SWARM INTELLIGENCE APPLICATION IN SOLVING ROBOT INVERSE KINEMATICS PROBLEMS

by

Hasan Danaci, B.S.

THESIS

Presented to the Faculty of

The University of Houston-Clear Lake

In Partial Fulfillment

Of the Requirements

For the Degree

MASTER OF SCIENCE

in Computer Engineering

THE UNIVERSITY OF HOUSTON-CLEAR LAKE

AUGUST, 2021

SWARM INTELLIGENCE APPLICATION IN SOLVING ROBOT INVERSE KINEMATICS PROBLEMS

by

Hasan Danaci

APPROVED BY

Luong Nguyen, PhD, Chair

Hakduran Koc, PhD, Committee Member

Thomas L. Harman, PhD, Committee Member

RECEIVED/APPROVED BY THE COLLEGE OF SCIENCE AND ENGINEERING:

David Garrison, PhD, Interim Associate Dean

Miguel A. Gonzalez, PhD, Dean

Dedication

First and foremost, I am incredibly grateful to my supervisor, Dr. Luong NGUYEN, for his invaluable advice, continuous support, guidance, and patience during my thesis. His immense knowledge and plentiful experience have encouraged me in my academic research and daily life. I would also like to thank Dr. Thomas L. HARMAN and Dr. Hakduran KOC for their support. Furthermore, I would like to thank all for being members of the thesis committee. Their kind help and support have made my study and life in Houston a wonderful time. Finally, I would like to express my gratitude to my parents Gulden DANACI, Hayat DANACI, and my live advisor Yunus TEMELLI who taught me umpteen things. Without their tremendous understanding and encouragement in the past years, it would be impossible for me to complete my study. Without their love, support, motivation, and help, I would have been at a pretty different place in life.

Acknowledgments

First, I would like to send my sincere thanks to my thesis advisor, **PhD Luong Nguyen,** for the continual assistance with my master's research project. His guidance, motivation, patience, and extensive knowledge helped me to achieve this research project. Especially, his experience with robotics kept me motivated, despite the glitches I had through the research. Hence, we accomplished this research that has significant importance.

Besides my advisor, I would like to acknowledge my committee, Director of Center for Robotics Software, Department Chair of Engineering, Professor of Computer Engineering, **PhD Thomas L. Harman** and Program Chair and Associate Professor of Computer Engineering, College of Science and Engineering **PhD Hakduran Koc** for their encouragement, comments, hard questions and all things that I learned from them during their classes.

ABSTRACT

SWARM INTELLIGENCE APPLICATION IN SOLVING ROBOT INVERSE KINEMATICS PROBLEMS

Hasan Danaci
University of Houston-Clear Lake, 2021

Thesis Chair: Luong Nguyen, PhD

This dissertation aims to find the inverse kinematics(IK) solution for redundant serial manipulators using the meta-heuristic method, Particle Swarm Optimization algorithm (PSO). Primarily this paper focuses on moving the end-effector to any desired pose in cartesian space accurately by converging position and orientation with the PSO algorithm. In order to demonstrate the study's accuracy, the results were compared to some previous PSO studies that just examined converging the position. All demonstrations were performed using a humanoid human-sized with 7 degrees of freedom robot (DOF), the Research Baxter Robot of the University of Houston Clear Lake (UHCL) Robotics and Control Laboratory. First, the Denavit-Hartenberg(DH) table of Baxter's left arm is created. Then, transformation matrices are calculated to calculate target position and orientation values according to several sets of joint angles.

Furthermore, joint angles are picked randomly for each particle, and the particles' pose is calculated by applying the forward kinematics formula. In order to obtain subsequent angle values, the PSO algorithm, conversion of quaternion to a rotation matrix, and Jacobian matrices are utilized. This research gives another perspective to solving inverse kinematics by using quaternions instead of Euler angles. The Euclidian function is used to compute the cost function, which estimates the distance between the target pose and the particles' pose. In this study, the algorithm is tested with several different concepts. Conclusively, the validity of the algorithm is verified via Gazebo simulation. The result confirms that the algorithm functions well in accuracy and merit of the swarm intelligence in solving the inverse kinematics problem for any serial robotic manipulators.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I:

INTRODUCTION

Background

These days, due to their dynamic and kinematic function, redundant serial manipulators draw both specialized researchers and industry leaders alike in the robot marketplace. Especially after the pandemic, robotic is captured companies' attention highly. Inverse kinematics has a significant role in robotics. This study focuses on solving IK, which is one of the most critical processes in robotics. IK uses the kinematics equations to find the joint settings that give each robot's end-effector the required configuration (position and rotation) [1]. The kinematic design, which is considered in various parts, particularly as inverse and forward kinematics, refers to a scientific study that explores robot movements. However, forward kinematics, which works to decide the end effector's status away from the joint directions, is comparatively easy.

On the other hand, the solution of the IK problem is computationally expensive and generally takes a very long time in real-time. Furthermore, singularities and nonlinearities complicate IK. As a result, numerous heuristic approaches were considered more appropriate for solving IK [2].

In recent years, optimization approaches such as artificial neural networks, particle swarm optimization, genetic algorithms, artificial bee colonies, harmony search, and the firefly algorithm have gained much traction among researchers [3]. As a result, we can encounter several research projects that use meta-heuristic approaches to solve IK and many other problems. Our method for solving IK brings an unusual perspective and a more reliable solution. We use iterated PSO velocity values in Jacobian to find updated joint angle values. Therefore, in this study, we prove how swarm intelligence methods such as particle swarm optimization are capable of furnishing an accurate solution orientally and positionally with the way we use in this research.

Related Works

Inverse kinematics is a fundamental challenge in robotics, requiring the calculation of a set of joint angles to operate the robot to the desired end-effector position and orientation**.** There are numerical methods such as Newton-Raphson, Jacobian inverse, analytical and other techniques using swarm intelligence developed for solving IK. However, recent investigations reveal that metaheuristic techniques for solving serial manipulator IK problems have piqued the interest of many researchers.

In 2019, Authors Dereli & Köker [58] focused on the IK solution for 7-DOF serial robotic manipulators using quantum-behaved particle swarm optimization (QPSO). Besides, they compared with some other swarm optimization methods such as Artificial Bee Colony(ABC), Firefly algorithm(FA). Although the IK aims to manipulate the end-effector to the desired pose, consisting of orientation and position, Dereli & Köker derived just position values using forward kinematics. In order to move the end-effector to an attempted location in cartesian space as accurately as possible, orientation and position must be the same. A similar endeavor was made by Huang[65] in 2012 via PSO in solving IK of 7-DOF robotic. The author's results were more adaptive and robust. The research resulted in hybridization, which is mainly inspired by the social behavior of an individual. The same results and highly supporting facts are also shown by Durmus et al. [66] on their investigation for an IK solution using particle swarm optimization. As per their results, even though slightly different from the other authors, the results must be used effectively for better performance. Mustafa Ayyildiz et al. [59] also proposed the comparison of four different optimization methods, Genetic Algorithm(GA), PSO, gravitational search algorithm (GSA), and QPSO for solving IK of real 4-DOF robot positionally. One great example was demonstrated by Sancaktar et al. [60] in 2018. They adopted the PSO algorithm for the IK solution of the 6-DOF robot built for fracture therapy with an external fixator. The classic PSO algorithm employed ensured that all the particles were channeled to the searched target. They used a finite number of iterations until a specified criterion was achieved. Rokbani [60] solved the IK of a 3-DOF biped robot during locomotion by using PSO in 2012. According to Rokbani [61], PSO for solving inverse kinematics delivers an IK solution that corresponds to the robot's Center Of Mass while maintaining joint restrictions. Collinsm and Shen [62] proposed that the stop criterion relies on achieving the optimal number of iterations or the fitness function. The swarm optimization procedure (IK-PSO) stops after the maximum possible iterations are attended. It can also stop after an error detection towards the target position appears to be less than the fixed amount. Hence, they solved the problem orientally for any DOF manipulator.

Swarm intelligence comprises numerous optimization techniques. Hence, other methods were proved that are applicable for solving IK. Starke et al. [63] implemented the Genetic Algorithm (GA) technique to settle and solve IK on arbitrary joint chains. The approach was applicable since it ensures high accuracy and sure success rates can thoroughly pose the objective's solution finding in real-time. To achieve this, Starke et al. [63] incorporated joint constraints.

Furthermore, evolutionary optimization ensured that the problem was solved since the genetic algorithms were merged with those relating to swarming intelligence. Finally, to solve the problem positionally, Starke optimized the problem to get the best parameter vector for reducing the objective function. Hsu-Chih Huang et al. [64] presented deoxyribonucleic acid (DNA) swarm intelligence to solve the IK redundancy problem of six-degree-of-freedom (DOF) humanoid robot arms.

Most of the earlier relevant works mainly interpret solving IK only positionally for robotic manipulators with seven and fewer DOF. This paper focuses on solving the complete (position and orientation) IK of any robotic manipulator. Quaternions will be used for the orientation part of the solution.

