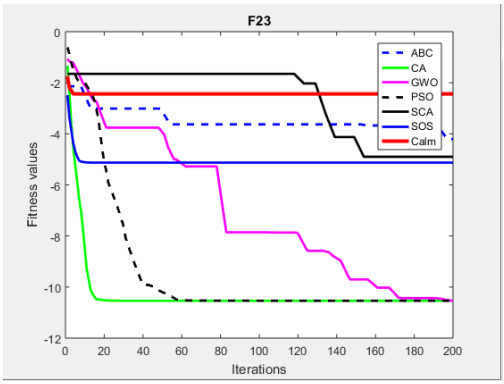


FIGURE 3.14 – Liste des figures des courbes de Fitness de 23 fonctions de test pour 7 méthodes appliquées.

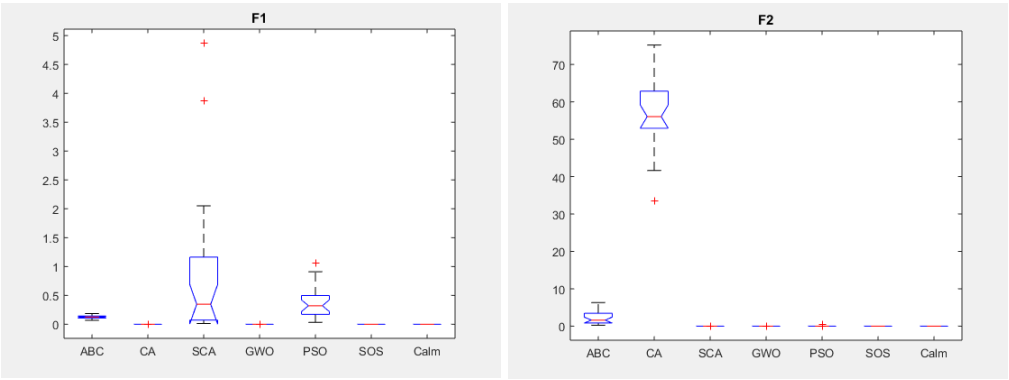
3.2.5 Analyse de variation ANOVA

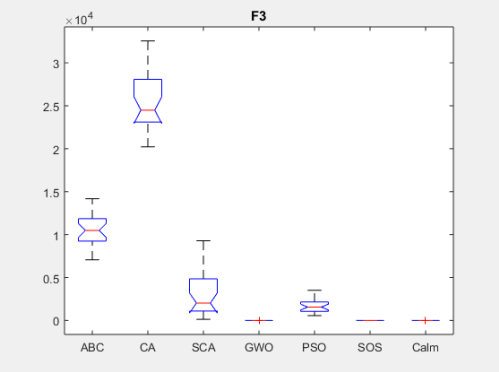
Pour expliciter notre évaluation de la méthode proposée par rapport aux autres algorithmes de comparaison, nous avons utilisés le modèle de la Figure 3.32 ≪ Test ANOVA ≫ qui est une représentation schématique de la distribution d'une variable.

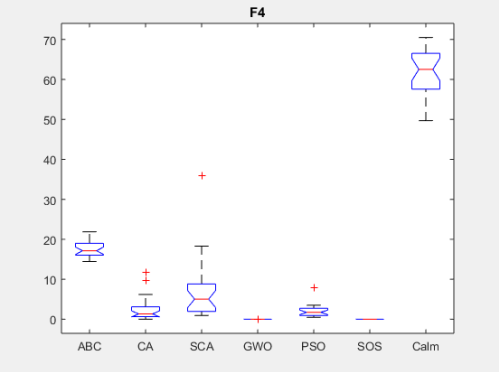
Le test Anova inventée en 1977 par John Tukey, appelée aussi diagramme en boîte, boîte de Tukey ou box-and-whisker plot en anglais, est utilisée pour représenter le schéma essentiel d'une série statistique quantitative.

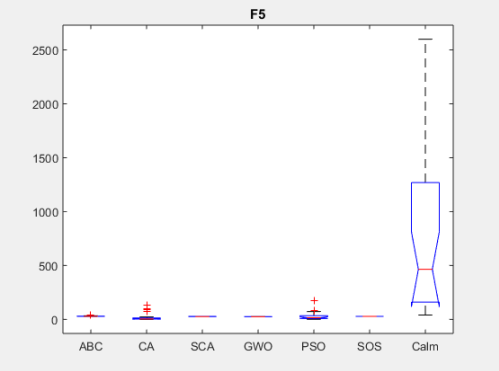
Le test Anova qui on effectué sur les 22 fonctions de test donne le résultat de 25 exécution indépendantes de notre méthode.

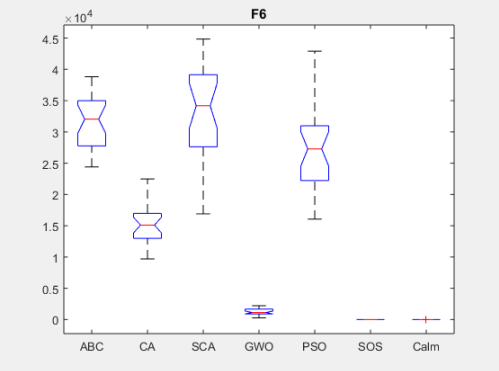
Les formes dans les figures obtenues par ce test qui sont des boites, représentent les variations des solutions optimales entre les valeurs minimales et maximales, centrée par la moyenne qui est la solution optimale de la fonction sujette de test. Nous remarquons que le test ANOVA confirme les résultats obtenus par les courbes de fitness, pour les fonctions de F1 à F12 notre méthode est très stable cela prouve qu'il génère une bonne performance avec une nette précision. Dans les autres cas l'algorithme CALM se comporte de manière acceptable.

