

511.8  
SIN

T 158

50503

CENTRAL LIBRARY  
TEZPUR UNIVERSITY  
Accession No. T 158  
Date 28/02/13



# **An Empirical Study on Evolutionary Algorithms**

**A thesis submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy**

**By  
Sanjeev Kumar Singh  
Registration No. 001 of 2010**



**School of Science and Technology**  
Department of Mathematical Sciences  
Tezpur University, Napaam, Tezpur-784028, Assam, India  
June, 2010

# ABSTRACT

Over the period of time many population based evolutionary algorithm have been developed such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA) and Differential Evolution (DE) Genetic Algorithm mimics the natural process of evolution Simulated Annealing exploits the physical process of cooling of a liquid or solid to regain the crystalline structure Particle Swarm Optimization is inspired by the social behavior of bird flocking and Differential Evolution has its origin in Chebychev polynomial fitting problem All the above evolutionary algorithm has one thing common, that they have the multi-starting points while starting the search process But, each of them renews their initial population also known as candidate solutions using their parameters differently GA uses the parameters called “crossover” and “mutations” PSO renew the candidate solutions called particles flying through the problem space by following current optimum particles Simulated Annealing uses a random search strategy which not only accepts new positions that increase the objective function (for minimization problems) but also accept the positions which decreases the objective function values The latter is accepted probabilistically based on the Boltzmann-Gibbs distribution Differential Evolution uses three parents to reproduce offspring by arithmetic crossover operator Though all the above algorithms starts with the same initial population (initial solution), they differ in a way they reproduce the new set of population (intermediate solution) and move towards the optimum solution So, it becomes important to study the performance of all these algorithms on the test functions There is a collection of test functions available in literature and numbers are increasing

In tune of further development we felt that there is a need to develop new set of test functions to test the robustness and performance of above evolutionary algorithms In this study we have developed a set of new test functions generated with specific properties and coded the functions in MATLAB to get the visual presentations of the functions using the mesh and surface plotter of the MATLAB The four evolutionary optimizers such as Differential Evolution, Genetic algorithm, Particle Swarm Optimization and Simulated Annealing have been used to find the optimum value of these newly developed test functions The algorithms run with different population sizes and for different number of iterations and results have been recorded in

tabular form. The results also have been analyzed and validated based on the minimum values found by the optimizers.

Since the development of above family of Evolutionary Computing algorithms were lacking the theoretical base and missing convergence criteria, it became important to study the performance and robustness of the above techniques using large number of test problems. So, as a further study on Evolutionary Algorithms (Evolutionary Computing) collection of test functions started appearing. Chattopadhyay (1971) studied some class of test functions for optimization algorithms and also explained the method of generating test functions with certain specific properties. Constrained and Unconstrained Testing Environment (CUTE) is suite for FORTRAN subroutines, scripts and test problems for linear and nonlinear optimizations is a large collection of test functions developed by Jorge et al (1981). In tune of further work on test functions, Floudas and Pardalos (1987) published a collection of test problems for constrained optimization and unconstrained optimization algorithms. Nagendra (1997) published a catalogue of test functions to test the performance of the Evolutionary Algorithm. Andrei (2008) also added another collection of test functions for unconstrained optimization.

Again in absence of strong convergence criteria, researchers started studying the performance and robustness of the evolutionary algorithms using the test functions. Ackley (1987) published the empirical study of vector function optimization. An experimental study in non convex optimization was done by Styblinski and Tang (1990). Deb (1991) used genetic algorithm to optimize multi-model functions. Fogel (1996) published evolutionary computation towards a new philosophy of machine intelligence. Michalewicz (1999)'s book entitled "Genetic Algorithms+ Data structure = Evolution program" deals with the real life numerical problem and step by step experimental studies. Jason and Konstantinos (2002) did experimental study of benchmarking test functions for Genetic Algorithm. Lewis (2008) in a "survey of meta-heuristics technique" argued that all these global optimization techniques falls under the Evolutionary Computing and known as population based Meta heuristics technique.

The aim of this thesis is two fold (1) Development of new test functions for the empirical study evolutionary computing such as genetic Algorithms, Particle Swarm Optimization, Simulated Annealing and Differential Evolutions, (2) Collect the benchmark test functions available in



literature and do the comparative study of the above optimizers using the benchmark test functions

\*\*\*\*\*

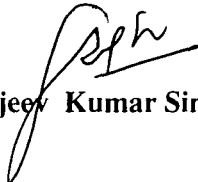
# DECLARATION

I hereby declare that the thesis entitled “**An Empirical Study on Evolutionary Algorithms.**” submitted to the Tezpur University, Tezpur is a bonafide research work carried out by me under the supervision of Prof. Munindra Borah, Department of Mathematical Sciences, Tezpur University, Tezpur, Assam.

The result embodied in this thesis has not been submitted to any other University or Institute for the award of any other Degree.

Place: Napaam, Tezpur

Date: 27/06/2011

  
Sanjeev Kumar Singh



# TEZPUR UNIVERSITY

## CERTIFICATE

This is to certify that thesis entitled “**An Empirical Study on Evolutionary Algorithms.**” submitted to Tezpur University in the Department of **Mathematical Sciences** under the school of **Science and Technology** in partial fulfillment for the award of the Degree of Philosophy in **Mathematics** is a record of research work carried out by **Mr. Sanjeev Kumar Singh** under my supervision and guidance.

All helps received by him from various sources have been duly acknowledged.

No part of this thesis has been reproduced elsewhere for the award of any other degree.

Place: Napaam, Tezpur

Date: 21/1/2011



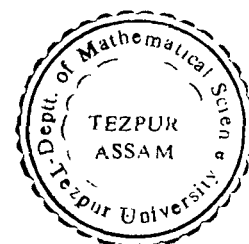
**Munindra Borah**

**Professor**

Department of Mathematical Science

School of Science and Technology

Tezpur University, Assam



# CONTENTS

	<b>Pages</b>
<b>Chapter 1. Introduction</b>	<b>1-14</b>
1.1 Empirical Study	1
1.2 Evolutionary Algorithms and its Evolution	4
1.3 Study Area	7
1.4 Population based Metaheuristics	7
1.4.1 Pseudo codes of Evolutionary Computations	8
(a) Genetic Algorithm	8
(b) Particle Swarm Optimization	10
(c) Simulated Annealing	10
(d) Differential Evolution	11
1.5 Objectives	12
1.6 Outline of the Thesis	13
<b>Chapter 2. New test function for Unconstrained Global Optimization</b>	<b>15-35</b>
2.1 Introduction	15
2.2 Purpose of developing new test functions	17
2.3 Some newly developed test functions	18
2.3.1 Generalized version of test functions from 201 to 211	24
2.3.2 Extended version of the Functions From 201 to 211	27
2.3.3 Extension of some benchmark test functions	29
2.4 Features of newly developed test functions	32
2.5 Reason that newly developed test function needed for study	34
2.6 Summary	34
<b>Chapter 3. Experimental Results of New Test Functions</b>	<b>36-51</b>
3.1 The results of new test functions	36
3.1.1 Experiment setup	36
3.1.2 Experimental results	39
3.2 Result recorded from the Differential Evolution at different iterations	41
3.3 Result recorded from the Genetic Algorithm at different iterations	43
3.4 Result recorded from the Particle Swarm at different iterations	44
3.5 Result recorded from the Simulated Annealing at different iterations	46
3.6 Analysis of the result recorded in the above table	48
(a) Analysis based on the results recorded from Differential Evolution	48

	(b) Analysis based on the results recorded from Genetic Algorithm	49
	(c) Analysis based on the results recorded from Particle Swarm	50
	(d) Analysis based on the results recorded from Simulated Annealing	50
3.7	Summary	51

## **Chapter 4. Comparative Study of Evolutionary Algorithm with Benchmark Test Function** **52-61**

<b>4.1</b>	<b>Comparative study of Modified Differential Evolution and Genetic Algorithm</b>	<b>52</b>
4.1.1	Methods used	52
4.1.2	Benchmark test functions used for the experiments	53
4.1.3	Experiments on Genetic Algorithms and Modified DE	53
4.1.4	Experimental results and analysis	53
4.1.5	Summary of section 4.1	54
<b>4.2</b>	<b>A comparative study of Swarm Intelligence Optimization Evolutionary Optimization</b>	<b>54</b>
4.2.1	Introduction	55
4.2.2	Methods and Algorithms used	55
4.2.3	Test functions used	55
4.2.4	Experiments on Genetic Algorithm and Particle Swarm	56
4.2.5	Experimental results and analysis	56
4.2.6	Summary of Section 4.2	57
<b>4.3</b>	<b>Study of Performance of Population based meta heuristics</b>	<b>57</b>
4.3.1	Methods used	57
4.3.2	Benchmark test functions used	57
4.3.3	Experiment setup	58
4.3.4	Experimental results and analysis	58
4.3.5	Summary of Section 4.3	59
<b>4.4</b>	<b>A Comparative study of Particle Swarm Optimization and Simulated Annealing</b>	<b>59</b>
4.4.1	Introduction	60
4.4.2	Method used	60
4.4.3	Test functions used	60
4.4.4	Experiments	60
4.4.5	Experimental results and analysis	61
4.4.6	Summary of Section 4.4	61

<b>Chapter 5. Concluding Discussions</b>	<b>62-64</b>
5.1 Conclusion for empirical study on evolutionary algorithms using new test functions	62
5.2 Comparative study (Analysis and conclusion)	63
5.3 Future direction of research	64
<b>References</b>	<b>65-75</b>
<b>Appendix-A Benchmark test functions for unconstrained global optimization</b>	<b>76-112</b>
(i) First set of test functions	77
(ii) Recently appeared test functions	99
(iii) Some more Benchmark test functions from (CUTE)	102
(iv) Some more test functions	109
<b>Appendix-B MATLAB codes for visual presentation of test functions</b>	<b>113-121</b>
(i) Code for Benchmark Test function for visual presentation	113
(ii) Code for the new functions introduced	119
<b>Appendix- C Executed commands for new test functions</b>	<b>122-124</b>

## List of Tables

		<b>Pages</b>
<b>Table 3.1</b>	(a) Optimum Value obtained by Differential Evolution and Genetic Algorithm	39
	(b) Optimum Value obtained by Particle Swarm Optimization and Simulated Annealing	40
<b>Table 3.2</b>	(a) Optimum Value recorded by Differential Evolution for 100,200,300 and 400 iterations	41
	(b) Optimum Value recorded by Differential Evolution for 500,600,700 and 1000 iterations	42
<b>Table 3.3</b>	(a) Optimum Value recorded by Genetic Algorithms for 100,200,300 and 400 iterations	43
	(b) Optimum Value recorded by Genetic Algorithms for 500,600,700 and 1000 iterations	44
<b>Table 3.4</b>	(a) Optimum Value recorded by Particle Swarm Optimization for 100,200,300 and 400 iterations	44
	(b) Optimum Value recorded by Particle Swarm Optimization for 500,600,700 and 1000 iterations	45
<b>Table 3.5</b>	(a) Optimum Value recorded by Simulated Annealing for 100,200,300 and 400 iterations	46
	(b) Optimum Value recorded by Simulated Annealing for 500,600,700 and 1000 iterations	47
<b>Table 4.1</b>	Comparative study of Genetic Algorithm and Modified DE	54
<b>Table 4.2</b>	Comparative study of Genetic Algorithm and Particle Swarm Optimization	56
<b>Table 4.3</b>	Comparative study of Genetic Algorithms, Particle Swarm Optimization and Simulated Annealing	58
<b>Table 4.4</b>	Comparative study of Particle Swarm Optimization and Simulated Annealing.	61

# List of Figures

		<b>Pages</b>
<b>Figure 2.1</b>	F201: Tortoise Function	18
	(a) Graph drawn by Meshz Plot	
	(b) Graph drawn by Surf Plot	
	(c) Graph drawn by Surfc Plot	
	(d) Graph drawn by Surf1 Plot	
<b>Figure 2.2</b>	F202: Crosscap Function	19
	(a) Graph drawn by Meshz Plot	
	(b) Graph drawn by Surf Plot	
	(c) Graph drawn by Surfc Plot	
	(d) Graph drawn by Surf1 Plot	
<b>Figure 2.3</b>	F203: Inverted Crosscap Function	19
	(a) Graph drawn by Meshz Plot	
	(b) Graph drawn by Surf Plot	
	(c) Graph drawn by Surfc Plot	
	(d) Graph drawn by Surf1 Plot	
<b>Figure 2.4</b>	F204: Four-hole table Function	20
	(a) Graph drawn by Meshz Plot	
	(b) Graph drawn by Surf Plot	
	(b) Graph drawn by Surfc Plot	
	(c) Graph drawn by Surf1 Plot	
<b>Figure 2.5</b>	F205: Cross on rough ceiling Function	21
	(a) Graph drawn by meshz Plot	
	(b) Graph drawn by Surf Plot	
	(c) Graph drawn by Surfc Plot	
	(d) Graph drawn by Surf1 Plot	
<b>Figure 2.6</b>	F206: Crosshut Function	21
	(a) Graph drawn by Meshz Plot	
	(b) Graph drawn by Surf Plot	
	(c) Graph drawn by Surfc Plot	
	(d) Graph drawn by Surf1 Plot	
<b>Figure 2.7</b>	F207: Inverted Crosshut Function	22
	(a) Graph drawn by Meshz Plot	
	(b) Graph drawn by Surf Plot	
	(c) Graph drawn by Surfc Plot	
	(d) Graph drawn by Surf1 Plot	



<b>Figure 2.8</b>	F208: Umbrella Function	23
	(a) Graph drawn by Meshz Plot	
	(b) Graph drawn by Surf Plot	
	(c) Graph drawn by Surfc Plot	
	(d) Graph drawn by Surf1 Plot	
<b>Figure 2.9</b>	F209: Inverted Umbrella Function	23
	(a) Graph drawn by Meshz Plot	
	(b) Graph drawn by Surf Plot	
	(c) Graph drawn by Surfc Plot	
	(d) Graph drawn by Surf1 Plot	
<b>Figure 2.10</b>	F210: Flower Function	24
	(a) Graph drawn by Meshz Plot	
	(b) Graph drawn by Surf Plot	
	(c) Graph drawn by Surfc Plot	
	(d) Graph drawn by Surf1 Plot	
<b>Figure 2.11</b>	F211: Royalbowl Function	24
	(a) Graph drawn by Meshz Plot	
	(b) Graph drawn by Surf Plot	
	(c) Graph drawn by Surfc Plot	
	(d) Graph drawn by Surf1 Plot	

\*\*\*

# ACKNOWLEDGEMENTS

It gives immense pleasure to express my heartfelt gratitude to my supervisor Prof Munindra Borah, Department of Mathematical Sciences, Tezpur University, without his constant inspiration and encouragement, the present work would not have been possible. When I approached him with the idea for a dissertation, he gave me the direction to approach the problem and taught me the research skill. Whenever I came to him with this problem he explained me the research methodology. During my stay in the department he extended all support and guidance and showed the direction at every chapter I used to complete. His valuable suggestions helped me to design my synopsis, abstract and finally the manuscript of this dissertation.

Special thanks go to Prof N D Baruah, Head, Department of Mathematical Sciences, Tezpur University for all kind of support during my stay in the department. Sincere thanks to all the faculty members, staffs and research scholars of the Mathematical Sciences department.

Thanks and gratitude is owed to many individuals for their help in various ways in the completion of this thesis. It gives me the immense pleasure to acknowledge all those who have been the driving force in completing this dissertation.

In the course of my research I met many distinguished personality with good research capabilities. I like to thank Prof S K Mishra Department of Economics, North Eastern Hill University with whom I used to discuss the topic and gave me the direction of work and explained the research technique. When I was still looking to frame the direction of my work he showed me the way to go ahead. I would like to thank Prof H K Mukherjee, Department of Mathematics, North Eastern Hill University for his helping hand in the process of my research.

I thank Prof K Deb, Director, Kanpur Genetic Algorithm Laboratory (KANGAL), IIT Kanpur to introduce me to the topic and allow me to use the resources of KANGAL. I would also like to thank all the members of KANGAL during my visit to the LAB. They also welcomed me in their laboratory in IIT Kanpur whenever I visited to IIT Kanpur.

Special thanks to my Principal Dr. S. R. Lyndem, Union Christian College to support me morally to complete my research work. When I was in my tough period of my work he advised me to go for Faculty Development Program (FDP) Fellowship from University Grant Commission (UGC), North East Regional Office (NERO). He helped me in communicating with UGC, NERO and supported me morally till the award of the fellowship.

I would like to thank Mr. S. C. Ray, Dy. Secretary UGC, NERO Guwahati, for granting me the Teachers fellowship under the faculty development programme to complete my work. I would also like to thank Department of Higher and Technical Education, Government of Meghalaya for approving my leave application towards the completion of my research work.

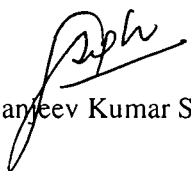
My thank goes to Mr. Joseph Mathew, Head Department of Mathematics and Dr. M. I. Devi, Lecturer Department of Mathematics Union Christian College for moral support and taking all the pain of the department during my absence. I would like to thank all my fellow colleagues of Union Christian College who supported me directly or indirectly.

I would like to thank Col. A. K. Singh, Officer Commanding, and 934 Ambulance with whom I stayed for long time in Tezpur, and who supported me morally during my stay in Tezpur.

Last but not the least I would like to thank my wife Mamta Singh and daughter Sanskriti Singh who supported me through out my work and took all the pain during my absence. I would also like to thank my parents, brother and sisters who supported me morally to complete my PhD work.

Place: Napaam, Tezpur

Date: 26 / 04 / 2011

  
Sanjeev Kumar Singh

## List of Publications

1. Singh, S. K. and Borah, M. A comparative study of Repulsive Particle Swarm Optimization and Simulated Annealing on some Numerical Benchmark Problems, *International Journal of Computational Intelligence Research*, **5(1)**, 75-82, 2009.
2. Singh, S. K., Borah, M. and Jha, U. C. Comparative study of modified differential evolution and genetic algorithm on some nonlinear, non convex & noisy test functions. *International Journal of Mathematical Modeling Simulations and Application*, **2(2)**, 156-162, 2009.
3. Singh, S. K. and Borah, M. Performance of Population Based Meta-heuristics on Some Non-Convex Noisy Deceptive Benchmark Test Function.. *Global Journal Pure and Applied Mathematics*, **5(1)**, 9–14, 2009
4. Singh, S. K. and Borah, M. A Comparative Study of Swarm Intelligence Optimization Method and Evolutionary Optimization Method on Some Noisy Numerical Benchmark Test Problems. *International Journal of Computation and Applied Mathematics*, **4(1)**, 1-9, 2009

# CHAPTER 1

---

## Introduction

### 1.1 Empirical Study

Over the years many a class of stochastic search technique, have been developed for the purpose of complex optimization and many variants of these stochastic search techniques such as evolutionary algorithm, Swarm Intelligence, Differential Evolution and Simulated Annealing, the list can continue, and they have demonstrated the high performance optimizers on a class of optimization problem. The algorithmic development involves an iterative process where the performance assessment plays a crucial role in improving our understanding of such optimizers and interplay between its different components. The performance understanding will greatly aid in the future development of better evolutionary optimizers. Therefore as more advanced evolutionary algorithms are being designed, the issue of performance assessment has become increasingly important. However, the assessment of evolutionary algorithms capability is not a trivial task. Due to its stochastic nature, the capability of evolutionary algorithms cannot precisely determine before its actual application as discussed in Chiam et al (2007).

The most practical and effective means for assessing the performance of evolutionary optimizers is via an empirical study, where the evaluated algorithm will be applied to a set of test functions and the evolved solution will be taken as an indication of algorithmic performance. Although performance assessment can also be done via theoretical study as discussed in He and Yoo (2003). This approach lacks the practicality and flexibility of empirical investigation. Again due to the stochastic nature of evolutionary algorithms and its complex relationship with optimization problem, it is difficult, if not impossible, to establish any formal mathematical treatment of algorithmic performance. Hence researchers will either get lost in the swamp of complexity or resort a substantial simplifications before any analysis can be done. Due

to this limitation of the theoretical studies, performance assessment via empirical approach has been adopted

In this research pertinent to Empirical study has been focused on the development of new test functions and the performance assessment of the optimizers taken for study on this class of test problems. Also, in this thesis our attention is on unconstrained Global Optimization (GO) problems for which it can be guaranteed that the global minimizer lies within a limited region

Although much work have been done to improve the reliability of empirical studies, there are little or no discussions at all on how it should be conducted with adequate substantiality on their statements made on the performance and behavior of the evaluated algorithms. So, in this study we have designed empirical study in following manner. A new class of test functions have been developed and the features of these test functions have been discussed to show the complexity of the class of test functions. The optimum value of these test function have been calculated using the different optimizers with a set parameters and the results are being compared. Because of the complexity of the studies any statistical analysis is not considered except to check whether the optimizers are able to find the optimum value in a CPU time or not

Absence of convergence theorem or very little theoretic development towards the convergence criteria as well as the stopping conditions, but the ability to solve the very complex real life global optimization problems of the evolutionary algorithms also keeps motivating the researchers to study the performance of meta-heuristics (population based evolutionary algorithms) on the large set of test functions. This inspired us to study population based meta-heuristics algorithms empirically on some new test function and some benchmark test functions in this thesis. So, it is important to understand the global optimization problems and evolutionary algorithms popularly known as population based meta-heuristics. In the next paragraph we explain global optimization and population based optimization

Global Optimization (GO) is one of the interesting topics of operation research. It refers to finding the extreme values of a given non-convex function in a certain feasible region. Global Optimization problems are classified in two classes, unconstrained and constrained problems. While solving global optimization problems Dekkers and Aarts (1991) had made great gain from the interest in the interface between computer science

and operations research. Ali and Torn (2004) worked on population based meta-heuristics algorithm and presented the numerical results. Other researchers who also contributed towards global optimization techniques are Aluffi-Pentini et al (1985), Easom (1990), Jansson and Knuppel (1992,1994), Levy et al (1981), Schutte (2003), Torn and Zilinskas (1989), Van Iwaarden (1996), Pinter (1996) and Dixon and Szego (1975, 1978)

The global optimization problem formulated by Easom (1990) in terms of finding the point  $x$  in a solution space set  $X$  (called the feasible region) where a certain function  $f: X \rightarrow T$  (called the objective function), attains a minimum or a maximum  $T$  in any ordered set (usually a subset of  $R^n$ ). The set  $X$  is usually a subset of  $R^n$  defined by constraints  $g(x) \leq 0$  or  $g(x) \geq 0$ , where  $g$  is a set of  $m$  possible nonlinear functions of  $x$ . The external point  $x$  can then be written as  $(x_1, x_2, x_3, \dots, x_n)$ , and the  $x_i$ 's are sometimes called decision variables. It is often practically useful to express the variable bounds explicitly as  $x^L \leq x \leq x^U$  ( $x^L, x^U \in R^n$ ). Some of the variables may be constrained to only take integer values ( $x_i \in \mathbb{Z} \forall i$  in an index set  $\mathbb{Z} \subseteq \{1, 2, 3, \dots, n\}$ ). Horst and Pardalos (1995) defined the global optimization in Mixed-Integer Nonlinear Programming problem (MINLP) as follows

$$\begin{aligned} \min_x \quad & f(x) \\ & g(x) \geq b \\ & g(x) \leq b \\ & x^L \leq x \leq x^U \\ & x_i \in \mathbb{Z} \quad \forall i \in \mathbb{Z} \end{aligned}$$

Finding the global optimum for a problem becomes extremely difficult, when the problem is deceptive, non-convex, noisy and non-differentiable in nature. Most of the optimization techniques available in literature stuck to the local minima. So, here we have tried to study the heuristics or more precisely the population based meta-heuristics in solving global optimization problems and studies the performance empirically.

## 1.2 Evolutionary Algorithms and its Evolution

The idea of using simulated evolution to solve the engineering and design problems have been studied in the 1950s and 1960s. Three persons, Box (1958), Friedberg (1958) and Bremmermann (1962) floated the idea independently. In the 1960's Rechenberg (1965) introduced "Evolutionary Strategies" to optimize real valued parameters for devices such as airfoils. Fogel et al (1966) developed "Evolutionary Programming" a technique in which candidate solutions to given tasks were presented as a finite state machines which were evolved by randomly mutating their state-transition diagrams and selecting the fittest. Genetic Algorithm (GA) was introduced by Holland (1962). In contrast, evolutionary strategies and evolutionary programming, Holland (1975) studied the phenomenon of adaptation as it occurs in nature and to develop ways in which the mechanisms of natural adaptation might be imported into computer systems. His book "Adaptation in Natural and Artificial Systems presented the genetic algorithm as an abstraction under genetic algorithm. Davis (1991)'s book "Handbook of Genetic Algorithm" were instrumental in further development in genetic algorithms. Tsutsui and Fujimoto (1993) developed the forking Genetic Algorithm with blocking and shrinking modes which increased the speed of the algorithm considerably and again in Tsutsui et al (1997a) modified the forking genetic algorithm with the space division scheme. Bi-population scheme for Real coded Genetic Algorithms was the another concept introduced by Tsutsui et al (1997a). Area of genetic programming was developing in parallel and Koza (1992) published a book in "Genetic Programming". Deb (2001) published a book entitled "Multi-objective optimization using Evolutionary Algorithm" were instrumental in the development of multi-objective optimization using evolutionary algorithms.

The above research not only fueled interest in evolutionary computing but they also were instrumental in bringing the evolutionary programming, evolutionary strategies, and genetic algorithm concepts together in a way that fostered unity and an explosion of new and exciting forms of evolutionary computing. Development of evolutionary computing generation wise can be categorized as follows, the first generation could be evolutionary programming by Fogel (1967), genetic algorithm by Holland (1965) and evolutionary strategy by Rechenberg (1965) and Schwefel (1965, 1975, 1977, 1981). The second generation evolutionary computing are hybrid genetic search by Davis



(1987), “Genetic Evolution + Data Structures = Evolutionary Algorithm” by Michalewicz (1999), genetic evolution programs by Koza (1992) and Tabu Search by Glover (2006) The third generation evolutionary computings are artificial immune systems by Farmer et al (1986), memetic algorithms by Moscato (1989), ant colony Optimization by Dorigo (1992), cultural algorithms by Reynolds (1994), DNA Computing, similar to parallel computing which takes advantages of many different molecules of DNA to try many different possibilities at once, developed by Adleman (1994), particle swarm optimization by Kennedy and Eberhart (1995), estimation of distribution algorithms some times called probabilistic model-building genetic algorithm by Larrañaga and Lozano (2002) After this for most of the researchers it will be interesting to see the 4<sup>th</sup> generation of evolutionary algorithm

Above three simulated evolution techniques were further used by many researchers to solve the real life problems Fogel et al (1966) was concern with solving prediction problems Rechenberg (1965) and Schwefel (1968, 1975, 1977, 1981) were concerned in solving parameter optimization problems Holland (1962) was concerned in developing robust adaptive system Each of these researchers successfully developed appropriate evolutionary computing for their particular problem independently But among all the three evolutionary techniques, evolutionary computing became most popular technique In United States, Goldberg (1989) popularized genetic algorithms (family of evolutionary computing) by the book entitled “Genetic Algorithms in search, optimization and machine learning” This book explained the concept of genetic algorithm in such a way that a wide variety of engineers and scientist could understand and apply Goldberg (1989) defined genetic algorithm as a search algorithms based on the mechanics of natural selection and natural Michalewicz (1985) studied on genetic algorithm for numerical optimization and constraints, Price (1994) worked on genetic annealing and Tu and Yong (2004) worked on a robust stochastic genetic algorithm for global numerical optimization

Inspired by different natural intelligence, evolutionary computing community researchers developed other variants of evolutionary algorithms Kirkpatrick et al (1983) proposed the simulated annealing which exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure Simulated annealing is an optimization process based on the above physical process belongs to the population based meta-heuristics Inspired by the foraging behavior of

ants Dorigo (1992) proposed a class of swarm intelligence population based stochastic optimization technique called ant colony algorithm in his PhD thesis. Eberhart and Kennedy (1995) proposed another swarm intelligence computing called Particle Swarm optimization inspired by the social behavior of bird flocking or fish schooling. Particle Swarm Optimization (PSO) shares many similarities with evolutionary computation techniques such as genetic algorithm. Later on many variants of particle swarm appeared in publication, to name some of them are Liang and Suganthan (2005) who worked on dynamic multi-swarm particle swarm optimizer and Liang et al (2006) worked on comprehensive learning particle swarm optimizer. In tune of further development Storn and Price (1997) proposed Differential Evolution (DE) while solving the Chebychev polynomial fitting problem posed by Storn (1995) which later on published in a book by Price et al (2005).

Since the development of above family of evolutionary computing algorithms were lacking the theoretical base and missing convergence criteria, it became important to study the performance and robustness of the above techniques using large number of test problems. So, as a further study on evolutionary algorithms (same as evolutionary computing) collection of test functions started appearing. Chattopadhyay (1971) studied some class of test functions for optimization algorithms and also explained the method of generating test functions with certain specific properties. CUTE (Constrained and Unconstrained Testing Environment) is a suite for FORTRAN subroutines, scripts and test problems for linear and nonlinear optimizations. It is a large collection of test functions developed by Jorge et al (1981). More work on test function, Floudas and Pardalos (1987) published a collection of test problems for constrained optimization and unconstrained optimization algorithms. Nagendra (1997) published a catalogue of test functions to test the performance of the evolutionary algorithm. Liang (2005) worked on the novel composition test functions for numerical global optimization, Andrei (2008) has published a collection of test functions for unconstrained optimization. Addis and Locatelli (2007) studied a new class of test function for Global Optimization and gave a new direction to the Empirical study research.

As stated above in absence of strong convergence criteria, researchers started studying the performance and robustness of the evolutionary algorithms using the test functions. Ackley (1987) published the empirical study of vector function optimization. An

experimental study in non convex optimization was done by Styblinski and Tang (1990) Deb (1991) used genetic algorithm to optimize multi-modal functions Fogel (1996) published evolutionary computation towards a new philosophy of machine intelligence Michalewicz (1999) published a book entitled “Genetic Algorithms+ Data structure = Evolution program” which deals with the real life numerical problem and step by step experimental studies Jason and Konstantinos (2002) did experimental study of benchmarking test functions for Genetic Algorithm Hsieh (2006) studied the Particle Swarm as a guided evolution strategy for real parameter optimization, Lewis (2008) in a “Survey of Meta-heuristics technique”, argued that all these global optimization techniques falls under the evolutionary computing and known as population based meta- heuristics technique

### **1.3 Study Area**

The interaction between computer science and optimization has yielded new practical solvers for global optimization problems, called meta-heuristics as defined in Glover and Kochenberger (2002) The structures of meta-heuristics are mainly based on simulating nature and artificial intelligence tools Meta-heuristics mainly invoke exploration and exploitation search procedures in order to diversify the search all over the search space and intensify the search in some promising areas Therefore, meta-heuristics cannot easily be entrapped in local optima However, meta-heuristics are computationally costly and there is always a question whether one algorithm will perform and find the optimum value for all type of global optimization problems Since none of the meta-heuristics have the stopping criteria and strong convergence theorem, one has to study the performance of these meta-heuristics on a large number of test problems and if necessary develop new test problems

### **1.4. Population based Meta-heuristics**

The term “Meta-heuristics” was first proposed by Glover (1986) that contains all heuristics methods that show evidence of achieving good quality solutions for the problem of interest within an acceptable time Usually, meta-heuristics offer no guarantee of obtaining the global solutions Back et al (1991) did a survey of the evolutionary strategies, Bethke (1980) submitted a doctoral thesis on genetic algorithm

as function optimizer, and Bukin (1997) worked on minimizing multi model functions for continuous variables. Coello-Coello (1998) surveyed the multi-objective optimization techniques, Suganthan et al (2005) studied the problem definition and evolution criteria in real parameter optimization, Srinivas and Deb (1994) studied non dominated multi-objective function optimization using non-dominated sorting Genetic Algorithm and Huang et al (2006) worked on swarm optimizer. Dennis and Schnabel (1983) worked on numerical methods for unconstrained optimization. Eberhart et al (1996) published a book on "Computational Intelligence" where he explained the problem solving capability of population based meta-heuristics. Another book who explained the population based meta-heuristics is "Method for Unconstrained Optimization Problem" published by Kowalik and Osborne (1968).

Glover and Kochenberger (2002) classified meta-heuristics into two classes, population-based methods and point-to-point meta-heuristics methods. In the latter methods, the search invokes only one solution at the end of each iteration from which the search will start in the next iteration. On the other hand, the population-based methods invoke a set of many solutions at the end of each iteration. Ali and Torn (2004) explained how genetic algorithms are population based meta-heuristics and tabu search as point to point Meta-heuristics.

Although there are many examples of meta-heuristics, in our study we are going to consider the genetic algorithm, particle swarm optimization, differential evolution and simulated annealing as population based meta-heuristics for our empirical study.

### **1.4.1 Pseudo codes of Evolutionary Computations**

#### **(a) Genetic Algorithm**

Genetic algorithm is search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of fittest among the string structures with structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search.

Genetic Algorithm (GA) work by evolving a population of individuals over a number of generations. A fit value is assigned to each individual in the population, where the fitness computation depends on the application. For each generation, individuals are selected from the population for reproduction, the individual's crosses to generate the new individuals, and new individuals are mutated with some low

mutation probability The new individuals may completely replace the old individuals in the population, with a distinct generations evolved by Goldberg (1989)

*Outline of Basic Genetic Algorithm*

---

```
Choose the initial populations
Evaluate each individual's fitness
Determine population's average fitness
  repeat
    select best ranking individuals to reproduce
    mate pairs at random
    apply crossover operator
    apply mutation operator
    evaluate each individual's fitness
    Determine population's average fitness
  Until termination condition is met (e.g. One individual has
the desired fitness or enough generations have been completed)
```

---

More elaborately

- Step 1** Choose a coding to represent problem parameters (Our program uses the real coding of the population), a selection operator, a crossover operator, and a mutation operator. Choose population size,  $n$ , crossover probability,  $p_c$  and mutation probability  $p_m$ . Initialize a random population of strings of size  $l$ . Choose a maximum allowable generation number  $t_{max}$ . Set  $t = 0$
- Step 2** Evaluate each string in the population. The program we have used uses the evolution via survival of fittest
- Step 3** If  $t > t_{max}$  or other termination criteria satisfied, Terminate
- Step 4** Perform reproductions on the population. The selection scheme we have used is the tournament selection with a shuffling technique for choosing random pairs for mating
- Step 5** Perform crossover on random pairs of strings, we have used single point crossover and there is option for uniform crossover
- Step 6** Perform mutations on every string, the program the jump mutation and creep mutation. Niching (Sharing) is also done
- Step 7** Evaluate strings in the new population. Set  $t = t + 1$  and go to step 3

## (b) Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a form of swarm intelligence developed by Kennedy and Eberhart (1995) and Kennedy et al (2001). This is modelled by particles in multidimensional space that have a position and a velocity. These particles are flying through hyperspace and have two essential reasoning capabilities: memory of their own best position and knowledge of the swarm's best. Members of a swarm communicate good positions to each other and adjust their own position and velocity based on these good positions. There are two main ways this communication is done:

- a global best that is known to all
- "neighborhood" bests where each particle only communicates with a subset of the swarm about best positions

There are several different realizations of particle swarm optimization. Common to all these realizations is the repulsion between the particles. This can prevent the swarm from being trapped in local minima, which would cause a premature convergence and would lead the optimization algorithm to fail to find the global optimum. The other variants use a dynamic scheme. In Repulsive Particle Swarm Optimization as it appears in Mishra (2006), the future velocity  $v_{i,t+1}$  of a particle at position  $x$  with a recent velocity  $v_i$  is

$$v_{i,t+1} = \omega v_i + ar_1(\hat{x}_i - x_i) + abr_2(\hat{x}_h - x_i) + \omega cr_3 z$$

$$x_{i,t+1} = x_i + v_{i,t+1}$$

where,

- $x$  is the position and  $v$  is the velocity of the individual particle. The subscripts  $i$  and  $i+1$  stand for the recent and the next (future) iterations, respectively
- $r_1, r_2, r_3$  are random numbers,  $\in [0,1]$ ,  $a, b, c$  are constants
- $\omega$  is inertia weight,  $\in [0.01, 0.7]$ ,  $z$  is a random velocity vector
- $\hat{x}$  is the best position of a particle,  $x_h$  is best position of a randomly chosen other particle from within the swarm

## (c) Simulated Annealing

Annealing refers to the cooling process of liquid or solid and the analysis of the behavior of substances as they cool, when the temperature reduces, the mobility of molecules reduces, with the tendency that molecules may align themselves in a

crystalline structure. The aligned structure is the minimum energy state of the system. To ensure that this alignment is obtained, cooling must occur at a sufficiently slow rate. If the substance is cooled at a too rapid rate, an amorphous state may be reached.

This idea of alignment in crystalline structure of the substance is being used in optimization process. So, speaking in mathematical terminology the minimum energy of the system represents the minimum of an objective function. Hence simulated annealing is an algorithmic implementation of the cooling process to find the optimum of an objective function developed by Kirkpatrick et al. (1983) and Kirkpatrick (1984). Also, Corana et al. (1987) studied simulated annealing to minimize the multimodal function of continuous variables. Davis and Steenstrup (1986) published the overview of genetic algorithm and simulated annealing.

**Simulated Annealing Algorithm:**  
 Create initial solution  $x(0)$ ;  
 Set initial temperature,  $T(0)$ ;  
 $t=0$ ;  
 repeat  
     Generate new solution,  $x$ ;  
     Determine quality  $f(x)$ ;  
     Calculate acceptance probability  

$$P_{ij} = \begin{cases} 1 & \text{if } f(x_j) < f(x_i) \\ e^{-\frac{f(x_j) - f(x_i)}{c_b T}} & \text{otherwise} \end{cases}$$
  
     where  $c_b > 0$ .  
     if  $U(0,1) \leq$  acceptance probability then  
          $x(t) = x$ ;  
     end  
 until stopping condition is true;  
 Return  $x(t)$  as the solution;

#### (d) Differential Evolution

Differential Evolution is a population-based search strategy very similar to evolutionary algorithm. The main difference is in the reproduction where the offspring is created from three parents using arithmetic cross-over operator. Differential Evolution is defined for floating-point representation of individuals. The method goes in following ways as defined by Storn and Price (1995).

- For each parents  $p_i(t)$ , of generation  $t$ ,

an offspring,  $o_j(t)$  is created by the expression  $o_j(t) = p_j(t) + \chi(p_j(t) - p_j(t))$

- for any three randomly selected parents for  $i_1 \neq i_2 \neq i_3$  and  $i_1, i_2, i_3 \sim U(1, n_p)$  By selecting a random number  $u \sim U(1, n_p)$  where  $n_p$  the number of genes or parameters of a single chromosome are

Then all parameters  $j=1, \dots, n_p$ , if  $U(0,1) < P_u$ , or if  $j=u$ , we get the offspring  $o_j(t)$  otherwise  $o_j(t) = p_j(t)$  Here  $P_u$  is the probability of reproduction with  $P_u \in [0,1]$  and  $\chi$  is the scaling factor with  $\chi \in (0, \infty)$ ,  $o_j(t)$  and  $p_j(t)$  are the  $j^{\text{th}}$  parameter of the offspring and parents

The algorithm for the simple differential evolution goes like this

---

```

Initialize and evaluate population P
While (not done) {
  for (i = 0, i < ps, i++) {
    Create candidate C[i]
    Evaluate C[i]
    if (C[i] is better than P[i])
      P0[i] = C[i]
    else
      P0[i] = P[i]
  }
  P = P0
}

Create candidate C[1]
Randomly select parents P[11], P[12], and P[13] where i1, i2, and i3
are different
Create initial candidate C1[1] = P[11] + fact (P[12] - P[13])
Create final candidate C[1] by crossing over the genes of P[1] and
C1[1] as follows
  For(j=0, j < N, j++){
    If (U(0,1) < pc)
      C[1][j] = c1[1][j]
    else
      C[1][j] = P[1][j]
  }
}
Here we have randomized the scaling factor as fact = 0.5*rand

```

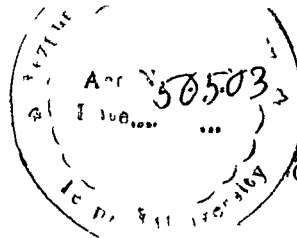
---

Though there more evolutionary algorithms like Ant colony search by Dorigo (1992), Tabu Search by Glover (1986) which are extensively used to solve the combinatorial optimization problem but we have not considered for our studies

## 1.5 Objectives

The objective of our study is to work on the development of the new set of test functions which are more difficult, deceptive, non-convex and noisy in nature. The





newly developed test functions are to be coded in MATLAB to give the visual presentation and analyze the difficulty level of the functions such as the noise, deceptiveness and many local minima from visual presentation Performance of optimizers mentioned in introduction has been studied and results have been recorded in tabular form for further analysis In short, below four point is objectives below

- (i) In literature survey we have collected the benchmark test functions All the functions have been coded in MATLAB to revisit the visual presentation of these test functions The graphs drawn of the benchmark test functions and that have helped the empirical study of the evolutionary optimizers
- (ii) The second objective was to develop some new test functions The optimum value of the newly developed test functions using genetic algorithm, particle swarm optimization, differential evolution and simulated annealing have been calculated The study has been done with different set of population size and the result is recorded in tabular form
- (iii) The comparative study of optimizers mentioned above in objective (ii) using the bench mark test functions have been considered in our investigation and results have been recorded in tabular form and conclusions are drawn
- (iv) The conclusion of the study has been drawn looking into the performance of above optimizers on the new set of test functions

## 1.6 Outline of the Thesis

The thesis is consist of five chapters Three APPENDIX has been given at the end of the thesis APPENDIX A contains the large collection of Benchmark test functions APPENDIX B contains the MATLAB code to represent the benchmark and test functions graphically APPENDIX C contains the executable code of new test functions Below the brief introduction of all the five chapters are presented

The **chapter-1** is an introductory one which highlights the global optimization with the progress and development of the evolutionary computations in finding the global optimum values The chapter also discusses the types of evolutionary algorithms with its pseudo codes of the optimizers such as differential evolution, genetic algorithm,

particle swarm optimization and simulated annealing. Some optimizers like ant colony and tabu search to solve the combinatorial optimization have been discussed with its pseudo codes. We have presented our objectives clearly here.

