KLE TECH KLE Technological University
Creating Value
Leveraging Knowledge

School

of

Electronics and Communication Engineering

Mini Project Report

on

Stability Analysis of Particle Swarm Optimization Technique

By:

1. **Spoorti R Angadi**	USN: 01FE19BEC259
2. **Ravikumar Radder**	USN: 01FE19BEC262
3. **Ifra Khan**	USN: 01FE19BEC266
4. **Suraj Baddi**	USN: 01FE19BEC268

Semester: V, 2021-2022

Under the Guidance of

Prof. Rohit Kalyani

K.L.E SOCIETY'S
KLE Technological University,
HUBBALLI-580031
2021-2022

SCHOOL OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that project entitled " **Stability Analysis Of Particle Swarm Optimization Technique** " is a bonafide work carried out by the student team of **"Spoorti R Angadi [01FE19BEC259], Ravikumar Radder [01FE19BEC262], Ifra Khan [01FE19BEC266], Suraj Baddi [01FE19BEC268]"**. The project report has been approved as it satisfies the requirements with respect to the mini project work prescribed by the university curriculum for BE (V Semester) in School of Electronics and Communication Engineering of KLE Technological University for the academic year 2021-2022.

Rohit Kalyani	**Nalini C. Iyer**	**N. H. Ayachit**
Guide	**Head of School**	**Registrar**

External Viva:

Name of Examiners	**Signature with date**
1.	
2.	

ACKNOWLEDGMENT

The sense of achievement that comes with completing the stability study of Particle Swarm Optimization would be incomplete if we didn't acknowledge the names of the people who assisted us in completing it due to their clear guidance, support, and encouragement. This project is the culmination of the entire team's efforts. It may not be achievable without the kind counsel and cooperation of several individuals and schools. We'd want to thank each and every one of them from the bottom of our hearts. We are grateful to our esteemed institution, KLE Technological University, Hubballi, for allowing us to realise a long-held ambition of reaching the pinnacle of our profession. We are grateful to our Head of School of Electronics and Communication, Dr. Nalini C. Iyer, for providing the motivation and guidance needed to complete this industrial project. Prof. Rohit Kalyani, our advisor, deserves special thanks for his unwavering support and ideas. We also appreciate the direction and support of the entire Autonomous Electric Vehicle (AEV) team. Finally, we'd want to express our gratitude to everyone who contributed to this effort, whether directly or indirectly. We, too, are grateful to our guardians for recognising, encouraging, and assisting us in our endeavour.

-Ifra, Suraj Baddi, Ravikumar Radder, Spoorti Angadi.

ABSTRACT

In recent years, autonomous driving has been a major study topic. People all throughout the world are worried about their safety. Accidents are growing increasingly common, putting people's lives at danger. The primary goals of autonomous vehicles have been to reduce accidents and improve safety. Autonomous driving is a multidisciplinary study topic that includes vehicle dynamics, electronics, computer vision, geolocation, control, and planning. Due to its inevitability in the development of new algorithms in practically every applied branch of mathematics, optimization has become a major topic. Regardless of how broad optimization approaches are in study domains, there is always room for improvement. We give an overview of nature-inspired optimization with a background in fundamentals and classification, as well as their reliability applications. Though there are a variety of strategies for achieving optimality in optimization issues, nature-inspired algorithms have proven to be quite effective and have received a lot of attention in recent study. Ant Colony Optimization, Genetic Algorithm, and PSO Algorithm are the three algorithms that were employed. It includes a thorough analysis of the algorithms, as well as recommendations for effective algorithms. The information concerning error boundaries comes from an algorithm's stability study. The stability of a PSO with inertia weight and constriction coefficient is examined using the Lyapunov stability criterion. Conditions for acceleration parameters, constriction coefficient, and inertia weight are computed to ensure stability.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The action of making the best or most effective use of a situation or resource, according to the dictionary. It can also be defined as the process of ensuring that the likelihood and magnitude of exposures, as well as the number of people exposed, are kept to a minimum while considering economic, societal, and environmental concerns. Optimisation can be applied in a number of different ways. Every process has the potential to be made better. There isn't a single company that doesn't want to improve their processes. Many complex problems in science and industry can be formulated as optimization problems. Time, cost, and risk minimization, as well as profit, quality, and efficiency maximisation, are examples of optimization. For example, there are a variety of ways to build a network to optimise cost and service quality; a variety of ways to schedule a production to maximise efficiency, and so on. A large number of real-world optimization problems are intricate and difficult to solve in science, engineering, economics, and business. It is not possible. In a reasonable length of time, solve precisely. Approximation approaches are the most popular method for tackling this type of problem. Approximate algorithms can further be decomposed into two classes:

1. Specific heuristics
2. Meta-heuristics

Specific heuristics are problem-specific; they are created for and used to a specific problem. This report discusses metaheuristics, which are more generic approximation algorithms that may be used to solve a wide range of optimization problems. They are adaptable to any optimization task. By investigating the often-huge solution search space of these cases, metaheuristics solve examples of problems that are thought to be difficult in general.

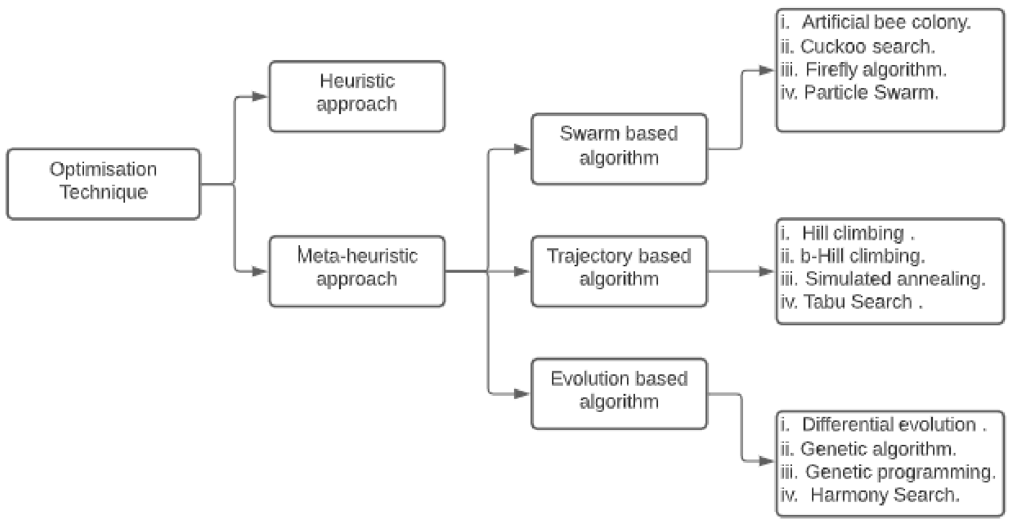


Figure 1.1: Types of Optimisation Technique

From figure 1.1 that Meta-heuristic approach are further classified into nature/swarm based algorithm, trajectory-based algorithm and evolutionary algorithm. A set of unique problem-solving procedures and approaches derived from natural processes is known as nature inspired algorithms. Nature-inspired algorithms include the genetic algorithm, particle swarm optimization, cuckoo search algorithm, and ant colony optimization.

