République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ MOHAMED KHIDER, BISKRA
FACULTÉ des SCIENCES EXACTES et des SCIENCES de la NATURE et de la VIE
DÉPARTEMENT DE MATHÉMATIQUES

Thèse présentée en vue de l'obtention du Diplôme de
Doctorat en Mathématiques
Option: Analyse Numérique et Optimisation
Par

Samiha Aichouche

Titre

Amélioration des Performances de Certaines Méthodes de Calcul Numérique à L'aide des Algorithmes Évolutionnaires.

Devant le jury

Président	M.C.A. **Zouheir Mokhtari**	Université M.K. Biskra
Rapporteur	M.C.A. **Naceur Khelil**	Université M.K. Biskra
Examinateur	M.C.A. **Leila Djerou**	Université M.K. Biskra
Examinateur	M.C.A. **El-Amir Djeffal**	Université L.K. Batna

2016

Improvement of Performance of Certain Numerical Computational Methods using Swarm Intelligence Algorithms

Improvement of Gregory's Formula using Artificial Bee Colony Algorithm.

Samiha Aichouche

AM Laboratory, University of Biskra, Algeria

Abstract

In this work, we prove that the Gregory Formula (G) can be optimized by minimizing some of their coefficients in the remainder term by using Artificial Bee Colony (ABC) Algorithm.
Experimental tests prove that obtained Formula can be rendered a powerful formula for library use.

Key words. Gregory Formula, Artificial Bee Colony (ABC) Algorithm, Numerical Integration.

All praise and thanks to God alone.

To my parents, my family and all of my friends,
Without whom none of my success would be possible.

Acknowledgements

I would like to express deepest gratitude to my advisor ***Dr. Naceur Khelil*** *for his full support, expert guidance, understanding and encouragement throughout my study and research. Without his incredible patience and timely wisdom and counsel, my thesis work would have been a frustrating and overwhelming pursuit.*

I express my appreciation to ***Dr.Zouheir Mokhtari***, ***Dr. Leila Djerou*** *and* ***Dr. El-Amir Djeffal*** *for accepting to read our work.*

My special thanks to the AM Laboratory, University of Biskra.

My thanks and appreciations go to our friends and colleagues.

Contents

List of Figures

List of Tables

Abbreviations

$\boldsymbol{F}$	Formal Power Series
$\boldsymbol{P}$	Polynomials (sub-algebra of $\boldsymbol{F}$)
$\boldsymbol{P*}$	Vector Space of all Linear Functional on $\boldsymbol{P}$
$deg\left(p\left(x\right)\right)$	Degree of $p\left(x\right)$
$O\left(f\left(t\right)\right)$	Order of $f\left(t\right)$
$M\left(f_k\right)$	Infinite Matrix associated of the Sequence $f_k\left(t\right)$
$< L \mid p\left(x\right) >$	Action of a Linear Functionals L on a Polynomial $p\left(x\right)$
G	Gregory Integration Formula
GABC	New Integration Formula
AI	**A**rtificial **I**ntelligence
CI	**C**omputational **I**ntelligence
SI	**S**warm **I**ntelligence
SO	**S**elf **O**rganization
ACO	**A**nt **C**olony **O**ptimization
PSO	**P**article **S**warm **O**ptimization
ABC	**A**rtificial **B**ee **C**olony

Introduction

Solving numerical integration is an important question in scientific calculations and engineering.

Consider classical Gregory integration formula [1],

$$\int_0^n f(x)dx = \sum_{k=0}^{n} f(k) + \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(\Delta_1^k f\,(0) + \Delta_{-1}^k f\,(n)\right). \tag{1}$$

And the Gregory integration formula (G) [2],

$$\int_0^n f(x)dx = \sum_{k=0}^{n} f(k) + \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(\Delta_{h_k}^k f\,(0) + \Delta_{-h_k}^k f\,(n)\right). \tag{2}$$

with end corrections where Δ_h is the forward difference operator with step size h. The formula (2) has a sense so $n \geq 1$, In the contrary case an appropriate variable change will permit us to do the integral without no difficulty.

The purpose of this thesis is to use Artificial Bee Colony (ABC) Algorithm in order to improve the Gregory integration formula (G) at the 5^{th} term by minimizing some of their coefficients in the remainder term $(\alpha_3, \alpha_4, \alpha_5)$.

Artificial Bee Colony (ABC) Algorithm is the best example of swarm intelligence algorithms for numerical optimization problem.

ABC Algorithm originally proposed by Karaboga in 2005 [3] and inspired by the foraging behaviour of honey bee swarm on finding food source which is called the nectar, food source position represents a possible solution to the problem to be optimized.

This is a thesis for the degree of Doctorate (PhD) in Mathematics: Numerical Analysis and Optimization. It is organized as follows:

In the first chapter; first of all, we give some of the basic definitions that related to this thesis. Then, we discuss the Theorem of Expansion of formal series $h(t)$ by series delta $f_k(t)$,

$$h\left(t\right)=\sum_{k=0}^{\infty}\frac{<h\left(t\right)\mid p_{k}\left(x\right)>}{k!}f_{k}\left(t\right),$$

where, $p_n\left(x\right)$ is the sequence of polynomials associated for $f_k\left(t\right)$.
Lastly, we give an example of this theorem for determining $p_k\left(x\right)$, where

$$\frac{e^{nt}-1}{t}=\sum_{k=0}^{\infty}\frac{<\frac{e^{nt}-1}{t}\mid p_{k}\left(x\right)>}{k!}\left(e^{h_{k}t}-1\right)^{k},$$

and $p_k\left(x\right)$ is the sequence of polynomials associated for $\left(e^{h_kt}-1\right)^k, k\geqslant 0$ and h_k is non-zero parameters.

In the second chapter; to begin with giving an overview of the Swarm Intelligence. Then, we discuss the main steps of the Artificial Bee Colony (ABC) Algorithm, with giving ABC Algorithm coded using Matlab language in Appendix. A.1. Finally we give simple example for applying this Algorithm for finding the global minimum of a function f in $[0;4]$ interval.

In the last chapter; first, we try to justify the formula (G) by using the umbral methods developed by Rota and his school [4–8], with calculating α_k where

$$\alpha_{k}=<\frac{1}{e^{t}-1}-\frac{1}{t}\mid p_{k}\left(x\right)>,k=0,...,5.$$

And $p_k\left(x\right)$ is the sequence of polynomials associated for $\left(e^{th_k}-1\right)^k$.
Next, we prove that the Gregory formula (G) at the 5^{th} term that can be optimized by minimizing $(\alpha_3,\alpha_4,\alpha_5)$, by using Artificial Bee Colony (ABC) Algorithm. Last, we attempt to give numerical examples for the new formula (GABC).

Finally; we conclude this thesis with a short conclusion and we are going to deal with future work.

Chapter 1

Preliminary

This chapter reviews some of the basic definitions related to this thesis; we start by discussing what the algebra of formal power series is, and what linear functionals are also, we discuss the theorem of expansion of formal series by series delta. Finally, we give an example of application of this theorem. See [1, 4–11].

1.1 The Algebra of Formal Power Series

Define α to be an infinite sequence of complex numbers or reel numbers $\alpha = [\alpha_0, \alpha_1, \alpha_2, ...]$.

By $\boldsymbol{F}$ we denote the class of all such infinite sequences α, and these are **the formal power series**.

Let t denote the particular element $[0, 1, 0, 0, ...]$ of $\boldsymbol{F}$ so that,

$$t^2 = [0, 0, 1, 0, 0, ...],$$

$$t^3 = [0, 0, 0, 1, 0, ...],$$

and in general t^{n-1} is the sequence with zeros in all positions except the n^{th}, where 1 occurs.

We now introduce the notation

$$\sum_{k=0}^{\infty} \alpha_k t^k = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + ...$$

For $\alpha = [\alpha_0, \alpha_1, \alpha_2, ...]$.

If $f(t) \in \boldsymbol{F}, f(t) = [\alpha_0, \alpha_1, \alpha_2, ...]$, say

$$f(t) = \sum_{k=0}^{\infty} \alpha_k t^k,$$

and $g(t) \in \boldsymbol{F}, g(t) = [\beta_0, \beta_1, \beta_2, ...]$, say

$$g(t) = \sum_{k=0}^{\infty} \beta_k t^k.$$

Define addition by

$$\begin{aligned} f(t) + g(t) &= \sum_{k=0}^{\infty} \alpha_k t^k + \sum_{k=0}^{\infty} \beta_k t^k \\ &= [\alpha_0, \alpha_1, \alpha_2, ...] + [\beta_0, \beta_1, \beta_2, ...] \\ &= [\alpha_0 + \beta_0, \alpha_1 + \beta_1, \alpha_2 + \beta_2, ...] \\ &= \sum_{k=0}^{\infty} (\alpha_k + \beta_k) t^k. \end{aligned} \tag{1.1}$$

Define multiplication by

$$\begin{aligned} f(t).g(t) &= \left(\sum_{k=0}^{\infty} \alpha_k t^k \right) . \left(\sum_{k=0}^{\infty} \beta_k t^k \right) \\ &= \left[\alpha_0\beta_0, \alpha_1\beta_0 + \alpha_0\beta_1, \alpha_2\beta_0 + \alpha_1\beta_1 + \alpha_0\beta_2, ..., \sum_{j=0}^{n} \alpha_j \beta_{n-j}, ... \right] \\ &= \sum_{k=0}^{\infty} \left(\sum_{j=0}^{k} \alpha_j \beta_{k-j} \right) t^k. \end{aligned} \tag{1.2}$$

Define multiplication of $\boldsymbol{F}$ with scalar $\lambda \in \mathbb{C}$ by

$$\begin{aligned} \lambda.f(t) &= [\lambda\alpha_0, \lambda\alpha_1, \lambda\alpha_2, ...] \\ &= \sum_{k=0}^{\infty} (\lambda\alpha_k) t^k. \end{aligned} \tag{1.3}$$

It is clear that all those operations are well defined, that is, $f+g, \lambda.f, f.g$, are all in $\boldsymbol{F}$.

The definition of equality is that $f(t) = g(t)$ if and only if $\alpha_j = \beta_j$ for all $j \geqslant 0$.

It is not difficult to establish that the set $\boldsymbol{F}$ is a commutative ring with a unit (with addition (1.1) and multiplication (1.2)), $\boldsymbol{F}$ is a vector space (with addition (1.1) and multiplication with scalar (1.3)) and $\boldsymbol{F}$ is an algebra over the field $\mathbb{K}$ ($\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$).

The zero element and the unit element are

$$\mathbf{0} = [0, 0, 0, ...],$$

and

$$\mathbf{1} = [1, 0, 0, ...].$$

Given any

$$f(t) = \sum_{k=0}^{\infty} \alpha_k t^k,$$

the additive inverse of $f(t)$ is

$$-f(t) = \sum_{k=0}^{\infty} (-\alpha_k) t^k.$$

Theorem 1.1. *Let $f(t) \in \boldsymbol{F}$ such that $f(t) = \sum_{k=0}^{\infty} \alpha_k t^k$, the series $f(t)$ has a multiplicative inverse, denote by $f(t)^{-1}$ or $\dfrac{1}{f(t)}$, if and only if $\alpha_0 \neq 0$, we shall say that $f(t)$ is invertible.*

Proof. $f(t) = \sum_{k=0}^{\infty} \alpha_k t^k$ has a multiplicative inverse in $\boldsymbol{F}$, if and only if it exists $f(t)^{-1} = \sum_{k=0}^{\infty} \gamma_k t^k$ such that $f(t).f(t)^{-1} = \mathbf{1}$, it means if and only if $\alpha_0\gamma_0 = 1$ and $\forall n \in \mathbb{N}^*, \sum_{j=0}^{n} \alpha_j \gamma_{n-j} = 0$.

If $f(t)$ is invertible, as $\alpha_0.\gamma_0 = 1$ so $\alpha_0 \neq 0$.

Reciprocally, if $\alpha_0 \neq 0$, the triangular system of equations

$$\begin{cases} \alpha_0\gamma_0 = 1 \\ \alpha_1\gamma_0 + \alpha_0\gamma_1 = 0 \\ \alpha_2\gamma_0 + \alpha_1\gamma_1 + \alpha_0\gamma_2 = 0 \\ \vdots \\ \alpha_n\gamma_0 + \alpha_{n-1}\gamma_1 + ... + \alpha_0\gamma_n = 0, \end{cases}$$

has a unique solution. □

Example 1.1. *We have* $(1-t).\sum_{k=0}^{\infty} t^k = \boldsymbol{1}$.

Where

$$(1-t) = [1,-1,0,0,...]\,; \sum_{k=0}^{\infty} t^k = [1,1,1,1,...]$$

Because, suppose that $g(t) = \sum_{k=0}^{\infty} \beta_k t^k$ *then,* $(1-t).g(t) = \boldsymbol{1}$ *so,*

$$[1,-1,0,0,...].[\beta_0,\beta_1,\beta_2,...] = [1,0,0,...].$$

We get the following system

$$\begin{cases} \beta_0 = 1 \\ -1\beta_0 + \beta_1 = 0 \\ -\beta_1 + \beta_2 = 0 \\ -\beta_2 + \beta_3 = 0 \\ \vdots \\ -\beta_{n-1} + \beta_n = 0 \\ \vdots \end{cases}$$

So,

$$\begin{cases} \beta_0 = 1 \\ \beta_1 = 1 \\ \beta_2 = 1 \\ \vdots \\ \beta_n = 1 \\ \vdots \end{cases}$$

We get $g(t) = \sum_{k=0}^{\infty} t^k$, *or* $(1-t)^{-1} = \sum_{k=0}^{\infty} t^k$.

1.1.1 The Ordrer of Power Series

Definition 1.2. Let $f(t) \in \boldsymbol{F}$ such that $f(t) = \sum_{k=0}^{\infty} \alpha_k t^k$. The order $O(f(t))$ of a power series $f(t)$ is the smallest k such that $\alpha_k \neq 0$;

$$O(f(t)) = \inf\{k \in \mathbb{N}, \alpha_k \neq 0\}.$$

We take $O(f(t)) = +\infty$ if $f(t) = 0$.

Example 1.2. $O(t^n) = n, n \in \mathbb{N}$.