The **chapter -2** discusses some newly developed unconstrained test functions in detail. Here we have placed three sets of newly developed test functions with visual presentations. From function F201 to F211 is one set. Second set is the generalization of the first set. Third set is the generalization and extension of some Benchmark test functions. Using the Meshz Plot, Surf Plot, Surfz Plot and Surf1 Plot of the MATLAB plotter we have drawn four graphs to have the different visibility to guess the optimum points.

In **chapter-3** we set up the experiments with four optimizers such as differential evolution, genetic algorithm, particle swarm optimization and simulated annealing to record the results for empirical study. In the first table we have recorded the results of newly developed eleven test functions (1<sup>st</sup> set) with the two sets of population size 50 and 250. The second, third and fourth table contains results obtained with two sets of population size (50 and 250) and the result is recorded at the iterations such as 100, 200, 300, 400, 500, 600, 700 and 1000.

**Chapter-4** is a reprint of our published work. A comparative study of evolutionary algorithm is performed using the benchmark test functions and the results are published in Singh and Borah (2009) and Singh et al (2009). The benchmark test function again coded in MATLAB to get the optimum value from the optimizers and results are recorded in tabular form and conclusions are drawn.

**Chapter-5** is the conclusion of the thesis. The direction of future research is also given here. Results have been validated from the empirical study on evolutionary algorithm using newly developed test functions which have been tabulated in chapter 3. The conclusions are also drawn from the results tabulated in chapter 4 of study on the evolutionary algorithm using the benchmark test functions. In short our finding can be stated as "It is impossible to have an evolutionary algorithm which can outperform on all class of problems in the domains". Hence the study validates the "No Free Lunch Theorem" by Wolpert and Macready (1997) and Ho and Pepyne (2002).

\*\*\*\*\*

# CHAPTER 2

---

## New Test Functions for Unconstrained Global Optimization

### 2.1 Introduction

A set of benchmark test problems have been considered for testing the Evolutionary Algorithm. Testing the algorithms with a test function with mild difficulty may not validate the algorithm. So, it is important to consider the wide variety of test functions with the degree of difficulties. In the field of global optimization there exist a set of test functions with a limited dimension and mild difficulties. Therefore testing any Global Optimization (GO) problems with those algorithms may be not appropriate way to validate the algorithm. We have collected the large class of test function to validate the Global optimization. The test functions have been defined with the magnitude of difficulties. Many other researchers have worked in generation of test functions and collection of benchmark test functions. Hock et al (1981), More (1981), Dennis (1985), Averick et al (1991), Back et al (1991), Bongartz et al (1995), De Jong et al (1999) worked on the test problem generator for non stationary environment, Shcherbina (2002), Shcherbina et al (2003), Adorio (2005) and Mishra (2006) are among those who worked on the collection of test functions or generation of test functions to check the performance robustness of the evolutionary algorithms. The difficulties of global optimization problem depend on many factors. Among the most relevant ones is the size of basin of attraction of the Global Optimizer, the shape of the function around the global optimizer, the classical example of the being the Rosenbrock function where the minimum point is inside a long narrow and a parabolic-shaped flat valley, which makes convergence difficult, dimension and high multimodality.

In this chapter, test problems are presented to test the performance of evolutionary algorithms considered in the thesis. These benchmark test functions are deceptive in nature, non-convex noisy. In most of the cases of this test problem traditional method is not able to find the optimum value, whereas these algorithms are able to find the

optimum values. Hence, the study will investigate that which algorithm performs better on these test suits given in section 2.3. The comparative study has been done using the newly developed test functions. For instances, the five test functions constructed by De-Jong (1975), popularly known as De-Jong's five test suit, four are uni-modal containing only one optimum point, and other test functions are multi-modal containing multi optimum point. Sphere function is smooth, uni-modal, strongly convex and symmetric, but has only one optimum point. Rosenbrock is considered to be difficult, because it has a very narrow ridge, the tip of the ridge is very sharp, and it runs around a parabola. Algorithms that are not able to discover good directions underperform in this problem. Step function is the representative of the problem of flat surfaces. It is piecewise continuous step function. Flat surfaces are obstacles for optimization algorithms, because they do not give any information as to which direction is favorable. Unless an algorithm has variable step sizes, it can get stuck on one of the flat plateaus. The background idea of the step function is to make the search more difficult by introducing small plateaus to the topology of an underlying continuous function. Quartic function is a simple uni-modal function padded with noise. The Gaussian noise makes sure that the algorithm never gets the same value on the same point. Algorithms that do not perform well on this test function will perform poorly on noisy data. Foxholes function is an example of a function with many local optima. Many standard optimization algorithms get stuck in the first peak it finds. The Schwefel, Rastrigin, Griewangk functions are typical examples of non-linear multimodal functions. Rastrigin's function is a fairly difficult problem for genetic algorithms due to the large search space and large number of local minima. Rastrigin has a complexity of  $O(n \ln(n))$ , where  $n$  is the number of the function parameters. This function contains millions of local optima in the interval of consideration. Schwefel's function is somewhat easier than Rastrigin's function, and is characterized by a second-best minimum which is far away from the global optimum. These are some of the features of benchmark test functions.

Below the eleven new test functions are being introduced of different complexity and difficulties. Some functions are noisy in nature. Some are dented, non-differentiable and deceptive in nature. The features of each test functions are explained in the section 2.4. The visual presentation of these functions gives the characteristic and some idea about the number of optimum (i.e. local minima or local maxima) and also the

complexity of the function. It can be seen from the graphical presentation that the test functions are highly multimodal. Large collection of benchmark test functions have been presented in APPENDIX A with its characteristics in a most exhaustive manner, but this may not be the complete list of test functions.

## 2.2 Purpose of developing new test functions

In the field of optimization definition of test problems is an important and non-trivial task. Test problems should reflect the wide variety of difficulties encountered when solving practical problems and are essential in validating algorithms as discussed in Addis and Locatelli (2007). In the field of Global Optimization there exists the old class of test functions (List of 200 test functions are provided in Appendix –A). Most of these test functions are limited in dimension and mild difficulty level and Constrained and Unconstrained Testing Environment (CUTE) which is a collection of generalized version of the test functions no other generalized version of test functions are not available. The test functions presented in Hock-Schittkowski (1981) and Schittkowski (1987) employed for constrained local optimization but many of these test problems have several local minimizers with different function values and thus also appropriate test functions for Global optimization methods. In Schoen (1983) a class of test functions is proposed whose global minimizer is priori known, whose smoothness is controllable by means of set of parameters, and for which the number and location of stationary points are controllable by users.

Unfortunately, these problems are of limited dimension and of mild difficulty. Therefore testing on them is not an appropriate way to validate the Global Optimizers (GO). The purpose of developing the new test functions was to develop a class of test functions which are unimodal or multimodal and deceptive in nature and its difficulty level can be controlled.

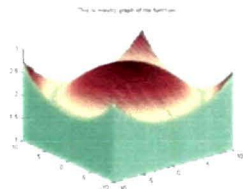
The interest for new and widely recognized Global Optimization test problems emerged in a number of recent publications such as a book published by Floudas et al (1999), papers published by Gaviano et al (2003), Lavor and Maculan (2004), Neumaier et al (2005), Pinter (2002), and the global optimization web site (GO-site 2005).

### 2.3 Some newly developed test functions

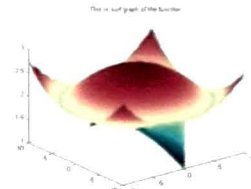
Below newly developed test functions are given. The numbers have been written from 201 because in APPENDIX A a collection 200 benchmark test functions are given. The names of these functions have been coined only looking at the visuals of the colored dimensional graph. The visual presentations have created using the Meshz Plot, Surf Plot, Surfz Plot and SurfI Plot of MATLAB 7.1.

**201. Tortoise function:** The function is defined as in the domain  $x, y \in (-10,10)$

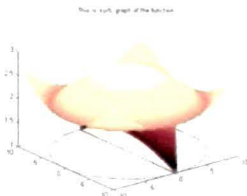
$$f(x, y) = \left| \sin(x) e^{\frac{|100-(x^{2/3}+y^{2/3})^{0.05}|}{\pi}} \right| \left| \cos(x) e^{\frac{|100-(x^{2/3}+y^{2/3})^{0.05}|}{\pi}} \right|$$



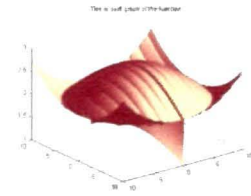
(a) Graph drawn by Meshz Plot



(b) Grpah drawn by Surf Plot



(c) Graph drawn by Surfz Plot

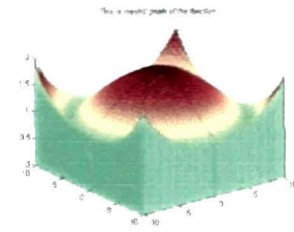


(d) Graph drawn by SurfI Plot

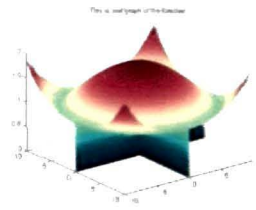
**Figure: 2.1** Tortoise Function

**202. I-Crosscap Function:** This function is defined as

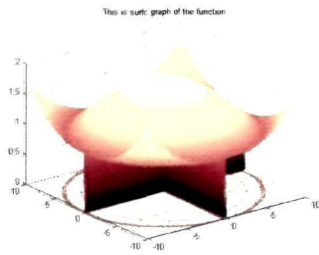
$$f(x, y) = \left| \sin(xy) e^{\frac{|100-(x^2+y^2)^{0.01}|}{\pi}} \right| \left| \cos(xy) e^{\frac{|100-(x^2+y^2)^{0.01}|}{\pi}} \right|. \text{ Where } x, y \in (-10,10)$$



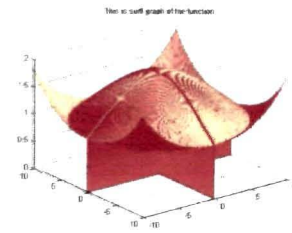
(a) Graph drawn by Meshz Plot



(b) Graph drawn by Surf Plot



(c) Graph drawn by Surfz Plot

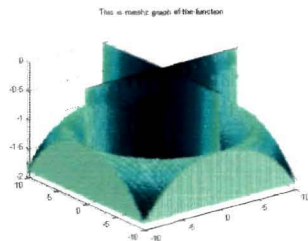


(d) Graph drawn by Surf1 Plot

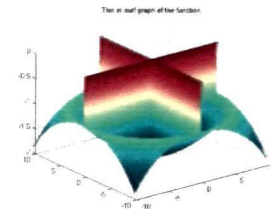
**Figure: 2.2** Crosscap Function

**203. Crosscap Function:** The function is defined as

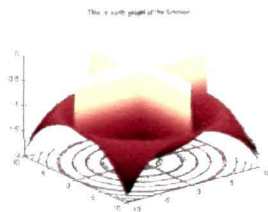
$$f(x, y) = \left| \sin(xy) e^{\frac{100-(x^2+y^2)}{\pi}} \right| \left| \cos(xy) e^{\frac{100-(x^2+y^2)}{\pi}} \right|, \text{ where } x, y \in (-10, 10)$$



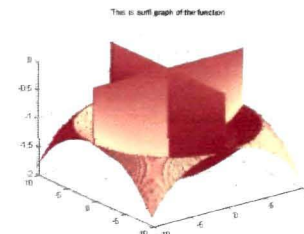
(a) Graph drawn by Meshz Plot



(b) Graph drawn by Surf Plot



(c) Graph drawn by Surfz Plot

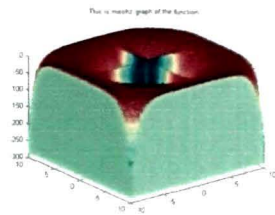


(d) Graph drawn by Surf1 Plot

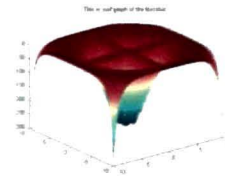
**Figure: 2.3** Inverted Crosscap Function

**204. Four-hole table Function:** The function is defined as

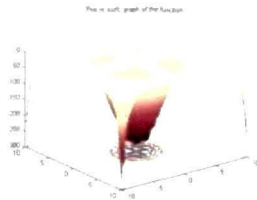
$$f(x, y) = \left| \begin{array}{c} \sin(xy)e^{\frac{100-(x^2+y^2)^{0.099}}{\pi}} \\ \cos(xy)e^{\frac{100-(x^2+y^2)^{0.099}}{\pi}} \end{array} \right| \text{ in the domain of } x, y \in (-10, 10).$$



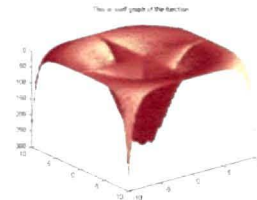
(a) Graph drawn by Meshz Plot



(b) Graph drawn by Surf Plot



(c) Graph drawn by Surfz Plot

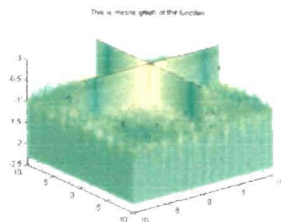


(d) Graph drawn by Surf1 Plot

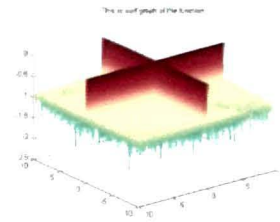
**Figure: 2.4** Four-hole tables Function

**205. Cross on rough ceiling Function:** The function is defined as

$$f(x, y) = \left| \begin{array}{c} \sin(xy)e^{\frac{100-(x^2+y^2)^{0.099}}{\pi}} \\ \cos(xy)e^{\frac{100-(x^2+y^2)^{0.099}}{\pi}} \end{array} \right| \text{ in the domain of } x, y \in (-10, 10).$$

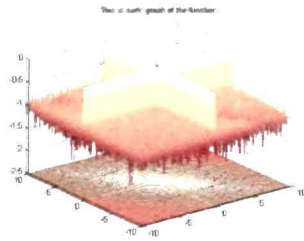


(a) Graph drawn by Meshz Plot

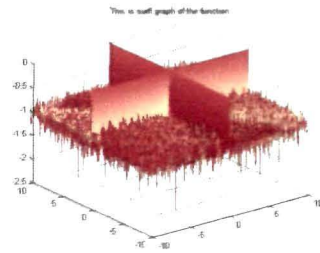


(b) Graph drawn by Surf Plot





(c) Graph drawn by Surfz Plot

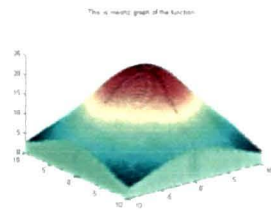


(d) Graph drawn by Surfz Plot

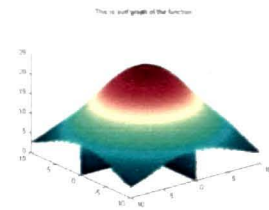
**Figure: 2.5** Cross on rough ceiling Function

**206. Crosshut Function:** This function is defined as

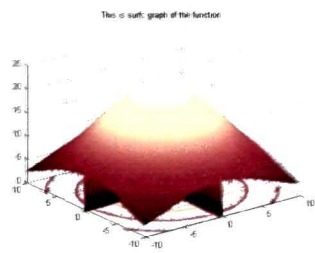
$$f(x, y) = \left| \sin(xy) e^{\frac{100 - (x^{2/3} + y^{2/3})^{0.05}}{\pi}} \right| \left| \cos(xy) e^{\frac{100 - (x^{2/3} + y^{2/3})^{0.05}}{\pi}} \right| \text{ in the domain of } x, y \in (-10, 10)$$



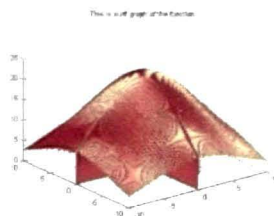
(a) Graph drawn by Meshz Plot



(b) Graph drawn by Surfz Plot



(c) Graph drawn by Surfz Plot

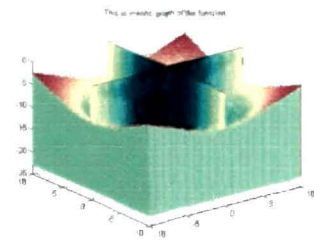


(d) Graph drawn by Surfz Plot

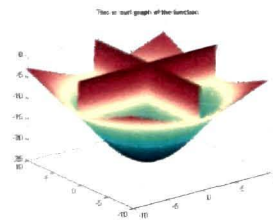
**Figure: 2.6** Crosshut Function

**207. Inverted Crosshut Function:** This function is defined as

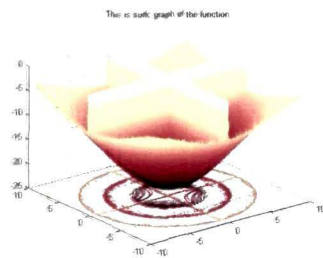
$$f(x, y) = - \left| \sin(xy) e^{\frac{100 - (x^{2/3} + y^{2/3})^{0.05}}{\pi}} \right| \left| \cos(xy) e^{\frac{100 - (x^{2/3} + y^{2/3})^{0.05}}{\pi}} \right| \text{ in the domain of } x, y \in (-10, 10)$$



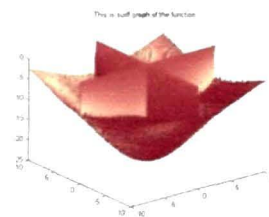
(a) Graph drawn by Meshz Plot



(b) Graph drawn by Surf Plot



(c) Graph drawn by Surfz Plot

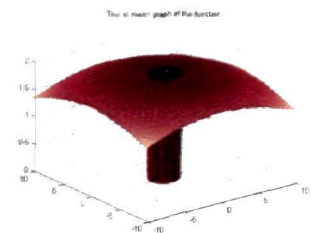


(d) Graph drawn by Surfz Plot

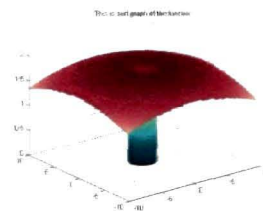
**Figure: 2.7** Inverted Crosshatch Function

**208. Umbrella Function:** The function is defined as

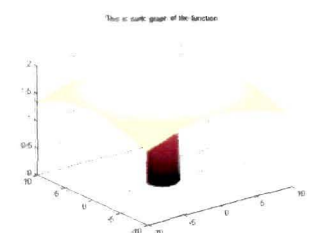
$$f(x, y) = \left\| \left\| x^{2/3} + y^{2/3} \right\| e^{\left\| \frac{100 - (x^{2/3} + y^{2/3})}{x} \right\|^{0.01}} \right\| \left\| \left\| x^{2/3} + y^{2/3} \right\| e^{\left\| \frac{100 - (x^{2/3} + y^{2/3})}{x} \right\|^{0.01}} \right\| \text{ in the domain of } x, y \in (-10, 10)$$



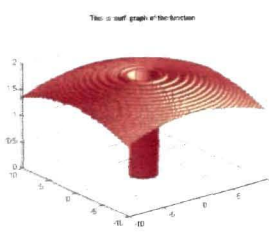
(a) Graph drawn by Meshz Plot



(b) Graph drawn by Surf Plot



(c) Graph drawn by Surfz Plot



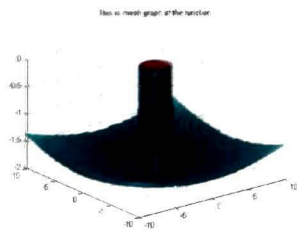
(d) Graph drawn by Surfz Plot

**Figure: 2.8** Umbrella Function

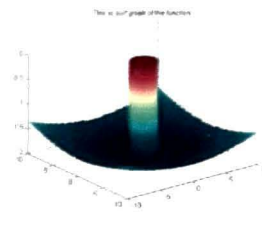
**209. Inverted-umbrella Function:** The function is defined as

$$y) = - \left| \left| x^{2/3} + y^{2/3} \right| e^{\left| \frac{100 - (x^{2/3} + y^{2/3})}{\pi} \right|^{0.01}} \right| \left| \left| x^{2/3} + y^{2/3} \right| e^{\left| \frac{100 - (x^{2/3} + y^{2/3})}{\pi} \right|^{0.01}} \right| \text{ in the domain of}$$

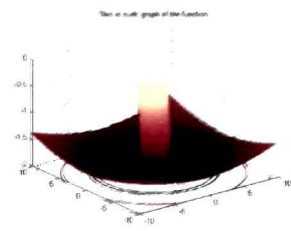
$$x, y \in (-10, 10)$$



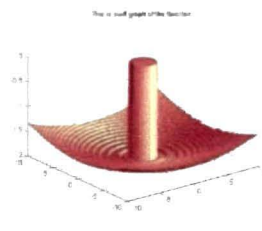
(a) Graph drawn by Meshz Plot



(b) Graph drawn by Surf Plot



(c) Graph drawn by Surfz Plot



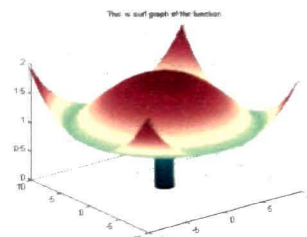
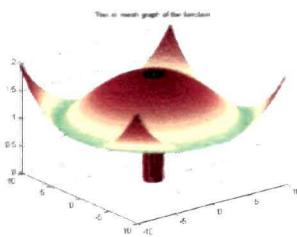
(d) Graph drawn by Surf1 Plot

**Figure: 2.9** Inverted Umbrella Function

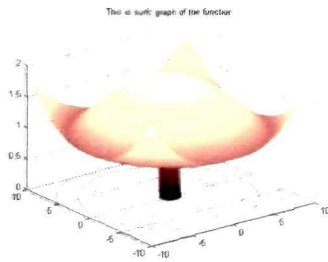
**210. Flower Function:** The function is defined as

$$f(x, y) = \left| \left| \sqrt{x^2 + y^2} \right| e^{\left| \frac{100 - (x^2 + y^2)}{\pi} \right|^{0.01}} \right| \left| \left| \sqrt{x^2 + y^2} \right| e^{\left| \frac{100 - (x^2 + y^2)}{\pi} \right|^{0.01}} \right| \text{ in the domain of}$$

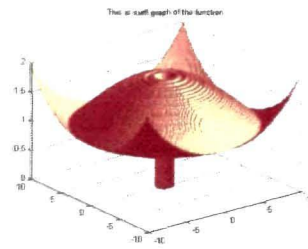
$$x, y \in (-10, 10)$$



(a) Graph drawn by Meshz Plot



(b) Graph drawn by Surf Plot



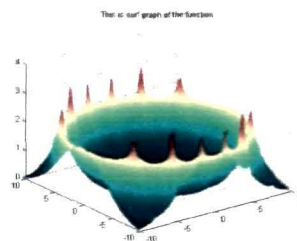
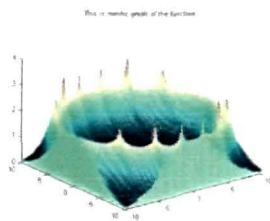
(c) Graph drawn by Surf Plot

(d) Graph drawn by Surf Plot

Figure: 2.10 Flower function

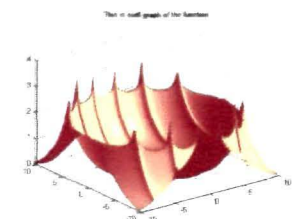
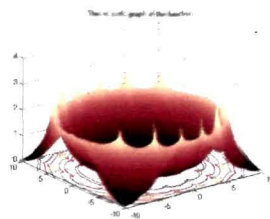
211. **Royalbowl Function:** This function is defined as

$$f(x) = \left| \sin(x) e^{\frac{|100-(x^2+y^2)^{-1}|}{x}} \right| + \left| \cos(x) e^{\frac{|100-(x^2+y^2)^{-1}|}{x}} \right|, \quad x, y \in (-10, 10)$$



(a) Graph drawn by Meshz Plot

(b) Graph drawn by Surf Plot



(c) Graph drawn by Surf Plot

(d) Graph drawn by Surf Plot

Figure: 2.11 Royalbowl function

### 2.3.1 Generalized version of test functions from 201- 211

The functions have been generalized from two variables to n number of variables. The optimum value of these functions can be calculated just by increasing the dimension of variable x.

**212. Generalized Tortoise Function:** The function is defined as

$$f(x) = \left| \sin \left( \prod_{i=1}^n x_i \right) e^{\left| \frac{100 - \sum_{i=1}^n v_i^{2/3}}{\pi} \right|^{100}} \right| \left| \cos \left( \prod_{i=1}^n x_i \right) e^{\left| \frac{100 - \sum_{i=1}^n v_i^{2/3}}{\pi} \right|^{100}} \right| \quad \text{in the domain of } -10 \leq x_i \leq 10$$

**213. Generalized Crosscap Function:** The function is defined as

$$f(x) = \left| \sin \left( \prod_{i=1}^n x_i \right) e^{\left| \frac{100 - \sum_{i=1}^n v_i^2}{\pi} \right|^{100}} \right| \left| \cos \left( \prod_{i=1}^n x_i \right) e^{\left| \frac{100 - \sum_{i=1}^n v_i^2}{\pi} \right|^{100}} \right| \quad \text{in the domain of } -10 \leq x_i \leq 10.$$

**214. Generalized four-hole table Function:** The function is defined as

$$f(x) = - \left| \sin \left( \prod_{i=1}^n x_i \right) e^{\left| \frac{100 - \sum_{i=1}^n v_i^2}{\pi} \right|^{100}} \right| \left| \cos \left( \prod_{i=1}^n x_i \right) e^{\left| \frac{100 - \sum_{i=1}^n v_i^2}{\pi} \right|^{100}} \right| \quad \text{in the domain of } -10 \leq x_i \leq 10.$$

**215. Generalized Cross on rough ceiling Function:** The function is defined as

$$f(x) = - \left| \frac{\sin \left( \prod_{i=1}^n x_i \right) e^{\left| \frac{100 - \sum_{i=1}^n v_i^2}{\pi} \right|^{100}}}{\cos \left( \prod_{i=1}^n x_i \right) e^{\left| \frac{100 - \sum_{i=1}^n v_i^2}{\pi} \right|^{100}}} \right| \quad \text{in the domain of } -10 \leq x_i \leq 10.$$

**216. Generalized Crosshut Function:** The function is defined as

$$f(x) = \left| \sin \left( \prod_{i=1}^n x_i \right) e^{\left| \frac{100 - \sum_{i=1}^n v_i^{2/\lambda}}{\pi} \right|^{0.05}} \right| \left| \cos \left( \prod_{i=1}^n x_i \right) e^{\left| \frac{100 - \sum_{i=1}^n v_i^{2/\lambda}}{\pi} \right|^{0.05}} \right| \quad \text{in the domain of } -10 \leq x_i \leq 10$$

**217. Generalized Inverted Crosshut function:** The function is defined as

$$f(x) = - \left| \sin \left( \prod_{i=1}^n x_i \right) e^{\left| \frac{100 - \sum_{i=1}^n v_i^{2/\lambda}}{\pi} \right|^{0.05}} \right| \left| \cos \left( \prod_{i=1}^n x_i \right) e^{\left| \frac{100 - \sum_{i=1}^n v_i^{2/\lambda}}{\pi} \right|^{0.05}} \right| \quad \text{in the domain of } -10 \leq x_i \leq 10.$$

**218. Generalised Umbrella Function** The function is defined as

$$f(x) = \left| \left[ \sum_{i=1}^n x_i^{2/\lambda} \right] e^{\left| \frac{100 - (\sum_{i=1}^n v_i^{2/\lambda})}{\pi} \right|^{0.01}} \right| \left| \left[ \sum_{i=1}^n x_i^{2/\lambda} \right] e^{\left| \frac{100 - (\sum_{i=1}^n v_i^{2/\lambda})}{\pi} \right|^{0.01}} \right| \quad \text{in the domain of } x_i \in (-10, 10)$$

**219. Generalised Inverted Umbrella Function** The function is defined as

$$f(x) = - \left| \left[ \sum_{i=1}^n x_i^{2/\lambda} \right] e^{\left| \frac{100 - (\sum_{i=1}^n v_i^{2/\lambda})}{\pi} \right|^{0.01}} \right| \left| \left[ \sum_{i=1}^n x_i^{2/\lambda} \right] e^{\left| \frac{100 - (\sum_{i=1}^n v_i^{2/\lambda})}{\pi} \right|^{0.01}} \right| \quad \text{in the domain of } x_i \in (-10, 10)$$

**220. Generalized Flower Function:** The function is defined as

$$f(x) = \left| \left| \left[ \sqrt{\sum_{i=1}^n x_i^2} \right] e^{\left| \frac{100 - (\sum_{i=1}^n v_i^2)}{\pi} \right|^{0.01}} \right| \left| \left[ \sqrt{\sum_{i=1}^n x_i^2} \right] e^{\left| \frac{100 - (\sum_{i=1}^n v_i^2)}{\pi} \right|^{0.01}} \right| \right| \text{ in the domain of } x, \in (-10, 10)$$

**221. Generalized Royalbowl Function:** This function is defined as

The function is defined as  $f(x) = \left| \sin(x) e^{\left| \frac{100 - \sum_{i=1}^n v_i^{2/11}}{\pi} \right|^{-1}} \right| \left| \cos(x) e^{\left| \frac{100 - \sum_{i=1}^n v_i^{2/11}}{\pi} \right|^{-1}} \right|$  in the domain of  $-10 \leq x, \leq 10$

### 2.3.2 Extended version of test functions from 201 to 211

**222. Extended Tortoise Function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} \left| \sin(x_{2i-1}) e^{\left| \frac{100 - (v_{2i-1}^{2/11} + v_{2i}^{2/11})}{\pi} \right|} \right| \left| \cos(x_{2i-1}) e^{\left| \frac{100 - (v_{2i-1}^{2/11} + v_{2i}^{2/11})}{\pi} \right|} \right| \text{ in the domain of } -10 \leq x, \leq 10$$

**223. Extended Crosscap Function:** This function is defined as

$$f(x) = \sum_{i=1}^{n/2} \left| \sin(x_{2i-1} x_{2i}) e^{\left| \frac{100 - (v_{2i-1}^2 + v_{2i}^2)^{0.01}}{\pi} \right|} \right| \left| \cos(x_{2i-1} x_{2i}) e^{\left| \frac{100 - (v_{2i-1}^2 + v_{2i}^2)^{0.01}}{\pi} \right|} \right|. \text{ Where } x, y \in (-10, 10)$$

**224. Extended Inverted Crosscap Function:** This function is defined as

$$f(x) = -\sum_{i=1}^{n/2} \left| \sin(x_{2i-1}x_{2i}) e^{\left| \frac{100-(v_{2i-1}^2+v_{2i}^2)^{0.01}}{\pi} \right|} \right| \left| \cos(x_{2i-1}x_{2i}) e^{\left| \frac{100-(v_{2i-1}^2+v_{2i}^2)^{0.01}}{\pi} \right|} \right|. \text{ Where}$$

$x, y \in (-10, 10)$ .

**225. Extended Four-hole table Function** The function is defined as

$$f(x) = -\sum_{i=1}^{n/2} \left| \sin(x_{2i-1}x_{2i}) e^{\left| \frac{100-(v_{2i-1}^2+v_{2i}^2)^{0.09}}{\pi} \right|} \right| \left| \cos(x_{2i-1}x_{2i}) e^{\left| \frac{100-(v_{2i-1}^2+v_{2i}^2)^{0.09}}{\pi} \right|} \right| \text{ in the domain of}$$

$x, y \in (-10, 10)$ .

**226. Extended cross on rough ceiling Function:** The function is defined as

$$f(x) = -\sum_{i=1}^{n/2} \left| \frac{\sin(x_{2i-1}x_{2i}) e^{\left| \frac{100-(v_{2i-1}^2+v_{2i}^2)^{0.09}}{\pi} \right|}}{\cos(x_{2i-1}x_{2i}) e^{\left| \frac{100-(v_{2i-1}^2+v_{2i}^2)^{0.09}}{\pi} \right|}} \right| \text{ in the domain of } x, y \in (-10, 10)$$

**227. Extended crosshut Function:** This function is defined as

$$f(x) = \sum_{i=1}^{n/2} \left| \sin(x_{2i-1}x_{2i}) e^{\left| \frac{100-(v_{2i-1}^{2/3}+v_{2i}^{2/3})^{0.05}}{\pi} \right|} \right| \left| \cos(x_{2i-1}x_{2i}) e^{\left| \frac{100-(v_{2i-1}^{2/3}+v_{2i}^{2/3})^{0.05}}{\pi} \right|} \right| \text{ in the domain of}$$

$x, y \in (-10, 10)$ .

**228. Extended Inverted crosshut Function:** This function is defined as

$$f(x) = -\sum_{i=1}^{n/2} \left| \sin(x_{2i-1}x_{2i}) e^{\left| \frac{100-(v_{2i-1}^{2/3}+v_{2i}^{2/3})^{0.05}}{\pi} \right|} \right| \left| \cos(x_{2i-1}x_{2i}) e^{\left| \frac{100-(v_{2i-1}^{2/3}+v_{2i}^{2/3})^{0.05}}{\pi} \right|} \right| \text{ in the domain of}$$

$x, y \in (-10, 10)$ .

**229. Extended Umbrella Function:** The function is defined as



$$f(x) = -\sum_{i=1}^{n/2} \left| \left[ x_{2i-1}^{2/3} + x_{2i}^{2/3} \right] e^{\left| \frac{100 - (x_{2i-1}^{2/3} + x_{2i}^{2/3})}{\pi} \right|^{100}} \right| \left| \left[ x_{2i-1}^{2/3} + x_{2i}^{2/3} \right] e^{\left| \frac{100 - (x_{2i-1}^{2/3} + x_{2i}^{2/3})}{\pi} \right|^{100}} \right| \quad \text{in the}$$

domain of  $x, y \in (-10, 10)$ .

**230. Extended Inverted-Umbrella Function:** The function is defined as

$$f(x) = -\sum_{i=1}^{n/2} \left| \left[ x_{2i-1}^{2/3} + x_{2i}^{2/3} \right] e^{\left| \frac{100 - (x_{2i-1}^{2/3} + x_{2i}^{2/3})}{\pi} \right|^{100}} \right| \left| \left[ x_{2i-1}^{2/3} + x_{2i}^{2/3} \right] e^{\left| \frac{100 - (x_{2i-1}^{2/3} + x_{2i}^{2/3})}{\pi} \right|^{100}} \right| \quad \text{in the}$$

domain of  $x, y \in (-10, 10)$ .

**231. Extended Flower Function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} \left| \left[ \sqrt{x_{2i-1}^2 + x_{2i}^2} \right] e^{\left| \frac{100 - (x_{2i-1}^2 + x_{2i}^2)}{\pi} \right|^{100}} \right| \left| \left[ \sqrt{x_{2i-1}^2 + x_{2i}^2} \right] e^{\left| \frac{100 - (x_{2i-1}^2 + x_{2i}^2)}{\pi} \right|^{100}} \right| \quad \text{in the domain}$$

of  $x, y \in (-10, 10)$ .

**232. Extended Royalowl function:** This function is defined as

$$f(x) = \sum_{i=1}^{n/2} \left| \sin(x_{2i-1}^2) e^{\left| \frac{100 - (x_{2i-1}^2 + x_{2i}^2)}{\pi} \right|^{100}} \right| + \left| \cos(x_{2i-1}^2) e^{\left| \frac{100 - (x_{2i-1}^2 + x_{2i}^2)}{\pi} \right|^{100}} \right| \quad x, y \in (-10, 10)$$

### 2.3.3 Extension of some benchmark test functions

**234. Extended Goldstein Price Function:** On  $x_i \in [-10, 10]; i = 1, 2$  this 2-variable function is defined as follows and has  $f_{\min}(0, -1) = 3$ .

$$f(x) = (f_1)(f_2)$$

where

$$f_1 = \sum_{i=1}^{n/2} \left[ 1 + (x_{2i-1} + x_{2i} + 1)^2 (19 - 14x_{2i-1} + 3x_{2i-1}^2 - 14x_{2i} + 6x_{2i-1}x_{2i} + 3x_{2i}^2) \right]$$

$$f_2 = \sum_{i=1}^{n/2} \left[ 30 + (2x_{2i-1} - 3x_{2i})^2 (18 - 32x_{2i-1} + 12x_{2i-1}^2 - 48x_{2i} - 36x_{2i-1}x_{2i} + 27x_{2i}^2) \right]$$

**235. Extended Hump Function:** It is a 2-variable ( $m=2$ ) function with search domain  $[-5 \leq x_i \leq 5]$ ;

( $i=1,2$ ) and dual (global) minima  $f(x^*) = -1.032$  at  $x^* = (\pm 1) (0.0898, -0.7126)$  It is given as

$$f(x) = \sum_{i=1}^{n/2} \left[ 4x_{2i-1}^2 - 2.1x_{2i-1}^4 + x_{2i-1}^6 / 3 + x_{2i-1}x_{2i} - 4x_{2i}^2 + 4x_{2i}^4 \right]$$

**236. Extended Hyperellipsoid function:** The function is defined as  $f(x) = \sum_{j=1}^n j^2 x_j^2$

with  $x_j \in [-1, 1]$  and the minimum value of the function is  $f^*(x) = 0.0$ .

**237. Extended modified Himmelblau function:** The modified Himmelblau function has only one global optimum  $f(x^*) = 0$  at  $x^* = (3, 2)$ . This (modified) function is given as

$$f(x) = \sum_{i=1}^{n/2} \left[ (x_{2i-1} + x_{2i}^2 - 7)^2 + (x_{2i-1}^2 + x_{2i} - 11)^2 + 0.1[(x_{2i-1} - 3)^2 + (x_{2i} - 2)^2] \right]$$

**238. Extended leon function:** In the search domain  $x_1, x_2 \in [-1.2, 1.2]$  this function is defined as follows and has  $f_{\min}(1, 1) = 0$ .

$$f(x) = \sum_{i=1}^{n/2} \left[ c(x_{2i} - x_{2i-1}^2) + (1 - x_{2i-1})^2 \right]; \text{ where } c=100.$$

**239. Extended Matyas function:** It is a 2-variable ( $m=2$ ) function with search domain  $[-10 \leq x_i \leq 10]$ ; ( $i=1,2$ ) and minimum  $f(x^*) = 0$  at  $x^* = (0, 0)$ . It is given as

$$f(x) = \sum_{i=2}^{n/2} \left[ c(x_{2i-1}^2 + x_{2i}^2) - dx_{2i-1}x_{2i} \right]; \text{ where } c=0.26 \text{ } d=0.48$$

**240. Extended Mc Cormick function:** In the search domain  $x_1 \in [-1.5, 4], x_2 \in [-3, 4]$  this function is defined as follows and has  $f_{\min}(-0.54719, -1.54719) = -1.9133$ .

$$f(x) = \sum_{i=1}^{n/2} \left[ \sin(x_{2i-1} + x_{2i}) + (x_{2i-1} - x_{2i})^2 - 1.5x_{2i-1} + 2.5x_{2i} + 1. \right]$$

**241. Extended Quintic function:** In the domain  $x \in [-10, 10]$  with  $f_{\min} = 0$  for  $x_i = -1$  or  $2; i = 1, 2, \dots, m$  this function (with multiple global minima) is defined as

$$f(x) = \sum_{i=1}^m |x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4|; x_i \in [-10, 10]; i = 1, 2, 3, \dots, m$$

**242. Extended Six hump Camel Function:** The camel function is defined as

$$f(x) = \sum_{i=1}^{n/2} \left[ 4x_{2i-1}^2 - 2.1x_{2i-1}^4 + \frac{x_{2i-1}^6}{3} + x_{2i-1}x_{2i} - 4x_{2i}^2 + 4x_{2i}^4 \right], -3 \leq x_1 \leq 3 \text{ and } -2 \leq x_2 \leq 2.$$

The global minimum value of the function is at  $x^* = (.0898, -0.7127)$  or  $(.0898, 0.7127)$  and  $f(x^*) = -1.0316$ .

**243. Extended three hump camel back function:** In the search domain  $x_1, x_2 \in [-5, 5]$  this function is defined as follows and has  $f_{\min}(0, 0) = 0$ .

$$f(x) = \sum_{i=1}^{n/2} \left[ 2x_{2i-1}^2 - 1.05x_{2i-1}^4 + \frac{x_{2i-1}^6}{6} + x_{2i-1}x_{2i} + x_{2i}^2 \right]$$

**244. Generalised Styblinski tang function:** In the search domain  $x_1, x_2 \in [-5, 5]$  this function is defined as follows and has  $f_{\min}(-2.903534, -2.903534) = -78.332$

$$f(x) = \frac{1}{2} \sum_{i=1}^n (x_i^4 - 16x_i^2 + 5x_i).$$

**245. Extended Zettle function:** In the search domain  $x_1, x_2 \in [-5, 5]$  this function is defined as follows and has  $f_{\min}(-0.0299, 0) = -0.003791$

$$f(x) = \sum_{i=1}^{n/2} \left[ (x_{2i-1}^2 + x_{2i}^2 - 2x_{2i-1})^2 + 0.25x_{2i-1} \right]$$

**246. Extended Treccani function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} \left[ 2x_{2i-1}^2 - 1.05x_{2i-1}^4 + \frac{x_{2i-1}^6}{6} - x_{2i-1}x_{2i} + x_{2i}^2 \right] \quad \text{where the bounds are}$$

$-3 \leq x_i \leq 3 (i=1, 2)$  The global minimum is at  $(0, 0)$  and  $(-2, 0)$  at  $f(x^*) = 0$

**247. Extended Booth Function :** A 2-variable ( $m=2$ ) function with search domain  $[-10 \leq x_i \leq 10]$ ,

$(i=1, 2)$  given as

$$f(x) = \sum_{i=1}^{n/2} \left[ (x_{2i-1} + 2x_{2i} - 7)^2 + (2x_{2i-1} + x_{2i} - 5)^2 \right]$$

**248. Extended Easom function** This function is in 2 variables ( $m=2$ ) with search domain  $[-100 \leq x_i \leq 100]$ ,  $(i=1, 2)$  and  $f(x^*) = -1$  at  $x^* = (\pi, \pi)$  It is given as

$$f(x) = - \sum_{i=1}^{n/2} \left[ \cos(x_{2i-1}) \cos(x_{2i}) \exp[-(x_{2i-1} - \pi)^2 - (x_{2i} - \pi)^2] \right]$$

## 2.4. Features of the newly developed test functions

One of the important features of the newly developed test function is shape of the test functions which is evident from the coloured dimensional graph for all the test functions from 201 to 211. For example in the four hole table function, there are four holes in the table and the global minimum value lies in lowest point of any of the four holes and same can be explained for Umbrella function. Rest of the features of the test functions are given below

**201. Tortoise function:** multimodal function with 2 dimensional test function but has been generalized up to n-dimensional in test-212

**202. I-Crosscap function:** multimodal function with 2 dimensional test function but has been generalized up to n-dimensional in test-213

**203. Crosscap function** is a negative of Crosscap multimodal function with 2 dimensional test function but has been generalized up to n-dimensional in test-214

This function is deceptive in nature. While searching the minima the algorithm can be easily trapped into the local minima.