These Nature Inspired algorithms have some gaps, which include:

1. Mathematical Framework for stability and convergence
2. Parameter Tuning
3. Role of benchmarking
4. Performance measure for fair comparision
5. Large-scale Scalability

Three of the gaps addressed in this research are stability analysis, rate of convergence, and parameter tuning. To close these gaps, one of the most basic yet useful nature-inspired algorithms, Particle Swarm Optimisation, is applied (PSO). PSO is a well-known algorithm that has been acknowledged in the literature and said to have been used to efficiently handle a variety of real-world issues. Particle Swarm Optimization (PSO) simulates the swarming behaviour seen in herds of animals, flocks of birds, and other social groups where individuals benefit from each other's discoveries and experiences while seeking for food. The system is started up with a random population of particles. Particles explore a D-dimensional space during optimization. Each particle keeps track of its own position, velocity, and optimal position. PSO is a global optimization algorithm that attempts to locate the optimum solution as a point in a D-dimensional space for problems. Each particle keeps track of its position in the issue space in terms of coordinates, which contributes to the optimum solution. Particles then move in solution space and are evaluated using fitness functions after each iteration. When compared to other optimization approaches, PSO has a faster convergence ability. The calculation of optimal value requires fewer parameters. PSO's stability can be evaluated using stability criteria such as Exploration, Exploitation, and Rate of Convergence. Some stability criteria, such as Exploration, Exploitation, and Rate of Convergence, can be used to assess PSO's stability. These stability criteria can be met by adjusting the parameters of the particle swarm algorithm and employing theorems such as constriction co-efficient, velocity clamping, and damping ratio.

1.1 Motivation

With the continual growth of automobiles from manual handling to machine moments to self-driving technology, the fundamental purpose behind this project was to evolve along with the rapidly changing technology, specifically the autonomous of vehicle. Where within the automotive industry, self-driving technology is the most fiercely debated innovation. The number of cars on the road has expanded considerably as a result of the widespread use of automobiles. As a result, new concerns have arisen, such as passenger safety and comfort, fuel consumption optimization, and pollutant emission reduction. Automatic control can help overcome these issues by assisting in the development of autonomous vehicles. Vehicles equipped with a reliable and stable controller. Study of Nature inspired algorithm is stressed as they have global optimality and can handle wide range of problems.

PSO uses a probabilistic approach to provide solutions. Despite the fact that particle swarm optimization (PSO) has been widely employed to solve a variety of complex engineering problems, it still has major drawbacks, such as premature convergence and low precision. Because the control parameter selection affects the final optimization outcome, an improved convergence particle swarm optimization algorithm is proposed, which can increase the update randomness for both particle velocity and position while also increasing the flexibility of algorithm parameters. Identifying error boundaries throughout the search process may thus aid in the development of a better PSO. The stability study of an algorithm provides information on error bounds.

1.2 Objectives

The following are the key objectives on which we focused:

1. Numerical Optimization Technique.
2. Particle Swarm Optimization.
3. Stability Analysis of Linear and Non-Linear systems.
4. Propose a new methodology to determine stability of Optimization Technique.
5. Evaluate Stability and rate of convergence of PSO.

1.3 Literature Survey

When we started our project, the team wanted to work in the field of optimization. Optimization is the action of making the best or most effective use of a situation or resource, according to the dictionary. Optimization can be applied in a number of different ways. Every method has the ability to be improved. There is not a single firm that isn't interested in optimising processes. Many complex problems in science and industry can be formulated as optimization problems. A large number of real-world optimization problems are intricate and difficult to solve in science, engineering, economics, and business. In a reasonable length of time, they are impossible to solve precisely. Optimization methodology is the most popular method for solving this type of problem. Optimization algorithms can further be decomposed into two classes:

1. Specific heuristic
2. Meta heuristic

Specific heuristics are problem-specific; they are created for and used to a specific problem. Metaheuristic are more generic approximation algorithms that may be used to solve a wide range of optimization problems. They are adaptable to any optimization task. By investigating the often-huge solution search space of these cases, metaheuristics solve examples of problems that are thought to be difficult in general [7]. Metaheuristic includes a number of algorithms. Some of them are algorithms that were inspired by nature.

Nature-inspired algorithms are a collection of one of a kind problem-solving strategies and approaches based on natural phenomena. Many scholars around the world have been enamoured with the creation of advanced intelligent algorithms and their usefulness in tackling a wide range of complicated engineering problems quickly because nature inspired algorithms are simple and versatile. This Special Issue contains some of the most interesting applications of nature-inspired optimization algorithms in resolving nonlinear complex problems such as automobiles, insurance, and fraud detection, using fuzzy clustering algorithms based on modified whale optimization algorithms, image compression using the explored bat algorithm, and accurate localization of sensor nodes in underwater sensor networks using a Doppler shift and modified genetic algorithm based localization techniques. Parameters tuning of a quadrotor PID controllers etc [5]. Some of the popular examples of nature inspired optimization algorithms include:

1. Artificial Bee Colony Optimization Algorithm
2. Particle Swarm Intelligence Optimization
3. Ant Colony Optimization Algorithm
4. Genetic Algorithm etc.

The team next went over each of the Nature Inspired algorithms in further depth, starting with Artificial Bee Colony Optimization, a population-based search technique in which artificial bees alter food placements. There are three categories of bees in a colony: employed bees, bystanders, and scouts. Bees tend to seek out food sources with higher nectar content based on their own and their nestmate's experiences. Food sources are chosen by both employed and onlooker bees [2].Ant Colony Optimisation is the next step. Because ants are blind, they will take the quickest path from their starting location to the food source. The chemical pheromone, which aids in the quest for the shortest path, is employed for unintentional communication between ants. When an ant comes across an obstruction in its path, it changes direction and searches for a new shortest path [3].

Then we went over to Genetic Algorithm. A GA starts with a population, which is a collection of solutions. The four operators are Initialization, Selection, Crossover, and Mutation. The initialization operator generates a population and assigns a fitness function for determining the fitness value. The selection operator picks chromosomes from the population for mating. The crossover operator is used to share information between two chromosomes. The mutation operator modifies the starting values of one or more genes on a chromosome. The evolution procedure is repeated until the end condition of the problem is reached [4].