CHAPTER II:

SWARM INTELLIGENCE

Swarm Intelligence (SI) is the collective behavior of decentralized, self-organized systems, whether natural or artificial. The idea is used in artificial intelligence research. In 1989, Gerardo Beni and Jing Wang used the concept in the context of cellular robotic systems [4]. SI is composed of a simple group of agents that interact locally in the same environment. Nature, particularly biological systems, is a frequent source of inspiration. The discipline focuses on the collective behaviors that emerge from individuals' local interactions with one another and their surroundings [5]. There are certain instances of swarm intelligence such as ant colonies and other social insect societies, hawks hunting, birds flocking, sheep herding, fish schooling, microbial intelligence, and bacterial growth.

The ability of a swarm intelligence system to function in a coordinated manner without the existence of a coordinator or an external controller is its distinguishing feature. Many examples of swarms doing collective behavior without any individual controlling or being aware of the overall group behavior can be seen in nature [5].

Even though SI was first established more than 30 years ago, it is being studied, expanded, and used in various scientific and engineering studies. As a result, swarm intelligence became a popular way for solving one of the fundamental engineering problems such as robotics. Hence, numerous swarm behaviors observed in natural systems have inspired novel approaches to problem-solving employing robot swarms. This dissertation utilized a population-based stochastic optimization technique, Particle Swarm Optimization (PSO), to solve any robotic manipulator IK problem.

Methods of Swarm Intelligence

Particle Swarm Optimization

Particle swarm optimization (PSO) is a computational algorithm that solves various problems by improving a candidate solution iteratively regarding a provided quality measure [6]. It solves a problem by moving dubbed particles throughout the search space according to a simple mathematical formula for the particle's position and velocity. In order to move all the particles toward the best solutions, each particle uses its local best-known position and global best-known positions that are updated by other particles as a better position. Particle Swarm Optimization is a population-based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995 [7], inspired by birds' social behavior.

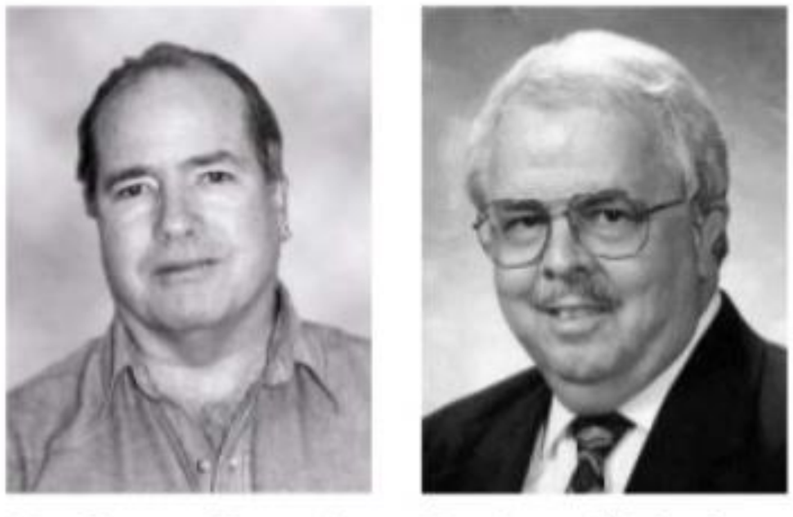


Figure 1:
James Kennedy – Russel Elbert

The origins of particle swarm optimization may be traced back to two key component techniques. Its linkages to artificial life (A-life) in general, bird flocking, fish schooling, and swarming theory [8]. The basic version of particle swarm optimization is based on a relatively simple concept, and therefore may be implemented in just a few lines of computer code. It merely uses primitive mathematical operations and is computationally inexpensive in terms of both memory and speed. The basic variation of the PSO algorithm employs a population of candidate solutions (called particles). A few simple equations (Equ 2.1 and Equ. 2.2) are used to move these particles about in the search space [9]. The particles' movements are controlled by their individual best-known position (Fitness) in the search space, as well as the best-known ($Gbest$) position of the entire swarm. When better positions are located, they will be used to steer the swarm's movements. The procedure is being repeated until either all iterations are completed, or $Gbest$ meets the solution's error criterion that satisfying the solution. The original PSO algorithm is as follows.

Equation (2.1)

$$\boldsymbol{V_{ij}^{t+1} = V_{ij}^{t} + c_1^t r_1^t \left(Pbest_{ij} - X_{ij}^{t}\right) + c_2^t r_2^t \left(Gbest - X_{ij}^{t}\right)}$$

Equation (2.2)

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1}$$

A particle in PSO can be defined as, $P_i \in [a\ b\ ]$ where j=1,2,3… is for dimensions , i is for particle numbers [10]. Each particle has its velocity and position, which are randomly initialized at the start. Each particle must maintain its positions $Pbest$ known as the local best position and the $Gbest$ known as the global best position among all the particles. Where v_i is the velocity, x_i is the position $Pbest$ is the personal best position of the particle, and $Gbest$ is the global best position for the PSO. $r_1\ and\ r_2$, are two random numbers of ranges [0, 1) and $c_1\ and\ c_2$ are the leaning factors.

The pseudo-code of the original PSO is:

```
Initialize the population randomly
**While** (Population Size)
        {
        **Loop**
        Calculate fitness
        **If** fitness value is better than the best fitness value ($Pbest$), Then Update $Pbest$
        End Loop
        }
        Select Global Best ( $Gbest$) from all particles' best fitness value
        **While** (max iterations or min error criteria is not attained)
        {
        Calculate particle best velocity with equation 2.1
        Update particle position according to equation 2.2
        }
```

Although the fundamental PSO algorithm is an outstanding tool for solving optimization problems, it still has the issue of being locked in local minima. The researchers proposed various PSO versions to increase the performance of PSO. Some academics attempted to improve it by optimizing the swarm's initialization. New parameters such as the constriction coefficient and inertia weight are introduced in some of them. Some academics specify the various approaches of inertia weighting in order to increase PSO performance. Some researchers use the mutation operators in the PSO to work on the global and local best particles. A set of standard executions has been devised, with the goal of showcasing PSO to a larger optimization community and evaluating the method's performance improvements. Having a reliable, well-defined benchmark methodology provides a useful point of comparison that may be used across the research field to test the subsequent advances.

Inertia Weight

Shi and Eberhart [11] developed Inertia Weight in 1998 to better regulate exploration and exploitation. The goal was to do away with the necessity for velocity range (-Vmax, +Vmax). As a result, the velocity update equation is defined as Equation 4.1. Because beginning velocity has such a massive impact on the balancing of the swarm's exploration and exploitation processes, Inertia Weight (w) is utilized to adjust the velocity. The Inertia Weight (Equation 4.1) plays a vital function in balancing the exploration and exploitation processes. The particle's initial velocity contribution to its velocity at the current time step is determined by the Inertia Weight. There is no Inertia Weight in the fundamental PSO described by Eberhart and Kennedy in 1995 [8]. Shi and Eberhart [11] introduced Constant Inertia Weight in 1998, which was the first time the notion of inertia weight was introduced. They claim that a large Inertia Weight makes a global search easier, whereas a small Inertia Weight makes a local search easier.

Eberhart and Shi [11] suggested a Random Inertia Weight method, which they found to effectively improve PSO convergence in early iterations of the algorithm. PSO's efficiency and performance are improved by using the Linearly Decreasing approach [12]. Experiments have shown that inertia weights ranging from 0.9 to 0.4 produce outstanding results. Despite its ability to converge to an optimal solution, it ends up in the local optimum when increasing apices [13]. Random inertia weight as follows:

Equation (2.3)

$$w = 0.5 + \frac{rand()}{2}$$

Fayek et al. [14] present an optimized Particle Swarm technique (PSOSA) that employs Simulated Annealing to optimize the Inertia Weight, and they evaluated the method on an urban planning problem. The proposed methodology performs significantly better in terms of convergence speed and long-term stability as the number of blocks to be fitted in the urban planning problem grows. A Sigmoid Increasing Inertia Weight was presented by Malik et al. [15]. They discovered that the sigmoid function helped them get the lowest fitness function, while Linearly Increasing Inertia Weight helped them achieve rapid convergence.

Various sorts of inertia weight techniques are introduced to handle various challenges to improve the PSO algorithm's effectiveness. In this study, we employ constant inertia weight and damping inertia weight with a constant value at each iteration until it falls below a particular value set in the program to solve any IK problems. Other inertia weight techniques could be investigated for comparison in future research.

Hybridization

In order to improve optimization output, first-hand and upgraded PSO variations are frequently offered. In such a study, there are clear trends or advancements. One of these ideas is to combine PSO with other optimizers to develop a hybrid optimization approach. In 2013, hybridization of particle swarm optimization with adaptive genetic algorithm (GPSO) operators was carried out by Masrom, S. et al. [16].

Alleviate Premature Point of Convergence

Another research trend is to use multidimensional swarms to research with and improve premature convergence spots by blocking or reversing the PSO particles' movement and coping with local minimum. Multi-aim optimization can also be done using various swarm techniques.