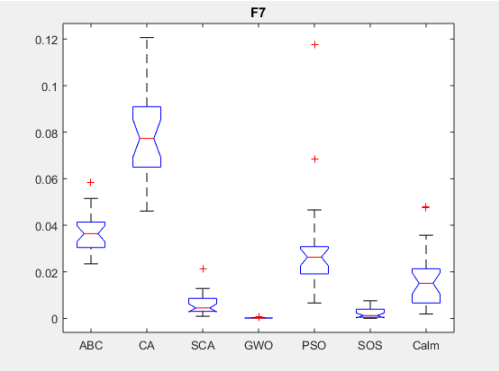


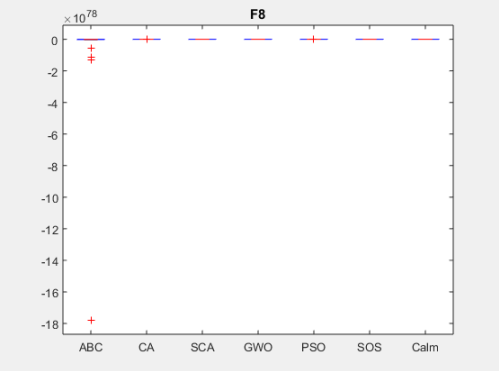

F3
×10^4
3
2.5
2
1.5
1
0.5
0
ABC
CA
SCA
GWO
PSO
SOS
Calm

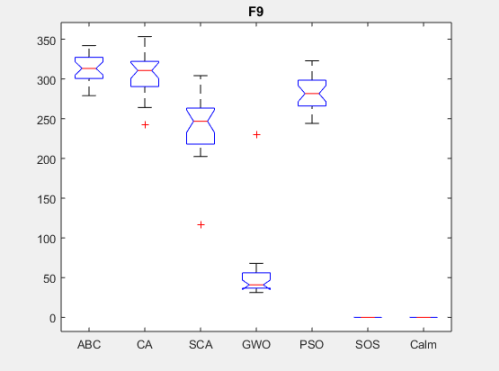

F4
70
60
50
40
30
20
10
0
ABC
CA
SCA
GWO
PSO
SOS
Calm

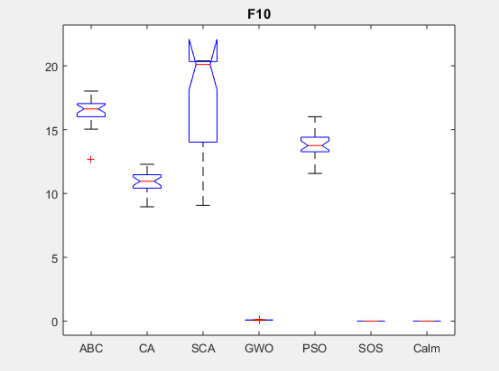

F5
2500
2000
1500
1000
500
0
ABC
CA
SCA
GWO
PSO
SOS
Calm

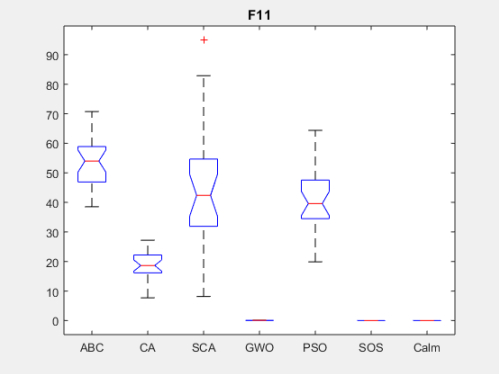

F6
×10^4
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
ABC
CA
SCA
GWO
PSO
SOS
Calm

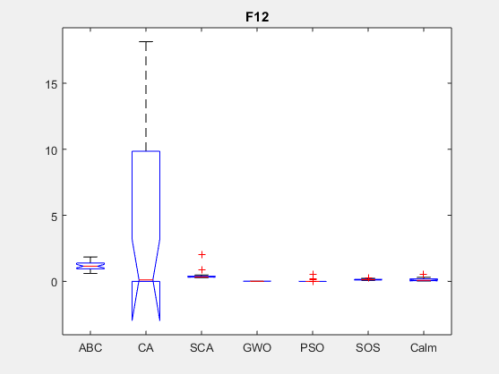

F7
0.12
0.1
0.08
0.06
0.04
0.02
0
ABC
CA
SCA
GWO
PSO
SOS
Calm

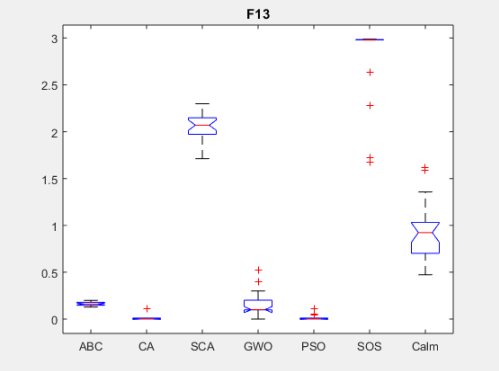

F8
×10^78
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
ABC
CA
SCA
GWO
PSO
SOS
Calm

