People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

MOHAMED KHIDER UNIVERSITY, BISKRA

FACULTY of EXACT SCIENCES

DEPARTMENT of MATHEMATICS

Thesis Submitted in Partial Execution of the Requirements of the Degree of

DOCTOR of SCIENCES

In the field of: *Applied Mathematics*

Option: *Analysis*

Submitted and Defended By

GHEDJEMIS Fatiha

Titled:

Introduction to Evolutionary Algorithms on Numerical Calculations

Board of Examiners:

Pr.	**Tidjani Menacer**	University of Biskra	Chairperson
Pr.	**KHELIL NACEUR**	University of Biskra	Supervisor
Pr.	**Tiaiba Abdelmoumen**	University of M'sila	Examiner
Pr.	**Saadi Khalil**	University of M'sila	Examiner

2025

Dedicace

To my family, friends and students.

Acknowledgements

I begin by expressing my deepest gratitude to Allah for the strength, patience, and unwavering support that made the completion of this research possible. Alhamdulillah.

This journey would not have been the same without the mentorship of my supervisor, Professor **Khelil Naceur**, and for his expertise, I am truly grateful. I also wish to thank my examination committee: the chairperson, Professor **Tidjani Menacer**, and the examiners, Professors **Tiaiba Abdelmoumen** and **Saadi Khalil**, for their time and constructive comments of this manuscript.

To the many people who stood by me, directly or indirectly, your encouragement meant the world. My appreciation also extends to the entire community of the Mathematics Department at the University of Biskra, whose collective efforts create an environment where students can thrive.

Abstract

This thesis proposes a new hybrid computational method that combines the accuracy of spectral methods with the optimization abilities of the Flower Pollination Algorithm to find solutions of differential equations, particularly boundary value problems. The approach uses Chebyshev polynomials for spectral approximation and combines FPA to minimize residual errors and optimize the coefficients, leading to accurate numerical solutions.

The study begins by exploring the structures of spectral methods and metaheuristic algorithms, concentrating on their mathematical properties and practical roles in optimization. It then introduces a new three-step hybrid methodology: extracting an initial approximation, calculating the residual error, and optimizing undetermined coefficients via FPA. The efficiency of this method is confirmed through several case studies, involving linear and nonlinear boundary value problems.

Experimental results validate that the proposed hybrid approach improves solution accuracy and computational efficiency contrast classical methods. The findings highlight the method's adaptability and potential in broader applications such as fluid dynamics, structural analysis, and data-driven modeling.

This work contributes a robust and flexible approach for solving complex differential problems, paving the way for future research in advanced numerical and optimization strategies.

Keywords: Differential equations ; Metaheuristic algorithms ; Chebyshev polynomials ; Flower Pollination Algorithm.

Resumé

Cette thèse présente une nouvelle méthode hybride de calcul numérique combinant la précision des techniques spectrales avec les capacités d'optimisation de l'algorithme de pollinisation des fleurs (FPA) pour la résolution des équations différentielles, en particulier les problèmes aux limites. L'approche s'appuie sur les polynômes de Chebyshev pour l'approximation spectrale et intègre le FPA afin de minimiser les erreurs résiduelles, per-

mettant ainsi d'obtenir des solutions approchées de haute précision.

Le travail débute par une étude des fondements des méthodes spectrales et des algorithmes métaheuristiques, en mettant l'accent sur leurs propriétés mathématiques et leur utilité en optimisation. Il propose ensuite une méthodologie hybride en trois étapes : dérivation d'une approximation initiale, calcul de l'erreur résiduelle, et optimisation des coefficients inconnus à l'aide du FPA. L'efficacité de cette approche est validée à travers plusieurs cas d'étude, incluant des problèmes aux limites linéaires et non linéaires.

Les résultats numériques confirment que la méthode hybride proposée améliore à la fois la précision des solutions et l'efficacité du calcul par rapport aux méthodes classiques. Ces résultats soulignent également l'adaptabilité de la méthode et son potentiel d'application dans des domaines variés tels que la dynamique des fluides, l'analyse des structures et la modélisation basée sur les données.

Ce travail apporte ainsi un cadre robuste et flexible pour la résolution de problèmes différentiels complexes, ouvrant la voie à de futures recherches en techniques numériques avancées et en optimisation.

Mots clés: Equations différentielles ; Algorithmes métaheuristique ; Polynomes de Chebyshev ; Algorithme de pollinisation des fleurs.

ملخص

العنوان : مقدمة حول الخوارزميات التطورية في التحليل العددي

تقدم هذه الأطروحة طريقة هجينة جديدة للحسابات العددية، تجمع بين دقة الطرق الطيفية وقدرات التحسين التي يوفرها خوارزمية تلقيح الزهور لحل المعادلات التفاضلية، خاصةً مشاكل القيم الحدية. تعتمد الطريقة على كثيرات الحدود تشيبيشيف للتقريب الطيفي، وتدمج خوارزمية تلقيح الأزهار لتقليل الأخطاء المتبقية، مما يؤدي إلى حلول تقريبية عالية الدقة.

يبدأ البحث بدراسة الأسس النظرية للطرق الطيفية والخوارزميات الميتاهيروسية، مع التركيز على خصائصها الرياضية وأدوارها التطبيقية في التحسين. ثم يتم عرض منهجية هجينة جديدة مكونة من ثلاث خطوات: اشتقاق تقريب أولي، حساب الخطأ المتبقي، وتحسين المعاملات المجهولة باستخدام خوارزمية تلقيح الزهور يتم التحقق من فعالية هذه الطريقة من خلال دراسات حالة تشمل مشاكل خطية وغير خطية.

تؤكد النتائج العددية أن الطريقة المقترحة تعزز دقة الحلول وكفاءتها الحسابية مقارنة بالطرق التقليدية. كما تظهر مرونة الطريقة وقابليتها للتطبيق في مجالات متنوعة مثل ديناميكا الموائع، تحليل الهياكل، والنمذجة المعتمدة على البيانات.

يساهم هذا العمل في تقديم إطار قوي ومرن لحل المعادلات التفاضلية المعقدة، ويفتح آفاقًا جديدة للبحث في مجال الطرق العددية وتقنيات التحسين المتقدمة.

الكلمات المفتاحية : المعادلات التفاضلية ,الخوارزميات الميتاهيروسية, كثيرات حدود تشيبيتشاف ,خوارزمية تلقيح الأزهار.

Achieved work

A significant outcome of this doctoral research is the publication of a scientific article entitled "Spectral Approximations Optimized by Flower Pollination Algorithm for Solving Differential Equations" in the International Journal of Computational Methods and Experimental Measurements, published by the International Information & Engineering Technology Association (IIETA), Canada, Vol. 13, No. 2, pp. 343-349. https://doi.org/10.18280/ijcmem.130211

This paper presents the core contribution of the thesis: a novel hybrid numerical approach that integrates spectral methods with the Flower Pollination Algorithm (FPA) to solve differential equations, involving boundary value problems with improved accuracy and computational efficiency. The publication reflects the originality and scientific relevance of the research, and demonstrates its applicability to a wide range of complex differential problems.

Abreviations and Notations

Different abreviations and notations on this thesis are:

ACE	Automatic Computing Engine
ALFPA	Adaptive-Lévy Flower Pollination Algorithm
APP	Antenna Positioning Problem
BA	Bat Algorithm
BFPA	Binary Fower Pollination Algorithm
BPFPA	Bee Pollinated Fower Pollination Algorithm
CD	Conjugate Direction
CEEMDAN	Complete Ensemble Empirical Mode Decomposition Adaptive Noise
CFPA	Chaos-based Flower Pollination Algorithm
CLSFPA	Flower Pollination Algorithm with Chaotic Local Search
CS	Cuckoo Search
DBP	Directed-Based Perturbations
DE	Differential Evolution
EFPA	Enhanced Flower Pollination Algorithm
EOFPA	Elite Opposition-based Fower Pollination Algorithm
FA	Firefly Algorithm
FPA	Flower Pollination Algorithm
FPP	Fractional Programming Problem
GGM	Gradient-Guided Moves
GSA	Gravitational Search Algorithm
HSA	Harmony Search Algorithm
IBPSO	Improved Binary Particle Swarm Optimization
IRW	Isotropic Random Walks
LTRW	Long-Tailed Scale-Free Random Walks

MFPA	Modified Flower Pollination Algorithm
MRLFPA	Modified Randomized-Location Flower Pollinatino Algorithm
NPL	National Physical Laboratory
OPF	Optimal Power Flow
PBIL	Population-Based Incremental Learning
PDE	Partial Differential Equation
PSO	Particle Swarm Optimization
RCGA	Real-Coded Genetic Algorithm
RP	Random Permutation
SA	simulated annealing
SI	Swarm Intelligence
SM	Spectral Method

Contents

List of Figures

List of Tables

Introduction

Several real life problems could be illustrated as differential equations after methematical modeling. Many of these problems aren't simple anough to have an analytical solution, therefore researchers tend to numerical domains to have approximate solutions. In the field of optimization methods, numerical methods ,evolutionary and metaheuristic algorithms play crucial role. This thesis is situated within the field of Evolutionary Algorithms (EAs), exploring their application to developing advanced numerical techniques for solving differential equations.

Spectral methods are considered as global numerical methods, this thesis starts with an investigation of spectral methods using Chebyshev polynomials [27], [38], [20], one of the orthogonal polynomials that have great properties and efficient approximation capabilities, which makes them a powerful tool to solve differential equations. Delving into examining the mathematical properties of spectral methods and some of their practical implementations.

The optimization component of our work is driven by Evolutionary Algorithms [1], [4], [48], [58]. Traditionally, the term 'Evolutionary Algorithm' refers to a specific class of metaheuristics directly inspired by Darwinian evolution, employing operators such as selection, crossover, and mutation. The Genetic Algorithm (GA) is the archetypal example.

However, the field of nature-inspired computation has produced a rich ecosystem of algorithms that exhibit core evolutionary processes, even if they don't use canonical genetic operators. These processes include the maintenance of a population of solutions, iterat-

ive improvement over generations, and a balance between exploration of the search space and exploitation of known good solutions. For the purposes of this thesis, we adopt this broader, more functional definition: an Evolutionary Algorithm is any population-based technique that evolves solutions towards an optimum through stochastic operators and selection.

Under this lens, many modern metaheuristics can be analyzed as specialized evolutionary systems. The Flower Pollination Algorithm (FPA) [56], which is central to this study, serves as a prime example. FPA's mechanics directly map to core concepts of evolutionary search discussed in this thesis. It evolves a population of solutions where:

- Global pollination serves as the primary diversification (exploration) mechanism. It implements a long-tailed, scale-free random walk, mathematically modeled by Lévy flights, to ensure the entire search space can be explored.
- Local pollination provides the intensification (exploitation) component, refining solutions in promising regions through localized random perturbations.
- A selection mechanism, based on solution fitness, ensures that the best traits discovered through this process survive and propagate into the next generation.

This evolutionary dynamic is what we harness. This thesis will therefore introduce the broad family of EAs, from classic to modern interpretations, including:

- Genetic Algorithm (GA), the foundational EA based on selection, crossover, and mutation [19].
- Particle Swarm Optimization (PSO), which evolves a 'swarm' of solutions through social learning [22].
- Flower Pollination Algorithm (FPA), which evolves solutions by mimicking the evolutionary reproductive strategy of flowering plants [56].

- Cuckoo Search (CS), where solutions evolve by mimicking the brood parasitic behavior of cuckoo [3].

The core processes of metaheuristic algorithms lies on trial and error just like childhood natural behavior.

These algorithms are extremely helpful in large complex search spaces, given near-optimal solutions within reasonable computational times. This thesis explores the definitions, properties, and applications of metaheuristic algorithms, focussing on Flower Pollination Algorithm.

The hybridization of algorithms is one of the famous techniques these days due to the powerfull of the results obtained from it, where several methods are combined to obtain the benefits of every approach on solving various problems.

The main contribution of this research provided by the innovative hybridization of an evolutionary algorithm - Flower Pollination Algorithm- with the spectral collocation method. By integrating the robust evolutionary search capabilities of FPA with the high precision of spectral methods, a noval approach is designed for solving differential equations especially boundary value problems. The proposed method is tested on different boundary value problems, proving its effectiveness and potential for wider applications.

Where the new approach is developed in the following manner: The first step is obtaining an approximate formula of the solution using the first step of spectral method. The second step based on calculating the residual using the root mean square formula. The final step is about implementing Flower Pollination Algorithm in minimizing the error and getting the unknown coefficients, now the approximate solution is found.

This thesis is organized as follows:

- **Chapter 1** provides simple definition of differential equations is given, and an overview of spectral methods is provided with specific focus on Chebyshev polynomials. It presents detailed examples and experiment results using MATLAB, to explain how collocation method works in his simple way.

- **Chapter 2** dedicates for metaheuristic approach and evolutionnary algorithms, where their definitions are provided and key properties are discussed. The chapter enfolds various algorithms with different properties, behaviors, and nature. Discussing their different classifications depending on several effects. In the end a brief history of metaheuristics and evolutionary algorithms is given.

- **Chapter 3** focusses on Flower Pollination Algorithm as a case study in evolutionary computation, the algorithm used for our innovative method, where it begins by making clear the process of flowering in nature and analyzes how the algorithm abstracts this into a set of evolutionary operators. The inspiration, structure, and operational mechanisms of the algorithm are explained, with mentioning its properties and utility in solving optimization problems. An exploration of hybridization's potentional with other methods, leads to the novel combination discussed in the last chapter.

- **Chapter 4** introduces the innovative hybrid method developed in this research. The intergration of the evolutionary search power of FPA with spectral methods, specifically with Chebyshev polynomials, is explained in detail. The chapter enfolds the theoretical framework, implementation details, and the resolution of three different boundary value problems, two linear problems and a non-linear problem, and an intrgo-differential equation formulated as an intial value problem. The performance of the hybrid approach is compared with the exact solution and a method introduced by Babaei in [32], demonstrating its effectiveness in enhancing solution accuracy and efficiency.

By combining advance numerical techniques with a powerful evolutionary algorithm, the objective of this research is to contribute a novel methodology that enhances the precision and efficiency of solving complex differential equations and boundary valu problems. The findings of this project have significant implications for various fields, giving a robust framework for further research and application in computational mathematics and

engineering.

Chapter 1

Spectral Methods

In numerical analysis field, the accurate and efficient solution for differential equations is very important. In the midst of various numerical methods, spectral methods have obtained significant attention, because of their high accuracy and efficiency. This chapter gives a brief introduction of differential equations, then discusses spectral methods, Chebyshev polynomials and Spectral-Collocation method, a powerful technique that sway the properties of Chebyshev polynomials for solving differential equations.

1.1 Differential Equations and Mathematical Formulation

A differential equation is an equation connecting an unknown function and one or more of its derivatives.

The work on differential equations has three principal goals:

1. Discovering the differential equation that expresses a specified physical situation.
2. Finding the appropriate axact or approximate solution of that equation.
3. Interpreting the results.

Let's see some real-life problems that have been translated to differential equations.

Problem N01

Newton's law of cooling can be interpreted in this way: the rate of change over time t of a body's temperature $T(t)$ is proportional to the difference between the two temperatures that of the body T and of the surrounding medium (A).

So,

$$\frac{dT}{dt} = -k(T - A), \tag{1.1}$$

where k is a positive constant.

Noting that in the case of $T > A$, $\frac{dT}{dt} < 0$ this means that the temperature decreases and the body is cooling. In the inverse case $(T < A)$, $\frac{dT}{dt} < 0$ and T increases.

Therefore, the physical law has described by a differential equation.

If k and A are given, the formula of $T(t)$ can be found, and the future temperature of the body can be predicted.

Problem N02

Torricelli's law says that the rate of change with respect to time t of the water's volume V in a draining tank is proportional to the square root of the depth y of water in the tank

$$\frac{dV}{dt} = -k\sqrt{y}, \tag{1.2}$$

k is a constant.

In the case of cylinder tank with vertical sides and cross-sectional area A, $V = Ay$ and $\frac{dV}{dt} = A(\frac{dy}{dt})$. Thus the equation takes the form:

$$\frac{dy}{dt} = -h\sqrt{y}, \tag{1.3}$$

$h = k/A$ is a constant.

Problem N03

In many simple cases, the rate of change over time t of a population $p(t)$ that have got birth and death rates constant is proportional to the population's size.

This is expressed by:

$$\frac{dP}{dt}kp, \tag{1.4}$$

where k is the proportionality constant. Observe that $p(t) = C\exp(kt)$ is a solution for the differential equation.

C is an arbitary constant, therefore the differential equation admit a particular solution could be chosen depending on additional information (initial or boundary conditions).

Mathematical Modeling

The process of mathematical modeling can be organized as:

1. Contructing of a mathematical model by formulate a real-world situation in mathematical terms.

2. Solving the resulting mathematical model.

3. Answering the question originally posed by interpreting the mathematical results in the context of the real-world problem.

1.2 Differential equations and types

The writting of a differential equation is not sufficient to guarantee that it has a solution.

Taking this equation: $(y')^2 + y^2 = -1$ doesn't have a real valued solution.

Here another example where the equation has only one solution: $(y')^2 + y^2 = 0$.

Consequently, the differential equation can has several solutions, one, or no one.

A differential equation's order is determined by the highest-order derivative present within it.

Let y be an unknown function of a single independent variable x. An $n-$order differential equation involving y and x is conventionally stated as:

$$F(x, y, y', y'', ..., y^{(n)}) = 0, \tag{1.5}$$

where F is a function of $n+2$ variable in $\mathbb{R}$.

- Differential equations can be subdivided into two types:

    1. Odinary differential equations, where the unknown function depends only on one independent variable in $\mathbb{R}$.

1. **Example 1.2.1** $a(x)f'' + b(x)f' + c(x)f = 0$ *(homogeneous linear differential equation of the second order).*

   $a(x)f'' + b(x)f' + c(x)f = g(x)$ *(non-homogeneous linear differential equation of the second order).*

- 1. Partial differential equations, it means that the unknown function depends on more then one independent variable in $\mathbb{R}$.

1. **Example 1.2.2** $\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$ *(Laplace equation, ellipitic)*

   $\frac{\partial^2 f}{\partial t^2} = c^2 \frac{\partial^2 f}{\partial x^2}$ *(Wave equation, hyperbolic)*

   $\frac{\partial f}{\partial t} = \alpha \frac{\partial^2 f}{\partial x^2}$ *(Heat equation, parabolic)*

- Differential equations equipped with conditions can be classified into:

    1. Initial value problem:

       Differential equation with initial condition.

       Example 1.2.3

$$\begin{cases} \frac{\partial f}{\partial t} = a, t \geq 0, \\ f(t) = 0, t = 0. \end{cases} \tag{1.6}$$

2. Boundary value problem:

1. The equation is associated with conditions at boundaries.

Example 1.2.4

$$\begin{cases} \frac{\partial f}{\partial x} = a, x \in [a, b] \\ f(x) = 0, x = a, \\ f(x) = \beta, x = b. \end{cases} \tag{1.7}$$

1.3 Chebyshev Polynomials

"Chebyshev polynomials are everywhere dense in numerical analysis"

Philip David, George Forsythe.

Chebyshev polynomials are named after the Russian mathematician P.L. Chebyshev (1821−1894), who originally examined them. The assembled studies of this noted savant are provided in Ruassian and French in [38].

Definition 1.3.1 *A polynomial is a function that is possibly written as the following shape:*

$$p(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n. \tag{1.8}$$

Where a_j are real numbers and x is a real variable. Supposing that $a_n \neq 0$, then p is of degree n.

Polynomials have several favorable properties, that makes them notably suitable to approximate more complex fcuntions. Some of the important properties are the differentiation and integration, where they are able to be differentiated without restrictions as frequently as needed for any value of x.

1.3.1 First-Kind Chebyshev Polynomials

Definition 1.3.2 *The first-kind Chebyshev polynomial $T_n(x)$ is a polynomial in x of degree n, introduced by the relation*

$$T_n(x) = \cos(n\theta), x = \cos(\theta). \tag{1.9}$$

From the definition of the first-kind Chebyshev polynomials it's obvious that $x \in [-1, 1]$, and θ can be taken in $[0, \pi]$.

Thus,

$T_0(x) = \cos(0x) = 1,$

$T_1(x) = \cos(1\theta) = x,$

$T_2(x) = \cos(2\theta) = 2\cos^2\theta - 1 = 2x^2 - 1,$

$T_3(x) = \cos(3\theta) = 4\cos^3\theta - 3\cos\theta = 4x^3 - 3x,$

$T_4(x) = \cos(4\theta) = 8\cos^4\theta - 8\cos^2\theta + 1 = 8x^4 - 8x^2 + 1.$

The reccurence relation to get Chebyshev polynomials can be deduced using the trigonometric identity:

$$\cos(n\theta) + \cos(n-2)\theta = 2\cos\theta\cos(n-1)\theta \tag{1.10}$$

Thus,

$$\begin{aligned} T_n(x) &= 2xT_{n-1}(x) - T_{n-2}(x), n = 2, 3, ... \\ T_0(x) &= 1, \\ T_1(x) &= x. \end{aligned}$$

Using the reccurence relation, $T_n(x)$ can be calculated easily than via the definition of Chebyshev polynomials.

Example 1.3.1 *Calculate $T_5(x)$ from $T_3(x)$ and $T_4(x)$.*

$$\begin{aligned} T_5(x) &= 2xT_4(x) - T_3(x) \\ &= 2x(8x^4 - 8x^2 + 1) - (4x^3 - 3x) \\ &= 16x^5 - 20x^3 + 5x. \end{aligned}$$

The set of Chebyshev polynomials of the first kind exhibits orthogonality when integrated with the weight function $\omega_k = (1 - x^2)^{-1/2}$.

The graphs (1.1)present a generation of the Chebyshev polynomials the first kind via MATLAB.

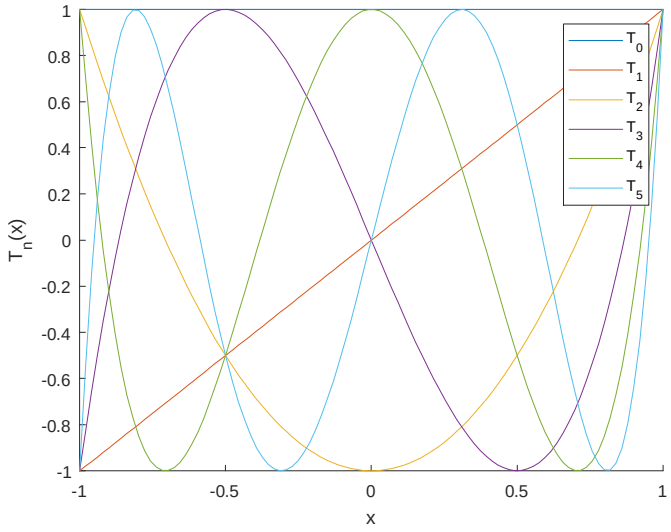

Figure 1.1: First-kind Chebyshev polynomials

1.3.2 Second-Kind Chebyshev Polynomials

Definition 1.3.3 *The first-kind Chebyshev polynomial $T_n(x)$ is a polynomial in x of degree n, introduced by the relation*

$$U_n(x) = \sin(n+1)\theta / \sin\theta,\ x = \cos\theta.$$

it's obvious that $x \in [-1, 1]$, and θ can be taken in $[0, \pi]$.

Using this formulaes, Chebyshev polynomials could be easy to be deduced,

$\sin 1\theta = \theta$,

$\sin 2\theta = 2\sin\,\theta\cos\theta$,

$\sin 3\theta = \sin\theta(4\cos^2\theta - 1)$,

$\sin 4\theta = \sin\theta(8\cos^3\theta - 4\cos\theta)$.

Thus,

$U_0(x) = 1$,

$U_1(x) = 2x$,

$U_2(x) = 4x^2 - 1$,

$U_3(x) = 8x^3 - 4x, ...$

From this trigonometric formulae,

$$\sin(n+1)\theta + \sin(n-1)\theta = 2\cos\theta\sin n\theta, \tag{1.11}$$

the reccurence relation is

$$\begin{aligned} U_n(x) &= 2xU_{n-1}(x) - U_{n-2}(x), n = 2, 3... \\ U_0(x) &= 1, \\ U_1(x) &= 2x. \end{aligned} \tag{1.12}$$

The trigonometric formulae,

$$\sin(n+1)\theta - \sin(n-1)\theta = 2\sin\theta\cos n\theta, \tag{1.13}$$

yields to the relationship between the first-kind $T_n(x)$ and the second-kind Chebyshev polynomials $U_n(x)$,

$$U_n(x) - U_{n-2}(x) = 2T_n(x), n = 2, 3 \tag{1.14}$$

1.3.3 Chebyshev Polynomials in [a,b]

Since Chebyshev polynomials are in the range $[-1, 1]$, a mapping should done to get a Chebyshev polynomials in a general range $[a, b]$.

Using the linear transformation

$$s = \frac{2x - (a + b)}{b - a}. \tag{1.15}$$

Where the Chebyshev polynomials in $[a, b]$ are $T_n(s)$.