**204. Four-hole table function** multimodal function with 2 dimensional test function but has been generalized up to n-dimensional in test-215. This function is deceptive in nature. While searching the minima the algorithm can be easily trapped into the local minima.

**205 Cross on rough ceiling function** multimodal function with 2 dimensional test function but has been generalized up to n-dimensional in test-216.

**206. Crosshut Function:** unimodal function with 2 dimensional test function but has been generalized up to n-dimensional in test-217.

**207 Icrosshut function** negative of Crosshut unimodal function with 2 dimensional test function but has been generalized up to n-dimensional in test-218.

**208 Umbrella function** Unimodal function with 2 dimensional test function but has been generalized up to n-dimensional in test-219. This function is deceptive in nature. While searching the minima the algorithm can be easily trapped into the local minima. The difficulty level of this test functions can be increased by decreasing the hole of the stand of the umbrella.

**209 I-Umbrella function :** Negative of Umbrella function, Unimodal function with 2 dimensional test function but has been generalized up to n-dimensional in test-220. This function is deceptive in nature. While searching the minima the algorithm can be easily trapped into the local minima. The difficulty level of this test functions can be increased by decreasing the hole of the stand of the umbrella.

**210 Flower Function:** Unimodal function with 2 dimensional test function but has been generalized up to n-dimensional in test-221. This function is deceptive in nature. While searching the minima the algorithm can be easily trapped into the local minima. The difficulty level of this test functions can be increased by decreasing the hole of the flower stand.

**211 Royalbowl Function:** Unimodal function with 2 dimensional test function but has been generalized up to n-dimensional in test-222.

## **2.5 Reason that newly developed test functions needed for the study**

The difficulty of a global optimization problem depends on many factors. Among the most important ones are size of the ridge and ditches, shape of the test functions, the classical example is the Rosenbrock function, where the minimum point is inside a long, narrow and parabolic-shaped flat valley, which makes convergence difficult. The dimension and high multi-modularity are the features which make the convergence again difficult.

In this thesis our main focus is on the multi-modularity and the shape of the test function. Although, we have generalized the eleven test functions in 212 to 222 and the dimension of the test functions can be considered up to any level, but we have not taken this feature into account for our study. The shape of the new test function is another aspect which can be viewed from the coloured dimensional graph and this feature also makes the problem difficult to find the global minimum.

The experiment has been conducted using the existing test functions also and the results and discussion of these studies have been incorporated in chapter-4 and all these results have been published.

We found that there are not many unimodal/multimodal test functions which are deceptive in nature and their difficulty level can be increased by adjusting their parameters, like one example could be "middle eye function". Here we have developed some difficult test functions to conduct the experiments.

## **2.6. Summary**

This is one of the important chapters of this thesis. Here a set of new test functions have been developed. In section 2.2 the purpose of developing the set of new test functions is given. In section 2.3, the functions from 201 to 211 are given. These new test functions are of dimension two. The performance of evolutionary algorithms will be studied empirically using these test functions. The graphical representation of each function from 201 to 211 has been given in section 3.1, in subsection 2.3.1, the functions 212 to 221 are the generalization of those functions with dimension  $n$ . The subsection 2.3.2 is the extended version of the functions 201 to 211. The subsection 2.3.3 contains the extension of some benchmark test functions which has been numbered from 234 to 248. In section 2.4, the main features of the test functions have

been defined and in section 2.5 the justification that why these new test function are need for the Empirical Study is explained. The visual presentation has been given with four functions of the MATLAB graphics toolbox to get the approximate vision of the optimum point through different color dimension. The first set of eleven test functions are used for empirical study.

\*\*\*\*\*

# CHAPTER 3

---

## Experimental Results of New Test Functions

### 3.1 The results of the new test functions

In this chapter, two type experiments are being conducted and results are recorded in tabular form. In the first experiment, the minimum values are obtained by all the four optimizers for eleven functions such as F201-Tortoise function, F202- Inverted Cross cap function, F203- Cross cap function, F204- Four hole table function, F205- Cross on rough ceiling function, F206- Cross-hut function, F207- Inverted Cross-hut function, F208- Umbrella function, F209- Inverted Umbrella function, F210- Flower function and F211-Royal-Baul function. Each of the functions have dimension two discussed in chapter 2. The results are recorded in Table 3.1 (a) and Table 3.1 (b). Parameters such as population size and number of iterations are fixed as 250, 50 and 100 respectively.

In the second experiment optimizers are run for different number of iterations such as 100, 200, 300, 400, 500, 600, 700 and 1000 with a population size 250 and 50. The minimum value obtained by Differential Evolution, genetic Algorithm, Particle Swarm Optimization and Simulated Annealing have been recorded in Table 3.2 (a) and (b), Table 3.3 (a) and (b), Table 3.4 (a) and (b) and Table 3.5 (a) and (b) respectively.

#### 3.1.1 Experiment setup

Parameter settings and machine configuration on which experiment have been conducted are given below. The program for Genetic Algorithm, Differential Evolution, Particle Swarm and Simulated Annealing is developed in MATLAB by Oldenhuis (2009). The experiments are conducted using MATLAB platform with the system configuration as

```
OS Name           Microsoft Windows XP Professional
Version           5.1.2600 Service Pack 2 Build 2600
System Model      Presario C700 Notebook PC
System Type       X86-based PC, 80 GB HDD
Processor         x86 Family 6 Models 15 Stepping 10 Genuine Intel ~1729
MHz
Hardware Abstraction Layer   Version = "5.1.2600.2180
(xpsp_sp2_rtm 040803-2158)"
```



```
Total Physical Memory  512 00 MB
Total Virtual Memory    2 00 GB
```

Through out the experiment we have considered the CPU speed to be fixed as the system configuration

**Differential Evolution** The `DIFFEVOLVE(func, popsize, lb,ub)` tries to find the optimum value of the objective function as `[func]` using transversal differential evolution strategy Population size set by `[popsize]` and the boundaries for each dimension is set by the vectors `[lb]` and `[ub]` respectively The `[opt, funcval, noofevals] = DIFFEVOLVE( )` returns the trail vector found to yield the global optimum in `[opt]`, and the corresponding function value by `[funcval]` The total amount of function evaluations that the algorithm performed is returned in `[noofevals]` The scaling factor is being set 0.5 and it is randomize with a **fact=0.5\*rand** The crossover probability is 0.9 The program runs for pop size 50 and 250 and the algorithm is run for 100 iterations and result is being recorded Here we have not recorded the function evaluations

**Genetic Algorithm:** The initial parameters for genetic algorithm are set as default For example, the crossover probability= 0.9 is the probability that the individual will perform a crossover The mutation probability = 0.01, is the probability that individual will mutate The Genetic optimizer `GENETIC(func, popsize, lb, ub)` tries to fund the global optimum of the fitness function `[func]` using the basic Genetic Algorithm (real coded) The crossover operator is implemented in following way

```
% generate parents indices
parentsinds = 1,
while (rem(sum(parentsinds), 2) > 0 || (sum(parentsinds) == 0))
    parentsinds = rand(popsize, 1) < crossprob,
end
parents      = pop(parentsinds, ),
parentsinds = popvec(parentsinds),
% randomize order of parents
[dummy, inds] = sort(rand(size(parents, 1), 1), 1),
parents      = parents(inds, ),
% separate sexes
numparents = size(parents, 1),
faths      = parents(1:2:numparents, ),
moths      = parents(2:2:numparents, ),
% determine crossoverpoints
numcrosses = numparents / 2,
crosspos   = round( (dims-1)*rand( numcrosses, 1 ) + 1 ),
crosspos   = numcrosses * (crosspos - 1) + (1:numcrosses)',
tempmatrix = zeros(size(moths)),
tempmatrix(crosspos) = true,
```

```

crosspos = cumsum(tempmatrix, 2),
% spawn children
daughs = ~crosspos * moths,
daughs = daughters + faths * crosspos,
sons = ~crosspos * faths,
sons = son + moths * crosspos,
children = [daughs, sons],

```

The mutation operator is implemented in following way

```

mutations = rand(popsiz, dims) * range + mins,
mutind = rand(popsiz, dims) < mutationprob,
pop(mutind) = mutations(mutind),

```

Default maximum number of iteration has been set 100 But the results have also been taken for 200,300,400,500,600 and 1000 with pop size 50 and 250

**Particle Swarm Optimization** The SWARM( func, popsize, lb,ub) tries to find the optimum value of the objective function as [func] using transversal differential evolution strategy Population size set by [popsize] and the boundaries for each dimension is set by the vectors [lb] and [ub] respectively The [opt, funcval, noofevals] = SWARM( ) returns the trail vector found to yield the global optimum in [opt], and the corresponding function value by [funcval] The total amount of function evaluations that the algorithm performed is returned in [noofevals] The parameters are being set as  $\eta_1 = 2$ , social factor when the population interact with each other  $\eta_2 = 2$ , Cooperative factor means each population passes the information to its immediate neighbor in left and right  $\eta_3 = 0.5$ , Nostalgia factor means the each population remembers its previous optimum value  $\varpi = 0.5$  Inertial factor, numneighbours=5, amount of neighbors for each particle Convalue = 150, maximum number of iterations without improvement

**Simulated Annealing** The SIMANNEAL(func, popsize,lb,ub) tries to find the optimum value of the objective function as [func] using population based simulated annealing strategy Population size set by [popsize] and the boundaries for each dimension is set by the vectors [lb] and [ub] respectively The [opt, funcval, noofevals] = SIMANNEAL( ) returns the trail vector found to yield the global optimum in [opt], and the corresponding function value by [funcval] The total amount of function evaluations that the algorithm performed is returned in [noofevals] The parameters T0 = 1 (initial temperature), minT = 1e-8 (final temperature), k =1 (Boltzmann constant)

and maximum iteration = 100 before the cooling scheduling is applied. The cooling schedule =  $(mint)^{(1/maxiter)}$

**Function setup and code example:** First the functions to be optimized are being coded in MATLAB programming code, and then it is passed through the optimizer with the external parameters in following way

```
eleon=@(x)sum((100*(x(:,2.2:end)-x(:,1:2:end-1).^2)))+(1-x(:,1:2:end-1).^2);
[x,f]=DIFFEVOLVE(eleon,250,-1.2*ones(1,10),1.2*ones(1,10))
[x,f]=SWARM(eleon,250,-1.2*ones(1,10),1.2*ones(1,10))
[x,f]=GENETIC(eleon,250,-1.2*ones(1,10),1.2*ones(1,10))
[x,f]=SIMANNEAL(eleon,250,-1.2*ones(1,10),1.2*ones(1,10)). The code
for all other functions has been given in APPENDIX B.
```

### 3.1.2 Expeimental results

**Table 3.1 (a)** Optimum Values obtained by Differential Evolution and Genetic Algorithm

Fn.	PS	Differential Evolution		Genetic Algorithm	
		Optimum point	Optimum Value	Optimum point	Optimum value
F201	250	9 4249 , -2 9999	1 91752151e+000	3 1416 , 9 4822	1 90030274e+000
F201	50	-1 5717 , -9 9134	1 93694969e+000	9 4249 , 2 9275	1 93180513e+000
F202	250	0 7097 , -0 7092	-1 86189748e+000	0 7936 , -0 6753	-1 86167298e+000
F202	50	0 7138 , 0 7152	-1 86189372e+000	-0 5890 , 0 7149	-1 86134062e+000
F203	250	0 3168 , 9 9167	9 21214865e-001	6 1622 , 7 9022	9 29639992e-001
F203	50	6 1013 , 7 7237	9 49330125e-001	5 5069 , -8 2709	9 57970538e-001
F204	250	-0 7072 , 0 7084	-2 68906905e+002	0 7491 , 0 6609	-2 68784125e+002
F204	50	-0 7128 , -0 7099	-2 68905747e+002	0 7549 , -0 6578	-2 68759895e+002
F205	250	0 3811 , -4 1214	-2 55748852e+000	1 6964 , -6 4818	-2 26232943e+000
F205	50	-5 8533 , 5 0988	-2 40525045e+000	0 3911 , 4 0137	-2 87854522e+000
F206	250	-9 9012 , -9 5192	2 45693245e+000	9 8920 , -9 8461	2 39197687e+000
F206	50	-9 8840 , 9 6959	2 56304786e+000	-9 9061 , -9 9749	2 69012588e+000
F207	250	-0 8215 , 0 8278	-2 29402230e+001	-0 8135 , -0 8149	-2 29394581e+001
F207	50	0 8166 , 0 8263	-2 29401223e+001	-0 8002 , -0 8787	-2 29375531e+001
F208	250	-0 3819 , 1 1534	<b>0.0000000e+000</b>	-0 8200 , 1 2718	<b>0.0000000e+000</b>
F208	50	0 0563 , -1 5010	<b>0.0000000e+000</b>	-0 8200 , 1 2718	<b>0.0000000e+000</b>
F209	250	-2 5270 , -0 1457	<b>-1.91155605e+000</b>	-2 0950 , 1 4744	<b>-1.91155605e+000</b>
F209	50	1 8102 , 1 7261	<b>-1.91155605e+000</b>	-0 6659 , 2 4363	<b>-1.91155605e+000</b>
F210	250	-0 6513 , 0 5427	0 0000000e+000	***	***
F210	50	0 3599 , -0 8754	0 0000000e+000	***	***
F211	250	0 6516 , -0 0008	4 98949259e-014	0 6507 , 0 0344	4 99137030e-014
F211	50	0 6513 , -0 0000	4 98949084e-014	0 6608 , 0 1126	5 01056034e-014

**Legends:** Fn Name of the function, PS Population size,

**Note :** All eleven newly developed test functions F201 to F211 are given in chapter 2 , section 2 3, pp 20

In the Table 3.1 (a) above the optimum point and optimum value of the eleven test functions have been obtained using two optimizers viz Differential Evolution and Genetic Algorithm with two set of population size 250 and 50. The values in bold face

is the best value obtained by the optimizer The \*\*\* value means the optimizer could not find the optimum value and it has resulted in overflow

From the above Table it is observed that for function F209 Differential Evolution and Genetic Algorithm have given the best value, whereas for other functions Genetic Algorithm could not perform better than the other optimizers such as particle swarm or simulated annealing

**Table 3.1 (b)** Optimum Value obtained by Particle Swarm Optimization and Simulated Annealing

Fn.	PS	Particle Swarm Optimization		Simulated Annealing	
		Optimum point	Optimum value	Optimum point	Optimum value
F201	250	-1 5707, 9 8985	1 91049833e+000	-7 8540, 6 1899	<b>1.88780730e+000</b>
F201	50	9 4248, 3 1703	<b>1.88572313e+000</b>	9 4248, -3 3427	1 88745345e+000
F202	250	-0 7085, 0 7085	-1 86189758e+000	9 9993, -9 9996	<b>-1.87444747e+000</b>
F202	50	0 7085, 0 7085	-1 86189758e+000	9 9996, -9 9992	<b>-1.87444232e+000</b>
F203	250	-7 3669, 6 6099	9 05167216e-001	9 8966, 1 5872	<b>8.61537441e-001</b>
F203	50	-6 2256, 7 8217	9 07984434e-001	9 1756, -3 7662	<b>8.95986856e-001</b>
F204	250	-0 7085, 0 7085	<b>-2.68906986e+002</b>	9 9996, 9 9991	-2 85652199e+002
F204	50	0 7085, 0 7085	<b>-2.68906986e+002</b>	-9 9997, 9 9972	-2 85532368e+002
F205	250	-2 5053, 8 1509	<b>-3.81158493e+000</b>	-5 6181, -6 9898	-3 49271313e+000
F205	50	-7 4480, 0 2109	<b>-3.36532242e+000</b>	-9 5173, -9 4076	-2 91218118e+000
F206	250	-9 9779, 9 7605	<b>1.55582079e+000</b>	9 9480, -9 4740	1 85613381e+000
F206	50	9 8851, 9 8521	<b>1.58416460e+000</b>	-9 9585, -9 9372	1 97094781e+000
F207	250	-0 8255, 0 8249	<b>-2.29402342e+001</b>	0 8238, 0 8262	-2 29402328e+001
F207	50	0 8251, -0 8251	<b>-2.29402343e+001</b>	-0 8254, 0 8266	-2 29402293e+001
F208	250	-1 0379, 0 5254	<b>0.00000000e+000</b>	-1 4606, -0 4748	<b>0.00000000e+000</b>
F208	50	-0 5708, 1 3242	<b>0.00000000e+000</b>	-1 4546, -0 3751	<b>0.00000000e+000</b>
F209	250	1 5023, 2 3486	<b>-1.91155605e+000</b>	2 6232, 0 1239	<b>-1.91155605e+000</b>
F209	50	-0 5622, 2 5566	<b>-1.91155605e+000</b>	2 5978, 0 6155	<b>-1.91155605e+000</b>
F210	250	0 2023, 0 7588	<b>0.00000000e+000</b>	0 6036, -0 0628	<b>0.00000000e+000</b>
F210	50	-0 4052, 0 5276	<b>0.00000000e+000</b>	-0 0593, 0 4253	<b>0.00000000e+000</b>
F211	250	0 6513, 0 0000	4 98949084e-014	-9 9971, -9 9988	<b>4.67427986e-014</b>
F211	50	-0 6513, 0 0000	4 98949084e-014	0 6513, -0 0032	4 98950694e-014

**Legends:** Fn Name of the function, PS Population Size

**Note :** All eleven newly developed test functions F201 to F211 are given in chapter 2 , section 2 3, pp 20

In the Table 3 1 (b) above the optimum point and optimum value of the eleven test functions have been obtained using two optimizers viz Particle Swarm Optimization and Simulated Annealing with two set of population size 250 and 50 The values in bold face is the best value obtained by the optimizer with that population size It is observed from the above table that the Particle Swarm and simulated Annealing have performed better on F201 then Differential Evolution and Genetic Algorithm For function F202, F203, F209 and F211 Simulated annealing has outperformed all other

optimizers For functions F204, F205, F206, F207, F208, F209 and F210 Particle swarm have outperformed all other algorithms For function F209 all the four optimizers, considered in our investigation have given the same value with both the set of population size

### 3.2 Results recorded from the Differential Evolution at different iterations

The experimental setup is kept like above and number of iterations is being changed to 200,300,400,500,600,700 and 1000 for all the four optimizers The result obtained from Differential Evolution have been recorded in Table 3 2(a) &Table 3 2(b)

**Table 3.2 (a)** Optimum values of the functions recorded by differential evolution for 100,200, 300, 400 iterations

Fn.	PS	Differential Evolution			
		100	200	300	400
F201	250	<b>1.91752151e+000</b>	2 60607012e+000	1 92969629e+000	1 92128170e+000
F201	50	1 93694969e+000	2 45892574e+000	1 95692161e+000	1 92789217e+000
F202	250	-1 86189748e+000	-1 86189751e+000	-1 86189569e+000	-1 86189428e+000
F202	50	-1 86189372e+000	-1 86189523e+000	-1 86189236e+000	-1 86189440e+000
F203	250	9 21214865e-001	9 21214865e-001	9 23446408e-001	<b>8.88070077e-001</b>
F203	50	9 49330125e-001	9 49330125e-001	9 54186491e-001	9 42273525e-001
F204	250	-2 68906905e+002	-2 68906909e+002	-2 68906845e+002	-2 68906879e+002
F204	50	-2 68905747e+002	-2 68906677e+002	-2 68905523e+002	-2 68904703e+002
F205	250	-2 55748852e+000	-2 42648317e+000	-2 87730783e+000	-2 84647550e+000
F205	50	-2 40525045e+000	-1 97432970e+000	-2 09707192e+000	-2 54237071e+000
F206	250	<b>2.45693245e+000</b>	4 62045594e+000	4 36443314e+000	4 53838033e+000
F206	50	2 56304786e+000	5 14521467e+000	5 20364223e+000	5 19345740e+000
F207	250	-2 29402230e+001	-2 29401978e+001	-2 29401329e+001	-2 29401189e+001
F207	50	-2 29401223e+001	-2 29400814e+001	-2 29402289e+001	-2 29400960e+001
F208	250	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F208	50	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F209	250	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>
F209	50	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>
F210	250	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F210	50	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F211	250	4 98949259e-014	4 98949481e-014	4 98952356e-014	4 98949393e-014
F211	50	<b>4.98949084e-014</b>	<b>4.98949084e-014</b>	4 98965805e-014	<b>4.98949084e-014</b>

**Legends:** Fn Name of the function, PS Population Size

**Note :** All eleven newly developed test functions F201 to F211 are given in chapter 2 , section 2 3, pp 20

The optimum value of the eleven test functions have been obtained by Differential Evolution with two set of population size 250 and 50 at the iterations 100, 200, 300 and 400 and recorded in Table 3 2 (a) The values in bold face is the best value obtained by the optimizer with that population size From the table it is observed that Differential Evolution has given the best value for the function F201 at 100 iterations,

F203 at 400 iterations, F204 at 200 iterations, F205 at 300 iterations, and F206 at 100 iterations. For functions F208, F209 and F210 the optimum value does not change after increasing the no of iterations respectively.

**Table 3.2 (b)** Optimum values of the functions recorded by differential evolution for 500,600, 700, 1000 iterations

Fn.	PS	Differential Evolution			
		500	600	700	1000
F201	250	1 93186832e+000	1 92956354e+000	1 93551409e+000	1 92480419e+000
F201	50	1 92848808e+000	1 92875804e+000	1 91824795e+000	1 93285085e+000
F202	250	-1 86189744e+000	-1 86189744e+000	-1 86189753e+000	-1 86189585e+000
F202	50	-1 86189561e+000	<b>-1.86189651e+000</b>	-1 86189428e+000	-1 86189534e+000
F203	250	9 26358931e-001	9 22588980e-001	9 22455915e-001	9 36622066e-001
F203	50	9 37265807e-001	9 40419929e-001	9 46030396e-001	9 46160103e-001
F204	250	-2 68905114e+002	-2 68906849e+002	-2 68906415e+002	-2 68906976e+002
F204	50	-2 68906777e+002	<b>-2.68903963e+002</b>	-2 68905313e+002	-2 68906717e+002
F205	250	-2 05435159e+000	-2 82959056e+000	-2 45212838e+000	-2 15456991e+000
F205	50	<b>-2.73461217e+000</b>	-2 30531027e+000	-2 65383976e+000	-2 35713981e+000
F206	250	4 42264619e+000	4 68592524e+000	5 00527646e+000	3 98279703e+000
F206	50	4 63434923e+000	5 66109600e+000	4 63151441e+000	5 75338253e+000
F207	250	-2 29401595e+001	<b>-2.29402304e+001</b>	-2 29401384e+001	-2 29402261e+001
F207	50	-2 29402343e+001	-2 29399390e+001	-2 29401323e+001	-2 29401597e+001
F208	250	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F208	50	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F209	250	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>
F209	50	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>
F210	250	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F210	50	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F211	250	4 98950768e-014	4 98950539e-014	4 98953163e-014	<b>4.98977918e-014</b>
F211	50	4 98955474e-014	4 98954565e-014	4 98980698e-014	4 98952548e-014

**Legends:** Fn Name of the function, PS Population Size

**Note :** All eleven newly developed test functions F201 to F211 are given in chapter 2 , section 2.3, pp 20

In the table 3.2 (b) above the optimum value of the eleven test functions have been obtained by Differential Evolution with two set of population size 250 and 50 at the iterations 500, 600, 700 and 1000. The values in bold face is the best value obtained by the optimizer with the population size 250 and 50. It is observed in the table that Differential Evolution has given the optimum value for F201 with population size 50 and F202 with pop size 250 at 700 iterations. Also for F202, F203 and F204 the optimizer has given the best value at 700, 500, 600 and 700 respectively. For the functions F208, F209, F210 and F211 the value remains unchanged, after increasing the number of iterations.

### 3.3 Results recorded from the Genetic Algorithm at different iterations

**Table 3.3 (a)** Optimum values of the functions recorded by Genetic Algorithms at 100,200, 300, 400 iterations

Fn.	PS	Genetic Algorithm			
		100	200	300	400
F201	250	2 04949000e+000	1 92416337e+000	1 93374375e+000	1 91354830e+000
F201	50	2 62253128e+000	1 94895262e+000	1 95608050e+000	1 94945681e+000
F202	250	-1 86167298e+000	-1 86189184e+000	-1 86189122e+000	-1 86185251e+000
F202	50	-1 86134062e+000	-1 86178770e+000	-1 86164848e+000	<b>-1.86183452e+000</b>
F203	250	9 29639992e-001	9 18600422e-001	9 33159688e-001	9 25872868e-001
F203	50	9 57970538e-001	9 59775667e-001	9 53196131e-001	9 45252476e-001
F204	250	-2 68784125e+002	-2 76758157e+002	-2 68798408e+002	-2 68906121e+002
F204	50	-2 68759895e+002	-2 68664793e+002	-2 68720667e+002	-2 68341625e+002
F205	250	-2 26232943e+000	-3 26335208e+000	-2 82682342e+000	-2 23202944e+000
F205	50	-1 87854522e+000	-2 29652881e+000	-2 08575158e+000	<b>-2.45752169e+000</b>
F206	250	5 26469049e+000	4 70047361e+000	4 15435657e+000	4 23678523e+000
F206	50	5 20870500e+000	<b>4.81554194e+000</b>	5 64967604e+000	5 95678427e+000
F207	250	-2 29394581e+001	-2 29398934e+001	-2 29385071e+001	-2 29401261e+001
F207	50	-2 29375531e+001	-2 29064018e+001	-2 29321000e+001	-2 29130121e+001
F208	250	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F208	50	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F209	250	-1 91155605e+000	1 91155605e+000	Result fluctuate	Result fluctuate
F209	50	-1 91155605e+000	-1 91155605e+000	Result fluctuate	Result fluctuate
F210	250	***	overflow	overflow	overflow
F210	50	***	overflow	overflow	overflow
F211	250	4 99137030e-014	5 00454156e-014	5 00084036e-014	4 99027284e-014
F211	50	5 01056034e-014	4 99699160e-014	5 00102078e-014	<b>4.90737275e-014</b>

**Legends:** Fn Name of the function, PS Population Size

**Note :** All eleven newly developed test functions F201 to F211 are given in chapter 2 , section 2 3, pp 20

In Table 3 3 (a) the optimum value of the eleven test functions have been obtained by Genetic Algorithm with two set of population size 250 and 50 at the iterations 100, 200, 300 and 400 The values in bold face are the best value obtained by the optimizer with that population size The \*\*\* value denote the overflow, means the optimizers did not converge From the above table it is observed that Genetic Algorithm finds the best value for functions F202, F204, F205 at the iterations 400, 200 and 400 respectively For the function F208 the value remains unchanged after increasing the number of iterations For F209, F210 the optimizer, gives the fluctuating value or it overflows For F211, the best value is obtained at the iterations 300 and 400

**Table 3.3 (b)** Optimum values of the functions recorded by Genetic Algorithms for 500,600, 700, 1000 iterations

Fn.	PS	Genetic Algorithm			
		500	600	700	1000
F201	250	1 94188299e+000	<b>1.90541800e+000</b>	1 95307818e+000	1 91901756e+000
F201	50	1 93386611e+000	1 94029430e+000	1 95091865e+000	1 93946091e+000
F202	250	-1 86171445e+000	-1 86187271e+000	-1 86469311e+000	-1 87077152e+000
F202	50	-1 86156356e+000	-1 86142687e+000	-1 86067167e+000	-1 86171268e+000
F203	250	<b>9.07322728e-001</b>	9 35419787e-001	9 32422247e-001	9 39023920e-001
F203	50	9 42806955e-001	9 48458557e-001	9 64817071e-001	9 39136908e-001
F204	250	-2 68730251e+002	-2 68730079e+002	-2 68890284e+002	-2 68871366e+002
F204	50	<b>-2.68892538e+002</b>	-2 68800296e+002	-2 68847171e+002	-2 68798891e+002
F205	250	-2 27214723e+000	-2 30373267e+000	-2 49500243e+000	-2 51221115e+000
F205	50	-2 36650898e+000	-2 27402162e+000	-2 17168158e+000	-2 22822329e+000
F206	250	<b>3.78401822e+000</b>	5 09392847e+000	4 29985636e+000	5 24951893e+000
F206	50	5 48705784e+000	5 44453899e+000	4 95147933e+000	5 90572643e+000
F207	250	-2 29387837e+001	-2 29391994e+001	-2 29401917e+001	-2 29389772e+001
F207	50	-2 29326382e+001	<b>-2.29400122e+001</b>	-2 29314146e+001	-2 29386139e+001
F208	250	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F208	50	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F209	250	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F209	50	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F210	250	overflow	overflow	overflow	overflow
F210	50	overflow	overflow	overflow	overflow
F211	250	4 99413089e-014	4 99940452e-014	4 99363483e-014	4 98411078e-014
F211	50	5 00392192e-014	5 02104377e-014	5 11792540e-014	5 00633123e-014

**Legends:** Fn Name of the function, PS Population Size

**Note :** All eleven newly developed test functions F201 to F211 are given in chapter 2 , section 2 3, pp 20

In the Table 3.3 (b), The optimum value of the eleven test functions have been obtained by Genetic Algorithm with two set of population size 250 and 50 at the iterations 500, 600, 700 and 1000 The values in bold face is the best value obtained by the optimizer with that population size In the above table it is observed that when the number of iterations are increased the Genetic Algorithm converges and gives the optimum values. Except for the function F210 where it could not converge, for all other functions F201, F203 and F204 the best value is obtained at the iterations 600 and 500 respectively For F208 and F209 the optimum value does not change

### 3.4 Result recorded from the Particle Swarm optimization at different iterations

**Table 3.4 (a)** Optimum values of the functions recorded by Particle Swarm Optimization at 100,200, 300, 400 iterations

Fn.	PS	Particle Swarm Optimization			
		100	200	300	400
F201	250	1 10899463e+000	1 86176451e+000	1 91195594e+000	1 88463958e+000
F201	50	<b>9.42486768e-001</b>	1 86131442e+000	1 85093636e+000	1 87736573e+000



F202	250	-1.86189758e+000	-1.86189758e+000	-1.86189758e+000	-1.86189758e+000
F202	50	-1.86189758e+000	-1.86189758e+000	-1.86189758e+000	-1.86189758e+000
F203	250	9 05167216e-001	9 19289128e-001	9 01959933e-001	<b>8.62815019e-001</b>
F203	50	9 07984434e-001	8 80223377e-001	8 95459497e-001	8 66018282e-001
F204	250	<b>-2.68906986e+002</b>	<b>-2.68906986e+002</b>	<b>-2.68906986e+002</b>	<b>-2.68906986e+002</b>
F204	50	<b>-2.68906986e+002</b>	<b>-2.68906986e+002</b>	<b>-2.68906986e+002</b>	<b>-2.68906986e+002</b>
F205	250	-3 81158493e+000	-3 12784593e+000	-3 51546244e+000	-3 55868875e+000
F205	50	-3 36532242e+000	-3 26404079e+000	-3 30218920e+000	-3 51137074e+000
F206	250	1 55582079e+000	2 20140193e+000	2 32486622e+000	2 10450677e+000
F206	50	<b>1.58416460e+000</b>	2 23086987e+000	1 98970786e+000	1 54813575e+000
F207	250	-2 29402342e+001	-2 29402343e+001	<b>-2.29402343e+001</b>	-2 29402340e+001
F207	50	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>
F208	250	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F208	50	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F209	250	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>
F209	50	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>
F210	250	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F210	50	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>	<b>0.00000000e+000</b>
F211	250	<b>4.98949084e-014</b>	<b>4.98949084e-014</b>	<b>4.98949084e-014</b>	<b>4.98949084e-014</b>
F211	50	<b>4.98949084e-014</b>	<b>4.98949084e-014</b>	<b>4.98949084e-014</b>	<b>4.98949084e-014</b>

**Legends:** Fn Name of the function, PS Population Size

**Note :** All eleven newly developed test functions F201 to F211 are given in chapter 2 , section 2 3, pp 20

In the Table 3 4 (a) the optimum value of the eleven test functions have been obtained by Particle Swarm Optimization with two set of population size 250 and 50 at the iterations 100, 200, 300 and 400 The values in bold face is the best value obtained by the optimizer with that population size It is also observed from the above table that Particle Swarm Optimization has performed fairly better than other optimizers. For functions F201, F204 and F206 best value have been found at 100 iterations For functions F202, F207, F208, F209, F210 and F211 the value remains unchanged and are the optimum value after increasing the number of iterations

**Table 3.4 (b)** Optimum values of the functions recoeded by Particle Sawrm Optimization at 500,600, 700, 1000 iterations

Fn.	PS	Particle Swarm Optimization			
		500	600	700	1000
F201	250	1 89499129e+000	1 87370701e+000	1 89147147e+000	1 89132774e+000
F201	50	1 90399212e+000	1 86655067e+000	1 78736187e+000	1 89753495e+000
F202	250	<b>-1.86189758e+000</b>	<b>-1.86189758e+000</b>	<b>-1.86189758e+000</b>	<b>-1.86189758e+000</b>
F202	50	<b>-1.86189758e+000</b>	<b>-1.86189757e+000</b>	<b>-1.86189758e+000</b>	<b>-1.86189758e+000</b>
F203	250	9 00336166e-001	8 87600013e-001	9 08343030e-001	8 92108439e-001
F203	50	8 99100153e-001	8 95359945e-001	9 05744351e-001	9 02746687e-001
F204	250	<b>-2.68906986e+002</b>	<b>-2.68906986e+002</b>	<b>-2.68906986e+002</b>	<b>-2.68906986e+002</b>
F204	50	<b>-2.68906986e+002</b>	<b>-2.68906986e+002</b>	<b>-2.68906986e+002</b>	<b>-2.68906986e+002</b>
F205	250	-3 80901825e+000	-3 07819011e+000	<b>-4.18844511e+000</b>	-3 24493892e+000
F205	50	-3 37678832e+000	-3 08405265e+000	-3 31437855e+000	-4 00760484e+000
F206	250	2 29605752e+000	2 26159865e+000	1 99167512e+000	1 96860973e+000
F206	50	2 18200740e+000	1 98092307e+000	2 32886388e+000	2 36256907e+000
F207	250	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>
F207	50	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>

F208	250	0.00000000e+000	0.00000000e+000	0.00000000e+000	0.00000000e+000
F208	50	0.00000000e+000	0.00000000e+000	0.00000000e+000	0.00000000e+000
F209	250	-1.91155605e+000	-1.91155605e+000	-1.91155605e+000	-1.91155605e+000
F209	50	-1.91155605e+000	-1.91155605e+000	-1.91155605e+000	-1.91155605e+000
F210	250	0.00000000e+000	0.00000000e+000	0.00000000e+000	0.00000000e+000
F210	50	0.00000000e+000	0.00000000e+000	0.00000000e+000	0.00000000e+000
F211	250	4.98949084e-014	4.98949084e-014	4.98949084e-014	4.98949084e-014
F211	50	4.98949084e-014	4.98949084e-014	4.98949084e-014	4.98949084e-014

**Legends:** Fn Name of the function, PS Population Size

**Note :** All eleven newly developed test functions F201 to F211 are given in chapter 2 , section 2 3, pp 20

In the Table 3 4 (b), The optimum value of the eleven test functions have been obtained by Particle Swarm Optimization with two set of population size 250 and 50 at the iterations 500, 600, 700 and 1000 The values in bold face is the best value obtained by the optimizer with population size 250 and 50 Here we observe that the value of the functions F202 and F207 to F211 does not change after increasing the number of iterations For the function F204 and F205 the optimum value becomes better after increasing the no of iterations

### 3.5 Results recorded from the Simulated Annealing at different iterations

**Table 3.5 (a)** Optimum values of the functions recoeded by Simulated Annealing for 100,200, 300, 400 iterations

Fn.	PS	Simulated Annealing			
		100	200	300	400
F201	250	1 81917980e+000	1 61464035e+000	1 46444345e+000	1 58020641e+000
F201	50	1 92632236e+000	1 73063007e+000	1 81089478e+000	1 78608337e+000
F202	250	-1 87444747e+000	-1 87447416e+000	-1 87448306e+000	-1 87448261e+000
F202	50	-1 87444232e+000	-1 87448303e+000	1 87447822e+000	-1 87448267e+000
F203	250	8 61537441e-001	8 65958803e-001	8 63451906e-001	8 50372161e-001
F203	50	8 95986856e-001	8 94752643e-001	8 89621549e-001	8 74339236e-001
F204	250	-2 85652199e+002	<b>-4.37041243e+000</b>	-2 85692285e+002	-2 85704455e+002
F204	50	-2 85532368e+002	-3 65608480e+000	-2 85714326e+002	-2 85715673e+002
F205	250	-3 49271313e+000	-5 08550404e+000	-4 72397803e+000	-5 21478553e+000
F205	50	-2 91218118e+000	-3 53617037e+000	-3 92858947e+000	-3 53587268e+000
F206	250	1 85613381e+000	2 82549476e+000	2 22263080e+000	1 95934483e+000
F206	50	1 97094781e+000	2 00803188e+000	2 58232390e+000	2 52601872e+000
F207	250	-2 29402328e+001	-1 91155605e+000	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>
F207	50	-2 29402293e+001	-1 91155605e+000	-2 29402340e+001	<b>-2.29402343e+001</b>
F208	250	0.00000000e+000	0.00000000e+000	0.00000000e+000	0.00000000e+000
F208	50	0.00000000e+000	0.00000000e+000	0.00000000e+000	0.00000000e+000
F209	250	-1.91155605e+000	-1.91155605e+000	-1.91155605e+000	-1.91155605e+000
F209	50	-1.91155605e+000	-1.91155605e+000	-1.91155605e+000	-1.91155605e+000
F210	250	0.00000000e+000	0.00000000e+000	0.00000000e+000	0.00000000e+000
F210	50	0.00000000e+000	0.00000000e+000	0.00000000e+000	0.00000000e+000
F211	250	4 67427986e-014	4 56195939e-014	4 55780188e-014	4 54638531e-014
F211	50	4 98950694e-014	4 55131865e-014	4 54490951e-014	4 55797674e-014

**Legends:** Fn Name of the function, PS Population Size

**Note :** All eleven newly developed test functions F201 to F211 are given in chapter 2 , section 2 3, pp 20

In the Table 3 5 (a), The optimum value of the eleven test functions have been obtained by Simulated Annealing with two set of population size 250 and 50 at the iterations 100, 200, 300 and 400 The values in bold face are the best value obtained by the optimizer with that population size From the above table it is observed that Simulated Annealing has the better value when the number of iterations is increased For the functions F201, F204, F206 and F207 at 300, 200, 100, and 400 respectively For functions F208, F209, F210 the value does not change after increasing the number of iterations

**Table 3.5 (b)** Optimum values of the functions recooded by Simulated Annealing for 500,600, 700, 1000 iterations

Fn.	PS	Simulated Annealing			
		500	600	700	1000
F201	250	1 56508886e+000	1 57200882e+000	1 47555787e+000	1 58126903e+000
F201	50	1 49024764e+000	1 75287432e+000	<b>1.22544459e+000</b>	1 65575530e+000
F202	250	-1 87448327e+000	-1 87448213e+000	<b>-1.87448376e+000</b>	-1 87448315e+000
F202	50	-1 87448358e+000	-1 87448312e+000	-1 87448060e+000	-1 87448302e+000
F203	250	8 65044900e-001	8 44390852e-001	8 42556562e-001	8 57859313e-001
F203	50	8 68019910e-001	<b>8.43758534e-001</b>	8 73755869e-001	8 56095865e-001
F204	250	-2 85708492e+002	-2 85712654e+002	-2 85716289e+002	-2 85715991e+002
F204	50	-2 85715829e+002	-2 85714278e+002	-2 85712261e+002	-2 85716298e+002
F205	250	-4 12041154e+000	-4 46160511e+000	<b>-6.99341415e+000</b>	-5 43144237e+000
F205	50	-5 47514353e+000	-4 23264582e+000	-4 01198447e+000	-4 78963915e+000
F206	250	2 16131524e+000	2 07405467e+000	1 91837266e+000	<b>1.45370698e+000</b>
F206	50	2 33218290e+000	2 18072424e+000	2 24050880e+000	2 24941174e+000
F207	250	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>
F207	50	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>	<b>-2.29402343e+001</b>
F208	250	0.00000000e+000	0.00000000e+000	0.00000000e+000	0.00000000e+000
F208	50	0.00000000e+000	0.00000000e+000	0.00000000e+000	0.00000000e+000
F209	250	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>
F209	50	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>	<b>-1.91155605e+000</b>
F210	250	0.00000000e+000	0.00000000e+000	0.00000000e+000	0.00000000e+000
F210	50	0.00000000e+000	0.00000000e+000	0.00000000e+000	0.00000000e+000
F211	250	4 54609034e-014	4 54615076e-014	4 54472559e-014	<b>4.54419629e-014</b>
F211	50	4 54520209e-014	4 54621191e-014	4 54421855e-014	4 54594475e-014

**Legends:** Fn Name of the function, PS Population Size

**Note :** All eleven newly developed test functions F201 to F211 are given in chapter 2 , section 2 3, pp 20

In the Table 3 5 (b), The optimum value of the eleven test functions have been obtained by the optimizer viz Simulated Annealing with two set of population size 250 and 50 at the iterations 500, 600, 700 and 1000 The values in bold face are the best value obtained by the optimizer with that population size In the above table we also observed that Simulated Annealing has given the best value for F201 at 700 iteration, F202 at

700 iterations, F203 at 600 iterations, F205 at 500 iterations, and F206 at 1000 iterations. The values remain unchanged for F207, F208, F209, F210. The minimum value is found at 1000 iterations for the function F211.