Theorem 1.3. *If* $f(t), g(t) \in \boldsymbol{F}$ *such that* $f(t) = \sum_{k=0}^{\infty} \alpha_k t^k$ *and* $g(t) = \sum_{k=0}^{\infty} \beta_k t^k$, *then*

1. $O(f(t) + g(t)) \geqslant \min(O(f(t)), O(g(t)))$.
2. $O(f(t).g(t)) = O(f(t)) + O(g(t))$.

Proof. 1. We have

$$f(t) + g(t) = \sum_{k=0}^{\infty} (\alpha_k + \beta_k) t^k.$$

If $k < \min(O(f(t)), O(g(t)))$ then $\alpha_k = \beta_k = 0$ and $\alpha_k + \beta_k = 0$.

So,

$$\inf\{k \in \mathbb{N}, \alpha_k + \beta_k \neq 0\} \geqslant \min(O(f(t)), O(g(t))).$$

We get

$$O\left(f\left(t\right)+g\left(t\right)\right)\geqslant\min\left(O\left(f\left(t\right)\right),O\left(g\left(t\right)\right)\right).$$

2. By definition,

$$f\left(t\right).g\left(t\right)=\sum_{k=0}^{\infty}\left(\sum_{j=0}^{k}\alpha_j\beta_{k-j}\right)t^k.$$

We shows that;

$$\inf\left\{n\in\mathbb{N},\sum_{j=0}^{n}\alpha_j\beta_{n-j}\neq 0\right\}=\inf\left\{k\in\mathbb{N},\alpha_k\neq 0\right\}+\inf\left\{k\in\mathbb{N},\beta_k\neq 0\right\}.$$

- Let $n\in\mathbb{N}$ such that $\sum_{j=0}^{n}\alpha_j\beta_{n-j}\neq 0$ so, it exists $(i;j)\in\mathbb{N}^2$ such that $i+j=n$ and $\alpha_i\beta_j\neq 0$.
  So, $i\geqslant\inf\left\{k\in\mathbb{N},\alpha_k\neq 0\right\}$ and $j\geqslant\inf\left\{k\in\mathbb{N},\beta_k\neq 0\right\}$.
  So,
  $$n=i+j\geqslant\inf\left\{k\in\mathbb{N},\alpha_k\neq 0\right\}+\inf\left\{k\in\mathbb{N},\beta_k\neq 0\right\}.$$
  So,
  $$O\left(f\left(t\right).g\left(t\right)\right)\geqslant O\left(f\left(t\right)\right)+O\left(g\left(t\right)\right).$$

- Now, we demonstrate another inequality; let $n=n_1+n_2$ such that $n_1=\inf\left\{k\in\mathbb{N},\alpha_k\neq 0\right\}$ and $n_2=\inf\left\{k\in\mathbb{N},\beta_k\neq 0\right\}$.
  Look at n^{th} term of the formal series $f\left(t\right).g\left(t\right)$;

$$\begin{aligned}\left(f\left(t\right).g\left(t\right)\right)_n=\alpha_0\beta_{n_1+n_2}+\alpha_1\beta_{n_1+n_2-1}+...+\alpha_{n_1-1}\beta_{n_2+1}\\+\alpha_{n_1}\beta_{n_2}+\alpha_{n_1+1}\beta_{n_2-1}+...+\alpha_{n_1+n_2}\beta_0.\end{aligned}$$

  For all $j<n_1$; we have $\alpha_j=0$ so, $\alpha_j\beta_{n-j}=0$.
  For all $k<n_2$; we have $\beta_k=0$ so, $\alpha_{n-k}\beta_k=0$.
  So, $\left(f\left(t\right).g\left(t\right)\right)_n=\alpha_{n_1}\beta_{n_2}\neq 0$ and

$$n_1+n_2\in\left\{n\in\mathbb{N},\alpha_0\beta_n+\alpha_1\beta_{n-1}+...+\alpha_n\beta_0\neq 0\right\}.$$

So,

$$\inf\left\{n\in\mathbb{N},\sum_{j=0}^{n}\alpha_j\beta_{n-j}\neq 0\right\}\leqslant \inf\left\{k\in\mathbb{N},\alpha_k\neq 0\right\}+\inf\left\{k\in\mathbb{N},\beta_k\neq 0\right\}.$$

So,

$$O\left(f\left(t\right).g\left(t\right)\right)\leqslant O\left(f\left(t\right)\right)+O\left(g\left(t\right)\right).$$

□

1.1.2 Polynomial Algebra

Let $\mathbb{K}$ be a field ($\mathbb{K}=\mathbb{R}$ or $\mathbb{C}$), a polynomials is a formal series such that $p\left(x\right)=\sum_{k\geqslant 0}\alpha_k x^k, \alpha_k\in\mathbb{K}$ and $\alpha_k=0$ for all but a finite number of k.
We next definie $\boldsymbol{P}$ as the set of all polynomials in x over $\mathbb{K}$.
So, $\boldsymbol{P}$ is a sub-algebra of $\boldsymbol{F}$.

Definition 1.4. Let $p\left(x\right)\in\boldsymbol{P}$ such that $p\left(x\right)=\sum_{k\geqslant 0}\alpha_k x^k$. The degree $deg\left(p\left(x\right)\right)$ of the polynomial $p\left(x\right)$ is the largest k such that $\alpha_k\neq 0$.
We take $deg\left(p\left(x\right)\right)=-\infty$ if $p\left(x\right)=0$.
Let $p\left(x\right)$ and $q\left(x\right)$ be polynomials of $\boldsymbol{P}$:

1. $deg\left(p\left(x\right)\right)+deg\left(q\left(x\right)\right)=deg\left(p\left(x\right).q\left(x\right)\right)$.
2. $deg\left(p\left(x\right)+q\left(x\right)\right)\leq\max\left(deg\left(p\left(x\right)\right),deg\left(q\left(x\right)\right)\right)$.

1.1.3 Generating Functions, Conjugate Sequences

Suppose $f_k\left(t\right)$ is a sequence in $\boldsymbol{F}$ such that

$$f_k\left(t\right)=\sum_{n=0}^{\infty}\alpha_{nk}t^n, k\geq 0.$$

Define the generating functions of the sequence $f_k\left(t\right)$ by

$$F\left(x,t\right)=\sum_{k=0}^{\infty}f_k\left(t\right)x^k.$$

Let $M(f_k) = (\alpha_{nk})$ be the infinite matrix associated of the sequence $f_k(t)$, where n is the index of the lines and k is the index of the columns. Then the data of $F(x,t)$ is equivalent to the data of the matrix $M(f_k)$.

The conjugate sequence of $f_k(t)$ is the sequence $f_n^c(t)$ in $\boldsymbol{F}$ such that

$$f_n^c(t) = \sum_{k=0}^{\infty} \alpha_{kn} t^k, n \geqslant 0.$$

In other words, the associated matrix of $f_n^c(t)$ is the transposed matrix $M(f_k)^t$ of $M(f_k)$.

Consider $f_k(t)$ a sequence of $\boldsymbol{F}$ such that

$$f_k(t) = \sum_{n=0}^{\infty} \alpha_{nk} t^n, k \geq 0,$$

that is said to be a sequence **admitting addition**. If corresponding to any integer $N = N(r)$ such that for all $r \geqslant N$;

$$\alpha_{0n} = \alpha_{1n} = \alpha_{2n} = ... = \alpha_{rn} = 0;$$

if this condition is satisfied we also say that $\sum_{k=0}^{\infty} f_k(t)$ is an **admissible sum**.

1.1.4 Compositional Umbral

Suppose $g_k(t)$ is a sequence admitting addition and $f(t) \in \boldsymbol{F}$ sush that $f(t) = \sum_{k=0}^{\infty} \alpha_k t^k$ so, the sequence $\alpha_k g_k(t)$ is admitting addition and the sum

$$S(t) = \sum_{k=0}^{\infty} \alpha_k g_k(t)$$

is a well-defined series in $\boldsymbol{F}$.

We call S is the compositional umbral of f and g, S is also denoted by $f \circ g$.

$$S(t) = (f \circ g)(t) = f(g(t)).$$

In particular, we may take $g_k(t) = g(t)^k$ where $0\,(g\,(t)) \geqslant 1$.
Let $f_k(t)$ be a sequence of $\boldsymbol{F}$, we have

$$I_k \circ f(t) = f_k \circ I(t) = f_k(t), \tag{1.4}$$

where $I_k(t) = t^k, k \geqslant 0$. In other words, the sequence $I_k(t)$ is the unit element for the compositional umbral.
If $I = M(t^k)$ is the identity matrix so, the equality (1.4) is equivalent

$$M(f_k).I = I.M(f_k) = M(f_k).$$

Let $g_k(t)$ and $f_k(t)$ be the sequences admitting addition, $g_k(t)$ is a compositional umbral inverse of $f_k(t)$ if

$$f_k \circ g(t) = g_k \circ f(t) = I_k(t), k \geqslant 0.$$

So, $M(g_k)$ is the inverse matrix of the $M(f_k)$, i.e

$$M(f_k).M(g_k) = M(g_k).M(f_k) = I.$$

We write, $g_k(t) = \overline{f}_k(t)$.
We call any $f_k(t)$ of $\boldsymbol{F}$ with $O\,(f_k(t)) = k$ a delta series.
We call any $p_n(x)$ of $\boldsymbol{P}$ with $deg\,(P_n(x)) = n$ a sequence of polynomials.
Delta series and sequence of polynomials are simple examples of the compositional umbral invertible because $M(f_k)$ and $M(p_n)$ are triangular matrix.

1.2 Linear Functionals

Let $\boldsymbol{P}$ be the algebra of polynomials in a single variable x, with coefficients in a field $\mathbb{K}$, which we often assume to be either the real or the complex field.
Let $\boldsymbol{P^*}$ be the vector space of all linear functional on $\boldsymbol{P}$.
We use the notation $< L \mid p\,(x) >$ for the action of a linear functionals L on a polynomial $p\,(x)$, and we recall that the vector space operations on $\boldsymbol{P^*}$ are defined

by

$$< L + M \mid p(x) >=< L \mid p(x) > + < M \mid p(x) >; \forall L, M \in \boldsymbol{P^*},$$
$$< cL \mid p(x) >= c < L \mid p(x) >; \forall L \in \boldsymbol{P^*}, c \in \mathbb{K}.$$

Since a linear functional is uniquely determined by its action on a basis, L in $\boldsymbol{P^*}$ is uniquely determined by the sequence of constants $< L \mid x^n >$ for $n \geqslant 0$.
The formal power series

$$f(t) = \sum_{k=0}^{\infty} \frac{\alpha_k}{k!} t^k,$$

defines a linear functional on $\boldsymbol{P}$ by setting

$$< f(t) \mid x^n >= \alpha_n, n \geqslant 0. \tag{1.5}$$

Notice that we have used the same notation $f(t)$ for the power series and the linear functional.
This should cause no confusion since if $f(t)$ and $g(t)$ are in $\boldsymbol{F}$, then $f(t) = g(t)$ if and only if $< f(t) \mid x^n >=< g(t) \mid x^n >$ for all $n \geqslant 0$.
In other words, $f(t)$ and $g(t)$ are equal as formal series if and only if they are equal as linear functionals.
As a consequence of (1.5) we have

$$< t^k \mid x^n >= n! \delta_{n,k},$$

such that

$$\delta_{n,k} = \begin{cases} 1, n = k \\ 0, n \neq k, \end{cases}$$

and

$$< \sum_{k=0}^{\infty} \alpha_k t^k \mid x^n >= \sum_{k=0}^{\infty} \alpha_k < t^k \mid x^n >, n \geqslant 0,$$

so, for any $p(x)$ in $\boldsymbol{P}$

$$< \sum_{k=0}^{\infty} \alpha_k t^k \mid p(x) >= \sum_{k=0}^{\infty} \alpha_k < t^k \mid p(x) > .$$

Now any linear functional L in $\boldsymbol{P^*}$ can be represented as a series in $\boldsymbol{F}$. In fact, if

$$f_L(t) = \sum_{k=0}^{\infty} \frac{< L \mid x^k >}{k!} t^k,$$

then

$$\begin{aligned} < f_L(t) \mid x^n > &=< \sum_{k=0}^{\infty} \frac{< L \mid x^k >}{k!} t^k \mid x^n > \\ &= \sum_{k=0}^{\infty} \frac{< L \mid x^k >}{k!} < t^k \mid x^n >, \end{aligned}$$

we have

$$< t^k \mid x^n >= \begin{cases} n!, n = k \\ 0, n \neq k, \end{cases}$$

then

$$< f_L(t) \mid x^n >=< L \mid x^n >,$$

and so as linear functionals

$$f_L = L.$$

Theorem 1.5. *The map $L \mapsto f_L(t)$ is a vector space isomorphism from $\boldsymbol{P^*}$ onto $\boldsymbol{F}$.*

Proof. Since $L = M$ if and only if $< L \mid x^k >=< M \mid x^k >$ for all $k \geqslant 0$, which holds if and only if $f_L(t) = f_M(t)$, the map is bijective. But we also have

$$\begin{aligned} f_{L+M}(t) &= \sum_{k=0}^{\infty} \frac{< L + M \mid x^k >}{k!} t^k \\ &= \sum_{k=0}^{\infty} \frac{< L \mid x^k >}{k!} t^k + \sum_{k=0}^{\infty} \frac{< M \mid x^k >}{k!} t^k \\ &= f_L(t) + f_M(t). \end{aligned}$$

And for $c \in \mathbb{K}$,

$$\begin{aligned} f_{cL}(t) &= \sum_{k=0}^{\infty} \frac{< cL \mid x^k >}{k!} t^k \\ &= c \sum_{k=0}^{\infty} \frac{< L \mid x^k >}{k!} t^k \\ &= c f_L(t). \end{aligned}$$

Thus our map is a vector isomorphism. □

Proposition 1.6. *If $f(t) \in \boldsymbol{F}$, then*

$$f(t) = \sum_{k=0}^{\infty} \frac{< f(t) \mid x^k >}{k!} t^k.$$

Proof. We have

$$\begin{aligned} < \sum_{k=0}^{\infty} \frac{< f(t) \mid x^k >}{k!} t^k \mid x^n > &= \sum_{k=0}^{\infty} \frac{< f(t) \mid x^k >}{k!} < t^k \mid x^n > \\ &=< f(t) \mid x^n > . \end{aligned}$$

□

Proposition 1.7. *If $p(x) \in \boldsymbol{P}$, then*

$$p(x) = \sum_{k=0}^{\infty} \frac{< t^k \mid p(x) >}{k!} x^k.$$

Proof. We have

$$\begin{aligned} < \sum_{k=0}^{\infty} \frac{< t^k \mid p(x) >}{k!} x^k \mid t^n > &= \sum_{k=0}^{\infty} \frac{< t^k \mid p(x) >}{k!} < x^k \mid t^n > \\ &= \sum_{k=0}^{\infty} \frac{< t^k \mid x^n >}{k!} < x^k \mid t^n > \\ &=< x^n \mid t^n >, n \geqslant 0 \\ &=< p(x) \mid t^n > . \end{aligned}$$

□

Example 1.3 (The Evaluation Functional)**.** *For a constant a, the linear functional ξ_a, defined by $< \xi_a \mid p(x) >= p(a)$, is called evaluation at a, we call this linear the evaluation functional.*
Let $y \in \mathbb{R}$, we have

$$\begin{aligned} < e^{yt} \mid x^n > &=< \sum_{k=0}^{\infty} \frac{y^k}{k!} t^k \mid x^n > \\ &= \sum_{k=0}^{\infty} \frac{y^k}{k!} < t^k \mid x^n > \\ &= y^n, n \geqslant 0, \end{aligned}$$

and

$$< e^{yt} \mid p(x) >= p(y),$$

for all $p(x) \in \boldsymbol{P}$.