Example 1.3.2 *Deducing the four first Chebyshev polynomials of the first kind in* $[1, 4]$:

$T_0(s) = T_0(\frac{2x-5}{3}) = 1,$

$T_1(s) = T_1(\frac{2x-5}{3}) = \frac{1}{3}(2x - 5),$

$T_2(s) = T_2(\frac{2x-5}{3}) = 2(\frac{2x-5}{3})^2 - 1 = \frac{1}{9}(8x^2 - 40x + 41),$

$T_3(s) = T_3(\frac{2x-5}{3}) = 4(\frac{2x-5}{3})^3 - 3(\frac{2x-5}{3}) = \frac{1}{27}(32x^3 - 240x^2 + 546x - 365),$

$T_4(s) = T_4(\frac{2x-5}{3}) = (1/81)(128x^4 - 1280x^3 + 4512x^2 - 6560x + 3281),$

$T_5(s) = T_5(\frac{2x-5}{3}) = (1/243)(512x^5 - 6400x^4 + 30560x^3 - 69200x^2 + 73810x - 29525),$

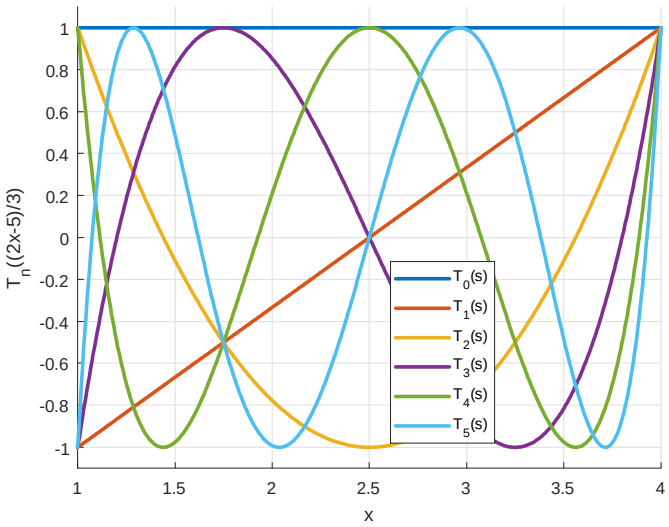

Figure 1.2: First-kind Chebyshev Polynomials in [1,4]

1.3.4 Shifted Chebyshev Polynomials

Real life problem are predomantly in $[0, 1]$, therefore the shifted Chebyshev polynomials are the Chebyshev polynomials mapped in this interval.

The shifted Chebyshev polynomials are extrated using $s = 2x - 1$,

Thus

$$T_n^*(x) = T_n(2x - 1), \tag{1.16}$$

where

$T_0^*(x) = 1,$

$T_1^*(x) = 2x - 1,$

$T_2^*(x) = 8x^2 - 8x + 1,$

$T_3^*(x) = 32x^3 - 48x^2 + 18x - 1,$

$T_4^*(x) = 128x^4 - 256x^3 + 160x^2 - 32x + 1,$

$T_5^*(x) = 512x^5 - 1280x^4 + 1120x^3 - 400x^2 + 50x - 1.$

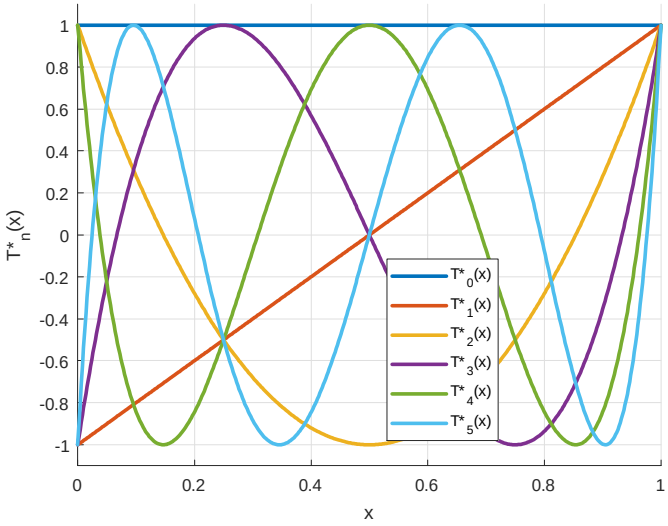

Figure 1.3: Shifted Chebyshev polynomials

1.4 Numerical Methods

In the world of differential equations, numerical methods can be classified into global or local methods.

1.4.1 Local Methods

These methods focus on segments of the problem's domain in the approximation of PDE's solutions. Where they begin by dividing the domain into smaller, dicrete elements or points and solve the differential problem within each element. Such as finite-difference and finite-element methods.

Local methods can be considered as a well-suited method for complex geometries and problems with regular boundaries, also they are flexible in handling no-uniform meshes and localized refinement. But they can caused an increased computational cost as they may require a large number of elements or points for high accuracy. In addition, the approximation made locally can accumulate errors.

1.4.2 Global Methods

Global methods utilizes basis functions that reach the entire domain to approximate the solutions of PDEs. They consider the entire domain rather than segmenting it into smaller parts and solve the problem in overall the domain.

Spectral methods are global methods.

For problems with smoothness , these methods can reach superior accuracy due to the global nature of basis functions. But in the case of complex geometries and problems with irregular boundaries, they can be less flexible. Also, the fact that they require global information caused intensive computational for large-scale problems.

Weighted Residual Methods

Spectral methods belong to the family of weighted residual methods, for that reason here a brief introduction to this family of methods. [28]

Suppose the general problem

$$\partial_t u(x,t) - \mathcal{L}u(x,t) = N(u)(x,t), t > 0, x \in \Omega, \tag{1.17}$$

Here $\mathcal{L}$ is a spatial derivative operator, and N is a linear or nonlinear spatial operator with lower-odrer, and Ω is a bounded domain on $\mathbb{R}^d, d = 1, 2$ or 3. Adding to the equation 1.17 initial or boundary conditions.

Considering only the weighted residual method for spatial discretization, and supposing that the time derivative is dicretized with a suitable time stepping scheme.

Considering the Crank-Nicolson leap-frog scheme for (1.17):

$$\frac{u^{n+1} - u^{n-1}}{2\tau} - \mathcal{L}(\frac{u^{n+1} + u^{n-1}}{2\tau}) = N(u^n), n \geqslant 1. \tag{1.18}$$

Where τ is the time step size, and $u^k(.)$ is an approximation of $u(., k\tau)$.

The equation (1.18) is equivalent to

$$Lu(x) := \alpha u(x) - \mathcal{L}u(x) = f(x), x \in \Omega, \tag{1.19}$$

here $u = \frac{u^{n+1}+u^{n-1}}{2}$, $\alpha = \tau^{-1}$ and $f = \alpha u^{n-1} + N(u^n)$. So a steady-state problem of the form (1.19) is needed to be solved, at each time step.

The first step of weighted residual methods is to write the approximate solution of (1.19) as a finite sum

$$u(x) \approx u_N(x) = \sum_{k=0}^{N} a_k \Phi_k(x), \tag{1.20}$$

Where $\{\Phi_k\}$ are basis functions, such as Chebyshev polynomials, Fourier series and Hermite

polynomials.

The next step is to determinate the expansion coefficients $\{a_k\}$.

The residual obtained after substituting u_N for u in (1.19) is

$$R_N(x) = Lu_N(x) - f(x) \neq 0, x \in \Omega. \tag{1.21}$$

Forcing the residual to zero by requiring

$$(R_N, \Psi_j)_\omega := \int_\Omega R_N(x)\Psi_j(x)\omega(x)dx = 0, 0 \leq j \leq N, \tag{1.22}$$

ω is a positive weight function, $\{\Psi_j\}$ are the test functions, where the most commonly utilized test functions are trigonometric functions or orthogonal polynomials, such as Chebyshev, Legendre, Laguerre and Hermite polynomials.

Or,

$$(R_N, \Psi_j)_\omega := \sum_{k=0}^{N} R_N(x_k)\Psi_j(x_k)\omega_k = 0, 0 \leq j \leq N, \tag{1.23}$$

where $\{x_k\}_{k=0}^N$ are a set of preselected collocation points, and $\{\omega_k\}_{k=0}^N$ are the weights of a numerical quadrature formula.

The choice of the test functions distinguishes spectral methods, for example Galerkin, collocation, Tau methods.

1.4.3 Collocation Method Using Chebyshev Polynomials

To understand the method let's solve some problems.

Example 1.4.1 *supposing the linear problem*

$$\begin{cases} -\frac{d^2u}{dx^2} = \exp(x); x \in [-1, 1], \\ u(-1) = u(1) = 0. \end{cases} \tag{1.24}$$

Solution of The example:

Let's solve the example with spectral-collocation method using Chebyshev polynomials of the first kind defined above.

The solution can be broken down into these main points:

1. Generate the grid of Chebyshev nodes with the cosine function.
2. For the function $f(x) = \exp(x)$, find its value at each node.
3. Build the associated second-order differentiation matrix D.
4. Construct the matrix A, using the collocation points to get a system $Au = b$.
5. Solve the linear system $Au = b$, and find the unkown coefficients of u, with enforcing boundary conditions.
6. Plot and compare the numerical solution with the exact one.

Here is the graph of the solution to the problem using Chebyshev collocation method;

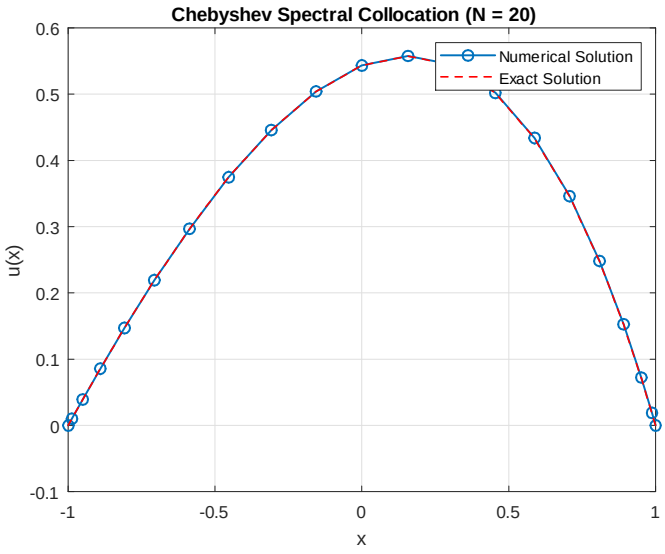

Figure 1.4: Solution using Spectral-Collocation Method with Chebyshev Polynomials for the Linear Example

Example 1.4.2 *Let's solve the non-linear boundary value problem defined by*

$$\begin{cases} \frac{d^2u}{dx^2} = \exp(u), x \in [-1, 1], \\ \quad u(-1) = u(1) = 0 \end{cases} \tag{1.25}$$

The steps used in the resolution of this problem using spectral-collocation method are:

1. Definition of nodes number, then generation of differentiation matrix.
2. Construction of the second differentiation matrix and applying boundary conditions.
3. Use *fsolve* to solve the system of nonlinear equations.
4. Reapplying boundary conditions.
5. Due to that the problem doesn't have simple exact solution, plotting the computed values at the collocation points, showing the smooth polynomial that passes through them, where it is the approximate solution.

The graph of the approximate solution is,

1.5 Conclusion

This chapter commences with a definition of differential equations then a proceeding of spectral methods, giving particular attention to Chebyshev polynomials. Furthermore, the fundamental mechanics of the spectral collocation technique are elucidated through practical examples and experimental results implemented in MATLAB.

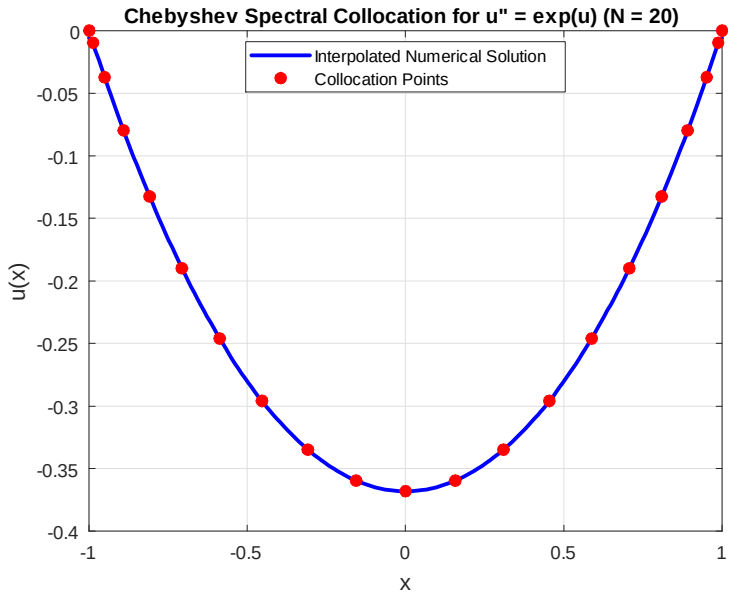

Figure 1.5: Solution using Spectral-Collocation Method with Chebyshev Polynomials for the Non-Linear Example

Chapter 2

Evolutionary Algorithms: An Introduction to Metaheuristic Optimization

This chapter introduces the field of Evolutionary Algorithms by presenting a broad, functional perspective. We will show how the core principles of evolution population based search, diversification, and intensification provide a powerful framework for understanding a wide range of modern metaheuristics.

2.1 Optimization

The majority of optimization challenges can be effectively written in the following generic form:

$$\min_{x \in \mathbb{R}} f_i(x), (i = 1, 2, ..., M), \tag{2.1}$$

subject to

$$h_j(x) = 0, (j = 1, 2, ..., J), g_k(x) \leqslant 0, (k = 1, 2, ..., K),$$

with $x = (x_1, x_2, ..., x_n)^T$.

Where

- Decision or design variables are components of x_i.
- f_i are the cost or objective functions .
- The search(design) space encompasses all possible decision variables.
- Solution space is the set of the objective function values.
- The equalities h_j and inequalities g_k are the constraints.

Quote here different classifications of optimization problems according to

1. Objective numbers:
    - For $M = 1$, it is called single-objective optimization.
    - For $M > 1$, it can be named multi-objective, multi-criteria or even multi-attribute optimization.
2. Constraint's number:
    - In the case where $J = K = 0$, the problem called an unconstrained optimization one.
    - If $K = 0$ and $J \geq 1$, then it will be equality-constrained problem .
    - If $J = 0$ and $K \geq 1$, so it named Inequality-constrained problem.
3. Linearity
    - In the linearly constrained problem, h_j and g_k are all linear.

- In linear programming problem, h_j, g_k, and all objective functions are linear.
- Or f_i, h_j, and g_k are non linear, it is a non linear optimization problem.

2.2 Search for Optimality

Once an optimization problem has been properly formulated, the subsequent step involves applying suitable mathematical methods and systematic approaches to derive optimal solutions. Searching for the optimal solution can be likened to a treasure hunt.

Suppose we are on a time-limited quest to discover a hidden treasure in a hilly landscape. There are two extremes. The first one is to be blind without any instructions, leading to arbitrary and inefficient search operations. The other extreme is to be informed about the treasure's location, situated on the highest summit of a known region. In this case, the initial action involves climbing to the steepest cliff and attempting to reach the highest peak. This framework aligns with the classic hill-climbing approach.

Often, our search is between these two extremes, indicating that we are clear-sighted, yet simultaneously unaware of where to look. It's not feasible to examine every single square inch of an extensive hilly region just to find the treasure. Thus, we resort to random walks. The preferred strategy involves employing random walks while searching for clues. We start from a random place, move to another, and then proceed to yet another, and so forth. In fact, contemporary search algorithms prominently feature these random walks as a fundamental characteristic.

Moreover, the efficiency of this approach lies in its adaptability to diverse landscapes. By initiating the search from random locations and navigating through various paths, we increase the likelihood of discovering hidden information or potential optimal solutions. This dynamic and exploratory method aligns with the evolving nature of search algorithms in modern problem-solving scenarios.

Engaging in treasure hunting can occur individually, where the entire path is perceived as

a trajectory-based search, paralleling techniques such as Simulated Annealing. Alternatively, it can involve a collaborative effort with a group of individuals sharing information. The latter approach embodies swarm intelligence, a concept applied in Particle Swarm Optimization.

The search process can take a significant amount of time when the area is extensive, especially if the treasure holds considerable importance. However, with an unrestricted timeframe and accessible locations, we have the opportunity to attain the ultimate treasure (the global optimal solution).

The search strategy can be enhanced by retaining the more effective hunters and introducing new ones, as seen in Genetic Algorithms and Evolutionary Algorithms.

It's noteworthy that almost all metaheuristic algorithms share the following strategies:

- Employ best solutions or agents.
- Randomize (or replace) the less optimal solutions by evaluating the competence (fitness) of each individual in collaboration with the system history (utilizing memory).

Better and more efficient optimization algorithms are expected to be achieved by attaining this balance.

2.3 Understanding Evolutionary and Metaheuristic Approaches

While classic Evolutionary Algorithms (EAs) are defined by genetic operators like crossover and mutation, we can adopt a more functional definition. Under this lens, any population-based algorithm can be considered 'evolutionary' if it iteratively improves solutions by balancing diversification (exploration) and intensification (exploitation). This perspective allows us to analyze algorithms like Particle Swarm Optimization and Flower Pollination

Algorithm as evolutionary systems that use different, nature-inspired operators to achieve the same goal.

Metaheuristic algorithms are applied to a vast range of optimization problems, where these problems are characterized as **I know it when I see it** problems. It means that we have limited heuristic information about the solution to proceed, given a candidate solution, testing it, and then assessing its effectiveness.

Using the hill-climbing strategy involves initiating a random set of solutions and introducing a small random modification to each. Subsequently, the modified solution is tested, and if it proves to be better than the original, it replaces the latter; otherwise, the original solution remains unchanged. This iterative process is repeated to explore the entire local search space and optimize the solution within the defined constraints.

To explore the entire space comprehensively, it is necessary to select solutions that are distant and distinct from each other at times. Subsequently, the process of discovering new local spaces is repeated, akin to the initial exploration. Finally, from the diverse set of solutions obtained, the optimal solution is chosen based on the specified criteria.

2.4 Classification of Metaheuristic Algorithms Based on Their Nature

To understand the mechanics of the evolutionary systems discussed in this thesis, it is essential to classify them based on their use of randomness. Evolutionary processes are inherently stochastic, and this is reflected in the design of most EAs. Optimization algorithms can be classified by their nature into:

2.4.1 Deterministic

These algorithms are designed to follow the same path, every time the program run, for the same strating point. For this kind of algorithm, the values of the design variables,

path and functions are repeatable, such as the hill-climbing algorithm.

Deterministic algorithms can be separated into gradient-based algorithms and gradient-free algorithms.

Gradient-based algorithms use both the function values and their derivatives. These types of algorithms are efficient for smooth unimodal problems but are not suitable for problems with discontinuities in the objective functions. An example of such an algorithm is the Newton-Raphson Algorithm.

Gradient-free algorithms use only the function values without resorting to derivatives, exemplified by algorithms like Hooke-Jeeves pattern search and Nelder-Mead downhill simplex. These algorithms can be used to solve problems that gradient-based algorithms couldn't address.

2.4.2 Stochastic

In this context, randomness plays a significant role, where in each run of the program, this remdomness causes variations in the solution within the population. While the differences between solutions may not be substantial, the paths of each individual are not consistent every time, like in Genetic Algorithms.

2.4.3 Hybrid of Stochastic and Deterministic Algorithms

It employs deterministic algorithms but starts with different initial points, introducing an element of randomness. A notable example is hill-climbing with a random restart; this form of randomness prevents the algorithm from getting stuck in a local peak.

In optimization literature, even hybrid algorithms are classified as stochastic algorithms. We have two types of stochastic algorithms: heuristic and metaheuristic. The difference between them is small, and there are those who use '**heuristic**' and '**metaheuristic**' interchangeably.The term 'heuristic' denotes 'to find' or 'to discover by trial and error.' Algorithms of this type can generate high-quality solutions to challenging optimization

problems within a reasonable time-frame, though they do not assure the achievement of optimal solutions. So, these algorithms provide good solutions but not necessarily the best solutions all the time. If we want to simplify the idea of heuristics, we can say that **heuristic is the process of generating acceptable solutions to a complex problem within a reasonable practical time, using trial and error**. The aim of these algorithms is to find a good feasible solution to a complex problem within an acceptable timescale.

On the flip side, '**meta**' implies '**high-level**,' distinguishing metaheuristic algorithms as superior to heuristics. Metaheuristics outperform heuristics by employing a combination of local search and randomization. This dual approach ensures an intensified exploration of local spaces, while simultaneously leveraging randomization to transition seamlessly from local to global search. Consequently, these algorithms prove particularly adept at tackling global optimization challenges.

So, metaheuristic algorithms work using a combination of:

- Intensification or exploitation, it means focusing on a local region and exploiting the information found by the current good solution.
- Diversification, it refers to the phase where the algorithms explore the search on the global scale by generating diverse solutions. This characteristic, made via randomization, avoids entrapment in local optima and enhances exploration diversity.
- The process of selecting best solution, drives the system toward convergence to optimality.

2.5 Classification of Metaheuristic Algorithms Based on Their Working System

Beyond their nature, Evolutionary Algorithms can be categorized by their implementation style, which often distinguishes classic from modern approaches. Procedure-based algorithms align with classic EAs like the Genetic Algorithm, while equation-based systems represent many of the contemporary EAs that this thesis explores.

2.5.1 Procedure-Based Algorithms

The Genetic Algorithm functions as procedure-based algorithm, characterized by an iterative approach in its main steps. In several algorithms, The recurring procedure follows a general structure with three distinct parts:

1. Solution's representations:

   In most cases, a solution vector x in a D-dimensional problem is encoded as either a fixed-length binary sequence or an array of real-valued numbers.

2. Solution's modifications:

   Changes can be achieved via mutation or crossover operations.Mutation involves modifying individual solutions at one or several locations,whereas crossover combines elements from two parent solutions by exchanging or blending their components to generate new solutions.

3. Solution's selection:

   The evaluation of the fitness's solution is achieved, according to its objective value. A solution is selected from a population based on its fitness (lowest values for minimization). The best solutions are then transmitted to the next generation.

2.5.2 Equation-Based Algorithms

Most recent nature-inspired algorithms are equation-based, where solutions are represented as vectors x_i, $(i = 1, 2, ...n)$, representing a population set of n solutions in the D-dimensional search space. For the selection of solutions, these algorithms use fitness values, wherein, for minimization problems (maximization problems), lower objective values (higher objective values) pass on to the next generation."

The distinction among these nature-inspired algorithms lies in the step of modifying solutions, where they employ different mathematical forms or search mechanisms. Actually, in iteration or generation t the solution (position) vector is denoted by x_i^t, and after using modification increment or mutation vector Δx_i^t, a new solution x_i^{t+1} is generated

$$x_i^{t+1} = x_i^t + \Delta x_i^t. \tag{2.2}$$

Usually, Δx_i^t represents a step size or a step vector.

On the gradient-based algorithms such as the Newton-Raphson method, the step is related to the negative gradient:

$$\Delta x_i^t = -\eta \nabla f(x), \tag{2.3}$$

where $f(x)$ is is the objective function, ∇f is the gradient of f and η is a positive learning parameter.

In certain other nature-inspired algorithms, the modification is associated with the increment of velocity, expressed as:

$$\Delta v_i^t = v_i^{t+1} - v_i^t, \tag{2.4}$$

The velocity of the solution (particle, agent, etc.) i at iteration t and iteration $t + 1$ respectively, is presented by v_i^t and v_i^{t+1}. Δt is the time increment and it is the difference in the iteration counter $\Delta t = t + 1 - t = 1$, so, these algorithms don't need a unit to be

used.

In this part, various equations utilized for modifying the solutions are presented:

Differential Evolution (DE)

In Differential Evolution, the main mutation is given by the formula:

$$\begin{aligned} x_i^{t+1} &= x_i^t + \Delta x_i^t, \\ \Delta x_i^t &= F(x_j^t, x_k^t). \end{aligned} \tag{2.5}$$

x_i^t, x_j^t and x_k^t are different solution vectors chosen from the population. The mutation strength is controlled by the parameter $F \in]0, 1[$.

Particle Swarm Optimization (PSO)

The swarming behavior of birds and fish serves as inspiration for creating Particle Swarm Optimization (PSO). Using equations below, the upgrade of position and velocity of particle i at iteration t is met

$$\begin{aligned} v_i^{t+1} &= v_i^t + \Delta v_i^t, \\ x_i^{t+1} &= x_i^t + \Delta x_i^t, \\ \Delta x_i^t &= v_i^{t+1} \Delta t = v_i^{t+1}, \\ \Delta v_i^t &= \alpha \varepsilon_1 [g^* - x_i^t] + \beta \varepsilon_2 [x^* - x_i^t]. \end{aligned} \tag{2.6}$$

Noting that $\varepsilon_1, \varepsilon_2$ are random numbers uniformly distributed in the interval $[0, 1]$, at iteration t, the best solution of the population is g^*, and x^* presents the individual best solution for particle i at iteration t.

Firefly Algorithm (FA)

The main characteristics of FA are found by imitating the attraction and flashing behavior of tropical fireflies.

At iteration t the firefly i has the position vector x_i^t, and it can be updated using the equation:

$$x_i^{t+1} = x_i^t + \Delta x_i^t, \tag{2.7}$$

where

$$\Delta x_i^t = \beta_0 e^{(-\gamma r_{ij}^2)}(x_i^j - x_i^t). \tag{2.8}$$

The attractiveness at zero distance ($r_{ij} = 0$) is indicated by β_0.

The visibility of fireflies is controlled by the scale-depending parameter γ, and the strength of randomization in FA is controlled by α.

Bat Algorithm (BA)

This algorithm is inspired by the echolocation of micro-bats and the associated frequency-tuning characteristics in a range from $f_{\min}$ to $f_{\max}$, combining by varying purse emission rate and loudness. The update of bat position is achieved through:

$$\begin{aligned} v_i^t &= v_i^{t-1} + \Delta v_i^t, \\ x_i^t &= x_i^{t-1} + \Delta x_i^t, \end{aligned} \tag{2.9}$$

Where

$$\begin{aligned} \Delta v_i^t &= (x_i^{t-1} - x_*)[f_{\min} + \beta(f_{\min} - f_{\max})], \\ \Delta x_i^t &= v_i^t \Delta t = v_i^t, \end{aligned} \tag{2.10}$$

best solution on the population of n bats is x_*, a random number on $[0, 1]$ is noted by β.

Cuckoo Search (CS)

Cuckoo Search mimic the strategy of of the aggressive reproduction done by some special cuckoo species and their interaction with host species. A discovery or an abundance of eggs laid by cuckoo is realized with a probability $p\alpha$, the similarity of two solutions or eggs (x_j, x_k) is measured by the quantity $(x_j - x_k)$. So the position of eggs x_i^t at iteration t is given by:

$$x_i^{t+1} = x_i^t + \Delta x_i^t, \tag{2.11}$$

and

$$\Delta x_i^t = \alpha s \otimes (x_i^t - x_k^t). \tag{2.12}$$

To limit the strength of the step size s it scaled by a parameter α, s follows lévy distribution with an exponent λ. Some sophisticated algorithms like the Mantegna's Algorithm is used to create a generation of this step size.