### 3.6 Analysis of results recorded in the above tables

In section 3.1.2, two experiments are conducted for empirical study of Evolutionary Algorithms. In the first experiment, the optimizers were set to run for 100 iterations with two sets of population size 250 and 50 respectively for all the eleven test functions (discussed in chapter 2) so that the results could record large population size and small population size. The optimum point and optimum values obtained by Differential Evolution and Genetic Algorithms are recorded in **Table 3.1 (a)** and Particle Swarm Optimization and Simulated Annealing are recorded in **Table 3.1 (b)** respectively. From the recorded result it is found that Simulated Annealing has outperformed for F201 with the population size 250, whereas Particle Swarm Optimization has outperformed all other optimizers for population size 50. Simulated Annealing has outperformed all the optimizers with both sets of population size for F202 and F203. Particle Swarm Optimization outperformed all other optimizers with both sets of population size 250 and 50 for function F204, F205, F206 and F207. The optimum value found for functions F208 and F209 by all optimizers considered for study finds the value 0 and  $-1.91155605 \times 10^0$  respectively and it remains the same with the change of population size. For F210, Differential Evolution, Particle Swarm Optimization and Simulated Annealing have given the optimum value zero, whereas Genetic Algorithm could not find the optimum value and an overflow caption was shown by the program. Simulated Annealing outperforms the other algorithms for function F211. Hence we conclude that no optimizer outperforms others on all types of optimization problems. In fact, any evolutionary algorithm is only suitable for a class of test problems with a specific feature. This validates the “No Free Lunch Theorem” of Wolpert and Macready (1997) and Ho and Pevy (2002).

#### (a) Analysis based on the results recorded from Differential Evolution:

In section 3.2 from the Table 3.2 (a) and 3.2 (b) it is observed that Differential Evolution found the optimum value as  $1.91752151 \times 10^0$  for F201 with population size 250 after 100 iterations,  $-1.86189651 \times 10^0$  for F202 with the population size 50

after 700 iterations and  $8.88070077e-001$  for F203 with the population size 250 after 400 iterations. Similarly, the optimum value found for F204 is  $-2.68903963e+002$  after 700 iterations with the population size 50,  $-2.73461217e+000$  for F205 after 500 iterations with the population size 50,  $2.45693245e+000$  for F206 after 100 iterations with the population size 250,  $-2.29402304e+001$  for F207 after 600 iterations with the population size 250 respectively. Again, Differential Evolution have given the optimum value as  $0.0000000e+000$ ,  $-1.91155605e+000$  and  $0.00000000e+000$  respectively for F208, F209 and F210 and the result do not change if the population size or the number of iterations are increased. In case of F211 the optimum value obtained is  $4.98977918e-014$  after 1000 iterations with the population size 250 and produces same result after 100, 200 and 400 iterations with the population size 50. Hence with this analysis it can be concluded that the performance of the Differential Evolution does not depend on either on number of iterations or population size. This validates the “No Free Lunch Theorem” of Wolpert and Macready (1997) and Ho and Pepyne (2002).

**(b) In Analysis based on the results recorded from Genetic Algorithm:**

In section 3.3 from the Table 3.3 (a) and 3.3 (b) it is observed that Genetic Algorithm found the optimum value as  $1.90541800e+000$  for F201 with population size 250 after 600 iterations,  $-1.86183452e+000$  for F202 with the population size 50 after 400 iterations and  $9.07322728e-001$  for F203 with the population size 250 after 500 iterations. Similarly, the optimum value found for F204 is  $-2.68892538e+002$  after 500 iterations with the population size 50,  $-2.45752169e+000$  for F205 after 400 iterations with the population size 250,  $3.78401822e+000$  for F206 after 500 iterations with the population size 250,  $-2.29400122e+001$  for F207 after 600 iterations with the population size 50 respectively. Again, Genetic Algorithm have given the optimum value as  $0.0000000e+000$  and  $0.00000000e+000$  respectively for F208 and F209 respectively. The Optimizer did not converge for F210 at any number of iterations. In case of F211 the optimum value obtained is  $4.90737275e-014$  after 400 iterations with the population size 50. Hence with this analysis it can be concluded that the performance of the Genetic Algorithm does not depend on either on number of iterations or population size but whenever it has converge it has given the better result than Differential Evolution. This validates the “No Free Lunch Theorem” of Wolpert and Macready (1997) and Ho and Pepyne (2002).

**(c) In Analysis based on the results recorded from Particle Swarm Optimization:**

Based on the results recorded in Table 3.4 (a) and 3.4 (b), it is observed that Particle Swarm Optimization found the optimum value as  $9.42486768 \times 10^{-1}$  for F201 with population size 50 after 100 iterations,  $-1.86189758 \times 10^0$  for F202 with the population size 250 after 100 iterations and value remains same even if the number of iterations increased and  $8.62815019 \times 10^{-1}$  for F203 with the population size 250 after 400 iterations. Similarly, the optimum value found for F204 is  $-2.68906986 \times 10^2$  and the value does not change even if we increase the number of iterations,  $-4.18844511 \times 10^0$  for F205 after 700 iterations with the population size 250,  $1.58416460 \times 10^0$  for F206 after 100 iterations with the population size 50,  $-2.29402343 \times 10^1$  for F207 after 100 iterations with the population size 250 respectively. Again, Particle Swarm Optimization have given the optimum value as  $0.0000000 \times 10^0$ ,  $-1.91155605 \times 10^0$ ,  $0.0000000 \times 10^0$  and  $4.98949084 \times 10^{-14}$  respectively for F208, F209, F210 and F11 and the result do not change if the population size or the number of iterations are increased. Hence with this analysis it can be concluded that the performance of the Particle Swarm Optimization does not depend on either on number of iterations or population size. It is also observed that the Particle Swarm has a better converges ability but it is slow. This validates the "No Free Lunch Theorem" of Wolpert and Macready (1997) and Ho and Pepyne (2002).

**(d) In Analysis based on the results recorded from Simulated Annealing:**

Based on the results recorded in Table 3.5 (a) and 3.5 (b), it is observed that Simulated Annealing found the optimum value as  $1.2254459 \times 10^0$  for F201 with population size 50 after 700 iterations,  $-1.87448376 \times 10^0$  for F202 with the population size 250 after 700 iterations and value remains same even if the number of iterations increased and  $8.43758534 \times 10^{-1}$  for F203 with the population size 50 after 600 iterations. Similarly, the optimum value found for F204 is  $-4.37041243 \times 10^0$  after 200 iterations with 250 population size,  $-6.99341415 \times 10^0$  for F205 after 700 iterations with the population size 250,  $1.45370698 \times 10^0$  for F206 after 100 iterations with the population size 50 respectively. The optimum value obtained for F207 is  $-2.29402343 \times 10^1$  after 300 iterations with the population size 250 and the value remains unchanged even if the number iterations are increased. Again, Simulated Annealing have given the optimum value as  $0.0000000 \times 10^0$ ,  $-1.91155605 \times 10^0$  and

0.00000000e+000 respectively for F208, F209 and F210 and the result do not change if the population size or the number of iterations are increased. In case of F211 the optimum value obtained is  $4.54419629e-014$  after 1000 iterations with the population size 50. Hence with this analysis it can be concluded that the performance of the Simulated Annealing does not depend on either on number of iterations or population size. This validates the “No Free Lunch Theorem” of Wolpert and Macready (1997) and Ho and Pepyne (2002).

### 3.7 Summary

In this chapter the optimum values of the eleven test functions developed in (chapter 2, section 2.1.1, 201-211) have been recorded by the optimizers such as Differential Evolution, Genetic algorithms, Particle Swarm Optimization and Simulated Annealing. In Table 3.1 (a) and Table 3.1 (b) the optimum value and optimum point obtained by Differential Evolution, Genetic Algorithms and Particle Swarm Optimization and Simulated Annealing have been recorded with two set of population size 250 and 50. The algorithm were run for 100 iterations. In Table 3.2 (a) and Table 3.2 (b) the optimum value obtained by Differential Evolution for 100,200,300, 400 and 500, 600, 700 and 1000 iterations respectively have been recorded respectively. In Table 3.3 (a) and Table 3.3 (b) the optimum value obtained by Genetic Algorithms for 100,200,300,400 and 500, 600, 700 and 1000 iterations respectively have been recorded. In Table 3.4 (a) and Table 3.4 (b) the optimum value obtained by Particle Swarm Optimization for 100, 200, 300, 400 and 500, 600, 700 and 1000 iterations have been recorded respectively. In Table 3.5 (a) and Table 3.5 (b) the optimum value obtained by Simulated Annealing for 100, 200, 300, 400 and 500, 600, 700 and 1000 iterations have been recorded respectively and all the results are analysed.

The results obtained in Table 3.1 (a) & (b), 3.2 (a) & (b), 3.3 (a) & (b), 3.4 (a) & (b) and 3.5 (a) & (b) are re-experimented on the eleven test functions on vector –Genetic Algorithms and some advanced evolutionary algorithms. It is found that the results have been almost same except the 201, 205 and 206 in which with the readjustment of crossover and mutation operator it has given better result. These algorithms are the state-of-the-art algorithm.

\*\*\*\*\*

## CHAPTER 4

---

### **Comparative Study of Evolutionary Algorithms with Benchmark Test Functions**

#### **4.1 Comparative study of modified Differential Evolution and Genetic Algorithm**

Several variations of meta-heuristics have been developed recently and each of them claims to outperform others. Also these algorithms lack convergence criteria or they have no convergence proof but claim to outperform others, there is a need of rigorous comparative study of all the algorithms together or pair wise with a sufficiently large number of test functions. Brest et al (2006) did the comparative study of control parameter in differential evolution on numerical benchmark problems. In this study we have done comparative study of modified differential evolution and genetic algorithm. Each of these methods has its origin in Von Neumann's Monte Carlo experiments. These methods have been tested with certain benchmark test problems and some new test functions introduced. Researchers who worked on the comparative study of evolutionary algorithms are Colville (1986), Vesterstorm and Thomsen (2004), Voigt (1992), Whitley et al (1996) and Yao et al (1999).

##### **4.1.1 Methods used**

**Genetic Algorithm:** The algorithm is explained in chapter 1, section 1.4.1 (a) page 9

**Modified Differential Evolution** The algorithm is explained in chapter 1, section 1.4.1 (d) page 11-12



## 4.1.2 Benchmark test functions used for the experiments

The following ten benchmark test functions i.e. Egg holder function, Corana function, Freudenstein Roth function, Leon function, Perm function #1, Trid function, Hougen Function, Weierstrass function, Needle-eye function, Shekel Function are used for our investigation.

## 4.1.3 Experiment on Genetic Algorithms and Modified Differential Evolution.

**Genetic algorithms:** We have used an input file to pass the different parameters i.e. npopsiz=5, pcross=.9d0, npsibl= (2<sup>n</sup> n= powers of 2) pmutate=0.02d0 and maxgen=200. Another params.f was included in the main program having three parameters population size=200, nchrommax=60 and nparamax=10. Other two parameters are adjustable according to the dimensions of the function used.

**Modified Differential Evolution:** In Differential Evolution (DE) exponential crossover is being used, parameter definition (ncross = 1). Other parameters such as Max. number of iterations Iter =10000, population size = 10 times the dimension of the function or 100 whichever maximum; crossover probability (pcross = 0.9), The scaling factor is being made random by the formula  $Fact = 0.5*(1+rand)$ . So, as the *rand* (random number changes) the value of F also changes. Random no. seed four digit number between -10000 to 10000; accuracy need;  $eps = 1.0e^{-08}$ . If x in f(x) violates the boundary then it is penalized to bring forcibly within specified limits through replacing it by a random number lying in the given limits of the function tested.

## 4.1.4 Experimental results and analysis

**Table 4.1** Comparative study of Genetic Algorithm (GA) and Modified Differential Evolution (MDE)

Results of some benchmark test problems					
Sn	Functions	Dim	GA	MDE	T. Value
1	Egg holder function [21]	2	0.00000	0.000000	0.000000
2	Corana function	4	0.00000	0.000000	0.000000

3	Freudenstein Roth	2	<b>0.00000</b>	-1.000000	0.000000
4	Leon function	30	-0.011039	<b>0.000000</b>	0.000000
5	Perm Function	4	-1.00000	<b>0.000000</b>	0.000000
6	Trid Function	6	<b>0.00000</b>	<b>0.000000</b>	-50.00000
7	Hougen Function	3	***	0.000000	-1.8013
8	Weierstrass Fun	30	<b>0.00000</b>	<b>0.000000</b>	0.00000
9	Niddle-eye Function	30	-1.00000	0.000000	-1
10	Shekel Function	4	0.00000	0.000000	0.00000

**Legend:** Sn: Serial No., Dim: Dimension of the function

**Note:** All the benchmark test function used for the study are given in APPENDIX A.

Ten benchmark functions are taken to compare the performance of Genetic Algorithms and Modified Differential Evolution. The performance of the algorithms shown in Table 4.1. The Dimension of each test functions are given in column two of the table. In the third and fourth column the results obtained by the algorithms are given. The results in bold face shows the values are very close to True value and also shows the better performance. It is observed that for Hougen Function Genetic Algorithms could not find the optimum value which has been shown by \*\*\* in the table.

#### 4.1.5 Summary of Section 4.1

It is clear from the above Table 4.1 that both the outperform for all the functions. For function 1 and 2 both GA and MDE are able to find the optimum value for both the function 1 and 2. But for freudenstein Roth function GA is able to find the optimum value and MDE fails. For Leon and perm function GA is not able to reach the global optimum but MDE reaches to optimum. For Trid function both method fails. For Hougen function GA overflows and MDE fails to reach the optimum. In case of Weierstrass and Shekel function both methods finds the optimum value but in case of Niddle-eye function GA is able to reach the optimum value whereas MDE fails. This work is published by Singh et. al. (2009).

## 4.2. A comparative study of Swarm Intelligence Optimization and Evolutionary Optimization

Recently many algorithms have been developed which mimics the natural procedure better known as Evolutionary Algorithm and claims to perform better than others. The

objective of this paper is to test the performance of Genetic algorithm and Particle Swarm (PS) method on some benchmark functions. Since Genetic Algorithm (GA) mimics the nature and Particle Swarm (PS) exploits the swarm intelligence, it will be interesting to see the performance of these two methods on the certain test functions. A brief idea of these functions are given in this section as follows. These functions are also represented by graph to facilitate conceptualization of the nature of these functions by visual means.

### 4.2.1 Introduction

Optimization is central to any problem involving decision making, whether in Mathematics, Engineering or in Economics. The area of optimization has received enormous attention in recent years primarily because of the rapid progress in computer technology, including the development and availability of user-friendly software, high speed and parallel processors. The optimization toolbox of MATLAB and other commercial software has given a new dimension to it.

### 4.2.2 Method and Algorithms used

Algorithms used for the comparative study are **Genetic Algorithm** and **Particle Swarm Optimization**. For all algorithms the dimensions were set manually based on the function used in the experiments.

### 4.2.3 Test functions used

The objective of this paper is to present a comparative study of the performance of the Genetic algorithm and particle swarm method on the functions such as Weierstrass function, Zettle function, Zero Sum Function, Dixon & Price function, Trid function, 6 Levy function No 3, Beale function, Booth Function, Easom function, Himmelblau function. These functions are difficult in nature and these functions are presented in details in APPENDIX A with graphical presentation.

#### 4.2.4 Experiment on Genetic Algorithm and Particle Swarm

**Genetic algorithms** We have used and input file to pass the different parameters i.e. npopsiz=5, pcross=9d0, npsibl= (2<sup>N</sup> N= powers of 2) pmutate=0.02d0 and maxgen=200. Another subroutine params.f was included in the main program having three parameters population size=200, nchrommax=60 and nparamax=10. Other two parameters are adjustable according to the dimensions of the test function.

**Particle Swarm setting** Particle Swarm has several parameters, population size=40. In most of the cases n=30 works fine. Its value can be increased up to 50 to 100. A randomly chosen neighbors =15. The maximum no. of decision variables =100, Number of iterations was set 1000.

#### 4.2.5 Experimental Results and Analysis

**Table 4.2** Comparative study of Genetic Algorithm (GA) and Particle Swarm Optimization (PSO)

SN	Functions	Dim	GA	PSO	True Value
1	Weierstrass function	2	<b>0.000000</b>	0.029900	0.7513280664284E-08
2	Zetle function	5	<b>0.000000</b>	0.00379	0.3791236557044E-02
3	Zero-sum Function #7	2	0.000000	<b>-1.000000</b>	-1.000000
4	Dixon & Price function	5	0.600000	<b>0.000000</b>	0.7368500368739E-08
5	Trid function	5	-2.000003	<b>-30</b>	-29.999999530
6	Levy function No 3	2	17.15224	<b>-176.54179</b>	-176.5417931343
7	Beale function	2	5.45315	<b>0.000000</b>	0.1080137747535E-09
8	Booth Function	2	20.99958	<b>0.000000</b>	0.4368455356320E-09
9	Easom function	2	<b>1.000000</b>	<b>1.000000</b>	0.9539719592261
10	Himmelblau function	2	<b>0.000000</b>	<b>0.000000</b>	0.1476885118888E-08

**Legend:** Dim Dimension

**Note:** All the benchmark test functions used for the study are given in APPENDIX A

Ten benchmark test functions are considered for the study of Genetic Algorithm and Particle Swarm Optimization. The results obtained in Table 4.2 show that the genetic algorithm has performed better than Particle Swarm optimization on functions 1 and 2, whereas for functions 3-8 Particle Swarm has given the result close to true value. For functions 9 and 10 both the algorithms have given the results close to true value. In Table 4.2 (the results shown in bold face are close to true value).

### 4.2.6 Summary of Section 4.2

Our program of Genetic algorithm has given a good result for the functions like Weierstrass function, Zettle function and Easom function but fails for other functions considered for experiments where the Particle Swarm (PS) have failed in these functions except the Easom function and Himmelblau function where both finds out the solution

Whereas Particle Swarm (PS) has performed better for Dixon Price function, Trig function, Levy #3 function, Zero sum function 7, Levy function 3, Bealy function, Booth function then Genetic Algorithm (GA) The results shown in the above table by bold where the methods has outperformed the other This shows that no algorithm is able to absolutely outperform the other This is a published work by Singh and Borah (2009)

## 4.3 Study of population based Meta-heuristics

Several variations of meta-heuristics have been developed recently and each of them claims to outperform others Through this paper we have done the comparative study of three methods, each of them has its origin in Von Neumann's Monte Carlo experiments We have tested these methods with certain benchmark test problems and some new test functions introduced by us first time

### 4.3.1 Methods used

Algorithms used for the comparative study were **Genetic Algorithm, Particle swarm Optimization and Modified Simulated Annealing**. For all algorithms the dimensions were set manually, based on the functions used in the experiments

### 4.3.2 Benchmark test functions used

We have used following bench mark test functions for comparative study The functions are Ackley function, Easom function, Griewank function, Beale function, Booth Function, Matyas function, Weierstrass function Michalewicz function, ,Simple

Quad Function and Hump function The detail explanation of these function are given in APPENDIX A with graphical representation

### 4.3.3 Experiment setup

**Genetic algorithms** We have used input file to pass the different parameters i.e.  $n_{popsiz}=5$ ,  $p_{cross}=9d0$ ,  $n_{psibl1} = (2^{*n} \text{ } n = \text{ powers of } 2)$   $p_{mutate}=0.02d0$  and  $maxgen=200$  Another subroutine to pass the parameter,  $params f$  was included in the main program having three parameters population size=200,  $n_{chrommax}=60$  and  $n_{paramax}=10$  Other two parameters are adjustable according to the dimensions of the problems

**Particle Swarm Optimization** Particle Swarm Optimization have several parameters population size=40, in most of the cases  $n=30$  works fine Its value can be increased up to 50 to 100 A randomly chosen neighbors  $=31$  The maximum no of decision variables  $maxX=100$ ,  $noofsteps=3$ , Number of iteration was set 1000

Here the algorithm allows each swarm is allowed to search one step left and right, up and down

**Modified Simulated Annealing** The parameter  $T$  is very crucial in using the SA Other parameters  $N$  is the dimension of the function can be changed from the parameter statement  $N=?$  VM step length  $T$  is imposed upon the system with the  $RT$  variable by  $T(I+1) = RT * T(i)$  The  $RT$  value was set 1.5

In a traditional SA for different random seed, results were different So, we modified the program to save the optimum value in a particular iteration by setting the extra variable  $ffopt$ , and  $indexopt$  to get the particular iteration which gave the value of  $ffopt$  We got these value printed This we called it as Modified SA

### 4.3.4 Experimental Results and Analysis

**Table 4.3** Comparative study of Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Simulated Annealing (SA)

Results of some benchmark test problems						
SN	Functions	Dim	GA	PSO	SA	T. Value
1	Ackley Fun	5	0.00000	0.000000	0.189945E-07	0
2	Easom Fun	2	-1.00001	-1.00000	-0.953971	-1
3	Griewank Fun	5	0.00000	0.000000	0.0172410	0
4	Beale Fun	5	5.45315	0.00000	0.1080137E-09	0

5	Weierstrass Fun	5	<b>0.00000</b>	0 02990	<b>0.7513280E-08</b>	0
6	Booth fun	2	-20 999	<b>0.00000</b>	<b>0.4368455E-09</b>	0 000000
7	Michalewicz Fun	2	*****	<b>-1.80130</b>	<b>-1.80130</b>	-1 8013
8	Simple Quad Fun	2	-3846 15	<b>-3872.7</b>	<b>-3873.7</b>	3873
9	Hump Fun	2	<b>-1.00000</b>	-1 03162	-1 03162	-1
10	Matya fun	2	0 00000	0 00000	0 4148318E-09	0 00000

**Legend:** Dim Dimension of the function, T Value True value

**Note:** All the benchmark test function used for the study are given in APPENDIX A

The performance of Genetic Algorithm, Particle Swarm Optimization and Simulated Annealing have been shown in Table 4.3 based selected ten benchmark test functions. It is clear that for function 1,2,3 the genetic Algorithm, Particle swarm optimization and Simulated Annealing have given the value close to true value for the functions 1, 2 and 3 respectively. Whereas for function 4 Particle Swarm and Simulated Annealing has given the result close to true value. For function 5 Genetic Algorithm and Simulated Annealing has given the results close to true value. For the functions 6-8 PSO and SA has performed better than GA where as GA has performed better for Hump function than PSO and SA. For Matya Function all the three algorithms have given the value close to true value. In the table (the results shown in bold face is close to true value)

### 4.3.5 Summary of Section 4.3

It is noted that non of the methods are able to outperform for all the functions. In functions 1-3, 10 three methods give the same results. Whereas for function 4 GA fails, 5-RPS fails, 6-GA fails, 7-GA overflows, 8, 10-Modified RPS & Modified SA outperforms GA, 9 This validates the No Free Lunch Theorem (NFLT). This is a published work by Singh and Borah (2009).

## 4.4 A comparative study of Particle Swarm Optimization and Simulated Annealing

The objective of this paper is to test the performance of Particle Swarm and Simulated Annealing on some test functions. Since Particle Swarm Optimizations mimics the nature and SA (Simulated Annealing) follows the physical criteria, it will be interesting to see the performance of these two methods on the certain benchmark test functions. A brief idea of these functions are given in this section as follows. These functions are

also represented by graph to facilitate conceptualization of the nature of these functions by visual means

#### **4.4.1 Introduction**

Optimization is central to any problem involving decision making, whether in Mathematics, Engineering or in Economics. The area of optimization has received enormous attention in recent years, primarily because of the rapid progress in computer technology, including the development and availability of user-friendly software, high speed and parallel processors. The optimization toolbox of MATLAB and many other commercial software like this has given a new dimension to it.

Extending the class of functions to include multimodal functions makes the global optimization problem unsolvable in general. In order to be solvable, some smoothness condition on the function in addition to continuity must be known.

#### **4.4.2 Methods used**

Algorithms used for the comparative study were **Particle swarm Optimization and Simulated Annealing**. For all algorithms the dimensions were set manually, based on the functions used in the experiments.

#### **4.4.3 Test functions used**

Functions such as Schaffer Function, Perm Function#1, Power-Sum Function, Weierstrass Function, Zero-Sum Function (N#7), Judge Function are used for comparative study and the detail explanation of these functions are given in APPENDIX A with the graphical presentation.

#### **4.4.4 Experiments**

(b) **Particle Swarm Optimization:** Particle Swarm Optimization has several parameters: population size 40, in most of the cases 30 works fine. Its value can be



increased up to 50 to 100 Randomly chosen neighbors 15 The maximum no of decision variables 100, The Local search Number of iteration was set 1000

(c) **Modified SA** The parameter T is very crucial in using the SA Other parameters N is the dimension of the function can be changed from the parameter statement  $N = ?$  VM step length T is imposed upon the system with the RT variable by  $T(I+1) = RT * T(I)$  The RT value was set 1.5

#### 4.4.5 Experimental Results and Analysis

**Table 4.4** Comparative study of Particle Swarm Optimization (PSO) and Simulated Annealing(SA)

Results comparing Particle swarm optimization and Simulated Annealing			
Test functions where PSO performs better than SA			
Name of Functions	PSO	SA	True Value
Schaffer function	0	9.72E-03	0
Perm function#1	0	16	0
Power Sum function4	0	100	0
Test function where SA performed better PSO			
Weierstrass function	0.0299	7.51E-09	0
Zero Sum function#7	1	0	0
Judge function	20.4050117	16.08173069	16.08173069

**Legend:** Dim Dimension of the function, T Value True value

**Note:** All the benchmark test function used for the study are given in APPENDIX A

From the Table 4.4 below we can conclude that for Schaffer function, Perm function and Power Sum Functions Particle Swarm Optimization has performed better than Simulated Annealing and given the results close to true Value whereas for Weierstrass function, Zero Sum and Judge function Simulated Annealing has given the value close to true value

#### 4.4.6 Summary of Section 4.4

Our program of Repulsive Particle Swarm has given a good result for the functions like Schaffer function, Perm function#1, Power-Sum Function 4 where Simulated Annealing have failed in these functions

Whereas Simulated Annealing has performed better for Weierstrass function, Zero-Sum function and Judge Function then Repulsive Particle Swarm This is a published work by Singh and Borah (2009)

\*\*\*\*\*

# CHAPTER 5

---

## Concluding Discussion

### 5.1 Discussion and conclusion for empirical study on evolutionary algorithms using new test functions

1 In this thesis eleven test new functions were proposed for the empirical study of evolutionary algorithms. The graphical representations of these test functions gave a shape of the test functions. The features of the all the eleven test functions have been spell out. These eleven test functions have been generalized up to  $n$  variable, which can be used as a higher dimension, by simply putting the value of  $n$  i.e  $n=500$  in the optimizer. The difficulty of these test functions can be controlled by the appropriate choice of some parameters. These functions have been coded in MATLAB to find the optimum value of the functions using the different global optimizer.

2 Two experiments have been performed. Based on first experiment, the results recorded in Table 3.1 (a) and (b) it is found that Particle Swarm Optimization has performed better on F204 to F210 where as on for F201, F202, F208, F209 and F210 Simulated Annealing has found the best optimum value. Genetic Algorithm and Differential Evolution have not performed better except the function F208 and F209. The Genetic Algorithm did not converge at all on F210. In the second experiment, results recorded in Table 3.2 (a) & (b) and 3.3 (a) & (b), Differential Evolution and Genetic Algorithm converges to optimum when the number of iterations have been increased. But, from the Table 3.4 (a) & (b) and 3.5 (a) & (b) the results do not improve when the number of iterations are increased. Hence this result validates the "No free lunch theorem empirically, though it has been proved theoretically also.

3 Depending on the classes of test functions there are corresponding evolutionary algorithms classes. Finding a suitable corresponding evolutionary algorithm for a specific class of application is difficult in general. However, in this research by analyzing the features of new and benchmark test functions used in this study it is

found a hint that the Differential Evolution and Genetic Algorithm performs better on numerical class of optimization problem when the number of iterations are increased whereas the results does not improve when the number of iterations are increased in Particle Swarm Optimization and Simulated Annealing Hence it is concluded that Particle Swarm optimization and Simulated Annealing perform better than Genetic Algorithm and Differential Evolution on class of multi-modal function

## 5.2 Comparative study (Analysis and conclusion)

**The Chapter 4** contains the reprint of the published paper Here we have studied the comparative performance of the four optimizers using the benchmark test function which are non-convex, non differentiable, noisy and deceptive in nature

In the first of the comparative performance of modified Differential Evolution and Genetic Algorithm on some non linear, non-convex and noisy test function have been studied In 4 1 5 the result clearly shows that none of the algorithm is able to outperform on all the test function considered for studies and hence validating the “No Free Lunch Theorem” by wolpert and Macready (1997) and Ho and Pepyne (2002)

In the second publication is on the comparative study of Swarm Intelligence and Evolutionary Optimization method where the comparative performance of Particle Swarm Optimization and Genetic Algorithm on some noisy numerical benchmark test problems have been studied and reflected in conclusion 4 2 6 pp 49 that none of the algorithm could outperform all the test functions into consideration and validates the “No Free Lunch Theorem” by wolpert and Macready (1997) and Ho and Pepyne (2002)

Third paper studies the performance of population based meta-heuristics on some non-convex noisy deceptive benchmark test functions Here Genetic Algorithm, Particle Swarm Optimization and Modified Simulated Annealing on the set of test function have been studied In 4 3 5 pp 51 validates the “No Free Lunch Theorem” by wolpert and Macready (1997) and Ho and Pepyne (2002)

In the forth publication the comparative study was done on Repulsive particle Swarm Optimization and Simulated Annealing on some numerical Benchmark test functions

In 446 pp 54 validates the “No Free Lunch Theorem” by Wolpert and Macready (1997) and Ho and Pepyne (2002)

### **5.3 Future direction of research**

Since the meta heuristics techniques is growing fast and third generation Evolutionary Computings like Artificial Immune Systems by Farmer et al (1986), Cultural Algorithms by Reynolds (1994), DNA Computing, similar to parallel computing which takes advantages of many different molecules of DNA to try many different possibilities at once, developed by Adleman (1994), Estimation of Distribution Algorithms some times called Probabilistic Model-Building Genetic Algorithm by Larrañaga and Lozano (2002) are being extensively used for complex computations and NP-hard problems Artificial Immune systems have been used in robotics computation, analysis of adaptive control and optimization Adleman (1994) solved Hamiltonian path problems using DNA computing Estimation of distribution algorithm has been used extensively in finding the protein structure predictions and genomics etc Again these algorithms do not have strong convergence conditions and theoretical background So, there will be a need to develop the test problem and study these forth generation Evolutionary computing empirically Another direction of research could be the development of test problem and study the characterizes of the test cases Another direction will be to use these test cases to study the performance of these forth generation Evolutionary Computings

\*\*\*\*\*

# References

---

- [1] Ackley, D H An Empirical study of vector function optimization, *Morgan Kaufmann Publisher*, 170-204 (1987)
- [2] Adorio, E P MVF-Multivariate Test Functions Library in C for Unconstrained Global Optimization (2005)
- [3] Addis, B and Locatelli, M A new class of test functions for global optimization, *Journal of Global Optimization*, **38**, 479-501 (2007)
- [4] Adleman, L Molecular computation of solutions to combinatorial problems, *Science*, **266**,1021-1024(1994)
- [5] Andrei, N An unconstrained optimization test functions collection, *Advanced Modeling and Optimization* **10**, 147-161 (2008)
- [6] Ali, M M and Torn A Population set-based global optimization algorithms some modifications and numerical studies, *Computers Operations Research*, **31**(10), 1703-1725 (2008)
- [7] Aluffi-Pentini, F, Parisi, V and Zirilli, F Global optimization and stochastic differential equations, *J Optimization Applications*, **47**(1), 1-16 (1985)
- [8] Averick, B M, Carter, R G and Moré, J J The MINPACK-2 *Test Problem Collection (Preliminary Version)*, Argonne National Laboratory, Mathematics and Computer Science Division, Technical Memorandum No 150 (1991)
- [9] Baker, K R Test Results for an Interval Branch and Bound Algorithm for Equality-Constrained Optimization *Preprint of the article appeared in state of the art in Global Optimization*, C Floudas and P Pardalos, Kluwer, Dordrecht, 181—200 (1996)
- [10] Back, T, Hoffmeister F and Schwefel, H P A Survey of Evolution Strategies, in *Proceedings of the Fourth International Conference on genetic Algorithms*, 2-9 (1991)
- [11] Bethke, A D *Genetic Algorithms as a function optimizers*, Doctoral Dissertations, Abstracts International, **41**(9), 3503B, (University of Michigan, 1980)
- [12] Bongartz, I, Conn, A R, Gould, N and Toint, Ph L (CUTE) constrained and unconstrained testing environment , *ACM Transactions on Mathematical Software*, **21**(3), 123-160 (1995)

## References

- [13] Box, G E P Evolutionary operation A method of increasing industrial productivity *Applied Statistics*, **6**, 2-13 (1958)
- [14] Brest, J , Greiner, S , Bošković, B Mernik, M and Žumer, V Self-adapting control parameters in differential evolution a comparative study on numerical benchmark problems, *IEEE Trans Evolutionary Computation*, **10**, 646-657 (2006)
- [15] Bremmerrmann, H J *Optimization through evolution and recombination*, M C Yovits, G T Jacoby and G D Goldstein, eds , Self-Organizing System (Spartan Books, Washington D C ,1962)
- [16] Bukin, A D *New Minimization Strategy for Non-smooth Functions* Budker Inst of Nucl Phys Preprint BUDKER-IMP-1997-79 (Novosibirsk 1997)
- [17] Chattopadhyay, R A study of test functions for optimization algorithms, *Journal of optimization theory and applications*, **8(3)** 231-236 (1971)
- [18] Chichinadze, V K *Solving Nonlinear and Nonconvex Optimization Problems*, (Nauka, Moscow, in Russian,1983)
- [19] Chiam, S C , Goh, C K and Tan, K C Adequacy of Empirical Performance Assessment for Multiobjective Optimizer S Obayashi et al (Eds ) EMO , LNCS 4403, 893-907 (2007)
- [20] Corana, M M , Martini C and Ridella S Minimizing multimodal functions of continuous variables with the Simulated Annealing Algorithm, *ACM Trans Math Soft* **13(3)**, 262-280 (1987)
- [21] Colville, A R A comparative Study on Nonlinear Programming Codes, IBM Scientific Center Report, New York, 320-2949 (1986)
- [22] Coello-Coello, C A *An updated survey of GA-Based Multiobjective Optimization Technique* technical Report Lania-RD-09-08, Laboratorio Nacional de Informática Avanzada (LANIA), Xalapa, Veracruz, (México, 1998)
- [23] Davis, P M *Cognition and learning A review of the literature with reference to ethnolinguistic minorities* (Dallas, TX Summer Institute of Linguistics, Review by Parrish,1991)
- [24] Davis, L *Genetic Algorithm and Simulated Annealing*, Morgan Kaufmann (1987)

## References

- [25] Davis, L and Steenstrup, M Genetic Algorithm and Simulated Annealing An Overview, Morgan Kaufmann Publisher, 1-11 (1986)
- [26] De Jong, K A *An analysis of the behaviour of a class of genetic adaptive systems* PhD thesis, (University of Michigan ,1975)
- [27] De Jong, K A and Morrisson, R W A test problem generator for Non stationary Environments In proceedings of the *IEEE Congress on Evolutionary Computation, IEEE Press*, 2047-2053, (1999)
- [28] Deb, K *Genetic algorithms in multimodal function optimization*, Master's Thesis, TCGA Report No 89002, (Tuscaloosa University of Alabama, 1991)
- [29] Deb, K *Multiobjective Optimization using Evolutionary Algorithms*, (Chichester, UK, Wiley 2001)
- [30] Dennis, J E and Schnabel, R B *Numerical Methods for unconstrained optimization and Nonlinear Equations* (Prentice-Hall, 1983)
- [31] Dennis, J E, Gay, D M and Vu, P A A new nonlinear equations test problem, Technical Report, Mathematical Sciences Department, Rice University, 83-16 (1985)
- [32] Dekkers, A and Aarts, E Global optimization and simulated annealing, *Mathematical Programming*, **50(1)**, 367-393 (1991)
- [33] Dixon, L C W and Szego, G *Towards Global Optimization*, (North-Holland, 1975)
- [34] Dixon, L C W and Szego, G P *Towards Global Optimization 2*, (North-Holland, and Amsterdam, 1978)
- [35] Dorigo, M *Optimization, Learning and Natural Algorithm*, Ph D Thesis Dip Electronica & Informazione, Ant Coloney optimization web page, <http://iridia.ulb.ac.be/mdorigo/ACO/ACO.html> (Politecnico de Milano, Italy, 1992)
- [36] Dorigo, M Editorial Introduction to the Special Issue on Learning Autonomous Robots (Editorial) *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, **26(3)**, 361-364 (1996)
- [37] Eberhart, R C and Kennedy, J A New Optimizer using Particle Swarm Theory, *Proceedings sixth symposium on Micro Machine and Human Science*, IEEE Service center Piscataway, N J, 39-43(1995)
- [38] Eberhart, R C, Simpson, P and Dobbins R *Computational Intelligence PC Tools* (Academic Press, 1996)

## References

- [39] Easom, E E *A Survey of Global Optimization Techniques* M Eng thesis (Univ Louisville, KY, Louisville, 1990)
- [40] Farmer, J D, Packard, N H and Perelson, A S The immune system, adaptation and machine learning *Physica D* **22** 187-204 Reprinted in *Evolution, Games and Learning*, D Farmer, A Lapedes, N Packard and B Wendroff, eds North Holland, Amsterdam , **22**,187-204 (1986)
- [41] Floudas, C A *et al Handbook of Test Problems in Local and Global Optimization*, Dordrecht, E Hansen, Global Optimization Using Interval Analysis, Marcel Dekker, (Kluwer Academic Publishers, New York, 1999)
- [42] Floudas, C A and Pardalos, P M *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Lecture Notes in Computer Science, **455**, Springer-Verlang (1987)
- [43] Fogel, D *Evolutionary Computation Towards a New Philosophy of Machine Intelligence* (IEEE Press, Piscataway, N J , 1996)
- [44] Fogel, L J Inanimate Intellect through Evolution, *Naval Research Reviews*, **20(11)**, 9-18(1967)
- [45] Fogel, L J, Owens, A J and Walsh M J *Artificial Intelligence through Simulated Evolution* (Wiley, 1996)
- [46] [ftp //138 48 4 14/pub/cute/](ftp://138.48.4.14/pub/cute/) CUTE Constraint and Unconstrained Testing Environment, NHSE Software Catalogue, (1995)
- [47] Friedberg, R M *A learning machine part 1 IBM Journal of Research and Development*, **2**,2-13(1958)
- [48] Gaviano, M , Kvasov,D E , Lera,D , Sergeyev, Y D Algorithm 829 software for generation of classes of test functions with known local and global minima for global optimization *ACM Trans Math Software* **29**, 469–480 (2003)
- [49] Global Optimization web site by A Neumaier, [http //www mat univie ac at/~neum/ glopt/test.html](http://www.mat.univie.ac.at/~neum/glopt/test.html)(2005)
- [50] Glover, F Future path for the Integer programming and Links to Artificial Intelligence, *Computer and Operation Research*, **13**,533-549 (1986)
- [51] Glover, F Parametric tabu-search for mixed integer programs *Computers & OR* **33**, 2449-2494 (2006)



## References

- [52] Glover, F and Kochenberger, G *Handbook of Metaheuristics* (Kluwer Academic Publishers, 2002)
- [53] Goldberg, D E *Genetic Algorithms in Search, Optimization and Machine Learning* (Kluwer Academic Publishers, Boston, MA, 1989)
- [54] Goss, S, Aron, S, Deneubourg, J L and Pasteels, J M Self-organized shortcuts in the Argentine ant *Naturwissenschaften*, **76**,579–581, (1989)
- [55] He, J, and Yao, X Towards an analytic framework for analyzing the computation time of evolutionary algorithms *Artificial Intelligence*, **145**, 59-97 (2003)
- [56] Holland, J H Outline for logical theory of adaptive systems, *Journal of Association for Computing Machinery*, **3**, 297-314 (1962)
- [57] Holland, J H *Adaptation in Natural and Artificial System* (University of Michigan Press, Ann Arbor, 1975)
- [58] Horst, R and Pardalos, P M *Handbook of Global Optimization* (Dordrecht, Netherlands Kluwer, 1995)
- [59] Hock, W and Schittkowski, K *Test Examples for Nonlinear programming code* (Lecture notes in Economics and Mathematical Systems, **187**, Springer-Verlag, 1981)
- [60] Hsieh, T *Particle Swarm Guided Evolution Strategy for Real-Parameter Optimization*, N C Lab Report No NCL-TR-2006007, Natural Computing Laboratory (N C Lab), Department of Computer Science, (National Chiao Tung University, 2006)
- [61] Huang, V L, Suganthan, P N and Liang, J J Comprehensive Learning Particle Swarm Optimizer for Solving Multi-objective *Optimization Problems*, *International Journal of Intelligent Systems*, (Wiley Periodicals, Inc Published online in Wiley Inter Science [www.interscience.wiley.com](http://www.interscience.wiley.com)), **21**,209-226 (2006)
- [62] <http://www.maths.uq.edu.au/CEToolBox/node3.html#SECTION00021300000000000000>, Non-Linear Continuous Multi-External Optimization (2004)
- [63] [http://www.netlib.org/utk/misc/sw\\_survey/urc/html/2151.html](http://www.netlib.org/utk/misc/sw_survey/urc/html/2151.html), NHSE Software Catalogue (1995)
- [64] <http://www.netlib.org/cgi-bin/nhse/nphredirect?ftp://138.48.4.14/pub/reports/cute.ps.gz>, NHSE Software Catalogue (1995)

## References

- [65] Jason, G. D. and Konstantinos G. M. An experimental study of benchmarking functions for genetic algorithms, *International Journal of Computer Math.*, Taylor and francis, **79(4)**, 403–416(2002).
- [66] Jorge, J. M, Garbow B. S. and Hillstorn, K. E. Testing Unconstrained Optimization software, *ACM Trans. Math. Software* **7(1)**, 136-140 (1981).
- [67] Jansson, C and Knüppel, O. *A Global Minimization Method: The Multi-Dimensional Case*, Report 92.1, Technische Universit'at Hamburg-Harburg, (Germany, 1992).
- [68] Jansson, C. and Knéuppel, O. *Numerical Results for a Self-validating Global Optimization Method*. Technical Report 94-1. TU (Hamburg; Harburg, 1994).
- [69] Kennedy, J. and Eberhart, R. C. *Particle swarm optimization*", in Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, N.J.,1942–1948 (1995).
- [70] Kennedy, J., Eberhart, R. C. and Shi, Y. *Swarm Intelligence* ( Morgan Kaufmann Publishers, 2001).
- [71] Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. Optimization by Simulated Annealing, *Science*, **220**, 671-680 (1983).
- [72] Kirkpatrick, S. Optimization by Simulated Annealing -Quantities Studies. *Journal of Statistical Physics*, **34**,75-986 (1984).
- [73] Kowalik, J. and Osborne, M. R. *Methods for Unconstrained Optimization Problems* (New York: American Elsevier Publishing Company Inc, 1968).
- [74] Koza, J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press. ISBN 0-262-11170-5 (1992).
- [75] Lavor, C.,Maculan, N.: A function to test methods applied to global minimization of potential energy of molecules. *Num. Algorithms* **35**, 287–300 (2004) .
- [76] Larrañga, P. and Lozano, J. A. *Estimation of distribution algorithms: A new tool for evolutionary computation* (Kluwer Academic Publishers, Boston, 2002).
- [77] Levy, A. M., Gomez, S. and Galderon, A. *Topics in Global Optimization*, Lecture Notes in Mathematics No. 909, (New York: Verlag-Springer, 1981).