After carefully examining the PSO variant's operations, Zhan et al. [17] concluded that there are trends in altering PSO's attitudinal behavioral standards while optimizing.

Simplifications

Another viewpoint is that PSO should be simplified to the greatest extent possible without compromising its results. The Occam's razor is the name given to this concept. According to Kennedy [18], it appears that optimization output was improved, standards were easier to alter, and they performed more consistently along with various optimization tasks. PSO is a metaheuristic that may or may not be correct, which increases the risk of making mistakes in its depiction and execution. A notable example is Zhenguo Tu's work, which was later found to be defective because it was strongly prejudiced in its optimization search., However, he stated that the partiality was caused by a programming error that has been corrected [19]. Additional inputs may be required to jumpstart the speeds. The initiation of these speeds does not require any speed, according to the Bare Bones PSO Variant proposed by Kennedy James 2013 [20]. Another simpler variant is the Accelerated PSO (APSO), which does not require speed and can improve convergence in various applications. Also, a different easier variation is the Accelerated PSO (APSO), which does not need to use speed and can up the convergence in several usages.

Discrete

Because of PSO's tremendous efficiency, there has been much effort put towards expanding continuous PSO to discrete PSO. The binary PSO (BPSO) algorithm described by Kennedy and Eberhart [23] based on the binary coding scheme was the first test. Later, the angle modulated PSO, and the discrete multiphase PSO was used to improve the method. The mapping of the sequential space into discrete form is one direct technique to convert a continuous PSO to a discrete one. Many discrete PSO algorithms defined by space transformation techniques have been proposed based on this concept. Other approaches to redefine a particle's position and velocity, such as fuzzy matrix-based, swap-operator-based, crisp set-based, and so on, have been proposed in the literature. Although numerous discrete PSO algorithms have been suggested, their performance when applied to the community detection issue is often unsatisfactory[24].

Other Population-Based Swarm Intelligence Algorithms

Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is a possible solution for problems involving discovering better routes via graphs that Dorigo proposed in 1992. When artificial ants, who serve as simulation agents, travel through a limited space that displays all possible solutions, they discover optimum solutions. As they explore their environment, natural ants produce pheromones that lead them to paths. Marco [25] noticed that his artificial ants keep track of their locations and the worth of their results in the same way as real ants do so that in future simulations, more ants can find them and get higher outcomes. Many similarities exist between PSO and ACO. Both start with a randomly generated population and update the solution every generation based on the fitness values. ACO and PSO, on the other hand, lack evolution operators such as crossover and mutation. Both, however, have memory, which is critical to the algorithms [26].

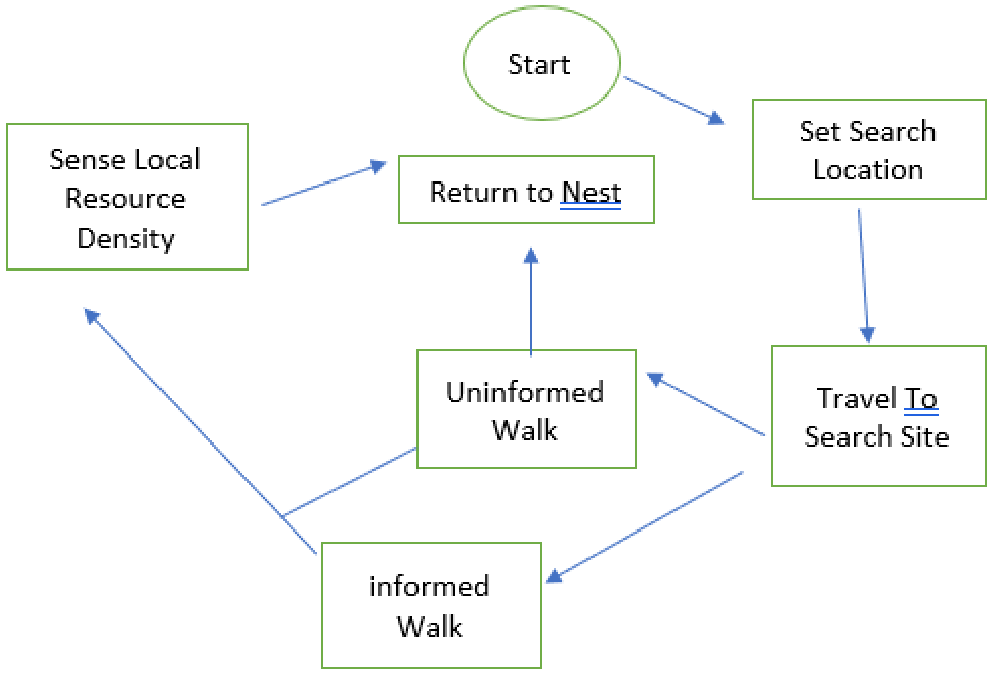


Figure 2:
State Diagram of Searching Food

Genetic Algorithm

Genetic algorithm **(**GA**)** is an inspired metaheuristic method by Darwin's theory, the process of natural selection that belongs to the larger group of evolutionary algorithms (EA) in computer science. Genetic algorithms generate high-quality solutions to optimization problems using biologically inspired operators like mutation, crossover, and selection [27]. In order to develop better solutions for an optimization problem, a Genetic algorithm uses the power of a population which has multiply candidate solutions (individuals, creatures, or phenotypes). Each candidate solution consists of a set of parameters (its chromosomes or genotype) represented in the binary numerical system as strings of 1s and 0s, but other encodings are also possible [28].

The evolution usually begins from a population of haphazardly generated candidates, and the evolution is an iterative cycle, with the population in each iteration called a *generation*. In an optimization problem, fitness is generally the value of the objective function. The fittest individuals are randomly chosen from the present population, and their genomes are transformed (recombined and maybe randomly altered) to create a new generation. In the following iteration of the algorithm, the new generation of candidate solutions is employed. Typically, the algorithm ends after a certain number of generations have been created, or the population has attained a suitable fitness level. [29] [30]. Five phases are considered in the basic genetic algorithm.

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

Bees Algorithm

The bees algorithm is a populace-based pursuit calculation created by Pham, Ghanbarzadeh et al. in 2005. The bees algorithm was inspired by the foraging behavior of honey bee colonies' food[31]. The basic algorithms can perform both combinational optimization and continuous optimization by executing a sort of neighborhood exploration. Also, the bees algorithm has proved its effectiveness and abilities in solving various problems. [31] [32] [33] [34]

Pseudo-code of the bees algorithm as it follows.

1. Initialize population with random solutions.
2. Evaluate the fitness of the population.
3. While (stopping criterion not met) / Forming new population.
4. Select sites for neighborhood search.
5. Determine the patch size.
6. Recruit bees for selected sites (more bees for best e sites) and evaluate fitness.

```
7. Select the fittest bee from each patch.
8. Abandon sites without new information.
9. Assign remaining bees to search randomly and evaluate their fitness.
10. End While
```

Usages of Swarm Intelligence

Swarm intelligence-oriented methods may be utilized at several points in a variety of applications. Some of these are briefly expatiated upon below.

The General Application of Swarm Intelligence

These are tested and proved via diverse modes. They are as follows:

Ant-oriented Routing

In the telecoms industry, in the formula of ant-oriented routing, SI usage has also been researched. Simply speaking, this routing uses a probable routing table that reinforces the direction effectively traversed by every ant that floods the network. The reinforcement of the route in the reverse direction, forwards, and both concurrently have been examined. A backward reinforcement demands a symmetric network and brings together the two routes. On the other hand, a forward reinforcement recompenses a direction before the result is identified. As the system performs stochastically and cannot repeat itself, Whitaker [35] foresaw that there will always be considerable obstacles to commercial usage. Courtesy of SI, new technologies and mobile media can alter the benchmark for joint action. The setting of transmission substructure for wireless communication systems remains an essential engineering issue that involves competing goals. Ant-based routing has also been utilized by airlines in the assignment of arrivals of aircraft to airport gates.

Crowd Simulation

As a medium of creating multifaceted interactive systems, artists are beginning to use S.I. Miller [36] cited some significant examples of movies that have utilized SI, including Lord of the Rings and Batman Returns. Airlines, too, have not hesitated in using swarm theory to feign passengers who are about to board a plane.

Human Swarming

The networks of decentralized users can also be arranged into human swarms via the execution of existing tight-loop control systems. As Rosenberg [37] remarked, such natural systems help collections of human participators act as a unitary combined intelligence that performs as a single entity to provoke opinions, proffer answers, and predict outcomes. Systems like this are also referred to as artificial SI. They have been revealed to greatly intensify human intelligence, ensuing in an array of prominent guesses of critical correctness. Critical high-profile research work has revealed the critical medical usefulness of human swarming. A recent publication from the Stanford University School of Medicine [57] showed that a collection of human doctors could carry out the diagnosis of medical conditions with significantly greater accuracy when they are linked to one another by actual swarming algorithms than individual medical practitioners who work as one making use of standard crowd-sourcing techniques.