Flower Pollination Algorithm (FPA)

The focal point of this FPA involves the pollination process and the features of flowering plants, primarily utilizing both biotic and abiotic pollination, while emphasizing flower constancy. The pollen x_i (the solution vector) can be updated by:

$$x_i^{t+1} = x_i^t + \Delta x_i^t. \tag{2.13}$$

Where

$$\Delta x_i^t = \begin{cases} \gamma L(\lambda)(g_* - x_i^t), \text{ if } f < p, \\ \varepsilon(x_i^t - x_k^t), \quad \text{otherwise.} \end{cases} \tag{2.14}$$

r is drawn by a uniform distribution in $[0, 1]$, γ is considered as a scaling parameter. At an iteration t the best solution is g_*, and L follows Lévy fly with an exponent λ.

Additional nature-inspired algorithms are presented here:

Simulated Annealing, Gravitational Search, Charged Particle System, Black-hole Algorithm, Eagle Strategy ...etc, where the way's generation of Δx_i^t and Δv_i^t is the main difference between in these algorithms.

The increments of the solution's position and velocity differ from one algorithm to another, and in some algorithms, there is no need for velocity. This distinction may be evident in the form of updating, or not. Such as in differential evolution and particle swarm optimization, the two equations $x_i^{t+1} = x_i^t + \Delta x_i^t$,may appear similar at first glance, but a significant distinction lies between them. In the Differential Evolution (DE) method, the increment is achieved through random permutation, whereas in Particle Swarm Optimization (PSO), it is calculated using a difference vector with perturbed directions, employing a uniform distribution. For a deeper understanding, it is essential to analyze the underlying search mechanisms along with their mathematical or statistical foundations.

2.6 Other Classifications

Researchers have delved into alternative classifications of metaheuristics. For instance, Gendreau and Potvin [33] categorized metaheuristics into two main groups: trajectory-based and population-based. In trajectory-based algorithms, the process initiates with

a single solution, and at each iteration, the current best solution is substituted by a new one. Conversely, population-based algorithms kick off by generating a population of initial solutions randomly. This initial population undergoes iterative improvements, with newly generated superior solutions replacing either the entire population or a portion of it. Generally, trajectory-based solutions tend to be more exploitation-oriented, focusing on refining the current best solution, while population-based metaheuristics lean towards exploration, emphasizing the generation and exploration of diverse solutions within the population.

Adding to this discourse, Fister et al. [21] introduced another classification framework, dividing all existing metaheuristics into non-nature inspired and nature-inspired categories. Within the nature-inspired category, further distinctions include swarm intelligence (SI) based, bioinspired (excluding SI), and physics/chemistry-based metaheuristics. A miscellaneous grouping is reserved for algorithms that defy classification under the previous categories, as they draw inspiration from a variety of characteristics originating from different sources such as social and emotional influences. It is crucial to note that this is just some among many classification perspectives, as a multitude of alternative frameworks exist, underscoring the diverse nature of metaheuristic optimization techniques.

2.7 Search Mechanisms and Theoretical Foundations

The 'evolution' in any computational evolutionary algorithm is driven by its search mechanisms. By analyzing these foundational components, such as random permutations and various types of random walks, we can understand precisely how a population of solutions improves over time.

The primary distinction among the search mechanisms of nature-inspired algorithms lies in their probability distributions. By considering the statistical foundations and the principal mechanisms of solution modifications, we can distinguish five ways or perturbations:

gradient-guided moves GGM, random permutation RP, directed-based perturbations DBP, isotropic random walks IRW, and long-tailed scale-free random walks LTRW.

2.7.1 Gradient-Guided Moves

Moves of this kind are employed in gradient-based optimization, as seen in the Newton-Raphson method, where modifications are made parallel to the gradient direction, and the step length can be controlled by a learning parameter.

2.7.2 Random Permutation

This permutation is performed by mixing up a set of n solutions and then randomly selecting k solutions to generate new ones. Several algorithms, such as DE and FPA, utilize these random permutations.

2.7.3 Direction-based Perturbations

These moves are executed by utilizing the difference between any two vectors $(x_j - x_k)$ or $(g_* - x_i)$. This difference determines a direction, which is then multiplied by a random number ε distributed uniformly. The directions of these permutations are randomly distributed within a cone.

2.7.4 Isotropic Random Walks

Algorithms utilizing these moves have the following equations for updating solutions:

$$x_i^{t+1} = x_i^t + \omega^{t+1}, \tag{2.15}$$

where x_i^t is the current solution, x_i^{t+1} is the solution after perturbation and ω^{t+1} are random numbers drawn from Gaussian distribution. $\omega^{t+1} \sim N(0,1)$. So, these random walks match a Brownian motion. The pseudotime counter t can be replaced by the number

of steps, given that iteration time is discrete. This implies that the average distance d_N covered by a Brownian random walk is proportional to $\sqrt{N}$. This square-law feature is typical for many diffusion phenomena.

In conclusion, isotropic random walks are the moves where steps are drawn from Gaussian distributions.

2.7.5 Long-tailed, Scale-free Random Walks

Algorithms with long-tailed, scale-free random walks, use a heavy-tailed or long-tailed distribution to draw the steps of moves, such as Cauchy distribution

$$p(x, \mu, \gamma) = \frac{1}{\pi\gamma}(\frac{\gamma^2}{(x-\mu)^2 + \gamma^2}), -\infty < x < +\infty, \tag{2.16}$$

where μ and γ are two parameters, it have an infinite or an undefined mean or variance.

A very important example of long-tailed distribution is Lévy distribution.

A random walk with steps drawn from Lévy distribution is called a Lévy flight.

Lévy probability distribution can be defined by:

$$p(x) = \frac{1}{\pi}\int_0^\infty \cos(kx)e^{-\alpha|k|^\beta}dk, 0 < \beta \leq 2, \alpha > 0. \tag{2.17}$$

This distribution becomes a Cauchy distribution if $\beta = 2$ and a normal distribution if $\beta = 2$.

Approximations for large steps can be used

$$L(s) \sim \frac{\alpha\beta\Gamma(\beta)\sin(\frac{\pi\beta}{2})}{\pi|s|^{1+\beta}}, s \gg 0, \tag{2.18}$$

with $\Gamma(\beta)$ is the standard gamma function.

The variance or distance covered by the Brownian random walks increases much more slowly than the distance covered by a Lévy Flight. After N steps, the mean distance

covered by a Lévy flight becomes:

$$d_N \propto N\frac{(3-\beta)}{2}. \tag{2.19}$$

This power-law is a typical feature for super-diffusion phenomena.

Search mechanisms of some nature-inspired algorithms and their underlying statistical characteristics are summarized in Table 2.1, while Table 2.2 provides a summary of their search characteristics.

Algorithm	**Position Increment** Δx	**Velocity Increment** Δv
Newton-Raphson	GGM	None
DE	RP, DBP	None
PSO	DBP	DBP
FA	DBP, IRW	None
BA	RP, DBP	RP, DBP
CS	RP, DBP, LTRW	None
FPA	DBP, LTRW	None
SA	IRW	None

Table 2.1: Search Mechanisms of Some Nature-Inspired Algorithms

Algorithm	**Probability**
DE	Uniform
	Permutation
PSO	Uniform
FA	Gaussian
	Uniform
BA	Uniform
CS	Lévy flights
	Long-tailed
FPA	Uniform
	Lévy flights

Table 2.2: search Characteristics of Some Nature-Inspired Algorithms

2.8 Random Walks and Lévy Flights

As we have discussed earlier, the primary characteristics of metaheuristic algorithms revolve around diversification and intensification. Recognizing the pivotal role of randomization in achieving these goals, the essence of such randomization lies in the concept of a random walk. In this section, we will provide a concise overview of the fundamentals of random walks and Lévy flights.

2.8.1 Random Variables

A random variable is an expression whose value represents the outcome or realization of events associated with a random process, such as the noise level on a street. Random variables can take real values and are classified as either discrete or continuous. Discrete random variables, like the number of cars on a road, have distinct, separate values. On the other hand, continuous random variables, such as the noise at a location, can take any value within an interval.

Additionally, random variables can be a mix of these two types, combining discrete and continuous characteristics. Mathematically, a random variable is a function that maps events to real numbers in a domain called the sample space.

To represent the probability distribution of a random variable, a probability density is employed. An illustrative example is the Poisson distribution, which models occurrences such as the number of phone calls per minute or the number of users on a web server per day. If the mean or expectation of the event during a unit interval is denoted as a parameter $\gamma > 0$, then the probability density function of the Poisson distribution is

$$p(n, \lambda) = \frac{\lambda e^{-\lambda}}{n!}, (n = 0, 1, 2, ...). \tag{2.20}$$

Many physical processes, such as light intensity and errors in measurements, follow the popular Gaussian or normal distribution, where

$$p(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(-x-\mu)^2}{2\sigma^2}}, -\infty < x < \infty, \tag{2.21}$$

with μ is the mean and $\alpha > 0$ is the standard deviation.

In the metaheuristic world, another important probability distribution is the Lévy distribution, which is the sum of N identical and independent distributions of random variables. The Fourier transform of these random variables is given by

$$F_N(k) = e^{-N|k|^\beta}. \tag{2.22}$$

Lévy-distribution can be written as:

$$L(s) = \frac{1}{\pi}\int_0^\infty \cos(\tau s) e^{-\alpha r^3} d\tau, 0 < \beta \leq 2. \tag{2.23}$$

It has an analytical form only for a few cases. When $\beta = 1$, the above integral becomes the Cauchy distribution, and for $\beta = 2$, it transforms into the normal distribution, while Lévy flights turn into the standard Brownian motion.

In mathematical terms, we can represent the preceding integral as an asymptotic series. The leading-order approximation for the flight length yields a power-law distribution

$$L(S) \sim |s|^{-1-\beta},$$

which is heavy-tailed. For $0 < \beta < 2$, his variance and moments are infinite, becoming a stumbling block for mathematical analysis.

2.8.2 Random Walks

A random walk can be defined by a process that takes series of consecutive random steps, it means that if X_i are random steps, then

$$\begin{aligned} S_N &= \sum_{i=1}^{N} X_1 + ... + X_N \\ &= S_{N-1} + X_N, \end{aligned} \tag{2.24}$$

is a random walk.

So, the next state S_N depends only on the current state S_{N-1} and the motion from the current state to the next state, denoted as X_N. This property is characteristic of a Markov chain. The step size in a random walk can be either the same or different.

Hence, random walks can be represented as $S_{t+1} = S_t + w_t$, where S_t denotes the current state at time t, and w_t is a random variable (step) following a known distribution. If the step adheres to a Gaussian distribution, then the random walk is classified as Brownian motion.

According to the central limit theorem, as the number of steps N increases, the random walk tends to approach a Gaussian distribution.

The Brownian motion follows a Gaussian distribution, meaning that $\beta(t) \sim N(0, \sigma^2(t))$.

Where

$$\sigma^2(t) = |v|^2 t^2 + (2dDt). \tag{2.25}$$

Here, v_0 is the drift velocity of the system, and $D = \frac{s^2}{2\tau}$ is the effective diffusion coefficient, which is related to the step length s over a short time interval τ during each jump.

The random walk can be more generalized if the step length follows a different distribution, such as in Lévy flight (Lévy walk), where the step length adheres to the Lévy distribution

2.8.3 Lévy Flight

Lévy flight is a type of random walk where the step length follows the Lévy distribution, $L(s) \sim |s|^{-1-\beta}$ where $0 < \beta \leq 2$.

A simplified form of the Lévy distribution can be expressed as

$$\Delta L(s,\gamma,\mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}}\exp[-\frac{\gamma}{2(s-\mu)}]\frac{1}{(s-\mu)^{3/2}}, & \text{if } 0<\mu<s<\infty, \\ 0, \quad \text{otherwise.} \end{cases} \tag{2.26}$$

$\mu > 0$ is a minimum step and γ is a scale parameter.

When $s \to \infty$, we have $L(s,\gamma,\mu) \approx \sqrt{\frac{\gamma}{2\pi}}\frac{1}{s^{3/2}}$,

this case can be considered as a special one of the generalized Lévy distribution.

On the other hand, in terms of the Fourier transform, the Lévy distribution is defined by:

$$F(k) = \exp[-\alpha|k|^{\beta}], 0 < \beta \leq 2, \tag{2.27}$$

with α is a scale parameter.

The inverse of this integral does not have an analytical form, except for special cases:

- When $\beta = 2$, the inverse Fourier transform yields a Gaussian distribution.
- If $\beta = 1$, the inverse Fourier transform corresponds to a Cauchy distribution

$$P(x,\gamma,\mu) = \frac{1}{\pi}\frac{\gamma}{\gamma^2+(x-\mu)^2}, \tag{2.28}$$

  where μ is a parameter corresponds to the location, and γ controls the scale of this distribution.

- Generally, the inverse integral can be defined as:

$$L(s) = \frac{1}{\pi}\int_0^{\infty}\cos(ks)e^{-\alpha|k|^{\beta}}dk, \tag{2.29}$$

  it can be estimated only when s is large, and we have

  $L(s) \to \frac{\alpha\beta\Gamma(\beta)\sin(\pi\beta/2)}{\pi|s|^{1+\beta}}$

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt, \tag{2.30}$$

Γ is the Gamma function.

For various reasons, when exploring an unknown, large-scale search space, Lévy flights tend to be more efficient than Brownian random walks.

One of the reasons is that the increase in variance in Lévy flights occurs much faster compared to the linear relationship in Brownian random walks.

$\sigma^2(t) \sim t^{3-\beta}, 1 \leq \beta \leq 2$ (for Lévy flights).

$\sigma^2(t) \sim t$ (for Brownian random walks).

The generation of random numbers using Lévy flights involves two stages.

The first stage entails choosing a random direction, drawn from a uniform distribution.

In the second stage, steps are generated using the Mantegna algorithm for a symmetric Lévy stable distribution. Here, 'symmetric' denotes that steps can take positive or negative values. A random variable and its probability distribution can be termed stable if a linear combination of its identical copies follows the same distribution.

Gaussian, Cauchy and Lévy distributions are stable distributions.

In Mantegna's algorithm, the step length is calculated using the formula

$$s = \frac{\mu}{|v|^{1/\beta}}, \tag{2.31}$$

where

$$\mu \sim N(0, \sigma_\mu^2), v \sim N(0, \sigma_v^2), \tag{2.32}$$

and

$$\sigma_\mu = [\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma((1+\beta)/2)2^{(\beta-1)/2}}]^{\frac{1}{\beta}}. \tag{2.33}$$

If s_0 represents the smallest step, then S follows a Lévy distribution for $|s| \geqslant s_0$, where $s \gg 0$. In reality, it is often chosen within the range of 0.1 to 1.

2.9 Intensification and Diversification:

The success of any evolutionary system, whether biological or computational, hinges on its ability to effectively balance the fundamental trade-off between intensification and diversification.

Metaheuristic algorithms prove their efficiency day by day by being applied to solve the most complicated optimization problems. This efficiency is attributed to the fact that they mimic the best features in nature. All metaheuristic algorithms share two main characteristics: intensification and diversification. Intensification involves searching locally and intensively, exploiting around the current best solution and selecting candidates found. Diversification, on the other hand, involves exploring the global solution space through large-scale randomization.

The efficiency of an algorithm is attributed to the balance between these two components, and even a small error in this balance can adversely affect the mechanism and efficiency of the algorithm. If the exploration is limited while exploitation is dominant, the system may become trapped in local optima. Conversely, if the exploitation is minimal and exploration is extensive, it can hinder the system from converging, thereby slowing down the overall search performance.

Attaining a proper balance between intensification and diversification constitutes an optimization problem that requires the application of an algorithm to achieve.

Another essential mechanism in metaheuristic algorithms is the best solutions selection. The 'survival of the fittest' criterion is commonly used, including the continuous update of the current best solution. Moreover, to guarantee that the fittest solutions are preserved and not lost in the process, elitism is often incorporated.

2.10 Ways for Intensification and Diversification:

Algorithms employ various methods to achieve a balance between diversification and intensification. Typically, metaheuristic algorithms utilize a combination of randomization and deterministic procedures to achieve exploration. This guarantees that the new generated solutions are distributed as diversely as possible within the feasible search space. We can mention here a common randomization technique, which involves

$$x_{new} = L + (U - L) * \epsilon_{\mu}, \tag{2.34}$$

where L is the lower bound and U is the upper bound. ϵ_{μ} is a random variable distributed uniformly in $[0, 1]$.

In fact, the distribution used to generate random walks can be a Lévy distribution rather than a uniform distribution, and it proves to be more efficient on a global scale. To achieve diversification, mutation and crossover are also employed. Mutation ensures that the newly generated solutions are as distinct as possible from the existing solutions, while crossover generates new solutions by swapping parts of the existing solution. This helps in limiting the degree of over-diversification.

Exploitation can be easily achieved through local random walks

$$x_{new} = x_{old} + sw, \tag{2.35}$$

Here, w is drawn from a Gaussian distribution with a zero mean, and the step size of the random walk must be very small to ensure visits only to the neighborhood.

To increase the efficiency of diversification Lévy distribution can be used to draw s with large step sizes, any distribution with long tail will help to increase the distance between such random walks.

A more selective or controlled walk around the current best, rather than any good solution, can be achieved using the following equation:

$$x_{new} = x_{best} + sw, \tag{2.36}$$

In many algorithms, there is no differentiation between exploration and exploitation; they are often intertwined and interactive. For example, in Genetic Algorithms, Harmony Search, and Bat Algorithms.

The consideration of selecting the best solutions is indispensable as it plays a pivotal role in determining the success of an algorithm. While opting for the best might prove effective in optimization problems with a unique global optimum, addressing multimodal and multi-objective challenges requires the implementation of elitism and the retention of the best solutions, proving to be efficient strategies for selecting the fittest candidates.

In addition, an efficient algorithm should incorporate a mechanism to discard the worse solutions, thereby enhancing the overall quality of the populations during evolution. This is often achieved through some form of randomization and probabilistic selection criteria, such as mutation in genetic algorithms.

The reduction of randomization is crucial for the convergence of the system. When better solutions are found and the system achieves convergence, failing to reduce the degree of randomness will slow down the convergence process.

We take Particle Swarm Optimization as an example, where randomization is decreased as the particle swarm gathers. This is due to the fact that the distance between each particle and the current global best particle becomes smaller and smaller. Another example is Differential Evolution, where randomness is reduced using the last term of the equation:

$$x_{new} = x_k + F(x_i - x_j), \tag{2.37}$$

where it decreases as the difference vector gets smaller and smaller.

Other algorithms control randomness rather than reducing it. For instance, randomness can be limited by employing a small mutation rate. In simulated annealing, the random-

ness during iterations may stay the same, but the moves or solutions are selectively chosen, and the acceptance probability becomes smaller.

Finally, in practice, the implementation's approach to the algorithm does affect the performance to some degree. Therefore, the pseudocode must provide clear guidance and should not lead to ambiguity. Consequently, validating and testing any algorithm implementation are crucial.

2.11 A Brief History of Metaheuristic and Evolutionary Algorithms

The history of metaheuristics is deeply intertwined with, and in many ways propelled by, the development of Evolutionary Algorithms. The intellectual groundwork can be traced to figures like Alan Turing. During World War II, his "heuristic search" methods proved highly successful in the effort to break German Enigma ciphers at Bletchley Park, demonstrating the power of search strategies that work most of the time without guaranteeing an optimal solution. Following the war, in a seminal 1948 report connected to his design for the Automatic Computing Engine (ACE), Turing outlined pioneering concepts in machine intelligence, Neural Networks, and principles that are now recognized as foundational to evolutionary algorithms. While Turing sowed these early seeds, it was the formalization of evolutionary concepts by pioneers such as Holland, Rechenberg, and Schwefel in the 1960s and 70s that truly launched the field as we know it today.

In the 1960s and 1970s, Genetic Algorithms were created by John Holland with his collaborators at the University of Michigan [19]. Holland studied adaptive systems and introduced crossover and recombination manipulations for modeling the system in 1962. In 1975, his book detailing the evolution of genetic algorithms was published. At the same time, De Jong demonstrated the potential and power of Genetic Algorithms in his thesis, showcasing their ability to solve a wide range of objective functions, including noisy,

multimodal, and even discontinuous ones.

At the same time, another search technique was developed by Ingo Rechenberg and Hans-Paul Schwefel for solving optimization problems in aerospace engineering in 1963. They named it evolutionary strategy. Subsequently, Peter Bienert collaborated with them to construct an automatic experimenter using selection and mutation. This was a simple trajectory-style hill-climbing algorithm combined with randomization.

"As early as 1960, Lawrence J. Fogel utilized simulated evolution to explore artificial intelligence. In 1966, he, along with A.J. Owen and M.J. Walsh, developed the evolutionary programming technique. In 1943, W. McCulloch and W. Pitts employed artificial neurons as simple information processing units. The concept of neural networks was likely first suggested by Alan Turing in his 1948 NPL report on intelligent machinery.

Significant advancements emerged from the 1940s and 1950s through the 1990s. In 1963, V. Vapnik proposed the support vector machine as a linear classification technique. Collaborating with others, he later extended it to nonlinear classification using kernel techniques in the 1990s. In 1995, V. Vapnik consolidated these techniques in his book 'The Nature of Statistical Learning Theory'.

In 1983, S. Kirkpatrick, C.D. Gellat, and M.P. Vecchi developed Simulated Annealing SA as an optimization technique, drawing inspiration from the annealing process of metals. This marked a significant milestone in metaheuristic algorithms.

Fred Glover is likely the first to have employed memory in modern metaheuristic, specifically Tabu search, in 1986. Subsequently, in 1997, he published his seminal book on Tabu search.

Ant Colony Optimization was introduced by Marco Dorigo in his PhD thesis in 1992, drawing inspiration from the swarm intelligence of social ants. Additionally, in 1992, following the publication of a treatise on genetic programming by John R. Koza of Stanford University, a new era in machine learning that revolutionized computer programming emerged.

In 1995, the American social psychologist James Kennedy and engineer Russel C. Eberhart drew inspiration from the swarm intelligence observed in fish and birds to develop Particle Swarm Optimization. Since its discovery, many researchers have developed more than twenty different variants of Particle Swarm Optimization.

Around 1996 and later in 1997, Differential Evolution was developed by R. Storn and K. Price, marking a vector-based evolutionary algorithm.

In 1997, D.H. Wolpert and W.G. Macready published 'No Free Lunch Theorems for Optimization,' proving that there is no universally best optimization algorithm for all problems. If algorithm A performs better than algorithm B for a specific problem, then algorithm B will outperform algorithm A for other problems. Subsequently, researchers shifted their focus to discovering the best and most efficient algorithm(s) for a given problem rather than seeking a universal solution for all problems.

In 2001, the Harmony Search algorithm was developed by Zang Woo Geem et al. and found widespread applications in solving different optimization problems, involving water distribution, transport modeling, and scheduling. In 2004, the Honey Bee Algorithm was proposed by S. Nakrani and C. Tovey, initially applied for optimizing Internet hosting centers. Subsequently, in 2005, D.T. Pharm et al. developed a novel Bee Algorithm. In the same year, D. Karaboga introduced the Artificial Bee Colony algorithm.

Xin-She Yang has developed various optimization algorithms, including the Firefly Algorithm in 2008 and Cuckoo Search in 2009 in collaboration with Suash Deb. In 2010, he introduced the Bat-inspired Algorithm for continuous optimization. Since then, several metaheuristic algorithms have been developed. We can mention some of them:

- Harmony Search (2001)
- Artificial Immune System (2002)
- Chemical Reaction Inspired Optimization (2010)
- Flower Pollination Algorithm (2012)

- Water Wave Optimization Algorithm (2015)
- Sine Cosine Algorithm (2016)...etc

2.12 Conclusion

Having established this broad framework for understanding algorithms as evolutionary systems based on their management of diversification and intensification, we are now prepared to analyze a specific case study. The next chapter will deconstruct the Flower Pollination Algorithm through this evolutionary lens.

Chapter 3

Flower Pollination Algorithm

Flowers, enchanting embodiments of plant life, possess a unique charm for humanity. They are cultivated in our gardens, adorn the interiors of our homes, and have left a lasting impact on our artistic legacy, embedding themselves deeply within the fabric of our lives. Beyond their visual appeal, flowers have sparked significant scientific interest, giving rise to a rich tradition of botanical research aimed at deciphering the complexities of their structure and ecology. This profound connection between humans and flowers goes beyond simple aesthetic enjoyment, extending into the realms of art, science, and a collective admiration for the natural world's marvels. Recently, this fascination has led humanity to leverage these characteristics, incorporating them into an algorithm within the field of artificial intelligence, inspired by their remarkable methods and organization. In the forthcoming chapter, we will analyze these biological features through an evolutionary lens, deconstructing the Flower Pollination Algorithm to understand its structure, mechanisms, and potential applications.

3.1 Flowers and Flowering

Commencing our exploration, we delve into the intricate mechanisms of pollination, unveiling the sophisticated processes and adaptations that foster successful pollen transfer.

3.1.1 Cross-Pollination and Self-Pollination

Cross-pollination is when pollen from one flower's male part goes to another flower's female part in a different plant, ensuring successful reproduction among a group of flowering plants.

In contrast, self-pollination happens when pollen moves from the male part of a flower to the female part within the same flower or between different flowers on the same plant.

Due to their stationary nature, plants rely on a pollen vector to facilitate the transfer of pollen between them. This essential transport mechanism can be facilitated by either abiotic or biotic agents acting as vectors.

Pollen relies on both non-living agents, mainly wind and water as abiotic vectors, and living organisms predominantly insects, birds, bats, and select vertebrates as biotic vectors for its essential transfer between flowers.

3.1.2 Flower Constancy

Honeybees exemplify effective pollinators or biotic vectors, showcasing a behavior known as flower constancy. This phenomenon entails these pollinators consistently visiting specific flower species while disregarding others. Flower constancy is believed to confer evolutionary advantages by optimizing the transfer of pollen to the same or conspecific plants, thereby maximizing the reproductive success of particular flower species. This behavioral trait may also benefit the pollinators themselves, ensuring a reliable nectar supply with their limited memory and minimizing the costs associated with learning or exploration. Rather than expending energy on unpredictable but potentially more rewarding new flower species, flower constancy demands minimal investment costs and provides a more secure intake of nectar.