## References

- [78] Liang, J J, Qin, A K, Suganthan, P N and Baskar, S Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE Trans Evolutionary Computation*, **10(3)**, 281-295(2006)
- [79] Liang, J J, Suganthan, P N and Deb, K Novel Composition test functions for numerical global optimization, *Proc Of IEEE International Swarm Intelligence Symposium*, 68-75 (2005)
- [80] Liang, J J and Suganthan, P N Dynamic Multi-swarm Particle Swarm Optimizer, *Proc of IEEE International Swarm Intelligence Symposium*, 124-129 (2005)
- [81] Lalescu L FET Manual, [http //www timetabling de/manual/FET-manual en.html](http://www.timetabling.de/manual/FET-manual.en.html) (2009)
- [82] Mishra, S K Some New Test Functions for Global Optimization and Performance of Repulsive Particle Swarm Method *SSRN* [http //ssrn com/abstract=926132](http://ssrn.com/abstract=926132) (2006)
- [83] Michalewicz, Z *Genetic Algorithms, Numerical Optimization and Constraints*, Proceedings of the Sixth International Conference on Genetic Algorithm, (Morgan Kaufmann, San Mateo, CA, 1985)
- [84] Michalewicz, Z *Genetic Algorithms+Data Structure=Evolution Program*, (New York Springer, 1999)
- [85] Moré, J J, Garbow, B S and Hillstom, K E Testing unconstrained optimization software, *ACM Trans Math Software* **7**, 17-41(1981)
- [86] Moré, J J, Garbow, B S and Hillstom, K E Algorithm 566 FORTRAN subroutines for testing unconstrained optimization software, *ACM Trans Math Software*, **7(1)**, 136-140 (1981)
- [87] Moscato, P *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts Towards Memetic Algorithms* (Caltech Concurrent Computation Program ,report 826, 1989)
- [88] Nagendra, S *Catalogue of Test Problems for Optimization Algorithm Verification*, Technical Report 97-CDR-110, and (General Electric Company, 1997)
- [89] Neumaier A Shcherbina, O, Huyer W, Vinkó T, A comparison of complete global optimization solvers *Math Program* **103(2)**, 335-356 (2005)
- [90] Oldenius, R MATLAB CENTRAL (An open exchange for Matlab and Simulink community)

## References

- <http://www.mathworks.com/matlabcentral/fileexchange/24838-godlike-a-robust-single-multi-objective-optimizer>, (2009)
- [91] Pintér, J D *Global Optimization in Action Continuous and Lipschitz Optimization Algorithms, Implementations, and Applications*, Dordrecht, Boston Kluwer Academic Publishers, (QA402 5 Pin, and Science Library, 1996)
- [92] Pintér, J D Global optimization software, test problems and applications In Pardalos P M and Romeijn H E (eds), *Handbook of Global Optimization, Volume 2* Kluwer Academic Publishers, Dordrecht, Boston, London (2002)
- [93] Price, K V Genetic Annealing, *Dr Dobb's Journal*, 127-132 (1994)
- [94] Price, K V, Storn, R M and Lampinen, J A *Differential Evolution a Practical Approach to Global Optimization* (Berlin Springer, 2005)
- [95] Rechenberg, I *Cybernetic solution path of an experimental problem Royal Aircraft Establishment*, (Farnborough p Library Translation 1122, 1965)
- [96] Reynold, R G An Introduction to Cultural Algorithms, in *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, World Scientific Publishing, 131–139(1994)
- [97] Schwefel, H P *Cybernetic Evolution as Strategy for Experimental Research in Fluid Mechanics* Diploma Thesis, Hermann Fottinger-Institute for Fluid Mechanics, (Technical University of Berlin, 1965)
- [98] Schwefel, H P *Evolutionsstrategie und numerische Optimierung Ph D thesis*, Technische Universität Berlin English translation Evolution strategy and numeric optimization, Ph D thesis, (Technical University of Berlin, 1975)
- [99] Schwefel, H P *Numerische Optimierung von Computer-Modellen mittels der Evolution sstrategie* (Basel Birkhauser English Translation Numerical Optimization of computer models by means of evolution strategies (Basel Birkhauser, 1977)
- [100] Schwefel, H P *Numerical Optimization of Computer Models* (Chichester, Wiley & Sons, 1981)
- [101] Schwefel, H P *Evolution and Optimum Seeking*, (John Wiley, Chichester, UK, 1995)

- [102] Schutte, J F and Groenwold, A A A study of Global Optimization Using Particle Swarms *Structural and Multidisciplinary Optimization*, **25(4)**, 261-269 (2003)
- [103] Schittkowski, K , More test examples for nonlinear programming *Lecture Notes in Economics and Mathematical Systems*, 282 Springer, Berlin (1987)
- [104] Schoen, F A wide class of test functions for global optimization *Journal of Global Optimization Optim* **3**, 133–137 (1983)
- [105] Shcherbina, O *Benchmarks for Global Optimization and Continuous Constraint Satisfaction*, Institut für Mathematik , Universität Wien Strudlhofgasse 4, A-1090 (Wien, Austria, 2002)
- [106] Shcherbina, O , Neumaier, A , Sam-Haroud, D and Tuan-Viet Nguyen, X V *Benchmarking Global Optimization and Constraint Satisfaction Codes* (University Vienna, Austria, 2003)
- [107] Singh, S K and Borah, M A comparative study of Repulsive Particle Swarm Optimization and Simulated Annealing on some Numerical Benchmark Problems, *International Journal of Computational Intelligence Research*, **5(1)**, 75-82 (2009)
- [108] Singh, S K and Borah, M Performance of Population based Meta-heuristics on some Non-Convex Noisy deceptive Test Functions *Global Journal of Pure and Applied Mathematics*, **5(1)**, 9-14 (2009)
- [109] Singh, S K and Borah, M A Comparative Study of Swarm Intelligence Optimization Method and Evolutionary Optimization Method on Some Noisy Numerical Benchmark Test Problems *International Journal of Computation and Applied Mathematics*, **4(1)**, 1-9 (2009)
- [110] Singh, S K , Borah, M and Jha, UC Comparative study of modified differential evolution on some nonlinear, non convex and noisy test functions *International Journal of Mathematical Modeling and Simulation Application* **2(2)**, 156-162 (2009)
- [111] Srinivas, N and Deb, K Multi-objective function optimization using non-dominated sorting genetic algorithms" *Evolutionary Computation Journal*, **2(3)**, 221-248 (1994)
- [112] Suganthan, P N *et al* Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Technical Report, Nanyang Technological University, Singapore, May, also KanGAL (Report 2005005, IIT Kanpur, India, 2005)
- [113] Storn, R M Constrained Optimization, *Dr Dobb's Journal*, 119-123 (1995)

- [114] Storn R M and Price, K *Differential Evolution – Simple and efficient adaptive scheme for global optimization over continuous space* (Technical Report TR-95-012, ICST, 1995)
- [115] Storn, R M and Price, K *Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces*, *Journal of Global Optimization, Kluwer Academic Publishers*, **11**, 341 – 359 (1997)
- [116] Styblinski, M and Tang, T *Experiments in Non-convex Optimization Stochastic Approximation with Function Smoothing and Simulated Annealing // Neural Networks* **3**, 467-483 (1990)
- [117] Torn, A and Zilinskas, A *Global Optimization*, (Springer-Verlag, Berlin, 1989)
- [118] Tu, Z G and Yong, L *A robust stochastic genetic algorithm (St GA) for global numerical optimization*, *IEEE Trans Evolutionary Computation*, **8(5)**, 456-470 (2004)
- [119] Tsutsui, S and Fujimoto, Y *Forking genetic algorithm with blocking and shrinking modes*, *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA 93)*, Morgan Kaufmann Publishers, 206-213 (1993)
- [120] Tsutsui, S, Ghosh, A, Fujimoto, Y and Corne, D *A bi-population scheme for real-coded GAs the basic concept*, *Proceedings of the First International Workshop on Frontiers in Evolutionary Algorithms*, **(1)**, 39-42 (1997a)
- [121] Tsutsui, S, Fujimoto, Y and Ghosh, A *Forking GAs GAs with search space divisions schemes*, *Evolutionary Computation* (1997b)
- [122] Van Iwaarden, R J *An Improved Unconstrained Global Optimization Algorithm* Ph.D thesis (Univ of Colorado at Denver Denver, 1996)
- [123] Vesterstrom, J and Thomsen, R *A comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems*, *Congress on Evolutionary Computation, CEC 2004*, **2**, 1980-1987 (2004)
- [124] Voigt, H M, Born, J and Santibanez-Koref, I *A Multivalued Evolutionary Algorithm* Technical Report TR-92-038 (Intern Comp Sci Inst Berkeley, CA, 1992)

References

- [125] Whitley, D., Rana, S. B., Dzubera, J. and Mathias, K. E. Evaluating evolutionary algorithms, *Artificial Intelligence*, **85(1)**, 245-276 (1996).
- [126] Wolpert, D.H. and Macready, W. G. No Free Lunch Theorem for Optimization, *IEEE transaction on evolutionary computation*, **1(1)**, 67-82 (1997).
- [127] Yao, X., Liu, Y. and Lin, G. Evolutionary programming made faster, *IEEE Trans. Evolutionary Computation*, **3(2)**, 82-102 (1999).
- [128] Ho, Y. C. and Pepyne, D. L. Simple explanation of the no free lunch theorem of optimization, *Cybernetics and Systems Analysis*, **38(2)**, 292-298 (2002).

\*\*\*\*\*

# Appendix A

---

## Benchmark test functions for unconstrained global optimization

Testing any global optimization algorithm depends on the benchmark test problems taken into consideration. Testing the algorithms with test function with mild difficulty may not validate the algorithm. So, it is important to consider the wide variety of test functions with the degree of difficulties. In the field of global optimization there exist a set of test functions with a limited dimension and mild difficulties. Therefore testing any Global Optimization (GO) with that algorithm is not appropriate way to validate the algorithm. We have collected the large class of test function to validate the Global optimization. The test functions have been defined with the magnitude of difficulties.

The difficulties of Global Optimization problem depends on many factors. Among the most relevant ones is the size of basin of attraction of the Global Optimizer, the shape of the function around the global optimizer, the classical example of the being the Rosenbrock function where the minimum point is inside a long narrow and a parabolic-shaped flat valley, which makes convergence difficult, dimension and high multimodality.

This appendix is a collection of test functions used to test the performance of the algorithms used for comparative study. The benchmark test functions are deceptive in nature, non-convex noisy. In most of the cases of these test problems, traditional method is not able to find the optimum value, whereas these algorithms are able to find the optimum values. So, the comparative studies have investigated which algorithm performs better on these test suits. The five test functions constructed by De-Jong (1975) popularly known as De-Jong's five test suit, four are uni-modal containing only one optimum point, where are other test functions are multimodal containing multi optimum point. Sphere function smooth, uni-modal, strongly convex, symmetric, but only one optimum point. Rosenbrock is considered to be difficult, because it has a very narrow ridge. The tip of the ridge is very sharp, and it runs around a parabola. Algorithms that are not able to discover good directions underperform in this problem. Step function is the representative of the problem of flat surfaces. Step function is piecewise continuous step function. Flat surfaces are obstacles for optimization algorithms, because they do not give any information as to which direction is favorable. Unless an algorithm has variable step sizes, it can get stuck on one of the flat plateaus. The background idea of the step function is to make the search more difficult by introducing small plateaus to the topology of an underlying continuous function.

Quartic function is a simple uni-modal function padded with noise. The Gaussian noise makes sure that the algorithm never gets the same value on the same point. Algorithms that do not do well on this test function will do poorly on noisy data. Foxholes function is an example of a function with many local optima. Many standard optimization algorithms get stuck in the first peak is find. The Schwefel, Rastrigin, Griewangk functions are typical examples of non-linear multimodal functions. Rastrigin's function is a fairly difficult problem for genetic algorithms due to the large search space and large number of local minima. Rastrigin has a complexity of  $O(n \ln(n))$ , where  $n$  is the number of the function parameters. This function contains millions of local optima in



the interval of consideration Schwefel's function is somewhat easier than Rastrigin's function, and is characterized by a second-best minimum which is far away from the global optimum

The test problems Nagendra (1997) have different complexity and difficulties. Some functions are noisy in nature. Some are dented non-differentiable and deceptive in nature. The visual presentation of these functions gives the characteristic and some idea about the number of optimum (i.e. local minima or local maxima) and also the complexity of the function. It can be seen from the graphical presentation that the test function are highly multi-modal. The afford was to collect and present the characteristic of the functions in a most exhaustive manner but this may not be the complete list of test functions. Also together with the analysis some new test functions are evolved which is presented in chapter 2.

More precisely the functions from 1 to 80 are the bench mark test functions adopted from different web site the reference is given below. The graphs of these test functions have been redrawn. For each functions four colored dimensional graph e.g. meshz & mesh, surf, surfc and surf1 of MATLAB have been used. The code of the functions is summarized in the code and reference chapter. The functions from 81 to 90 have been adopted from <http://ssrn.com/abstract=926132> and the graph of all those functions has been redrawn by us. Functions from 90 to 182 the functions have been adopted from the CUTE (CUTE Constraint and Unconstrained Testing Environment). Again the functions from 183 to 199 has been adopted from CUTE but we have kept these functions as another set for the different nature of the problem. The survey of the test functions are from the following [www.maths.uq.edu.au/CEToolBox/node3.html#SECTION\\_0002130000\\_0000000000](http://www.maths.uq.edu.au/CEToolBox/node3.html#SECTION_0002130000_0000000000), [http://www.netlib.org/utk/misc/sw\\_survey/urc/html/2151.html](http://www.netlib.org/utk/misc/sw_survey/urc/html/2151.html) and <http://www.netlib.org/cgi-bin/nhse/nphredirect?ftp://138.48.4.14/pub/reports/cute.pl.gz>. The reference of other test functions could be traced from (though some of them may not be the original source) are Chattopadhyay (1971), More et al (1981), Jorge et al (1981), Hock and Schittkowski (1981), Nagendra (1987), Floudas and Pardalos (1987), Floudas et al (1999), Averick et al (1991), Jansson and Knuppel (1992, 1994), Bongartz et al (1995), Van-Iwaarden (1996), Baker (1996), Adorio (2005), Mishra (2006) and Andrei (2008).

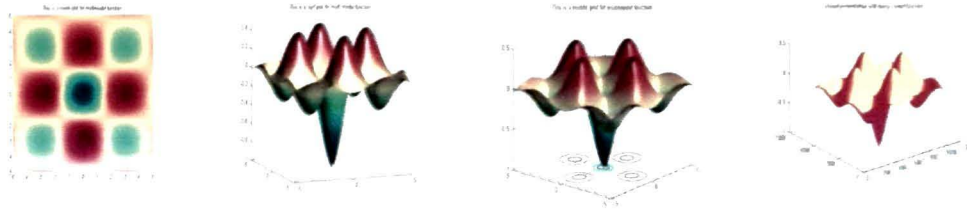
These benchmark test functions are being redrawn with mesh, meshc, surf and surf1 are created using MATLAB 7.1 version with different angles.

## (i) First set of test functions

**1. A typical multi-modal function** A multi-modal (non-convex) function (in 2 variables,  $m = 2$ ) is

$$f(x) = \cos(x_1)\cos(x_2) \exp\left[-(1/4)\sqrt{x_1^2 + x_2^2}\right] \quad -5 \leq x_i \leq 5 \text{ and}$$

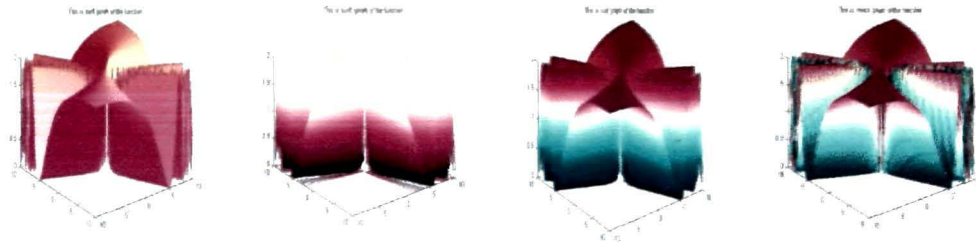
$i = 1, 2$ . It has global minimum  $f(x^*) = -1$  at  $x^* = (0, 0)$ . The function is graphically represented below



**2. A typical non-linear function:** Consider the function in  $m$  variables ( $m \geq 2$ ) that has the optimal value  $f(x^*) = 0$  in the search domain  $|x_i| \leq 10; i = 1, 2, \dots, m$  given as

$$f(x) = \sum_{i=2}^m \cos[|x_i - x_{i-1}| / |x_i + x_{i-1}|] + (m-1)$$

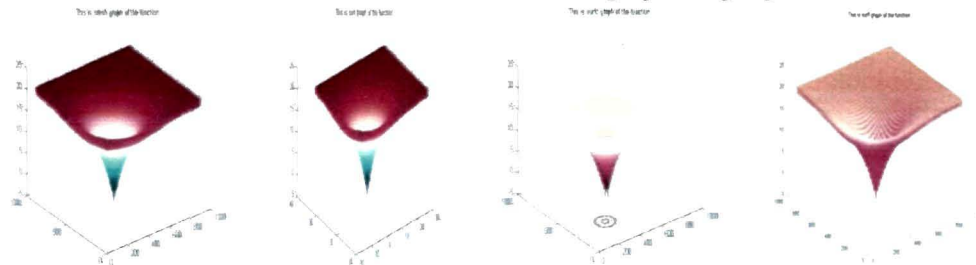
The function is graphically represented below



**3. Ackley function:** An  $m$ -variable ( $m \geq 1$ ) function with search domain  $[-15 \leq x_i \leq 30]$  for  $(i = 1, 2, \dots, m)$  given by Ackley (1987)

$$f(x) = 20 + \exp(1) - 20 \exp \left[ -0.2 \left( \frac{\sum_{i=1}^m x_i^2}{m} \right)^{0.5} \right] - \exp \left[ \frac{1}{m} \sum_{i=1}^m \cos(2\pi x_i) \right]$$

is called the Ackley function. It is a multi-modal function. The global minimum of this function is  $f(x^*) = 0$  for  $x^* = (0, 0, \dots, 0)$ . The function is graphically represented below



**4. ANN (Artificial Neural Network's) XOR function:** This function is in nine variables. It is defined as follows:

$$f = f_1 + f_2 + f_3 + f_4$$

where

$$f_1 = \left[ 1 + \exp \left\{ \frac{-x_7}{1 + e^{(-x_1 - x_2 - x_5)}} - \frac{x_8}{1 + e^{(x_3 - x_4 - x_6)}} - x_9 \right\} \right]^{-2}$$

$$f_2 = \left[ 1 + \exp \left\{ -\frac{x_7}{1 + e^{(-x_5)}} - \frac{x_8}{1 + e^{(-x_6)}} - x_9 \right\} \right]^{-2}$$

$$f_3 = \left[ 1 - \left[ 1 + \exp \left\{ -\frac{x_7}{(1 + e^{(-x_4 - x_5)})} - \frac{x_8}{(1 + e^{(-x_3 - x_6)})} - x_9 \right\} \right]^{-1} \right]^2$$

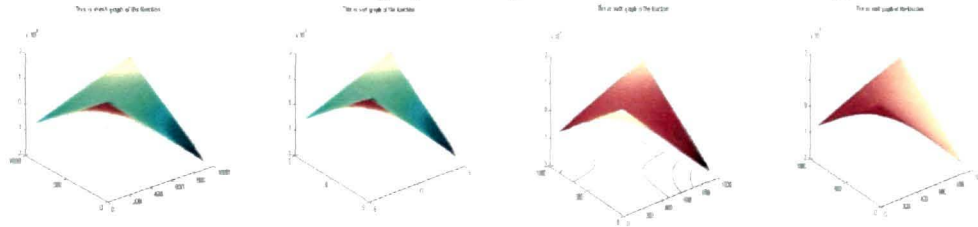
$$f_4 = \left[ 1 - \left[ 1 + \exp \left\{ -\frac{x_7}{1 + e^{(-x_2 - x_3)}} - \frac{x_8}{1 + e^{(-x_4 - x_6)}} - x_9 \right\} \right]^{-1} \right]^2$$

It is a difficult function to minimize. We obtained by PSO  $f_{\min} = 0.95979$  for  $X=(0.999999, 0.99993, -0.89414, 0.999994, 0.55932, 0.99994, -0.99963, -0.08272)$ . By GA we found the value true 0.00000 and By SA (Simulated Annealing) result came 0.9669081752844. So in this function GA has outperformed PSO and SA.

**5. Beale function:** A 2-variable ( $m=2$ ) function with search domain  $[-4.5 \leq x_i \leq 4.5]$ ; ( $i = 1, 2$ ) given as

$$f(x) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$$

is called the Beale function. The graphical representation of the function given below



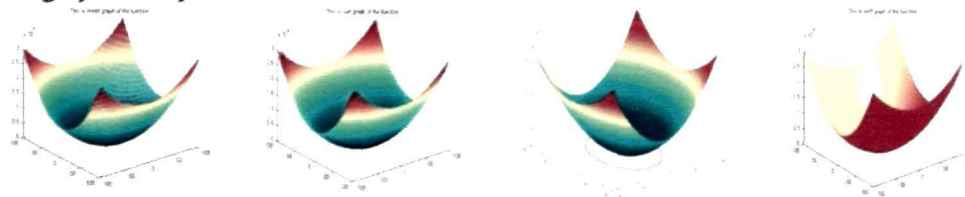
**6. Bohachevsky functions:** Three 2-variable functions ( $m=2$ ) characterizing  $f(x^*) = 0$ ,  $x^* = (0, 0)$  in the search domain  $[-100 \leq x_i \leq 100]$ ;  $i = 1, 2$  are called Bohachevsky functions, which are given as

$$f_1(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) - 0.4 \cos(4\pi x_2) + 0.7$$

$$f_2(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) \cos(4\pi x_2) + 0.3$$

$$f_3(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1 + 4\pi x_2) + 0.3$$

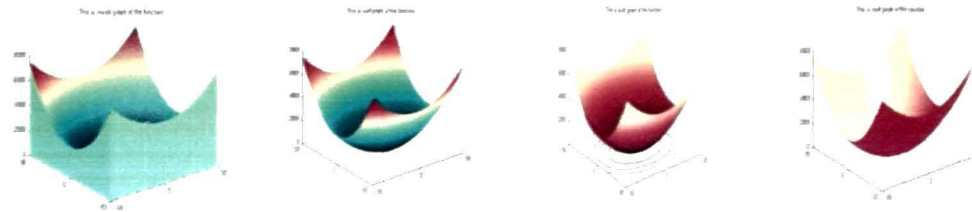
The graphical representation of the function is



**7. Bohachevsky 1:** This function is defined as

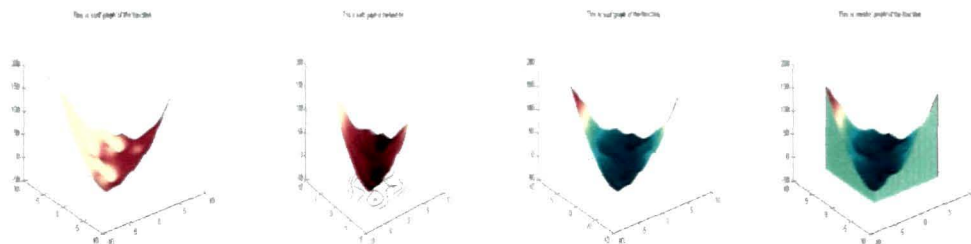
$f(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) - 0.4 \cos(4\pi x_2) + 0.7$  with  $x_1, x_2 \in [-50, 50]$  and the minimum value of the function is  $f^*(x_1, x_2) = 0.0$ .

The function is graphically represented below



**8. Bird function:** This is a bi-modal function with  $f(x^*) = -106.764537$  in the search domain  $x_i \in [-2\pi, 2\pi]; i = 1, 2$  given by

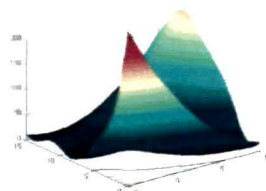
$$f(x) = \sin(x_1)e^{\left[\frac{1-\cos(x_2)}{2}\right]^2} + \cos(x_2)e^{\left[\frac{1-\sin(x_1)}{2}\right]^2} + (x_1 - x_2)^2$$



**9. Branin function:** A 2-variable function ( $m = 2$ ) characterizing  $f(x^*) = 0.397887$ , with three global minima in the search domain  $[-5 \leq x_1 \leq 10; 0 \leq x_2 \leq 15]$  is called Bohachevsky function, which is given as

$$f(x) = (x_2 - 5x_1^2 / (4\pi^2) + (5x_1 / \pi) - 6)^2 + 10(1 - (8\pi)^{-4}) \cos(x_1) + 10$$

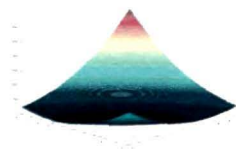
The graphical representation of the function is



**10. Booth Function:** A 2-variable ( $m = 2$ ) function with search domain  $[-10 \leq x_1 \leq 10]; (i = 1, 2)$  given as

$$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

This function is multimodal with the global minimum  $f(x^*) = 0$  at  $x^* = (1, 3)$ . The function is graphically represented below

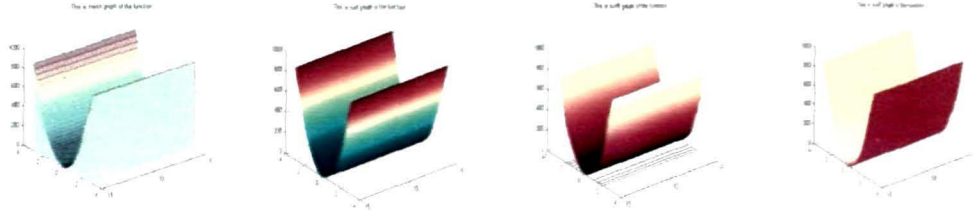


**11. Bukin function:** Bukin function is almost fractal (with fine seesaw edges) in the surroundings of their minimal points. Due to this property, they are extremely difficult to optimize by any method of global (or local) optimization. The search domain  $x_1 \in [-15, -5], x_2 \in [-3, 3]$  these functions are defined as follows in Bukin (1997).

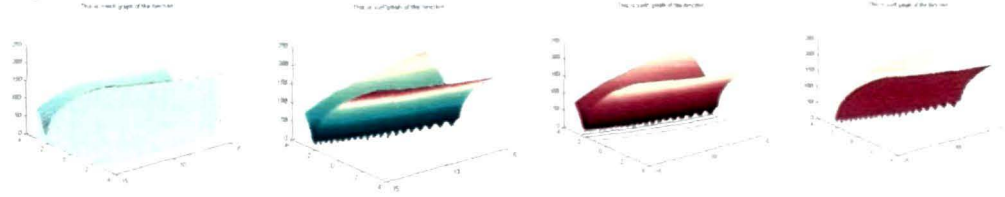
$$f_4 = 100x_2^2 + 0.01|x_1 + 10| \quad \text{and has its minima } f_{\min} = (-10, 1) = 0$$

$$f_6 = 100\sqrt{|x_2 - 0.01x_1^2|} + 0.01|x_1 + 10| \quad \text{and has its minima } f_{\min} = (-10, 1) = 0$$

Graph for 1<sup>st</sup> function:

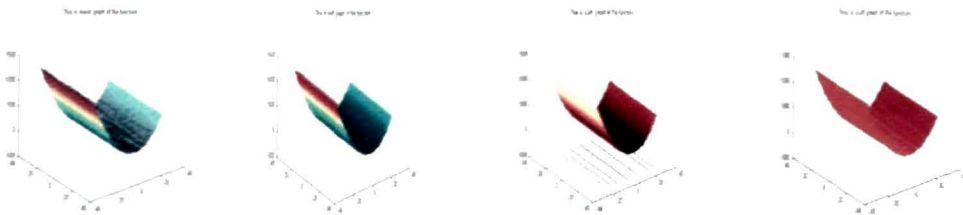


Graph for 2<sup>nd</sup> function:



**12. Chichinadze function:** In the search domain  $x_1, x_2 \in [-30, 30]$  this function is defined as follows and has  $f_{\min}(5.90133, 0.5) = -43.3159$ . Refer Chichinadze (1983).

$$f(x) = x_1^2 - 12x_1 + 11 + 10\cos\left(\frac{\pi x_1}{2}\right) + 8\sin(5\pi x_1) - \left(\frac{1}{5}\right)^{0.5} e^{-0.5(x_2 - 0.5)^2}$$



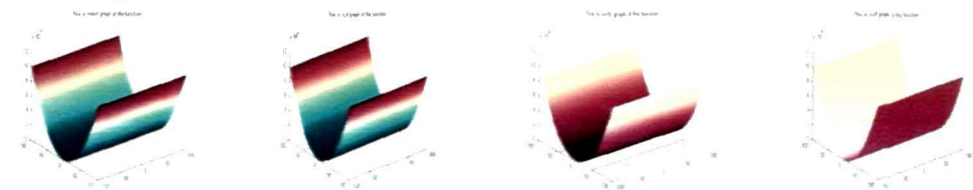
**13. Corana function:** With reference to Corana et al. (1987) four variable function is defined as follows and has  $f_{\min}(0, 0, 0, 0) = 0$ .

$$f(x) = \sum_{i=0}^4 0.15(z_i - 0.05 \operatorname{sgn}(z_i))^2 d_i \text{ if } |x_i - z_i| < 0.05$$

$$z_i = 0.2 \left\lfloor \frac{x_i}{0.2} \right\rfloor + 0.49999 \operatorname{sgn}(x_i)$$

$$= \sum_{i=1}^4 d_i x_i^2 \text{ otherwise}$$

$$d_i = 1, 1000, 10, 100$$





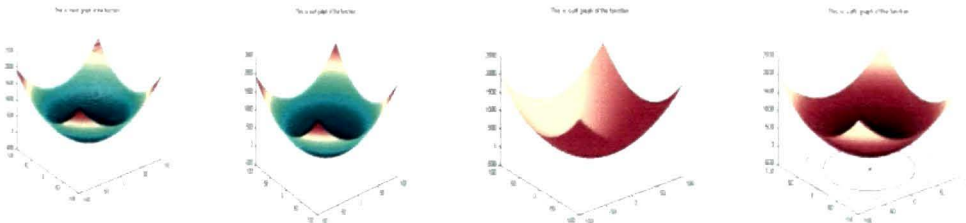
**14. Colville function:** The function is defined as  $f(x_1, x_2, x_3, x_4) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1(x_2 - 1)^2 + (x_4 - 1)^2 + 19.8(x_2 - 1)(x_4 - 1)$  with  $x_1, x_2, x_3, x_4 \in [-10, 10]$   $f^*(x_1, x_2, x_3, x_4) = 0.0$  at  $(1, 1, 1, 1)$ . (Colville, 1986).

The volume curve of the Colville function which is known as isonormal surface is



**15. Deflected Corrugated Spring function:** This function defines the deflected corrugated spring in n dimensions is given by

$$f = -\cos(kR) + 0.1R^2 \quad \text{where } R \text{ is the radius \& } R = \sqrt{(x_1 - c_1)^2 + (x_2 - c_2)^2 + \dots + (x_n - c_n)^2}, \text{ where } \bar{c} \text{ the minimum is point and } k \text{ defines the periodicity nature of the corrugations. For the case } n=2, c_1=2, c_2 = 5, \& k=5.$$



**16. Deb's Deceptive 4-Bit Function:** This is a binary problem, usually used in multiples i.e., a 40-bit version might consist of 10 of these sub problems, either concatenated or interleaved. The problem can be made harder by using different weightings for different sub problems. The equation below gives the fitness of a single block of length 4, as a function of the number of 1's it contains. The maximum value is 1.0 for  $(1, 1, 1, 1)$

$$f(\bar{x}) = \begin{cases} 0.6 - 0.2u(\bar{x}), & u(\bar{x}) < 4 \\ 1, & u(\bar{x}) = 4 \end{cases}$$

Where  $U(\bar{x}) = \sum_{i=1}^4 x_i$ .

**17. DE Jong's Function (#1):** The function is defined as  $f_1 = \sum_{i=1}^3 x_i$ ,  $-5.12 \leq x_i \leq 5.12$  ( $i = 1, 2, 3$ ). It is simple strongly convex function. It is 3-dimensional, continuous, uni-model, separable and scalable. Its global minimum is  $f(0, 0, 0) = 0$ . Refer De-Jong and Morrisson (1999)

**18. DE-Jong's Function (#5):** It is a 2-dimensional, multi-model, continuous, separable with local minima (foxholes)  $\{(a_{1,j}, a_{2,j})\}_{j=1}^9$  and global minima is

$f(-32, -32) \approx 0.998004$  and the function is defined as

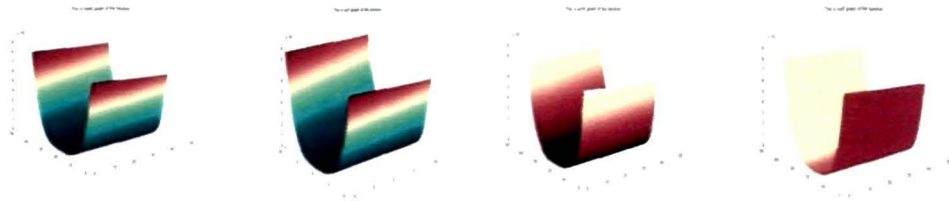
$$f(x) = \frac{1}{0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6}}, -65.536 \leq x_i \leq 65.536 \quad (i=1,2).$$

Here,  $[a_{ij}] = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & \dots & 16 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & \dots & 32 \end{bmatrix}$ .

**19. Dixon & Price function:** This function is in  $m$  ( $m \geq 2$ ) variables with search domain  $[-10 \leq x_i \leq 10]$ ; ( $i=1,2,\dots,m$ ) and the minimum  $f(x^*) = 0$ . It is defined by Dixon and Szego (1975, 1978) as

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^m i(2x_i^2 - x_{i-1})^2$$

The visual presentation of the function is



**20. Easom function:** This function is in 2 variables ( $m=2$ ) with search domain  $[-100 \leq x_i \leq 100]$ ; ( $i=1,2$ ) and  $f(x^*) = -1$  at  $x^* = (\pi, \pi)$ . It is given as

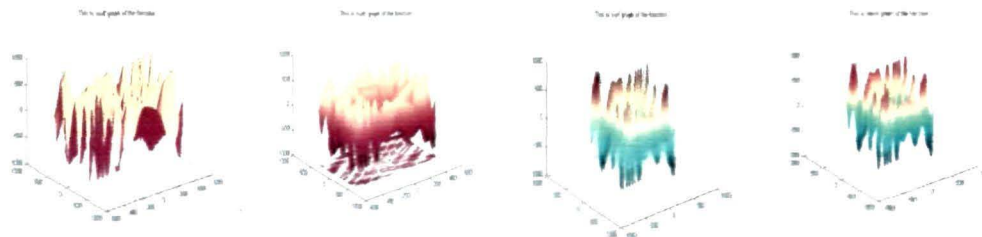
$$f(x) = -\cos(x_1)\cos(x_2)\exp[-(x_1 - \pi)^2 - (x_2 - \pi)^2].$$

The graphical representation of the function is



**21. Egg holder function:** This function is in  $m$  ( $m \geq 2$ ) variables and given as:

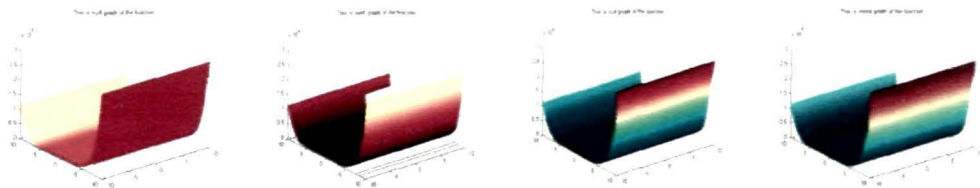
$$f(x) = \sum_{i=1}^{m-1} \left\{ -(x_{i+1} + 47) \sin(\sqrt{|x_{i+1} + x_i/2 + 47|}) + \sin(\sqrt{|x_i - (x_{i+1} + 47)|}) (-x_i) \right\}; -512 \leq x_i \leq 512; i=1,2,\dots,m$$



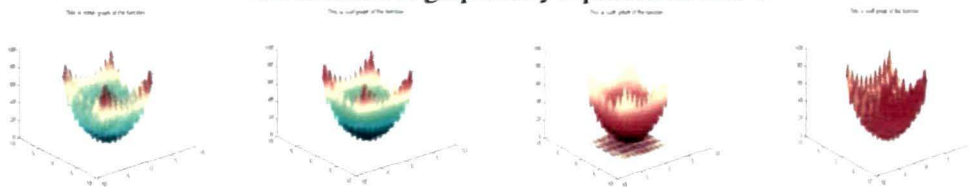
**22. Fletcher- Powell function:** This is a m-variable function with  $f_{\min}(c_1, c_2, c_3, \dots, c_m) = 0$  is given as follows:  $f(x) = \sum_{i=1}^m (A_i - B_i)^2$  where  $A_i = \sum_{j=1}^m [u_{ij} \sin(c_j) + v_{ij} \cos(c_j)]$ ;  $B_i = \sum_{j=1}^m [u_{ij} \sin(x_j) + v_{ij} \cos(x_j)]$ ;  $u_{ij}, v_{ij} = \text{rand}[-100, 100]$  and  $c_i \in [-\pi, \pi]$ .

**23. Freudenstein Roth function:** On  $x_i \in [-10, 10]; i = 1, 2$  this 2-variable function is defined as follows and has  $f_{\min}(5, 4) = 0$ .

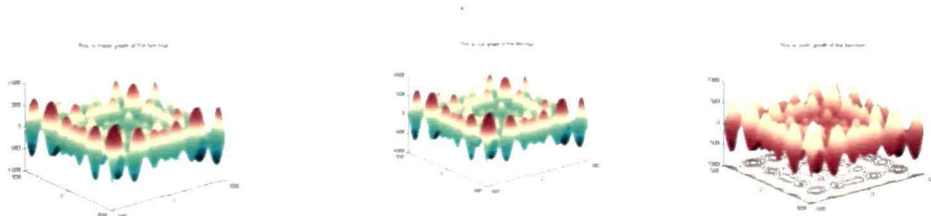
$$f(x) = [-13 + x_1 + ((5 - x_2)x_2 - 2)x_2]^2 + [-29 + x_1 + ((x_2 + 1)x_2 - 14)x_2]^2$$



**24. Generalized Rastrigin function:** A typical multimodal function defined as  $f(x) = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10)$ ,  $-5.12 \leq x_i \leq 5.12$  and the min value of the function is  $f(0, 0, 0, \dots, 0) = 0$ . The function is graphically represented below



**25 Generalized Schwefel function:** The function is defined as  $f(x) = \sum_{i=1}^m -x_i \sin(\sqrt{|x_i|})$ ,  $-500 \leq x_i \leq 500$ . The minimum value of the function best known is  $f(420.9687, \dots, 420.9687) = -4189.5$ . The function is graphically represented below

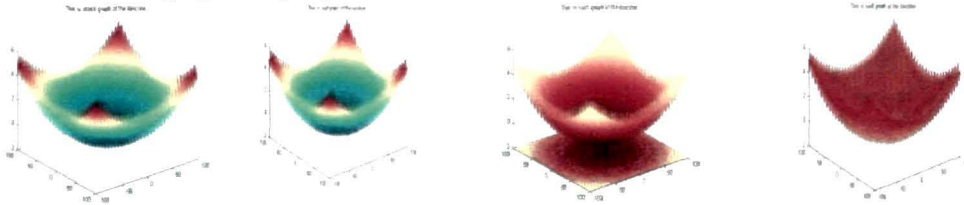


**26. Griewank function:** It is a typical multi-modal function with a large number of local minima in the search domain  $[-600 \leq x_i \leq 600], i = 1, 2, \dots, m$  and global minimum  $f(x^*) = 0$  at  $x^* = (0, 0, \dots, 0)$ . It is given as



$$f(x) = \sum_{i=1}^{100} (x_i^2 / 4000) - \prod_{i=1}^{100} \cos(x_i / \sqrt{i}) + 1$$

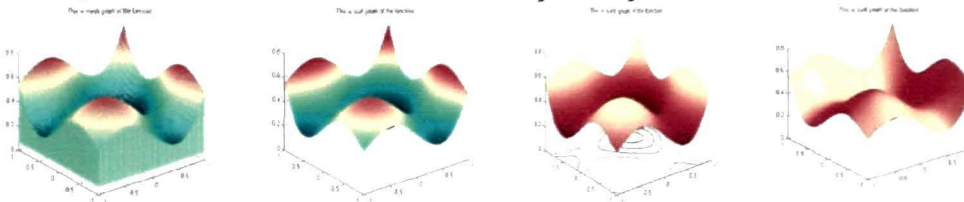
The function is graphically represented below



**27. Giunta function:** In the search domain  $x_1, x_2 \in [-1, 1]$  the function is defined as follows and has  $f_{\min}(0.45834282, 0.45834282) = 0.0602472184$ .

$$f(x) = 0.6 + \sum_{i=1}^2 \left[ \sin\left(\frac{16}{15}x_i - 1\right) + \sin^2\left(\frac{16}{15}x_i - 1\right) + \frac{1}{50} \sin\left(4\left(\frac{16}{15}x_i - 1\right)\right) \right]$$

We have got the minimum value of this function using GA, RPS & SA **0.06447, 0.06447 and 0.6447044840070E-01** respectively.



**28. Goldstein Price function:** On  $x_i \in [-10, 10]; i=1, 2$  this 2-variable function is defined as follows and has  $f_{\min}(0, -1) = 3$ .