Example 1.4 (The Forward Difference Functional)**.** *The forward difference functional is the delta functional $e^{yt} - 1$ and*

$$< e^{yt} - 1 \mid p(x) >= p(y) - p(0), y \in \mathbb{R}.$$

Example 1.5. *The functional $f(t)$ that satisfies*

$$< f(t) \mid p(x) >= \int_0^y p(u)\, du,$$

for all polynomial $p(x)$ can be determined from proposition 1.6;

$$\begin{aligned} f(t) &= \sum_{k=0}^{\infty} \frac{< f(t) \mid x^k >}{k!} t^k \\ &= \sum_{k=0}^{\infty} \frac{\int_0^y u^k du}{k!} t^k \\ &= \sum_{k=0}^{\infty} \frac{y^{k+1}}{(k+1)!} t^k \\ &= \frac{e^{yt} - 1}{t}, \end{aligned}$$

so,

$$f(t) = \frac{e^{yt} - 1}{t}.$$

1.3 Expansion of Formal Series by Series Delta

A sequence $g_k(t)$ for which $O(g_k(t)) = k$ forms pseudobasis for $\boldsymbol{F}$. In other words, for each series $f(t)$ there is a unique sequence of constants α_k for which

$$f(t) = \sum_{k=0}^{\infty} \alpha_k g_k(t).$$

In particular, the powers of delta series form a pseudobasis for $\boldsymbol{F}$.
Also, the sequence of polynomials form a pseudobasis for $\boldsymbol{P}$.

Proposition 1.8. *If $O(f_k(t)) = k$, for all $k \geqslant 0$, then*

$$< \sum_{k=0}^{\infty} \alpha_k f_k(t) \mid p(x) > = \sum_{k=0}^{\infty} \alpha_k < f_k(t) \mid p(x) >,$$

for all $p(x)$ in $\boldsymbol{P}$.

Proof. Suppose that $deg(p(x)) = d$, then

$$\begin{aligned} < \sum_{k=0}^{\infty} \alpha_k f_k(t) \mid p(x) > &= < \sum_{k=0}^{d} \alpha_k f_k(t) \mid p(x) > + < \sum_{k=d+1}^{\infty} \alpha_k f_k(t) \mid p(x) > \\ &= < \sum_{k=0}^{d} \alpha_k f_k(t) \mid p(x) > \\ &= \sum_{k=0}^{d} \alpha_k < f_k(t) \mid p(x) > \\ &= \sum_{k=0}^{\infty} \alpha_k < f_k(t) \mid p(x) > . \end{aligned}$$

□

Proposition 1.9. *If $O(f_k(t)) = k$ ($f_k(t)$ is a delta series), for all $k \geqslant 0$ and if*

$$< f_k(t) \mid p(x) > = < f_k(t) \mid q(x) >,$$

for all k, *then* $p(x) = q(x)$.

Proof. Since the sequence $f_k(t)$, forms a pseudobasis for $\boldsymbol{F}$, for all $n \geqslant 0$ there exist constants $\alpha_{n,k}$ for which

$$t^n = \sum_{k=0}^{\infty} \alpha_{n,k} f_k(t).$$

Thus

$$\begin{aligned} < t^n \mid p(x) > &= < \sum_{k=0}^{\infty} \alpha_{n,k} f_k(t) \mid p(x) > \\ &= \sum_{k=0}^{\infty} \alpha_{n,k} < f_k(t) \mid p(x) > \\ &= \sum_{k=0}^{\infty} \alpha_{n,k} < f_k(t) \mid q(x) > \\ &= < \sum_{k=0}^{\infty} \alpha_{n,k} f_k(t) \mid q(x) > \\ &= < t^n \mid q(x) >, \end{aligned}$$

so proposition 1.7 shows that

$$\begin{aligned} p(x) &= \sum_{k=0}^{\infty} \frac{< t^k \mid p(x) >}{k!} x^k \\ &= \sum_{k=0}^{\infty} \frac{< t^k \mid q(x) >}{k!} x^k \\ &= q(x). \end{aligned}$$

□

Proposition 1.10. *If* $deg(p_k(x)) = k$, *for all* $k \geqslant 0$ *and if*

$$< f(t) \mid p_k(x) > = < g(t) \mid p_k(x) >,$$

for all k, *then* $f(t) = g(t)$.

Proof. For each $n \geqslant 0$ there exist constants $\alpha_{n,k}$ for which

$$x^n = \sum_{k=0}^{N} \alpha_{n,k} p_k(x).$$

Thus

$$\begin{aligned} < f(t) \mid x^n > &= \sum_{k=0}^{N} \alpha_{n,k} < f(t) \mid p_k(x) > \\ &= \sum_{k=0}^{N} \alpha_{n,k} < g(t) \mid p_k(x) > \\ &=< g(t) \mid x^n >, \end{aligned}$$

so proposition 1.6 shows that $f(t) = g(t)$. □

By a sequence $p_n(x)$ in $\boldsymbol{P}$ we shall always imply that $deg(p_n(x)) = n$.

Theorem 1.11. *Let $f_k(t)$ be a delta series. Then exists a unique sequence $p_n(x)$ of polynomials satisfying the orthogonality conditions*

$$< f_k(t) \mid p_n(x) >= n! \delta_{n,k}, \tag{1.6}$$

for all $n, k \geqslant 0$.

Proof. The uniqueness follows from proposition 1.9.
If $< f_k(t) \mid p_n(x) >=< f_k(t) \mid q_n(x) >$ then $p_n(x) = q_n(x)$.
For the existence, suppose

$$p_n(x) = \sum_{j=0}^{n} \alpha_{n,j} x^j,$$

where $\alpha_{n,n} \neq 0$, and

$$f_k(t) = \sum_{i=k}^{\infty} \beta_{k,i} t^i,$$

where $\beta_{n,n} \neq 0$, then (1.6) becomes

$$\begin{aligned} n!\delta_{n,k} &= < \sum_{i=k}^{\infty} \beta_{k,i} t^i \mid \sum_{j=0}^{n} \alpha_{n,j} x^j > \\ &= \sum_{i=k}^{\infty} \sum_{j=0}^{n} \beta_{k,i} \alpha_{n,j} < t^i \mid x^j >, \end{aligned}$$

and we have $< t^i \mid x^j >= i!$ if $i = j$,

so,

$$n!\delta_{n,k} = \sum_{i=k}^{n} \beta_{k,i} \alpha_{n,i} i!.$$

Taking $k = n$, one obtain $n! = \beta_{n,n}\alpha_{n,n} n!$ so,

$$\alpha_{n,n} = \frac{1}{\beta_{n,n}}.$$

Taking $k = n - 1$,

$$\begin{aligned} n!\delta_{n,n-1} &= \sum_{i=n-1}^{n} \beta_{n-1,i} \alpha_{n,i} i! \\ 0 &= \beta_{n-1,n-1}\alpha_{n,n-1} (n-1)! + \beta_{n-1,n}\alpha_{n,n} n!, \end{aligned}$$

so,

$$\alpha_{n,n-1} = -\frac{n\beta_{n-1,n}\alpha_{n,n}}{\beta_{n-1,n-1}}$$

By successively taking $k = n, n-1, ..., 0$. We obtain a triangular system of equations that can be solved for $\alpha_{n,k}$. □

Definition 1.12. We say that the sequence $p_n(x)$ in theorem 1.11 is the sequence of polynomials associated for $f_k(t)$.

Theorem 1.13 (Expansion Theorem)**.** *Let $f_k(t)$ be a delta series. Then for any $h(t)$ in $\boldsymbol{F}$*

$$h(t) = \sum_{k=0}^{\infty} \frac{< h(t) \mid p_k(x) >}{k!} f_k(t).$$

Proof. From proposition 1.8 we have

$$\begin{aligned} < \sum_{k=0}^{\infty} \frac{< h(t) \mid p_k(x) >}{k!} f_k(t) \mid p_n(x) > &= \sum_{k=0}^{\infty} \frac{< h(t) \mid p_k(x) >}{k!} < f_k(t) \mid p_n(x) > \\ &= \frac{< h(t) \mid p_n(x) >}{n!} n! \\ &= < h(t) \mid p_n(x) > . \end{aligned}$$

From proposition 1.10 we have

$$h(t) = \sum_{k=0}^{\infty} \frac{< h(t) \mid p_k(x) >}{k!} f_k(t) .$$

□

Corollary 1.14. *Let $f_k(t)$ be a delta series and let $p_n(x)$ be the sequence of polynomials associated for $f_k(t)$. then*

$$p_n(x) = \overline{f_n^c}(x) .$$

Proof. From theorem 1.13 , for $a \in \mathbb{R}$ we have

$$\begin{aligned} e^{at} &= \sum_{n=0}^{\infty} \frac{< e^{at} \mid p_n(x) >}{n!} f_n(t) \\ &= \sum_{n=0}^{\infty} \frac{p_n(a)}{n!} f_n(t) , \end{aligned}$$

so,

$$\sum_{k=0}^{\infty} \frac{a^k}{k!} t^k = \sum_{n=0}^{\infty} \frac{p_n(a)}{n!} f_n(t) ,$$

so,

$$\sum_{k=0}^{\infty} \frac{a^k}{k!} \overline{f_k}(t) = \sum_{n=0}^{\infty} \frac{p_n(a)}{n!} t^n ,$$

so,

$$\sum_{n=0}^{\infty} \frac{\overline{f_n^c}(a)}{k!} t^n = \sum_{n=0}^{\infty} \frac{p_n(a)}{n!} t^n ,$$

we get,

$$\overline{f_n^c}(a) = p_n(a) ,$$

so,

$$p_n(x) = \overline{f_n^c}(x).$$

In other words,

$$M(p_n) = M\left(\overline{f_n^c}\right).$$

□

1.4 Illustration

From the Expansion Theorem 1.13, the functional $f(t) = \frac{e^{nt}-1}{t}$ can be developed by using the delta series

$$f_k(t) = \left(e^{h_k t} - 1\right)^k, k \geqslant 0,$$

where h_k non-zero parameters.

We have

$$\frac{e^{nt}-1}{t} = \sum_{k=0}^{\infty} \frac{\alpha_k}{k!} \left(e^{h_k t} - 1\right)^k,$$

where

$$\alpha_k = < \frac{e^{nt}-1}{t} \mid p_k(x) >,$$

$p_k(x)$ is the sequence of polynomials associated for $f_k(t)$.

$p_k(x)$ can be determined by using corollary 1.14;

$$M(p_n) = M\left(\overline{f_n^c}\right).$$

We have,

$$f_1(t) = \left(e^{h_1 t} - 1\right) = \sum_{n=1}^{\infty} \frac{h_1^n}{n!} t^n.$$

$$f_2(t) = \left(e^{h_2 t} - 1\right)^2 = \left(\sum_{n=1}^{\infty} \frac{h_2^n}{n!} t^n\right)^2.$$

$$f_3(t) = \left(e^{h_3 t} - 1\right)^3 = \left(\sum_{n=1}^{\infty} \frac{h_3^n}{n!} t^n\right)^3.$$

$$\vdots$$

$$f_k(t) = \left(e^{h_k t} - 1\right)^k = \left(\sum_{n=1}^{\infty} \frac{h_k^n}{n!} t^n\right)^k.$$

Suppose $\dfrac{h_k^n}{n!} = C_k^n$, for $k, n = 1, 2, \ldots$, so

$$M(f_k) = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & \cdots \\ 0 & C_1^1 & 0 & 0 & \cdots & 0 & \cdots \\ 0 & C_1^2 & C_2^1.C_2^1 & 0 & \cdots & 0 & \cdots \\ 0 & C_1^3 & C_2^2.C_2^2 & C_3^1.C_3^1.C_3^1 & \cdots & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & C_1^k & C_2^{k-1}.C_2^{k-1} & C_3^{k-2}.C_3^{k-2}.C_3^{k-2} & \cdots & C_k^1.C_k^1.C_k^1.\ldots.C_k^1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Which can be written

$$M(f_k) = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & \cdots \\ 0 & C_1^1 & 0 & 0 & \cdots & 0 & \cdots \\ 0 & C_1^2 & C_2^1(2) & 0 & \cdots & 0 & \cdots \\ 0 & C_1^3 & C_2^2(2) & C_3^1(3) & \cdots & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & C_1^k & C_2^{k-1}(2) & C_3^{k-2}(3) & \cdots & C_k^1(k) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Where $C_k^n(i) = C_k^n.C_k^n.\ldots.C_k^n$, i times. So,

$$M(f_k^c) = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & \cdots \\ 0 & C_1^1 & C_1^2 & C_1^3 & \cdots & C_1^k & \cdots \\ 0 & 0 & C_2^1(2) & C_2^2(2) & \cdots & C_2^{k-1}(2) & \cdots \\ 0 & 0 & 0 & C_3^1(3) & \cdots & C_3^{k-2}(3) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & C_k^1(k) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Thus,

- $M\left(f_0^c\right)=1$. So, $M\left(\overline{f_0^c}\right)=1$. And $p_0\left(x\right)=1$, and consequently

$$\alpha_0=\int_0^n dx=n.$$

- $$M\left(f_1^c\right)=\begin{pmatrix}1 & 0\\ 0 & C_1^1\end{pmatrix}=\begin{pmatrix}1 & 0\\ 0 & h_1\end{pmatrix}$$