The Robotic Application of Swarm Intelligence

Swarms are typically made up of many simple, homogeneous or heterogeneous agents [38]. They have typically cooperated without centralized control and acted in local and straightforward ways. Only by interacting with one another can they form a collective behavior capable of completing complicated tasks. Swarms' key advantages include adaptability, robustness, and scalability as a result of these features. In addition, swarms are a type of quasi-organism that can adapt to changes in the environment by exhibiting specific characteristics [39].

Swarm robotics is a relatively new topic of study that has yet to gain widespread acceptance in the industry. On the other hand, Swarm robotics researchers have devised several platforms for testing and analyzing swarm algorithms [40]. The authors consistently noted that the simplicity of swarm robotic research platforms allowed them to anticipate future industry applications [41]. For example, Radhika Nagpal and Michael Rubenstein of Harvard University developed the Kilobot (Figure 3) swarm robot. They can act in groups of up to a thousand to carry out directives programmed by users that individual robots would be unable to carry out [42]. Kilobot collaborates with others to complete tasks that would be impossible to complete on an individual level. In addition, the Kilobots are capable of collective transportation, such as searching for food and moving in a shape by cooperating.

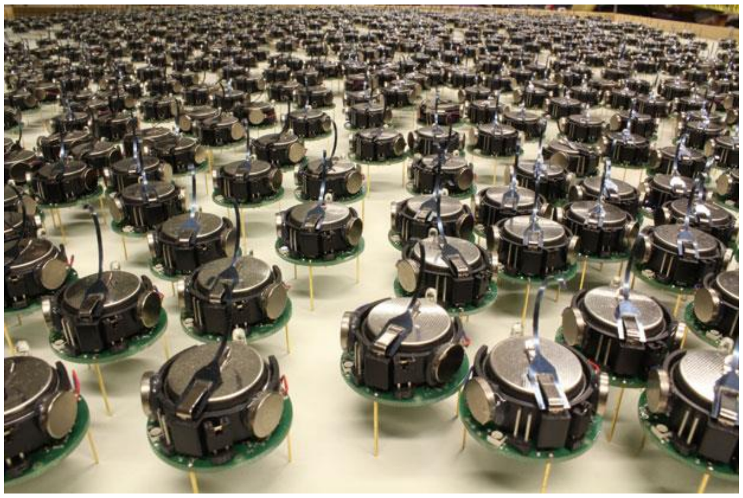


Figure 3:
Kilobot Robot

Space Agency is considering the utilization of the swarm technique for planetary exploration. So, Nasa is working on a swarm rover that will act as a mobile base, and

Marsbees that a swarm of robotic bees for more exploration and science missions on the Red Planet [43]. Moreover, like the US, advanced nations have sought to deploy robots using SI to war fronts to attack enemy militia. Other sets of usages of SI include micro air vehicles swarms that are widely researched these days. Many tasks on collective swarms of unguarded ground and aerial motors have focused on collaborative environment checking, convoy security, simultaneous localization, and plotting. An excellent example for military purpose swarm applications would be CARACAS (Contro Architecture for Robotic Agent Command and Sensing) which was developed by the Office of Naval Research (ONR), in collaboration with partners from industry, academia, and other government agencies were able to get a "swarm" of rigid hull inflatable boats (RHIBs) and other small boats to collectively perform patrol missions autonomously, rather than relying on direct human operation, using a unique combination of software, radar, and other sensors[44]. In addition, in 2013, the Strategic Capabilities Office of the United States Department of Defense conducted The Perdix drone (Figure 4) that is not controlled. Instead, it shares a collective, distributed "brain," traveling in leaderless "swarms," members of which can adapt to changes in drone numbers and remain coordinated with their counterparts [45][46][47].

Figure 4:
Perdix Swarm Drones

CHAPTER III:

BAXTER ROBOT

Introducing Baxter

The Baxter (Figure 5) is a two-armed, industrial, humanoid robot with a stationary pedestal, torso, two DOF heads, a safety system, a robot control system, a vision system, and an animated face. Rethink Robotics has created the Baxter as a kinematically redundant robot to operate fully in desired cartesian space. The Baxter has a research and education version to help numerous research and education programs such as human-robot interaction, collaborative robotics, planning, object recognition, object manipulation, computer science, perception, neurosciences, and cognitive sciences. Although Baxter's research version is identical to the industrial version, these two versions come with different software. The research and education Baxter robot come with a ROS-compatible Robot Operating System. Software Development Kit (SDK), while the manufacturing version comes with a production software created by Rethink Robotics. This research employed the Baxter to simulate the exactness of results via Gazebo, the robot simulation program.

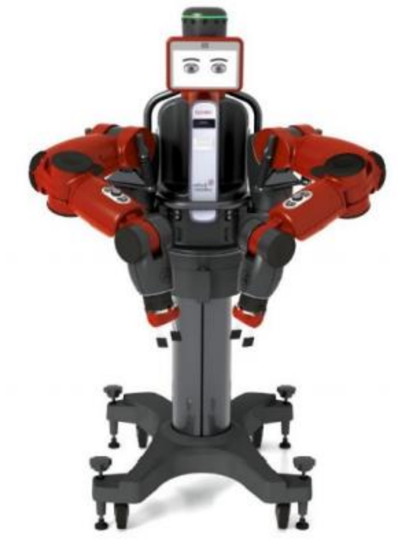


Figure 5:
Baxter Robot

Baxter remains an animated monitor for a human-like face that shows multiple facial postures as its present status determines. There are ranges of sensors all over its head, enabling it to sense people close by and offer the robot the chance of adjusting to its surroundings. As opposed to its other industrial robot counterparts, which either shut down or proceed to run inaccurately, their surroundings change. As a good instance, in a situation whereby Baxter drops a tool without which it may not continue with its job, Baxter will desist from working. However, most other robots try to continue their assignment without the appropriate tools, resulting in unsatisfactory results at the end of the day [48].

Interestingly, Baxter performs on the open-source ROS on a typical Personal Computer located in its chest. Users can place Baxter on 4-wheeled support to make them moveable. Also, Baxter possesses light sensors located in its hands which allow it to attend to details carefully.

In line with a model report from Leeds Robotic Commands as against commonplace robots, which are automated to follow a particular set of rules, users can automate Baxter by adjusting its hands to carry out some job whose signals the computer system will consequently commit to memory and be capable of repeating the job. Extra buttons, dials, and switches are also present on the arm of Baxter, making it suitable for features and precision. Furthermore, while many other industrial robots demand computer coders to program them over long hours, coding Baxter may be carried out by inexperienced staff in a short time.

While other industrial robots are designed to carry out one job fast with many rapidly-moving components unsafe for humans working around, Baxter possesses sensors over grippers, enabling it to sense and adjust to its environment. It can perceive collision occurrences on time and work towards reducing the force before the impact. What makes this possible is the presence of a spring that controls Baxter's arm rather than just rigid actuators controlling its arms. Spare cameras and sensors within Baxter's hands enable it to focus on details while working with those hands. In short, these light sensors and capabilities cause Baxter to be less unsafe.

Many colleges have adopted Baxter as part of their curriculum to offer students the familiarity of using present robotics tech to offer hands-on usages in the larger society. Several courses, such as computer sciences, mechanical engineering, and robotics, have allowed students to make the most of Baxter's benefits. This particular research was carried out perfectly with the help of Baxter. More specifically, we have used Baxter's arm to solve IK problems using PSO.

Baxter Arm

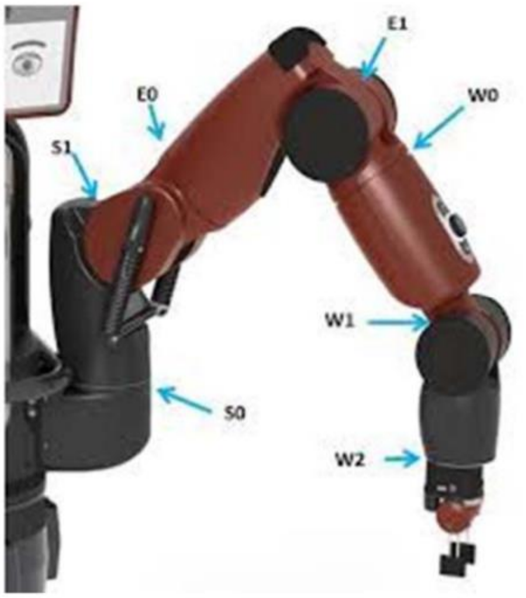


Figure 6:
Baxter Left Arm

Table 1:

Seven DOF Arm joints Naming Convention

Joint Name	**Joint Motion**
S_0	Shoulder roll
S_1	Shoulder pitch
E_0	Elbow roll
E_1	Elbow pitch
W_0	Wrist roll
W_1	Wrist pitch
W_2	Wrist roll

Table 2:

Seven DOF left and right arm joint limits

Joint Name	Joint Variable	θ_i min	θ_i max	θ_i range
S_0	θ_1	+51°	-141°	192°
S_1	θ_2	+60°	-123°	183°
E_0	θ_3	+173°	-173°	346°
E_1	θ_4	+150°	-3°	153°
W_0	θ_5	+175°	-175°	350°
W_1	θ_6	+120°	-90°	210°
W_2	θ_7	+175°	-175°	350°

Denavit-Hartenberg

The Denavit–Hartenberg parameters **(**DH parameters**)** are presented by Jacques Denavit and Richard Hartenberg in 1955. DH has four parameters associated with a particular convention for attaching reference frames to a spatial kinematic chain or robot manipulator's links [48][49]. Richard Paul demonstrated the significance of this system on kinematic analysis of the robotic system in 1981. However, since many other conventions for attaching reference frames have been developed, The DH convention has held the most popular approach [50]. The cartesian reference frame definitions for Baxter's 7-DOF left arm are shown in Figure 7. Table 3 gives the associated DH parameters for the 7-DOF left arm, and Table 4 provides the length of the 7-DOF arm.