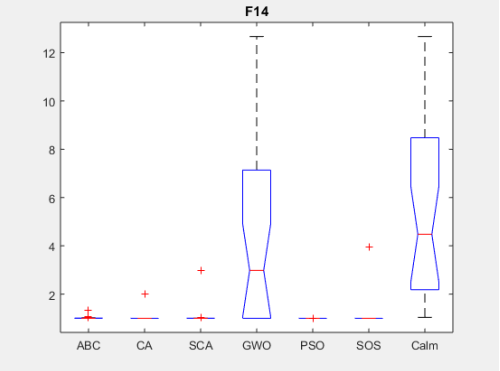

F9
350
300
250
200
150
100
50
0
ABC
CA
SCA
GWO
PSO
SOS
Calm

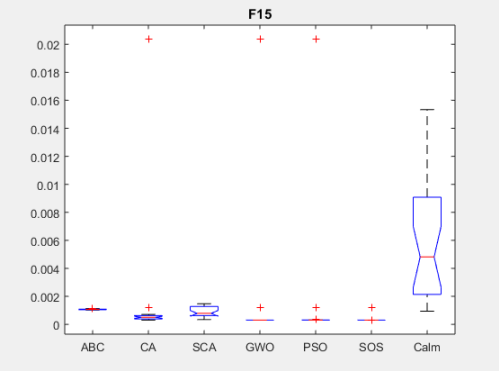

F10
20
15
10
5
0
ABC
CA
SCA
GWO
PSO
SOS
Calm

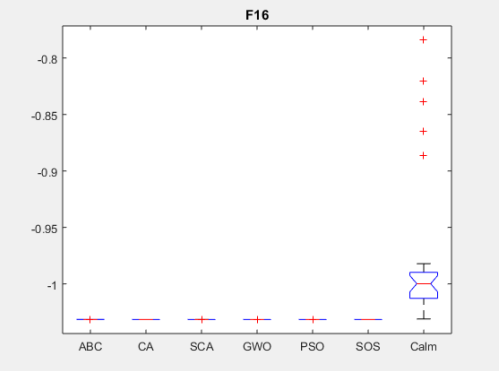

F11
90
80
70
60
50
40
30
20
10
0
ABC
CA
SCA
GWO
PSO
SOS
Calm

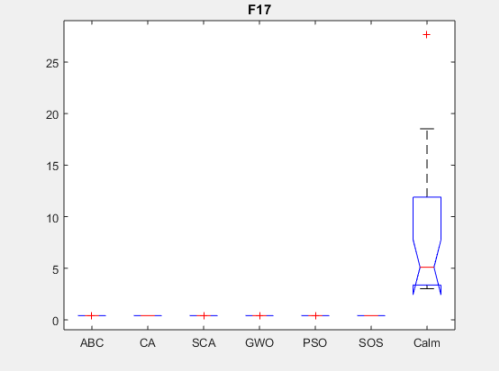

F12
15
10
5
0
ABC
CA
SCA
GWO
PSO
SOS
Calm

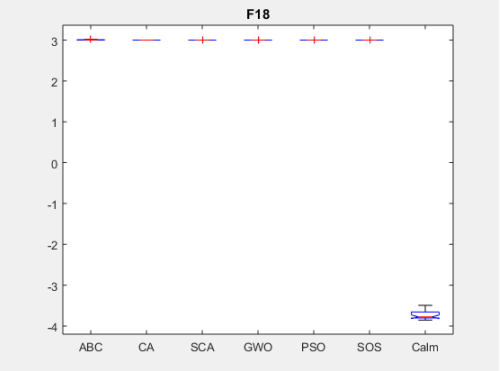

F13
3
2.5
2
1.5
1
0.5
0
ABC
CA
SCA
GWO
PSO
SOS
Calm


F14
12
10
8
6
4
2
ABC
CA
SCA
GWO
PSO
SOS
Calm


F15
0.02
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
ABC
CA
SCA
GWO
PSO
SOS
Calm


F16
-0.8
-0.85
-0.9
-0.95
-1
ABC
CA
SCA
GWO
PSO
SOS
Calm


F17
25
20
15
10
5
0
ABC
CA
SCA
GWO
PSO
SOS
Calm


F18
3
2
1
0
-1
-2
-3
-4
ABC
CA
SCA
GWO
PSO
SOS
Calm

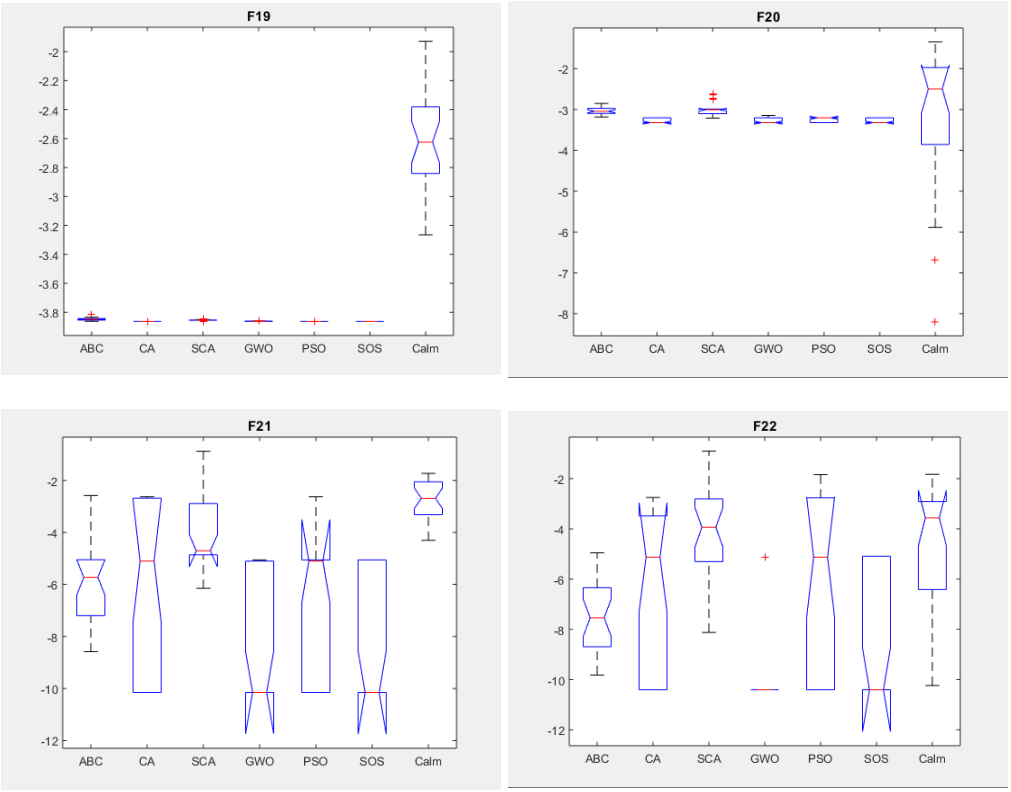


FIGURE 3.25 – Variation Anova pour 23 fonctions de test pour 7 méthodes appliquées.

3.3 La deuxième expérimentation

Dans la deuxième expérimentation, nous avons appliquer l'algorithme CALM pour effectuer une régression sur une série temporelle.

3.3.1 La première application

Nous avons procédé la simulation des séries temporelles des inscriptions de l'UA (l'université d'Alabama). Où les inscriptions historiques de l'UA sont utilisées comme données d'apprentissage et données de test, respectivement.