  So,

$$M\left(\overline{f_1^c}\right)=\begin{pmatrix}1 & 0\\ 0 & \dfrac{1}{h_1}\end{pmatrix}$$

  And $p_1\left(x\right)=\dfrac{1}{h_1}x$, and consequently

$$\begin{aligned}\alpha_1&=\int_0^n\frac{1}{h_1}xdx\\ &=\frac{n^2}{2h_1}.\end{aligned}$$

- $$M\left(f_2^c\right)=\begin{pmatrix}1 & 0 & 0\\ 0 & C_1^1 & C_1^2\\ 0 & 0 & C_2^1\left(2\right)\end{pmatrix}=\begin{pmatrix}1 & 0 & 0\\ 0 & h_1 & \dfrac{h_1^2}{2!}\\ 0 & 0 & h_2^2\end{pmatrix}$$

  So,

$$M\left(\overline{f_2^c}\right)=\begin{pmatrix}1 & 0 & 0\\ 0 & \dfrac{1}{h_1} & \dfrac{-h_1}{2h_2^2}\\ 0 & 0 & \dfrac{1}{h_2^2}\end{pmatrix}$$

  And $p_2\left(x\right)=\dfrac{-h_1}{2h_2^2}x+\dfrac{1}{h_2^2}x^2$, and consequently

$$\begin{aligned}\alpha_2&=\int_0^n\left(\frac{-h_1}{2h_2^2}x+\frac{1}{h_2^2}x^2\right)dx\\ &=\frac{-h_1}{4h_2^2}n^2+\frac{1}{3h_2^2}n^3.\end{aligned}$$

In the same way, we calculate $\alpha_3,\alpha_4\cdots$

Chapter 2

Artificial Bee Colony (ABC)

2.1 Artificial Intelligence

Artificial Intelligence (AI) is one of the oldest and the best known research fields [12]. It attempts to understand intelligent entities. Thus, one reason to study it is to learn more about ourselves. But unlike philosophy and psychology, which are also concerned with intelligence, AI strives to build intelligent entities as well as understand them. Another reason to study AI is that these constructed intelligent entities are interesting and useful in their own right. AI has produced many significant and impressive products even at this early stage in its development. Although no one can predict the future in detail, it is clear that computers with human-level intelligence (or better) would have a huge impact on our everyday lives and on the future course of civilization [13].

AI addresses one of the ultimate puzzles. How is it possible for a slow, tiny brain, whether biological or electronic, to perceive, understand, predict, and manipulate a world far larger and more complicated than itself? How do we go about making something with those properties? These are hard questions, but unlike the search for faster-than-light travel or an antigravity device, the researcher in AI has solid evidence that the quest is possible. All the researcher has to do is look in the mirror to see an example of an intelligent system. AI is one of the newest disciplines, John McCarthy, who coined the term Artificial Intelligence in 1956 at the Dartmouth conference, defined it as “the science and engineering of making

intelligent machines" [12, 14, 15], also other textbooks [13] define this discipline as:

"exciting new effort to make computers think ... machines with minds, in the full and literal sense" (Haugeland, 1985).

"The automation of activities that we associate with human thinking, activities such as decision-making, problem solving, learning ..." (Bellman, 1978).

"The study of mental faculties through the use of computational models" (Charniak and McDermott, 1985).

"The study of the computations that make it possible to perceive, reason, and act" (Winston, 1992).

"The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil, 1990).

"The study of how to make computers do things at which, at the moment, people are better" (Rich and Knight, 1991).

"A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes" (Schalkoff, 1990).

"The branch of computer science that is concerned with the automation of intelligent behavior" (Luger and Stubblefield, 1993).

Intelligent behavior is shown in many ways, including: perceiving one's environment, learning and understanding from experience, knowledge applying successfully in new situations, communicating with others, and more like, acting in complex environments, reasoning to solve problems and discover hidden knowledge, thinking abstractly and using analogies, creativity, ingenuity, expressive-ness, curiosity.

The general goals of AI can be summarized as follows: replicate human intelligence (still a distant goal), solve knowledge intensive tasks, Make an intelligent connection between perception and action and enhance human-human, human-computer and computer to computer interaction/communication [14].

Computational Intelligence (CI) is a sub-branch of AI and a fairly new research area and commonly referred to as AI [16]. It is defined as the study of the design of intelligent agents where an intelligent agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment

through effectors [13]. For example, a human agent has eyes, ears, and other organs for sensors, and hands, legs, mouth, and other body parts for effectors. CI includes a set of nature-inspired computational methodologies and approaches to address complex problems of the real world applications. There is a clear difference between them. For example, CI uses subsymbolic knowledge processing whereas classical AI uses symbolic approaches [17].

2.2 What is Swarm Intelligence?

2.2.1 Main Idea

Suppose that you with a group of friends are on a treasure finding mission. You have knowledge of the approximate area of the treasure, but do not know exactly where it is located. Each one in the group has a metal detector and can communicate the signal and current position to the nearest neighbors. Each person therefore knows whether one of his neighbors is nearer to the treasure than he is. The person who found the treasure getting a higher reward than all others, and the rest being rewarded based on distance from the treasure at the time when the first one finds the treasure. By using the information that you get from your neighboring friends, and acting upon it, you increase your chances to find the treasure than if you were on you own.

This is one of the examples of the benefits of cooperation in situations where you do not have global knowledge of an environment. the interact of the individuals with the group can be helpful to solve the global objective by exchanging locally available information.

In loose terms, the group can be referred to as a swarm. Formally, a swarm can be defined as a group of agents that communicate with each other, by acting on their local environment [15], such as bird flocks (See figure 2.3).

Insects that live in colonies, ants, bees, wasps, and termites, have fascinated naturalists as well as poets for many years. “What is it that governs here? What is it that issues orders, foresees the future, elaborates plans, and preserves equilibrium?” wrote Maeterlinck [18].

FIGURE 2.1: Insects colonies preserves equilibrium

These, indeed, are puzzling questions. Every single insect in a social insect colony seems to have its own agenda, and yet an insect colony looks so organized. The seamless integration of all individual activities does not seem to require any supervisor, such as ant colonies (See figure 2.2).

FIGURE 2.2: Ant colonies

FIGURE 2.3: Bird flocks

This model of the collective behavior of social swarms in nature is called **Swarm Intelligence (SI)**. SI can therefore be defined as a relatively new branch of Computational intelligence that is a term given by Eric Bonabeau in his book " Swarm Intelligence: From Natural to Artificial " [18], says this: " Swarm Intelligence is

the emergent collective intelligence of groups of simple agents (hardware or software) ". In other words " Swarm Intelligence is any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies " [3].

2.2.2 Fundamentals of SI in Social Insects

There are two fundamentals concepts that are very important and necessary to obtain swarm intelligent behavior, these two fundamentals of SI are discussed bellow:

- **Division of labor**: The division of labor is the phenomenon of performing different simultaneous tasks by specialized individuals, In a social insect colony, a worker usually does not perform all tasks, but rather specializes in a set of tasks, according to its morphology, age, or chance because simultaneous task performance by groups of cooperating specialized individuals is believed to be more efficient than the sequential task performance by unspecialized individuals [3, 18], there are a further division of labor between workers, which may take three possibly, basic forms:
  - **Temporal polyethism**: temporal polyethism is a mechanism of division of labor that are allocated among workers based on their age; individuals of the same age tend to accomplish common sets of tasks. Individuals in the same age class form an *age caste*.
  - **Worker polymorphism**: one of the most well knows mechanisms of division of labor, where workers within a colony have morphological differences, workers that differ by their morphologies are said to belong to different *morphological or physical castes*. Workers in different morphological castes tend to accomplish different tasks.
  - **Individual variability**: are the differences that vary from one caste to another in the task performance, but at *behavioral castes* to describe groups of individuals that perform the same set of tasks within a given period.

Division of labor also enables the swarm to respond to changed conditions in the search space.

- **Self organization (SO)**: The term self-organization describes " the process by which individuals organize their communal behavior to create global order by interactions amongst themselves rather than through external intervention or instruction." [19], SO relies on four basic properties [18]:
  - **Positive feedback** is a simple behavioral " rules of thumb " that promote the creation of structures.
  - **Negative feedback** counterbalances positive feedback and helps to stabilize the collective pattern.
  - **Fluctuations** such as random walks, errors, random, task-switching.
  - **Multiple interactions** is the different activities that is resulted from a minimal density of mutually tolerant individuals.

2.2.3 Swarm Intelligence Models

In recent years, SI becomes more and more attractive for the researchers, who work in the related research field, The objective of SI models is to model the simple behavior of social insect colonies or other animal societies that can be used to solve complex problems, mostly optimization problems.
Examples of SI models are:

- **Ant Colony Optimization (ACO)**: the first example of a successful swarm intelligence model is Ant Colony Optimization (ACO), which was introduced by M. Dorigo and al [20, 21], that is the very simple pheromone trail following behavior of real ants that helps find the shortest route between their nest and a food source, where each ant perceives pheromone concentrations in its local environment and acts by probabilistically selecting the direction with the highest pheromone concentration.
- **Particle Swarm Optimization (PSO)**: The second example of a successful swarm intelligence model is Particle Swarm Optimization (PSO), which was

introduced by Russell Eberhart, an electrical engineer, and James Kennedy, a social psychologist, in 1995 [22, 23], the term 'particle' means any natural agent that describes the swarm behavior. The PSO model based upon a mathematical description of the social behavior of birds flocking or fish schooling.

- **Artificial Bee Colony (ABC)**: The third best example of a successful swarm intelligence model is ABC, the ABC algorithm was firstly introduced by Karaboga and al. [3, 24–27] for numerical optimization problems based on the foraging behavior of a honey bee swarm. In the next section, we are going to talk about this algorithm in more detail.

2.3 Artificial Bee Colony (ABC)

By now, there are many intelligent optimization techniques is proposed by researchers in the literature, the known examples are PSO inspired by the social behavior of flocks of birds and ACO inspired by the foraging behavior of ants. They performs well in most cases but there still exist some problems it cannot solve very well.
In this section we discuss ABC algorithm. It is considered one of the most recently proposed swarm intelligence algorithms for numerical optimization problems. ABC algorithm is simple and very flexible when compared to other algorithms and there are many possible applications of ABC.
ABC algorithm is proposed by Karaboga in 2005 [3] and the performance of ABC is analyzed in 2007 [26]. It depends on the foraging behavior of honey bee swarm on finding food source (nectar).

2.3.1 Description of The Foraging Behavior of Reel Honey Bees

In nature, A colony of honey bees can exploit a large number of food sources in big fields and they can fly up to 11 km to exploit food sources [28]. Tereshko developed a minimal model of foraging behavior of a honey bee colony based on

reaction–diffusion equations [29, 30]. This model consists of two essential components as below:

- **Food sources**: The value of a food source depends on different parameters such as its proximity to the nest, richness of energy and ease of extracting this energy. For the sake of simplicity, the "profitability" of a food source can be represented with a single quantity [3, 28].
- **Foragers**:
  - **Employed foragers**: An employed forager is employed at a specific food source which she is currently exploiting. She carries information about this specific source and shares it with other bees waiting in the hive. The information includes the distance, the direction and the profitability of the food source [24].
  - **Unemployed foragers**: A forager bee that looks for a food source to exploit is called unemployed. It can be either a scout who searches the environment randomly or an onlooker who tries to find a food source by means of the information given by the employed bee. The mean number of scouts is about 5–10% [24].

(See figure 2.4)

FIGURE 2.4: Honey bees and Food sources

The sharing of information between bees is the most important occurrence in the foraging behavior of honey bees where it is done on part of the hive that called dancing area or dance floor. Foragers bees discover the environment out the nest

for finding new food sources when the bees have found it, they go to the dance floor to communicate information about the quality of the food source, the distance of the source from the hive and the direction of the source to the other members of the colony with no difficulty by repeating specific movements in such a way as to attract the other bees's attention, known as the **waggle dance** [31].

Let us examine figure 2.5 and figure 2.6 for understand waggle dance.

The waggle dance path has a figure of eight shape. Initially the bees vibrates its wing muscles which produces a loud buzz and runs in a straight line the direction which is related to the vertical on the hive and indicates the direction of the food source relative to the sun in the field, for example; if the dances is in the straight line in hive, it means the source of nectar exactly in the direction of the sun, if the source were located in the opposite direction, the bee makes lines in that direction, see figure 2.5. The bees then circles back, alternating a left and a right return path. The speed/duration of the dance indicates the distance to the food source; the frequency of the waggles in the dance and buzzing convey the quality of the source [31], see figure 2.6.

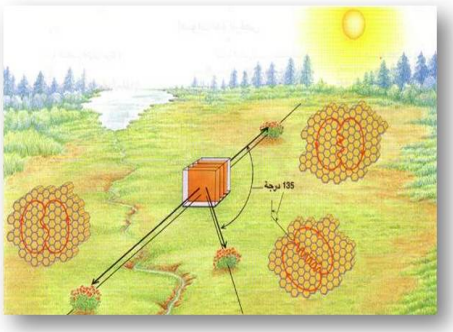


FIGURE 2.5: Orientation of waggle dance with respect to the food source, hive and sun

FIGURE 2.6: waggle dance of honey bees

But there is a question; The bees explain the direction according to the position of the sun but the sun is certainly moving, every four minutes the sun moves 1° to the west which would to expect that bees would make the errors. However, observation are shown that the bees take the count's sun movement as the foragers bees give the direction every four minute, the angle describe moves 1° to the west [32].

In the case of honey bees, the basic properties on which self organization relies are as follows [3]:

- **Positive feedback**: As the nectar amount of food sources increases, the number of onlookers visiting them increases, too.
- **Negative feedback**: The exploration process of a food source abandoned by bees is stopped.
- **Fluctuations**: The scouts carry out a random search process for discovering new food sources.
- **Multiple interactions**: Bees share their information about food source positions with their nest mates on the dance area.

2.3.2 Studies on ABC Optimizations

In table 2.1, we are going to give a general look about the different studies that were conducted on this algorithm:

The success of the ABC algorithm has motivated researchers to extend the use of this algorithm to other areas, for more examples see [12].