Finally, we started our survey on Particle Swarm Optimisation (PSO). PSO is a well-known algorithm that has been acknowledged in the literature and said to have been used to efficiently handle a variety of real-world issues. Particle Swarm Optimization (PSO) simulates swarming behaviour seen in herds of animals, flocks of birds, and other social groups where individuals benefit from each other's discoveries and experiences while hunting for food. The system is started up with a random population of particles. Particles explore a D-dimensional space during optimization. Each particle keeps track of its own position, velocity, and optimal position. PSO is a global optimization algorithm that attempts to locate the optimum solution as a point in a D-dimensional space for problems. Each particle keeps track of its position in the issue space in terms of coordinates, which contributes to the optimum solution. Particles then move in solution space and are evaluated using fitness functions after each iteration. When compared to other optimization approaches, PSO has a faster convergence ability. The calculation of optimal value requires fewer parameters [6].

Despite the success and popularity of nature-inspired algorithms, there are still a number of concerns with them, particularly from a theoretical standpoint.Though academics have a good understanding of how such algorithms function in practise, it is unclear why they work and under what conditions they work. There are five unsolved issues in the field of nature-inspired algorithms:

1. Mathematical framework for stability and convergence.
2. Parameter tuning.
3. Role of benchmarking.
4. Performance measures for fair comparison.
5. Large-scale scalability.

Under mathematical framework for stability and convergence, the subject of how to construct a unified framework for mathematically assessing all nature inspired algorithms in order to acquire in-depth information about their rate of convergence, stability, and robustness is addressed. Then there's parameter tuning, which involves determining how to optimally adjust a particular algorithm's parameters so that it may attain its greatest performance for a specific set of issues, as well as how to alter or regulate these parameters to maximise an algorithm's performance. The No-Free-Lunch Theorem argues that employing benchmark functions to examine how a new algorithm performs in contrast to previous algorithms is an important research for any new algorithm, especially a new nature-inspired algorithm.

Researchers can use this form of benchmarking to learn more about the algorithm's convergence behaviour, stability, and benefits and drawbacks. Finally, performance metrics and how they can influence the conclusions achieved when benchmarking different algorithms are discussed in the last issue. To create a comparison, researchers must use relevant performance measures.

In the current literature, accuracy, computational effort, stability, and success rates are the primary concerns of comparison studies. The issue here is deciding which performance measurements are best for comparing all algorithms fairly. Is it possible to develop an uniform framework that allows all algorithms to be compared objectively and fairly [9]. As these were the most common issues, the team decided to work on some of these gaps, which included stability control, rate of convergence, and parameter tuning.

Starting with rate of convergence. Particle velocities typically build up too quickly, and the maximum of the goal function is neglected. The step size of the swarm is determined by particle velocity in PSO. Every step adjusts the velocity with which all particles move in every dimension of the search space. The following techniques have been developed to speed up convergence and improve the quality of PSO solutions:

1. Velocity clamping
2. Inertia weight
3. Constriction co-efficient

In order to comb over the search space, velocity clamping helps particles stay within the boundary and take reasonable step sizes. The inertia weight is denoted by (ω) and is thought to replace by modifying the influence of prior velocities in the process, i.e. it regulates the particle's momentum by weighing the contribution of previous velocities. At each time step, the inertia weight (ω) will be compounded by the velocity from the preceding time step. It can be implemented as a fixed value or as values that change dynamically. Earlier versions employed a single fixed value for all particles throughout the operation, but now dynamically varying inertia values are used. Convergence can be ensured using the inertia weight technique. The constriction co-efficient is the next step. The constriction factor (χ) was introduced as a new parameter in this technique. The constriction coefficient reduces the amplitude of the particle's oscillation and focuses on the local and neighbourhood previous best points. If the particle's prior best location and the best position in the neighbourhood are close to each other, the particles will do a local search. If their positions are widely apart, on the other hand, the particles will conduct a worldwide search. The constriction coefficient ensures particle convergence over time while simultaneously preventing collapse. Eberhart and Shi demonstrated empirically that using the constriction coefficient and velocity clamping together results in a faster convergence rate [8].

Exploration entails searching the entire sample space (exploring the sample space), whereas exploiting entails pursuing the potential areas discovered during the exploration. As a result, in a good optimization algorithm, these two properties must be balanced.

Despite PSO's durability in addressing complex optimization problems, quick convergence can easily trap the swarm in some local optima when solving multimodal issues with it. As a result, effectively utilising the swarm's exploration (global examination of the search area) and exploitation (finer search around a local optimum) abilities is critical to PSO's effectiveness, particularly for complicated multimodal issues with a significant number of local minima. There are three primary approaches to this problem:

1. PSO with adjusted parameters
2. PSO with various topology structures
3. PSO with hybrid strategies

However, we concentrated on the first type, which is PSO with modified parameters. Appropriate control parameters, such as inertia weight and acceleration coefficients c1 and c2, clearly have a major impact on the swarm's exploration and exploitation capacities. By increasing the value of considerably, the swarm's variety grows, while increasing the values of c1 and c2 pushes the particles toward their historical optimal location, or the swarm's overall optimal position [10].In addition to these, characteristics such as velocity clamping, inertia weight, and constriction coefficient help Exploration and Exploitation work better.

In addition, during our implementation on PSO, there was an issue with the algorithm's halting criterion. Stopping criteria are requirements that must be met before an algorithm can be ended. Some of the most typical stopping criteria are execution time, total number of iterations, non-improving iterations, optimal (lower bound for min, upper bound for max) solution identified, and so on [1].

Some of the stopping criteria are:

1. When the maximum number of iterations or function evaluations is achieved, the algorithm is halted. If the maximum number of iterations is set too low, the search process may end before a satisfactory result is found.
2. The method is ended in the second approach when there is no significant progress after a certain number of iterations. This progress can be monitored in a variety of ways. For example, if the average change in the particle positions is very modest or the average

velocity of the particles is almost zero across a number of iterations, the process may be assumed to have finished.

3. When the normalised swarm radius approaches zero, the method is finished. The ratio of the maximum swarm radius to the swarm's diameter is known as the normal swarm radius [8].

1.4 Problem Statement

To develop stability criteria and hence evaluate the rate of convergence of particle swarm optimization.

1.5 Application in Societal Context

The main goal of this project is to create a stable algorithm inspired by nature. Once that is accomplished, we can make the most significant contribution to society by applying this algorithm to any closed loop system or controller that automatically tunes its parameters based on the algorithm, and because the algorithm is stable, it in turn makes the controller stable, reducing feedback errors in closed loop systems.