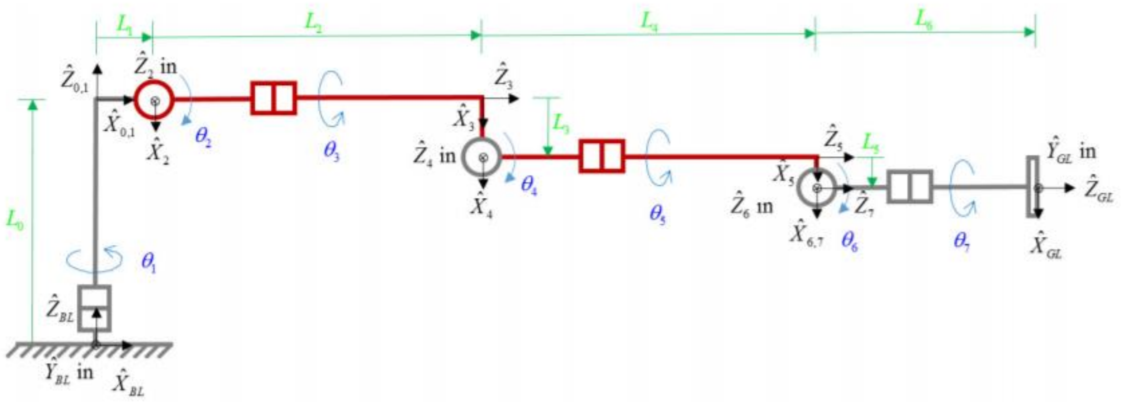


Figure 7:
Baxter's Left Arm Cartesian Frame Assignment

The following four transformation parameters are D–H parameters

- d - the distance between the previous x-axis and the current x-axis along the previous z-axis.
- θ - the angle around the z-axis between the previous x-axis and the current x-axis.
- a - the length of the common normal, which is the distance between the previous z-axis and the current z-axis
- α - the angle around the common normal between the previous z-axis and the current z-axis.

Table 3:

Seven DOF left arm DH table

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	-90	L_1	0	θ_2+90
3	90	0	L_2	θ_3
4	-90	L_3	0	θ_4
5	90	0	L_4	θ_5
6	-90	L_5	*0*	θ_6
7	90	0	L_6	θ_7

Table 3 illustrates the Denavit-Hartenberg parameters for the Baxter. Although all lengths are given in mm, the lengths are used in meters in the algorithm. Also, the angles are used in radian in the algorithm.

Table 4:

Baxter's Arm Length

Length	Value (mm)
L0	281.35
L1	125.00
L2	364.35
L3	69.00
L4	374.29
L5	10.00
L6	229.5

Forward Kinematics

Forward kinematics refers to using the kinematic equations to compute the end-effector position from specified values for the joint parameters [50]. In other words, given a kinematic chain composed of links and joints with multiple degrees of freedom, finding the end-effector's position and orientation in the functional workspace when all the joint parameters are known [51].

The homogeneous transformation matrix from frame {i} to frame {i-1} is given as followed:

Equation (3.1)

$$\left[{}^{i-1}_{i}T\right] = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i\ \sin\alpha_{i-1} \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i\ \cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \left[{}^{i-1}_{i}R\right] & \left\{{}^{i\text{-}1}_{i}P_i\right\} \\ 0 \quad 0 & 0 \quad 1 \end{bmatrix}$$

The equation (3.1) represents the pose (position and orientation) of frame {i} concerning frame {i–1} by using a 4x4 homogeneous transformation matrix. The upper left 3x3 matrix is the rotation matrix giving $\left[{}^{i-1}_{i}R\right]$ the orientation of frame {i} with respect to frame {i–1}, expressed in { i–1} coordinates. The upper right 3x1 vector $\left\{{}^{i\text{-}1}_{i}P_i\right\}$ is the position vector from the origin of {i–1} to the origin of {i}, expressed in { i–1} coordinates. In this project, in order to find the end effector's pose, homogeneous transformation equations are used concerning the base reference frame $[{}^{BL}_{0}T]$ to complete the FK solution for each consecutive chain by substituting each row of the DH parameters in Table 3.3 into the equation to obtain the seven neighboring homogeneous transformation matrices below as a function of the joint angles for the 7-DOF left arm.

$$[{}^{0}_{1}T] = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & L_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} [{}^{1}_{2}T] = \begin{bmatrix} c_2 & -s_2 & 0 & L_1 \\ 0 & 0 & 1 & 0 \\ -s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [{}^{2}_{3}T] = \begin{bmatrix} c_3 & -s_3 & 0 & 0 \\ 0 & 0 & -1 & -L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[{}^{3}_{4}T] = \begin{bmatrix} c_4 & -s_4 & 0 & L_3 \\ 0 & 0 & 1 & 0 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [{}^{4}_{5}T] = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & -L_4 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[{}^{5}_{6}T] = \begin{bmatrix} c_6 & -s_6 & 0 & L_5 \\ 0 & 0 & 1 & 0 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [{}^{6}_{7}T] = \begin{bmatrix} c_7 & -s_7 & 0 & 0 \\ 0 & 0 & -1 & -L_6 \\ s_7 & c_7 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where the following abbreviations are used: $s_i = \sin\theta_i$, $c_i = \cos\theta_i$

We derive the FK result by substituting seven neighboring homogeneous transformation matrices through the homogeneous transform equation below.

(Equation 3.2)

$$[{}^{0}_{7}T] = [{}^{0}_{1}T(\theta_1)]\ [{}^{1}_{2}T\ (\theta_2)]\ [{}^{2}_{3}T(\theta_3)]\ [{}^{3}_{4}T(\theta_4)]\ [{}^{4}_{5}T(\theta_5)]\ [{}^{5}_{6}T(\theta_6)]\ [{}^{6}_{7}T(\theta_7)]$$

Since we need Baxter's pose in the base frame, we multiply forward kinematics solution $[{}^{0}_{7}T]$ with the transformation matrix $[{}^{BL}_{0}T]$ below.

$$[{}^{BL}_{0}T] = \begin{bmatrix} 0.707106 & -0.707106 & 0 & 0.026 \\ 0.707106 & 0.707106 & 0 & 0.219 \\ 0 & 0 & 1 & 0.108 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(Equation 3.3)

$$[{}^{BL}_{7}T] = [{}^{BL}_{0}T] * [{}^{0}_{7}T]$$

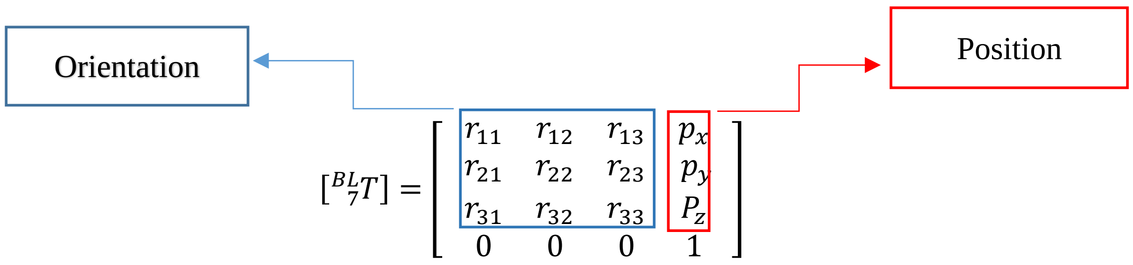

Orientation
Position
$[{}^{BL}_{7}T] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Figure 8:
Forward Kinematics Expressions

Jacobian

Jacobian matrices are a helpful tool and are heavily used throughout robotics [52][53]. Jacobian is the matrix in robotics that provides the relation between joint velocities ($\dot{q}$) & end-effector velocities ($\dot{X}$) of a robot manipulator.