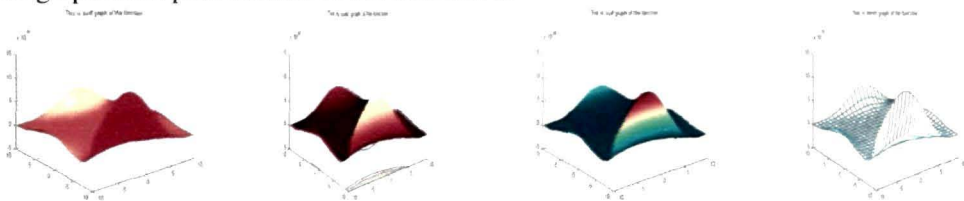
$$f(x) = (f_1)(f_2)$$

where

$$f_1 = \left[ 1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \right]$$

$$f_2 = \left[ 30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2) \right]$$

The graphical representation of the function is



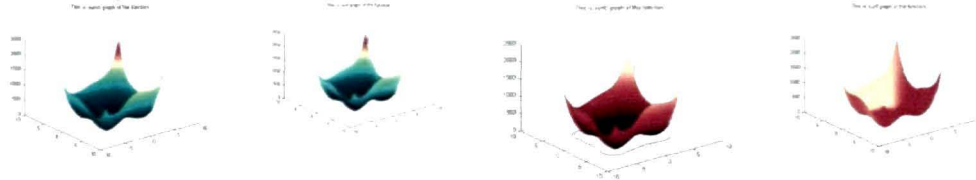
**29. Himmelblau function:** It is a 2-variable ( $m=2$ ) function with search domain  $[-6 \leq x_i \leq 6]; (i=1, 2)$  and 4 global minima  $f(x^*)=0$ , one each in the four Cartesian quadrants. The optimal values of  $x$  are:  $(3, 2)$ ,  $(-2.805, 3.131)$ ,  $(-3.779, -3.283)$  and  $(3.584, -1.848)$ . The function is written as:

$$f(x) = (x_1 + x_2 - 7)^2 + (x_1^2 + x_2 - 11)^2$$

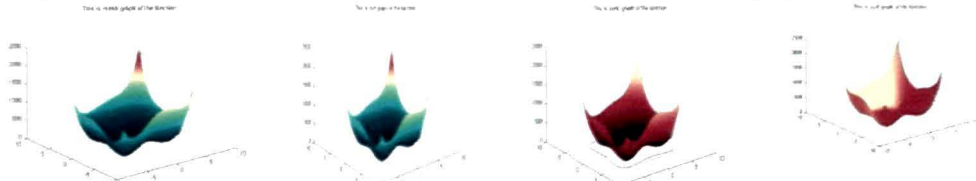
The modified Himmelblau function has only one global optimum  $f(x^*) = 0$  at  $x^* = (3, 2)$ . This (modified) function is given as

$$f(x) = (x_1 + x_2^2 - 7)^2 + (x_1^2 + x_2 - 11)^2 + 0.1[(x_1 - 3)^2 + (x_2 - 2)^2]$$

The visual presentation of the function is



The generalized himmelblau function is depicted below with four graphs



**30. Hougen Function:** Hougen function is typical complex test for classical non-linear regression problems. The Hougen-Watson model for reaction kinetics is an example of this a non-linear regression problem. The form of the model is

$$rate = \frac{\beta_1 x_2 - x_3 / \beta_5}{1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 x_3}$$

where the betas are the unknown parameters,  $x = (x_1, x_2, x_3)$  are the explanatory variables and ‘rate’ is the dependent variable. The parameters are estimated via the least squares criterion. That is, the parameters are such that the sum of the squared differences between the observed responses and their fitted values of rate is minimized.

The input data given alongside are used

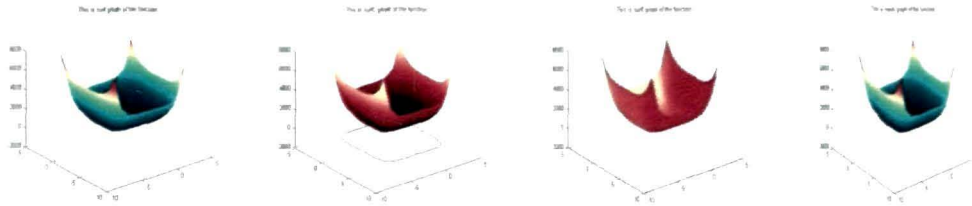
$x_1$	470	285	470	470	470	100	100	470	100	100	100	285	285
$x_2$	300	80	30	80	80	190	80	190	300	300	80	300	190
$x_3$	80	10	10	120	10	10	65	65	54	120	120	10	120
rate	8.55	3.79	4.82	0.02	2.75	14.39	2.54	4.35	13.00	8.50	0.05	11.32	3.13

The values by tradition method are  $\hat{\beta}_1 = 1.253031$ ;  $\hat{\beta}_2 = 1.190943$ ;  $\hat{\beta}_3 = 0.062798$ ;  $\hat{\beta}_4 = 0.040063$ ;  $\hat{\beta}_5 = 0.112453$ . The Particle Swarm method also does not ordinarily perform well in estimating the betas of the Hougen function. However, with  $\eta(a3) = 0.0005$  and  $\omega = 0.05$ , run for 50,000 iterations we obtain:  $\hat{\beta}_1 = 1.5575204$ ;  $\hat{\beta}_2 = 0.0781010629$ ;  $\hat{\beta}_3 = 0.050866667$ ;  $\hat{\beta}_4 = 0.138796292$ ;  $\hat{\beta}_5 = 0.955739322$ . The sum of squares of deviations ( $S^2$ ) is = 0.301933528. A comparison of Rosenbrock-Quasi-Newton results with these (RPS) results indicates that the betas exhibit very high degree of instability in the neighborhood of the minimal  $S^2$ . The above analysis is adopted from Mishra (2006).

**31. Hump function:** It is a 2-variable ( $m = 2$ ) function with search domain  $[-5 \leq x_i \leq 5]$ ; ( $i = 1, 2$ ) and dual (global) minima  $f(x^*) = -1.032$  at  $x^* = (\pm 1) (0.0898, -0.7126)$ . It is given as

$$f(x) = 4x_1^2 - 2.1x_1^4 + x_1^6 / 3 + x_1 x_2 - 4x_2^2 + 4x_2^4$$

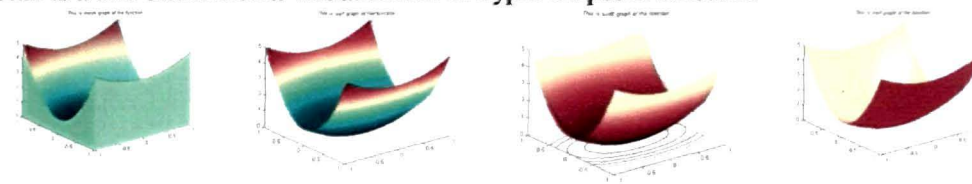
The function is graphically represented below



**32. Hyper ellipsoid function:** The function is defined as  $f(x) = \sum_{j=1}^n j^2 x_j^2$  with

$x_j \in [-1, 1]$  and the minimum value of the function is  $f^*(x) = 0.0$ .

This is a two dimensional visualization of hyper ellipsoid function.

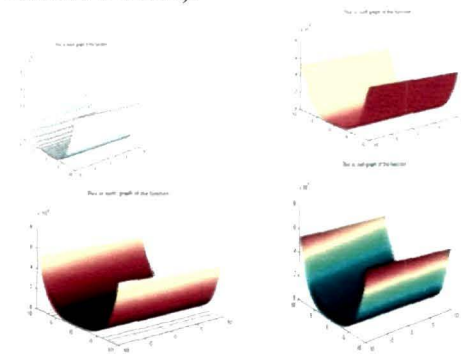


**33. Judge function:** This is a multimodal function defined as

$$f(x) = (x_1 + x_2 \sin^2(u_i) + x_2^2 \cos(v_i) - y_i)^2$$

This function has two optima  $f(0.846, 1.23) = 16.0817$  which is a global minima and  $f(2.35, -0.319) = 20.9805$  which is a local minima.

This function has been taken from Bill Goffe's Simman (Simulated Annealing Problem website).



I	U(I)	V(I)	Y(I)
1	.286	.645	4.284
2	.973	.585	4.149
3	.348	.310	3.877
4	.276	.058	.533
5	.973	.455	2.211
6	.543	.779	2.389
7	.957	.259	2.145
8	.948	.202	3.231
9	.543	.028	1.998
10	.793	.099	1.379
11	.936	.142	2.106
12	.889	.296	1.428
13	.006	.175	1.011
14	.828	.180	2.179
15	.399	.842	2.858
16	.617	.039	1.388
17	.939	.103	1.651
18	.784	.620	1.593
19	.072	.158	1.046
20	.889	.704	2.152

**34. Keane's function:** The keane's function is defined as

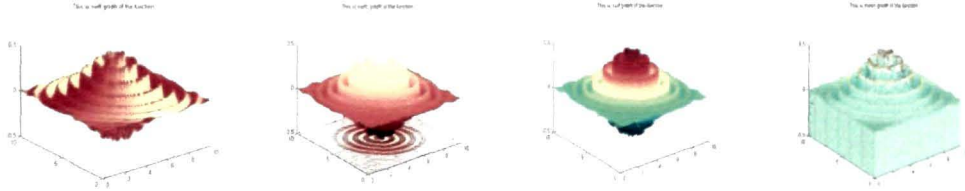
$$f(x) = \left| \frac{\left( \sum_{i=1}^n \cos^4(x_i) - 2 \prod_{i=1}^n \cos^2(x_i) \right)}{\sqrt{\sum_{i=1}^n i x_i^2}} \right|$$

With a constraints

$$\prod_{i=1}^n x_i \geq 0.75$$

$$\sum_{i=1}^n x_i \leq 7.5n$$

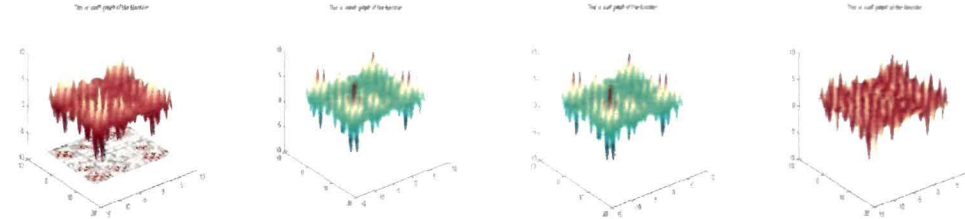
$$0 \leq x_i \leq 10$$



**35. Levy function No. 3:** It is a 2-variable ( $m=2$ ) multi-modal function with search domain  $[-10 \leq x_i \leq 10]$ . It has some 760 local minima and 18 global minima in this search domain. Its global minimum is  $f(x^*) = -176.542$ .

$$f(x) = \sum_{i=1}^5 i \cos[(i+1)x_1 + i] \sum_{i=1}^5 i \cos[(i+1)x_2 + i]$$

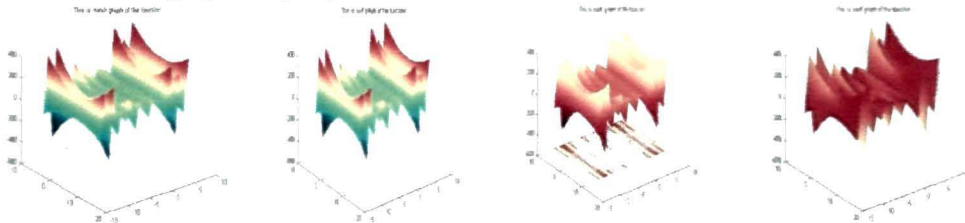
The function is graphically represented below



**36. Levy function No. 5:** It is a 2-variable ( $m=2$ ) multi-modal function with search domain  $[-10 \leq x_i \leq 10]$ . It has some 760 local minima in this search domain. Its global minimum is  $f(x^*) = -176.1375$  at  $x^* = (-1.3068, -1.4248)$ .

$$f(x) = \sum_{i=1}^5 i \cos[(i+1)x_1 + i] \sum_{i=1}^5 i \cos[(i+1)x_2 + i] + (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2$$

The function is graphically represented below



**37. Levy function No. 8:** It is a 3-variable ( $m=3$ ) multi-modal function with search domain  $[-10 \leq x_i \leq 10]$ . It has some 125 local minima in this search domain. Its only global minimum is  $f(x^*) = 0$  at  $x^* = (1, 1, 1)$ . this function is specified as

$$f(x) = \sin^2(\pi y_1) + \sum_{i=1}^{m-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_m - 1)^2$$

where  $y_i = 1 + (x_i - 1)/4$ ;  $i = 1, \dots, m$ .

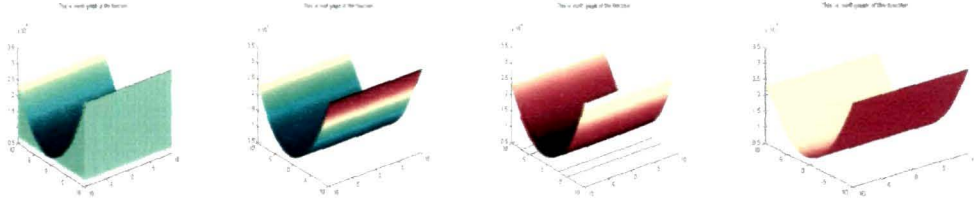
in Levy et al. (1981).



**38. Levy function (#13):** In the search domain  $x_1, x_2 \in [-10, 10]$  this function is defined as follows and has  $f_{\min}(1, 1) = 0$ . as defined in Levy et al. (1981).

$$f(x) = \sin^2(3\pi x_1) + (x_1 - 1)^2 [1 + \sin^2(3\pi x_2)] + (x_2 - 1)^2 [1 + \sin^2(2\pi x_2)].$$

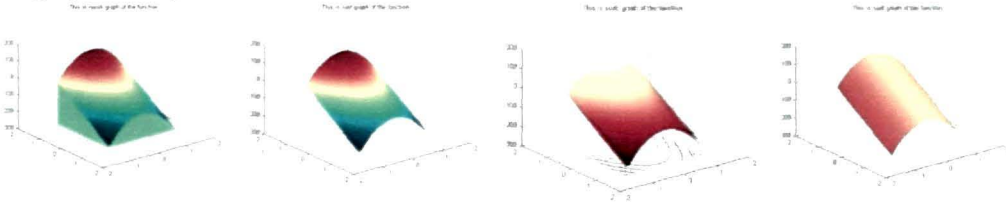
The graphical representation of the function is



**39. Leon function:** In the search domain  $x_1, x_2 \in [-1.2, 1.2]$  this function is defined as follows and has  $f_{\min}(1, 1) = 0$ .

$$f(x) = 100(x_2 - x_1^2) + (1 - x_1)^2$$

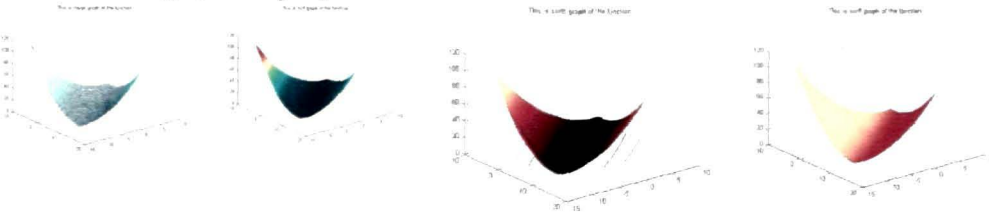
The graphical representation of the function is



**40. Matyas function:** It is a 2-variable ( $m=2$ ) function with search domain  $[-10 \leq x_i \leq 10]; (i=1, 2)$  and minimum  $f(x^*) = 0$  at  $x^* = (0, 0)$ . It is given as

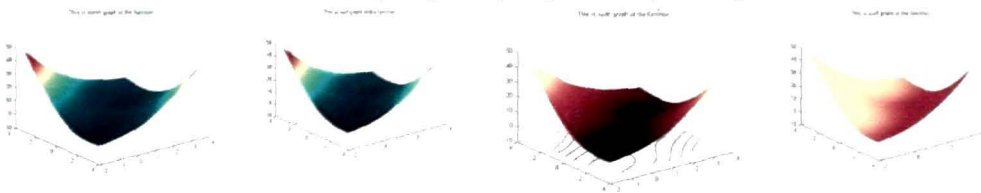
$$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$$

The function is graphically represented below



**41. McCormick function:** In the search domain  $x_1 \in [-1.5, 4], x_2 \in [-3, 4]$  this function is defined as follows and has  $f_{\min}(-0.54719, -1.54719) = -1.9133$ .

$$f(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1.$$



**42. Meyer function:** The Meyer function is defined as cubic polynomial divided by a quadratic

$$f(y) = \frac{1 + Ay + By^2 + Cy^3}{1 + Dy + Ey^2} \quad \text{Where}$$

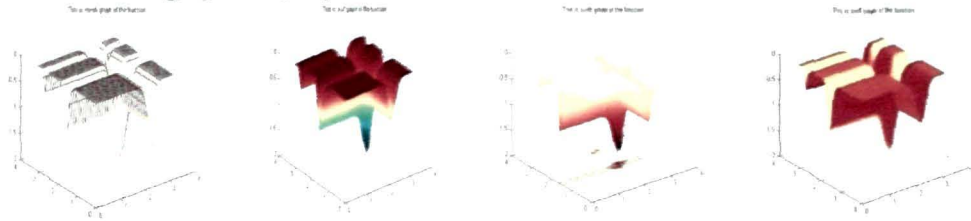
A	B	C	D	E
1.3604	0.0962	-0.5127	-0.6722	-0.3278

$$\text{and } y = \left( \frac{v}{v_\infty} \right)^{2/3} \quad v_\infty = \frac{\pi}{6} (\sqrt{6} - 1).$$

**43. Michalewicz function:** In Michalewicz (1985) the function is an interesting multi-modal function in the search domain  $[0 \leq x_i \leq \pi]$ ,  $i=1,2,\dots,m$ . It has an additional parameter, p that determines its surface. For p=10, its global minima at different dimensions (m) are :  $f(x^*) = -1.8013$  (while m=2),  $f(x^*) = -4.6877$  (while m=5),  $f(x^*) = -7.664$  (while m=7). This function is given as

$$f(x) = -\sum_{i=1}^m \sin(x_i) (\sin(ix_i^2 / \pi))^{2p}$$

The function is graphically represented below



**44. Modified RCOS function:** In the domain  $x_1 \in [-5,10]$ ,  $x_2 \in [0,15]$  this 2-variable function has  $f_{\min}(-3.196989, 12.52626) = -0.179891$ . It is specified as where  $f_1 = a(x_2 - bx_1^2 + cx_1 - d)^2$ ;  $f_2 = e(1 - g) \cos(x_1) \cos(x_2)$ ;  $f_3 = \log(x_1^2 + x_2^2 + 1)$

$$\text{where } g = \frac{1}{8\pi}; b = \frac{5.1}{4\pi^2}; c = \frac{5}{\pi}; a = 1; d = 6; e = 10$$

**45. Modified Schaffer function #1:** In the search domain  $x_1, x_2 \in [-100,100]$  this function is defined as follows and has  $f_{\min}(0,0) = 0$

$$f(x) = 0.5 + \frac{\sin^2(x_1^2 + x_2^2) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$$

**46. Modified Schaffer function #2:** In the search domain  $x_1, x_2 \in [-100,100]$  this function is defined as follows and has  $f_{\min}(0,0) = 0$

$$f(x) = 0.5 + \frac{\sin^2(x_1^2 - x_2^2) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$$

**47. Modified Schaffer function #3:** In the search domain  $x_1, x_2 \in [-100,100]$  this function is defined as follows and has  $f_{\min}(0, 1.253115) = 0.00156685$

$$f(x) = 0.5 + \frac{\sin^2 \cos |x_1^2 - x_2^2| - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$$

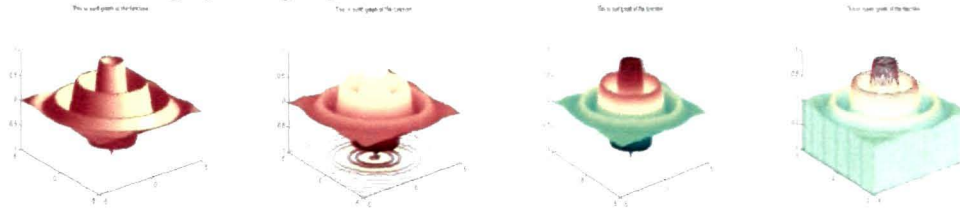
**48. Modified Schaffer function #4:** In the search domain  $x_1, x_2 \in [-100, 100]$  this function is defined as follows and has  $f_{\min}(0, 1.253132) = 0.292579$

$$f(x) = 0.5 + \frac{\cos^2 \sin |x_1^2 - x_2^2| - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$$

**49. Masters Cosine wave function:** The function is defined as

$$f(x) = -\sum_{i=1}^{n-1} e^{-\frac{1}{8}(\sqrt{x_{i+1}^2 + 0.5x_i x_{i+1}} + x_i^2)} \cos\left(4\sqrt{x_{i+1}^2 + 0.5x_i x_{i+1}} + x_i^2\right), -5 \leq x_i \leq 5.$$

The function is graphically represented below



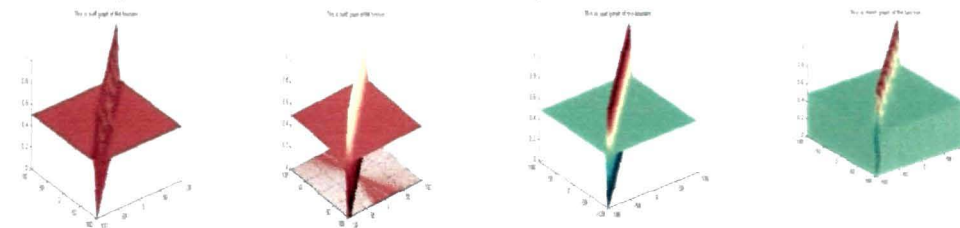
**50. Needle-eye function :** This function is m-dimensional ( $m \geq 1$ ) and defined with a small (say 0.0001) eye. If  $|x_i| \leq eye \forall i$  then  $f(x) = 1$  else

$$f(x) = \sum_{i=1}^m (1.00 + |x_i|)^{t_i}; t_i = 1 \text{ if } |x_i| > eye, 0 \text{ otherwise. Minimization of this problem}$$

becomes more difficult with smaller eye and larger m(dimension).

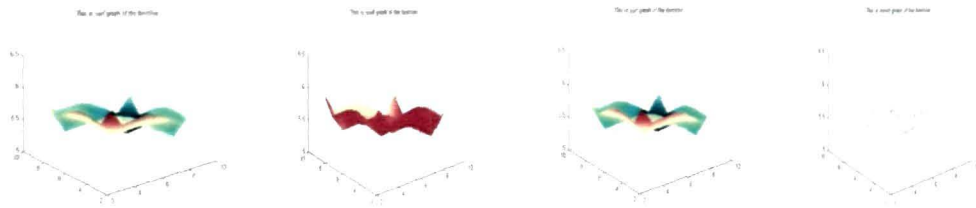
**51. Pathological function:** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} \left( \frac{\sin^2 \left( \sqrt{x_{i+1}^2 + 100x_i^2} \right) - 0.5}{0.001 \left( x_{i+1}^2 - 2x_{i+1}x_i + x_i^2 \right)^2 + 1.0} + 0.5 \right), -100 \leq x_i \leq 100.$$



**52. Paviani function:** It is a 10-variable function ( $m=10$ ) in the search domain  $x_i \in (2, 10)$ , with  $f(x^*) = -45.77847; x^* = (9.3502, 9.3502, \dots, 9.3502)$  given as

$$\sum_{i=1}^{10} [\ln^2(x_i - 2) + \ln^2(10 - x_i)] - \left[ \prod_{i=1}^{10} x_i \right]^{0.2}$$



**53. Perm function #1 :** In the domain  $x \in [-4, 4]$ , the function has  $f_{\min} = 0$  for  $x = (1, 2, 3, 4)$ .

It is specified as  $f(x) = \left[ \sum_{k=1}^4 \sum_{i=1}^4 (i^k + \beta) \left\{ \left( \frac{x_i}{i} \right)^k - 1 \right\} \right]^2$  The value of  $\beta (= 50)$  introduces difficulty to optimization. Smaller values of  $\beta$  raises difficulty further. The graphical representation of the function is



**54. Perm function #2:** In the domain  $x \in [-1, 1]$ , and for a given  $\beta (= 10)$ , this m-variable function has  $f_{\min} = 0$  for  $x_i = (i)^{-1}, i = 1, 2, \dots, m$ . It is specified as

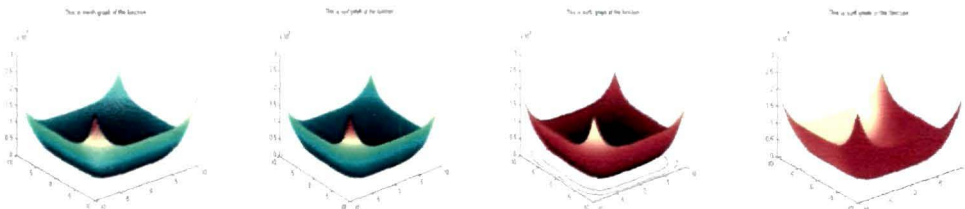
$$f(x) = \left[ \sum_{k=1}^4 \sum_{i=1}^4 (i + \beta) \{ (x_i)^k - (i)^{-k} \} \right]^2$$

**55. Power-sum function:** Defined on four variables in the domain  $x \in [0, 4]$ , this function has  $f_{\min} = 0$  for any permutation of  $x = (1, 2, 2, 3)$ . The function is defined as

$$f(x) = \left[ \sum_{k=1}^4 b_k - \sum_{i=k}^4 x_i^k \right]^2; b_k = (8, 18, 44, 114) \text{ for } k = (1, 2, 3, 4) \text{ respectively.}$$

**56. Quintic function:** In the domain  $x \in [-10, 10]$  with  $f_{\min} = 0$  for  $x_i = -1$  or  $2; i = 1, 2, \dots, m$  this function (with multiple global minima) is defined as

$$f(x) = \sum_{i=1}^m |x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4|; x_i \in [-10, 10]; i = 1, 2, 3, \dots, m$$



**57. Quadric Function:** This function is defined as

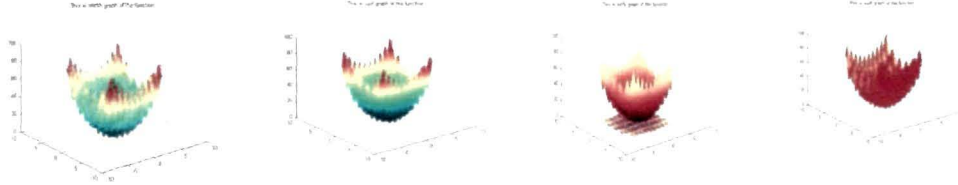
$$f(x) = \sum_{j=1}^n \left( \sum_{k=1}^j x_j \right)^2 \text{ with } x_j \in [-100, 100] \text{ and the minimum value is } f^*(x) = 0.0$$



**58. Rastrigin function:** It is a typical multi-modal function in  $m$  ( $m \geq 1$ ) variables with search domain  $[-5.12 \leq x_i \leq 5.12]$ ; ( $i=1,2,\dots,m$ ) and the minimum  $f(x^*) = 0$  at  $x^* = (0,0,\dots,0)$ . It is a difficult function to optimize. It is given as

$$f(x) = 10m + \sum_{i=1}^m (x_i^2 - 10 \cos(2\pi x_i))$$

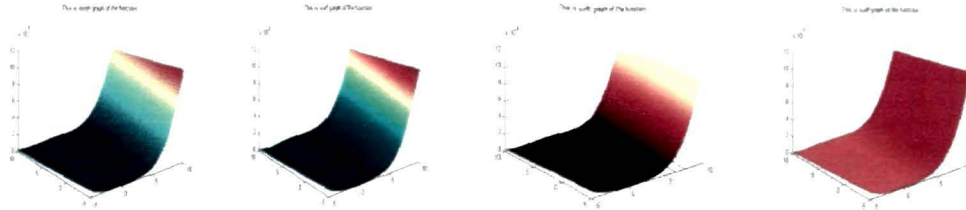
The function is graphically represented below



**59. Rosenbrock function:** This function is in  $m$  ( $m \geq 2$ ) variables with search domain  $[-5 \leq x_i \leq 10]$ ; ( $i=1,2,\dots,m$ ) and the minimum  $f(x^*) = 0$  and  $x^* = (1, 1,\dots,1)$ . It is very similar to the Dixon and Price function. It is often referred to for its very slow convergence in the neighborhood of the minimum. It is given as

$$\sum_{i=1}^{m-1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$$

The function is graphically represented below in two variables



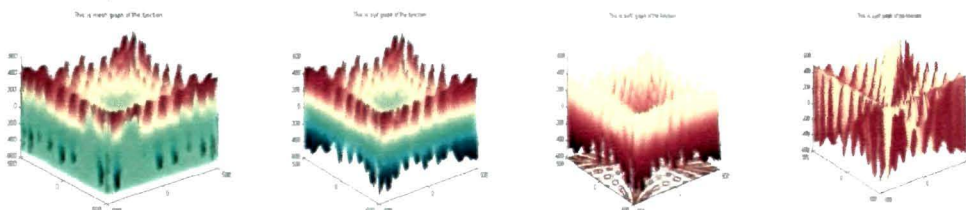
**60. Royal Road Function:** The Royal Road function is a binary problem with one optimum but many large plateaus. Technically, this is a type-R1 Royal road function. This version assumes there are  $a$  blocks, each of  $b$  bits, so that  $L = a \cdot b$ .

$$f(\bar{x}) = \sum_{i=1}^a \prod_{j=i \cdot b + 1}^{i \cdot (b+1)} x_j$$

**61. Rana's function :** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} \left( (x_{i+1} + 1) \cos(\sqrt{|x_{i+1} - x_i + 1|}) \sin(\sqrt{|x_{i+1} + x_i + 1|}) + x_i \cos(\sqrt{|x_{i+1} + x_i + 1|}) \sin(\sqrt{|x_{i+1} - x_i + 1|}) \right)$$

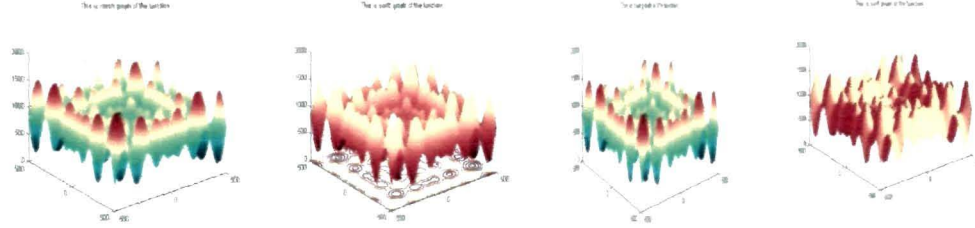
,  $-500 \leq x_i \leq 500$ .



**62. Schwefel function:** It is another difficult but interesting multi-modal multi-dimensional function in the search domain  $[-500 \leq x_i \leq 500]$ ,  $i=1,2,\dots,m$  with its global minimum at  $f(x^*)=0$ . It is given as defined in Schwefel (1981, 1995) as,

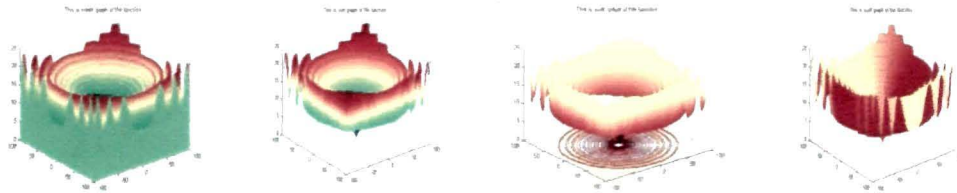
$$f(x) = 418.9829m - \sum_{i=1}^m [x_i \sin(\sqrt{|x_i|})]$$

The function is graphically represented below



**63. Schaffer Function (#7) :** This function is non-separable and non-scalable. The global minimum value of the function is  $f(0,0)=0$ .

$$f(x) = (x_1^2 + x_2^2)^{0.25} [\sin^2(50(x_1^2 + x_2^2)^{0.1}) + 1.0] \quad -100 \leq x_i \leq 100 (i=1,2).$$



**64. Schaffer function (#5):** In the search domain  $x_1, x_2 \in [-100,100]$  this function is defined as follows and has  $f_{\min}(0,0)=0$ .

$$f(x) = 0.5 + \frac{\sin^2 \sqrt{(x_1^2 + x_2^2)} - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$$

**65. Shekel Function:** A 4-variable function ( $m=4$ ) for parameter  $p$  ( $2 \leq p \leq 10$ ;  $p$  is an integer) in the search domain  $x_i \in (2, 10)$  is given as:

$$f_p(x) = -\sum_{i=1}^p \left( \sum_{j=1}^4 (x_j - a_{ij})^2 + c_i \right)^{-1}$$

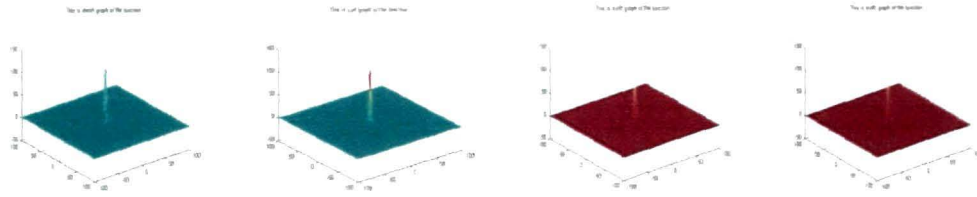
The optimal values of  $f_p(x^*)$  lie between -10.02 and 10.54

The matrix  $A$  and the vector  $C$  are given alongside.

$$A = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{bmatrix}; \quad C = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.5 \end{bmatrix}$$

**66. Sine envelope sine wave function:** This function characterizes repeating couplets of optimal values of  $x^*$ , except their sign. The function is given as:

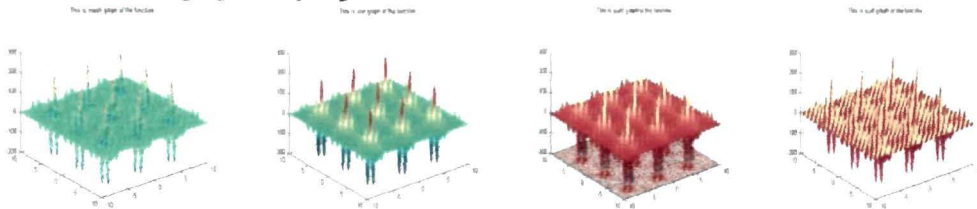
$$f(x) = \sum_{i=1}^{m-1} \left( \frac{\sin^2(\sqrt{x_{i+1}^2 + x_i^2} - 0.5)}{(0.001(x_{i+1}^2 + x_i^2) + 1)^2} + 0.5 \right); \quad -100 \leq x_i \leq 100; \quad i=1,2,\dots,m.$$



**67. Shubert function:** It is a 2-variable ( $m=2$ ) typically difficult multi-modal function with search domain  $[-10 \leq x_i \leq 10]$ ; ( $i=1,2$ ) and minimum  $f(x^*) = -186.7309$ . It is given as

$$f(x) = \prod_{j=1}^2 \sum_{i=1}^5 [i \cos((i+1)x_j + i)]$$

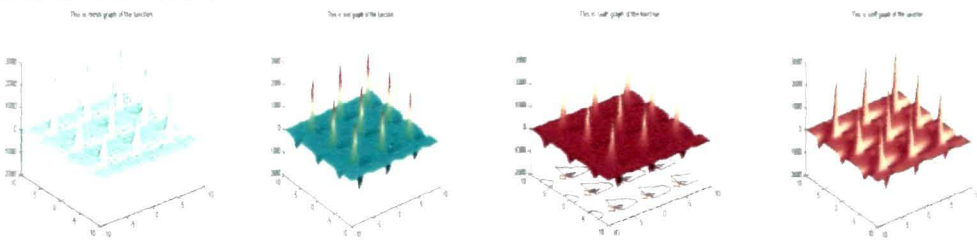
The function is graphically represented below



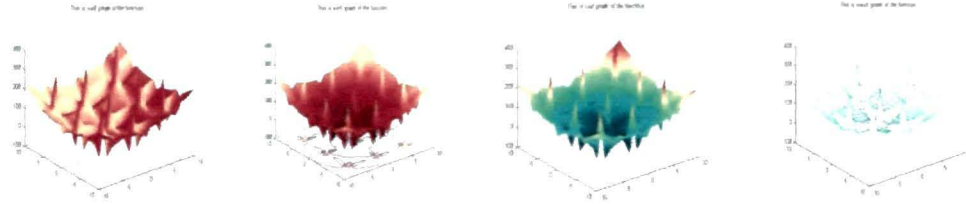
**68. Shubert function #1:** This function is defined as  $f(x) = \left( \sum_{i=1}^5 i \cos[(i+1)x_1 + i] \right) * \left( \sum_{i=1}^5 i \cos[(i+1)x_2 + i] \right)^2$  and the global minimum is at  $X^* = X^* (:i)$ ,

$$X^* = \begin{bmatrix} -7.0835 & -7.0835 & -1.4251 & -1.4251 & -1.4251 & 4.8581 & 5.4829 \\ -1.4251 & -7.7083 & 5.4829 & -7.0835 & -0.8003 & -0.8003 & 4.8581 \end{bmatrix}$$

Where  $X^*$  is the part of the global minima set and  $f(x^*) = -186.7309$ . This function has about 400 minima and about 18 global minima. The graph of the function 21 and 22 is same as function 20.

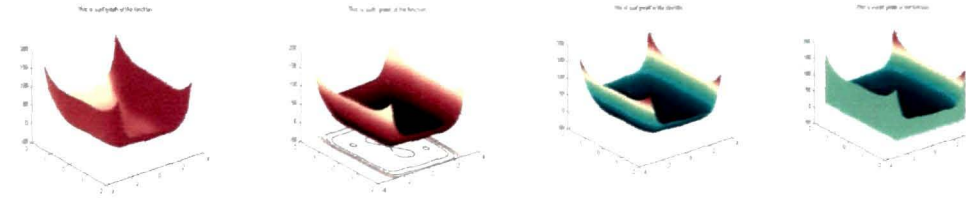


**69. Shubert function #2:** The function is defined  $f(x) = \left( \sum_{i=1}^5 i \cos[(i+1)x_1 + i] \right) * \left( \sum_{i=1}^5 i \cos[(i+1)x_2 + i] \right) + (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2$ . Where the bounds are  $-10 \leq x_1, x_2 \leq 10$ . The global minimum is at  $x^* = (-1.42513, -0.80032)$ ,  $f(x^*) = -186.7309$ . This function has about 400 minima. The function is graphically represented below



**70. The six-hump camel back function:** The camel function is defined as

$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{x_1^6}{3} + x_1x_2 - 4x_2^2 + 4x_2^4$ ,  $-3 \leq x_1 \leq 3$  and  $-2 \leq x_2 \leq 2$ . The global minimum value of the function is at  $x^* = (.0898, -0.7127)$  or  $(.0898, 0.7127)$  and  $f(x^*) = -1.0316$ .



**71. The Stuckman function:** This function is defined as

$$f_8(x_1, x_2) = \begin{cases} \left[ \left( \lfloor m_1 \rfloor + \frac{1}{2} \right) \sin(a_1) / a_1 \right] & \text{if } 0 \leq x_1 \leq b \\ \left[ \left( \lfloor m_2 \rfloor + \frac{1}{2} \right) \sin(a_2) / a_2 \right] & \text{if } b \leq x_2 \leq 10 \end{cases}$$

Where  $0 \leq x_i \leq 10$  for  $i=1,2$  and  $m_i$  is a random variable between 0 and 100 ( $i=1,2$ ), and  $b$  is a random variable between 0 and 10, and  $a_i = \lfloor |x_1 - r_{1i}| \rfloor + \lfloor |x_2 - r_{2i}| \rfloor$ , where  $r_{11}$  is a random variable between 0 and  $b$ ,  $r_{12}$  is a random variable between  $b$  and 10, and  $r_{21}$  is a random variable between 0 and 10,  $r_{22}$  is a random variable between 0 and 10 (all random variables are uniform).