In biotic cross-pollination, the process can extend over significant distances, facilitated by pollinators like bees, bats, birds, and flies that possess the ability to cover vast areas, earning them the designation of global pollinators. Notably, bees and birds may exhibit

Lévy flight behavior, where their movement involves jumps or flights conforming to a Lévy distribution. Additionally, the concept of flower constancy serves as an incremental step, incorporating the similarity or difference between two flowers into the pollination process.

3.2 The Algorithm

We can distill the characteristics of the pollination process, flower constancy, and pollinator behavior into the following rules:

- Biotic cross-pollination acts as a global pollination process, where pollen-carrying pollinators execute Lévy flights.
- Abiotic and self-pollination are categorized as local pollination.
- Flower constancy can be seen as the probability of reproduction being directly linked to the similarity of the two flowers involved.
- Local pollination and global pollination are governed by a switch probability, denoted as $p \in [0, 1]$. Owing to physical proximity and factors like wind, local pollination can contribute a significant fraction, denoted as p, to the overall pollination activities.

In reality, each plant usually has multiple flowers, and a single flower patch often releases millions, if not billions, of pollen gametes. However, for simplicity, Xin-She Yang assumes that each plant has only one flower, and each flower produces a single pollen gamete. This simplification allows him to consider a solution x_i as interchangeable with a flower and/or a pollen gamete, eliminating the need to distinguish between them in problem-solving scenario.

The primary attributes and components of the Flower Pollination Algorithm (FPA) are concisely depicted in Table 3.1 This table elucidates the correlation or equivalency between the terminologies used in optimization strategies and their counterparts in the context of

Flower Pollination	Optimization Components
Pollinators(insects, butterflies, birds)	Moves/ modification of variables
Biotic	Global search
Abiotic	Local search
Lévy flight	Step size (obeying power law)
Pollen/flower	Solution vectors
Flower constancy	Similarity in solution vector
Evolutionof flowers	Iteration evolution of solutions
Optimal flower reproduction	Optimal solution set

Table 3.1: Pollination Process and its Optimization Components

floral pollination. By drawing these parallels, the table effectively bridges the conceptual gap between the natural processes that inspire FPA and its application in solving optimization problems.

Based on the aforementioned characteristics, the Flower Pollination Algorithm is made up of two fundamental steps: global pollination and local pollination.

During the global pollination step, pollinators such as insects move flower pollen over long distances, benefitting from their ability to fly and cover extensive ranges. This process guarantees the pollination and reproduction of the fittest, represented as g_*. The first rule, along with the concept of flower constancy, can be expressed mathematically as

$$x_i^{t+1} = x_i^t + L(x_i^t - g_*). \tag{3.1}$$

Here, x_i^t denotes the pollen i or solution vector x_i at iteration t, and g_* represents the current best solution among all solutions in the current generation/iteration. The parameter L denotes the strength of pollination, essentially serving as a step size.

As insects often traverse long distances with varying step lengths, employing a Lévy flight allows us to efficiently mimic this characteristic. Where

$$L \sim \frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}}, s \gg s_0 > 0 \tag{3.2}$$

Where $\Gamma(\lambda)$ signifies the standard gamma function, and this distribution is suitable for large steps ($s > 0$). In all simulations of the algorithm presented below $\lambda = 1.5$.

Expressing the local pollination (Rule 2) and flower constancy can be done as follows:

$$x_i^{t+1} = x_i^t + \epsilon(x_j^t - x_k^t), \tag{3.3}$$

where x_j^t and x_k^t represent pollens obtained from distinct flowers within the same plant species.

This effectively simulates flower constancy within a limited neighborhood. Mathematically, if x_j^t and x_k^t originate from the same species or are chosen from the same population, this process transforms into a local random walk, especially when we draw ϵ from a uniform distribution in $[0, 1]$.

The dynamics of flower pollination extend across various scales, encompassing both local and global levels.

In practice, nearby flower patches or those within the immediate vicinity are more likely to receive pollination from local flower pollen than those situated farther away. To account for this, we introduce a switch probability (Rule 4) or proximity probability p to transition between widespread global pollination and concentrated local pollination.

Initially, set p to 0.5, and subsequently, conduct a parametric study to determine the optimal parameter range. Through simulations, it has been observed that $p = 0.8$ proves to be more effective across various applications.

The two critical steps mentioned above, along with the switch condition, are succinctly encapsulated in the pseudocode presented by the Table 3.2

3.2.1 Numerical Results

Any novel optimization technique requires thorough validation, including a comprehensive comparison with other existing algorithms. While there exist numerous test functions-well over a hundred, each serving specific evaluation purposes-there is a notable absence of a

Flower Pollination Algorithm Pseudo-code

Objective min $f(x)$,$x \in \mathbb{R}^d$
Initialize a population of n flowers/pollen with random solutions
Find the best solution B in the initial population
Define a switch probability $p \in [0, 1]$
Calculate all $f(x)$ for n solutions
$t = 0$
While $t \leq \max generation$ **do**
  for $i = 1, ..., n$ **do**
    if $rnd \leq p$ **then**
      Draw a (d-dimensional) step vector L which obeys a Lévy distribution
      Global pollination via $x_i^{t+1} = x_i^t + L * (B - x_i^t)$
    else
      Draw from a uniform distribution $U \in [0, 1]$
      Randomly choose j and k among all solutions
      Do local pollination via $x_i^{t+1} = x_i^t + U * (x_j^t - x_k^t)$
    end if
    Calculate all new $f(x_i^{t+1})$
    if $f(x_i^{t+1}) \leq f(x_i^t)$ **then**
      $x_i^t = x_i^{t+1}$
    end if
  end for
  Find the current best solution B among all x_i^t
  $t = t + 1$
end While

Table 3.2: Flower Pollination Algorithm Pseudo-code

universally agreed-upon set of test functions for validating emerging algorithms.

In the original research paper introducing the Flower Pollination Algorithm (FPA), the author thoughtfully curated a diverse subset of these test functions. This subset was specifically chosen to rigorously evaluate and validate the efficacy of the proposed FPA against established benchmarks.

Test Functions

The Ackley function can be expressed as

$$a \exp[-b\sqrt{\frac{1}{d}\sum_{i=1}^{2} x_i^2}] - \exp[\frac{1}{d}\sum_{i=1}^{d} \cos(2\pi x_i)] + a + \exp(1), \tag{3.4}$$

where a, b and c are constants that determine the characteristics of the function. Common choices for these constants are $a = 20$, $b = 0.2$ and $c = 2/\pi$.

\newline

which attains its global minimum, denoted as $f_* = 0$, precisely at the point $(0, 0, ..., 0)$.

The most basic among De Jong's functions is the sphere function, represented by:

$$\sum_{i=1}^{d} x^2, -5.12 \leq x \leq 5.12, \tag{3.5}$$

Manifesting a clear global minimum at $f_* = 0$, situated at the origin $(0, 0, ..., 0)$, the sphere function is characterized by its unimodal nature and convexity.

Easom's function is a mathematical optimization test function often used to assess the performance of optimization algorithms. It is defined in two dimensions as:

$$f(x) = -\cos(x)\cos(y)\exp[-(x-\pi)^2 + (y-\pi)^2], \tag{3.6}$$

Its global minimum occurs when $f_* = -1$ at the point $x_* = (\pi, \pi)$ within $-100 \leq x \leq 100$. It possesses numerous local minima.

Griewank's function is a commonly used mathematical optimization benchmark. It is expressed by the following equation:

$$f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=0}^{d} \cos(\frac{x_i}{\sqrt{i}}), -600 \leq x_i \leq 600. \tag{3.7}$$

This function is characterized by its complex landscape, involving a balance between the quadratic term and a product of cosine functions. Griewank's function is often employed to evaluate the robustness and efficiency of optimization algorithms due to its multimodal nature.

Michaelwicz's function

$$f(x) = -\sum_{i=1}^{d} \sin(x_i).[\sin(\frac{2x_i^2}{\pi})]^{2m}, \tag{3.8}$$

m is a constant typically set to 10 and $0 \leq x_i \leq \pi$ for $i = (1, 2, ..., d)$. In the case of two dimensions, the Michaelwicz's function will be:

$$f(x, y) = -\sin(x)\sin^{20}(\frac{x^2}{\pi}) - \sin(y)\sin^{20}(\frac{2y^2}{\pi}), \tag{3.9}$$

Where $(x, y) \in [0, 5] \times [0, 5]$. It has a global mimnimum $f_* = -1.8013$ at $x_* = (x_*, y_*) = (2.20319, 1.57049)$.

Rastrigin's function is a non-convex mathematical function commonly used for testing optimization algorithms. It is named after its creator, Leonid Rastrigin, a Russian mathematician. The function is often employed to evaluate the performance of optimization algorithms due to its multimodal and rugged nature.

The d-dimensional version of Rastrigin's function is defined as follows:

$$f(x) = A.d + \sum_{i=1}^{d} [x_i^2 - A.\cos(2\pi x_i)], -5.12 \leq x_i \leq 5.12, \tag{3.10}$$

where A is a constant, often set to 10. Rastrigin's function is characterized by numerous local minima, and the global minimum is at $f_* = f(0, 0, ..., 0) = 0$.

The Rosenbrock's function, often referred to as the Rosenbrock's valley or the Banana function, is a non-convex mathematical function used for testing optimization algorithms. It is named after Howard H. Rosenbrock, who introduced it in 1960. The function is particularly interesting due to its long, narrow, parabolic shape, which poses challenges for optimization methods.

The n-dimensional version of Rosenbrock's function is presented as follows:

$$f(x) = \sum_{i=1}^{d-1} [100.(x_{i+1} - x_i^2)^2 + (1 - x_i)^2], \tag{3.11}$$

Rosenbrock's function has a global minimum at $f_* = f(1, 1, ..., 1)$, in the domain $-5 \leq x_i \leq 5$ for $i = 1, 2, ..., d$.

For $d = 2$, banana function is often written as:

$$f(x, y) = (x - 1)^2 + 100.(y - x^2)^2. \tag{3.12}$$

Schwefel's function is a mathematical optimization problem frequently used in benchmarking and testing optimization algorithms. It was introduced by R. M. Schwefel, a German computer scientist, and it is known for its challenging and rugged landscape.

Schwefel's Problem is expressed in n dimensions and defined as follows:

$$f(x) = -\sum_{i=1}^{d} x_i \sin(\sqrt{|x_i|}), -500 \leq x_i \leq 500. \tag{3.13}$$

With a global minimum $f_* = -418.9829d$ achieved at 420.9687 for all $i = 1, 2, ..., d$.

Function/Algorithms	GA	PSO	FPA
Michalewiez ($d = 16$)	$89325 \pm 7914(95\%)$	$6922 \pm 537\,(98\%)$	$3341 \pm 649\,(100\%)$
Rosenbrock ($d = 16$)	$55723 \pm 8901\,(90\%)$	$32756 \pm 5325\,(98\%)$	$5532 \pm 1464\,(100\%)$
De Jong ($d = 256$)	$25412 \pm 1237\,(100\%)$	$17040 \pm 1123\,(100\%)$	$4245 \pm 545\,(100\%)$
Schwefel ($d = 128$)	$227329 \pm 7572\,(95\%)$	$14522 \pm 1275\,(97\%)$	$6851 \pm 448\,(100\%)$
Ackley ($d = 128$)	$32720 \pm 3327\,(90\%)$	$23407 \pm 4325\,(92\%)$	$3357 \pm 968\,(100\%)$
Rastrigin	$110523 \pm 5199\,(77\%)$	$79491 \pm 3715\,(90\%)$	$10840 \pm 2689\,(100\%)$
Easom	$19239 \pm 3307\,(92\%)$	$17273 \pm 2929\,(90\%)$	$4017 \pm 982\,(100\%)$
Griewank	$70925 \pm 7652\,(90\%)$	$55970 \pm 4223\,(92\%)$	$4918 \pm 1429\,(100\%)$

Table 3.3: Algorithm Performance Comparison Based on the Number of Iterations

Xin-She Yang employed three algorithms Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and the novel Flower Pollination Algorithm (FPA) to determine their optimal solutions within a specified tolerance of 10^{-5}. For each algorithm, he conducted 100 independent simulations, utilizing a population size of $d = 25$ and setting $p = 0.8$ for

the FPA, a crossover probability of 0.95 and a mutation probability of 0.05 for the GA, along with learning parameters set to 2 for the PSO. The outcomes of these simulations are concisely summarized in Table 3.3, where the results are presented as the mean ± standard deviation, alongside the success rate. For instance, a result of $3341 \pm 649(100\%)$ indicates an average number of iterations at 3341, with a standard deviation of 649, and a success rate of 100%.

The total number of function evaluations is calculated as d times the average number of iterations. For example, with 3341 iterations listed in the table, the total number of function evaluations is computed as $3341 \times d = 3341 \times 25 = 83525$.

Out of the three methods, the proposed Flower Pollination Algorithm (FPA) achieved the optimal outcome and demonstrated the fastest convergence.

Design Optimization

Design optimization embodies a systematic approach to enhance and refine the attributes of a product or system, with the objective of achieving the best possible performance according to specific criteria and objectives. In the realm of bottle design, this entails a thorough evaluation of various aspects to bolster the bottle's functionality, sustainability, aesthetics, and market attractiveness.

For a specified volume and operating pressure, the primary goal in designing a cylindrical vessel is to minimize the total cost. Typically, the design variables include the thickness d_1 of the head, the thickness d_2 of the body, the inner radius r, and the length L of the cylindrical section. This scenario presents a classic optimization challenge, which can be articulated as follows:

minimize $f(x) = 0.6224 d_1 r L + 1.7781 d_2 r^2 + 3.1661 d_1^2 L + 19.84 d_1^2 r$, guided by the ensuing constraints:

- $g_1(x) = -d_1 + 0.0193r \leq 0$,
- $g_2(x) = -d_2 + 0.00954r \leq 0$,

- $g_3(x) = -\pi r^2 L - \frac{4\pi}{3} r^3 + 1296000 \leq 0,$
- $g_4(x) = L - 240 \leq 0,$

with simple bounded

$$0.06251 d_2 \leq 99 \times 0.0625, \text{ where } 10 \leq r, L \leq 200.$$

In 2008, Cagnina et al. employed an advanced Particle Swarm Optimization technique to address this issue, successfully identifying the optimal solution:

$$f_* \approx 6059.714,$$

at

$$x_* \approx (0.8125, 0.4375, 42.0984, 176.6366).$$

This implies that the minimum price is approximately 6059.71.
Utilizing the suggested Flower Pollination Algorithm, Xin-She Yang effortlessly obtained a solution, $f_* \approx 6059.714$, which aligns with the result achieved by Cagnina et al.
This conclusively establishes the effectiveness and robust performance of the proposed Flower Pollination Algorithm (FPA).

Discussion

Flowering plants exhibit intriguing characteristics in their flower pollination processes, inspiring the successful development of a novel flower algorithm by the author. Simulation outcomes robustly demonstrate the exceptional efficiency of the proposed flower pollination algorithm, surpassing the performance of both genetic algorithms and particle swarm optimization. Notably, the algorithm exhibits an exponential convergence rate, as evidenced by the comparison presented in the previously.

The efficiency of the FPA can be attributed to two main factors: long-distance pollination and flower consistency. Insects, serving as long-distance pollinators, possess the capability to traverse vast distances, enabling them to escape local landscapes and explore larger search spaces. This aspect facilitates exploration within the algorithm. Conversely, flower consistency ensures that similar solutions from the same species of flowers are selected more frequently, thereby promoting quicker convergence. This aspect serves as an exploitation step. The synergy between these key components and the selection of the best solution g_* guarantee the algorithm's high efficiency.

3.3 Variants of Flower Pollination Algorithm

Though the standard FPA works well for many applications [23], it can still be improved. Given the complex nature of real-world optimization problems, researchers have modified the basic structure of FPA to enhance its performance. The modifications have been made in various parts of the FPA structure. Furthermore, several FPA hybridization schemes have been implemented to accelerate convergence and improve the balance between exploration and exploitation. Researchers have also produced a multi-objective version of FPA specifically designed for the category of multi-objective optimization [60].

Modified Flower Pollination Algorithm Based on Operators

Yamany et al. [50] introduced a modified FPA based on an attribute reduction approach. The primary objective of their algorithm is to address the challenges posed by a potentially large search space. This approach recommends a minimal set of attributes while achieving comparable, if not superior, classification accuracy compared to utilizing all attributes and traditional attribute reduction techniques. The algorithm's strategy enhances three new initialization phases, driven by forward selection and backward selection. Their proposed technique was evaluated on eight datasets from the UCI machine learning benchmarks,

demonstrating superior performance compared to other metaheuristic algorithms such as GA and PSO.

Zhou et al. [59] introduced an elite Opposition-based Flower Pollination Algorithm (EOFPA), a novel variant designed for solving function optimization and structure designs. EOFPA demonstrated an enhancement in the balance between exploration and exploitation. The authors evaluated their proposed algorithm across 18 standard benchmark functions, yielding impressive results.

For addressing economic load dispatch problems in power generation systems, Sarjiya et al. [46] presented a modified FPA (MFPA). This approach incorporated a dynamic switching probability, employed real-coded GA (RCGA) as a mutation for both global and local search, and distinguished between temporary local search and the optimal solution. The performance of MFPA was then assessed across 10 benchmarks of power systems, with experimental results indicating a lower fuel cost compared to that obtained by the standard FPA and other similarly applicable solutions for comparable economic dispatch problems. In a separate study, Regalado et al. [25] proposed MFPA to optimize fuel cost value and the time required to achieve a global optimal solution. When tested on the IEEE 30-bus test system, MFPA demonstrated superior results over the standard FPA and other metaheuristic optimization algorithms.

To reduce real power losses and enhance bus voltages, Namachivayam et al. [15] proposed the Modified Flower Pollination Algorithm (MFPA) for network reconfiguration and optimal placement of shunt capacitors. The proposed algorithm incorporates the adaptation of the local search from the standard Flower Pollination Algorithm (FPA) and augments the global search through a dynamic switching probability approach. The efficacy of their proposed MFPA was assessed using 118-bus, 69-bus, and 33-bus radial distribution test feeders. The results demonstrated superior performance compared to other metaheuristic algorithms, including Harmony Search algorithm (HS), Simulated Annealing (SA), and Improved Binary Particle Swarm Optimization (IBPSO).

The Modified Flower Pollination Algorithm (MFPA) introduced by Dubey et al. [17] offers a solution to economic dispatch issues in large-scale power systems through a two-phase enhancement process. Initially, it incorporates a scaling factor to improve the algorithm's local search capability. The second phase intensifies the search for the optimum solution by focusing on exploitation. Tested across various mathematical benchmarks and four substantial power systems, the MFPA's performance was benchmarked against recent economic dispatch methods, showcasing its effectiveness and the promising results of this novel approach.

Binary Versions of Flower Pollination Algorithm

The initial Flower Pollination Algorithm (FPA) was specifically crafted to address continuous optimization challenges. However, to extend its application to discrete and combinatorial optimization problems, significant adjustments are necessary. Rodrigues et al. [10] introduced a variant of this algorithm, termed the Binary Flower Pollination Algorithm (BFPA), which was specifically engineered for feature selection tasks. Upon evaluating the performance of BFPA across six distinct datasets, it was found that BFPA outperformed several established algorithms in this domain, including Particle Swarm Optimization (PSO), Harmony Search (HS), and the Firefly Algorithm (FA), showcasing its superior efficacy in handling feature selection challenges.

Later, Rodrigues et al. [11] employed the Binary Flower Pollination Algorithm (BFPA) to tackle the challenge of minimizing the number of sensors necessary for identifying individuals through EEG signals. The BFPA was strategically utilized to identify the most effective subset of channels that could deliver the highest accuracy in recognition. The outcomes of experiments utilizing BFPA demonstrated that it could achieve recognition rates as high as 87% based on the Optimum-Path Forest classifier, underscoring the algorithm's potential in enhancing sensor efficiency for EEG-based person identification.

Shilaja et al. [6] introduced a method named CEED, aimed at addressing 20 photovoltaic

and 5 thermal power generation challenges. The CEED approach integrates the Economic Dispatch Euclidean Affine Flower Pollination Algorithm with the Binary Flower Pollination Algorithm (BFPA). Furthermore, when the CEED method was applied to the IEEE 30 bus and IEEE 57 bus systems for testing, it demonstrated superior performance compared to traditional techniques.

Dahi et al. [61] undertook a comprehensive investigation to assess the efficacy of the Binary Flower Pollination Algorithm (BFPA) in tackling the Antenna Positioning Problem (APP). The evaluation of BFPA was conducted using a mix of realistic, synthetic, and randomly generated datasets of varying dimensions. Its performance was benchmarked against that of population-based incremental learning (PBIL) and the Differential Evolution (DE) algorithm, both recognized as proficient solutions within the APP field. The outcomes revealed that BFPA delivered more competitive results in the domain of APP when compared to PBIL and DE, showcasing its robustness and efficiency.

Integrating Chaos for Enhanced Flower Pollination Performance

The conventional Flower Pollination Algorithm (FPA) relies on random numbers, and additional randomization can be introduced through the incorporation of chaotic maps. In the realm of mechanical engineering, Meng et al. [37] have introduced a refined version termed the Modified Flower Pollination Algorithm (MFPA) designed to address a specific design problem. This enhanced algorithm integrates adaptive inertia weight and chaos theory to bolster local search capabilities. When assessing the performance of MFPA across five mechanical engineering benchmarks-speed reducer, gear train, tubular column design, pressure vessel, and tension/compression spring design the outcomes surpassed those of alternative algorithms, demonstrating its superior efficacy in solving mechanical engineering problems.

Metwalli et al. [35] proposed an innovative approach to address fractional programming problems (FPPs) by leveraging the development of a Chaos-based Flower Pollination Al-

gorithm (CFPA). The efficacy of CFPA was validated across various FPP benchmarks, showcasing its robust performance. A comparative analysis with other metaheuristic solution methods for FPPs demonstrated the clear superiority of the proposed algorithm, solidifying its position as a leading approach for addressing fractional programming problems.

Numerous techniques for wind speed forecasting in power systems have been proposed, but a common limitation is the absence of an efficient model for data preprocessing. Addressing this gap, Zhang et al. [51] introduced an innovative model that integrates three short-term techniques for wind speed forecasting. Their novel system incorporates Complete Ensemble Empirical Mode Decomposition Adaptive Noise (CEEMDAN), Flower Pollination Algorithm with Chaotic Local Search (CLSFPA), five neural networks, and the no negative constraint theory. CLSFPA is specifically designed to optimize the weight coefficients of the combined model. Through the evaluation of 15-minute wind speed data from four distinct farms in eastern China, the study demonstrated the remarkable effectiveness of their combined algorithms in accurately forecasting wind speed.

3.3.1 Hybridized Variants of Flower Pollination Algorithm

One primary challenge faced by metaheuristic methods lies in finding the optimal balance between comprehensive global exploration and focused local exploitation throughout the search process. Some approaches excel at extensively exploring diverse regions of the problem landscape but may fall short in fully exploiting each region. Typically, these algorithms belong to the category of population-based or swarm-based methods. Conversely, other techniques prove adept at exploiting favorable elements within a specific region of the search space, often at the cost of simultaneously exploring multiple regions. This characteristic is common in gradient-based methods or trust-region methods.

Recognizing this trade-off, research communities have endeavored to enhance the Flower Pollination Algorithm (FPA) by combining it with other algorithms. The aim is to leverage

the strengths of different methods and improve the overall performance of FPA. The various forms of such hybridization approaches are summarized below.

Hybridization with Local-Based Search Algorithms

The integration of Flower Pollination Algorithm (FPA) with Simulated Annealing (SA) for engineering optimization problems, denoted as FPSA, was pioneered in [31]. In this approach, solutions generated by FPA undergo local refinement through the SA algorithm, resulting in an enhanced search performance and accelerated convergence. Notably, FPSA demonstrated superior performance compared to existing methods documented in the literature.

Jensi and Jiji [39] introduced a hybridization of the flower pollination algorithm (FPA) with the K-Means algorithm for data clustering. They employed the K-Means algorithm to bolster the local exploitation capabilities of FPA. The hybrid algorithm they proposed exhibited superior performance compared to utilizing either classical K-Means or FPA in isolation.

Emad Nabil [13] unveiled an enhanced hybrid variant of the Flower Pollination Algorithm (FPA), fusing elements from the Modified FPA (MFPA) and the cuckoo search algorithm (CSA). In the evaluation of MFPA's performance, a total of 23 optimization benchmark problems were employed for testing. The efficiency of MFPA was benchmarked against Simulated Annealing (SA), Genetic Algorithm (GA), FPA, Bat Algorithm (BA), and Firefly Algorithm (FA). The outcomes revealed that the proposed hybrid MFPA consistently outperformed both the standard FPA and the other four metaheuristic algorithms, demonstrating its enhanced efficacy in optimization tasks.

A novel hybridization of the flower pollination algorithm (FPA) with the Path Relinking metaheuristic was introduced in 29, specifically applied in the creation of health-conscious and nutritious meals for older adults. This innovative combination sought to enhance the exploration of optimal or near-optimal personalized menu recommendations, focusing on

improving both execution time and solution quality. Performance testing of this hybrid version, conducted on real-world datasets, revealed the algorithm's superiority over the conventional FPA in terms of both solution quality and execution time.

Hybridization with Population-Based Algorithms

Abdel-Raouf et al. [36] proposed an innovative hybrid approach known as the hybrid FPA for addressing optimization challenges. This unique method combines the strengths of the Flower Pollination Algorithm with the Particle Swarm Optimization (PSO) algorithm, strategically enhancing the search accuracy for optimal solutions. Their findings demonstrated that this hybrid methodology outperformed existing techniques in terms of precision, reliability, and efficiency. Notably, the hybrid FPA showcased superior performance, making it a promising and advanced solution for tackling optimization problems compared to other methods documented in the literature.

Nigdeli et al. [45] ingeniously combined the Flower Pollination Algorithm (FPA) with the Harmony Search (HS) algorithm to fine-tune mass dampers. In their approach, they employed four distinct generations, incorporating both the global and local search processes of HS and the global and local pollination mechanisms of FPA. The determination of the generation type in constructing new solutions was achieved through a probability-based method. This probability was computed based on the optimization objective, revealing that their innovative probability-based FPA outperformed the classical FPA in terms of convergence rates. This signifies a significant advancement in optimizing mass dampers for structural applications.