(Equation 3.4)

$$\dot{X} = J\ \dot{q}$$

where, $\dot{q}$, the column matrix is representing the joint velocities. The size of this matrix is $nx1$. 'n' is the number of joints of the robot. $\dot{X}$, the column matrix is representing the end-effector's velocities. The size of this matrix is $mx1$. J is the Jacobian matrix which is a function of the current pose. Size of Jacobian matrix is mxn. The expanded matrix form of the above equation (3.4) is given as follows.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} j_{11} & j_{12} & . & . & . & j_{1n} \\ j_{21} & j_{22} & . & . & . & j_{2n} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ j_{61} & j_{62} & . & . & . & j_{6n} \end{bmatrix}_{6xn} * \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ . \\ . \\ . \\ \dot{q}_n \end{bmatrix}_{nx1}$$

Each column is related to each joint of the manipulator. Jacobian matrix represents the effect on end-effector velocities due to variation in each joint velocity. Hence the number of columns in the Jacobian matrix is equal to the number of joints in the manipulator. The first three elements of the end-effector velocity matrix ($\dot{X}$) are linear $[\dot{x} \quad \dot{y} \quad \dot{z}]$ velocities [rate of change of position] and the last three elements $[\dot{\alpha} \quad \dot{\beta} \quad \dot{\gamma}]$ are the angular velocities [rate of change of orientation] in $[\dot{x} \quad \dot{y} \quad \dot{z}]$ direction, respectively.

Inverse Kinematics

In computer animation and robotics, IK is the method of measuring the variable joint parameters required to place the end of a kinematic chain, such as a robot manipulator, in a specified location and orientation relative to the chain's start. With Given joint parameters, the position and orientation of a manipulator can typically be calculated directly using multiple applications of trigonometric formulas, with the forward kinematics method has been explained above. The reverse operation, on the other hand, is even more complex [54].

IK is a technique for determining the inverse of a W to Q mapping:

(Equation 3.5)

$$Q_{inv=} \; F^{-1}(W)$$

Modeling and solving IK problems can be done in a variety of ways. Due to the complexity of inverting the forward kinematics equation and the probability of an empty solution space, the most versatile approaches usually use iterative optimization to find an approximate solution. Heuristic methods can also be used to estimate the IK problem. These methods use basic iterative operations to arrive at a solution approximation eventually. Heuristic algorithms are simple to use (they return the final pose quickly) and typically support joint constraints.

Quaternion

A quaternion is a four-element vector in a three-dimensional coordinate system that can be used to encode any rotation. A quaternion is a mathematical construct made up of one real and three complex elements that can be used for much more than rotations. Quaternions are simpler than Euler Angles, avoid ambiguity, are numerically stable, and are more efficient in terms of computing implementations [55]

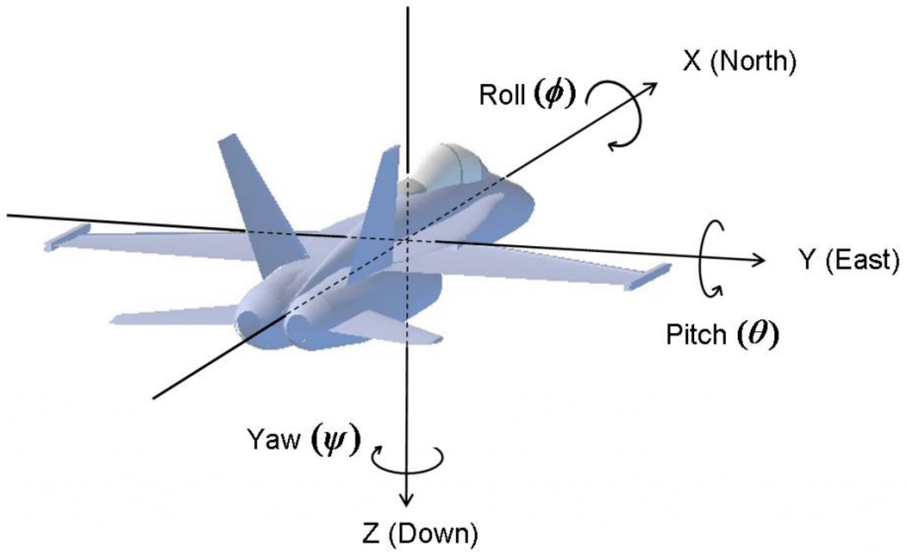


Figure 9:
The Inertial Frame

As shown in Figure 9, the inertial frame is an Earth-fixed coordinate frame with the x-axis pointing north, the y-axis pointing east, and the z-axis pointing down. Any rotation or series of rigid body rotations about a fixed point is equivalent to a single rotation by a given angle about a fixed axis moving through the fixed point, like Euler's rotation theorem shows. Quaternions are a four-number representation of rotation that uses three numbers to represent the axis vector and one number representing the angle.

Identifying individual objects in an image and determining their location and orientation relative to any coordinate system is common in computer vision and robotics. This knowledge can then be used to allow a robot to control an object or prevent a robot from colliding. A rotation and translation transformation can be used to define the pose. A rotation and translation transformation that takes the object from a reference pose to the observed pose may define the pose. This rotation transformation can be modeled in a variety of ways. For example, Baxter uses the quaternion representation.

Gazebo Simulation in Brief

The Gazebo is a 3D robotics simulator that is free and open source. From 2004 through 2011, the Gazebo was an essential part of the Player Project. The ODE physics engine, OpenGL rendering, and support code for sensor simulation and actuator control were incorporated into Gazebo. Gazebo became a self-contained project with Willow Garage's assistance in 2011. The Gazebo project was stewarded by the Open Source Robotics Foundation (OSRF) in 2012. In 2018, the Open Robotics Research Foundation (OSRF) changed its name to Open Robotics[21]. Gazebo promotes a variety of high-performance physics engines, including ODE, Bullet, and others. It gives realistic rendering environments, including high-quality lighting, shadows, and textures. It can model sensors that "see" the simulated environment, such as laser range finders, cameras (including wide-angle), Kinect style sensors.[22] In Addition, Gazebo provides exceptional conveniences for studying industrial and research robots such as Baxter. In this study, we have verified the accuracy of our result thanks to Gazebo.

CHAPTER IV:

PROPOSED ALGORITHM FOR SOLVING INVERSE KINEMATICS

This chapter introduces the swarm intelligence algorithm to solve any robotic manipulator IK problems by moving the particles at each iteration to perceive a better solution. If there is no improvement after several iterations, the re-hope portion code is performed for more iterations.

Solving Inverse Kinematics with PSO

The PSO that has been advanced for a long time now got its inspiration from a drove of birds and has become reckoned as having a robust search algorithm. First, it refers to a method utilized by researchers [56]. Many benefits are tied to using this algorithm. For instance, it can be applied seamlessly, and it has robust control benchmarks compared with most other meta-heuristic methods. Essentially, the PSO is chosen concertedly for non-linear issues, which form one of the main challenges with the search space and offer improved results. In PSO, particles get optimal results alongside making use of the neighborhood as well as experience. However, just two equations are used to get on to the aim of the particles in particle swarm optimization.

The goal of an optimization problem is to determine a variable represented by a vector
$X = [x_1\ x_2 \dots \dots x_n]$ that minimizes or maximizes depending on the proposed optimization formulation of the function $f(X)$. The variable vector X is known as the position vector; this vector represents a variable model. It is n dimensions vector, where n represents the number of variables that may be determined in a problem. In this problem, n is determined seven $[P_x\ P_y\ P_z]$ as the position and $[q_0\ q_1\ q_2\ q_3]$ as the orientation, which also representing quaternion values. In order to derive $[q_0\ q_1\ q_2\ q_3]$ orientation

from the upper left 3x3 (rotm) matrix of Baxter's forward kinematics matrix, we used the conversion of the rotational matrix to quaternion with the MATLAB code below;

Code 4.1

```
quat = rotm2quat(rotm)
```

Regarding a swarm with particles, there is a position vector X_{ij}^{t} and a velocity vector V_{ij}^{t} at iteration(t) for each one of the i particle that composes it. These vectors are updated through the dimension j according to the following equations:

(Equation 4.1)

$$V_{ij}^{t+1} = wV_{ij}^{t} + c_1^t r_1^t \left(Pbest_{ij} - X_{ij}^{t}\right) + c_2^t r_2^t \left(Gbest - X_{ij}^{t}\right)$$

(Equation 4.2)

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1}$$

Taking j = 1,2, …., n as the dimension while i = 1, 2…, P refers to the swarm size even as c_1 alongside c_2 refers to the weight of personal best and global best, respectively. In the same vein, r_1 and r_2 are merely random figures that spread equally (0, 1).

Euclidian

In this research work, the objective of the optimization challenge is to discover the ideal angle value (θ) for every joint. At the end of it all, the robot arm's end-effector will be taken to the desired destination by θ. Therefore, the correct computations of θ values are highly essential.

Equation (4.3)

$$\sqrt{(P_{q3}^2 - T_{q3}^2) + (P_{q2}^2 - T_{q2}^2) + (P_{q1}^2 - T_{q1}^2) + (P_{q0}^2 - T_{q0}^2) + (P_{px}^2 - T_{px}^2) + (P_{py}^2 - T_{py}^2) + (P_{pz}^2 - T_{pz}^2)}$$

The major objective of this research is to bring a solution to the optimization problem by working out the PSO. For this reason, we implemented a cost function that is dependent on Euclidian distance with particular reference to the equation above, which is between the target pose $[\, Tp_x \; Tp_y \; Tp_z \;\; Tq_0 \; Tq_1 \; Tq_2 \; Tq_3 \;]$ and the particles' pose $[\, Pp_x \; Pp_y \; Pp_z \;\; Pq_0 \; Pq_1 \; Pq_2 \; qP_3 \;]$.