1.6 Organization of the Report

We provide the brief description about introduction, motivation for opting this topic in the Chapter 1 : the main objectives concerned for this project, literature survey, which is our brief study for understanding this project and its working and problem statement which is selecting by the need of current requirement. Project planning, the process or structure which we followed accomplishing this project, Applications in Societal Context, which gives a brief description of usage of this topic in society. Chapter 2 : It main describes the System design which defines the block diagram of the system, the alternate solutions for same problem statement and selecting the final solution which is optimised and works better than other methodology. Chapter 3 : Implementation Details outlines the logic and flow of the intended system, as well as the specifications and detailed explanations of each stage. Chapter 4 : This Chapter is discussion on the results obtained from the system, which gives a detailed study of which method is better compared to its comparative methods which are also described above. Chapter 5 : This is the project's conclusion, including a discussion of the project's future scope, such as what tools and techniques can be applied in other areas and how they might be used more effectively.

Chapter 2

Mathematical Framework

The functions that our design must perform regarding the various constraints and various means of achieving these functions are described in system design. The basic blocks of the model, as well as what each block does, are included in the design process. This chapter also covers the algorithm's implementation and the changes made to tune the parameters to make it more stable.

2.1 Particle Swarm Optimisation:

Particle Swarm Optimization (PSO) is a multi-agent parallel search technique that maintains a swarm of particles, each of which represents one of the swarm's potential solutions. Every particle moves across a multidimensional search space, altering its position based on its own and neighbouring particles' experiences. As a result, in a PSO technique, all particles are released at random and evaluated to determine fitness, as well as the personal best (best value of each particle) and global best (best value of particle in the entire swarm). After that, a loop is launched in order to get the optimal response. The personal and global bests are used to update the particle's velocity in the loop, and the current velocity is then utilised to update the position of each particle. A defined termination criterion brings the loop to a close. Using the formula below, we compute the particles' updated position and velocity:

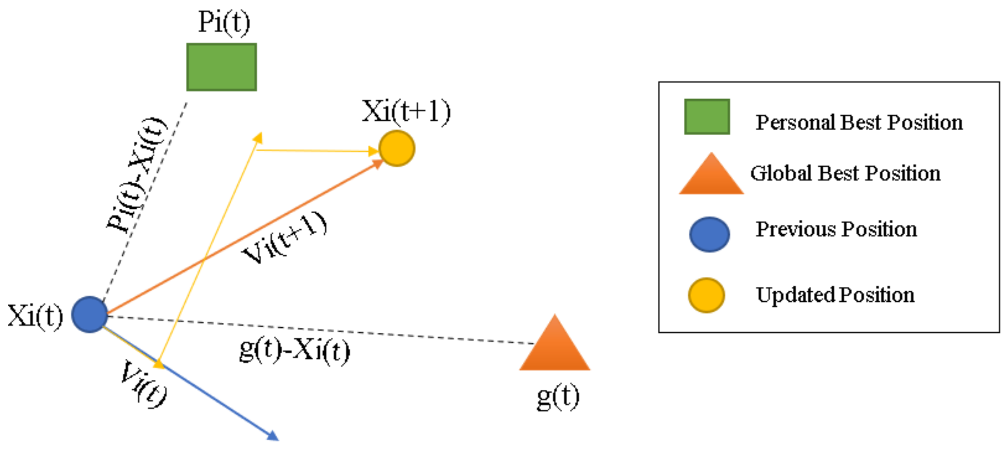


Figure 2.1: Formulation of PSO

$$Vi(t+1) = wVi(t) + r1c1(Pi(t) - Xi(t)) + r2c2(G(t) - Xi(t)) \tag{2.1}$$

$$Xi(t+1) = Xi(t) + Vi(t+1) \tag{2.2}$$

Here in equation (2.1) and (2.2), Vi(t) is the current velocity and Vi(t+1) is the updated velocity of particle ‘i’, r1 and r2 are the random variables, c1 and c2 are the acceleration co-efficient, Xi(t) and Xi(t+1) is the current and the updated position of particle i.

2.1.1 Algorithm of PSO

The following steps shows the Algorithm of how particle swarm optimisation works. Starting with initialising a group of particles, we evaluate the personal and global best using the necessary mathematical formula of velocity and position.

Step 1: Start

Step 2: Initialise a group of particles

Step 3: Evaluate personal best for each particle

Step 4: If current position is better than personal best, update personal best

Step 5: Assign the personal best to global best

Step 6: Compute Velocity

Step 7: Update particle position

Step 8: Repeat from step 3 to step 7 until the target is reached

Step 9: End

2.1.2 Flowchart of PSO

The following figure 2.2 represents a workflow or process of how particle swarm optimisation works. As mentioned in the above section (2.1.2) the steps remains the same and are represented in terms of boxes of various kinds.

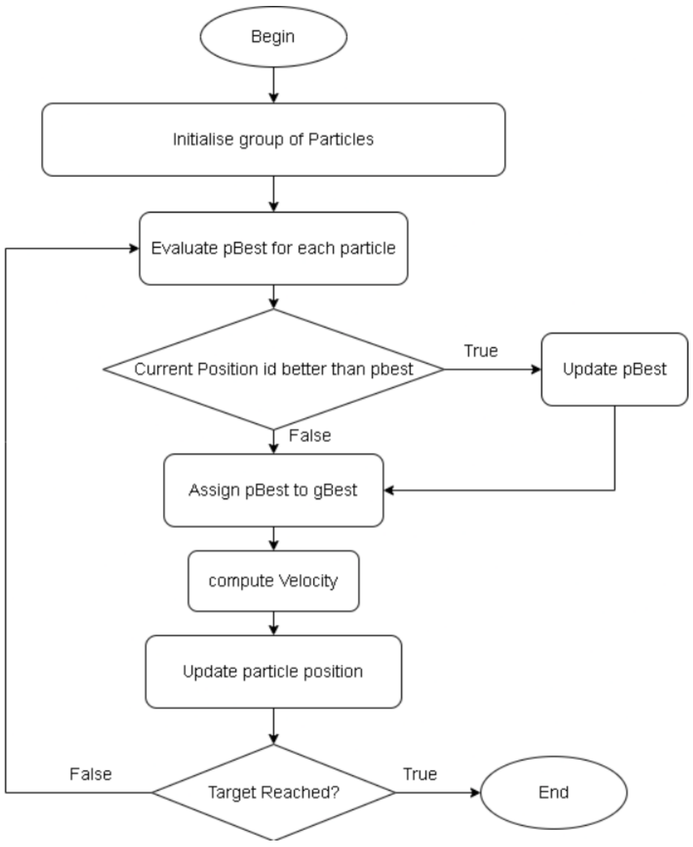


Figure 2.2: Flowchart of PSO

2.2 Stability Analysis of PSO

As discussed earlier that PSO provides solutions in probabilistic manner, we must ensure that there are some error bounds so that the PSO is stable and performs well. Certain stability criteria must be met in order for PSO to be stable. They are exploration, exploitation, and convergence rate. Exploration is the ability to explore multiple areas of the search space in order to find a decent optimum, whereas exploitation is the ability to focus the search on a specific area in order to refine a promising answer. Certain concepts, such as constriction coefficient and inertia weight, can be used to meet these stability criteria. If the stability criteria are not met, use the following principles to tweak the PSO parameters so that the respective conditions are met.