In a notable development outlined in reference [24], a pioneering hybridization of Artificial Bee Colony (ABC) with Flower Pollination Algorithm (FPA) gave rise to the Bee Pollinated Flower¨Pollination Algorithm (BPFPA), specifically tailored for solar PV parameter estimation. Within the BPFPA framework, the discarding of pollen, inspired by bee behavior, is seamlessly integrated with FPA, and a mutation operation based on elite in-

dividuals replaces FPA's local pollination process. These strategic modifications not only enriched the randomness of FPA but also endowed the hybrid method with accelerated execution speed and heightened robustness, surpassing the performance of other methods in the domain.

The integration of Differential Evolution (DE) with the Flower Pollination Algorithm (FPA), forming the DE-FPA hybrid as proposed in [7] for addressing benchmark optimization problems, represents a strategic fusion of the distinctive strengths of both algorithms. This fusion is designed to achieve an optimal balance between exploration and exploitation capabilities, leveraging the complementary attributes of DE and FPA. The DE algorithm's role as a potent explorer is retained, strategically combined with FPA's strong exploitation characteristics.

In this hybrid approach, the DE-FPA algorithm harnesses DE's inherent exploration capabilities while capitalizing on FPA's efficiency in exploitation. The resulting synergy not only maintains the exploration prowess of DE but also enhances the exploitation characteristics crucial for navigating optimization landscapes. Experimental results validate the efficacy of DE-FPA, showcasing its superior performance and convergence rates compared to classical DE and FPA.

In their work detailed in [44], Kalra and Arora enhanced the performance of the Flower Pollination Algorithm (FPA) by integrating it with the Firefly Algorithm (FA) to effectively address multimodal optimization functions and mitigate the individual shortcomings of each algorithm. This synergistic approach not only accelerated the convergence speed of FPA but also mitigated the risk of being trapped within local optima by diminishing the impact of randomness inherent in FA.

The hybrid algorithm proposed by Kalra and Arora demonstrates notable improvements in terms of both accuracy and convergence speed when compared to the standalone FA and FPA counterparts. Through experimental validation, it was observed that the integrated approach yielded superior results, highlighting its efficacy in navigating complex optimiz-

ation landscapes. The reduction in the influence of randomness within FA, coupled with the strengths of FPA, contributed to the enhanced performance of the hybrid algorithm, showcasing its potential as a robust solution for addressing multimodal optimization challenges.

In their study outlined in [8], Chakraborty et al. devised an effective integration by combining the global search capabilities of the Flower Pollination Algorithm (FPA) with the local search behavior of the Gravitational Search Algorithm (GSA) for training feedforward neural networks. This integration aimed to strike a balanced approach between exploration and exploitation during the search process. To enhance performance, the authors introduced dynamic switch probabilities and adaptive weights for the GSA velocity operator. These adjustments aimed to prevent getting trapped in local minima and guide the search toward global minima, respectively. The authors assessed their approach using a variety of real-world benchmark datasets obtained from the UCI Machine Learning Repository, encompassing domains such as cancer, glass, iris, vertebral column, and wine. The numerical experiments clearly demonstrated the superior performance of their method across all datasets when compared to standalone FPA and GSA, showcasing its effectiveness in tackling diverse real-world challenges.

Hybridization with Other Components

Zawbaa et al. developed a novel approach for multi-objective feature selection by combining the Flower Pollination Algorithm (FPA) with rough set theory in [18], aiming to pinpoint the most effective features for classification purposes. This innovative strategy melds the best aspects of filter-based and wrapper-based feature selection methods the former focusing on the data itself, and the latter on how well the features improve classification accuracy. The effectiveness of this method was thoroughly evaluated using eight UCI datasets, revealing its impressive competitiveness against traditional algorithms such as the classic FPA, Particle Swarm Optimization (PSO), and Genetic Algorithms (GA).

This advancement underscores the potential of their approach in enhancing feature selection processes.

Abdel-Baset and Hezam ingeniously merged the Flower Pollination Algorithm (FPA) with the Conjugate Direction (CD) method to address the challenge of solving ill-conditioned systems of linear and nonlinear equations [31]. The FPA component was strategically employed to achieve rapid convergence and the capability to discover multiple roots. In contrast, the CD method was integrated to refine the accuracy of the outcomes and prevent the algorithm from being trapped in local minima. The efficacy of this hybrid approach was demonstrated through numerical simulations, which revealed that their method stands out for its competitiveness when juxtaposed with other existing techniques in the literature.

Valenzuela et al. introduced FFPA [30], a novel hybrid method that combines the Flower Pollination Algorithm (FPA) with a fuzzy inference system. Their innovation lies in using the fuzzy inference system to adjust the probability of transitioning between local and global pollination phases. This adaptive feature brings an extra layer of sophistication to their algorithm. Tested on eight benchmark mathematical functions, FFPA showed outstanding performance, proving itself as a strong contender among other advanced approaches in the field.

Xu and Wang presented an enhanced hybrid version of the Flower Pollination Algorithm (FPA) tailored for the precise estimation of solar cell and photovoltaic (PV) module parameters [47]. Their approach integrates FPA with the Nelder-Mead simplex method to bolster the local search capabilities inherent in classical FPA. Additionally, they incorporated a generalized opposition-based learning mechanism to steer clear of local optima pitfalls. The evaluation of their method encompassed three distinct solar models, including the single diode model, double diode model, and a PV module.

Numerical results compellingly illustrated the superiority of the proposed hybrid FPA over alternative methods, particularly in terms of solution accuracy, convergence speed, and overall stability. This novel hybridization not only elevates the FPA's local search efficacy

but also establishes its prowess in accurately estimating parameters critical for solar cell and PV module optimization.

The integration of the Flower Pollination Algorithm (FPA) with a randomized-location modification operator resulted in a novel approach termed the modified randomized-location Flower Pollination Algorithm (MRLFPA) for medical image segmentation, as detailed in [42]. The introduction of the randomized-location strategy in MRLFPA effectively addressed the limitations of the classical FPA. The algorithm's performance was systematically assessed using eight medical images with diverse characteristics. A comprehensive comparative evaluation against other algorithms underscored the efficacy of MRLFPA, showcasing superior solution quality, stability, and computational efficiency.

Multi-Objective Versions of Flower Pollination Algorithm

In a pioneering effort, Yang et al. [52] introduced the first extension of the Flower Pollination Algorithm (FPA) to address multi-objective engineering optimization problems (MOFPA). This innovative adaptation utilized a random weighted sum method to enhance the algorithm's capacity in handling multiple conflicting objectives. The effectiveness of MOFPA was systematically assessed across various engineering optimization problems, demonstrating its capability to yield optimal results.

Subsequently, the same authors advanced their approach [53], introducing a novel technique for MOFPA. This iteration involved the incorporation of diverse multi-objective test functions and two bi-objective design benchmarks. The outcomes of this enhanced algorithm proved highly efficient when compared with alternative algorithms, establishing its prowess in efficiently navigating the complexities of multi-objective optimization challenges. Yang et al.'s contributions mark significant strides in extending the applicability of FPA to the realm of multi-objective engineering optimization.

Shilaja et al. [6] proposed the Enhanced Flower Pollination Algorithm (EFPA) as a solution for the optimal power flow (OPF) problem. EFPA was intricately crafted to optimize

various objectives, encompassing transmission loss, power plant emissions, generation cost minimization, and voltage stability improvement. The team then implemented EFPA and assessed its performance through the standard IEEE 30 test. The findings revealed EFPA's superiority over other optimization algorithms, underscoring its efficacy in tackling the intricate challenges of power system optimization.

Emary et al. [12] implemented a multi-objective Flower Pollination Algorithm (FPA) coupled with pattern search (PS) for retinal vessel localization. The FPA was employed to optimize the clustering within a given retinal image, while PS served as a local search strategy to refine segmentation results. Testing on the DRIVE dataset, a standard benchmark, demonstrated the proposed technique's competitive performance in terms of accuracy, sensitivity, and specificity, along with promising extendable features.

3.4 Conclusion

On this chapter a powerful metaheuristic algorithm was discussed with an explication of its properties. the algorithm has been pass by alot of hybridizations because of his features and capacity on solving optimization problems.

Chapter 4

Chebyshev Metaheuristic Solver Approach

Differential equations present always challenges for researchers, there are huge quantity of problems to be solved. Actually the differential equations are just the problems that human confront on their daily lives. The majority of this problems can't be solved using analytical methods, so researchers go to the method that gives approximate solution to this redundant (or hard to solve) problems.

Regarding to the power of classic numerical methods on solving differential equations, and to the efficiency and rapidity of metaheuristic algorithms, we thought about combining them to have a new approach. On this chapter, we chose spectral method and Flower Pollination Algorithm to create a new method for solving different types of boundary value problems and an integro-differential equation.

The proposed method is constructed by the following manner:

First, use the initial step of spectral method to have an approximate solution formula. Then, using the root mean square to compute the global residue function. Finally, implement Flower Pollination Algorithm to minimize the constructed residual and find the unknowns on the approximate solution.

4.1 Construction of the Chebyshev Metaheuristic Solver Approach

The aim of this new approach is to find good approximate solutions for a wide range of differential equations, independent from their forms, order or linearity.

Beginning by the well-posed differential equation problem:

$$\begin{cases} Lu(x) = f(x), x \in \Omega, \\ C_i(u) = d_i, 1 \leq i \leq m \end{cases} \tag{4.1}$$

Where d_i are initial or boundary conditions. Let $\{x_k\}_{k=0}^{N} \in \Omega$ be the pre-selected points. Assuming that the approximate solution take the following form,

$$u(x) = \sum_{j=0}^{N} u_j T_j(x). \tag{4.2}$$

$\{T_j(x)\}$ are Chebyshev basis polynomials.

The combination of this form of approximate solution and the differential equation written above gives this system of equations,

$$\begin{cases} Lu(x_k) - f(x_k) = 0, \quad 1 \leq k \leq N-1, \\ C_i(u) = d_i, 1 \leq i \leq m \end{cases} \tag{4.3}$$

Replacing $u(x)$ by $u(x) = \sum_{j=0}^{N} u_j T_j(x)$, the previous system of equations becomes,

$$\begin{cases} \sum_{j=0}^{N} [LT_j(x_k)]u_j - f(x_k) = 0, \quad 1 \leq k \leq N-1, \\ C_i(u) = d_i, 1 \leq i \leq m. \end{cases} \tag{4.4}$$

Thus,

n	T_n
0	1
1	x
2	$2x^2 - 1$
3	$4x^3 - 3x$
4	$8x^4 - 8x^2 + 1$
5	$16x^5 - 20x^3 + 5x$
6	$32x^6 - 48x^4 + 18x^2 - 1$
7	$64x^7 - 112x^5 + 56x^3 - 7x$
8	$128x^8 - 256x^6 + 160x^4 - 32x^2 + 1$
9	$256x^9 - 576x^7 + 432x^5 - 120x^3 + 9x$
10	$512x^{10} - 1280x^8 + 1120x^6 - 400x^4 + 50x^2 - 1$
11	$1024x^{11} - 2816x^9 + 2816x^7 - 1232x^5 + 220x^3 - 11x$
12	$2048x^{12} - 6144x^{10} + 9612x^8 - 3584x^6 + 840x^4 - 72x^2 + 1$
13	$4096x^{13} - 13312x^{11} + 16640x^9 - 9984x^7 + 2912x^5 - 364x^3 + 13x$
14	$8192x^{14} - 28672x^{12} + 39424x^{10} - 26880x^8 + 9408x^6 - 1568x^4 + 98x^2 - 1$
15	$16384x^{15} - 61440x^{13} + 92160x^{11} - 70400x^9 + 28800x^7 - 6048x^5 + 560x^3 - 15x$

Table 4.1: Chebyshev Polynomials, First Kind 40

$$\begin{cases} \sum_{j=0}^{N} [LT_j(x_k)]u_j - f(x_k) = 0, \quad 1 \leq k \leq N-1, \\ C_i(u) - d_i = 0, 1 \leq i \leq m \end{cases} \tag{4.5}$$

So,

$$\begin{cases} \sum_{j=0}^{N} [LT_j(x_1)]u_j - f(x_1) = 0, \\ \sum_{j=0}^{N} [LT_j(x_2)]u_j - f(x_2) = 0, \\ . \\ . \\ . \\ \sum_{j=0}^{N} [LT_j(x_{N-1})]u_j - f(x_{N-1}) = 0, \\ C_i(u) - d_i = 0, 1 \leq i \leq m. \end{cases} \tag{4.6}$$

In spectral method this system must be solved using some numerical tools, to find the coefficients u_j in the approximate solution 4.2.

In Chebyshev metaheuristic solver approach, this last step (solving the equations system) will be eliminated, instead of that, Flower Pollination Algorithm will be used to find the unknowns coefficients u_j.

The aim of the implementation of Flower Pollination Algorithm is minimizing the residual R constructed by calculating the mean square error obtained from the first side of every equation defined in the system 4.6.

Thus,

$$R = \sum_{k=1}^{N-1} \left[\sum_{j=0}^{N} \left[LT_j(x_k) \right] u_j - f(x_k) \right]^2 + \sum_{i=0}^{m} w_i ConditionPenalty_i. \tag{4.7}$$

Where the $ConditionPenalty_i = (C_i(u) - d_i)^2, 1 \leq i \leq m$.

The minimization of the residual R offers the coefficients u_j for an optimum solution $u(x)$ written in the form 4.2.

The construction of spectral metaheuristic algorithm can be summarized on these main steps:

- Transfer the differential equation into a system of algebraic equations using Chebyshev polynomials.
- Transfer the boundary or initial conditions to algebraic equations utilizing the same mechanism that has been carried out for the differential equation.
- Compute the residual of every equation by subtracting the both sides in each one.
- Construct the global residual by making the sum of all square residual gotten from the previous equations.
- Implement a metaheuristic algorithm to minimize the constructed residual.
- Use the unknowns found in the last step for building an approximate solution.

Remark 4.1.1 *In the results shown bellow, $ConditionPenalty_i$ are treated as small residuals where $w_i = 1$.*

For dfifferent boundaries then [-1,1], Chebyshev polynomials must be transformed using the mapping $T_n^*(x) = T_n(\frac{2x-(b+a)}{b-a})$.

4.2 Parameters of Flower Pollination Algorithm

The step of implementing Flower Pollination Algorithm, aiming to minimize the value of R_{glob}, controlled by some parameters, where the outcome solutions vary depending on their values modifications, it means that if we change one of these parameters we get different results.

The way to choose the best solution is to observe the path of $f_{\min}$ offered by the algorithm, after each execution.

Where on a particular execution the $f_{\min}$ will be constant and doesn't want to reduce more then that, here we get the best solution and the best parameters for the operation.

The parameters of Flower Pollination Algorithm can be summarized in:

- The bounds Ub and Lb, where Lb presents the lower bound of the solution's space, and Ub presents the upper bound of the solution's space.

  The intensification and the diversification on the solution's search process is controlled by the selection of these bounds.

  If the bounds are nearer the examination of the solution will be more intense, in contrast the examination of the solution is expected to be more diverse.

- The population size n, n controls the intensification of the searching operation, where it's by default from 10 to 25, this number are proposed by the creator of this algorithm Xin-She Yang.

- The probability switch p, it controls the transfers between global (diversification) and local pollination (intensification).

- In wide range of the algorithm's application, the p is chosen between 0.5 and 0.8 toward the local pollination, for example in the case of $p = 0.8$, it means that for 80% the local pollination is selected. Mostly, $p = 0.8$ gives more efficient results. [56]

- The number of iterations N_{iter}.

  In every iteration, each individual from the population is a suggested solution. Therefore, in an execution of an algorithm with 25 population size and 10000 iterations, there are 10000×25 selected solutions, whose been tested. So the total number of evaluations is 250000.

- d is a parameter that depends only on the dimension of the solution. [23]

4.3 Pseudocode of Chebyshev Meatheuristic Solver Approach

1. Define Solution Structure:

   Assume the solution is a Chebyshev series: $y(x) = \sum a_i * T_i(x)$.

2. Define a Fitness Function $fitness(a)$:

   This function takes coefficients a and returns a total error, combining:

   - The error from not satisfying the differential equation.
   - The error from not meeting the boundary conditions.

3. Find Best Coefficients:

Use the Flower Pollination Algorithm to find the coefficient vector a_best that minimizes $fitness(a)$.

4. Final Answer:

   The approximate solution is the Chebyshev series constructed with a_best.

END

4.4 Results

On this part we will apply the proposed Chebyshev metaheuristic solver approach to solve some boundary value problems and an integro-differential equation, the application will be done in different types for boundary value problems beginning by linear problems, then a nonlinear problem, after that an integro-differential equation will be solved using the new approach.

4.4.1 Linear Boundary Value Problems

Here we will treat two different type of linear boundary value problems, first an homogeneous problem, then a non-homogeneous one.

Homogeneous Problems

On this part we will solve an homogeneous linear boundary value problem just to see how works the new approach.
Considering the following problem

$$\begin{cases} u'' - u = 0, \text{ if } x \in [0,1] \\ u(0) = 2, \\ u(1) = \exp(1) + \exp(-1) \end{cases} \tag{4.8}$$

Analytical solution We will start by getting the exact solution of this simple linear homogeneous boundary value problem. After that, we calculate the approximate solutions using spectral method, then Chebyshev metaheuristic solver approach. Finally, we compare the exact solution with their approximations.

Let's find the exact solution.

We have,

$$u'' - u = 0 \Leftrightarrow u'' = u$$

Supposing that $u(x) = \exp(rx)$, it means that $u' = r\exp(rx)$ and $u'' = r^2\exp(rx)$.

Thus, our differential equation becomes

$r^2\exp(rx) - \exp(rx) = 0$

$\exp(rx)(r^2 - 1) = 0$

$r^2 - 1 = 0$

Here the differential equation has been reduced to a characteristic equation.

After factorizing the first side of the equation to $(r-1)(r+1)$, we get $r = 1$ or $r = -1$.

So we have two special solutions,

$u(x) = \exp(x)$, and $u(x) = \exp(-x)$.

To get more general solution we combine different multiples of those two solutions, and the general solution becomes

$$u(x) = A\exp(x) + B\exp(-x).$$

Let's return to our problem, after subtituting by the boundary conditions, the constants A and B will be found.

$u(0) = 2 \Longleftrightarrow A\exp(0) + B\exp(0) = 2,$

so,

$$A + B = 2 \tag{4.9}$$

$$u(1) = \exp(1) + \exp(-1) \Longleftrightarrow A\exp(1) + B\exp(-1) = \exp(1) + \exp(-1) \tag{4.10}$$

Concluding from 4.9 and 4.10 the coefficients are $A = 1$ and $B = 1$.

Here we find the exact solution of this problem, which is

$$u(x) = \exp(x) + \exp(-x).$$

Chebyshev metaheuristic solver approach: Let's Solve the homogeneous linear problem using the new approach.

The principal idea of this novel method is to minimize the error and get an approximate solution, so on the initial step is, calculate the error.

First, we choose the order of Chebyshev polynomial to have a solution's formula. After, we calculate the residual. Then, we implement Flower Pollination Algorithm to minimize the residual.

The interval of the problem is not $[-1, 1]$, therefore we will use Chebyshev polynomials after mapping in $[0, 1]$,

N	Chebyshev ploynomials in $[0, 1]$
0	$T_0^*(x) = T(2x-1) = 1$,
1	$T_1^*(x) = 2x - 1$,
2	$T_2^*(x) = 8x^2 - 8x + 1$,
3	$T_3^*(x) = 32x^3 - 48x^2 + 18x - 1$,
4	$T_4^*(x) = 128x^4 - 256x^3 + 160x^2 - 32x + 1$,
5	$T_5^*(x) = 512x^5 - 1280x^4 + 1120x^3 - 400x^2 + 50x - 1$,
6	$T_6^*(x) = 2048x^6 - 6144x^5 + 6912x^4 - 3584x^3 + 840x^2 - 72x + 1$,
7	$T_7^*(x) = 8192x^7 - 28672x^6 + 3942x^5 - 26880x^4 + 9408x^3 - 1567x^2 + 98x - 1$,
8	$T_8^*(x) = 32768x^8 - 131072x^7 + 212992x^6 - 180224x^5 + 84480x^4 - 21504x^3 + 2816x^2 - 128x + 1$,
9	$T_9^*(x) = 131072x^9 - 589824x^8 + 1118208x^7 - 1146880x^6 + 658944x^5 - 215040x^4 + 42240x^3$ $-4352x^2 + 162x - 1$.

Table 4.2: Chebyshev polynomials in [0,1]

Supposing that $u(x) = \sum_{j=0}^{N} u_j T_j^*(x)$ with $N = 5$, $N = 7$ and $N = 9$.

N=5 First, we choose the pre-selected points, $N = 5$, it means that we have to choose from $N - 1$ pre-selected points.

$$\begin{cases} x_1 = 0, \\ x_2 = -1/3, \\ x_3 = 1/3, \\ x_4 = 1. \end{cases}$$

There the equation's system becomes,

$$\begin{cases} u(0) = 2, \\ u''(0) - u(0) = 0, \\ u''(1/3) - u(1/3) = 0, \\ u''(2/3) - u(2/3) = 0, \\ u''(1) - u(1) = 0, \\ u(1) = \exp(1) + \exp(-1). \end{cases}$$

The global residual is:

$$R_{glob} = R_0^2 + R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2. \tag{4.11}$$

There are 6 small residuals, because here on our example we chose to use Chebyshev polynomials with 5^{th} order. So we have 6 unknowns, it means that we have 6 equations, and instead of solving them, we use FPA.

R_0 represents the error given from the first equation, R_1 which is the error given from the second equation, ...

$$\begin{cases} R_0 = u(0) - 2, \\ R_1 = u''(0) - u(0), \\ R_2 = u''(1/3) - u(1/3), \\ R_3 = u''(2/3) - u(2/3), \\ R_4 = u''(1) - u(1), \\ R_5 = u(1) - \exp(1) + \exp(-1). \end{cases}$$

Therefore,

$$\begin{aligned} R_{glob} &= [u(0)-2]^2 + [u''(0)-u(0)]^2 \\ &\quad +[u''(1/3)-u(1/3)]^2 \\ &\quad +[u''(2/3)-u(2/3)]^2 \\ &\quad +[u''(1)-u(1)]^2 + [u(1)-\exp(1)+\exp(-1)]^2. \end{aligned} \tag{4.12}$$

The next step, is implementing the algorithm to minimize the value of R_{glob} subsequent to many changes of the algorithm's parameters we found our best approximate solution when:

$Ub = 3$;

$Lb = -2$;

$n = 20$;

$p = 0.8$;

$N_{iter} = 10000$;

$d = 6$ because we have 6 unknowns a, b, c, d, e, f to be found, so the dimension of the search space is 6,

$w = 1$, just for simplification.

The evaluations total number is considered as 200000, and the $f_{\min}$ obtained is 1.864280×10^{-17}.

The best solutions given by Flower Pollination Algorithm are:

$$\begin{cases} a = 2.3984, \\ b = 0.5376, \\ c = 0.1439, \\ d = 0.0055, \\ e = 0.0007, \\ f = 0.0000. \end{cases}$$

Therefore,

$$u_{FPA}(x) = 2.3984T_0^*(x)+0.5376T_1^*(x)+0.1439T_2^*(x)+0.0055T_3^*(x)+0.0007T_4^*(x)+0.0000T_5^*(x). \tag{4.13}$$

We conclude that the approximate solution given by the new method is

$$\begin{aligned} u_{FPA}(x) &= 1.999999990674 - 0.000040337537x + 0.999576672282x^2 + 0.003987886841x^3 \\ &\quad +0.073817304084x^4 + 0.008819743968x^5. \end{aligned} \tag{4.14}$$

Here we present the graphs produced by the exact and the approximate solutions produced by the new approach in the case of $N = 5$, for our homogeneous linear boundary value problem 4.8.

u_{FPA} presents the approximate solution obtained from Chebyshev metaheuristic solver approach.

u_{exact} presents the exact solution.

The figure 4.1 displays the graphs generated by the exact solution, the Chebyshev metaheuristic solver approach's solution.

- It's obvious that Chebyshev metaheuristic solver approach's approximation graph is identical to the curve produced by the exact solution.

For N=7: Using Chebyshev polynomials for $N = 7$ and the residual,

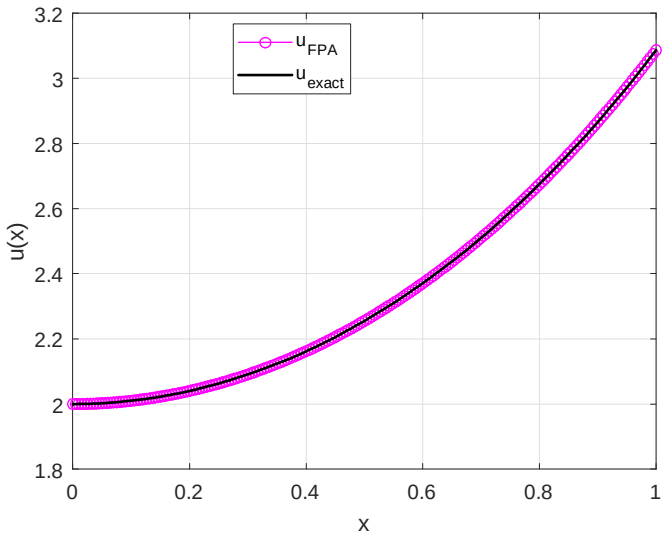

Figure 4.1: Fig 4.1 Exact Solution vs. Approximated Results: The first example N=5

$$R_{glob} = R_0^2 + R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2 + R_6^2 + R_7^2. \tag{4.15}$$

Parameters of Flower Pollination Algorithm utilized for problem are,

$Ub = 3;$

$Lb = -3;$

$n = 25;$

$p = 0.5;$

$N_{iter} = 10000;$

$d = 8,$

$w = 1.$

The coefficients got from FPA are,

$$\begin{cases} a = 2.3984, \\ b = 0.5376, \\ c = 0.1439, \\ d = 0.0055, \\ e = 0.0007, \\ f = 0.0000, \\ g = 0.0000, \\ h = 0.0000. \end{cases}$$

From this results we conclude the value of $u(x)$,

$$\begin{aligned} u_{FPA} &= 2.3984T_0^*(x) + 0.5376T_1^*(x) + 0.1439T_2^*(x) + 0.0055T_3^*(x) + 0.0007T_4^*(x) + 0.0000T_5^*(x) \\ &\quad +0.0000T_6^*(x) + 0.0000T_7^*(x). \end{aligned} \tag{4.16}$$

Therefore,

$$\begin{aligned} u_{FPA} &= 2.000000000001 - 0.000000079607x + 0.999999510364x^2 + 0.000017908176x^3 \\ &\quad +0.083227911527x^4 + 0.000266886874x^5 + 0.002440202636x^6 + 0.000208929657x^7. \end{aligned} \tag{4.17}$$

The figure 4.2 reveals that ,the exact solution and the approximate solution resulted from the new method for $N = 7$, are similar and we can't distinguish the difference between them in the graph.