TABLE 3.3 – Inscriptions réelles, Inscriptions prévues par CALM et l'erreur

Années	Inscriptions réelles	Inscriptions prévues Par CALM	Erreur
1945	3.1830	6.6136	3.43
1946	5.7150	7.3086	1.59
1947	5.8850	10.0371	4.15
1948	6.2140	10.2203	4.00
1949	6.6610	10.5748	3.91
1950	6.6900	11.0565	4.36
1951	7.5800	11.0877	3.50
1952	7.6610	12.0468	4.38
1953	7.7350	12.1341	4.39
1954	7.8360	12.2138	4.37
1955	8.1170	12.3226	4.20
1956	8.2900	12.6254	4.33
1957	8.5730	12.8119	4.23
1958	8.6260	13.1168	4.49
1959	8.7290	13.1739	4.44
1960	9.0090	13.2849	4.27
1961	9.1050	13.5867	4.48
1962	9.3230	13.6901	4.36
1963	9.6730	13.9250	4.25
1964	10.4760	14.3022	3.82
1965	11.8170	10.4760	1.34
1966	12.9950	11.8170	1.17
1967	13.0920	12.9950	0.09
1968	13.2360	13.0920	0.14
1969	13.3590	13.2360	0.12
1970	13.3640	13.3590	0.00
1971	13.5640	13.3640	0.20
1972	14.3490	13.5640	0.78
1973	14.9380	14.3490	0.58
1974	15.6190	14.9380	0.68
1975	15.6380	15.6190	0.01
1976	15.6610	15.6380	0.02
1977	15.9930	15.6610	0.33
1978	16.0220	15.9930	0.02
1979	16.2470	16.0220	0.22

1980	16.5670	16.2470	0.32
1981	16.9160	16.5670	0.34
1982	16.9200	16.9160	0.00
1983	17.0480	16.9200	0.12
1984	17.2020	17.0480	0.15
1985	17.5050	17.2020	0.30
1986	17.7760	17.5050	0.27
1987	17.9180	17.7760	0.14
1988	18.0070	17.9180	0.08
1989	18.3790	18.0070	0.37
1990	18.4760	18.3790	0.09
1991	18.5470	18.4760	0.07
1992	18.7820	18.5470	0.23
1993	19.0430	18.7820	0.26
1994	19.1710	19.0430	0.12
1995	19.2670	19.1710	0.09
1996	19.3180	19.2670	0.05
1997	19.3660	19.3180	0.04
1998	19.4690	19.3660	0.10
1999	19.5160	19.4690	0.04
2000	19.6330	19.5160	0.11
2001	19.8270	19.6330	0.19
2002	19.8280	19.8270	0.00
2003	20.3330	19.8280	0.50
2004	20.9690	20.3330	0.63
2005	21.8350	20.9690	0.86
2006	23.8780	21.8350	2.04
2007	25,5800	23.8780	1.70
RMSE			2.11

Nous avons opter pour plus d'une seule exécution pour confirmer les résultats de notre méthode.D'après le tableau 3.3 nous remarquons que dans les premières années la convergence était faible ensuite notre méthode a converger idéalement vers les valeurs réelles.

Les figures 3.26 montrent bien la convergence de l'algorithme CALM vers les valeurs actuelles.

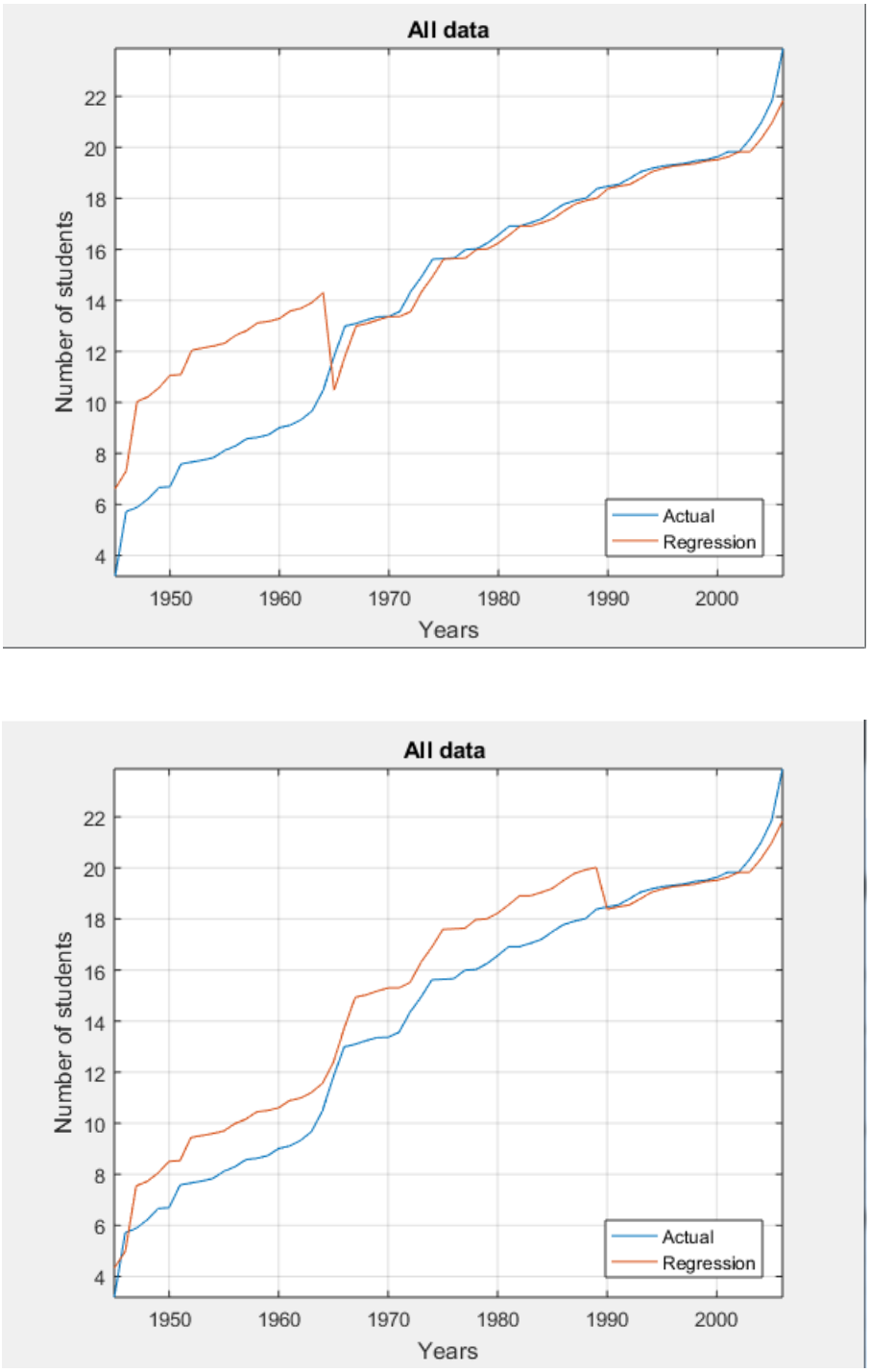


FIGURE 3.26 – Différences entre les inscriptions réelles et les inscriptions prévues pour l'UA sur la base de la méthode CALM.