The global maximum is located at

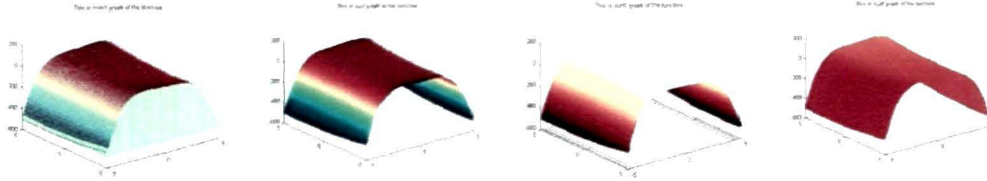
$$(x_1, x_2) = \begin{cases} (r_{11}, r_{21}) & \text{if } m_1 \geq m_2 \\ (r_{12}, r_{22}) & \text{otherwise} \end{cases}$$

**72. Three-humps camel back function:** In the search domain  $x_1, x_2 \in [-5, 5]$  this function is defined as follows and has  $f_{min}(0,0) = 0$ .

$$f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$$

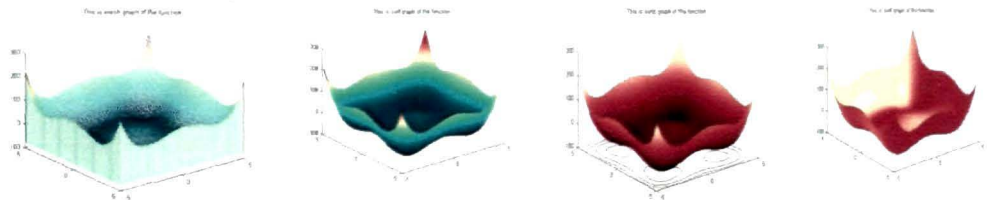
The graphical representation of the function is





**73. Styblinski-tang function:** In the search domain  $x_1, x_2 \in [-5, 5]$  this function is defined as follows and has  $f_{\min}(-2.903534, -2.903534) = -78.332$

$$f(x) = \frac{1}{2} \sum_{i=1}^2 (x_i^4 - 16x_i^2 + 5x_i). \text{ By Styblinski and Tang (1990).}$$

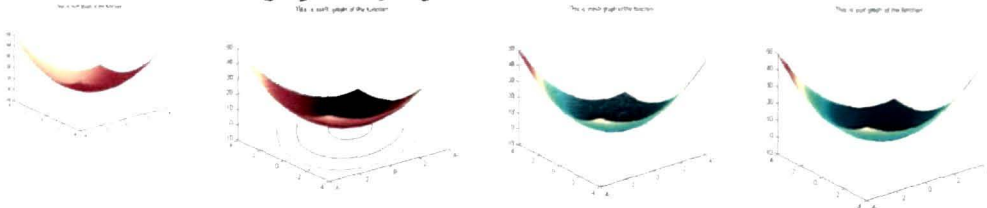


**74. Trid function:** This function is in  $m$  ( $m \geq 2$ ) variables with search domain  $[-m^2 \leq x_i \leq m^2]; (i=1, 2, \dots, m)$ . The Trid function is given as

$$f(x) = \sum_{i=1}^m (x_i - 1)^2 - \sum_{i=2}^m x_i x_{i-1}$$

m	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$	$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$	$x_{15}$	$f(x^*)$
15	15	28	39	48	55	60	63	64	63	60	55	48	39	28	15	-665
10	10	18	24	28	30	30	28	24	18	10						-210
6	6	10	12	12	10	6										-50

The values of  $f(x^*)$  and those of  $x^*$  at different  $m$  are given in the table above. The pattern observed in the values taken on by decision variables is interesting, Mishra (2006). The function is graphically represented below



**75. Weierstrass function:** The Weierstrass function [in its original form,  $f(x) = \sum_{k=0}^{\infty} a^k \cos(b^k x)$  while  $b$  is an odd integer,  $0 < a < 1$ ;  $ab > (1 + 3\pi/2)$ ] is one of the most notorious functions (with almost fractal surface) that changed the course of history of mathematics. Weierstrass proved that this function is throughout continuous but nowhere differentiable. In its altered form Liang and Suganthan (2005) this function in  $m$  ( $m \geq 1$ ) variables with search domain  $[-0.5 \leq x_i \leq 0.5]; (i=1, 2, \dots, m)$  and the minimum  $f(x^*) = 0$  for  $x^* = (0, 0, \dots, 0); a = 0.5; b = 3; k = 20$ , is given as

$$f(x) = \sum_{i=1}^m \sum_{k=0}^k [a^k \cos(2\pi b^k (x_i + 0.5))] - m \sum_{k=0}^k [a^k \cos(2\pi b^k 0.5)]; \quad x_i \in [-0.5, 0.5]; \quad i = 1, 2, \dots, m$$

**76. Yao-Liu #15 function or Kowalick:** It is 4 variable functions in the domain  $x \in [-5.5]$ , that has a global minimum  $f_{\min}(0.19, 0.19, 0.12, 0.14) = 0.3075$ . This

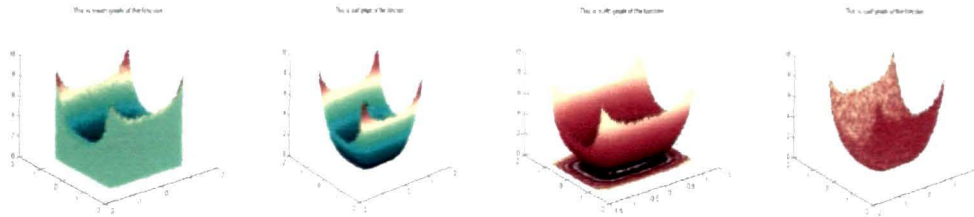
function is defined by Yao et. al. (1999) as  $f(x) = 1000 \sum_{i=1}^{11} \left[ a_i - \frac{x_i (b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$ ; **where**

$$\left( b = \frac{1}{0.25}, \frac{1}{0.5}, \frac{1}{1}, \frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16} \right) \quad \text{and}$$

$$a = (0.1957, 0.1947, 0.1735, 0.1600, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.0235, 0.0246)$$

**77. Yao-Liu #7 function:** It is a m-variable function in the domain  $x \in [-1.28, 1.28]$ , that has a global minima  $f_{\min}(0, 0, 0, 0, \dots, 0) = 0$ . This function is given by Yao et al. (1999) as

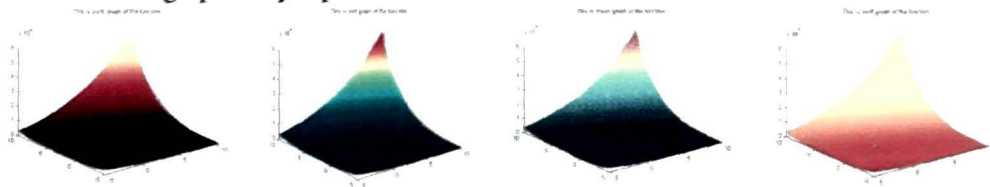
$$f(x) = \text{rand}[0, 1] + \sum_{i=1}^m i(x_i^4)$$



**78. Zakharov function:** This function is in m ( $m \geq 2$ ) variables with search domain  $[-5 \leq x_i \leq 10]; (i = 1, 2, \dots, m)$  and the minimum  $f(x^*) = 0$  and  $x^* = (0, 0, \dots, 0)$ . The function is:

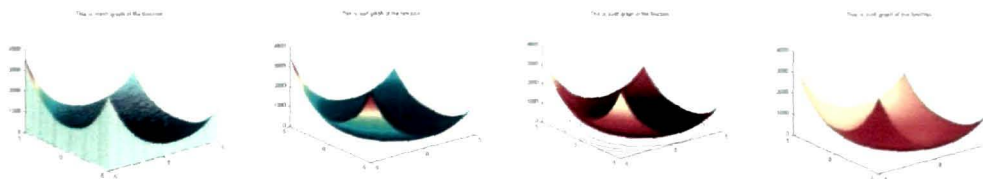
$$f(x) = \sum_{i=1}^m x_i^2 + \left[ \sum_{i=1}^m ix, /2 \right]^2 + \left[ \sum_{i=1}^m ix, /2 \right]^4$$

The function is graphically represented below

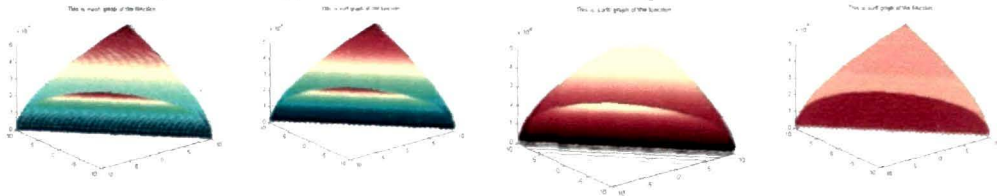


**79. Zettle function:** In the search domain  $x_1, x_2 \in [-5, 5]$  this function is defined as follows and has  $f_{\min}(-0.0299, 0) = -0.003791$

$$f(x) = (x_1^2 + x_2^2 - 2x_1)^2 + 0.25x_1$$



**80. Zero-sum Function:** Defined in the domain  $x \in [-10,10]$  this function (in  $m \geq 2$ ) has  $f(x)=0$  if  $\sum_{i=1}^m x_i = 0$ . Otherwise  $f(x) = 1 + \left(10000 \left| \sum_{i=1}^m x_i \right| \right)^{0.5}$ . This function has innumerable many minima but it is extremely difficult to obtain any of them. Larger is the value of  $m$  (dimension), it becomes more difficult to optimize the function.

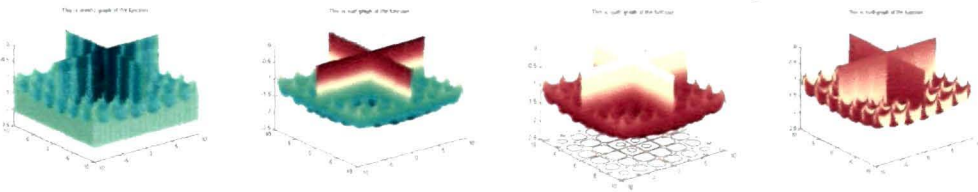


**(ii) Recently appeared test functions**

These functions are credited to Mishra (2006) and the graph of these functions is regenerated by us.

**81. Cross in tray function:** This function has a multiple local minima with four global minima at  $f(x^*) = 2.06261218$  in the search domain  $x_i \in [-10,10], i = 1,2$ . This function is given as:

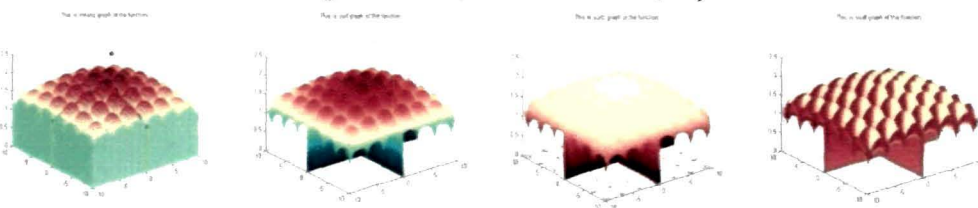
$$f(x) = -0.0001 \left\{ \sin(x_1) \sin(x_2) e^{\left| 100 - \left[ (x_1^2 + x_2^2)^{0.5} \right] / \pi \right|} + 1 \right\}^{0.1}$$



**82. Crowned cross function :** This is a negative form of the cross in tray function. It has

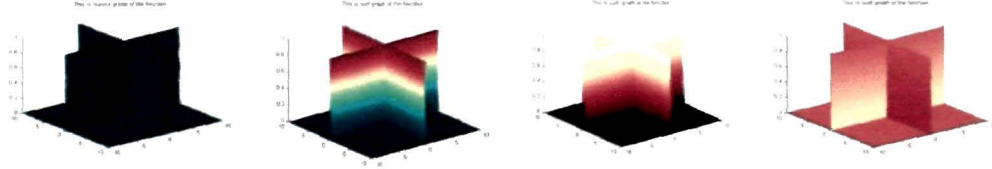
$f(x^*) = 0$  In the search domain  $x_i \in [-10,10], i = 1,2$ . It is a difficult function to optimize. The minimal value obtained by us is approximately 0.1 by Repulsive Particle Swarm

$$f(x) = 0.0001 \left\{ \sin(x_1) \sin(x_2) e^{\left| 100 - \left[ (x_1^2 + x_2^2)^{0.5} \right] / \pi \right|} + 1 \right\}^{0.1}$$



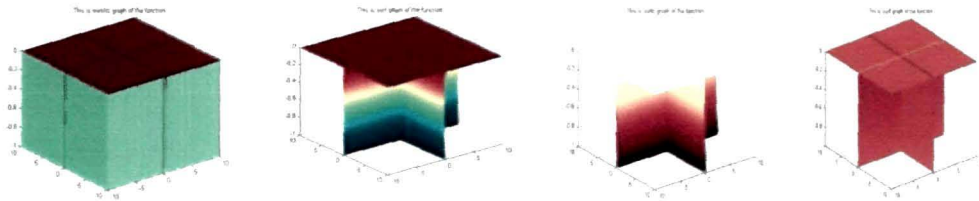
**83. Cross function:** This is a multi-modal function with  $f(x^*)=0$ . It is given by

$$f(x) = \left\{ \sin(x_1) \sin(x_2) e^{\left| 100 - \left[ (x_1^2 + x_2^2)^{0.5} \right] / \pi \right|} + 1 \right\}^{-0.1}$$



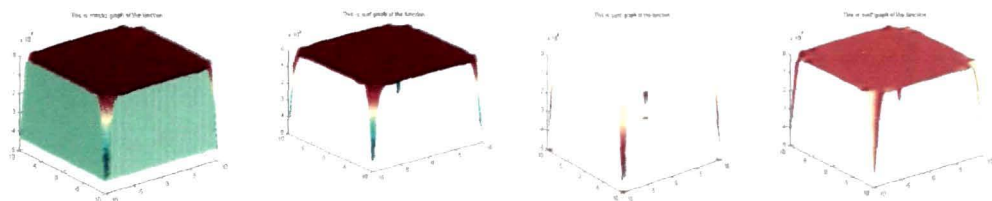
**84. Cross-leg table function:** This function is the negative form of the cross function and may be called as ‘inverted cross’ function. It has  $f(x^*)=-1$ . in the search domain  $x_i \in [-10,10], i=1,2$ . It is difficult to optimize. We have failed to optimize by all our methods which we have used in this thesis. Repulsive Particle Swarm Method as given us the result 0.001305, Genetic Algorithm has given us 0.00000 and Simulated Annealing has given us  $-0.8470640834163E-04$ .

$$f(x) = - \left\{ \sin(x_1) \sin(x_2) e^{\left| 100 - \left[ (x_1^2 + x_2^2)^{0.5} \right] / \pi \right|} + 1 \right\}^{-0.1}$$



**85. Carrom Table Function:** This function has a multiple local minima with four global minima at  $f(x^*) = 24.1568155$  in the search domain  $x_i \in [-10,10], i=1,2$ . This function is given as :

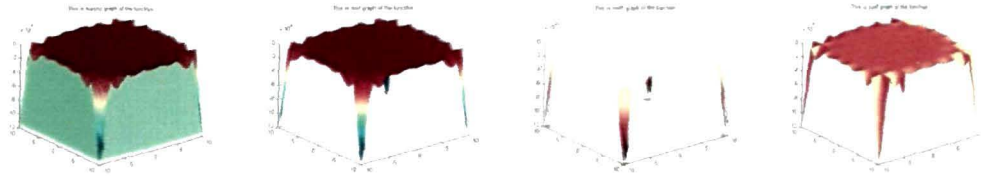
$$f(x) = - \left\{ \cos(x_1) \cos(x_2) e^{\left| 1 - \left[ (x_1^2 + x_2^2)^{0.5} \right] / \pi \right|} \right\}^2 / 30$$



**86. Holder Table Function:** This ‘tabular holder’ function has multiple local minima with four global minima at  $f(x^*) = 26.92$ . This function is given as:

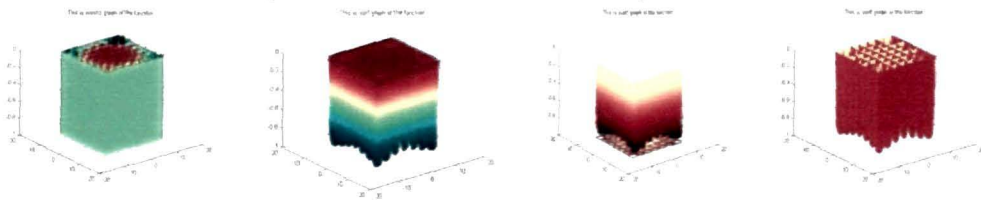
$$f(x) = - \left| \cos(x_1) \cos(x_2) e^{\left| 1 - \left[ (x_1^2 + x_2^2)^{0.5} \right] / \pi \right|} \right|$$





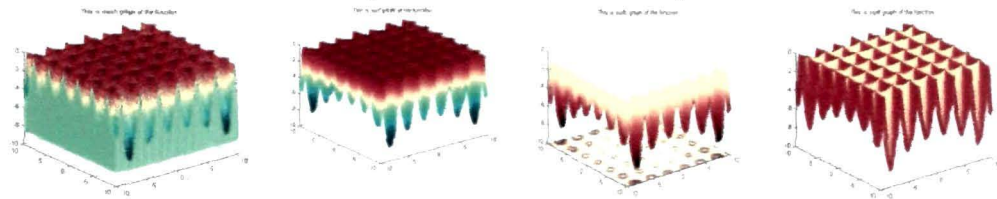
**87. Pen holder function:** This is a multi-modal function with  $f(x^*) = -0.96354$  in the search domain  $x_i \in [-11, 11]$ , is given as:

$$f(x) = -\exp\left\{-\left|\cos(x_1)\cos(x_2)e^{\left|1-\left[(x_1^2+x_2^2)^{0.5}/\pi\right]^{-1}}\right|}\right\}$$



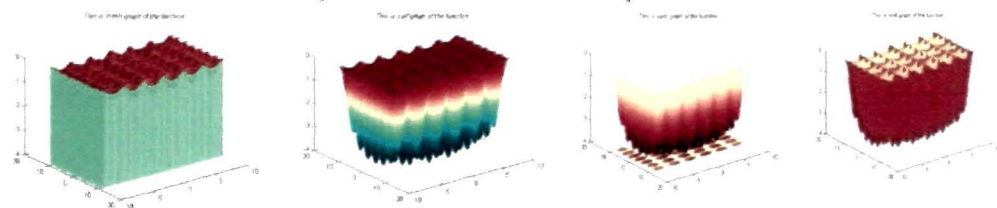
**88. Test tube Holder Function (a) :** This multimodal function is defined as follows. We obtain  $x^* = -10.8723$  in the domain  $x_i \in [-10, 10], i = 1, 2$

$$f(x) = -4\left|\sin(x_1)\cos(x_2)e^{\left|\cos(x_1^2+x_2^2)/200\right|}\right|$$



**89. Test tube Holder Function (b):** This multimodal function is defined as follows. We obtain  $x^* = -10.8723$  in the domain  $x_1 \in [-9.5, 9.4], x_2 \in [-10.9, 10.9]$ .

$$f(x) = -4\left|\sin(x_1)\cos(x_2)e^{\left|\cos((x_1^2+x_2^2)/200)\right|}\right|$$



(iii) **Some More Benchmark Test Functions (These functions have been adopted from CUTE)**

(<ftp://138.48.4.14/pub/cute/>)

**90. Extended Freudenstein & Roth function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} (-13 + x_{2i-1} + ((5 - x_{2i})x_{2i} - 2)x_{2i})^2 + (-29 + x_{2i-1} + ((x_{2i} + 1)x_{2i} - 14)x_{2i})^2 \text{ where } x_0 = [0.5, -2, 0.3, -2, 0.5, -2, \dots, 0.5, -2] \text{ and } x_i \in [-10, 10], i = 1, 2.$$

**91. Extended Trigonometric Function:** The function is defined as

$$f(x) = \sum_{i=1}^n \left( \left( n - \sum_{j=1}^n \cos x_j \right) + i(1 - \cos x_i) - \sin x_i \right)^2 \text{ where } x_0 = [0.2, 0.2, 0.2, \dots, 0.2]$$

**92. Extended Rosenbrock function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} c(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2, \text{ where } x_0 = [-1.2, 1, \dots, -1.2, 1] \text{ and } c=100.$$

**93. Generalized Rosenbrock function:** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} c(x_{i+1} - x_i^2)^2 + (1 - x_i)^2, \text{ where } x_0 = [-1.2, 1, \dots, -1.2, 1] \text{ and } c=100.$$

**94. Extended White & Holst function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} c(x_{2i} - x_{2i-1}^3)^2 + (1 - x_{2i-1})^2, \text{ where } x_0 = [-1.2, 1, \dots, -1.2, 1] \text{ and } c=100.$$

**95. Extended Beale function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} (1.5 - x_{2i-1}(1 - x_{2i}))^2 + (2.25 - x_{2i-1}(1 - x_{2i}^2))^2 + (2.625 - x_{2i-1}(1 - x_{2i}^3))^2 \text{ where } x_0 = [1, 0.8, \dots, 1, 0.8].$$

**96. Extended Penalty function:** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} (x_i - 1)^2 + \left( \sum_{j=1}^n x_j^2 - 0.25 \right)^2 \text{ where } x_0 = [1, 2, 3, \dots, n].$$

**97. Perturbed Quadratic function:** The function is defined as

$$f(x) = \sum_{i=1}^n ix_i^2 + \frac{1}{100} \left( \sum_{i=1}^n x_i \right)^2 \text{ where } x_0 = [0.5, 0.5, 0.5, \dots, 0.5].$$

**98. Raydan function #1:** The function is defined as  $f(x) = \sum_{i=1}^n \frac{i}{10} (\exp(x_i) - x_i)$  where

$$x_0 = [1, 1, 1, \dots, 1].$$

**99. Raydan function #2:** The function is defined as  $f(x) = \sum_{i=1}^n (\exp(x_i) - x_i)$  where

$$x_0 = [1, 1, 1, \dots, 1].$$

**100. Diagonal function #1:** The function is defined as  $f(x) = \sum_{i=1}^n (\exp(x_i) - ix_i)$  where

$$x_0 = [1/n, 1/n, 1/n, \dots, 1/n].$$

**101. Diagonal function #2:** The function is defined as  $f(x) = \sum_{i=1}^n \left( \exp(x_i) - \frac{x_i}{i} \right)$  where  $x_0 = [1/1, 1/2, 1/3, \dots, 1/n]$ .

**102. Diagonal function #3:** The function is defined as  $f(x) = \sum_{i=1}^n (\exp(x_i) - i \sin(x_i))$ , where  $x_0 = [1, 1, 1, \dots, 1]$ .

**103. Hager function:** The function is defined as  $f(x) = \sum_{i=1}^n (\exp(x_i) - \sqrt{i}x_i)$ , where  $x_0 = [1, 1, 1, \dots, 1]$ .

**104. Generalized Tri-diagonal 1 function:** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} (x_i + x_{i+1} - 3)^2 + (x_i - x_{i+1} + 1)^4 \quad \text{where } x_0 = [2, 2, 2, \dots, 2].$$

**105. Extended Tri-diagonal 1 function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} (x_{2i-1} + x_{2i} - 3)^2 + (x_{2i-1} - x_{2i} + 1)^4 \quad \text{where } x_0 = [2, 2, 2, \dots, 2].$$

**106. Extended (TET) function: (Three exponential term):** The function is defined

$$\text{as } f(x) = \sum_{i=1}^{n/2} (\exp(x_{2i-1} + 3x_{2i} - 0.1) + \exp(x_{2i-1} - 3x_{2i} - 0.1) + \exp(-x_{2i-1} - 0.1)),$$

$$x_0 = [0.0, 0.1, 0.1, \dots, 0.1].$$

**107. Generalized Tri-diagonal 2 function:** The function is defined as

$$f(x) = ((5 - 3x_1 - x_1^2)x_1 - 3x_2 + 1)^2 + \sum_{i=1}^{n-1} ((5 - 3x_i - x_i^2)x_i - x_{i-1} - 3x_{i+1} + 1)^2 + ((5 - 3x_n - x_n^2)x_n - x_{n-1} + 1)^2$$

$$x_0 = [-1, -1, -1, \dots, -1].$$

**108. Diagonal function #4 :** The function is defined as  $f(x) = \sum_{i=1}^{n/2} \frac{1}{2} (x_{2i-1}^2 + cx_{2i}^2)$

$$x_0 = [1, 1, 1, \dots, 1] \text{ and } c=100.$$

**109. Diagonal function #5:** The function is defined as

$$f(x) = \sum_{i=1}^n \log(\exp(x_i) + \exp(-x_i)) \quad \text{where } x_0 = [1.1, 1.1, \dots, 1.1].$$

**110. Extended Himmelblau function :** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} (x_{2i-1}^2 + x_{2i} - 11)^2 + (x_{2i-1} + x_{2i}^2 - 7)^2 \quad \text{where } x_0 = [1, 1, \dots, 1].$$

**111. Generalized White and Holst function:** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} c(x_{i+1} - x_i^3)^2 + (1 - x_i)^2 \quad \text{where } x_0 = [-1.2, 1, \dots, -1.2, 1], c = 100.$$

**112. Generalized PSC1 function:** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} (x_i^2 + x_{i+1}^2 + x_i x_{i+1})^2 + \sin^2(x_i) + \cos^2(x_i), \quad \text{where } x_0 = [3, 0.1, \dots, 3, 0.1].$$

**113. Extended PSC1 function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} (x_{2i-1}^2 + x_{2i}^2 + x_{2i-1}x_{2i})^2 + \sin^2(x_{2i-1}) + \cos^2(x_{2i}), \quad \text{where } x_0 = [3, 0.1, \dots, 3, 0.1].$$

**114. Extended Powell function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/4} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4 \text{ where } x_0 = [3, -1, 0, 1, \dots, 3, -1, 0, 1].$$

**115. Full Hessian FH1 function:** The function is defined as

$$f(x) = (x_1 - 3)^2 + \sum_{i=2}^n (x_i - 3 - 2(x_1 + x_2 + \dots + x_i))^2, \text{ where } x_0 = [0.01, 0.01, 0.01, \dots, 0.01].$$

**116. Full Hessian FH2 function:** The function is defined as

$$f(x) = (x_1 - 5)^2 + \sum_{i=2}^n (x_i + x_2 + x_3 + \dots + x_i - 1)^2, x_0 = [0.01, 0.01, 0.01, \dots, 0.01].$$

**117. Extended BD1 (Block Diagonal) function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} (x_{2i-1}^2 + x_{2i}^2 - 2)^2 + (\exp(x_{2i-1} - 1) - x_{2i})^2, \text{ where } x_0 = [0.1, 0.1, 0.1, \dots, 0.1].$$

**118. Extended Maratos function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} x_{2i-1} + c(x_{2i-1}^2 + x_{2i}^2 - 1)^2 \text{ where } x_0 = [1.1, 0.1, 1.1, \dots, 1.1, 0.1].$$

**119. Extended Cliff Function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} \left( \frac{x_{2i-1} - 3}{100} \right)^2 - (x_{2i-1} - x_{2i}) + \exp(20(x_{2i-1} - x_{2i})), \text{ where } x_0 = [0, -1, 0, -1, \dots, 0, -1].$$

**[129] Perturbed Quadratic Diagonal function:** The function is defined as

$$f(x) = \left( \sum_{i=1}^n x_i \right)^2 + \sum_{i=1}^n \frac{1}{100} x_i^2, \text{ where } x_0 = [0.5, 0.5, \dots, 0.5].$$

**121. Extended Wood Function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/4} 100(v_{4i-1} - v_{4i-2})^2 + (v_{4i-1} - 1)^2 + 90(v_{4i-1} - v_{4i})^2 + (1 - v_{4i-1})^2 + 10 \{ (v_{4i-2} - 1)^2 + (v_{4i} - 1)^2 \} + 19.8(v_{4i-1} - 1)(v_{4i} - 1)$$

Where  $x_0 = [-3, -1, -3, -1, \dots, -3, -1, -3, -1]$ .

**122. Extended Hiebert Function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} (x_{2i-1} - 10)^2 + (x_{2i-1}x_{2i} - 50000)^2, \text{ where } x_0 = [0, 0, 0, \dots, 0].$$

**123. Quadratic Function (QF1):** The function is defined as  $f(x) = \frac{1}{2} \sum_{i=1}^n ix_i^2 - x_n$

where  $x_0 = [1, 1, 1, \dots, 1]$ .

**124. Extended Quadratic penalty QP1 function:** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} (x_i^2 - 2)^2 + \left( \sum_{i=1}^n x_i^2 - 0.5 \right)^2 \text{ where } x_0 = [1, 1, 1, \dots, 1].$$

**125. Extended Quadratic penalty QP2 function:** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} (x_i^2 - \sin x_i)^2 + \left( \sum_{i=1}^n x_i^2 - 100 \right)^2 \text{ where } x_0 = [1, 1, 1, \dots, 1].$$

**126. Extended Quadratic QF2 function:** The function is defined as

$$f(x) = \frac{1}{2} \sum_{i=1}^n i(x_i^2 - 1)^2 - x_n, \quad x_0 = [0.5, 0.5, \dots, 0.5].$$

**127. Extended Quadratic exponential EP1 Function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} (\exp(x_{2i-1} - x_{2i}) - 5)^2 + (x_{2i-1} - x_{2i})^2 (x_{2i-1} - x_{2i} - 11)^2,$$

$$x_0 = [1.5, 1.5, 1.5, \dots, 1.5].$$

**128. Extended Tri-diagonal function:** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} (x_i x_{i+1} - 1)^2 + c(x_i + 1)(x_{i+1} + 1), \quad \text{where } x_0 = [1, 1, 1, \dots, 1], \quad \text{and } c = 0.1.$$

**129. FLET CBV3 function (CUTE):** The function is defined as

$$f(x) = \frac{1}{2} p(x_1^2 + x_n^2) + \sum_{i=1}^{n-1} \frac{p}{2} (x_i - x_{i+1})^2 - \sum_{i=1}^n \left( \frac{p(h^2 + 2)}{h^2} x_i + \frac{cp}{h^2} \cos(x_i) \right), \quad \text{where}$$

$$p = 1/10^8, \quad h = 1/(n+1), \quad c = 1, \quad x_0 = [h, 2h, \dots, nh].$$

**130. FLETCHCR function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} c(x_{i+1} - x_i - x_i^2)^2, \quad x_0 = [0, 0, 0, \dots, 0] \quad c = 100.$$

**131. BDQRTIC function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n-4} (-4x_i + 3)^2 + (x_i^2 + 2x_{i+1}^2 + 3x_{i+2}^2 + 4x_{i+3}^2 + 5x_n^2)^2, \quad x_0 = [1, 1, 1, \dots, 1].$$

**132. TRIDIA Function (CUTE):** The function is defined as

$$f(x) = \gamma(\delta x_1 - 1)^2 + \sum_{i=2}^n i(\alpha x_i - \beta x_{i-1})^2, \quad \text{and}$$

$$\alpha = 2, \quad \beta = 1, \quad \gamma = 1, \quad \delta = 1, \quad x_0 = [1, 1, 1, \dots, 1]$$

**133. ARGLINB function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^m \left( \sum_{j=1}^n ijx_j - 1 \right)^2, \quad x_0 = [1, 1, 1, \dots, 1].$$

**134. ARWHEAD function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} (-4x_i + 3)^2 + \sum_{i=1}^{n-1} (x_i^2 + x_n^2)^2, \quad \text{and } x_0 = [1, 1, 1, \dots, 1].$$

**135. NONDIA functions (CUTE):** The function is defined as

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^n 100(x_i - x_{i-1}^2)^2, \quad x_0 = [-1, -1, -1, \dots, -1].$$

**136. NONDQUAR function (CUTE):** The function is defined as

$$f(x) = (x_1 - x_2)^2 + \sum_{i=1}^{n-2} (x_i + x_{i+1} + x_n)^4 + (x_{n-1} + x_n)^2 \quad x_0 = [1, -1, 1, -1, \dots, -1].$$

**137. DQDRTIC function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n-2} (x_i^2 + cx_{i+1}^2 + dx_{i+2}^2), \quad c = 100, \quad d = 1000 \quad x_0 = [3, 3, 3, \dots, 3].$$

**138. EG2 function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} \sin(x_i + x_i^2 - 1) + \frac{1}{2} \sin(x_n^2), \quad x_0 = [1, 1, \dots, 1].$$

**139. CURLY20 function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^n q_i^4 - 20q_i^2 - 0.1q_i,$$

where

$$q_i = \begin{cases} x_i + x_{i+1} + \dots + x_{i+k} & i \leq n - k \\ x_i + x_{i+1} + \dots + x_n & i > n - k \end{cases} \quad k=20.$$

$$x_0 = [0.001/(n+1), \dots, 0.001/(n+1)]$$

**140. DIXMAANA-DIXMAANL function:** The function is defined as

$$f(x) = 1 + \sum_{i=1}^n \alpha x_i^2 \left(\frac{i}{n}\right)^{k1} + \sum_{i=1}^{n-1} \beta x_i^2 (x_{i+1} + x_{i+1}^2) \left(\frac{i}{n}\right)^{k2} + \sum_{i=1}^{2m} \gamma x_i^2 x_{i+1}^4 \left(\frac{i}{n}\right)^{k3} + \sum_{i=1}^m \delta x_{i+2m} \left(\frac{i}{n}\right)^{k4}, \quad m = n/3,$$

$$x_0 = [2., 2., 2., \dots, 2.].$$

	A	$\beta$	$\gamma$	$\delta$	K1	K2	K3	K4
A	1	0	0.125	0	0	0	0	0
B	1	0.0625	0.0625	0.0625	0	0	0	1
C	1	0.125	<b>0.125</b>	<b>0.125</b>	0	0	0	0
D	1	0.26	0.26	0.26	0	0	0	0
E	1	0	0.125	0.125	1	0	0	1
F	1	0.0625	<b>0.0625</b>	<b>0.0625</b>	1	0	0	1
G	1	0.125	<b>0.125</b>	<b>0.125</b>	1	0	0	1
H	1	0.26	0.26	0.26	1	0	0	1
I	1	0	0.125	0.125	2	0	0	2
J	1	0.0625	<b>0.0625</b>	<b>0.0625</b>	2	0	0	2
K	1	0.125	<b>0.125</b>	<b>0.125</b>	2	0	0	2
L	1	0.26	<b>0.26</b>	<b>0.26</b>	2	0	0	2

**141. Partial Perturbed Quadratic function:** The function is defined as

$$f(x) = x_1^2 + \sum_{i=1}^n \left( ix_i^2 + \frac{1}{100} (x_1 + x_2 + \dots + x_i)^2 \right), \quad x_0 = [0.5, 0.5, 0.5, \dots, 0.5].$$

**142. Broyden Tri-diagonal function:** The function is defined as

$$f(x) = (3x_1 - 2x_1^2)^2 + \sum_{i=2}^{n-1} (3x_i - 2x_i^2 - x_{i-1} - 2x_{i+1} + 1)^2 + (3x_n - 2x_n^2 - x_{n-1} + 1)^2, \quad x_0 = [-1, -1, -1, \dots, -1]$$

**143. Almost Perturb Quadratic function:** The function is defined as

$$f(x) = \sum_{i=1}^n ix_i^2 + \frac{1}{100} (x_1 + x_n)^2, \quad x_0 = [0.5, 0.5, 0.5, \dots, 0.5].$$

**144. Staircase 1 function:** The function is defined as

$$f(x) = \sum_{i=1}^n \left( \sum_{j=1}^i x_j \right)^2, \quad x_0 = [1, 1, 1, \dots, 1].$$

**145. Staircase 2 function:** The function is defined as  $f(x) = \sum_{i=1}^n \left[ \left( \sum_{j=1}^i x_j \right) - i \right]^2$ , and

$$x_0 = [0, 0, 0, \dots, 0].$$

**146. LIARWHD function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^n 4(-x_i + x_i^2)^2 + \sum_{i=1}^n (x_i - 1)^2, \quad x_0 = [4, 4, 4, \dots, 4].$$

**147. POWER function (CUTE):** The function is defined as  $f(x) = \sum_{i=1}^n (ix_i)^2$ ,

$$x_0 = [1, 1, 1, \dots, 1].$$

**148. ENGVAl1 function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} (x_i^2 + x_{i+1}^2)^2 + \sum_{i=1}^{n-1} (-4x_i + 3), \quad x_0 = [2, 2, 2, \dots, 2].$$

**149. CRAGGLVY function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^m (\exp(x_{2i-1}) - x_{2i})^4 + 100(x_{2i} - x_{2i+1})^6 + (\tan(x_{2i+1} - x_{2i+2}) + x_{2i+1} - x_{2i+2})^4 + x_{2i-1}^8 + (x_{2i+2} - 1)^2, \quad x_0 = [1, 2, 2, 2, \dots, 2]$$

**150. EDENSCH function (CUTE):** The function is defined as

$$f(x) = 16 + \sum_{i=1}^{n-1} [(x_i - 2)^4 + (x_i x_{i+1} - 2x_{i+1})^2 + (x_{i+1} + 1)^2], \quad x_0 = [0, 0, 0, \dots, 0].$$

**151. INDEF function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^n x_i + \sum_{i=2}^{n-1} \frac{1}{2} \cos(2x_i - x_n - x_1), \quad x_0 = \left[ \frac{1}{n+1}, \frac{2}{n+1}, \dots, \frac{n}{n+1} \right].$$

**152. CUBE function (CUTE):** The function is defined as

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^n 100(x_i - x_{i-1}^3)^2, \quad x_0 = [-1.2, 1, -1.2, 1, \dots, -1.2, 1].$$

**153. EXPLIN1 function (CUTE):** The function is defined as

$$f(x) = \exp(0.1x_i x_{i+1}) - 10 \sum_{i=1}^n (ix_i), \quad x_0 = [0, 0, 0, \dots, 0].$$

**154. EXPLIN2 function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^m \exp\left(\frac{ix_i x_{i+1}}{10m}\right) - 10 \sum_{i=1}^n (ix_i), \quad x_0 = [0, 0, 0, \dots, 0].$$

**155. ARGLINC function (CUTE):** The function is defined as

$$f(x) = 2 + \sum_{i=2}^{m-1} \left( \sum_{j=2}^{n-1} jx_j (i-1) - 1 \right)^2, \quad x_0 = [1, 1, 1, \dots, 1].$$

**156. BDEXP function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n-2} (x_i + x_{i+1}) \exp(-x_{i+2} (x_i + x_{i+1})), \quad x_0 = [1, 1, 1, \dots, 1]$$

**157. HARKERP2 function (CUTE):** The function is defined as

$$f(x) = \left( \sum_{i=1}^n x_i \right)^2 - \sum_{i=1}^n \left( x_i + \frac{1}{2} x_i^2 \right) + 2 \sum_{j=2}^n \left( \sum_{i=j}^n x_i \right)^2, \quad x_0 = [1, 2, 3, \dots, n].$$

**158. GENHUMPS function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} \sin(2x_i)^2 \sin(2x_{i+1})^2 + 0.05(x_i^2 + x_{i+1}^2), \quad x_0 = [-506., 506.2, \dots, 506.2].$$

**159. MCCORMCK function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} \left( -1.5x_i + 2.5x_{i+1} + 1 + (x_i - x_{i+1})^2 + \sin(x_i + x_{i+1}) \right), \quad x_0 = [1, 1, 1, \dots, 1].$$

**160. NONSCOMP function (CUTE):** The function is defined as

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^n 4(x_i - x_{i-1}^2)^2, \quad x_0 = [3, 3, 3, \dots, 3].$$

**161. VARDIM function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^n (x_i - 1)^2 + \left( \sum_{i=1}^n ix_i - \frac{n(n+1)}{2} \right)^2 + \left( \sum_{i=1}^n ix_i - \frac{n(n+1)}{2} \right)^4, \quad x_0 = \left[ 1 - \frac{1}{n}, 1 - \frac{2}{n}, \dots, 1 - \frac{n}{n} \right].$$

**162. QUARTC function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^n (x_i - 1)^4, \quad x_0 = [2, 2, \dots, 2].$$

**163. Diagonal 6 function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^n e^{x_i} - (1 - x_i), \quad x_0 = [1, 1, 1, \dots, 1].$$

**164. SINSQAD function (CUTE):** The function is defined as

$$f(x) = (x_1 - 1)^4 + \sum_{i=2}^{n-1} \left( \sin(x_i - x_n) - x_i^2 + x_i^2 \right)^2 + (x_n^2 - x_1^2)^2, \quad x_0 = [0.1, 0.1, 0.1, \dots, 0.1].$$

**165. Extended DENSCHNB function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} (x_{2i-1} - 2)^2 + (x_{2i-1} - 2)^2 x_{2i}^2 + (x_{2i} + 1)^2, \quad x_0 = [1, 1, 1, \dots, 1].$$

**166. Extended DENSCHNF function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} \left( 2(x_{2i-1} + x_{2i})^2 + (x_{2i-1} - x_{2i})^2 - 8 \right)^2 + \left( 5x_{2i-1}^2 + (x_{2i} - 3)^2 - 9 \right)^2, \quad x_0 = [2, 0, 2, 0, \dots, 2, 0].$$

**167. LIARWHD function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^n 4(x_i^2 - x_i)^2 + \sum_{i=1}^n (x_i - 1)^2, \quad x_0 = [4, 4, \dots, 4].$$

**168. DIXON3DQ function (CUTE):** The function is defined as

$$f(x) = (x_1 - 1)^2 + \sum_{j=1}^{n-1} (x_j - x_{j+1})^2 + (x_n - 1)^2, \quad x_0 = [-1, -1, \dots, -1].$$

**169. COSINE function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} \cos(-0.5x_{i+1} + x_i^2), \quad x_0 = [1, 1, 1, \dots, 1].$$

**170. SINE function:** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} \sin(-0.5x_{i+1} + x_i^2), \quad x_0 = [1, 1, 1, \dots, 1].$$

**171. BIGGSB1 function (CUTE):** The function is defined as

$$f(x) = (x_1 - 1)^2 + \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2 + (1 - x_n)^2, \quad x_0 = [0, 0, 0, \dots, 0].$$



**172. Generalized Quartic function:** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} x_i^2 + (x_{i+1} + x_i^2)^2, \quad x_0 = [1, 1, 1, \dots, 1].$$

**173. Diagonal 7 function:** The function is defined as

$$f(x) = \sum_{i=1}^n \exp(x_i) - 2x_i - x_i^2, \quad x_0 = [1, 1, 1, \dots, 1].$$

**174. Diagonal 8 function:** The function is defined as

$$f(x) = \sum_{i=1}^n x_i \exp(x_i) - 2x_i - x_i^2, \quad x_0 = [1, 1, 1, \dots, 1].$$

**175. Full Hessian FH3 function:** The function is defined as

$$f(x) = \left( \sum_{i=1}^n x_i \right)^2 + \sum_{i=1}^n (x_i \exp(x_i) - 2x_i - x_i^2), \quad x_0 = [1, 1, 1, \dots, 1].$$

**176. SINCOS function:** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} (x_{2i-1}^2 + x_{2i}^2 + x_{2i-1}x_{2i})^2 + \sin^2 x_{2i-1} + \cos^2 x_{2i}, \quad x_0 = [3, 0.1, 3, 0.1, \dots, 3, 0.1].$$

**177. Diagonal 9 function:** The function is defined as

$$f(x) = \sum_{i=1}^{n-1} (\exp(x_i) - ix_i) + 10000x_n^2, \quad x_0 = [1, 1, 1, \dots, 1].$$

**178. HIMMELBG function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} (2x_{2i-1}^2 + 3x_{2i}^2) \exp(-x_{2i-1} - x_{2i}), \quad x_0 = [1.5, 1.5, 1.5, \dots, 1.5].$$

**179. HIMMELH function (CUTE):** The function is defined as

$$f(x) = \sum_{i=1}^{n/2} (-3x_{2i-1} - 2x_{2i} + 2 + x_{2i-1}^3 + x_{2i}^2), \quad x_0 = [1.5, 1.5, 1.5, \dots, 1.5].$$

**180. Box 2-variable function:** The function is defined as

$$f(x) = \sum_{i=1}^{10} \left\{ \left[ \exp\left(\frac{-x_1 i}{10}\right) - \exp\left(\frac{-x_2 i}{10}\right) \right] - \left[ \exp\left(\frac{-i}{10}\right) - \exp(-i) \right] \right\}, \text{ with 5 different}$$

starting point (5,0), (0,0), (0,20), (2.5,10), (5, 20).