2.3.3 Artificial Bee Colony (ABC) Algorithm Optimization

The ABC algorithm is developed by inspecting the behaviors of the real bees on finding food source, which is called the **nectar**, and sharing the information of food source to the other bees in the nest. In this algorithm colony bees are classified into three types with certain responsibilities:
employed bees, onlooker bees, and scout bees.
The main steps of the algorithm are given below [3]:

```
Send the scouts onto the initial food sources.
REPEAT

  Move the employed bees onto the food sources and determine their nectar amounts.
```

TABLE 2.1: The different studies that were conducted on ABC.

Authors	Years	Description
Karaboga	2005 [3]	Invention of ABC, based on the intelligent behaviour honey bee swarm
Basturk and Karaboga	2006	The first conference paper introducing ABC
Karaboga and Basturk	2007 [25]	The first journal article describing ABC and compared the performance of the ABC with that of GA, PSO,...
Karaboga and Basturk	2008 [26]	The second article presenting a performance evaluation of ABC.
	2009 [33]	A public domain web-site
Karaboga and Akay	2009 [24]	Presented a comparative study of ABC in which ABC is used for optimizing a large set of numerical test functions and its results were compared with GA, PSO,...
Karaboga and Akay	2009 [27]	Compared the performance of ABC wich harmony search and bees algorithms on numerical optimization.
Mala et .al	2009 [34]	Applied ABC for test suite optimization and compared it with ACO and concluded that ABC based approach has sereval advantages over ACO.

```
  Move the onlooker bees onto the food sources and determine their nectar amounts.
  Move the scouts into the search area for discovering new food sources.
  Memorize the best food source found so far.

UNTIL (termination condition satisfied)
```

A food source position represents a possible solution to the problem to be optimized. The amount of nectar of a food source corresponds to the **fitness, fit,** (quality) of the solution represented by that food source.
Firstly, each scout bee that finds a food source location saves the current location in her memory and becomes an employed bee.

In the employed bees phase, employed bees are the bees that have already been assigned to a food source. Each of them saves the food source position and selects another food source in her neighbor and chooses out of two the one that

has a better nectar. Then they return to the hive and start to dance based on the quality of the nectar of their associated food source.

In the onlooker bees phase, an onlooker bees watches the dance of employed bees at the hive and selects an employed bee based on the dances observed so that the probability of choosing an employed bee is proportional to the nectar quality of that employed bee. Then the onlooker bee receives the information of the chosen employed bee associated with food source (the food source position and its nectar quality) from her and becomes an employed bee associated with that food source. Since then, the new employed bee (former onlooker) performs the same act as the employed bee in the previous phase; that is, it searches for a new food source in the neighbor of her associated food source for higher nectar quality and saves the best food source and its nectar to her memory.

In the scout bees phase, scout bees are free bees responsible for finding new food sources and evaluating their nectar. As soon as a scout bee finds a food source, she turns into an employed bee. In this phase, one of the employed bees is selected and classified as the scout bee. The selection is controlled by a control parameter called (Limit). If a solution representing a food source is not improved by a predetermined number of trials, then that food source is abandoned by its employed bee and the employed bee is converted to a scout. The number of trials for releasing a food source is equal to the value of Limit which is an important control parameter of ABC.

The algorithm ABC assumes that there is only one employed bee for every food source; thus the number of employed bees is equal to the number of food sources around the hive. The employed bee whose food source has been exhausted by the bees becomes a scout [3, 35–37].

Now, we use ABC algorithm to optimize the following problem:

$$\min_{Y} F(Y)$$

Where
$Y_i = (y_{1,i}, y_{2,i}, ..., y_{j,i}, ..., y_{d,i}), y_{j,i} \in [y_{j,\min}; y_{j,\max}], i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, ..., d\}$,
n is the number of employed bees, d is the dimension of the solution space.

In the ABC algorithm, The percentages of onlooker bees and employed bees were 50% of the colony and the number of scout bees was selected to be one.
Parameters adopted for the ABC algorithm are given in table 2.2:

TABLE 2.2: Parameters adopted for the ABC algorithm

Max cycle	
Swarm size	Number of colony bees
Limit	Number of onlooker bees ∗d
Number of onlookers bees	50% of the swarm
Number of employed bees	50% of the swarm
Number of scouts	1

The general algorithmic structure of the ABC optimization approach is given as follows:

Step 1: (Initialization)

- The initial swarm $Y_i = (y_{1,i}, y_{2,i}, ..., y_{j,i}, ..., y_{d,i})$ by using equation

$$y_{j,i} = y_{j,\min} + rand\,[0; 1] * (y_{j,\max} - y_{j,\min}) \tag{2.1}$$

Where
$i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, ..., d\}$.
n is the number of employed bees, d is the dimension of the problem.
$y_{j,i}$ is parameter to be optimized for the i^{th} employed bee on the dimension j of the d-dimensional solution space.
$y_{j,\max}$ and $y_{j,\min}$ are the upper and lower for $y_{j,i}$ respectively.

- Calculate $F(Y_i)$ by the equation specific for the problem.
- Calculate the fitness value (fit_i) of each food source by using equation

$$fit_i = \begin{cases} \frac{1}{1+F(Y_i)} & \text{if} \quad (F\,(Y_i) \geq 0) \\ 1 + abs(F\,(Y_i)) & \text{if} \quad (F\,(Y_i) < 0) \end{cases} \tag{2.2}$$

We are going to give the fitness function coded using MATLAB language in Appendix. A.2.

- Reset the abandonment counter.

Step 2: (Move the employed bees) For each employed bee:

- Select a neighbor employed bee randomly.
- Calculate the new solution $y^*_{j,i}$ by using equation

$$y^*_{j,i} = y_{j,i} + rand\,[-1;1] * (y_{j,i} - y_{j,k}) \tag{2.3}$$

Where
$i \neq k; i, k \in \{1, 2, ..., n\}$.
j, k are selected randomly.
$y_{j,k}$ is a neighbor bee of $y_{j,i}$.

- Calculate $F(Y^*_i)$ by the equation specific for the problem.
- Calculate the fitness value (fit_i) of each food source by using equation (2.2).
- If the fitness value of the new solution is better than the fitness value of the old solution then replace the old solution with new one and reset the abandonment counter of the new solution, else increase the abandonment counter of the old solution by 1.

Step 3: (Move the onlooker bees) For each onlooker bee:

- Select an employed bee as neighbor randomly.
- An onlooker bee selects a food source by evaluating the information received from all of the employed bees based on the following probability p_i

$$p_i = \frac{fit_i}{\sum_{j=1}^{n} fit_j} \tag{2.4}$$

- Improve the solution of the employed bee by using equation (2.3) and the neighbor.
- Calculate $F(Y^*_i)$ by the equation specific for the problem.

- Calculate the fitness value (fit_i) of each food source by using equation (2.2).
- If the fitness value of the new solution is better than the fitness value of the old solution then replace the old solution with new one and reset the abandonment counter of the new solution, else increase the abandonment counter of the old solution by 1.

Step 4: (Move the scout bees)
- Fixe the abandonment counter H with the highest content.
- **If** the content of the counter H is higher than the predefined limit then reset the counter H and by using equation (2.1) generate a new solution for the employed bee to which the counter H belongs. Else continue.

Step 5: If a termination condition is met, the process is stopped and the best food source is reported, otherwise the algorithm returns to Step 2.

We are going to give ABC algorithm coded using MATLAB language in Appendix. A.1.

2.3.4 Simulation Studies

In the simulation studies, Artificial Bee Colony (ABC) Algorithm was applied for finding the global minimum of the function f in $[0; 4]$ interval,

$$f(x) = x^5 - 10x^4 + 35x^3 - 50x^2 + 24x$$

Parameters adopted for the ABC algorithm are given in table 2.3:

TABLE 2.3: Parameters adopted for the ABC algorithm

Max cycle	20
Swarm size	10
Limit	5
Number of onlookers bees	5
Number of employed bees	4
Number of scouts	1

Initialization: • Number of food sources are initialized randomly and fitness of each food source is computed. See figure 2.7, table 2.4.

The complete MATLAB code for figure 2.7 is listed in Appendix. A.3.

TABLE 2.4: Initialize the food source positions.

Foods	**Objval**	**Fitness**
0.8178	1.2234	0.4498
3.8992	-1.9454	2.9454
0.1582	2.6783	0.2719
1.9672	-0.1312	1.1312
2.5637	1.4161	0.4139

- The best food source is memorized by the bee:
  GlogalMin $= (3.8992; -1.9454)$**.**

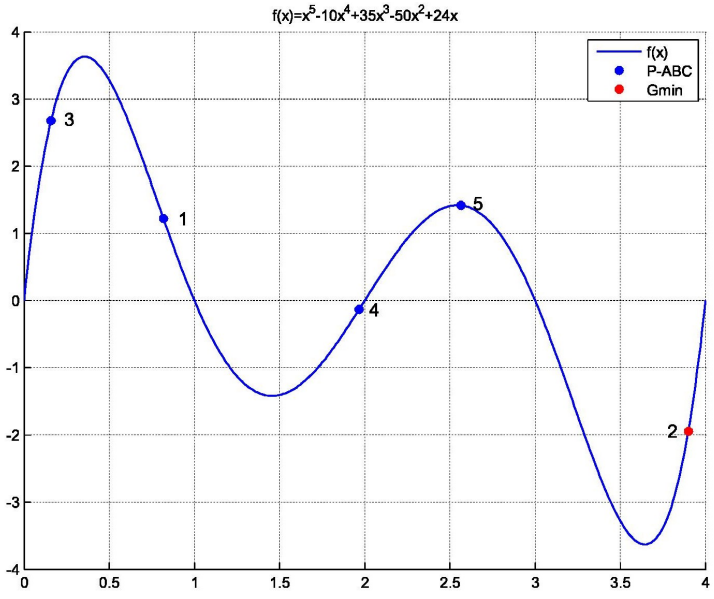
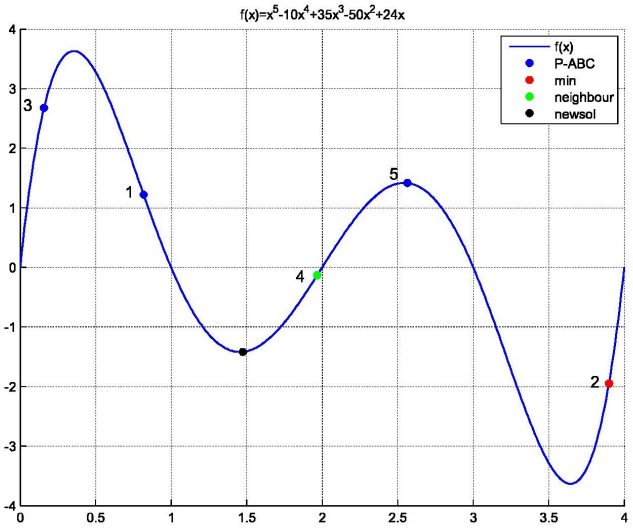


FIGURE 2.7: Initialize the food source positions

Cycle 01**:** ***Employed Bee 01***, see figure 2.8

- Neighboor $= 4$
- New-sol $=$ Food(1)$+$(Food(1)$-$Food(4))$\times$rand
- New-sol $= 1.4739$, New-objval $= -1.4166$
- Fitness(New-sol) $= 2.4166$
- As Fitness(New-sol) $>$ Fitness(01), **So change**

- Trial=(0 0 0 0 0)



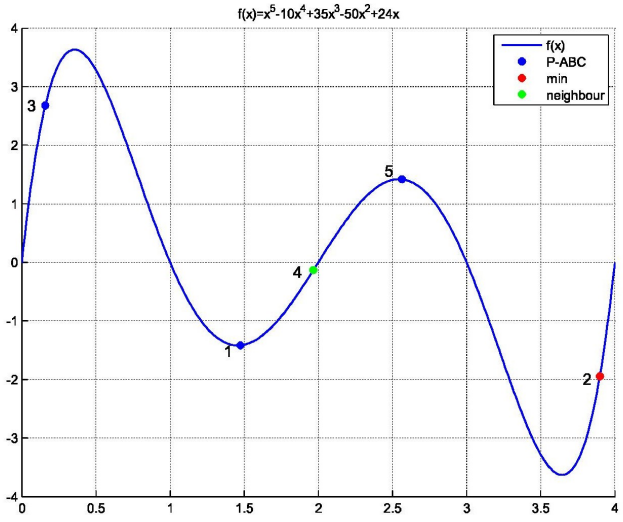


FIGURE 2.8: Food source positions of employed bee 01.

Employed Bee 02, see figure 2.9

- Neighboor = 4
- New-sol = Food(2)+(Food(2)−Food(4))×rand = 5.3639
- New-sol = 4, New-objval = 0
- Fitness(New-sol) = 1

- As Fitness(New-sol) < Fitness(02), **So don't change**
- Trial=(0 1 0 0 0)

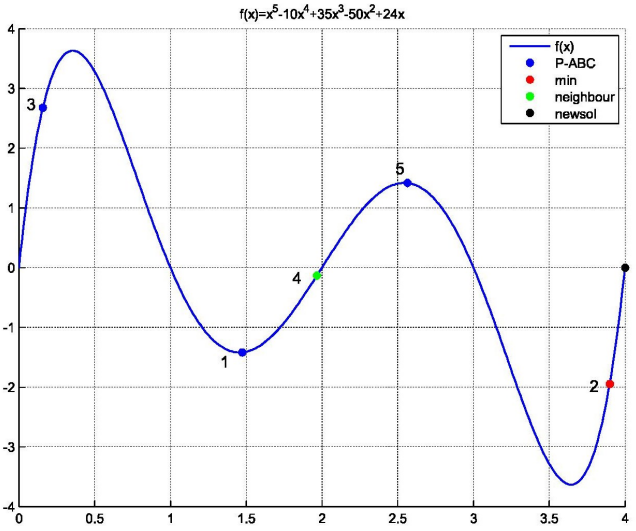


FIGURE 2.9: Food source positions of employed bee 02.