Le tableau 3.4 montre une comparaison des résultats de la prédiction des inscriptions de l'UA obtenues par les deux algorithmes CALM et PSO. Nous remarquons que la méthode CALM de 1945 à 1966 a donnée des valeurs loin de les valeurs réelles, Par contre la méthode PSO a donnée une bonne prédiction, mais de 1967 à 2006 CALM a prédit tous les étudiants qui s'inscriraient dans ces années avec une très faible erreur, parce-que cette méthode n'est pas comme les autres métaheuristiques, elle trouve un ordre dans des désordres (les méthodes chaotiques ne sont pas comme les autres méthodes de métaheuristiques, ces méthodes effectuent un balayage sur tout l'espace de recherche en désordre afin de trouver l'ordre, comme le désordre causer dans le monde par CORONAVirus) .

TABLE 3.4 – Une comparaison des résultats de la prévision des inscriptions de l'UA pour les algorithmes CALM et PSO.

Années	Inscriptions réelles	Inscriptions prévues par CALM	Inscriptions prévues par PSO
1945	3.1830	6.6136	3.9899
1946	5.7150	7.3086	4.6456
1947	5.8850	10.0371	7.2195
1948	6.2140	10.2203	7.3924
1949	6.6610	10.5748	7.7268
1950	6.6900	11.0565	8.1812
1951	7.5800	11.0877	8.2107
1952	7.6610	12.0468	9.1154
1953	7.7350	12.1341	9.1978
1954	7.8360	12.2138	9.2730
1955	8.1170	12.3226	9.3757
1956	8.2900	12.6254	9.6613
1957	8.5730	12.8119	9.8372
1958	8.6260	13.1168	10.1249
1959	8.7290	13.1739	10.1788
1960	9.0090	13.2849	10.2835
1961	9.1050	13.5867	10.5681
1962	9.3230	13.6901	10.6657
1963	9.6730	13.9250	10.8873
1964	10.4760	14.3022	11.2431
1965	11.8170	10.4760	12.0594
1966	12.9950	11.8170	13.4227
1967	13.0920	12.9950	14.6202
1968	13.2360	13.0920	14.7188
1969	13.3590	13.2360	14.8652

1970	13.3640	13.3590	14.9902
1971	13.5640	13.3640	14.9953
1972	14.3490	13.5640	15.1986
1973	14.9380	14.3490	15.9966
1974	15.6190	14.9380	16.5954
1975	15.6380	15.6190	17.2877
1976	15.6610	15.6380	17.3070
1977	15.9930	15.6610	17.3304
1978	16.0220	15.9930	17.6679
1979	16.2470	16.0220	17.6973
1980	16.5670	16.2470	17.9261
1981	16.9160	16.5670	18.2514
1982	16.9200	16.9160	18.6062
1983	17.0480	16.9200	18.6102
1984	17.2020	17.0480	18.7403
1985	17.5050	17.2020	18.8969
1986	17.7760	17.5050	19.2049
1987	17.9180	17.7760	19.4804
1988	18.0070	17.9180	19.6248
1989	18.3790	18.0070	19.7152
1990	18.4760	18.3790	20.0934
1991	18.5470	18.4760	20.1920
1992	18.7820	18.5470	2.2642
1993	19.0430	18.7820	20.5031
1994	19.1710	19.0430	20.7684
1995	19.2670	19.1710	20.8985
1996	19.3180	19.2670	20.9961
1997	19.3660	19.3180	21.0480
1998	19.4690	19.3660	21.0967
1999	19.5160	19.4690	21.2015
2000	19.6330	19.5160	21.2492
2001	19.8270	19.6330	21.3682
2002	19.8280	19.8270	21.5654
2003	20.3330	19.8280	21.5664
2004	20.9690	20.3330	22.0798
2005	21.8350	20.9690	22.7263
2006	23.8780	21.8350	23.6067

2007	25,5800	23.8780	23.8780
RMSE		2.11	0.92

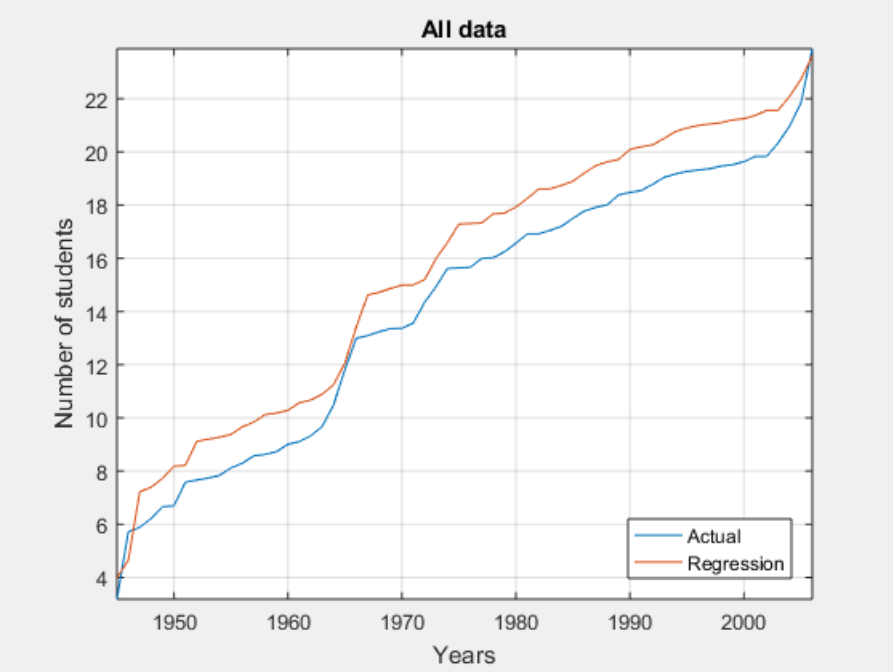


FIGURE 3.27 – Différences entre les inscriptions réelles et les inscriptions prévues pour l'UA sur la base de la méthode PSO.