2.2.1 Algorithm with Stabilty Analysis of PSO

The following steps shows the Algorithm of how particle swarm optimisation works with Stability analysis. Once we get an optimal value from PSO after using the algorithm mentioned in section 2.1.1, we check for all the stability criteria. If not met, we tune the parameters of PSO and check again for stability criteria.

```
Step 1: Start
Step 2: Initialise a group of particles
Step 3: Evaluate personal best for each particle
Step 4: If current position is better than personal best, update personal best
Step 5: Assign the personal best to global best
Step 6: Compute Velocity
Step 7: Update particle position
Step 8: Repeat from step 3 to step 7 until the target is reached
Step 9: Once the stability criteria is reached check for the stability criteria
Step 10: If stability criteria is not met, tune the parameters
Step 11: End
```

2.2.2 Flowchart with Stabilty Analysis of PSO

The following figure 2.3 shows represents a workflow or process of how particle swarm optimisation works along stability analysis of it. As mentioned in the above section (2.2.1) the steps remains the same and are represented in terms of boxes of various kinds.

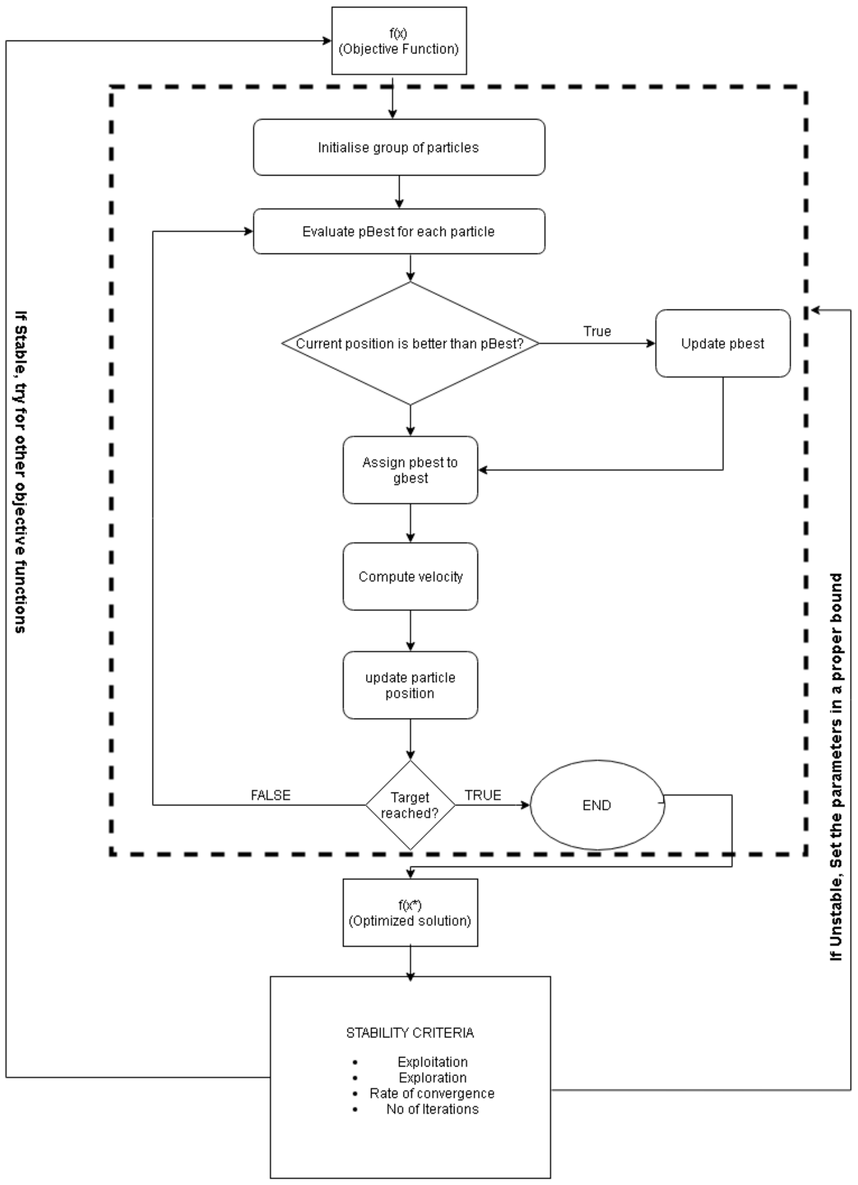


Figure 2.3: Flowchart with Stability analysis of PSO

2.3 Functional Block Diagram

The following is the block diagram of stability analysis of Particle Swarm Optimisation.

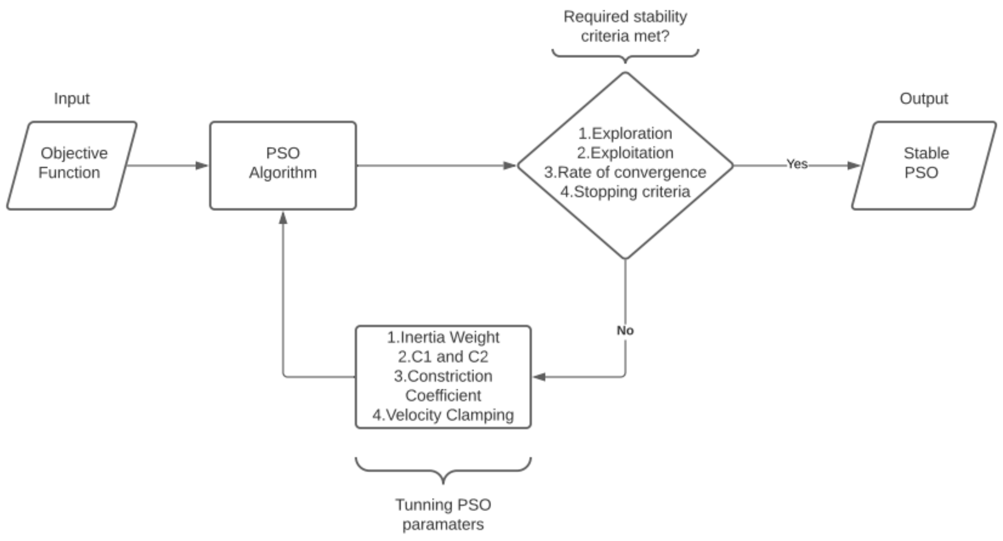


Figure 2.4: Functional block diagram of PSO

From figure 2.4 the input might be any objective function that has to be optimised. The Particle Swarm Algorithm is used to optimise the objective function. When we receive an output from PSO, we examine it for stability using three or four stability criteria, which include exploration, exploitation, rate of convergence, and stopping criteria. We consider the algorithm to be stable if all of the stability conditions have been satisfied. Otherwise, we tune the parameters of PSO using concepts such as constriction co-efficient, velocity clamping, inertia weight, and acceleration co-efficients.