Employed Bee 03, see figure 2.10

- Neighboor = 4
- New-sol = Food(3)+(Food(3)−Food(4))×rand = 0.4452
- New-sol = 0.4452, New-objval = 3.4876
- Fitness(New-sol) = 0.2228
- As Fitness(New-sol) < Fitness(03), **So don't change**
- Trial=(0 1 1 0 0)

Employed Bee 04, see figure 2.11

- Neighboor = 1
- New-sol = Food(4)+(Food(4)−Food(1))×rand = 1.8495
- New-sol = 1.8495, New-objval = −0.5850
- Fitness(New-sol) = 1.5850
- As Fitness(New-sol) > Fitness(04), **So change**
- Trial=(0 1 1 0 0)

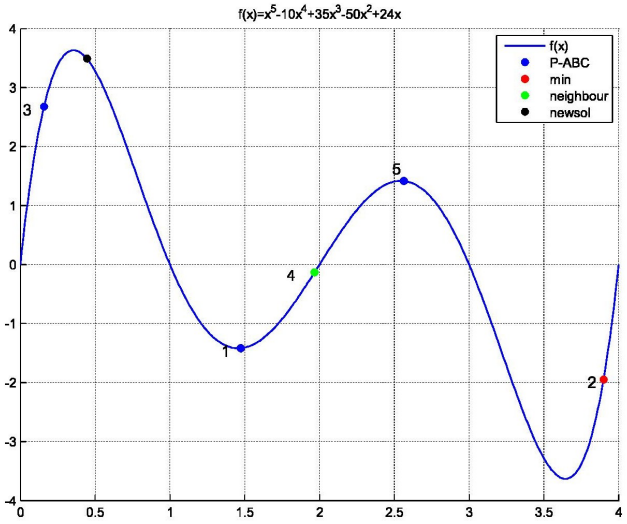


FIGURE 2.10: Food source positions of employed bee 03.

***Employed Bee 05*,**

- Neighboor = 3
- New-sol = Food(5)+(Food(5)−Food(3))×rand = 1.4754
- New-sol = 1.4754, New-objval = 1.4163
- Fitness(New-sol) = 2.4163
- As Fitness(New-sol) > Fitness(05), **So change**....
- Trial=(0 1 1 0 0)

Onlooker Bee,

- Calculate probability for each solution

  0.8384 1.0000 0.1831 0.5843 0.8383
- Select solution due to probability
- ,,,,,,,,,,,,,,,,,,,,,,
- Finally Trial=(0 2 1 0 1)

Scout Bee

- Come counter of Trial ≠ Limit, no send scouts to find new source.

At the end,

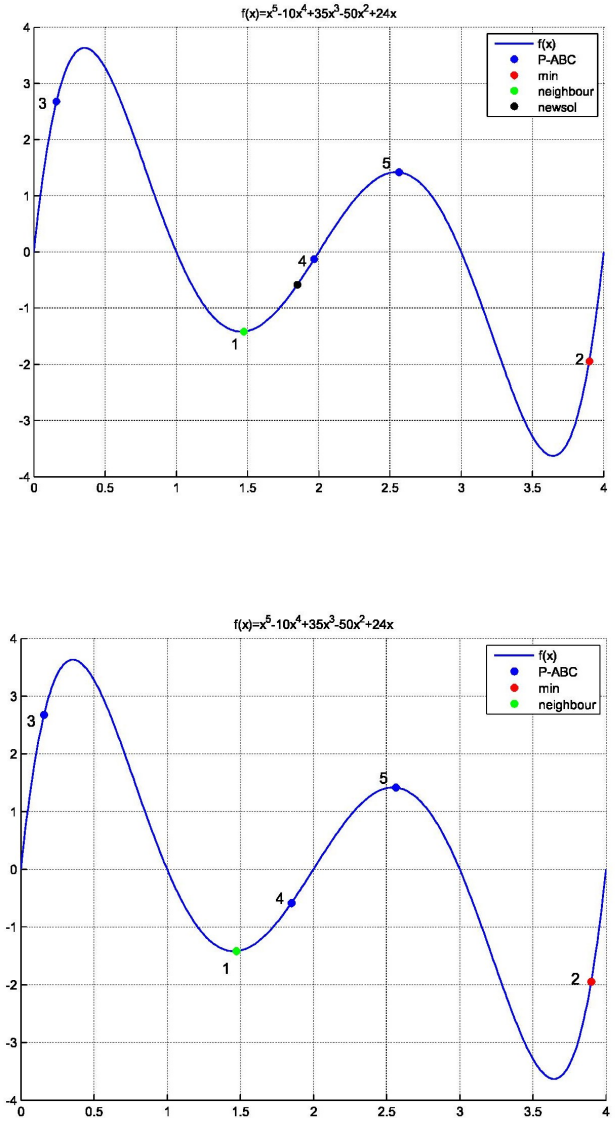


FIGURE 2.11: Food source positions of employed bee 04.

- We get the positions of food sources for Cycle 1. See figure 2.12, table 2.5.
- The best food source is memorized by the bee:
  GlogalMin $= (3.8992; -1.94545)$

Cycle 03**:** ***At the end***,

- We get the positions of food sources for Cycle 3. See figure 2.13, table 2.6.

TABLE 2.5: Food source positions of Cycle 01.

Foods	**Objval**	**Fitness**
1.4727	-1.4169	2.4169
3.8992	-1.9454	2.9454
0.7369	1.8082	0.3561
1.8139	-0.7125	1.7125
1.4754	-1.4163	2.4162

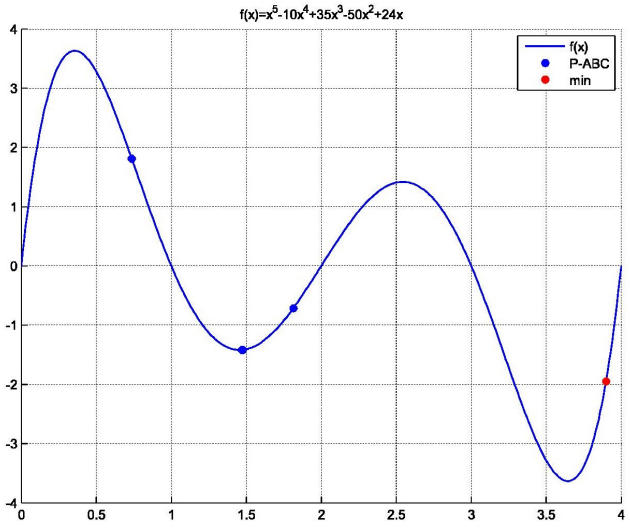


FIGURE 2.12: Food source positions of Cycle 01.

TABLE 2.6: Food source positions of Cycle 03.

Foods	**Objval**	**Fitness**
3.8003	-3.0623	4.0623
3.8377	-2.7209	3.7209
0	0	1
1.8139	-0.7125	1.7125
3.5991	-3.5928	4.5928

- The best food source is memorized by the bee:
  GlogalMin $= (3.5991; -3.5928)$

Cycle 04**:** ***At the end***,

- We get the positions of food sources for Cycle 4. See figure 2.14, table 2.7.

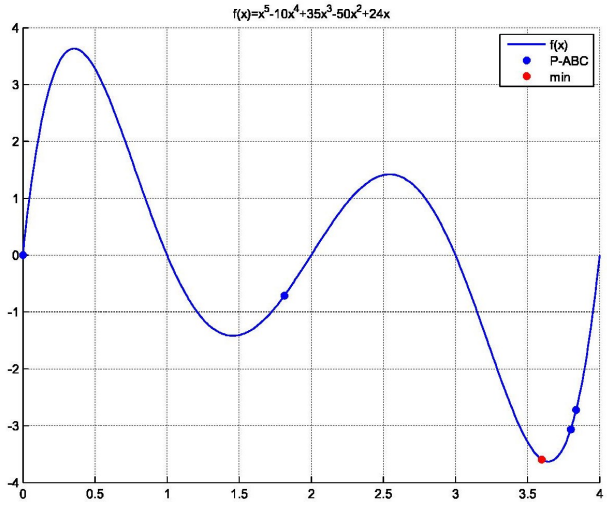


FIGURE 2.13: Food source positions of Cycle 03.

TABLE 2.7: Food source positions of Cycle 04.

Foods	**Objval**	**Fitness**
3.8003	-3.0623	4.0623
3.4916	-3.2433	4.2433
1.3598	1.3564	2.3564
1.5932	-1.3018	2.3018
3.5995	-3.5934	4.5934

- The best food source is memorized by the bee:
  GlogalMin $= (3.5995; -3.5934)$.

Cycle 10**:** ***At the end***,

- We get the positions of food sources for Cycle 10. See figure 2.15, table 2.8.

TABLE 2.8: Food source positions of Cycle 10.

Foods	**Objval**	**Fitness**
3.6530	-3.6300	4.6300
3.7427	-3.4186	4.4186
3.6481	-3.6312	4.6312
3.6481	-3.6312	4.6312
3.6485	-3.6311	4.6311

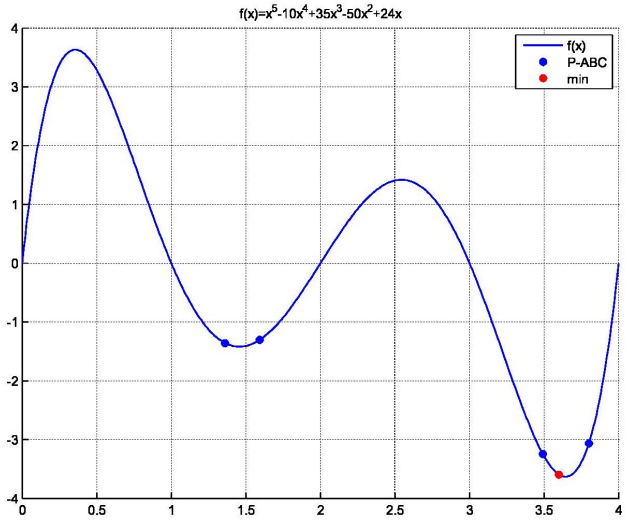


FIGURE 2.14: Food source positions of Cycle 04.

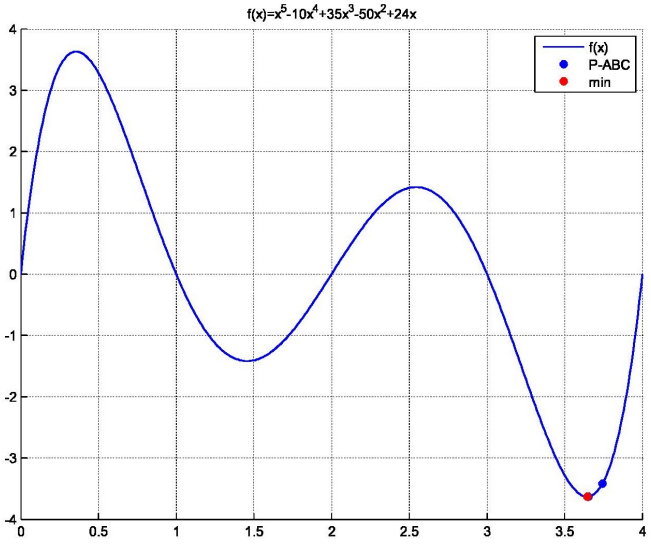


FIGURE 2.15: Food source positions of Cycle 10.

- The best food source is memorized by the bee:
  GlogalMin $= (3.6481; -3.6312)$.

Chapter 3

Improvement of Gregory's Formula using Artificial Bee Colony Algorithm

3.1 Gregory's Formula for Solving Numerical Integration

Solving numerical integration is an important question in scientific calculations and engineering. Gregory's method is among the very first quadrature formulas ever described in the literature, dating back to James Gregory (1638-1675)[38–42]. It seems to have been highly regarded for centuries.

Consider the Gregory integration formula (G) [2],

$$\int_0^n f(x)dx = \sum_{k=0}^{n} f(k) + \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(\Delta_{h_k}^k f(0) + \Delta_{-h_k}^k f(n)\right). \tag{3.1}$$

with end corrections where Δ_h is the forward difference operator with step size h. Note that for $h_k = 1$ $(k = 1, 2, ...)$, the formula (3.1) reduces to the classical Gregory integration formula[1],

$$\int_0^n f(x)dx = \sum_{k=0}^{n} f(k) + \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(\Delta_1^k f(0) + \Delta_{-1}^k f(n)\right). \tag{3.2}$$

The formula (3.1) has a sense so $n \geq 1$, In the contrary case an appropriate variable change will permit us to do the integral without no difficulty.

To justify the formula (3.1) we shall use the umbral methods developed by Rota and his school [4–8], instead of classical generating function technique.

So, we shall replace $f(x)$ by e^{tx} (e^{tx} is the generating function of the sequence $\frac{t^n}{n!}$).

We have,

$$\int_0^n e^{tx} dx = \frac{e^{nt} - 1}{t}.$$

$$\sum_{k=0}^{n} e^{tk} = \frac{(e^t)^{n+1} - 1}{e^t - 1} = \frac{e^{nt}.e^t - 1}{e^t - 1}.$$

$$\Delta_{h_k}^k e^{tx} = e^{tx} \left(e^{th_k} - 1\right)^k.$$

Then (3.1) becomes

$$\frac{e^{nt} - 1}{t} = \frac{e^{nt}.e^t - 1}{e^t - 1} + \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(e^{th_k} - 1\right)^k + \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(e^{-th_k} - 1\right)^k e^{tn},$$

so,

$$\frac{e^{nt}}{t} - \frac{1}{t} = \frac{e^{nt}.e^t}{e^t - 1} - \frac{1}{e^t - 1} + \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(e^{th_k} - 1\right)^k + e^{nt} \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(e^{-th_k} - 1\right)^k,$$

so,

$$e^{nt} \left(\frac{1}{t} - \frac{e^t}{e^t - 1} - \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(e^{-th_k} - 1\right)^k\right) + \left(\frac{1}{-t} + \frac{1}{e^t - 1} - \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(e^{th_k} - 1\right)^k\right) = 0,$$

so,

$$e^{nt} \left(\frac{1}{t} + \frac{1}{e^{-t} - 1} - \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(e^{-th_k} - 1\right)^k\right) + \left(\frac{1}{-t} + \frac{1}{e^t - 1} - \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(e^{th_k} - 1\right)^k\right) = 0.$$

Suppose that

$$G(t) = \frac{1}{t} + \frac{1}{e^{-t} - 1} - \sum_{k\geq 0} \frac{\alpha_k}{k!} \left(e^{-th_k} - 1\right)^k,$$

then (3.1) becomes

$$e^{nt} G(t) + G(-t) = 0.$$

we want the formula (3.1) that is independent of n. So $G(t) = 0$; from The Expansion Theorem, we have

$$\frac{1}{e^t - 1} - \frac{1}{t} = \sum_{k \geq 0} \frac{\alpha_k}{k!} \left(e^{th_k} - 1\right)^k,$$

where,

$\alpha_k =< \frac{1}{e^t - 1} - \frac{1}{t} \mid p_k(x) >$,

$p_k(x)$ is the sequence of polynomials associated for $\left(e^{th_k} - 1\right)^k$.