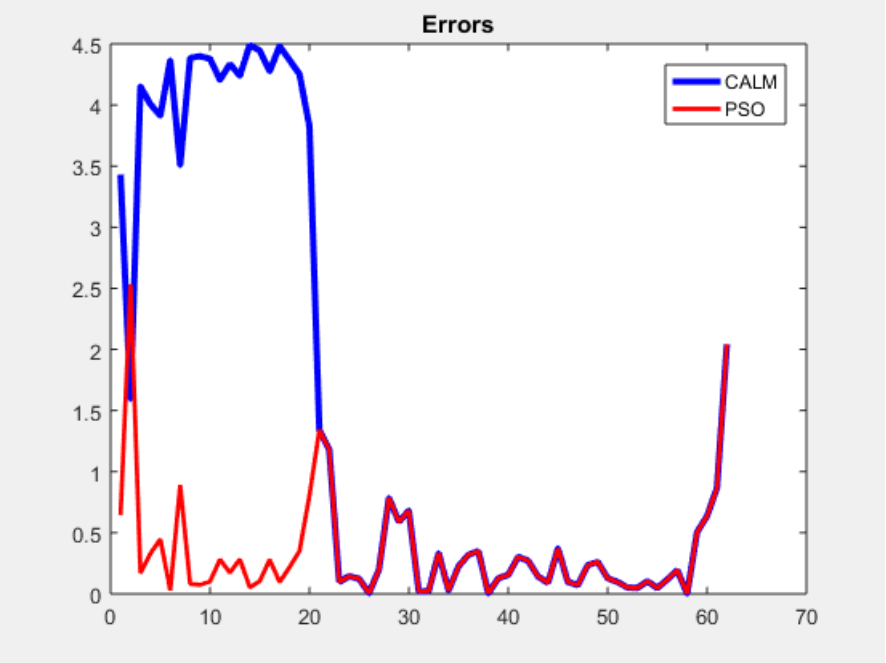


FIGURE 3.28 – Différence entre les erreurs de la prévision des inscriptions de l'UA obtenus par les algorithmes CALM et PSO.

3.3.2 La deuxième application

Dans cette application, nous avons appliquer l'algorithme CALM directement sur les séries temporelles au lieux de l'hybridé avec les réseaux de neurones .

Nous avons proposé la conversion d'un problème de prédiction des séries temporelles en un problème d'optimisation, pour ce fait nous avons formuler la fonction fitness suivante pour optimiser l'erreur entre les valeurs réelles et les valeurs prévues :

$$\sqrt{\frac{1}{L}\sum_{i=1}^{L-F_N}\left(A_{i+1}-F_{t-1}+\frac{1}{F_N}\sum_{i=1}^{F_N}\alpha\left(A_i-F_{t-1}\right)^2\right)} \tag{3.1}$$

La fonction précédente représente la combinaison des trois types de prédiction qui sont :

- La moyenne mobile simple.
- La moyenne mobile pondérée
- Le lissage exponentiel

La figure (3.29) montre l'organigramme du programme utilisé :

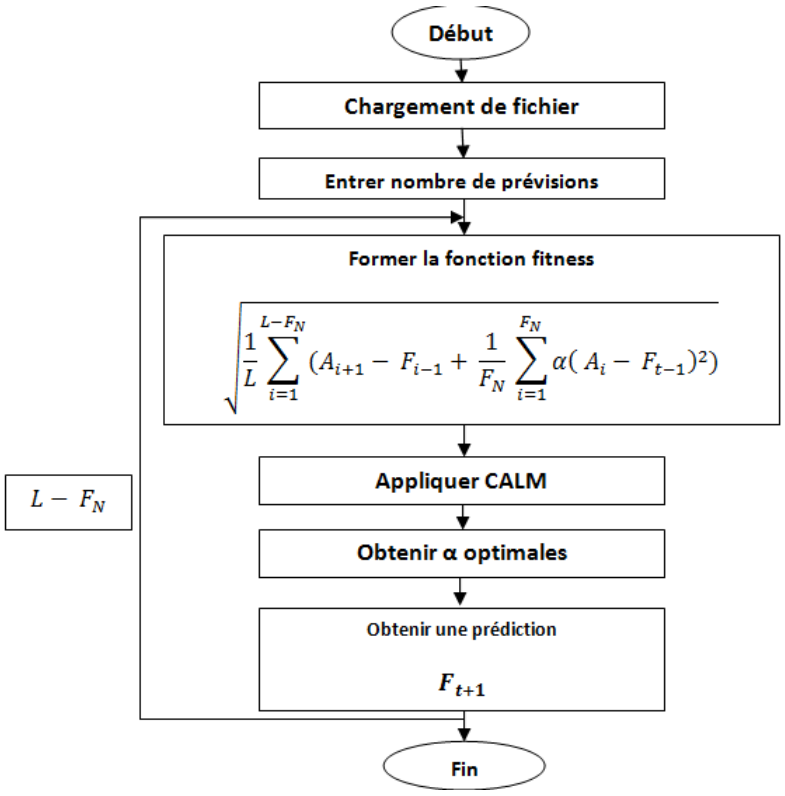


FIGURE 3.29 – l'organigramme du programme utilisée pour convertir la prédiction en optimisation

Pour cette application nous avons utiliser la base de données du volume global de glace.

Les figures suivantes montrent bien la convergence de la méthode utilisée vers les valeurs actuelles et la variation de l'erreur.