Using the equation (2.1) and (2.2), we update the particle's velocity and position. When the position is updated, we compare it to the particle's personal best position. If the current position is better than the personal best, the personal best position is updated. After updating the personal best, we compare the personal best value to the global best and update the global best.

Repeat this process for as many iterations as necessary until the stopping requirement is reached. When the stopping criteria's are met, we exit the PSO and examine the output to see if it is stable using stability criteria such as Exploration, Exploitation, and Rate of convergence. If the stability condition is not met, we adjust the PSO parameters using the constriction co-efficient, inertia weight, and velocity clamping.

Inertia Weight 'ω'':

To control the balance between global and local exploration, produce rapid convergence, and reach an optimum, the inertia weight, whose value falls linearly with iteration number, is set according to the following equation. The inertia weight can be implemented as a constant amount or as a variable number. Initially, all particles' inertia values were fixed throughout the process, but since this parameter influences the exploration and exploitation of the search space, dynamically changing inertia values are now used. The huge inertia value is often high at first, allowing all particles to easily travel in the search space, but it steadily decreases over time. As a result, the procedure is moving from the exploratory to the exploitative stage. The inertia weight, represented by, is assumed to replace by managing the influence of prior velocities in the process, i.e. by weighing the contribution of previous velocity to the particle's momentum.

Constriction Co-efficient 'χ':

This coefficient is critical for controlling the exploration and exploitation trade-off, ensuring convergence behaviour, and excluding the inertia weight and maximum velocity.The constriction co-efficient in the form of equation can be written as

$$\chi = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \phi = \phi 1 + \phi 2 \tag{2.3}$$

Here in the equation (2.3), $\phi_1 = r1(t)$ c1 and $\phi_2 = r2(t)$ c2.

2.4 Software Requirements

2.4.1 MATLAB

The MATLAB programming language is a powerful tool for creating scientific workflows. Calculation, visualisation, and programming are all combined in a user-friendly environment, with problems and answers presented in familiar mathematical language. Because Matlab is widely used in the automobile industry, it was picked. The PSO algorithm was created and implemented in Matlab, and the output was tested for a variety of parameters.

Chapter 3

Optimisation

3.1 Introduction to Optimisation

The goal of optimization is to create the "best" design possible with the resources available. The purpose is to increase productivity, strength, dependability, efficiency, and usage. Code that is to be run on an embedded system must be optimised so that it uses all of the available resources while also meeting all of the requirements. Using optimization techniques has several advantages, including efficient use of embedded system resources, the ability to improve existing functions, the ability to add functionalities to upgrade systems, and the ability to reduce embedded system power consumption. As a result, during the design of embedded systems, the factor of optimization plays a critical role.

3.2 Types of Optimisation

1. The conditional statements are used to achieve code specification optimization.
2. Expression simplification optimization technique is achieved concerning
   (a) Constant folding.
   (b) Algebraic expression, for example, taking variable common instead of having it multiple times in an equation.
   (c) Strength reduction like instead of multiplying a constant to a variable we can shift it left by making so we can increase the speed of operation.
   (d) Dead code execution, dead-code means a part of code that is never executed but will be a part of the code.
3. In-lining a procedure is accomplished by creating a statement rather than a function. This reduces time by eliminating process linkage overhead.
4. Loop unrolling is a loop transformation technique that aids in reducing a program's execution time. Iterations are removed or reduced. By removing loop control and loop test instructions, loop unrolling boosts the program's speed.
5. Two successful loop transformation approaches for optimising programme execution are loop distribution and loop fusion. Loop fusion (also known as loop jamming) is a loop transformation and compiler optimization that replaces many loops with a single one. When two loops iterate over the same range without referencing each other's data, this is conceivable.

3.3 Selection and Justification of Optimization Method

We chose the code specification optimization technique, which includes 'if-else ladders' and 'for loops', from the types of optimization techniques mentioned in section 3.2. If statements are run from top to bottom. The statement associated with 'if' is executed as soon as the condition controlling that particular 'if' is 'true,' and the rest of the else-if ladder is skipped. A 'for' loop is used to repeat a snippet of code a set number of times until the loop's condition becomes false.

We used functions to avoid rewriting the same logic/code throughout the programme. We called functions a certain number of times and in different places throughout the programme. Because our large programme is divided into functions, it is easy to track.

Chapter 4

Results and Discussions

By tuning different parameters, the PSO algorithm was implemented and checked for different stability criteria, such as Exploration, Exploitation, and Rate of Convergence. The results of tuning the parameters are discussed in the following section.

4.1 Result Analysis of PSO

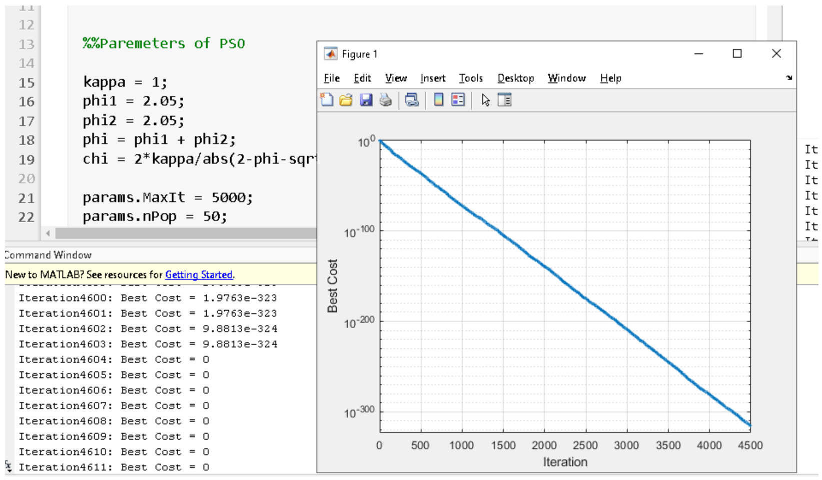
As previously mentioned, while working on stability criteria the team came across two concepts: Constriction co-efficient and damping ratio. Each of the two concepts separately were used to see what each one had an effect on. Starting from:

4.1.1 Constriction Co-efficient χ with Dependency of ϕ_1 and ϕ_2:

The following conditions can be shown along with the different graphs got using constriction coefficient.The semi log graph of Best cost v/s Number of iterations is plotted in figures.