Suppose that

$$f(t) = \frac{1}{e^t - 1} - \frac{1}{t},$$

so,

$$f(t) = \frac{t - (e^t - 1)}{t(e^t - 1)},$$

so,

$$\left(e^t - 1\right).f(t) = 1 - \frac{(e^t - 1)}{t},$$

and suppose that

$$f(t) = \sum_{n \geq 0} \gamma_n t^n,$$

so,

$$\sum_{k \geq 0} \frac{t^k}{(k+1)!} \sum_{k \geq 0} \gamma_k t^k = -\sum_{k \geq 0} \frac{t^k}{(k+2)!},$$

so,

$$\sum_{k \geq 0} \left(\sum_{n=0}^{k} \frac{1}{(k+1)!} \gamma_{n-k} \right) t^k = -\sum_{k \geq 0} \frac{t^k}{(k+2)!},$$

then the last equality is equivalent to the system

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & \cdots \\ \frac{1}{2!} & 1 & 0 & 0 & \cdots & 0 & \cdots \\ \frac{1}{3!} & \frac{1}{2!} & 1 & 0 & \cdots & 0 & \cdots \\ \frac{1}{4!} & \frac{1}{3!} & \frac{1}{2!} & 1 & \cdots & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{n!} & \frac{1}{(n-1)!} & \frac{1}{(n-2)!} & \cdots & \cdots & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \cdot \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \vdots \\ \gamma_n \\ \vdots \end{bmatrix} = \begin{bmatrix} -\frac{1}{2!} \\ -\frac{1}{3!} \\ -\frac{1}{4!} \\ -\frac{1}{5!} \\ \vdots \\ -\frac{1}{(n+2)!} \\ \vdots \end{bmatrix}$$

Therefore

$$\gamma_0 = \frac{-1}{2}; \gamma_1 = \frac{1}{12}; \gamma_2 = 0; \gamma_3 = \frac{-1}{720}; \gamma_4 = 0; \cdots$$

So,

$$\begin{aligned} \alpha_0 =< f(t) \mid p_0(x) > &=< f(t) \mid 1 > \\ &= \gamma_0 \\ &= -\frac{1}{2}. \end{aligned}$$

$$\begin{aligned} \alpha_1 =< f(t) \mid p_1(x) > &=< f(t) \mid \frac{1}{h_1}x > \\ &= \frac{1}{h_1}\gamma_1 \\ &= \frac{1}{12h_1}. \end{aligned}$$

$$\begin{aligned} \alpha_2 =< f(t) \mid p_2(x) > &=< f(t) \mid -\frac{h_1}{2h_2^2}x + \frac{1}{h_2^2}x^2 > \\ &= -\frac{h_1}{2h_2^2}\gamma_1 + \frac{1}{h_2^2}\gamma_2 \\ &= -\frac{h_1}{24h_2^2}. \end{aligned}$$

In the same way, we find

$$\alpha_3 = \frac{1}{720h_3^3}\left(-10h_1^2 + 30h_2h_1 - 1\right).$$

$$\alpha_4 = \frac{-1}{480h_4^4}\left(\frac{5}{3}h_1^3 - 10h_3h_1^2 - \frac{35}{3}h_2^2h_1 + 30h_3h_2h_1 - h_3\right).$$

$$\begin{aligned} \alpha_5 = \frac{1}{60480h_5^5})(&-42h_1^4+630h_2^3h_1+1050h_3^2h_2^2-3150h_3^2h_2h_1+420h_4h_1^3-2520h_4h_1^2h_3 \\ &- 2940h_4h_2^2h_1 + 7560h_4h_3h_2h_1 + 105h_3^2 - 252h_4h_3 + 2). \end{aligned}$$

Truncating the right member of (3.1) at the 5^{th} term, we get the approximation:

$$\begin{aligned}\int_0^n f(x)dx \approx & \sum_{k=0}^{n} f(k) + a_0\Big(f(0)+f(n)\Big)\\ & +\alpha_1(h_1)\Big(f(h_1)-f(0)+f(n-h_1)-f(n)\Big)\\ & +\frac{\alpha_2(h_1,h_2)}{2!}\Big(f(2h_2)-2f(h_2)+f(0)+f(n-2h_2)-2f(n-h_2)+f(n)\Big)\\ & +\frac{\alpha_3(h_1,\cdots,h_3)}{3!}\Big(\Delta^3_{h_3}f(0)+\Delta^3_{-h_3}f(n)\Big)\\ & +\frac{\alpha_4(h_1,\cdots,h_4)}{4!}\Big(\Delta^4_{h_4}f(0)+\Delta^4_{-h_4}f(n)\Big)\\ & +\frac{\alpha_5(h_1,\cdots,h_5)}{5!}\Big(\Delta^5_{h_5}f(0)+\Delta^5_{-h_5}f(n)\Big). \qquad (3.3)\end{aligned}$$

3.2 Improvement of Gregory's Formula using Artificial Bee Colony Algorithm

In this section we prove that the Gregory Formula (G) can be optimized by minimizing some of their coefficients in the remainder term ($\alpha_3, \alpha_4, \alpha_5$) by using Artificial Bee Colony (ABC) Algorithm [43–45].

For it; we try to determine h_1, h_2, h_3, h_4 and h_5; we take $h_4, h_5 = 1$, in this study as parameters and let's solve non linear system by Artificial Bee Colony Algorithm.

The ABC algorithm was firstly introduced by Karaboga in 2005 [3] for numerical optimization problems based on the foraging behaviour of honey bee swarm, and the performance of ABC is analyzed in 2007 [26].

The algorithm consists of the following bee groups: employed bees, onlookers and scouts as in nature. A bee which has found a food source to exploit is called an employed bee. Onlookers are those waiting in the hive to receive the information about the food sources from the employed bees and Scouts are the bees which are randomly searching for new food sources around the hive. After exploiting a food source, an employed bee returns to the hive and shares the information about the nectar amount of the food source with other bees by dancing in the dance area of the hive.

The position of a food source represents a possible solution of the optimization

problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution.

In the ABC algorithm, half of the colony are employed bees while the other half consists of onlookers, also it is assumed that the number of food sources is equal to the number of employed bees. After abandoning a food source, the employed bee of that food source becomes a scout and carries out a random search. Same as other swarm intelligence based algorithms, the ABC algorithm has an iterative process.

Now, we use Artificial Bee Colony Algorithm to optimize the following function;

$$\begin{aligned} F(H_i) &= F(h_{1,i}, h_{2,i}, h_{3,i}) \\ &= |\alpha_3|^2 + |\alpha_4|^2 + |\alpha_5|^2. \end{aligned} \tag{3.4}$$

Where $h_{j,i} \in [h_{j,\min}; h_{j,\max}]\,, i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, 3\}\,, n$ is the number of employed bees.

We are going to give the objective function coded using MATLAB language in Appendix. A.4.

The main steps of the algorithm of the simulation of foraging and dance behaviors of honey bee colony adopted from [3, 35–37]; to optimize the function (3.4); are given below:

Step 1 Initialize the initial swarm $H_i = (h_{1,i}, h_{2,i}, h_{3,i})$ by using equation

$$h_{j,i} = h_{j,\min} + rand\,[0; 1] * (h_{j,\max} - h_{j,\min}) \tag{3.5}$$

Calculate $F(H_i)$ by using equation (3.4) and the fitness (fit_i) of each food source by using equation

$$fit_i = \begin{cases} \frac{1}{1+F(H_i)} & \text{if} \quad (F\,(H_i) \geq 0) \\ 1 + abs(F\,(H_i)) & \text{if} \quad (F\,(H_i) < 0) \end{cases} \tag{3.6}$$

Step 2 (Move the employed bees) Calculate the new solution $h^*_{j,i}$ by using equation

$$h^*_{j,i} = h_{j,i} + rand\,[-1;1] * (h_{j,i} - h_{j,k}), i \neq k; i,k \in \{1,2,...,n\} \tag{3.7}$$

Where j,k are selected randomly and $h_{j,k}$ is a neighbor bee of $h_{j,i}$.
Calculate $F(H^*_i)$ by using equation (3.4) and its fitness (fit_i) by using equation (3.6), After that we compare this fitness with its old one. If the new food source fitness has equal or better than the old fitness, the old one is replaced by the new one. Otherwise, the old one is retained.

Step 3 (Move the onlookers) Calculate The probability p_i of selecting the food source i by

$$p_i = \frac{fit_i}{\sum_{j=1}^{n} fit_j} \tag{3.8}$$

For Improving the solution H^*_i we use the main operations of Step 2.

Step 4 (Move the Scouts) If the fitness values of the employed bees are not improved by a continuous predetermined number of iterations, which is called *Limit* those food sources are abandoned, and these employed bee become the scouts, and by using equation (3.5) generate a new solution for the employed bee.

Step 5 If the termination condition is met, the stop and the best food source is memorized; otherwise the algorithm returns to Step 2.

3.3 Simulation Results

In the simulation studies, Artificial Bee Colony (ABC) Algorithm was applied for optimize a_3, a_4 and a_5, where the maximum number of cycles was taken as 2000. The percentages of onlooker bees and employed bees were %50 of the colony and the number of scout bees was selected to be one.
The increase in the number of scouts encourages the exploration process while the

increase of onlookers on a food source encourages the exploitation process.

Parameters adopted for the ABC algorithm are given in table 3.1.

TABLE 3.1: Parameters

Max cycle	Swarm size	Limit	Number of onlookers bees	Number of employed bees	Number of scouts
2000	30	45	15	14	1

The ABC algorithm provides us the solution: $h_1 = 0.08, h_2 = 0.4, h_3 = 0.5$.

To test the performance of this algorithm we took various functions and we looked for an approximation with Gregory formula (G) and formula $GABC$. The Comparison is given in table 3.2.

TABLE 3.2: Comparison

Function	Interval	Exact value	Formula	Approx.value	Rel. Error
$exp(x)$	$[0, 5]$	$147,4131591025766$	G	149.2289234815335	0.01231
			GABC	147.3572775013750	0.00037
$\frac{1}{1+x^2}$	$[0, 5]$	1.373400766945016	G	1.328883861236802	0.00324
			GABC	1.375035677680117	0.00119
$x(1-x)$	$[0, 15]$	-1025	G	-1012.75	0.01195
			GABC	-1025	0.00000
$\frac{e^x}{\sqrt{e^x+1}}$	$[0, 20]$	44050.1032078	G	44099.751818	0.00112
			GABC	44049.336590	0.00001
$\frac{3x+1}{(x+1)^2}$	$[0, 10]$	5.415908040617334	G	5.326509523358369	0.01650
			GABC	5.387058431515922	0.00532

Conclusion and Future Work

This thesis has presented a ***new numerical integration formula (GABC)*** [43, 45],

$$\begin{aligned}\int_0^n f(x)dx \approx \sum_{k=0}^{n} f(k) - 0.5\,(f(0)+f(n)) \\ + 1.04\,(f(0.08) - f(0) + f(n-0.08) - f(n)) \\ - 0.01\,(f(0.8) - 2f(0.4) + f(0) + f(n-0.8) - 2f(n-0.4) + f(n))\,.\end{aligned}$$

Experimental results on several well-known functions (badly to integrate by the classic methods $(exp(x), \frac{1}{1+x^2}, ...)$ show that the proposed formula give good results and prove that obtained formula can be rendered a powerful formula for library use.

In future work; we introduce the ABC algorithm to find an approximate solution of initial-value problem (IVP),

$$\begin{cases} y' = f(x,y) \\ y(a) = y_0 \end{cases}$$

Where $x \in [a,b]$ and $f = f(x,y)$ is a real-valued function of two real variables.