Quaternion conversion to Angular Velocity ($\boldsymbol{\omega}$)

A quaternion q is a set of four parameters, a real value q_0 and three imaginary values q_1i, $q_2 j$, $q_3 k$ with $q_0, q_1, q_2 \in R$ ∈ R; it may be written as follows.

Equation (4.4)

$$q = q_0 + q_1 i + q_2 j + q_3 k$$

Euler angles are more human-friendly and can decompose rotations into individual degrees of freedom (for kinematic joints and the like). However, they have drawbacks such as ambiguity and gimbal lock, which can cause a problem at specific end effector orientations, causing the solution to failing to converge. However, Quaternions do not suffer from this ambiguity since they only represent a single rotation with a well-defined axis. Therefore, using quaternion values makes our result exactly accurate and reliable. In order to derive each joint velocity, we have to use the conversion of quaternion velocity (Equation 4.5) to the angular velocity. (Equation 3.4).

Equation (4.5)

$$\omega = 2 * \underbrace{\begin{bmatrix} -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix}}_{\boxed{E}} * \begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}$$

or

$$\omega \ = \ 2 * E * \dot{q}$$

Pseudo Inverse($\boldsymbol{j^{+}}$)

As we said in chapter three, Jacobian is the matrix in robotics that provides the relation between joint velocities ($\dot{q}$) and end-effector velocities ($\dot{X}$) of a robot manipulator (Equation 3.4). Hence the Jacobian velocities equation cannot be used directly for Baxter. So we have to make a relation between end-effector displacement and joint positions instead of their velocities. In order to calculate each joint angle, we have to use Pseudo Inverse (j^{+}) as it is shown:

(Equation 4.6)

$$\boldsymbol{j^{T} * \dot{X} = j^{T} * j * \dot{q}}$$

$$=$$

$$(j^{T} * j)^{-1} * j^{T} * \dot{X} = (j^{T} * j)^{-1} * j^{T} * \dot{q}$$

$$=$$

$$j^{+} * \ \dot{X} = \dot{q}$$

The parameters for the proposed PSO algorithm are shown below (Table 5)

Table 5*:*

PSO Parameters

Parameter	*Description*
j	Number of dimensions
i	Number of particles
t	Number of iterations
X_{ij}^{t+1}	Next angle.
V_{ij}^{t+1}	Next velocity
ω	Inertia coefficient
c_1, c_2	Personal acceleration(c_1), Global acceleration(c_2)
r	Random number (0-1)
ω_{damp}	Damping inertia
	Global Best Pose
$Pbest$	Personal best pose
X_{ij}^{t}	Current particle pose
V_{ij}^{t}	Current particle velocity
Q_j	Angle
Rh	Re-hope

The pseudo-code for the PSO algorithm is given below:

```
Step 1:  Set number of joints = 7 or any number of joints.
Step 2: Set PSO parameters (Max Iteration, Population, Inertia Coefficient ($\omega$), $\omega_{damp}$=0.973, Personal acceleration($c_1$), Global acceleration($c_2$), $RH = 0$)
Step 3: Set target joint angle.
Step 4: Compute transformation matrix $\left[{}^{i-1}_{i}T\right]$ according to Equations (3.2 and 3.3)
Step 5: Convert rotation matrix $\left[{}^{i-1}_{i}T\right]_{\mathbf{3*3}}$ to quaternion $[q_1\ q_2\ q_3\ q_4]$ according to code (4.1
Step 6: Derive target Pose $\left[\, p_x\ p_y\ p_z\ \ q_0\ q_1\ q_2\ q_3\ \ \right]$
Step 7: Initialize particles ( $p_i$) angle randomly in the limit of angles (Table 2)
Step 8: Initialize  $p_i$' velocity ($v_i$) as  0,000001;
Step 9: Compute $p_i$' transformation matrix $\left[{}^{i-1}_{i}T\right]$ according to Equations (3.2 and 3.3)
Step 10: Convert rotation matrix $\left[{}^{i-1}_{i}T\right]_{\mathbf{3*3}}$ to quaternion $[q_1\ q_2\ q_3\ q_4]$ code (4.1)
Step 11: Derivate  $p_i$' pose  $\left[\, p_x\ p_y\ p_z\ \ q_0\ q_1\ q_2\ q_3\ \ \right]$= ($X_{ij}^{t}$)
Step 12:  Calculate Cost function for $p_i$ with Euclidean (4.3)
Step 13: If particle fitness < Global best cost then global  = $Pbest$
Step 14: If i < Population then go to step 7 , $i$ = i + 1
Step 15: Update the  $p_i$ velocity ($v_i$) according to equation (4.1)
Step 16: The next step is to compute the particle velocity (translational velocity + rotational velocity) using the equation Equation (4.5)
Step 17:  Convert quaternion to angular velocity as shown in equation (4.5)
Step 18 Then, the particle' joint velocities are computed using the Moore-Penrose pseudo inverse of the Jacobian matrix.
Step 19: derive the each angle joint according to Equation (3.4 and 4.6)
```

Step 20: **if** end effector's velocity > error*10^{-1} **then** end- effector= error*10^{-1} **if else <** end effector's velocity > error*10^{-2} **then** end- effector = error*10^{-2} .

Step 21: Update joint angles according to equation (4.2)

Step 22: If Q_{ij}^{t+1} exceeds the angle limits, keep previous Q_{ij}^{t} angle value.

Step 23: Compute P_i^t transformation matrix $\left[{}^{i-1}_{i}T\right]$ according to Equations (3.1)

Step 23: Convert rotation matrix $\left[{}^{i-1}_{i}T\right]_{\mathbf{3*3}}$ to quaternion $[q_1\ q_2\ q_3\ q_4]$ according to code (4.1)

Step 23: Compute pose [$p_x\ p_y\ p_z\ \ q_1\ q_2\ q_3\ q_4$] = (X_{ij}^t) for updated angle

Step 24: Calculate Cost function for p_i with Euclidean (4.3)

Step 25: **If** particle fitness < Global best cost **then** = $Pbest$, RH=0

Step 26: **If** i < Population **then** go to **step 15,** $i = \text{i} + 1$

Step 27: damping the inertia value each iteration $w = \omega_{damp} * w$

Step 28: **if** RH > 5 **then** randomly picked 20 Particles = Global Best Angle, set $RH = 0$

Step 27: **If** $Error\ Criteria$ < **then** go to **step 15,** $t = \text{t} + 1$ **otherwise** end

CHAPTER V:
RESULT AND ANALYSIS

This research aims to examine the effectiveness of the PSO algorithm to proffer a solution to any robotic manipulator's IK problems. These simulations are carried out with the following parameters: Iteration = 500-2000 and the particle = 50-80. The proposed algorithm was tested under three different scenarios, which consist of 6 DOF, 7 DOF, and 8 DOF manipulators. Our work has shown the possibility of finding infinite solutions which lead the manipulator to the exact pose in the cartesian space. The first scenario is aimed to find the exact target angles. The second scenario is proposed to obtain different joint angles for the 7-DOF robot Baxter. Lastly, we have found a solution for 8 DOF manipulators beneath the third scenario. All scenarios are carried out via MATLAB coding on a MacBook with 2.6 GHz Intel Core i7.

Obtained Result For Scenario 1

In this scenario, we kept the first angle constant θ_1=0.7850(rad) to compute the optimal configurations of 6 DOF manipulators. Then, randomly generated six dimension particles' were used to get the end-effector desired pose (position and orientation). In order to increase the efficiency of PSO, we have accelerated the first velocity as 0.00001 rad/sec Fig. 10 shows the proposed algorithm can converge to less than 1 mm after 750 iterations. As shown in table 7, we obtained the lowest error value at the 2314th iteration. As shown in Fig 10, the last 200 iterations errors graph is stagnated. As expected, the algorithm proved its accuracy over the test run, whose results are presented in Table 6.