For N=9 Using Chebyshev polynomials for $N = 9$ and the residual,

$$R_{glob} = R_0^2 + R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2 + R_6^2 + R_7^2 + R_8^2 + R_9^2. \tag{4.18}$$

Parameters of Flower Pollination Algorithm utilized for problem are,

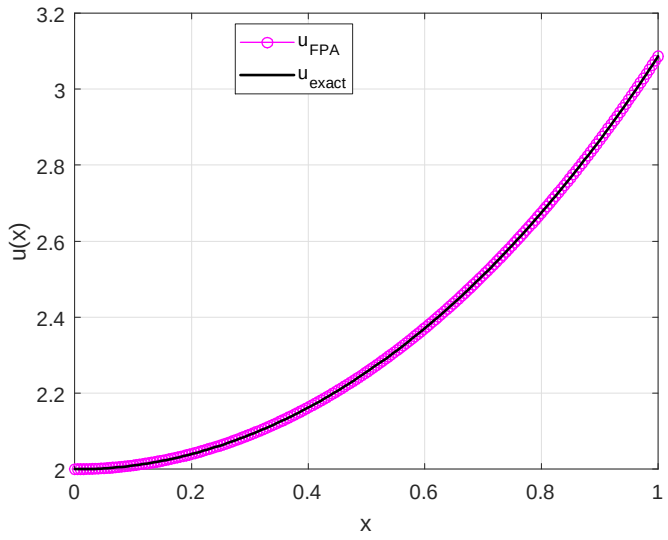

Figure 4.2: Fig 4.2 Exact Solution vs. Approximated Results: The first example N=7

$Ub = 3;$

$Lb = 3;$

$n = 25;$

$p = 0.5;$

$N_{iter} = 10000;$

$d = 10,$

$w = 1.$

The coefficients got from FPA are,

$$\begin{cases} a = 2.3984, \\ b = 0.5376, \\ c = 0.1439, \\ d = 0.0055, \\ e = 0.0007, \\ f = 0.0000, \\ g = 0.0000, \\ h = 0.0000, \\ i = 0.0000, \\ j = 0.0000. \end{cases}$$

From this results we conclude the value of $u(x)$,

$$\begin{aligned} u_{FPA} &= 2.3984T_0^*(x) + 0.5376T_1^*(x) + 0.1439T_2^*(x) + 0.0055T_3^*(x) + 0.0007T_4^*(x) + 0.0000T_5^*(x) + 0.0 \\ &\quad +0.0000T_7^*(x) + 0.0000T_8^*(x) + 0.0000T_9^*(x). \end{aligned}$$

Therefore,

$$\begin{aligned} u_{FPA} &= 2.000000000000 - 0.000000000097x + 0.999999999821x^2 + 0.000000036043x^3 \\ &\quad +0.083332943801x^4 + 0.000001843136x^5 + 0.002773030964x^6 \\ &\quad +0.000007118339x^7 + 0.000043403937x^8 + 0.000002893686x^9. \end{aligned} \tag{4.32}$$

The figure 4.3 confirms that the resulted solution for $N = 9$ is approximately the same as the exact one.

The table 4.5 illustrates the Root Mean Square Error and the Mean Square Error given from the approximate solution of the linear homogeneous problem 4.8, utilizing the Chebyshev metaheuristic solver approach. The RMSE and the MSE given by the CMSA, for

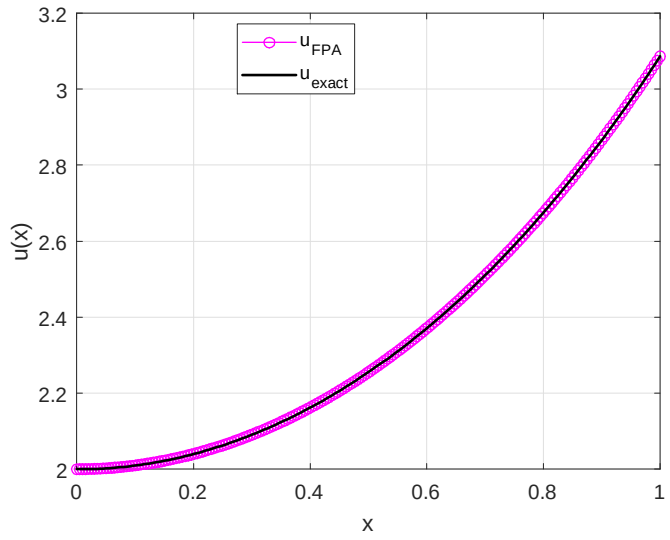

Figure 4.3: Fig 4.3 Exact Solution vs. Approximated Results: The first example N=9

Optimizer	RMSE	MSE
CMSA $N = 5$	5.509954×10^{-06}	3.035960×10^{-11}
CMSA $N = 7$	5.137463×10^{-09}	2.639352×10^{-17}
CMSA $N = 9$	2.900512×10^{-12}	8.412970×10^{-24}

Table 4.3: Comparison table of RMSE for the linear homogeneous differential problem

$N = 5$ and $N = 7$, and $N = 9$ are nearly negligible, specially for $N = 9$. As the error decreases the N increases.

Non-homogeneous Problems

Let's use Chebyshev metaheuristic solver approach to find an approximate solution for a non-homogeneous boundary value problem.

Supposing the non-homogeneous problem defined by:

$$\begin{cases} u'' - 2u' + u = x, \\ u(0) = 0, \\ u(2) = 4. \end{cases} \tag{4.21}$$

Analytical solution To solve a non-homogeneous second order differential equation we will first find the complementary function, where it is the homogeneous equation's solution. Our homogeneous equation is

$$u'' - 2u' + u = 0 \tag{4.22}$$

Supposing that

$$\begin{aligned} u &= \exp(rx), \\ u'(x) &= r\exp(rx), \\ u''(x) &= r^2\exp(rx). \end{aligned}$$

The equation 4.22 becomes,

$$\begin{aligned} &r^2\exp(rx) - r\exp(rx) + \exp(rx) = 0, \\ &\exp(rx)[r^2 - 2r + 1] = 0, \\ &r^2 - 2r + 1 = 0. \end{aligned}$$

We find two real equal roots to this characteristic equation $r_1 = r_2 = 1$, it means that the solution of the equation 4.22 is

$$u_c = A\exp(x) + Bx\exp(x). \tag{4.23}$$

Now , let's find the particular solution for the differential equation defined in the problem 4.21 Assuming that

$u_p = \alpha x^2 + \beta x + \gamma$,

e. i $u_p' = 2\alpha x + \beta$,

and $u_p'' = 2\alpha$.

Substitute in the equation from 4.21,

$u'' - 2u' + u = x$,

$2\alpha - 2(2\alpha x + \beta) + (\alpha x^2 + \beta x + \gamma) = x,$

$\alpha x^2 + (-4\alpha + \beta)x - 2\beta + \gamma + 2\alpha = x,$

$$\Leftrightarrow \begin{cases} \alpha = 0, \\ -4\alpha + \beta = 1, \\ -2\beta + \gamma + 2\alpha = 0. \end{cases}$$

$$\Leftrightarrow \begin{cases} \alpha = 0, \\ \beta = 1, \\ \gamma = 2. \end{cases}$$

thus,

$$y_p = x + 2.$$

The general solution for the problem 4.21, is presented by

$$u = u_p + u_c = A\exp(x) + Bx\exp(x) + x + 2. \tag{4.24}$$

The final step to get the analytical solution is finding the constants A and B, using the bounday conditions,

$u(0) = 0$, gives $A = -2$,

$u(2) = 4$, affords $B = 1$.

Therefore,

$$u(x) = \exp(x)(x - 2) + x + 2. \tag{4.25}$$

Chebyshev metaheuristic solver approach Begining by choosing the order of Chebyshev polynomials and calculate the residual,

Supposing that $u(x) = \sum_{j=0}^{N} u_j T_j^*(x)$ with $N = 5$, $N = 7$ and $N = 9$.

N=5 First, we choose the pre-selected points, $N = 5$, it means that we have to choose from $N - 1$ pre-selected points, and calculate the global residual:

N	Chebyshev ploynomials in $[0, 2]$
0	$T_0^*(x) = T(x-1) = 1,$
1	$T_1^*(x) = x - 1,$
2	$T_2^*(x) = 2x^2 - 4x + 1,$
3	$T_3^*(x) = 4x^3 - 12x^2 + 9x - 1,$
4	$T_4^*(x) = 8x^4 - 32x^3 + 48x^2 - 24x + 1,$
5	$T_5^*(x) = 16x^5 - 80x^4 + 160x^3 - 160x^2 + 70x - 5.$
6	$T_6^*(x) = 32x^6 - 192x^5 + 480x^4 - 640x^3 + 440x^2 - 144x + 1,$
7	$T_7^*(x) = 128x^7 - 448x^6 + 1344x^5 - 2240x^4 + 2240x^3 - 1344x^2 + 392x - 7,$
8	$T_8^*(x) = 128x^8 - 1024x^7 + 3584x^6 - 7168x^5 + 8960x^4 - 7168x^3 + 3584x^2 - 896x + 1,$
9	$T_9^*(x) = 256x^9 - 2304x^8 + 9216x^7 - 21504x^6 + 32256x^5 - 32256x^4 + 21504x^3$ $-9216x^2 + 2040x - 9.$

Table 4.4: Chebyshev polynomials in [0,2]

$$R_{glob} = R_0^2 + R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2. \tag{4.26}$$

There are 6 small residuals, because here on our example we chose to use Chebyshev polynomials with 5^{th} order. So we have 6 unknowns, it means that we have 6 equations, and instead of solving them, we use FPA.

The next step, is implementing the algorithm to minimize the value of R_{glob} subsequent to many changes of the algorithm's parameters we found our best approximate solution when:

$Ub = 3;$

$Lb = -2;$

$n = 20;$

$p = 0.8;$

$N_{iter} = 10000;$

$d = 6$

$w = 1.$

The best solutions given by Flower Pollination Algorithm are:

$$\begin{cases} a = 1.0949, \\ b = 1.7379, \\ c = 0.8571, \\ d = 0.2557, \\ e = 0.0482, \\ f = 0.0062. \end{cases}$$

Therefore,

$$u_{FPA}(x) = 1.0949T_0^*(x)+1.7379T_1^*(x)+0.8571T_2^*(x)+0.2557T_3^*(x)+0.0482T_4^*(x)+0.0062T_5^*(x). \tag{4.27}$$

We conclude that the approximate solution given by the new method is

$$\begin{aligned} u_{FPA}(x) &= 0.000321039110 - 0.005009143962x - 0.047424609829x^2 \\ &\quad +0.349597726960x^3 - 0.110792373530x^4 + 0.099225981229x^5. \end{aligned} \tag{4.28}$$

Here we present the graphs produced by the exact and the approximate solutions produced by the new approach in the case of $N = 5$, for our homogeneous linear boundary value problem 4.21.

u_{FPA} presents the approximate solution obtained from Chebyshev metaheuristic solver approach.

u_{exact} presents the exact solution.

The figure 4.4 displays the graphs generated by the exact solution, the Chebyshev metaheuristic solver approach's solution for $N = 5$.

- It's obvious that Chebyshev metaheuristic solver approach's approximation graph is identical to the curve produced by the exact solution.

For N=7: Using Chebyshev polynomials for $N = 7$ and the residual,

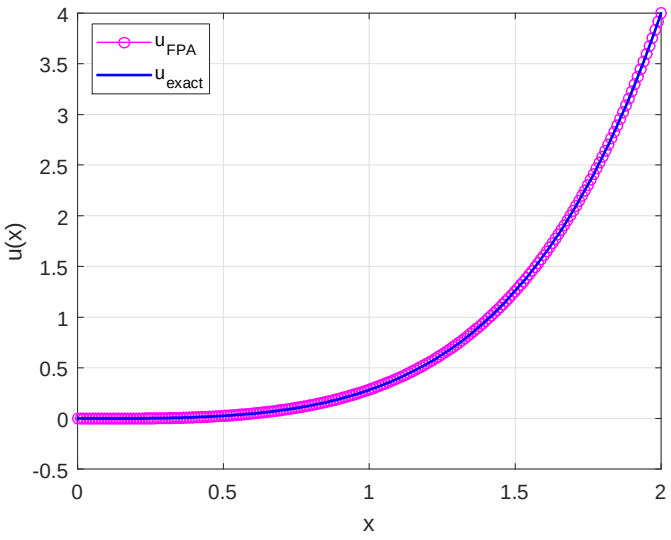

Figure 4.4: Fig 4.4 Exact Solution vs. Approximated Results: The second example N=5

$$R_{glob} = R_0^2 + R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2 + R_6^2 + R_7^2. \tag{4.29}$$

Parameters of Flower Pollination Algorithm utilized for problem are,

$Ub = 3;$

$Lb = -3;$

$n = 25;$

$p = 0.5;$

$N_{iter} = 10000;$

$d = 8,$

$w = 1.$

The coefficients got from FPA are,

$$\begin{cases} a = 1.0947, \\ b = 1.7380, \\ c = 0.8585, \\ d = 0.2559, \\ e = 0.0461, \\ f = 0.0060, \\ g = 0.0006, \\ h = 0.0001. \end{cases}$$

From this results we conclude the value of $u(x)$,

$$\begin{aligned} u_{FPA} &= 1.0947T_0^*(x) + 1.7380T_1^*(x) + 0.8585T_2^*(x) + 0.2559T_3^*(x) + 0.0461T_4^*(x) + 0.0060T_5^*(x) \\ &\quad +0.0006T_6^*(x) + 0.0001T_7^*(x). \end{aligned} \tag{4.30}$$

Therefore,

$$\begin{aligned} u_{FPA} &= 0.000000042834 - 0.000065651534x - 0.000412100113x^2 + 0.171662398718x^3 + 0.0697654192 \\ &\quad +0.041090852589x^5 - 0.003765513897x^6 + 0.003424370017x^7. \end{aligned}$$

In the figure 4.5 the etwo graphs of exact solution and approximate solution via Chebyshev metaheuristic solver approach's solution for $N = 7$ are somewhat identical.

For N=9 Using Chebyshev polynomials for $N = 9$ and the residual,

$$R_{glob} = R_0^2 + R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2 + R_6^2 + R_7^2 + R_8^2 + R_9^2. \tag{4.32}$$

Parameters of Flower Pollination Algorithm utilized for problem are,

$Ub = 3$;

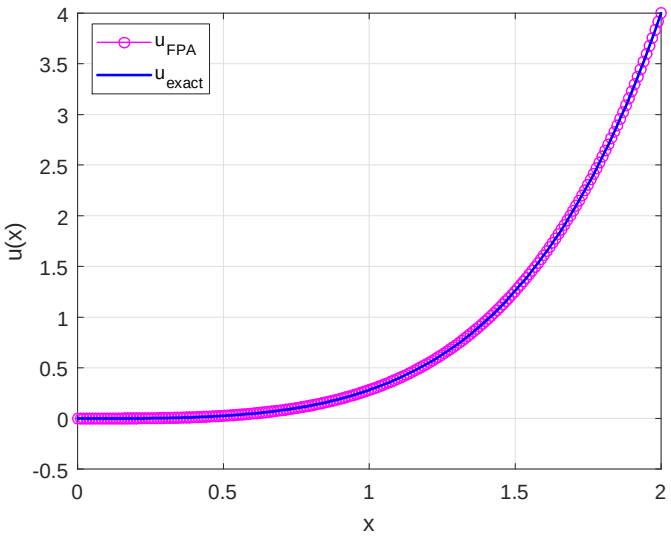

Figure 4.5: Fig 4.5 Exact Solution vs. Approximated Results: The second example N=7

$Lb = 3;$

$n = 25;$

$p = 0.5;$

$N_{iter} = 10000;$

$d = 10,$

$w = 1.$

The coefficients got from FPA are,

$$\begin{cases} a = 1.0947, \\ b = 1.7380, \\ c = 0.8585, \\ d = 0.2559, \\ e = 0.0461, \\ f = 0.0060, \\ g = 0.0006, \\ h = 0.0001, \\ i = 0.0000, \\ j = 0.0000. \end{cases}$$

From this results we conclude the value of $u(x)$,

$$\begin{aligned} u_{FPA} &= 1.0947T_0^*(x) + 1.7380T_1^*(x) + 0.8585T_2^*(x) + 0.2559T_3^*(x) + 0.0461T_4^*(x) + 0.0060T_5^*(x) \\ &\quad +0.0006T_6^*(x) + 0.0001T_7^*(x) + 0.0000T_8^*(x) + 0.0000T_9^*(x). \end{aligned} \tag{4.33}$$

Therefore,

$$\begin{aligned} u_{FPA} &= 0.000000000003 - 0.000000373246x - 0.000001360784x^2 + 0.166717474423x^3 \\ &\quad +0.083071329901x^4 + 0.025602511673x^5 + 0.004804667279x^6 \\ &\quad +0.001529402208x^7 - 0.000067807552x^8 + 0.000062385326x^9. \end{aligned} \tag{4.34}$$

In the figure 4.6 the etwo graphs of exact solution and approximate solution via Chebyshev metaheuristic solver approach's solution for $N = 9$ are roughly the same.

The table 4.5 shows the Root Mean Square Error and the Mean Square Error obtained from the approximate solution of the linear diffrential problem introduced in 4.21, using the Chebyshev metaheuristic solver approach. The RMSE and the MSE given by the

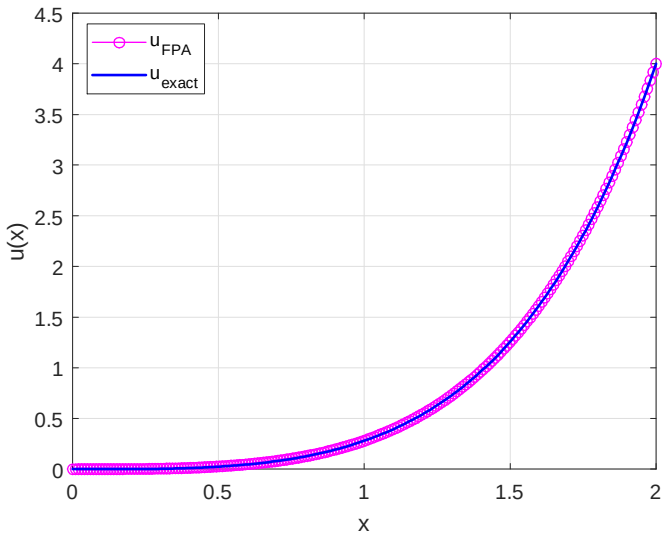

Figure 4.6: Fig 4.6 Exact Solution vs. Approximated Results: The second example N=9

Optimizer	RMSE	MSE
CMSA $N = 5$	2.176850×10^{-03}	4.738677×10^{-06}
CMSA $N = 7$	1.089595×10^{-05}	1.187218×10^{-10}
CMSA $N = 9$	3.378533×10^{-08}	1.141448×10^{-15}

Table 4.5: Comparison table of RMSE for the linear non-homogeneous differential problem

CMSA, for $N = 5$ and $N = 7$, and $N = 9$ are nearly negligible, specially for $N = 9$. As the error decreases the N increases.

4.4.2 Non-Linear Boundary Value Problems:

On this section we will use the Chebyshev metaheuristic solver approach to give an approximate solution to a non-linear boundary value problem.

Bernoulli equation

Assuming the non-linear boundary value problem (Bernoulli equation): [40]

$$u'' + (u')^2 - 2\exp(-u) = 0, \tag{4.35}$$

accompanied with the endpoints

$$u(0) = 0, u(1) = 0. \tag{4.36}$$

Analytical solution: beginning by transforming the non-linear equation to a Bernoulli equation, where we use the method of substituting where:

$$v = \frac{du}{dx} \text{ and } \frac{d^2u}{dt^2} = \frac{dv}{dt} = \frac{dv}{du} \times \frac{du}{dx} = \frac{dv}{dx}.$$

Hence,

$$\begin{aligned} \frac{d^2u}{dt^2} + (\frac{du}{dx})^2 &= 2\exp(-u), \\ \text{i.e. } \frac{dv}{du}v + v^2 &= 2\exp(-u), \end{aligned}$$

putting $\exp(-u) = z$ [5] yields to

$$\frac{dv}{du} = \frac{dv}{dz}\frac{dz}{du} = -\frac{dv}{dz} \times \exp(-u) = -z\frac{dv}{dz}.$$

Thus,

$$\begin{aligned} -z\frac{dv}{dz} + v^2 &= -2z, \\ vz\frac{dv}{dz} - v^2 &= -2z. \end{aligned}$$

Set $\mu = v^2$ imply $\frac{d\mu}{dz} = 2v\frac{dv}{dz}$.

The equation becomes,

$$\begin{aligned} \frac{z}{2} \times \frac{d\mu}{dz} - \mu &= -2z, \\ \frac{d\mu}{dz} - 2\frac{\mu}{z} &= -4. \end{aligned}$$

This final result is a Bernoulli equation.

Let's return to the first equation where we could solve it just by the substituting method.

The equation is,

$$\frac{d^2u}{dx^2} + (\frac{du}{dx})^2 - 2\exp(-u) = 0.$$

By multiplying with $\exp(u)$, we've got

$$\exp(u) \times \frac{d^2u}{dx^2} + \exp(u) \times (\frac{du}{dx})^2 = 2.$$

Assuming that $t = \exp(u)$, so $\frac{dt}{dx} = \frac{du}{dx}\exp(u)$ and $\frac{d^2t}{dx^2} = \frac{d^2u}{dx^2}\exp(u) + \exp(u)\frac{du}{dx}$.

The equation becomes,

$$\frac{d^2t}{dx^2} = 2,$$

imply that $\frac{dt}{dx} = 2x + A$, it means that $t = x^2 + Ax + B$.

Therefore,

$\exp(u) = x^2 + Ax + B$.

So, $u = \ln|x^2 + Ax + B|$.

Boundary conditions give

$$\begin{cases} y(0) = 0 \Rightarrow B = 1, \\ y(1) = 0 \Rightarrow A = -1. \end{cases}$$

Here the final solution of our problem

$$u = \ln|x^2 - x + 1|. \tag{4.37}$$

Chebyshev metaheuristic solver approach solution: This problem has been solved previoulsy by Babaei in [32], using Particle Swarm Optimization, for that reason we will solve it using the new approach by upgrading the degree of Chebyshev polynomials until getting better results.

The Chebyshev used are

N	Chebyshev ploynomials
0	$T_0^*(x) = T(2x-1) = 1,$
1	$T_1^*(x) = 2x - 1,$
2	$T_2^*(x) = 8x^2 - 8x + 1,$
3	$T_3^*(x) = 32x^3 - 48x^2 + 18x - 1,$
4	$T_4^*(x) = 128x^4 - 256x^3 + 160x^2 - 32x + 1,$
5	$T_5^*(x) = 512x^5 - 1280x^4 + 1120x^3 - 400x^2 + 50x - 1.$
6	$T_6^*(x) = 32x^6 - 192x^5 + 480x^4 - 640x^3 + 440x^2 - 144x + 1$
7	$T_7^*(x) = 64x^7 - 448x^6 + 1344x^5 - 2240x^4 + 2240x^3 - 1344x^2 + 392x - 7$
8	$T_8^*(x) = 128x^8 - 1024x^7 + 3584x^6 - 7168x^5 + 8960x^4 - 7168x^3 + 3584x^2 - 896x + 1$
9	$T_9^*(x) = 256x^9 - 2304x^8 + 9216x^7 - 2150x^6 + 32256x^5 - 32256x^4 + 2150x^3$ $-9216x^2 + 2040x - 9$
10	$T_{10}^*(x) = 512x^{10} - 5120x^9 + 23040x^8 - 61440x^7 + 107520x^6 - 120960x^5 + 80640x^4$ $-33600x^3 + 7560x^2 - 880x + 1$

Table 4.6: Chebyshev polynomials in [0,1]

For N=5: Calculating the residual using Chebyshev polynomials same as previous examples.

$$R_{glob} = R_0^2 + R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2. \tag{4.38}$$

Parameters of Flower Pollination Algorithm utilized for problem are,

$Ub = 2;$

$Lb = -2;$

$n = 25;$

$p = 0.5;$

$N_{iter} = 10000;$

$d = 6,$

$w = 1.$

Here the results,

$$\begin{cases} a = 0.03728, \\ b = -0.20964, \\ c = -0.030246, \\ d = 0.27013, \\ e = -0.067533, \\ f = 1.6928 \times 10^{\{-17\}}. \end{cases}$$

From this results we conclude the value of $u(x)$,

$$\begin{aligned} u_{FPA} &= 0.03728T_0(x) - 0.20964T_1(x) - 0.030246T_2(x) + 0.27013T_3(x) - 0.067533T_4(x) \\ &\quad +1.6928 \times 10^{\{-17\}}T_5(x). \end{aligned} \tag{4.39}$$

Therefore,

$$u_{FPA} = -7\times10^{\{-6\}} - 1.02x + 0.4798x^2 + 1.08052x^3 - 0.5403x^4 + 2.7069\times10^{\{-16\}}x^5. \tag{4.40}$$

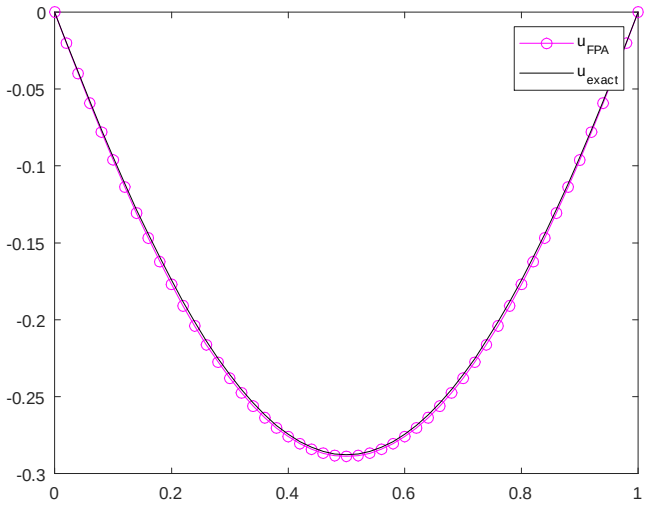

Figure 4.7: Fig 4.7 Exact Solution vs. Approximate Results: Bernoulli Problem N=5

The figure 4.7 shows two curves representing solutions to the non-linear problem presented on this example. The black line denotes the exact solution, while the pink one shows the approximations obtained by Chebyshev metaheuristic solver approach for $N = 5$. The approximations are close to the exact solution, demonstrating great accuracy.