**181. Box 3-variable function:** The function is defined as

$$f(x) = \sum_{i=1}^{10} \left\{ \left[ \exp\left(\frac{-x_1 i}{10}\right) - \exp\left(\frac{-x_2 i}{10}\right) \right] - x_3 \left[ \exp\left(\frac{-i}{10}\right) - \exp(-i) \right] \right\}, .$$

**182. Step function:** This function is defined as  $f(x) = \sum_{i=1}^5 \text{int}(x_i)$  where  $-5.12 \leq x_i \leq 5.12$ .

#### (iv) Some more test functions

**183. Sum of different power functions:** The sum of different powers is a commonly used uni modal test function. The function is defined as  $f(x) = \sum_{i=1}^n |x_i|^{i+1}$  Test area is

usually restricted to hypercube  $-1 \leq x_i \leq 1, i=1,2,\dots,n$ . Its global minimum equal  $f(x) = 0$  is obtainable for  $x_i = 0, i=1,2,\dots,n$ .

**184. Langermann’s function:** The Langermann function is a multimodal test function.

The local minima are unevenly distributed. The function is defined as

$$f(x) = \sum_{i=1}^m c_i \exp \left[ -\frac{1}{\pi} \sum_{j=1}^n (x_j - a_{ij})^2 \right] \cos \left[ \pi \sum_{j=1}^n (x_j - a_{ij})^2 \right]$$

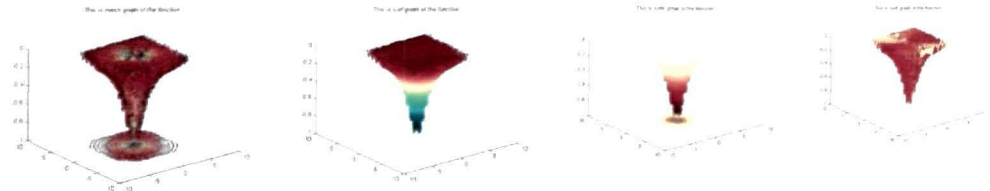
where  $c_i$  are constant numbers.

It is recommended to set the value of  $m=5$ .

**185. “Drop wave” function:** This is a multimodal test function. The given form of function has only two variable and the function is defined as

$$f(x_1, x_2) = -\frac{1 + \cos \left( 12\sqrt{x_1^2 + x_2^2} \right)}{\frac{1}{2}(x_1^2 + x_2^2) + 2}$$

where  $-5.12 \leq x_1, x_2 \leq 5.12$ .



**186. Shekel’s Foxholes function:** This is multi-model function. It is defined as

$$f(x) = -\sum_{i=1}^m \left( \sum_{j=1}^n \left[ (x_j - a_{ij})^2 + c_j \right] \right)^{-1}$$

where

$(c_i, i=1,2,\dots,m), (a_{ij}, j=1,2,\dots,n, i=1,\dots,m)$ . It is recommended to set  $m=30$ .

**187. Rotated Hyper ellipsoid function:** The function is defined as  $f(\bar{x}) = \sum_{i=1}^n (\sum_{j=1}^i x_j^2)$ .

Where  $-65.536 \leq x_i \leq 65.536$ . This is a uni-model continuous function. The optimum value is 0.

**188. Treccani function:** The function is defined as

$$f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} - x_1x_2 + x_2^2$$

where the bounds are  $-3 \leq x_i \leq 3 (i=1,2)$ . The global minimum is at  $(0,0)$  and  $(-2,0)$  at  $f(x^*) = 0$ .

**189. Function #1 :** The function is defined as  $f(x) = \sum_{i=1}^{30} (ix_i^4 + \text{Gauss}(0,1))$  and  $-1.28 \leq x_i \leq 1.28$ .

**190. Function#2:** The function is defined as

$$f(x) = 0.002 + \sum_{j=1}^{25} \left( \frac{1}{j} + \sum_{i=1}^2 (x_i - a_{ij})^6 \right), -65536 \leq x_i \leq 65536$$

**191. Function #3:** The function is defined as  $f(x) = 10V + \sum_{i=1}^{10} (-x_i \sin(\sqrt{|x_i|}))$ , where  $-500 \leq x_i \leq 500$   $V = 4189.829101$ .

**192. Function #4:** The function is defined as

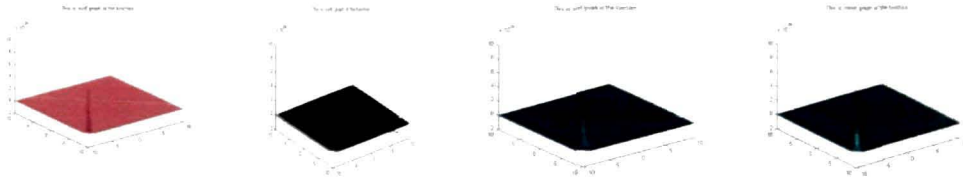
$$f(x) = 20A + \sum_{i=1}^{20} (x_i^2 - 10 \cos(2\pi x_i)), -512 \leq x_i \leq 512 \quad A = 10.$$

**193. Function #5:** The function is defined as  $f(x) = 1 + \sum_{i=1}^{10} \left( \frac{x_i^2}{4000} \right) - \prod_{i=1}^{10} \left( \cos \left( \frac{x_i}{\sqrt{i}} \right) \right)$

and  $-600 \leq x_i \leq 600$ .

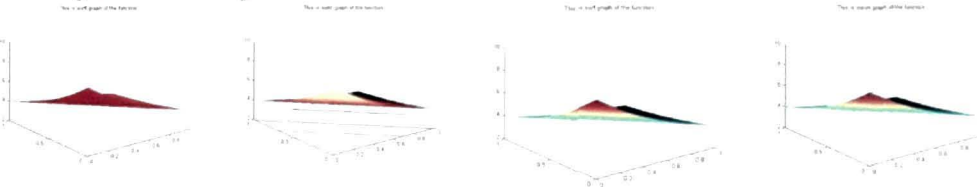
**194. Function #6 :** This function with  $f_{\min}(1,1,1,\dots,1) = 2$  may be defined as follows:

$f(x) = (1 + x_m)^{x_m}; x_m = m - \sum_{i=1}^{m-1} x_i; \forall i = 1, 2, 3, \dots, m$ . This function is not very difficult to optimize. However, its modification that gives us a new function (B) is considerably difficult.



**195. Function#7:** This function with  $f_{\min}(1,1,1,\dots,1) = 2$  may be defined as follows:

$f(x) = (1 + x_m)^{x_m}; x_m = m - \sum_{i=1}^{m-1} \frac{(x_i + x_{i+1})}{2}; x \in [0,1] \forall i = 1, 2, 3, \dots, m$ .  $X_m$  of the prior iteration indirectly enters into the posterior iteration. As a result, this function extremely difficult to optimize.



**196. Function#9:** The function is defined as

$$f(x) = \frac{\pi}{n} \left\{ 10 \sin^2(\pi y_1) + \sum_{k=1}^{n-1} (y_k - 1)^2 \left[ 1 + 10 \sin^2(\pi y_{k+1}) \right] + (y_n - 1)^2 \right\}.$$

$y_i = 1 + \frac{1}{4}(x_i - 1)$ , and  $-10 \leq x_i \leq 10$ ,  $i = 1, 2, 3, \dots, n$  and minimum value of the

function is 0. This function has roughly  $5^n$  local minima and a unique global minima located at  $x = (1, 1, 1, 1, \dots, 1)$ .

**197. New function #10:** A 2-d problems with  $f_{\min}(-8.4666, -9.9988) = -0.18466$

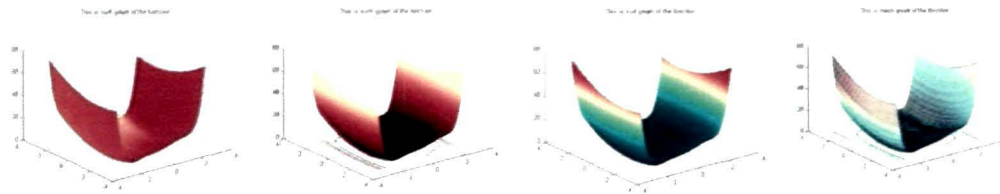
$$f(x) = \left| \cos \sqrt{x_1^2 + x_2^2} \right|^{0.5} + \frac{x_1 + x_2}{100}; x_i \in [-10, 10]; i = 1, 2$$

**198. New function #11:** This function is a variant of function #1 where  $\cos(\cdot)$  is replaced by  $\sin(\cdot)$ . Function has the optimum  $f_{\min}(-9.94112, -9.99952) = -0.199441$ . It is given by

$$f(x) = \left| \sin \sqrt{x_1^2 + x_2^2} \right|^{0.5} + \frac{x_1 + x_2}{100}; x_i \in [-10, 10]; i = 1, 2$$

**199. New function #12:** In the domain  $x \in [-10, 10]$  with  $f_{\min}(2.8863, 1.82326) = -2.28395$  This function is defined as

$$f(x) = -\ln \left[ \left\{ (\sin((\cos(x_1) + \cos(x_2)))^2)^2 - (\cos((\sin(x_1) + \sin(x_2)))^2)^2 \right\} + x_1 \right]^2 + \frac{(x_1 - 1)^2 + (x_2 - 1)^2}{10}$$

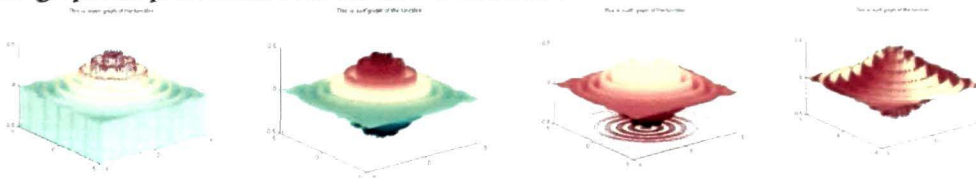


**200. Modified Masters Cosine wave function:** The function is defined as

$$f(x) = -\sum_{i=1}^{n-1} e^{-\frac{1}{8}(x_{i+1}^2 + 0.5x_i x_{i+1} + x_i^2)} \cos\left(4\sqrt{x_{i+1}^2 + 0.5x_i x_{i+1} + x_i^2}\right) \sin\left(4\sqrt{x_{i+1}^2 + 0.5x_i x_{i+1} + x_i^2}\right),$$

$-5 \leq x_i \leq 5$ .

The graphical presentation for the above function is



\*\*\*\*\*

# Appendix B

---

The MATLAB code for the visual representation of the collected test function and some new test function created are given here. The m-code of few functions are reported and most of them are available with the author. The code to draw the graph using meshz, surf, surfc and surf1 are also there as last paragraph.

## MATLAB code for visual presentation of test functions (Benchmark test functions and New Test functions)

### (i) Code for Benchmark Test function for visual presentation

```
% This Program draws four 3-D coloured graph of different function.
% Program developed by Sanjeev Kumar Singh.
%-----
%3. Ackley Function
r=-15:.05:30;
[X,Y]=meshgrid(r,r);
Z=20+exp(1)-20*exp(-0.2*sqrt((X.^2+Y.^2)/2))-
exp(0.5*(cos(2*pi.*X)+cos(2*pi.*Y)));
%-----
% 5.Beale Function
r=-4.5:.01:4.5;
[X,Y]=meshgrid(r,r);
Z=(1.5-X+X.*Y)^2+(2.25-X+X.*Y^2)^2+(2.625-X+X.*Y^3)^2;
%-----
%8. Bird Function
r=-2*pi:.1:2*pi;
[X,Y]=meshgrid(r,r);
Z=sin(X).*exp(abs(1-cos(Y))).*2+cos(Y).*exp(abs(1-cos(X))).*2+(X-
Y).^2;
%-----
% 6. Bohachevsky Function
r=-100:.9:100;
[X,Y]=meshgrid(r,r);
Z=X.^2+2*Y.^2-0.3*cos(3*pi*X)-0.4*cos(4*pi*Y)+0.7;
%-----
%7. Bohachevsky1 Function
r=-50:.5:50;
[X,Y]=meshgrid(r,r);
Z=X.^2+2*Y.^2-0.3*cos(3*pi*X)-0.4*cos(4*pi*Y)+0.7;
%-----
%10. Booth Function
r=-10:.01:10;
[X,Y]=meshgrid(r,r);
Z=(X+2*Y-7).^2+(2*X+Y-5).^2;
%-----
% 9. Branian function
r=-512:.9:512;
[X,Y]=meshgrid(r,r);
Z=-(Y+47).*sin(sqrt(abs(Y+(X/2)+47)))+sin(sqrt(abs(X-(Y+47))))*(-X);
```

```

%-----
% 11. Bukin Function
x=-15:0.1:-5;
y=-3:0.1:3;
[X,Y]=meshgrid(x,y);
%Z=100*Y.^2+0.01*abs(X+10);
Z=100*sqrt(abs(Y-0.01*X.^2))+0.01*abs(X+10);
%-----
% Carrom Table Function
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=-(cos(X).*cos(Y).*exp(abs(1-(sqrt(X.^2+Y.^2))))/pi).^2/30;
%-----
%12. Chichinadze function
r=-30:.9:30;
[X,Y]=meshgrid(r,r);
Z=X.^2-12*X+11+10*cos(pi*X/2)+8*sin(5*pi*X/2) -1/sqrt(5)*exp(-(Y-
0.5).^2/2);
%-----
% 14. Colville Function
r=-10:1.5:10;
%[X,Y,Z,T]=ndgrid(r,r,r,r);
[X,Y,Z,T]=flow;
M=100*(Y-X.^2).^2+(1-X).^2+90*(T-Z.^2).^2+(1-Z).^2+10.1*(Y-1).^2 ...
+(T-1).^2+19.8*(Y-1).*(T-1);
%-----
% Colville Function
%This is mesh graph of the function.
figure(1)
hpatch = patch(isosurface(X,Y,Z,T,M));
isonormals(X,Y,Z,T,hpatch)
set(hpatch,'FaceColor','red','EdgeColor','none')
daspect([1,4,4])
view([-65,20])
axis tight
camlight left;
set(gcf,'Renderer','zbuffer'); lighting phong
%-----
% 13. Corana function
r=-100:.9:100;
[X,Y]=meshgrid(r,r);
T1=0.2*abs(abs(X/0.2)+0.49999)*sign(X);
T2=0.2*abs(abs(Y/0.2)+0.49999)*sign(Y);
if ((X-T1) & (Y-T2)) < 0.05
Z=0.15*(T1-0.05*sign(T1)).^2+0.15*(T2-0.05*sign(T2)).^2;
else
Z= 1*X.^2+1000*Y.^2;
end
%-----
% 83. Cross function
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=(abs(sin(X).*sin(Y).*exp(abs(100-(sqrt(X.^2+Y.^2)/pi))))+1) .^(-
0.1);
%-----
%84. Crossleg function
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=-(abs(sin(X).*sin(Y).*exp(abs(100-(sqrt(X.^2+Y.^2)/pi))))+1) .^(-
0.1);
%-----

```

```

%81. Crossintray Function
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=0.0001*(abs(sin(X).*sin(Y).*exp(abs(100-(sqrt(X.^2+Y.^2)/pi))))+1)
.^0.1;
%-----
%82. Crowncross Function
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=0.0001*(abs(sin(X).*sin(Y).*exp(abs(100-(sqrt(X.^2+Y.^2)/pi))))+1)
.^0.1;
%-----
%15. Deflected Corrugated Spring Function
r=-100:.9:100;
[X,Y]=meshgrid(r,r);
Z=-cos(5*sqrt((X-2).^2+(Y-5).^2))+0.1*((X-2).^2+(Y-5).^2);
%-----
% 18. De Jong Function
c=-pi:.1:pi;
C1=-pi:.1:pi;
C2=-pi:.1:pi;
[X,Y]=meshgrid(c,c);
a=-100;
b=100;
ran=a+(b-a)*rand(63,63);
Z=((ran.*sin(C1)+rand.*cos(C1)+ran.*sin(C2)+rand.*cos(C2)) ...
-(ran.*sin(X)+rand.*cos(X)+ran.*sin(Y)+rand.*cos(Y))).^2;
%-----
% 19. Dixon Function
r=-10:.5:10;
[X,Y]=meshgrid(r,r);
Z=(X-1).^2+2*(2*Y.^2-X).^2;
%-----
% 185 Drop wave Function
r=-5.12:.1:5.12;
[X,Y]=meshgrid(r,r);
Z=-(1+cos(12*sqrt(X.^2+Y.^2)))./(0.5*(X.^2+Y.^2)+2);
%-----
% 21. Egg Holder Function
r=-512:.9:512;
[X,Y]=meshgrid(r,r);
Z=-((Y+47).*sin(sqrt(abs(Y+(X/2)+47)))+sin(sqrt(abs(X-(Y+47))))).*(-X));
%-----
% 20. Esom Function
r=-100:.5:100;
[X,Y]=meshgrid(r,r);
Z=-cos(X).*cos(Y).*exp(-(X-pi).^2-(Y-pi).^2);
%-----
% Extended Freudens Function
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=(-13+X+((5-Y).*Y-2).*Y).^2+(-29+X+((Y+1).*Y-14).*Y).^2;
%-----
% 22. Fletcher Powel Function
c=-pi:.1:pi;
C1=-pi:.1:pi;
C2=-pi:.1:pi;
[X,Y]=meshgrid(c,c);
a=-100;
b=100;

```

```

ran=a+(b-a)*rand(63,63);
Z=((ran.*sin(C1)+rand.*cos(C1)+ran.*sin(C2)+rand.*cos(C2)) ...
-(ran.*sin(X)+rand.*cos(X)+ran.*sin(Y)+rand.*cos(Y))).^2;
%-----
% 23. Freudenstein Function
r=-4:.1:4;
[X,Y]=meshgrid(r,r);
Z=(51*(X+X.^2+X.^3+X.^4-4)).^2+ ...
(52*(Y/2-1)+54*(Y.^2/4-1)+58*(Y.^3/8-1)+66*(Y.^4/16-1)).^2;
%-----
% 27. Giunta function
r=-1:0.01:1;
[X,Y]=meshgrid(r,r);
Z=0.6+(sin((16*X/15)-1)+(sin((16*X/15)-1)).^2+.02*sin(4*((16*X/15)-
1))). ...
+(sin((16*Y/15)-1)+(sin((16*Y/15)-1)).^2+.02*sin(4*((16*Y/15)-1)));
%-----
% 28. Goldstein Price Function
r=-10:.9:10;
[X,Y]=meshgrid(r,r);
Z=(1+(X+Y+1).^2.*(19-14*X+3*X.^2-14*Y+6*X.*Y+3*Y.^2)).* ...
(30+(2*X-3*Y).^2.*(18-32.*X+12*X.^2-48*Y-36*X.*Y+27*Y.^2));
%-----
% 24. Generalized Rastrigin Function
r=-5.12:.1:5.12;
[X,Y]=meshgrid(r,r);
Z=((X.^2-10*cos(2*pi.*X))+10)+(Y.^2-10*cos(1*pi.*Y))+10);
%-----
% 26. Griewank Function
r=-100:.5:100;
[X,Y]=meshgrid(r,r);
Z=((X.^2/4000)+(Y.^2/4000))-(cos(X).*cos(Y/sqrt(2)));
%-----
% 25. Generalized Schewefel Function
r=-500:.9:500;
[X,Y]=meshgrid(r,r);
Z=-((X.*sin(sqrt(abs(X)))+Y.*sin(sqrt(abs(Y)))));
%-----
%29. Himmelblau Function
r=-6.1:6;
[X,Y]=meshgrid(r,r);
Z=(X+Y.^2-7).^2+(X.^2+Y-11).^2;
%-----
% Holder Table Function
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=-abs(cos(X).*cos(Y).*exp(abs(1-(sqrt(X.^2+Y.^2))))/pi);
%-----
% 72. Three Hump Camel Function
r=-5:0.1:5;
[X,Y]=meshgrid(r,r);
Z=2*X.^2-1.05*X.^4+(X.^4/6)+X.*Y+Y.^2;
%-----
%31. Hump Function
r=-5.1:5;
[X,Y]=meshgrid(r,r);
Z=4*X.^2-2.1*X.^4+X.^6./3+X.*Y-4*Y.^2+4*Y.^4;
%-----
%33. Judge Function
r=-100:.9:100;
[X,Y]=meshgrid(r,r);

```



```

Z= (X+Y*sin(.286).^2+Y.^2*(cos(.645)-4.284)).^2 ...
+ (X+Y*sin(.973).^2+Y.^2*(cos(.585)-4.149)).^2 ...
+ (X+Y*sin(.348).^2+Y.^2*(cos(.310)-3.877)).^2 ...
+ (X+Y*sin(.276).^2+Y.^2*(cos(.058)-.533)).^2 ...
+ (X+Y*sin(.973).^2+Y.^2*(cos(.455)-2.211)).^2 ...
+ (X+Y*sin(.543).^2+Y.^2*(cos(.779)-2.389)).^2 ...
+ (X+Y*sin(.957).^2+Y.^2*(cos(.259)-2.145)).^2 ...
+ (X+Y*sin(.948).^2+Y.^2*(cos(.202)-3.231)).^2 ...
+ (X+Y*sin(.543).^2+Y.^2*(cos(.028)-1.998)).^2 ...
+ (X+Y*sin(.793).^2+Y.^2*(cos(.099)-1.379)).^2 ...
+ (X+Y*sin(.936).^2+Y.^2*(cos(.142)-2.106)).^2 ...
+ (X+Y*sin(.889).^2+Y.^2*(cos(.296)-1.428)).^2 ...
+ (X+Y*sin(.006).^2+Y.^2*(cos(.175)-1.011)).^2 ...
+ (X+Y*sin(.828).^2+Y.^2*(cos(.180)-2.179)).^2 ...
+ (X+Y*sin(.399).^2+Y.^2*(cos(.842)-2.858)).^2 ...
+ (X+Y*sin(.617).^2+Y.^2*(cos(.039)-1.388)).^2 ...
+ (X+Y*sin(.939).^2+Y.^2*(cos(.103)-1.651)).^2 ...
+ (X+Y*sin(.784).^2+Y.^2*(cos(.620)-1.593)).^2 ...
+ (X+Y*sin(.072).^2+Y.^2*(cos(.158)-1.046)).^2 ...
+ (X+Y*sin(.889).^2+Y.^2*(cos(.704)-2.152)).^2;
%-----
% 34. Keane Function
r=0:.1:10;
[X,Y]=meshgrid(r,r);
if ((X.*Y) >= 0.75 & (X+Y) <= 15)
Z=abs((((cosX).^4+(cosY).^4)-2*((cosX).^2*(cosY).^2)) ...
./sqrt(X.^2+2*Y.^2));
end
%-----
% 39. Leon Function
r=-1.2:0.01:1.2;
[X,Y]=meshgrid(r,r);
Z=100*(Y-X.^2)+(1-X).^2;
%-----
% 38. Levy Function
r=-10.1:10;
[X,Y]=meshgrid(r,r);
Z=(cos(2.*X+1)+2.*cos(3.*X+2)).*(cos(2.*Y+1)+2.*cos(3.*Y+2));
%-----
% 49. Master Cosine wave Function
r=-5:.1:5;
[X,Y]=meshgrid(r,r);
Z=-exp((-1/8)*(Y.^2+0.5*X.*Y+X.^2)).*cos(4*sqrt(Y.^2+0.5*X.*Y+X.^2))
...
.*sin(4*sqrt(Y.^2+0.5*X.*Y+X.^2));
%-----
% 40. Matyas Function
r=-10.1:10;
[X,Y]=meshgrid(r,r);
Z=0.26*(X.^2+Y.^2)-0.48*X.*Y;
%-----
% 44. Modified RCO Function
x=-5:.5:10;
y=0:.5:15;
[X,Y]=meshgrid(x,y);
%Z=(Y-(5.1/4*pi).^2)*Y.^2+(5/p1)*X-6).^2;
% Z=10*(1-(1/8*p1)).*cos(X).*cos(Y);
Z= log(X.^2+Y.^2+1);
%-----
% 51. Pathological Function
r=-100:.9:100;

```

```

[X,Y]=meshgrid(r,r);
Z=(((sin(sqrt(Y.^2+100*X.^2))).^2-0.5))./ ...
  ((0.001*(Y.^2-2*Y.*X+X.^2).^2+1.0))+0.5;
%-----
% 52. Paviani Function
r=2:.9:10;
[X,Y]=meshgrid(r,r);
Z=(log(X-2).^2+log(10-X).^2+log(Y-2).^2+log(10-Y).^2)-(X.*Y).^0.2;
%-----
% 87. Pen Holder Function
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=-exp(-abs(sin(X).*cos(Y).*exp(abs(1-(sqrt(X.^2+Y.^2)/pi))))).^(-1));
%-----
% 56. Quintic Function
x=-5:.5:10;
y=0:.5:15;
[X,Y]=meshgrid(x,y);
Z=(Y-(5.1/4*pi.^2)*Y.^2+(5/pi)*X-6).^2;
% Z=10*(1-(1/8*pi)).*cos(X).*cos(Y);
% Z= log(X.^2+T.^2+);
%-----
% 61. Rana Function
r=-500:.9:500;
[X,Y]=meshgrid(r,r);
Z=(Y+1).*cos(sqrt(abs(Y-X+1))).*sin(sqrt(abs(Y+X+1)))+ ...
  X.*cos(sqrt(abs(Y+X+1))).*sin(sqrt(abs(Y-X+1)));
%-----
% 73. Styblinski Tang Function
r=-5:0.1:5;
[X,Y]=meshgrid(r,r);
Z=0.5*((X.^4-16*X.^2+5*X)+(Y.^4-16*Y.^2+5*Y));
%-----
% 67. Subert Function
r=-10:.9:10;
[X,Y]=meshgrid(r,r);
Z=30+(cos(2*X)+2*cos(3*X)+3*cos(4*X)+4*cos(5*X)+5*cos(6*X)).* ...
  (cos(2*Y)+2*cos(3*Y)+3*cos(4*Y)+4*cos(5*Y)+5*cos(6*Y)).^2;
%-----
% 76. Yao Function
r=-1.28:.01:1.28;
[X,Y]=meshgrid(r,r);
a=0;
b=1;
ran=a+(b-a)*rand(257,257);
Z=ran + (X.^4+2*Y.^4);
%-----
% Treccani Function
r=-3:.1:3;
[X,Y]=meshgrid(r,r);
Z=2*X.^2-1.05*X.^4+(X.^6/6)-X.*Y+Y.^2;
%-----
% 78. Zakharov Function
r=-4:.1:10;
[X,Y]=meshgrid(r,r);
Z=(X.^2+Y.^2)+(X/2+(2*Y)/2).^2+(X/2+(2*Y)/2).^4;
%-----
% 80. Zero Sum Function
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=1+(10000*abs(sqrt((X+Y))));

```

```

%-----
% 79. Zettle function
r=-5:0.1:5;
[X,Y]=meshgrid(r,r);
Z=(X.^2+Y.^2-2*X).^2+0.25*X;
%-----
% New Test Function
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=abs(cos(sqrt(abs(X.^2+Y))))).^5+(X+Y)./100;
%-----

%-----Plotting code-----

%This is meshz graph of the function.
figure(1)
meshz(Z,
'facecolor','interp','Edgecolor','none','facelighting','phong')
title('This is meshz graph of the function.')
%-----
%This is surf graph of the function
figure(2)
surf(X,Y,Z,'facecolor','interp','Edgecolor','none','facelighting','phong')
axis('on')
title('This is surf graph of the function.')
%-----
%This is a surface contour of the function.
figure(3)
surfc(Z,'facecolor','interp','Edgecolor','none','facelighting','phong')
colormap hot
axis('on')
title('This is surfc graph of the function.')
%-----
%This is surface l graph of the function.
figure(4)
surfl(Z)
title('This is surfl graph of the function.')
shading interp
colormap hot

```

## (ii) Code for the new functions introduced

```

%-----
% This Program draws four 3-D coloured graph of different function.
% Program developed by Sanjeev Kumar Singh.
% F201 Tortoise function
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=abs(sin(X).*exp(abs(100-(X.^2+Y.^2))/p1)).^(.01)+abs(cos(X) ...
.*exp(abs(100-(X.^2+Y.^2))/p1)).^(.01);
%-----
% F202 (Inverted crosscap function)
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=abs(sin(X).*exp(abs(100-(X.^2-Y.^2))/p1)).^(.1)+abs(sin(X) ...
.*exp(abs(100-(X.^2+Y.^2))/p1)).^(.1),
%-----
% F203 (Crosscap function)
r=-10:.1:10;

```

```

[X,Y]=meshgrid(r,r);
Z=- (abs(sin(X.*Y).*exp(abs(100-
(X.^2+Y.^2))/pi)).^(.01)).*(abs(cos(X.*Y) ...
.*exp(abs(100-(X.^2+Y.^2))/pi)).^(.01));
%-----
% F204 (Fourhole table function)
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=- (abs(sin(X.*Y).*exp(abs(100-(X.^2+Y.^2))/pi)).^(.09)) ...
.* (abs(cos(X.*Y).*exp(abs(100-(X.^2+Y.^2))/pi)).^(.09));
%-----
%F205 (Cross on rough ceiling function)
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=- (abs(sin(X.*Y).*exp(abs(100-(X.^2+Y.^2))/pi)).^(.09)) ...
./ (abs(cos(X.*Y).*exp(abs(100-(X.^2+Y.^2))/pi)).^(.09));
%-----
% F206 (Crosshut function)
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=(abs(sin(X.*Y).*exp(abs(100-(X.^2/3+Y.^2/3))/pi)).^(.05)) ...
.* (abs(cos(X.*Y).*exp(abs(100-(X.^2/3+Y.^2/3))/pi)).^(.05));
%-----
% F207 (Inverted crosshut function)
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=- (abs(sin(X.*Y).*exp(abs(100-(X.^2/3+Y.^2/3))/pi)).^(.05)) ...
.* (abs(cos(X.*Y).*exp(abs(100-(X.^2/3+Y.^2/3))/pi)).^(.05));
%-----
% F208 (Umbrella function)
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=- (abs(ceil((X.^2/3+Y.^2/3)).*exp(ceil(abs(100-
(X.^2/3+Y.^2/3))/pi))).^(.01)) ...
.* (abs(floor((X.^2/3+Y.^2/3)).*exp(floor(abs(100-
(X.^2/3+Y.^2/3))/pi))).^(.01));
%-----
% F209 (Inverted Umbrella function)
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=- (abs(ceil((X.^2/3+Y.^2/3)).*exp(ceil(abs(100
(X.^2/3+Y.^2/3))/pi))).^(.01)) ...
.* (abs(floor((X.^2/3+Y.^2/3)).*exp(floor(abs(100-
(X.^2/3+Y.^2/3))/pi))).^(.01));
%-----
% F210 (Flower function)
r=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=(abs(ceil(sqrt(X.^2+Y.^2)).*exp(ceil(abs(100-
(X.^2+Y.^2))/pi))).^(.01)) ...
.* (abs(floor(sqrt(X.^2+Y.^2)).*exp(floor(abs(100-
(X.^2+Y.^2))/pi))).^(.01));
%-----
% F211 (Royalbaul function)
R=-10:.1:10;
[X,Y]=meshgrid(r,r);
Z=@(x) (abs(sin(x(:,1)).*exp(abs(100-(x(:,1).^2+x(:,2).^2))/pi)).^(-
1)+abs(cos(x(:,1)).*exp(abs(100-(x(:,1).^2+x(:,2).^2))/pi)).^(-1));
%-----
% Code to draw the Graph
%This is mesh graph of the function.

```

## Appendix B

```
figure(1)
mesh(X,Y,Z)
title('This is mesh graph of the function ')
%-----
%This is surf graph of the function
figure(2)
surf(X,Y,Z,'facecolor','interp','Edgecolor','none','facelighting','phong')
axis('on')
%view(0,30)
title('This is surf graph of the function.')
%-----
%This is a surface contour of the function
figure(3)
surfc(X,Y,Z,'facecolor','interp','Edgecolor','interp','facelighting','phong')
colormap hot
axis('on')
%view(0,40)
title('This is surfc graph of the function.')
%-----
%This is surface illuminated graph of the function.
figure(4)
surfl(X,Y,Z)
title('This is surfl graph of the function.')
shading interp
colormap hot
%view(0,60)
%-----
```

\*\*\*\*\*

# Appendix C

## Executed commands for new test functions

```
%-----  
F201 tortoise=@(x) (abs(sin(x(:,1)).*exp(abs(100-  
(x(:,1).^2/3+x(:,2).^2/3))/pi)).^(.05)).*abs(cos(x(:,1)).*exp(abs(100-  
(x(:,1).^2/3+x(:,2).^2/3))/pi)).^(.05));  
  
[x,f]=diffevolve(tortoise,250,[-10 -10],[10 10])  
[x,f]=diffevolve(tortoise,50,[-10 -10],[10 10])  
[x,f]=genetic(tortoise,250,[-10 -10],[10 10])  
[x,f]=genetic(tortoise,50,[-10 -10],[10 10])  
[x,f]=swarm(tortoise,250,[-10 -10],[10 10])  
[x,f]=swarm(tortoise,50,[-10 -10],[10 10])  
[x,f]=simanneal(tortoise,250,[-10 -10],[10 10])  
[x,f]=simanneal(tortoise,50,[-10 -10],[10 10])  
  
%-----  
F202 1crosscap=@(x) (- (abs(sin(x(:,1)).*x(:,2)).*exp(abs(100-  
(x(:,1).^2+x(:,2).^2))/pi)).^(.01)).*(abs(cos(x(:,1)).*x(:,2)).*exp(abs  
(100-(x(:,1).^2+x(:,2).^2))/pi)).^(.01));  
  
[x,f]=diffevolve(1crosscap,250,[-10 -10],[10 10])  
[x,f]=diffevolve(1crosscap,50,[-10 -10],[10 10])  
[x,f]=genetic(1crosscap,250,[-10 -10],[10 10])  
[x,f]=genetic(1crosscap,50,[-10 -10],[10 10])  
[x,f]=swarm(1crosscap,250,[-10 -10],[10 10])  
[x,f]=swarm(1crosscap,50,[-10 -10],[10 10])  
[x,f]=simanneal(1crosscap,250,[-10 -10],[10 10])  
[x,f]=simanneal(1crosscap,50,[-10 -10],[10 10])  
  
%-----  
F203 crosscap=@(x) ((abs(sin(x(:,1)).*x(:,2)).*exp(abs(100-  
(x(:,1).^2+x(:,2).^2))/pi)).^(.01)).*(abs(cos(x(:,1)).*x(:,2)).*exp(abs  
(100-(x(:,1).^2+x(:,2).^2))/pi)).^(.01));  
  
[x,f]=diffevolve(crosscap,250,[-10 -10],[10 10])  
[x,f]=diffevolve(crosscap,50,[-10 -10],[10 10])  
[x,f]=genetic(crosscap,250,[-10 -10],[10 10])  
[x,f]=genetic(crosscap,50,[-10 -10],[10 10])  
[x,f]=swarm(crosscap,250,[-10 -10],[10 10])  
[x,f]=swarm(crosscap,50,[-10 -10],[10 10])  
[x,f]=simanneal(crosscap,250,[-10 -10],[10 10])  
[x,f]=simanneal(crosscap,50,[-10 -10],[10 10])  
  
%-----  
F204 Four-holetable=@(x) (- (abs(sin(x(:,1)).*x(:,2)).*exp(abs(100-  
(x(:,1).^2+x(:,2).^2))/pi)).^(.09)).*(abs(cos(x(:,1)).*x(:,2)).*exp(abs  
(100-(x(:,1).^2+x(:,2).^2))/pi)).^(.09));  
  
[x,f]=diffevolve(function204,250,[-10 -10],[10 10])  
[x,f]=diffevolve(function204,50,[-10 -10],[10 10])  
[x,f]=genetic(function204,250,[-10 -10],[10 10])  
[x,f]=genetic(function204,50,[-10 -10],[10 10])  
[x,f]=swarm(function204,250,[-10 -10],[10 10])
```

```

[x,f]=swarm(function204,250,[-10 -10],[10 10])
[x,f]=simanneal(function204,250,[-10 -10],[10 10])
[x,f]=simanneal(function204,50,[-10 -10],[10 10])

%-----
F205 Crossonroughceiling =@(x) (- (abs(sin(x(:,1)).*x(:,2)).*exp(abs(100-
(x(:,1).^2+x(:,2).^2)/pi)).^(.09)) ./ (abs(cos(x(:,1)).*x(:,2)).*exp(abs
(100-(x(:,1).^2+x(:,2).^2)/pi)).^(.09)));

[x,f]=diffevolve(function205,250,[-10 -10],[10 10])
[x,f]=diffevolve(function205,50,[-10 -10],[10 10])
[x,f]=genetic(function205,250,[-10 -10],[10 10])
[x,f]=genetic(function205,50,[-10 -10],[10 10])
[x,f]=swarm(function205,250,[-10 -10],[10 10])
[x,f]=swarm(function205,50,[-10 -10],[10 10])
[x,f]=simanneal(function205,50,[-10 -10],[10 10])
[x,f]=simanneal(function205,250,[-10 -10],[10 10])

%-----
F206 crossshut =@(x) ((abs(sin(x(:,1)).*x(:,2)).*exp(abs(100-
(x(:,1).^2/3+x(:,2).^2/3)/pi)).^(.05)) .* (abs(cos(x(:,1)).*x(:,2)).*exp
(abs(100-(x(:,1).^2/3+x(:,2).^2/3)/pi)).^(.05)));

[x,f]=diffevolve(function206,250,[-10 -10],[10 10])
[x,f]=diffevolve(function206,50,[-10 -10],[10 10])
[x,f]=genetic(function206,250,[-10 -10],[10 10])
[x,f]=genetic(function206,50,[-10 -10],[10 10])
[x,f]=swarm(function206,50,[-10 -10],[10 10])
[x,f]=swarm(function206,250,[-10 -10],[10 10])
[x,f]=simanneal(function206,250,[-10 -10],[10 10])
[x,f]=simanneal(function206,50,[-10 -10],[10 10])

%-----
F207 invertedcrossshut=@(x) (- (abs(sin(x(:,1)).*x(:,2)).*exp(abs(100-
(x(:,1).^2/3+x(:,2).^2/3)/pi)).^(.05)) .* (abs(cos(x(:,1)).*x(:,2)).*exp
(abs(100-(x(:,1).^2/3+x(:,2).^2/3)/pi)).^(.05)));

[x,f]=diffevolve(function207,250,[-10 -10],[10 10])
[x,f]=diffevolve(function207,50,[-10 -10],[10 10])
[x,f]=genetic(function207,50,[-10 -10],[10 10])
[x,f]=genetic(function207,250,[-10 -10],[10 10])
[x,f]=swarm(function207,250,[-10 -10],[10 10])
[x,f]=swarm(function207,50,[-10 -10],[10 10])
[x,f]=simanneal(function207,50,[-10 -10],[10 10])
[x,f]=simanneal(function207,250,[-10 -10],[10 10])

%-----
F208 umrella=@(x) (-
(abs(ceil((x(:,1).^2/3+x(:,2).^2/3)).*exp(ceil(abs(100-
(x(:,1).^2/3+x(:,2).^2/3)/pi))))).^(.01)) .* (abs(floor((x(:,1).^2/3+x(:,
2).^2/3)).*exp(floor(abs(100-(x(:,1).^2/3+x(:,2).^2/3)/pi))))).^(.01));

[x,f]=diffevolve(function208,250,[-10 -10],[10 10])
[x,f]=diffevolve(function208,50,[-10 -10],[10 10])
[x,f]=genetic(function208,50,[-10 -10],[10 10])
[x,f]=genetic(function208,250,[-10 -10],[10 10])
[x,f]=swarm(function208,250,[-10 -10],[10 10])
[x,f]=swarm(function208,50,[-10 -10],[10 10])
[x,f]=simanneal(function208,50,[-10 -10],[10 10])
[x,f]=simanneal(function208,50,[-10 -10],[10 10])

```

```

%-----
F209 invertedumbrella =@(x) (-
(abs(ceil((x(:,1).^2/3+x(:,2).^2/3)).*exp(ceil(abs(100-
(x(:,1).^2/3+x(:,2).^2/3)/pi))).^(.01)).*(abs(floor((x(:,1).^2/3+x(:,
2).^2/3)).*exp(floor(abs(100-
(x(:,1).^2/3+x(:,2).^2/3)/pi))).^(.01))));

[x,f]=diffevolve(function209,250,[-10 -10],[10 10])
[x,f]=diffevolve(function209,50,[-10 -10],[10 10])
[x,f]=genetic(function209,50,[-10 -10],[10 10])
[x,f]=genetic(function209,250,[-10 -10],[10 10])
[x,f]=swarm(function209,250,[-10 -10],[10 10])
[x,f]=swarm(function209,50,[-10 -10],[10 10])
[x,f]=simanneal(function209,50,[-10 -10],[10 10])
[x,f]=simanneal(function209,250,[-10 -10],[10 10])

%-----
F210 flower
= @(x) ((abs(ceil(sqrt(x(:,1).^2+x(:,2).^2)).*exp(ceil(abs(100-
(x(:,1).^2+x(:,2).^2)/pi))).^(.01)).*(abs(floor(sqrt(x(:,1).^2+x(:,2)
.^2)).*exp(floor(abs(100-(x(:,1).^2+x(:,2).^2)/pi))).^(.01))));

[x,f]=diffevolve(function210,250,[-10 -10],[10 10])
[x,f]=diffevolve(function210,50,[-10 -10],[10 10])
[x,f]=genetic(function210,250,[-10 -10],[10 10])
[x,f]=genetic(function210,50,[-10 -10],[10 10])
[x,f]=swarm(function210,250,[-10 -10],[10 10])
[x,f]=swarm(function210,50,[-10 -10],[10 10])
[x,f]=simanneal(function210,250,[-10 -10],[10 10])
[x,f]=simanneal(function210,50,[-10 -10],[10 10])

%-----
F211 royalbowl=@(x) (abs(sin(x(:,1)).*exp(abs(100-
(x(:,1).^2+x(:,2).^2)/pi))).^(-1)+abs(cos(x(:,1)).*exp(abs(100-
(x(:,1).^2+x(:,2).^2)/pi))).^(-1);

[x,f]=diffevolve(baul,250,[-10 -10],[10 10])
[x,f]=diffevolve(baul,50,[-10 -10],[10 10])
[x,f]=genetic(baul,250,[-10 -10],[10 10])
[x,f]=genetic(baul,50,[-10 -10],[10 10])
[x,f]=swarm(baul,250,[-10 -10],[10 10])
[x,f]=swarm(baul,50,[-10 -10],[10 10])
[x,f]=simanneal(baul,50,[-10 -10],[10 10])
[x,f]=simanneal(baul,250,[-10 -10],[10 10])

```

\*\*\*\*\*