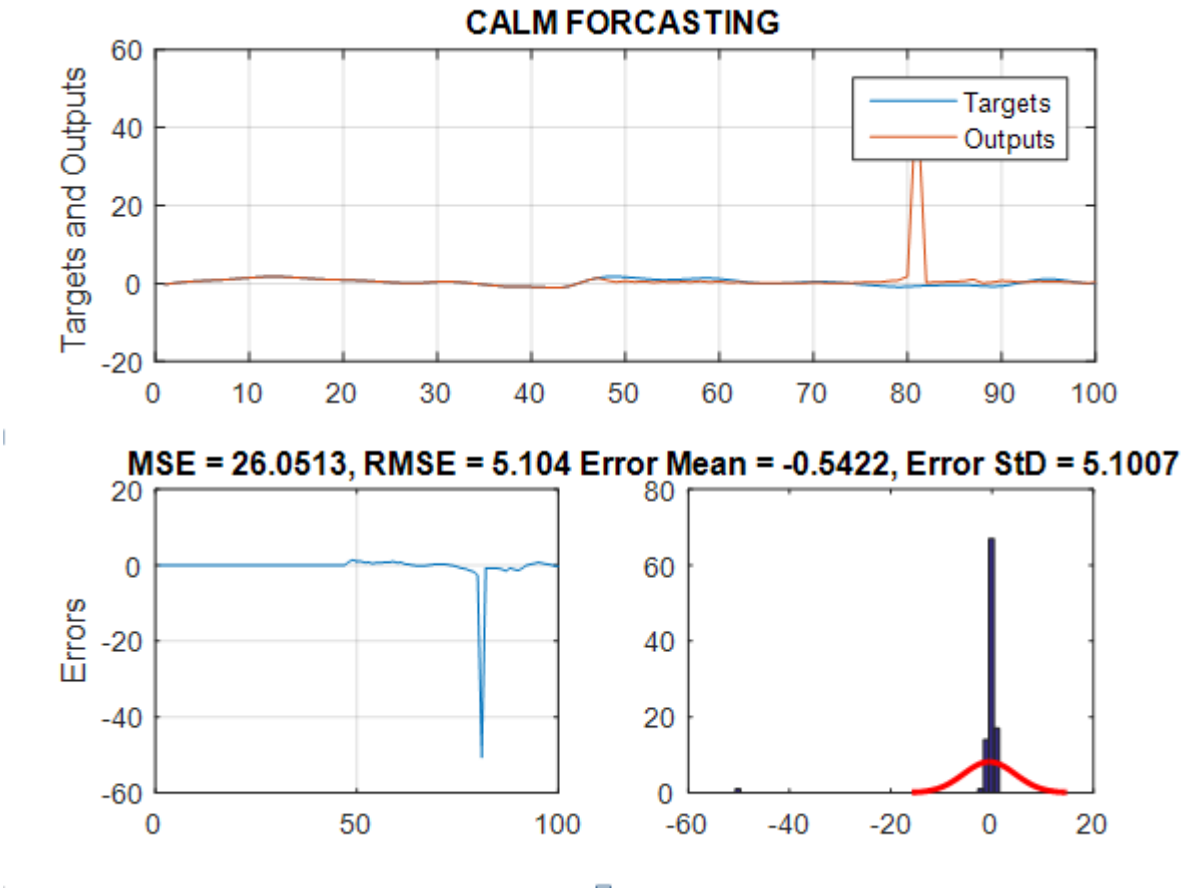


FIGURE 3.30 – la convergence de la méthode utilisée vers les valeurs actuelles et la variation de l'erreur.

D'après la figure (3.30), nous remarquons que pour 100 données notre méthode a donner une excellente prédiction, les valeurs prévues étaient similaires au valeurs réelles dont l'erreur était très faible, nous avons trouver un pic dans la valeur 81 (une divergence) inexplicable, puis il a directement repris sa convergence.

3.4 Conclusion

Dans ce chapitre nous avons commencé par une présentation de l'algorithme CALM à travers son aspect fonctionnel sous forme d'organigramme pour bien illustrer les détails de son implémentation.

Par la suite, nous avons explicités les différentes fonctions de fitness nécessaires pour l'évaluation des différentes techniques qui nous ont servis comme modèle de comparaison, il s'agit de l'algorithme PSO, l'algorithme ABC et aussi l'algorithme CA suivi du SCA et GWO et SOS.

A travers les expérimentations utilisées, on note que l'algorithme CALM est plus performant et donne de meilleurs résultats pour les problèmes de grande dimension.

Par contre, pour certaine fonctions de type multimodale à faible dimension , cet algorithme semble moins fort.

Pour mettre en valeur la robustesse de l'algorithme CALM, le domaine de valorisation et d'expérimentation de ce dernier est la prédiction des séries temporelles.

Dans la première application, d'après les résultats obtenues et en comparaison avec la méthode PSO, nous remarquons que CALM n'est pas comme les autres métaheuristiques, elle essaye de retrouver l'ordre dans le désordre.

Dans la deuxième application, d'après les courbes obtenues, nous conclurons que la méthode que nous avons proposer a bien marché et elle a donner de très bons résultats.

Conclusion Générale

Conclusion Générale

En conclusion de ce projet qui consistait à une modélisation d'une approche pour la résolution de problèmes d'optimisation combinatoire, nous dressons une synthèse comportant les différentes parties développées.

Dans un premier temps, notre étude s'est portée sur la rédaction d'un état de l'art sur les méthodes dans la littérature permettant de résoudre les problèmes d'optimisation difficiles à savoir les métaheuristiques.

Dans cette partie, un regard particulier a concerné un nouvel algorithme métaheuristique appelé Chaotic augmented lagrange multiplier (Multiplicateur de Lagrange augmenté chaotique) (CALM) .

Dans la seconde partie de ce mémoire, nous nous sommes intéressés à la description de notre champ d'application, il s'agit de la prédiction des séries temporelles.

Nous signalons aussi que la prédiction des séries temporelles est présente dans beaucoup de domaines technologiques comme le traitement de signal, la météorologie, la santé...

Pour la mise en valeur de nos choix théoriques basés sur l'exploitation de l'algorithme CALM pour la prédiction des séries chronologiques, nous avons consacrés la troisième partie de ce mémoire aux points suivants :

- Evaluation de CALM sur des fonctions de test ainsi que sa comparaison avec d'autres métaheuristiques. Le constat fait preuve de la capacité de CALM de générer des solutions avec une qualité significativement meilleure aux méthodes concurrentes pour les fonctions à haute dimension .
- Application de CALM dans le domaine de la prédiction des séries chronologiques en utilisant deux bases de données publiques de différentes complexités. Les résultats obtenus démontrent que CALM est unique et pas comme les autres méthodes, il effectue un balayage sur tout l'espace de recherche afin de trouver l'ordre.

- La conversion d'un problème de prédiction en un problème d'optimisation à base de l'algorithme CALM, Les résultats obtenues était bons mais la méthode a besoin d'amélioration.

Nous proposons également des perspectives à la fin de ce travail qui nous semble très prometteuses et qui sont comme suit :

- Utiliser l'algorithme CALM pour un autre domaine de recherche et faire un couplage avec une autre technique.
- Optimiser le comportement de CALM dans les fonctions de type multimodale à faible dimension .
- Améliorer le comportement de la méthode de prédiction basé sur CALM.

Bibliographie

[1] Hanaa Hachimi. *Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications*. PhD thesis, INSA de Rouen ; École Mohammadia d'ingénieurs (Rabat, Maroc), 2013.

[2] Javad Alikhani Koupaei and Marjan Firouznia. A chaos-based constrained optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing*, 12(11) :9953–9976, 2021.