For N=7: Using Chebyshev polynomials for $N = 7$ and the residual,

$$R_{glob} = R_0^2 + R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2 + R_6^2 + R_7^2. \tag{4.41}$$

Parameters of Flower Pollination Algorithm utilized for problem are,

$Ub = 2$;

$Lb = -2$;

$n = 25$;

$p = 0.5$;

$N_{iter} = 10000$;

$d = 8$,

$w = 1$.

The coefficients got from FPA are,

$$\begin{cases} a = -0.00082011, \\ b = -0.13476, \\ c = -0.096989, \\ d = 0.32353, \\ e = -0.10515, \\ f = 0.020998, \\ g = -0.0089923, \\ h = 0.002038. \end{cases}$$

From this results we conclude the value of $u(x)$,

$$u_{FPA} = -0.00082011T_0^*(x) - 0.13476T_1^*(x) - 0.096989T_2^*(x) + 0.32353T_3^*(x) - 0.10515T_4^*(x) + 0.0209$$
$$-0.0089923T_6^*(x) + 0.002038T_7^*(x).$$

Therefore,

$$u_{FPA} = -1.0146x + 0.4854x^2 + 0.9883x^3 - 0.4096x^4 + 0.1077x^5 - 0.2878x^6 + 0.1304x^7. \quad (4.43)$$

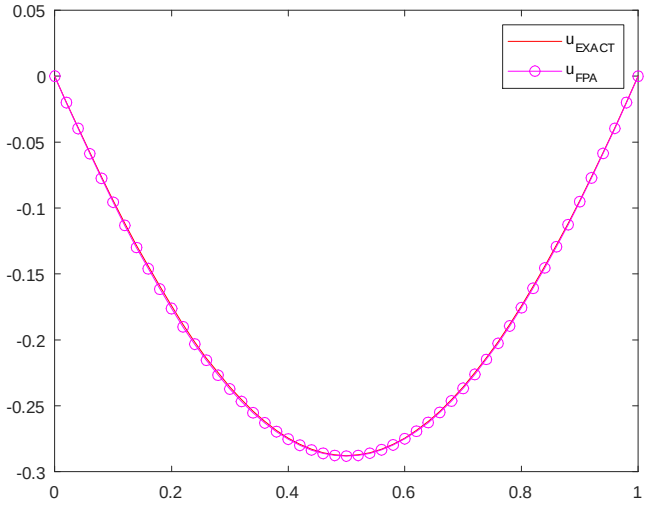

Figure 4.8: Fig 4.8 Exact Solution vs. Approximate Results: Bernoulli Problem N=7

The figure 4.8 illustrates the exceptional concordance between the numerical solution generated by our proposed algorithm and the exact analytical solution, confirming the method's high efficiency and successful convergence.

For N=9 Using Chebyshev polynomials for $N = 9$ and the residual,

$$R_{glob} = R_0^2 + R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2 + R_6^2 + R_7^2 + R_8^2 + R_9^2. \tag{4.44}$$

Parameters of Flower Pollination Algorithm utilized for problem are,

$Ub = 2$;

$Lb = -2$;

$n = 25$;

$p = 0.5$;

$N_{iter} = 10000$;

$d = 10$,

$w = 1$.

The coefficients got from FPA are,

$$\begin{cases} a = -0.047647, \\ b = -0.044291, \\ c = -0.17647, \\ d = 0.38347, \\ e = -0.14118, \\ f = 0.034289, \\ g = -0.007565, \\ h = -0.0045317, \\ i = 0.0044153, \\ j = -0.0010502. \end{cases}$$

From this results we conclude the value of $u(x)$,

$$\begin{aligned} u_{FPA} &= -0.047647T_0^*(x) - 0.044291T_1^*(x) - 0.17647T_2^*(x) + 0.38347T_3^*(x) - 0.14118T_4^*(x) + 0.034289 \\ &\quad -0.0045317T_7^*(x) + 0.0044153T_8^*(x) - 0.0010502T_9^*(x). \end{aligned}$$

Therefore,

$$u_{FPA} = -0.0001 - 1.0010x + 0.4990x^2 + 0.7203x^3 - 0.0599x^4 + 0.6025x^5 \quad (4.46)$$
$$-1.3724x^6 + 0.3149x^7 + 0.5652x^8 - 0.2689x^9.$$

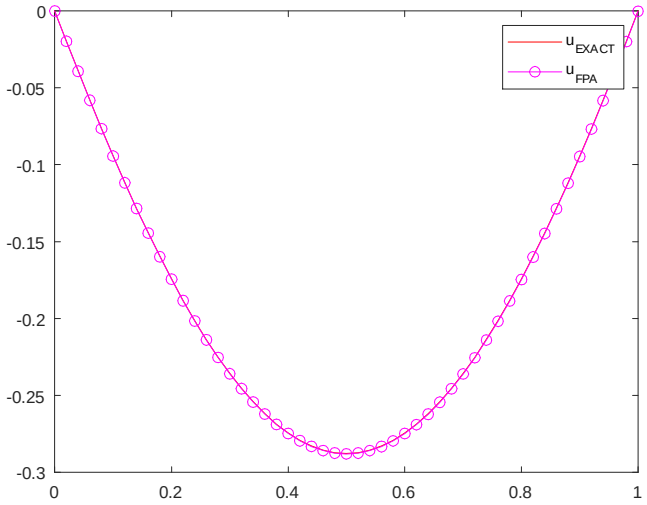

Figure 4.9: Fig 4.9 Exact Solution vs. Approximate Results: Bernoulli Problem N=9

As illustrated in the figure 4.9, the proposed algorithm significantly outperforms prior numerical schemes. Its solution is visually indistinguishable from the exact solution, a result that is quantitatively supported by it yielding the lowest error norm in our comparative analysis.

Optimizer	*RMSE*
CMSA $N = 5$	1.9×10^{-03}
CMSA $N = 7$	1.1×10^{-03}
CMSA $N = 9$	2.8144×10^{-04}
PSO [32]	3.0503×10^{-04}

Table 4.7: Comparison table of RMSE for the non-linear problem

The table 4.7 shows the Root Mean Square Error obtained from the approximate solution of the Bernoulli problem, using the Chebyshev metaheuristic solver approach and the

approach introduced in [32]. The RMSE given by the CMSA, for $N = 5$ and $N = 7$, are smaller then the RMSE obtained by the method introduced in [32]. The RMSE obtained from the CMSA is the smaller one for $N = 9$.

As seen in the table the error decreases when the N increases, and it gives te lowest error comparing by the method proposed in [32].

4.4.3 Initial Value Problem

Let's solve an integro-differential problem as an initial value problem, using the proposed method

Integro-Differential Equation

Supposing the linear integro-differential equation

$$\begin{cases} u'(x) + 2u(x) + 5\int_0^x u(t)dt = H(x), x \in [0, \pi], \\ u(0) = 0. \end{cases} \tag{4.47}$$

Where $H(x)$ is the Heaviside step function,

$$\begin{cases} H(x) = 1, \text{ if } x \geq 0, \\ H(x) = 0, \text{ else.} \end{cases}$$

After differentiating the problem it can be converted to this ordinary differential equation,

$$\begin{cases} u''(x) + 2u'(x) + 5u(x) = 0, x \in [0, \pi], \\ u(0) = 0, u'(0) = 1. \end{cases} \tag{4.48}$$

Analytical solution:

The exact solution of the proposed problem is

$$u_{exact}(x) = 1/2 \exp(-x) \sin(2x). \tag{4.49}$$

Chebyshev metaheuristic solver approach solution:

This problem also has been solved previoulsy by Babaei in [32], using Particle Swarm Optimization, there we will solve it using the new approach by upgrading the degree of Chebyshev polynomials until getting better results, like in the previous problem.
The Chebyshev polynomials used are calculated using the mapping $T_N^*(x) = T_N(\frac{2x-\pi}{\pi})$

For N=5: Calculating the residual using Chebyshev polynomials,

$$R_{glob} = R_0^2 + R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2. \tag{4.50}$$

Parameters of Flower Pollination Algorithm utilized for problem are,

$Ub = 2;$

$Lb = -2;$

$n = 25;$

$p = 0.5;$

$N_{iter} = 10000;$

$d = 6,$

$w = 1.$

Here the results,

$$\begin{cases} a = -0.37182, \\ b = 0.87766, \\ c = -0.3681, \\ d = 0.036336, \\ e = -0.0037344, \\ f = 1.44 \times 10^{\{-05\}}. \end{cases}$$

From this results we conclude the value of $u(x)$,

$$u_{FPA} = -0.37182T_0(x) + 0.87766T_1(x) - 0.3681T_2(x) + 0.036336T_3(x) - 0.0037344T_4(x) \quad (4.51)$$
$$+1.44 \times 10^{\{-05\}}T_5(x).$$

Thus,

$$u_{FPA}(x) = 1.4400 \times 10^{\{-05\}} + 0.7660x - 0.7661x^2 + 0.1559x^3 + 0.0299x^4 - 0.0085x^5. \quad (4.52)$$

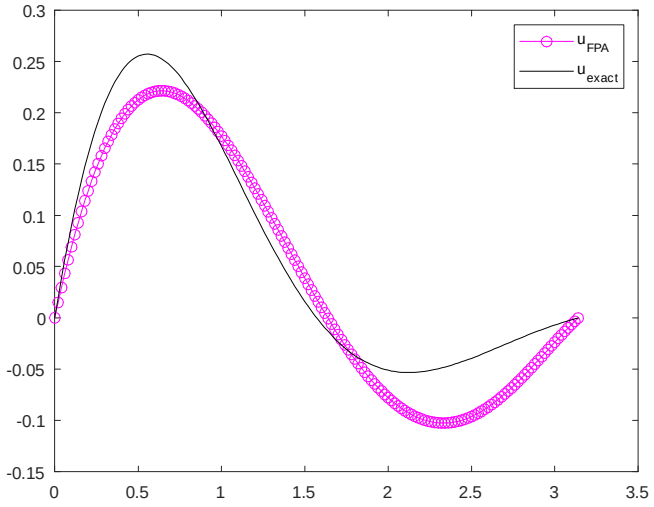

Figure 4.10: Fig 4.10. Exact Solution vs. Approximate Results: Integro-Differential Problem N=5

This graph 4.10 shows our new method's approximate solution with the exact solution to the integro-differential problem. the approximate solution is near to the exact one but there are some deviations in several points.

For N=7 Calculating the residual using Chebyshev polynomials,

$$R_{glob} = R_0^2 + R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2 + R_6^2 + R_7^2. \quad (4.53)$$

Parameters of Flower Pollination Algorithm utilized for problem are,

$Ub = 2;$

$Lb = -2;$

$n = 25;$

$p = 0.5;$

$N_{iter} = 10000;$

$d = 8,$

$w = 1.$

Here the results,

$$\begin{cases} a = -0.257686, \\ b = 0.671156, \\ c = -0.17463, \\ d = -0.13963, \\ e = 0.084822, \\ f = -0.01814, \\ j = 0.00177379, \\ h = -0.000067. \end{cases}$$

From this results we conclude the value of $u(x)$,

$$\begin{aligned} u_{FPA} &= -0.257686T_0(x) + 0.671156T_1(x) - 0.17463T_2(x) - 0.13963T_3(x) + 0.084822T_4(x) \\ &\quad -0.01814T_5(x) + 0.00177379T_6(x) - 0.000067T_7(x). \end{aligned} \tag{4.54}$$

Thus,

$$u_{FPA}(x) = -7.7900\times10^{\{-06\}} + 0.9998x - 0.9959x^2 - 0.1995x^3 + 0.5934x^4 - 0.2827x^5 + 0.0568x^6 - 0.0043x^7. \tag{4.55}$$

The graph 4.11 shows that increasing N gives more efficient approximate solution, and a

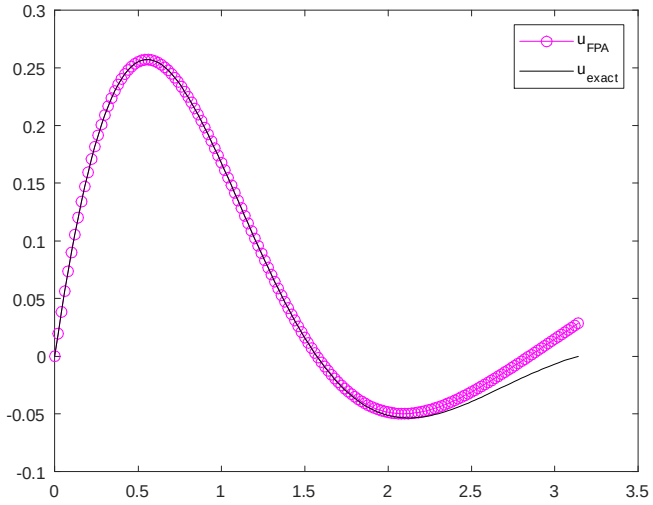

Figure 4.11: Fig 4.11 Exact Solution vs. Approximate Results: Integro-Differential Problem N=7

smaller deviation from the exact solution.

For N=9 Using Chebyshev polynomials for $N = 9$ and the residual,

$$R_{glob} = R_0^2 + R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2 + R_6^2 + R_7^2 + R_8^2 + R_9^2. \tag{4.56}$$

Parameters of Flower Pollination Algorithm utilized for problem are,

$Ub = 2;$

$Lb = -2;$

$n = 30;$

$p = 0.5;$

$N_{iter} = 10000;$

$d = 10,$

$w = 1.$

The coefficients got from FPA are,

$$\begin{cases} a = 0.055832, \\ b = -0.096777, \\ c = -0.004309, \\ d = 0.091663, \\ e = -0.055955, \\ f = 0.006679, \\ g = 0.004310, \\ h = -0.001654, \\ i = 0.000083, \\ j = 0.000057. \end{cases}$$

We conclude that,

$$\begin{aligned} u_{FPA} &= 0.055832T_0(x) - 0.096777T_1(x) - 0.004309T_2(x) + 0.091663T_3(x) - 0.055955T_4(x) \\ &\quad +0.006679T_5(x) + 0.004310T_6(x) - 0.001654T_7(x) + 0.000083T_8(x) + 0.000057T_9(x). \end{aligned} \tag{4.57}$$

Therefore,

$$\begin{aligned} u_{FPA} &= -0.000007004795304 + 0.999959206263 03x - 0.999544852205 02x^2 - 0.177733710469 94x^3 \\ &\quad +0.538549052301467x^4 - 0.213683022676613x^5 + 0.010093517682470x^6 \\ &\quad +0.012946975958491x^7 - 0.003284930647576x^8 + 0.000252664933546x^9. \end{aligned} \tag{4.58}$$

This graph 4.12 illustrates the powerful convergence of our method, as increasing N yields an approximate solution that is nearly indistinguishable from the exact solution. The exceptional alignment underscores the precision and reliability of our approach.

The table shows the Root Mean Square Error obtained by the approximate solution of integro-differential equation, using the Chebyshev metaheuristic solver approach and the approach introduced in [32]. The RMSE given by the CMSA, for $N = 5$ and $N = 7$, are

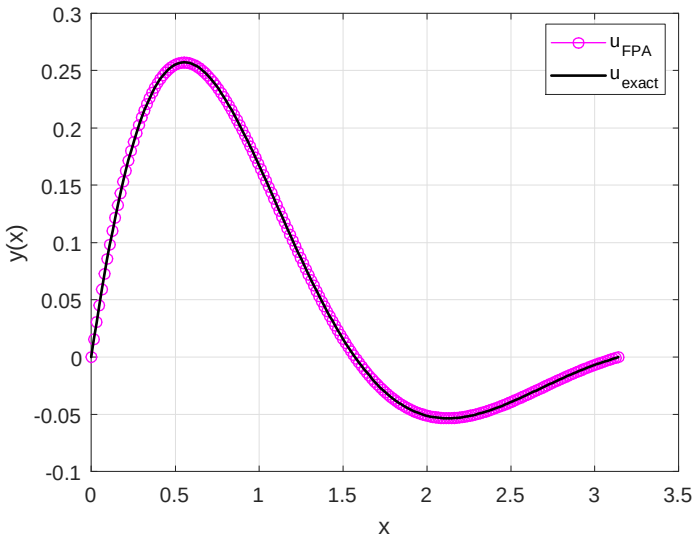

Figure 4.12: Fig 4.12. Exact Solution vs. Approximate Results: Integro-Differential Problem N=7

Optimizer	*RMSE*
CMSA $N = 5$	3.14×10^{-02}
CMSA $N = 7$	1.05×10^{-02}
CMSA $N = 9$	7.704629×10^{-05}
PSO [32]	1.805×10^{-01}

Table 4.8: Comparison table of RMSE for the integro-differential problem

smaller then the RMSE obtained by the method introduced in [32]. The RMSE obtained from the CMSA is the smaller one for $N = 9$.

4.5 Conclusion

In this chapter, a novel method for calculating approximate solutions have been introduced called Chebyshev metaheuristic solver approach, which was tested for different types of boundary value problems and an initial value problem. We solved several boundary problems and an integdo-diffrential problem using the new proposed method for different degrees N, accompanied by a comparison between the results given by the new method and the method introduced in [32] and the exact solution. The exact solutions were given for

all cases.

Our comparative analysis showed that the Chebyshev metaheuristic solver approach consistently yielded superior approximations. These findings emphasizes the effectiveness and accuracy of the proposed approach, validating its potential to enhance boundary value problem solving capabilities.

The results validate the theoretical framework discussed earlier, proving the practical viability of the new method.

Future explorations should research on the applicability of this method across different domains and problem sets, potentially give rise to further advancements. Other studies can be made to inhence the method by changing the parameters of the algorithm.

This chapter's victory in achieving better approximations underscores the importance of continual innovation in mathematical problem-solving techniques.

General Conclusion

This research has conducted a comprehensive exploration of merging numerical computation techniques with intelligent optimization algorithms, resulting in the creation of an innovative hybrid approach for differential equation resolution. The primary goal was to address the shortcomings of conventional solution methods by establishing a dependable, precise, and broadly applicable framework. This study effectively illustrates the substantial benefits of merging the exceptional precision characteristics of spectral techniques with the robust global optimization capabilities of the Flower Pollination Algorithm (FPA). The developed framework, designated as the *Chebyshev Metaheuristic Solver Approach,* constitutes a noteworthy advancement in computational mathematics.

The academic exploration commenced with a fundamental examination of spectral techniques in Chapter 1, where we demonstrated their theoretical advantages for differential equation solutions, with particular emphasis on the Chebyshev collocation approach. This method's effectiveness stems from its capacity to reach spectral precision by converting complex differential equations into algebraic equation systems through Chebyshev polynomial approximation. Nevertheless, resolving the resulting system, particularly for nonlinear scenarios, creates a significant optimization challenge.

Chapter 2 addressed this challenge through an extensive examination of the evolutionary algorithms and metaheuristic domain. This section explored the theoretical foundations of optimization, categorizing algorithms and analyzing their fundamental search strategies, spanning from random walks to Lévy flights. This investigation highlighted the essential

equilibrium between intensification and diversification, which is critical for avoiding local optima and achieving global convergence. This theoretical foundation supported the choice of an advanced, nature-inspired algorithm over traditional gradient-based approaches.

Chapter 3 concentrated on the specific metaheuristic selected for this research: the Flower Pollination Algorithm (FPA). We examined its biological foundation, the pollination mechanisms of flowering plants and transformed its cross-pollination principles (global search through Lévy flights) and self-pollination (local search) into an effective optimization algorithm. An extensive review of FPA variations and parameter optimization strategies demonstrated our dedication to not merely implementing, but thoroughly comprehending and enhancing the optimization methodology.

The integration of these separate domains is outlined in Chapter 4, which describes the development and verification of the proposed Chebyshev Metaheuristic Solver Approach. This chapter represents the primary contribution of this thesis. The differential equation is initially discretized using the Chebyshev collocation method, generating a residual function as the objective function for an optimization problem. The FPA is subsequently utilized to systematically identify the optimal Chebyshev coefficients that minimize this residual, thus providing the solution to the original equation. The effectiveness of this hybrid solver was thoroughly evaluated using various benchmark problems, encompassing both linear (homogeneous and non-homogeneous), nonlinear boundary value problems, and an integro-differential problem. The computational results consistently showed outstanding accuracy and stability, confirming the hypothesis that this hybrid approach successfully combines the spectral precision of Chebyshev polynomials with the powerful optimization capabilities of the FPA.

Considering future prospects, the significance of this research extends considerably beyond the specific problems examined here. The adaptability and proven performance of the *Chebyshev metaheuristic solver approach* indicate its potential for significantly broader applications across scientific and engineering domains. This encompasses complex, multi-

dimensional challenges in areas such as fluid mechanics, thermal transfer, structural analysis, and quantum mechanics. Additionally, the concept of transforming complex mathematical problems into optimization tasks and solving them using metaheuristics has significant relevance in emerging fields.

Multiple promising directions for future investigation arise from this work. Initially, the methodology can be expanded to address more complex systems, including partial differential equations (PDEs), integral equations, and coupled differential equation systems. Additionally, while the FPA demonstrated high effectiveness, a comparative analysis incorporating other advanced metaheuristics or hybrid versions could produce additional performance improvements. Finally, exploring adaptive parameter control within the FPA could automate the optimization process, making the solver more autonomous and efficient.

In summary, this thesis successfully developed, implemented, and validated a powerful hybrid numerical-optimization framework. Through carefully combining the advantages of Chebyshev spectral methods and the Flower Pollination Algorithm, this work not only delivers a highly accurate solution tool for a challenging category of differential equations but also establishes a solid and innovative foundation for developing future high-precision computational approaches. The demonstrated capabilities of this methodology promise to enhance computational efficiency and accuracy, creating pathways for new discoveries and innovations across multiple scientific fields.

Bibliography

[1] A. E. Eiben, J.E. Smith. (2015). Introduction to Evolutionary Computing. Springer-Verlag Berlin Heidelberg, Natural Computing Series, Natural computing series, 2.

[2] A. S. Walter. (2007). Parial Differential Equations, an Introduction, second edition. Wiley.

[3] A. H. Gandomi, X. S. Yang, A. H. Alavi. (2013). Cuckoo search algorithm: a meta-heuristic approach to solve structural optimization problems. Engineering with Computers, 29(2): 17–35.

[4] C. Grosan, A. Abraham. (2011). Evolutionary Algorithms. Intelligent Systems. Intelligent Systems Reference Library. Springer, Berlin, Heidelberg, 17:345–386.

[5] C. H. Edwards, D. E. Penney. (2008). Elementary differential equations, sixth edition. Pearson Education, Inc.

[6] C. Shilaja, K. Ravi. (2007). Optimization of emission/economic dispatch using euclidean affine Flower Pollination Algorithm (EFPA) and binary FPA (BFPA) in solar photo voltaic generation. Renewable Energy, 107:550-566.

[7] D. Chakraborty, S. Saha, O. Dutta. (2014). DE-FPA: A hybrid Differential Evolution-Flower Pollination Algorithm for function minimization. 2014 International Conference on High Performance Computing and Applications (ICHPCA),IEEE, 1-6.

[8] D. Chakraborty, S. Saha, S. Maity (2015). Training feedforward Neural Networks using hybrid Flower Pollination-Gravitational Search Algorithm. 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE),IEEE, pages: 261-266.

[9] D. Karaboga. (2010). Artificial Bee Colony Algorithm. Scholarpedia, 5(3):6915.

[10] D. Rodrigues, X.-S. Yang, A. N. de Souza, J. P. Papa. (2014). Binary Flower Pollination Algorithm and its application to feature selection. Recent Advances in Swarm Intelligence and Evolutionary Computation, Springer, 585:85-100.

[11] D. Rodrigues, G. F. A. Silva, J. P. Papa, A. N. Marana, X.-S. Yang. (2016). EEG-based person identification through binary Flower Pollination Algorithm. Expert systms with applications, 62:81-90.

[12] E. Emary, H. M. Zawbaa, A. A. Hassanien, B. Parv. (2017). Multi-objective retinal vessel localization using Flower Pollination Search Algorithm with pattern search. Advances in Data Analysis and Classification, 11:611-627.

[13] E. Nabil. (2016). A modified Flower Pollination Algorithm for global optimization. Expert Systems with Applications, 57:192-203

[14] F. Ghedjemis, N. Khelil. (2025). Spectral approximations optimized by flower pollination algorithm for solving differential equations. International Journal of Computational Methods and Experimental Measurements, 13(2):343-349.

[15] G. Namachivayam, C. Sankaralingam, S. K. Perumal, S. T. Devanathan. (2016). Reconfiguration and capacitor placement of radial distribution systems by modified Flower Pollination Algorithm. Electric Power Components and Systems, 44(13):1492-1502.

[16] G. S. McDonald. (2004). Differential equation second order (inhomogeneous). Promoting Phisics Learning & Teaching opportunities.

[17] H. M. Dubey, M. Pandit, B. K. Panigrahi. (2015). A biologically inspired modified Flower Pollination Algorithm for solving economic dispatch problems in modern power systems. Cognitive Computation, 7(5):594-608.