Keeping [ϕ_1= ϕ_2 = 2.05] :

By keeping the values of ϕ_1 and ϕ_2 constant and equal to 2.05 we get the following graph which shows the slow rate of convergence and more number of iterations.



```
%%Paremeters of PSO

kappa = 1;
phi1 = 2.05;
phi2 = 2.05;
phi = phi1 + phi2;
chi = 2*kappa/abs(2-phi-sqr

params.MaxIt = 5000;
params.nPop = 50;
```

Command Window

New to MATLAB? See resources for Getting Started.

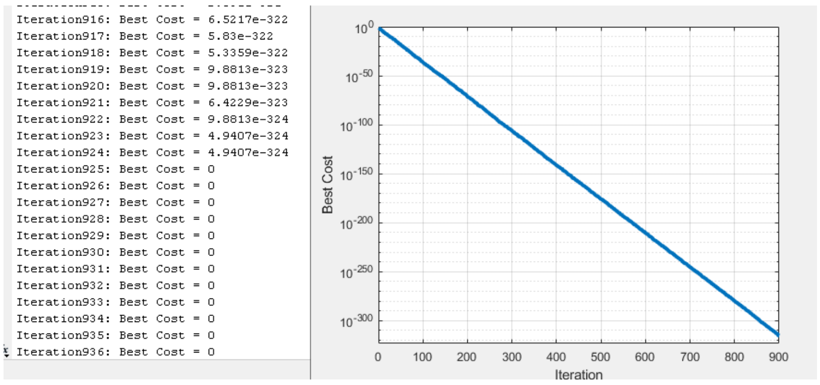
```
Iteration4600: Best Cost = 1.9763e-323
Iteration4601: Best Cost = 1.9763e-323
Iteration4602: Best Cost = 9.8813e-324
Iteration4603: Best Cost = 9.8813e-324
Iteration4604: Best Cost = 0
Iteration4605: Best Cost = 0
Iteration4606: Best Cost = 0
Iteration4607: Best Cost = 0
Iteration4608: Best Cost = 0
Iteration4609: Best Cost = 0
Iteration4610: Best Cost = 0
Iteration4611: Best Cost = 0
```

Figure 4.1: Plot by Keeping [ϕ_1= ϕ_2 = 2.05]

From figure 4.1. When the constriction coefficient is taken into account and when the ϕ_1 and ϕ_2 are kept constant , the number of iterations increases. As a result, we concluded that the constriction coefficient focuses primarily on Exploration, which increases number of iterations.

Increasing ϕ_1 and ϕ_2 :

By increasing the value of ϕ_1 and ϕ_2 we get the following graph which shows the rate of convergence and number of iterations.



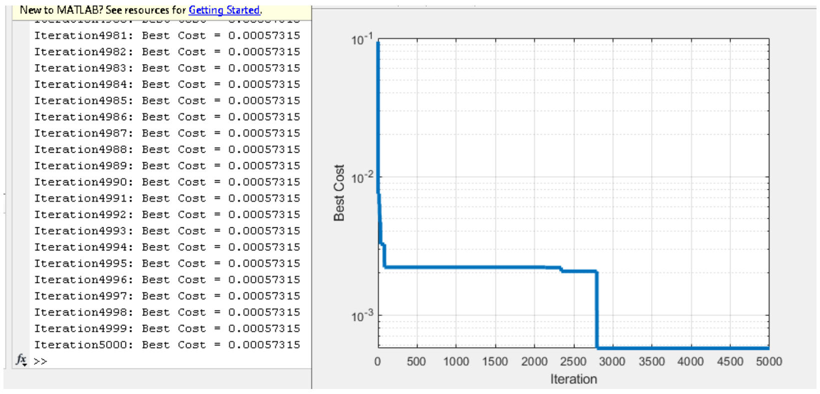
```
Iteration916: Best Cost = 6.5217e-322
Iteration917: Best Cost = 5.83e-322
Iteration918: Best Cost = 5.3359e-322
Iteration919: Best Cost = 9.8813e-323
Iteration920: Best Cost = 9.8813e-323
Iteration921: Best Cost = 6.4229e-323
Iteration922: Best Cost = 9.8813e-324
Iteration923: Best Cost = 4.9407e-324
Iteration924: Best Cost = 4.9407e-324
Iteration925: Best Cost = 0
Iteration926: Best Cost = 0
Iteration927: Best Cost = 0
Iteration928: Best Cost = 0
Iteration929: Best Cost = 0
Iteration930: Best Cost = 0
Iteration931: Best Cost = 0
Iteration932: Best Cost = 0
Iteration933: Best Cost = 0
Iteration934: Best Cost = 0
Iteration935: Best Cost = 0
Iteration936: Best Cost = 0
```

Figure 4.2: Plot by Increasing ϕ_1 and ϕ_2

As can be seen from the preceding formula (2.3) , as ϕ increases the χ decreases and primarily affects exploration and exploitation, it decreases, resulting in a high rate of convergence.

Decreasing ϕ_1 and ϕ_2 :

By decreasing the value of ϕ_1 and ϕ_2 we get the following graph which shows the rate of convergence and number of iterations.