[3] Salihu Aish Abdulkarim et al. *Time series forecasting using dynamic particle swarm optimizer trained neural networks*. PhD thesis, University of Pretoria, 2018.

[4] Mohammed Azmi Al-Betar, Ahamad Tajudin Khader, Mohammed A Awadallah, Mahmmoud Hafsaldin Alawan, and Belal Zaqaibeh. Cellular harmony search for optimization problems. *Journal of Applied Mathematics*, 2013, 2013.

[5] H Velásquez and Juan David. R-chaosoptimiser : an optimiser for unconstrained global nonlinear optimisation written in r language for statistical computing. *Ingeniería e Investigación*, 31(3) :50–55, 2011.

[6] Charles S Beightler, Don T Phillips, and Douglass J Wilde. *Foundations of optimization*. Prentice-Hall, 1979.

[7] Catherine Mancel. *modélisation et résolution de problèmes d'optimisation combinatoire issus d'applications spatiales*. PhD thesis, INSA de Toulouse, 2004.

[8] Nadia Smairi. *Optimisation par essaim particulaire : adaptation de tribes à l'optimisation multiobjectif*. PhD thesis, Paris Est, 2013.

[9] Ana Luísa Custódio and Luís Nunes Vicente. Using sampling and simplex derivatives in pattern search methods. *SIAM Journal on Optimization*, 18(2) :537–555, 2007.

[10] Omessaad HaJJI. Contribution au développement de méthodes d'optimisation stochastiques. application à la conception des dispositifs électrotechniques. *Mémoire de thèse de Doctorat, Université des sciences et technologies de Lille*, 2003.

[11] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5) :533–549, 1986.

[12] Abbas El Dor. *Perfectionnement des algorithmes d'optimisation par essaim particulaire : applications en segmentation d'images et en électronique*. PhD thesis, Université Paris-Est, 2012.

[13] ABDESSELAM Salim. *Conception d'un système d'optimisation pour le positionnement de caméras pour la motion capture MOCAP (Utilisation des métaheuristiques OEP et RFS)*. PhD thesis, Université Mohamed Khider-Biskra, 2018.

[14] John H Holland and Judith S Reitman. Cognitive systems based on adaptive algorithms. In *Pattern-directed inference systems*, pages 313–329. Elsevier, 1978.

[15] May Robert. Simple mathematical models with complicated dynamics. *Nature*, 261 :459–467, 1976.

[16] BING LI WEISUN JIANG. Optimizing complex functions by chaos search. *Cybernetics & Systems*, 29(4) :409–419, 1998.

[17] Maide Bucolo, Riccardo Caponetto, Luigi Fortuna, Mattia Frasca, and Alessandro Rizzo. Does chaos work better than noise ? *IEEE Circuits and Systems Magazine*, 2(3) :4–19, 2002.

[18] Raihane Mechgoug. *La Prédiction des Séries Temporelles utilisant les Paradigmes de Soft Computing*. PhD thesis, Université Mohamed Khider–Biskra, 2013.

[19] Chris Chatfield. *The analysis of time series : an introduction*. Chapman and hall/CRC, 2003.

[20] Kun Chang, Rong Chen, and Thomas B Fomby. Prediction-based adaptive compositional model for seasonal time series analysis. *Journal of Forecasting*, 36(7) :842–853, 2017.

[21] Georg Dorffner. Neural networks for time series processing. In *Neural network world*. Citeseer, 1996.

[22] Keith W Hipel and A Ian McLeod. *Time series modelling of water resources and environmental systems*. Elsevier, 1994.

[23] Girish K Jha, Parimala Thulasiraman, and Ruppa K Thulasiram. Pso based neural network for time series forecasting. In *2009 International Joint Conference on Neural Networks*, pages 1422–1427. IEEE, 2009.

[24] Paulo SG de M Neto, Gustavo G Petry, Rodrigues LJ Aranildo, and Tiago AE Ferreira. Combining artificial neural network and particle swarm system for time series forecasting. In *2009 International Joint Conference on Neural Networks*, pages 2230–2237. IEEE, 2009.

[25] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1) :21–27, 1967.

[26] Klaus Hechenbichler and Klaus Schliep. Weighted k-nearest-neighbor techniques and ordinal classification. 2004.

[27] Steen Magnussen and Erkki Tomppo. The k-nearest neighbor technique with local linear regression. *Scandinavian Journal of Forest Research*, 29(2) :120–131, 2014.

[28] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[29] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3) :273–297, 1995.

[30] Harris Drucker, Christopher J Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in neural information processing systems*, 9, 1996.

[31] K-R Müller, Alexander J Smola, Gunnar Rätsch, Bernhard Schölkopf, Jens Kohlmorgen, and Vladimir Vapnik. Predicting time series with support vector machines. In *International conference on artificial neural networks*, pages 999–1004. Springer, 1997.

[32] Vladimir Kurbalija. *Time series analysis and prediction using Case Based Reasoning Technology*. PhD thesis, University of Novi Sad (Serbia), 2009.

[33] Li-Juan Cao and Francis Eng Hock Tay. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on neural networks*, 14(6) :1506–1518, 2003.

[34] Christiane Lemke. *Combinations of time series forecasts : when and why are they beneficial ?*. PhD thesis, Bournemouth University, 2010.

[35] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5) :359–366, 1989.

[36] Ratnadip Adhikari and Rajiv Kumar Agrawal. Effectiveness of pso based neural network for seasonal time series forecasting. In *IICAI*, volume 3, pages 231–244, 2011.

[37] Guoqiang Zhang, B Eddy Patuwo, and Michael Y Hu. Forecasting with artificial neural networks : : The state of the art. *International journal of forecasting*, 14(1) :35–62, 1998.

[38] Stephen Grossberg. Recurrent neural networks. *Scholarpedia*, 8(2) :1888, 2013.

[39] Igor M Coelho, Vitor N Coelho, Eduardo J da S Luz, Luiz S Ochi, Frederico G Guimarães, and Eyder Rios. A gpu deep learning metaheuristic based model for time series forecasting. *Applied Energy*, 201 :412–418, 2017.

[40] Yamur K Al-Douri, Hussan Al-Chalabi, and Jan Lundberg. Time series forecasting using genetic algorithm. In *The Twelfth International Conference on Advanced Engineering Computing and Applications in Sciences*, 2018.