Appendix A

MATLAB Code

A.1 ABC Algorithm Coded using MATLAB Language

```
%/* ABC algorithm coded using MATLAB language */

%/* Artificial Bee Colony (ABC) is one of the most recently defined algorithms by
Dervis Karaboga in 2005, motivated by the intelligent behavior of honey bees. */

clear all
close all
clc

%/* Control Parameters of ABC algorithm*/
NP= %/* The number of colony size (employed bees+onlooker bees)*/
FoodNumber=NP/2 %/*The number of food sources equals the half of the colony size*/
limit= %/*A food source which could not be improved through "limit" trials is
abandoned by its employed bee*/
maxCycle= %/*The number of cycles for foraging {a stopping criteria}*/

%/* Problem specific variables*/
objfun='' %cost function to be optimized
D=  %/*The number of parameters of the problem to be optimized*/
ub=ones(1,D)*1 %/*lower bounds of the parameters. */
lb=ones(1,D)*0%/*upper bound of the parameters.*/

runtime=1%/*Algorithm can be run many times in order to see its robustness*/
%Foods [FoodNumber][D]; /*Foods is the population of food sources. Each row of Foods
matrix is a vector holding D parameters to be optimized. The number of rows of Foods
matrix equals to the FoodNumber*/
%ObjVal[FoodNumber];  /*f is a vector holding objective function values associated
```

```
with food sources */
%Fitness[FoodNumber]; /*fitness is a vector holding fitness (quality) values
associated with food sources*/
%trial[FoodNumber]; /*trial is a vector holding trial numbers through which solutions
can not be improved*/
%prob[FoodNumber]; /*prob is a vector holding probabilities of food sources (solutions)
to be chosen*/
%solution [D]; /*New solution (neighbour) produced by
v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij})
j is a randomly chosen parameter and k is a randomlu chosen solution different
from i*/
%ObjValSol; /*Objective function value of new solution*/
%FitnessSol; /*Fitness value of new solution*/
%neighbour, param2change; /*param2change corrresponds to j, neighbour corresponds
to k in equation v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij})*/
%GlobalMin; /*Optimum solution obtained by ABC algorithm*/
%GlobalParams[D]; /*Parameters of the optimum solution*/
%GlobalMins[runtime]; /*GlobalMins holds the GlobalMin of each run in multiple
runs*/

GlobalMins=zeros(1,runtime)

for r=1:runtime

% /*All food sources are initialized */
%/*Variables are initialized in the range [lb,ub]. If each parameter has different
range, use arrays lb[j], ub[j] instead of lb and ub */

Range = repmat((ub-lb),[FoodNumber 1])
Lower = repmat(lb, [FoodNumber 1])
Foods = rand(FoodNumber,D) .* Range + Lower
ObjVal=feval(objfun,Foods)
Fitness=calculateFitness(ObjVal)

%reset trial counters
trial=zeros(1,FoodNumber)

%/*The best food source is memorized*/
BestInd=find(ObjVal==min(ObjVal))
BestInd=BestInd(end)
GlobalMin=ObjVal(BestInd)
GlobalParams=Foods(BestInd,:)

iter=1;
while ((iter <= maxCycle)),

%%%%%%%%% EMPLOYED BEE PHASE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:(FoodNumber)
```

```
  %/*The parameter to be changed is determined randomly*/
  Param2Change=fix(rand*D)+1

  %/*A randomly chosen solution is used in producing a mutant solution of
  the solution i*/
  neighbour=fix(rand*(FoodNumber))+1

  %/*Randomly selected solution must be different from the solution i*/
     while(neighbour==i)
     neighbour=fix(rand*(FoodNumber))+1
     end;

  sol=Foods(i,:)
  %  /*v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij}) */
  sol(Param2Change)=Foods(i,Param2Change)+(Foods(i,Param2Change)-
  Foods(neighbour,Param2Change))*(rand-0.5)*2

  %  /*if generated parameter value is out of boundaries, it is shifted
  onto
  the boundaries*/
    ind=find(sol<lb)
    sol(ind)=lb(ind)
    ind=find(sol>ub)
    sol(ind)=ub(ind)

  %evaluate new solution
   ObjValSol=feval(objfun,sol)
   FitnessSol=calculateFitness(ObjValSol)

  % /*a greedy selection is applied between the current solution i and its mutant*/
  if (FitnessSol>Fitness(i)) %/*If the mutant solution is better than the
  current solution i, replace the solution with the mutant and reset the
  trial counter of solution i*/
     Foods(i,:)=sol
     Fitness(i)=FitnessSol
     ObjVal(i)=ObjValSol
     trial(i)=0
  else
  trial(i)=trial(i)+1 %/*if the solution i can not be improved, increase
  its trial counter*/
  end;

  end;

%%%%%%%%%%%%%%%%%%%%%%% CalculateProbabilities %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%/* A food source is chosen with the probability which is proportioal to its
quality*/
```

```
%/*Different schemes can be used to calculate the probability values*/
%/*For example prob(i)=fitness(i)/sum(fitness)*/
%/*or in a way used in the metot below prob(i)=a*fitness(i)/max(fitness)+b*/
%/*probability values are calculated by using fitness values and normalized by
dividing maximum fitness value*/

prob=(0.9.*Fitness./max(Fitness))+0.1

%%%%%%%%%%%%%%%%%%%%%%% ONLOOKER BEE PHASE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

i=1
t=0
while(t<FoodNumber)
    if(rand<prob(i))
        t=t+1
        %/*The parameter to be changed is determined randomly*/
        Param2Change=fix(rand*D)+1

        %/*A randomly chosen solution is used in producing a mutant
        solution of the solution i*/
        neighbour=fix(rand*(FoodNumber))+1

        %/*Randomly selected solution must be different from the solution
        i*/
            while(neighbour==i)
                neighbour=fix(rand*(FoodNumber))+1
            end;

       sol=Foods(i,:);
       %  /*v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij}) */
       sol(Param2Change)=Foods(i,Param2Change)+(Foods(i,Param2Change)-
       Foods(neighbour,Param2Change))*(rand-0.5)*2

       %  /*if generated parameter value is out of boundaries, it is
       shifted onto the boundaries*/
        ind=find(sol<lb)
        sol(ind)=lb(ind)
        ind=find(sol>ub)
        sol(ind)=ub(ind)

        %evaluate new solution
        ObjValSol=feval(objfun,sol);
        FitnessSol=calculateFitness(ObjValSol)

       % /*a greedy selection is applied between the current solution i and
       its mutant*/
       if (FitnessSol>Fitness(i)) %/*If the mutant solution is better than
       the current solution i, replace the solution with the mutant and
```

```
       reset the trial counter of solution i*/
            Foods(i,:)=sol
            Fitness(i)=FitnessSol
            ObjVal(i)=ObjValSol
            trial(i)=0
        else
            trial(i)=trial(i)+1 %/*if the solution i can not be improved,
            increase its trial counter*/
       end;
    end;

    i=i+1
    if (i==(FoodNumber)+1)
        i=1
    end;
end;

%/*The best food source is memorized*/
         ind=find(ObjVal==min(ObjVal))
         ind=ind(end)
         if (ObjVal(ind)<GlobalMin)
         GlobalMin=ObjVal(ind)
         GlobalParams=Foods(ind,:)
         end;



%/*determine the food sources whose trial counter exceeds the "limit" value.
%In Basic ABC, only one scout is allowed to occur in each cycle*/

ind=find(trial==max(trial))
ind=ind(end)
if (trial(ind)>limit)
    trial(ind)=0
    sol=(ub-lb).*rand(1,D)+lb
    ObjValSol=feval(objfun,sol)
    FitnessSol=calculateFitness(ObjValSol)
    Foods(ind,:)=sol
    Fitness(ind)=FitnessSol
    ObjVal(ind)=ObjValSol
end
%/*The best food source is memorized*/
         ind=find(ObjVal==min(ObjVal))
         ind=ind(end)
         if (ObjVal(ind)<GlobalMin)
         GlobalMin=ObjVal(ind)
         GlobalParams=Foods(ind,:)
         end;
```

```
fprintf('iter=%d ObjVal=%g\n',iter,GlobalMin)
GlobalParams
iter=iter+1

end % End of ABC

GlobalMins(r)=GlobalMin

end; %end of runs

save all
```

A.2 The Fitness Function Coded using MATLAB Language

```
%/*the fitness function coded using MATLAB language */

function fFitness=calculateFitness(fObjV)

fFitness=zeros(size(fObjV));

ind=find(fObjV>=0);

fFitness(ind)=1./(fObjV(ind)+1);

ind=find(fObjV<0);

fFitness(ind)=1+abs(fObjV(ind));
```

A.3 The Complete MATLAB Code for Figure 2.7

```
%/*the objective function coded using MATLAB language */

clc

clear

x=0:0.01:4;

y=x.^5-10.*x.^4+35.*x.^3-50.*x.^2+24.*x;

xi=[0.8178 3.8992 0.1582 1.9672 2.5637];

yi=[1.2234 -1.9454 2.6783 -0.1312 1.4161];

xmin=3.8992;
ymin=-1.9454;

hold on
plot(x,y,'-','LineWidth',1);

plot(xi,yi,'b*','LineWidth',2);

plot(xmin,ymin,'r*','LineWidth',2);

grid;

title('f(x)=x^5-10x^4+35x^3-50x^2+24x');

legend('f(x)','P-ABC','Gmin');

hold off
```

A.4 The Objective Function Coded using MATLAB Language

```
%/*the objective function coded using MATLAB language */

function ObjVal = gregory1(Chrom,switc);

%Dimension of objective function

     Dim=size(Chrom,2);

% Compute population parameters

      Mat1 = Chrom(:,1);
      Mat2 = Chrom(:,2);
      Mat3 = Chrom(:,3);
      ObjVal =
      (((1./(720.*Mat3.^3)).*(-10.*Mat1.^2+30.*Mat2.*Mat1-1)).^2)+
      (((-1./480).*((5./3).*Mat1.^3-10.*Mat3.*Mat1.^2
        +30.*Mat3.*Mat2.*Mat1-Mat3)).^2)+
      (((1./60480).*((-42.*Mat1.^4)+(630.*Mat2.^3.*Mat1)
         +(1050.*Mat3.^2.*Mat2.^2)-(3150.*Mat3.^2.*Mat2.*Mat1)
         -(420.*Mat1.^3)-(2520.*Mat1.^2.*Mat3)-(2940.*Mat2.^2.*Mat1)
         +(7560.*Mat3.*Mat2.*Mat1)+(105.*Mat3.^2-252.*Mat3+2))).^2)

end
% End of function
```

Bibliography

[1] M.K. Belbahri. Generalized gregory formula. *Doctoral Thesis, Stevens Institute of Technology*, 1982.

[2] N. Khelil and M.K. Belbahri. Integration de gregory avec correction parametisee. *Magister Thesis, University of Batna*, 1991.

[3] D. Karaboga. An idea based on honey bee swarm for numerical optimization. *Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department*, October 2005. URL karaboga@erciyes.edu.tr.

[4] G.C. Rota. Finite operator calculus. *Academic press, Inc., New York*, 1975.

[5] I. Niven. Formal power series. *The American Mathematical Monthly*, 76(8): 871–889, October 1969.

[6] S. Roman. The theory of the umbral calculus. *Mathematical Analysis and Applications, Academic press, Inc., New York*, 87(1):58–115, May 1982.

[7] S. Roman. The umbral calculus. *Academic press, Inc.*, 1984.

[8] S. Roman and G.C. Rota. The umbral calculus. *Advances in Math, Academic press, Inc.*, 27:95–188, 1978.

[9] O.S. Zariski and P. Samuel. Commutative algebra volume 1. *Springer, D. VAN Nostrand Company, Inc.*, 1960.

[10] O.S. Zariski and P. Samuel. Commutative algebra volume 2. *Springer, D. VAN Nostrand Company, Inc.*, 1960.

[11] X.X. Gan and N. Knox. On composition of formal power series. *Hindawi Publishing Corporation*, pages 761–770, 2002.

[12] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga. A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Springer Science+Business*, March 2012.

[13] S. J. Russell and P. Norvig. Artificial intelligence a modern approach. *Alan Apt, United States of America*, 1995.

[14] RC. Chakraborty. Introduction to artificial intelligence. URL http://www.myreaders.info/htmlartificial_intelligence.html.

[15] J. McCarthy. What is artificial intelligence? *Computer Science Department. Stanford University*, 2007.

[16] A. P. Engelbrecht (2nd edition). Computational intelligence: An introduction. *John Wiley and Sons, England*, 2002.

[17] T. D. Kelley. Symbolic and sub-symbolic representations in computational models of human cognition. what can be learned from biology? *Sage Publications*, 13(6):847–860, 2003.

[18] E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm intelligence: From natural to artificial systems. *New York, NY: Oxford University Press*, 1999.

[19] D. Willshaw. Self-organization in the nervous system. *Institute for Adaptive and Neural Computation, School of Informatics, University of Edinburgh This*, 2006.

[20] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. *Tech. Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy*, 1991.

[21] M. Dorigo and T. Stützle. Ant colony optimization. *MIT Press, Cambridge, ISBN: 978-0-262-04219-2*, 2004.

[22] J. Kennedy and R. C. Eberhart. Particle swarm optimization. *In Proceedings of IEEE International Conference on Neural Networks, Perth, Australia*, page 1942–1948, 1995.

[23] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. *In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan*, page 39–43, 1995.

[24] D. Karaboga and B. Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation, Elsevier Inc.*, (214):108–132, 2009.

[25] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Springer Science and Business Media B.V*, (39):459–471, 2007.

[26] D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Elsevier B.V*, (8):687–697, 2008.

[27] D. Karaboga and B. Akay. Artificial bee colony (abc), harmony search and bees algorithms on numerical optimization. *Erciyes University, The Dept. of Computer Engineering, 38039, Melikgazi, Kayseri, Turkiye*, 2009.

[28] T. D. Seeley. The wisdom of the hive. *Harvard University Press, Cambridge, MA*, 1995.

[29] V. Tereshko. Reaction–diffusion model of a honey bee colony's foraging behaviour. *Springer–Verlag, Berlin*, 1917:807–816, 2000.

[30] V. Tereshko and T. Lee. How information mapping patterns determine foraging behaviour of a honey bee colony. *Open Systems and Information Dynamics, Kluwer Academic Puplisher*, (9):181–193, 2002.

[31] B. Yuce and D.T. Pham A. Lambiase M.S. Packianather, E. Mastrocinque. Honey bees inspired optimization method: The bees algorithm. *Insects ISSN 2075-4450*, (4):646–662, 2013.

[32] H. Yahya. The miracle of the honey bees. *Gursel Mh. Darulaceze Cd. No: 9 Funya Sk. Eksioglu Is Merkezi B Blok D: 5, Okmeydani-Istanbul/Turkey*, March 2007.

[33] URL http://mf.erciyes.edu.tr/abc/.

[34] D. Jeya Mala and .al. A non-pheromone based intelligent swarm optimization technique in software test suite optimization. *IEEE*, 2009.

[35] G. Zhu and S. Kwong. Gbest-guided artificial bee colony algorithm for numerical function optimization. *Elsevier*, page 3166–3173, 2010.

[36] M.S. Kiran and M. Gunduz. A novel artificial bee colony-based algorithm for solving the numerical optimization problems. *International Journal of Innovative Computing, Information and Control ICIC*, 8(9):6107–6121, 2012.

[37] P.W. TSai, J.S. Pan, B.Y. Liao, and S.C.Chu. Enhanced artificial bee colony optimization. *International Journal of Innovative Computing, Information and Control ICIC*, 5(12), 2009.

[38] C.E Froberg. Introduction to numerical analysis. *Addison-Wesley, Second Edition*, 1969.

[39] G. M. Phillips. Gregory's method for numerical integration. *The American Mathematical Monthly*, 79(3):270–274, 1972.

[40] A. Ralston and P. Rabinowitz. A first course in numerical analysis, 2nd ed. *New York:McGraw-Hill*, 1978.

[41] R. W. Hamming and R. S. Pinkhan. A class of integration formulas. *the Association for Computing Machinery*, 13(3):430–438, July 1966.

[42] C. Jordan. Calculus of finite differences, 3rd ed. *New York: Chelsea*, pages 284–287, 1965.

[43] S. Aichouche, N. Khelil, and L. Djerou. Improvement of gregory's formula using artificial bee colony algorithm. *Applied Computer Science and Mathematics, Suceava*, 10(21):22–24, 2016.

[44] N. Khelil, L. Djerou, A. Khernane, and S. Aichouche. Particle swarm optimization algorithm for improve the gregory's formula. *Conference on Metaheuristics and Nature Inspired Computing - META'2014*, 29-31 October 2014.

[45] S. Aichouche, N. Khelil, and L. Djerou. Improvement of gregory's formula using artificial bee colony algorithm. *Conference ICMSAO'15 6th International*

Conference on Modeling, Simulation AND Applied Optimization, Istambul-Turky, 27-29 May 2015.