[18] H. M. Zawbaa,A. E. Hassanien, W. Yamani, B. Parv. (2015). Hybrid Flower Pollination Algorithm with rough sets for feature selection. 11th International Computer Engineering Conference (ICENCO),IEEE, 278-283.

[19] J. H. Holland. (1975). Genetic Algorithms and Adaptation. Adaptive Control of Ill-Defined Systems. NATO Conference Series, Springer, Boston, MA, 16:317-333.

[20] J. P. Boyed. (2000). Chebyshev and fourier spectral methods, second edition. University of Michigan, DOVER Publications, INC.

[21] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest ,D. Fister. (2013). A Brief Review of nature-inspired algorithms for optimization. Elektrotehniški Vestnik, 80(3).

[22] J. Kennedy, R. Eberhart. (1995). Particle Swarm Optimization. Proceeding of ICNN'95-International Conference on Neural Networks 4, 1942-1948.

[23] I. E. Mergos Jr, X.-S. Yang. (2021). Flower Pollination Algorithm parameters tuning. Soft Computing, Springer, 25:1429-1447.

[24] J. P. Ram, T. S. Babu, T. Dragicevic, N. Rajasekar. (2017). A new hybrid Bee Pollinator Flower Pollination Algorithm for solar PV parameter estimation. Energy Conversion and Management, 135:463-476.

[25] J. A. Regalado, B. E. Emilio, E. Cuevas. (2015). Optimal power flow solution using Modified Flower Pollination Algorithm. 2015 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), 1-6.

[26] I. Sehili.(2018). On Numerical resolution of boundary value problems using spectral methods. University of Mohamed Khider, Biskra.

[27] J. C. Mason, D. C. Handscomb. (2002). Chebyshev polynomials. A CRC Press Company.

[28] J. Shen, T. Tang and L.L. Wang. (2011). Spectral methods algorithms, analysis and applications. Springer Series in Computational Mathematics 41.

[29] K. B. Pop, V. R. Chifu, I. Salomie, D. S. Racz, R. M. Bonta. (2017). Hybridization of the Flower Pollination Algorithm: a case study in the problem of generating healthy nutritional meals for older adults. Nature-Inspired Computing and Optimization, Springer,151-183.

[30] L. Valenzuela, F. Valdez, P. Melin. (2016). Flower Pollination Algorithm with fuzzy approach for solving optimization problems. Nature-Inspired Design of Hybrid Intelligent Systems, 667:357-369.

[31] M. Abdel-Baset, I. Hezam. (2016). A hybrid Flower Pollination Algorithm for engineering optimization problems. International Journal of Computer Applications, 140(12):10-23.

[32] M. Babaei. (2013). A general approach to approximate solutions of nonlinear differential equations using Particle Swarm Optimization. Applied Soft Computing,13(7):3354-3365.

[33] M. Gendreau, J.-Y. Potvin. (2005). Metaheuritics in combinational optimization. Annals of Operations Research, 140(1):189-213.

[34] M. J. Kochenderfer, T.A. Wheeler. (2019). Algorithms for Optimization. The MIT Press, Cambridge, Massachusetts, London, England.

[35] M. Metwalli, M. abdel-Baset, I. Hazem. (2015). A modified Flower Pollination Algorithm for fractional programming problems. International Journal of Intelligent Systems and Applications in Engineering, 3(3):116-123.

[36] O. Abdel-Raouf, M. Abdel-Baset, I. El-Henawy. (2014). A new hybrid Flower Pollination Algorithm for solving constrained global optimization problems. Computer Science International Journal of Applied Operational Research, 4(2):1-13.

[37] O. K. Meng, O. Pauline, S. C. Kiong, A. Waheb Hanani, N. Jafferi. (2016). Application of Modified Flower Pollination Algorithm on mechanical engineering design problem. IOP Conference Series: Materials Science and Engineering, 165.

[38] R. L. Adler, T. J. Rivlin. (1964). Ergodic and mixing properties of Chebyshev polynomials. Proceedings of the American Methematical Society, 15: 794-796.

[39] R. Jensi, G. W. Jiji. (2015). Hybrid data clustering approach using K-Means and Flower Pollination Algorithm. arXiv:1505.03236.

[40] R. Rastogi, O.P. Misra, R. Mishra. (2023). A Chebyshev polynomial approach to approximate solution of differential equations using Differential Evolution. Engineering Application of Artificial Intelligence, 126.

[41] R. Salgotra, U. Singh. (2017). Application of mutation operators to Flower Pollination Algorithm. Expert Systems with Applications, 79:112-129.

[42] R. Wang, Y. Zhou, C. Zhao, H. Wu. (2015). A hybrid Flower Pollination Algorithm based modified randomized location for multi-threshold medical image segmentation. Biomedical Materials and Engineering, 26(1):45-51.

[43] S. Dan. (2013). Evolutionary optimization algorithms: Biologically-inspired and population- based approaches to computer intelligence. John Wiley & Sons, Inc.

[44] S. Kalra, S. Arora. (2016). Firefly algorithm hybridized with Flower Pollination Algorithm for multimodal functions. Proceedings of the International Congress on Information and Communication Technology, Springer, 207-219.

[45] S. M. Nigdeli, G. Bekdas, X.-S. Yang. (2017). Optimum tuning of mass dampers by using a hybrid method using harmony search and Flower Pollination Algorithm. International Conference on Harmony Search Algorithm, Springer, 222-231

[46] S. Sarjiya, P. H. Putra, T. A. Saputra. (2016). Modified Flower Pollination Algorithm for nonsmooth and multiple fuel options economic dispatch. International Conference on Information & Communication Technology and Systems (ICTS), 1-5.

[47] S. Xu, Y. Wang. (2017). Parameter estimation of photovoltaic modules using a hybrid Flower Pollination Algorithm. Energy Conversion and Management, 144:53-68.

[48] T. Bäck. (1996). Evolutionary algorithms in theory and practice. Oxford University Press, New York.

[49] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. (2007). Numerical recipes the art of scientific computing, third edition. Cambridge University Press.

[50] W. Yamany, H. M. Zawbaa, E. Emary, A. E. Hassanien. (2015). Attribute reduction approach based on Modified Flower Pollination Algorithm. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE),1-7.

[51] W. Zhang, Z. Qu, K. Zhang, W. Mao, Y. Ma, X. Fan. (2017). A combined model based on CEEMDAN and Modified Flower Pollination Algorithm for wind speed forecasting. Energy Conversion and Management, 136:439-451.

[52] X.-S. Yang, M. Karamanoglu, X. He. (2013). Multi-objective Flower Pollination Algorithm for optimization. Procedia Computer Science, 18:861-868.

[53] X.-S. Yang, M. Karamanoglu, X. He. (2014). Flower Pollination Algorithm: a novel approach for multiobjective optimization. Engineering Optimization, 46(9):1222-1237.

[54] X.-S. Yang. (2010). Bat Inspired Algorithm, A new metaheuristic Bat-Inspired Algorithm. Nature inspired cooperative strategies for opitimization, 65-74.

[55] X.-S. Yang. (2010). Firefly Algorithm: An introduction with metaheuristic applications. Research and Development in Intelligent Systems, Springer-Verlag London limited, 209-218.

[56] X.-S. Yang. (2012). Flower Pollination Algorithm for global optimization. Unconventional Computation and Natural Computation, Springer, 7445:240-249.

[57] X.-S. Yang. (2010). Nature-Inspired Metaheuristic Algorithms. Luniver Press.

[58] Y. Xinjie, G. Mitsuo. (2010). Introduction to evolutionary algorithms. Springer Verlag London Limited.

[59] Y. Zhou, R. Wang, Q. Luo. (2016). Elite opposition-based Flower Pollination Algorithm. Neurocomputing, 188:294-310.

[60] Z. A. Alkareem Alyasseri, A. T. Khader, M.A. AlBetar, M.A. Awadallah, X.S. Yang. (2018). Variants of the Flower Pollination Algorithm: A review. Nature-Inspired Algorithms and Applied Optimization, 744:91-118.

[61] Z. A. E. Moiz Dahi, C. Mezioud, A. Draa. (2016). On the efficiency of the Binary Flower Pollination Algorithm: Application on the antenna positioning problem. Applied Soft Computing, 47:395-414.

Appendix A: MATLAB

MATLAB (short for MATrix LABoratory) is a high-level programming environment and numeric computing platform widely used across scientific research, engineering, and applied mathematics domains. Originating in the late 1970s to facilitate matrix computations, MATLAB has evolved into a versatile tool integrating numerical analysis, visualization, algorithm development, and system simulation. Its intuitive syntax, extensive function libraries, and powerful graphical capabilities enable efficient handling of complex data, numerical modeling, and iterative algorithmic design.

Distinctive for its matrix-based architecture, MATLAB allows users to seamlessly perform vectorized operations, linear algebra, differential equations, signal processing, and other computational tasks with remarkable ease and speed. The environment supports script and function development, interactive command execution, and integration with external codebases, enhancing reproducibility and extensibility in research workflows.

In the context of this thesis, MATLAB served as a critical computational backbone—enabling rigorous data analysis, automated simulations, and visualization of results. Its contributions span from preliminary data preprocessing and model prototyping to the final validation stages, ensuring robustness and accuracy. This appendix aims to provide readers unfamiliar with MATLAB a concise overview of its foundational features, operational philosophy, and relevance to contemporary scientific inquiry.

Appendix B : MATLAB's Code Used

4.6 MATLAB Code of the First Chapter

4.6.1 Generation of Chebyshev Polynomials of the First Kind

```
% Define the Chebyshev polynomial function recursively
function T = chebyshev T(n,x)
if n==0
T=ones(size(x));
elseif n==1
T=x;
else
T=2*x.*chebyshevT(n-1,x)-chebyshevT(n-2,x);
end
end
% Example N=5
n_max=5;
x=linspace(-1,1,100);
figure;
hold on;
```

```
for n=0:n_max
T=chebyshevT(n,x);
plot(x,T,'DisplayName', ['T_'num2str(n)]);
end
hold off;
legend;
title('Chebyshev Polynomials of the First Kind');
xlabel('x');
ylabel('T_n(x)');
```

4.6.2 Generation of Chebyshev Polynomials of the First Kind in [1,4]

```
clear;
clc;
close all;
x = linspace(1, 4, 500);
T0s = ones(size(x));
T1s = (1/3) * (2*x - 5);
T2s = (1/9) * (8*x.^2 - 40*x + 41);
T3s = (1/27) * (32*x.^3 - 240*x.^2 + 546*x - 365);
T4s = (1/81) * (128*x.^4 - 1280*x.^3 + 4512*x.^2 - 6560*x + 3281);
T5s = (1/243) * (512*x.^5 - 6400*x.^4 + 30560*x.^3 - 69200*x.^2 + 73810*x - 29525);
figure;
hold on;
plot(x, T0s, 'LineWidth', 2);
plot(x, T1s, 'LineWidth', 2);
plot(x, T2s, 'LineWidth', 2);
```

```
plot(x, T3s, 'LineWidth', 2);
plot(x, T4s, 'LineWidth', 2);
plot(x, T5s, 'LineWidth', 2);
hold off;
xlabel('x');
ylabel('T_n((2x-5)/3)');
legend('T_0(s)', 'T_1(s)', 'T_2(s)', 'T_3(s)', 'T_4(s)', 'T_5(s)', 'Location', 'best');
grid on;
ylim([-1.1, 1.1]); % Focus on the interesting part of the graph
```

4.6.3 Generation of Shifted Chebyshev Polynomials

```
clear;
clc;
close all;
x = linspace(0, 1, 500);
T0_star = ones(size(x));
T1_star = 2*x - 1;
T2_star = 8*x.^2 - 8*x + 1;
T3_star = 32*x.^3 - 48*x.^2 + 18*x - 1;
T4_star = 128*x.^4 - 256*x.^3 + 160*x.^2 - 32*x + 1;
T5_star = 512*x.^5 - 1280*x.^4 + 1120*x.^3 - 400*x.^2 + 50*x - 1;
figure;
hold on;
plot(x, T0_star, 'LineWidth', 2);
plot(x, T1_star, 'LineWidth', 2);
plot(x, T2_star, 'LineWidth', 2);
plot(x, T3_star, 'LineWidth', 2);
```

```
plot(x, T4_star, 'LineWidth', 2);
plot(x, T5_star, 'LineWidth', 2);
hold off;
xlabel('x');
ylabel('T*_n(x)');
legend('T*_0(x)', 'T*_1(x)', 'T*_2(x)', 'T*_3(x)', 'T*_4(x)', 'T*_5(x)', 'Location',
'best');
grid on;
ylim([-1.1, 1.1]);
clear;
clc;
close all;
x = linspace(0, 1, 500);
T0_star = ones(size(x));
T1_star = 2*x - 1;
T2_star = 8*x.^2 - 8*x + 1;
T3_star = 32*x.^3 - 48*x.^2 + 18*x - 1;
T4_star = 128*x.^4 - 256*x.^3 + 160*x.^2 - 32*x + 1;
T5_star = 512*x.^5 - 1280*x.^4 + 1120*x.^3 - 400*x.^2 + 50*x - 1;
figure;
hold on;
plot(x, T0_star, 'LineWidth', 2);
plot(x, T1_star, 'LineWidth', 2);
plot(x, T2_star, 'LineWidth', 2);
plot(x, T3_star, 'LineWidth', 2);
plot(x, T4_star, 'LineWidth', 2);
plot(x, T5_star, 'LineWidth', 2);
```

hold off;

title('Verification Plot using Explicit Formulas');

xlabel('x');

ylabel('T*_n(x)');

legend('T*_0(x)', 'T*_1(x)', 'T*_2(x)', 'T*_3(x)', 'T*_4(x)', 'T*_5(x)', 'Location', 'best');

grid on;

ylim([-1.1, 1.1]);

4.6.4 MATLAB Code to Solve the First Example Using Chebyshev Collocation Method

% MATLAB code to solve -d^2u/dx^2=exp(x) with u(-1)=u(1)=0 using Chebyshev spectral collocation

% 1. Define the number of collocation points (N+1)

N=20; % Degree of the polynomial approximation. More points = higher accuracy.

% 2. Compute Chebyshev-Gauss-Lobatto (CGL) points

% These are the roots of T_N′(x) and are given by x_k=cos(pi*k/N)

k=0:N;

x=cos(pi*k/N)'; % Column vector of collocation points

%3. Construct the second-order Chebyshev differentiation matrix (D2)

% We use a function for this, often provided in spectral methods libraries

% For demonstration, we'll construct it directly.

% For N=0,D=0. For N=1,D=[-1/2,1/2;1/2,-1/2].

% Building the full D and then D2 is more robust.

D= zeros(N+1,N+1);

for i=1:N+1

for j=1:N+1

```
if i==j
if i==1% x_0=1
D(i,j)=(2*N^2+1)/6;
elseif i==N+1% x_N=-1
D(i,j)=-(2*N^2+1)/6;
else % x_k internal point
D(i,j)=-x(i)/(2*(1-x(i)^2));
end
else
c_i=1; if i==1 || i==N+1, c_i=2;
end
c_j=1; ifj==1 ||j==N+1,c_j=2;
end
D(i,j)=(c_i*(-1)^(i+j))/(c_j*(x(i)-x(j)));
end
end
end
D2=D*D; % Second differentiation matrix
% 4. Apply boundary conditions and set up the linear system
% The boundary conditions are u(-1)=0 and u(1)=0.
% In our CGL points, x(1)=1 (for k=0) and x(N+1)=-1 (for k=N).
% So, u(1) corresponds to u(x(1)) and u(-1) corresponds to u(x(N+1)).
% The system we need to solve is -D2*u_vec=f_vec
% where u_vec is the vector of u values at collocation points.
% The original system is -D2*U=F, where F=exp(x)
F=exp(x);
% Modify D2 and F to incorporate boundary conditions
```

% The first and last rows of D2 correspond to x=1 and x=-1 respectively.

% For Dirichlet boundary conditions, we effectively replace these rows

% with identity rows that enforce u(1)=0 and u(-1)=0.

% Create a modified D2_mod and F_mod

D2_mod=-D2; % The left-hand side is -d^2u/dx^2

% Set boundary condition rows

D2_mod(1,:)=0; % Clear the first row

D2_mod(1,1)=1; % Set u(x_1)=0 (which is u(1)=0)

F(1)=0;

D2_mod(N+1,:)=0; % Clear the last row

D2_mod(N+1,N+1)=1; % Set u(x_N+1)=0 (which is u(-1)=0)

F(N+1)=0;

% 5. Solve the linear system

u_numerical=D2_mod\F;

% 6. Plot the numerical solution

figure;

plot(x, u_numerical, 'o-', 'LineWidth', 1.5, 'DisplayName', 'Numerical Solution');

hold on;

% Compare with the exact solution

% The exact solution for $-u" = exp(x)$ with u(-1)=u(1)=0

% is u(x)=exp(x)-(cosh(1)+sinh(1)*x)/cosh(1)

% or u(x)=exp(x)-(e^1*(x+1)+e^-1*(1-x))/2

% or u(x)=exp(x)-((e-e^-1)/2*x+(e+e^-1)/2)

% Let's use the integrated form: u(x)=-exp(x)+Ax+B

% $u(-1) = -exp(-1) - A + B = 0 \Rightarrow B = A + exp(-1)$

% $u(1) = -exp(1) + A + B = 0 \Rightarrow -exp(1) + A + A + exp(-1) = 0$

% $2A = exp(1) - exp(-1) \Rightarrow A = (exp(1) - exp(-1))/2 = sinh(1)$

% $B = sinh(1) + exp(-1)$

% Exact solution is u_exact(x)=-exp(x)+sinh(1)*x+sinh(1)+exp(-1)

% Let's derive it correctly: $u"(x) = -exp(x) \Rightarrow u\prime(x) = -exp(x) + C1 \Rightarrow u(x) = -exp(x) + C1 * x + C2$

% u(-1)=-exp(-1)-C1+C2=0

% u(1)=-exp(1)+C1+C2=0

% Subtracting the first from the second: (-exp(1)+exp(-1))+2*C1=0⇒2*C1=exp(1)-exp(-1)⇒C1=(exp(1)-exp(-1))/2=sinh(1)

% Add the two equations: -(exp(1)+exp(-1))+2*C2=0⇒2*C2=exp(1)+exp(-1)⇒C2=(exp(1)+exp(-1))/2=cosh(1)

u_exact=-exp(x)+sinh(1)*x+cosh(1);

plot(x,u_exact, 'r–', 'LineWidth', 1, 'DisplayName', 'Exact Solution');

title(['Chebyshev Spectral Collocation (N=',num2str(N), ')']);

xlabel('x');

ylabel('u(x)');

legend('show');

grid on;

4.6.5 MATLAB Code to Solve the Second Example Using Chebyshev Collocation Method

% MATLAB code to solve d^2u/dx^2=exp(u) with u(-1)=u(1)=0

clear; clc; close all;

%% 1. Define the number of collocation points (N+1)

N = 20;

%% 2. Compute Chebyshev-Gauss-Lobatto (CGL) points

k = 0:N;

x = cos(pi*k/N)';

```
%% 3. Construct the second-order Chebyshev differentiation matrix (D2)
if N == 0
D = 0;
else
c = [2; ones(N-1,1); 2] .* (-1).^(0:N)';
X = repmat(x, 1, N+1);
dX = X - X';
D = (c*(1./c)')./(dX + eye(N+1)); % First derivative matrix
D = D - diag(sum(D,2)); % Correcting the diagonal
end
D2 = D*D; % Second differentiation matrix
% 4. Set up the NONLINEAR system of equations
residual = @(U) bvp_residual(U, D2, N);
U_guess = zeros(N+1, 1);
%% 5. Solve the nonlinear system using fsolve
options = optimoptions('fsolve', 'Display', 'iter', 'TolFun', 1e-12);
% Call fsolve to find the root U of the residual function.
[U_numerical, fval, exitflag] = fsolve(residual, U_guess, options);
% Check for convergence
if exitflag <= 0
error('fsolve did not converge. Try a different initial guess or change solver options.');
end
fprintf('\nSolver converged successfully.\n');
%% 6. Plot the solution
x_fine = linspace(-1, 1, 200)';
u_fine = barycentric_interp(x, U_numerical, x_fine);
figure;
```

```
plot(x_fine, u_fine, 'b-', 'LineWidth', 2, 'DisplayName', 'Interpolated Numerical Solution');
hold on;
plot(x, U_numerical, 'ro', 'MarkerFaceColor', 'r', 'MarkerSize', 6, 'DisplayName', 'Collocation Points');
title(['Chebyshev Spectral Collocation for u" = exp(u) (N = ', num2str(N), ')']);
xlabel('x', 'FontSize', 12);
ylabel('u(x)', 'FontSize', 12);
legend('show', 'Location', 'best');
grid on;
box on;
%% Helper Functions
function F = bvp_residual(U, D2, N)
interior_eqs = D2(2:N, :) * U - exp(U(2:N));
bc1 = U(1) - 0;
bc_neg1 = U(N+1) - 0;
F = [bc1; interior_eqs; bc_neg1];
end
function u_interp = barycentric_interp(x_nodes, u_nodes, x_eval)
N = length(x_nodes) - 1;
w = [0.5; ones(N-1, 1); 0.5] .* (-1).^(0:N)';
numerator = zeros(size(x_eval));
denominator = zeros(size(x_eval));
for j = 1:length(x_nodes)
exact_match = abs(x_eval - x_nodes(j)) < 1e-12;
if any(exact_match)
u_interp(exact_match) = u_nodes(j);
```

```
end
non_match = ~exact_match;
term = w(j) ./ (x_eval(non_match) - x_nodes(j));
numerator(non_match) = numerator(non_match) + term * u_nodes(j);
denominator(non_match) = denominator(non_match) + term;
end
u_interp(~(denominator==0)) = numerator(~(denominator==0)) ./ denominator(~(denominator==0)
end
```

4.7 Code MATLAB for the Fourth Chapter

4.7.1 Flower Pollination Algorithm

```
% ------------------------------------------------------------%
% Flower pollenation algorithm (FPA), or flower algorithm %
% Programmed by Xin-She Yang @ May 2012 %
% ------------------------------------------------------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Notes: This demo program contains the very basic components of %
% the flower pollination algorithm (FPA), or flower algorithm (FA), %
% for single objective optimization. It usually works well for %
% unconstrained functions only. For functions/problems with %
% limits/bounds and constraints, constraint-handling techniques %
% should be implemented to deal with constrained problems properly. %
% %
% Citation details: %
%1)Xin-She Yang, Flower pollination algorithm for global optimization,%
% Unconventional Computation and Natural Computation, %
```

```
% Lecture Notes in Computer Science, Vol. 7445, pp. 240-249 (2012). %
%2)X. S. Yang, M. Karamanoglu, X. S. He, Multi-objective flower %
% algorithm for optimization, Procedia in Computer Science, %
% vol. 18, pp. 861-868 (2013). %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [best,fmin,N_iter]=fpa_demo(para)
% Default parameters
if nargin<1,
para=[20 0.8];
end
n=para(1); % Population size, typically 10 to 25
p=para(2); % probabibility switch
% Iteration parameters
N_iter=2000; % Total number of iterations
% Dimension of the search variables
d=3;
Lb=-2*ones(1,d);
Ub=2*ones(1,d);
% Initialize the population/solutions
for i=1:n,
Sol(i,:)=Lb+(Ub-Lb).*rand(1,d);
Fitness(i)=Fun(Sol(i,:));
end
% Find the current best
[fmin,I]=min(Fitness);
best=Sol(I,:);
S=Sol;
```

```
% Start the iterations – Flower Algorithm
for t=1:N_iter,
% Loop over all bats/solutions
for i=1:n,
% Pollens are carried by insects and thus can move in
% large scale, large distance.
% This L should replace by Levy flights
% Formula: x_i^{t+1}=x_i^t+ L (x_i^t-gbest)
if rand>p,
%% L=rand;
L=Levy(d);
dS=L.*(Sol(i,:)-best);
S(i,:)=Sol(i,:)+dS;
% Check if the simple limits/bounds are OK
S(i,:)=simplebounds(S(i,:),Lb,Ub);
% If not, then local pollenation of neighbor flowers
else
epsilon=rand;
% Find random flowers in the neighbourhood
JK=randperm(n);
% As they are random, the first two entries also random
% If the flower are the same or similar species, then
% they can be pollenated, otherwise, no action.
% Formula: x_i^{t+1}+epsilon*(x_j^t-x_k^t)
S(i,:)=S(i,:)+epsilon*(Sol(JK(1),:)-Sol(JK(2),:));
% Check if the simple limits/bounds are OK
S(i,:)=simplebounds(S(i,:),Lb,Ub);
```

```
end
% Evaluate new solutions
Fnew=Fun(S(i,:));
% If fitness improves (better solutions found), update then
if (Fnew<=Fitness(i)),
Sol(i,:)=S(i,:);
Fitness(i)=Fnew;
end
% Update the current global best
if Fnew<=fmin,
best=S(i,:) ;
fmin=Fnew ;
end
end
% Display results every 100 iterations
if round(t/100)==t/100,
best
fmin
end
end
% Output/display
disp(['Total number of evaluations: ',num2str(N_iter*n)]);
disp(['Best solution=',num2str(best),' fmin=',num2str(fmin)]);
% Application of simple constraints
function s=simplebounds(s,Lb,Ub)
% Apply the lower bound
ns_tmp=s;
```

```
I=ns_tmp<Lb;
ns_tmp(I)=Lb(I);
% Apply the upper bounds
J=ns_tmp>Ub;
ns_tmp(J)=Ub(J);
% Update this new move
s=ns_tmp;
% Draw n Levy flight sample
function L=Levy(d)
% Levy exponent and coefficient
% For details, see Chapter 11 of the following book:
% Xin-She Yang, Nature-Inspired Optimization Algorithms, Elsevier, (2014).
beta=3/2;
sigma=(gamma(1+beta)*sin(pi*beta/2)/(gamma((1+beta)/2)*beta*2^((beta-1)/2)))^(1/beta);
u=randn(1,d)*sigma;
v=randn(1,d);
step=u./abs(v).^(1/beta);
L=0.01*step;
% Objective function and here we used Rosenbrock's 3D function
function z=Fun(u)
z=(1-u(1))^2+100*(u(2)-u(1)^2)^2+100*(u(3)-u(2)^2)^2;
```