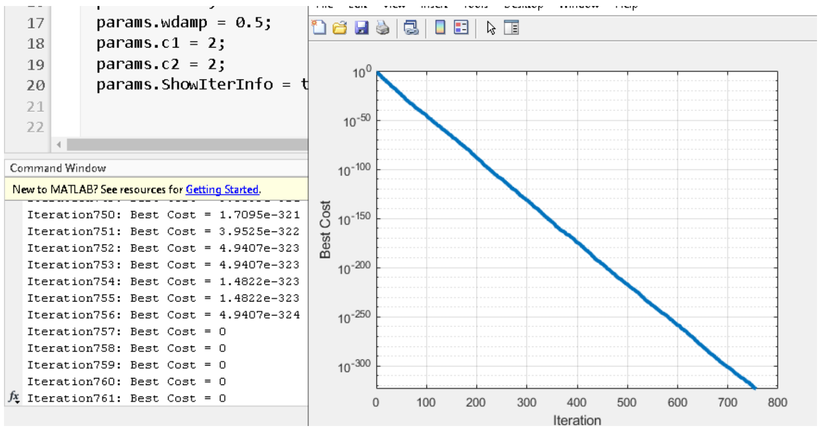
```
New to MATLAB? See resources for Getting Started.
Iteration4981: Best Cost = 0.00057315
Iteration4982: Best Cost = 0.00057315
Iteration4983: Best Cost = 0.00057315
Iteration4984: Best Cost = 0.00057315
Iteration4985: Best Cost = 0.00057315
Iteration4986: Best Cost = 0.00057315
Iteration4987: Best Cost = 0.00057315
Iteration4988: Best Cost = 0.00057315
Iteration4989: Best Cost = 0.00057315
Iteration4990: Best Cost = 0.00057315
Iteration4991: Best Cost = 0.00057315
Iteration4992: Best Cost = 0.00057315
Iteration4993: Best Cost = 0.00057315
Iteration4994: Best Cost = 0.00057315
Iteration4995: Best Cost = 0.00057315
Iteration4996: Best Cost = 0.00057315
Iteration4997: Best Cost = 0.00057315
Iteration4998: Best Cost = 0.00057315
Iteration4999: Best Cost = 0.00057315
Iteration5000: Best Cost = 0.00057315
fx >>
```

Figure 4.3: Plot by Decreasing ϕ_1 and ϕ_2

As from figure 4.3 we see that when we decrease ϕ_1 and ϕ_2 the constriction coefficient increases which in turn increases the exploration and cause all the particles to spiral toward and around the best solution in the searching space without any assurance of convergence.

4.1.2 Using the Damping Ratio $[\omega_{damp}]$:

Here damping ratio is used along with constriction co-efficient.



```
params.wdamp = 0.5;
params.c1 = 2;
params.c2 = 2;
params.ShowIterInfo = t
```

Command Window

New to MATLAB? See resources for Getting Started.

```
Iteration750: Best Cost = 1.7095e-321
Iteration751: Best Cost = 3.9525e-322
Iteration752: Best Cost = 4.9407e-323
Iteration753: Best Cost = 4.9407e-323
Iteration754: Best Cost = 1.4822e-323
Iteration755: Best Cost = 1.4822e-323
Iteration756: Best Cost = 4.9407e-324
Iteration757: Best Cost = 0
Iteration758: Best Cost = 0
Iteration759: Best Cost = 0
Iteration760: Best Cost = 0
Iteration761: Best Cost = 0
```

Figure 4.4: Plot by Using Damping R atio $[\omega_{damp}]$

The rate of convergence can be seen in the graph 4.4 shown above. The number of iterations reduces when the constriction coefficient is not taken into account. As a result, we came to the conclusion that the constriction coefficient is largely concerned with rate of convergence and not exploration, which reduces the number of iterations required.

Chapter 5

Conclusions and Future Scope

This chapter summarises the research's findings as well as the accomplishments made as a result of it. The acquired results are reviewed in detail, as well as future work that may be done to increase the quality of the summary.

5.1 Conclusion

The parameters of Particle Swarm Optimization were tuned to maintain stability criteria like Exploration, Exploitation, and Convergence Rate. We made some observations using concepts such as constriction co-efficient, velocity clamping, and damping ratio and came to a conclusion. We used particle swarm optimization (PSO) for two fitness functions in this study.

1. Rastrigin's function
2. Function of a sphere

The algorithm will try to discover the minimal value of these fitness functions for a predetermined number of maximum iterations. The speed/rate of convergence can be seen and how better is the exploration can be understood. Both the functions provided the same results and some observation where made which is shown in the below table.

Table 5.1: Table of observation for different parameters

Parameters	Condition	Result	Observations
Constriction Co-efficient (χ)	Using the concept of constriction co-efficient (χ)	1.Slow rate of convergence 2.Higher Exploration 3.More number of iterations	Since there is slow rate of convergence and it is taking more iterations to find the optimal value, the exploration is good
Damping Ratio (ω_{damp})	Decreasing the damping ratio (ω_{damp})	1.Fast rate of convergence 2.Poor exploration	The standard value for damping ratio should be 0.9
Population of particles	Increasing the population size	1.Faster rate of convergence 2.Time taken to run the code increases	As the population increases, the search rate increases leading to faster convergence
ϕ_1 and ϕ_2	Increasing the value of ϕ_1 and ϕ_2	1.χ decreases 2.Faster rate of convergence 3.Poor exploration	Since ϕ is inversely proportional to χ, the exploration decreases
ϕ_1 and ϕ_1	Decreasing the value of ϕ_2 and ϕ_2	1.χ increases 2.Good Exploration 3.Slow rate of convergence	χ increases which in tun increases the exploration which causes all particles to slowly spiral toward and around the best solution in the searching space without convergence guaranteed.

5.2 Future Scope

PSO may be used to solve a variety of optimization issues. It is most useful for determining the maximum or minimum of a function specified in a multidimensional vector space. PSO is used to tune the parameters of any controller in any system. Because we are focusing on PSO stability, the parameters of every controller that will be trained will be stable, resulting in the lowest possible error in any closed loop. We will continue to work on employing reliable PSO in controllers and attempting to make the controller stable.

References

[1] Mario Arioli, Iain Duff, and Daniel Ruiz. Stopping criteria for iterative solvers. *SIAM Journal on Matrix Analysis and Applications*, 13(1):138–144, 1992.

[2] Shivam Chaturvedi, Pallavi Pragya, and HK Verma. Comparative analysis of particle swarm optimization, genetic algorithm and krill herd algorithm. In *2015 International Conference on Computer, Communication and Control (IC4)*, pages 1–7. IEEE, 2015.

[3] Seyedali Mirjalili. Ant colony optimisation. In *Evolutionary Algorithms and Neural Networks*, pages 33–42. Springer, 2019.

[4] Seyedali Mirjalili. Genetic algorithm. In *Evolutionary algorithms and neural networks*, pages 43–55. Springer, 2019.

[5] Janmenjoy Nayak, Bighnaraj Naik, Asit Kumar Das, and Danilo Pelusi. Nature inspired optimization and its application to engineering, 2021.

[6] Matthew Settles. An introduction to particle swarm optimization. *Department of Computer Science, University of Idaho*, 2:12, 2005.

[7] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[8] Satyobroto Talukder. Mathematicle modelling and applications of particle swarm optimization, 2011.

[9] Xin-She Yang. Nature-inspired optimization algorithms: Challenges and open problems. *Journal of Computational Science*, 46:101104, 2020.

[10] Haohao Zhou and Xiangzhi Wei. Particle swarm optimization based on a novel evaluation of diversity. *Algorithms*, 14(2):29, 2021.