Table 6:

Target and Obtained Angles Values

	θ_1 (rad)	θ_2 (rad)	θ_3 (rad)	θ_4 (rad)	θ_5 (rad)	θ_6 (rad)	θ_7 (rad)
Target Angles	-0.7850	-0.7850	0	1.5710	0	-0.7850	0
Obtained Angles	-0.7850	-0.7850	-2.78e-05	1.5710	-4.98e-05	-0.7850	6.19e-05

Table 7:

Simulation Results for 6 DOF

Parameters	
Euclidian Distance (Meter)	1.2925e-05
Number Of Iteration	2314
Number of Particles	60

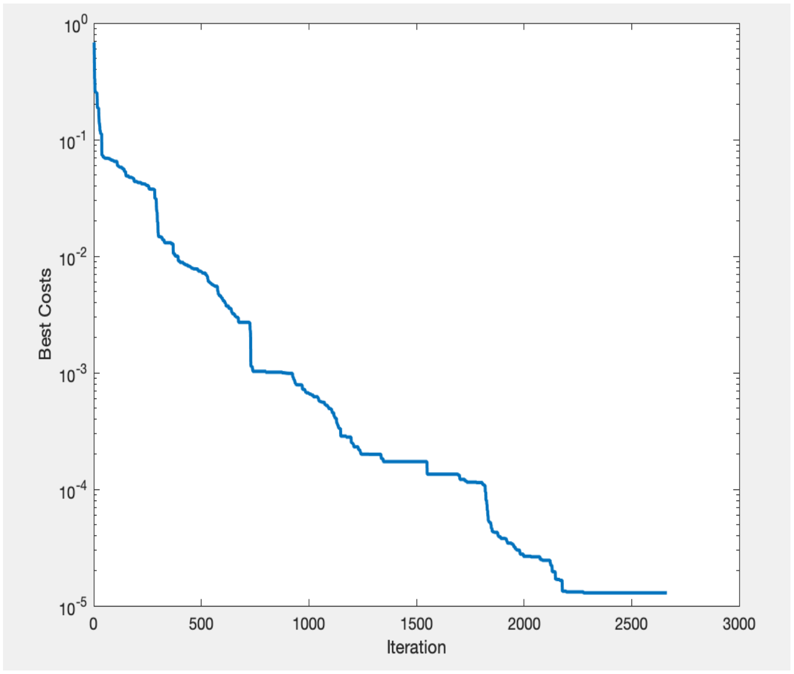


Figure 10:
Error's graphic for 6-DOF robot

Result Obtained For Scenario 2

We use two different setup joint angles as goals in this scenario. The inverse kinematics solution was performed with 60 particles that have seven dimensions which represent pose [$Pp_x \; Pp_y \; Pp_z \;\; Pq_0 \; Pq_1 \; Pq_2 \; qP_3$] of Baxter and first angle values are randomly chosen within the minimum and maximum angle values (Table 3.2). we applied the algorithm with different initial parameters.

$$i = 60,\; w_{damp} = 0.973,\;\; c1 = 1,\;\; c2 = 1,\;\; V_{initial} = 0.00001\, rad/sec,\;\; w = 0.90.$$

We found three different alternative solutions (Table 6 and Table 7) for each setup joint angle that move Baxter's end-effector to the same point in cartesian space. The accuracy of the results is demonstrated using gazebo simulation.

Experiment-1 Untucking Setup Joint Angles

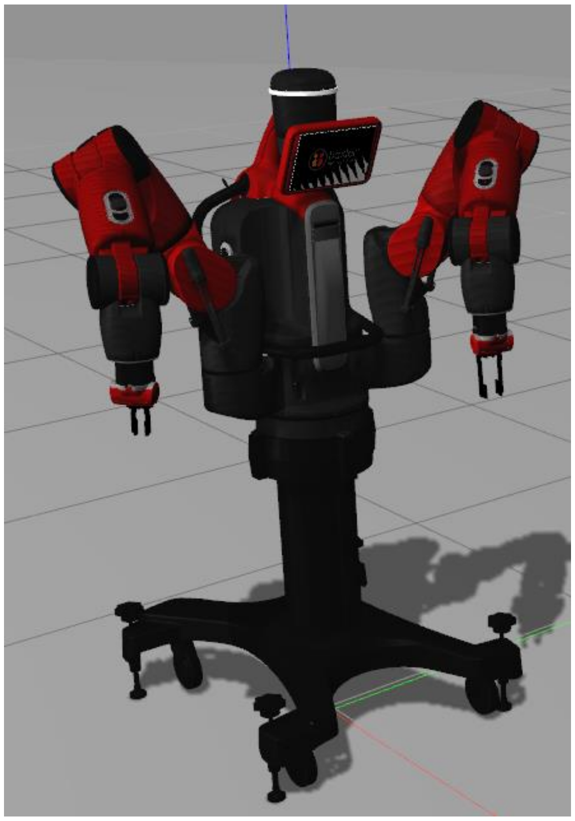


Figure 11:
How Baxter looks like with 1st Setup Joint Angle

Figure 11 shows how Baxter's left arm corresponds to the angles listed in Table 8. As there is randomness picking up initial angles, there is a chance of ending up at different angles to attain the same pose. As listed in table 8, we obtained three different solutions that move Baxter's end-effector to the same place in cartesian space with different joint angles. Table 9 depicts how many iterations later the distance decreased to less than 1 micrometer(µm) for each solution with sixty particles.

Table 8:

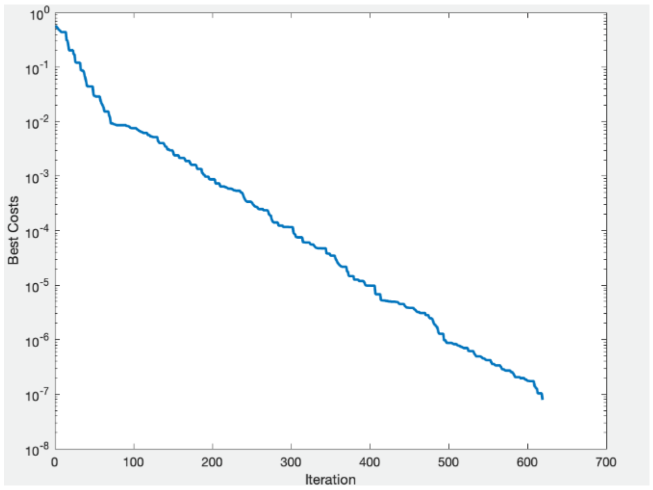
Target and Obtained Angle Values for first setup joint angles

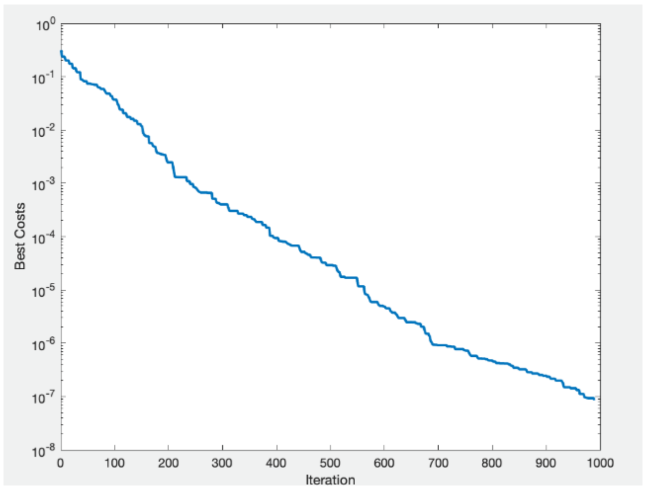
Angle	First Setup Joint Angles	First Obtained Angles	Second Obtained Angles	Third Obtained Angles
θ_1(rad)	- 0.08	-1.1938	-1.7632	-0.0992
θ_2 (rad)	- 1.00	-1.3554	-0.3839	0.3053
θ_3(rad)	- 1.19	0.4464	1.5796	-2.0289
θ_4(rad)	+ 1.94	2.1104	1.9317	1.9728
θ_5(rad)	+ 0.67	3.0312	1.9006	1.7649
θ_6(rad)	+ 1.03	-0.7850	-1.4011	2.0944
θ_7(rad)	- 0.50	-2.7564	-1.9215	-0.8646

Table 9*:*

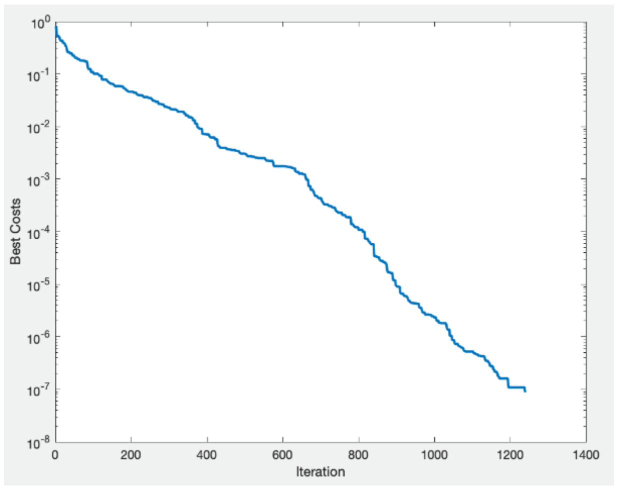
Simulation Results for 7 DOF first setup joint angles

Solutions	Euclidian Distance (Meter)	Number Of Iteration	Number of Particles
1st Solution	7.8634e-08	619	60
2nd Solution	8.4836e-08	989	60
3rd Solution	8.8069e-08	1241	60





(a) (b)



(c)

Figure 12:
Pose errors of (a) first (b) second (c) third obtained angles for first setup joint angles

The pose error graph of the three optimum solutions is shown in Figure 12. The developed PSO method effectively searched for distinct three configurations of the robotic manipulator, as illustrated in Fig. 13(a, b, c). The differences can be seen clearly among target Fig.13(d) and obtained solutions. These simulations show that the suggested PSO method effectively solves the IK problem for the 7-DOF robotic manipulator. Even though the derived solutions varied, they all lead the end-effector to the same place as the setup joint angle.

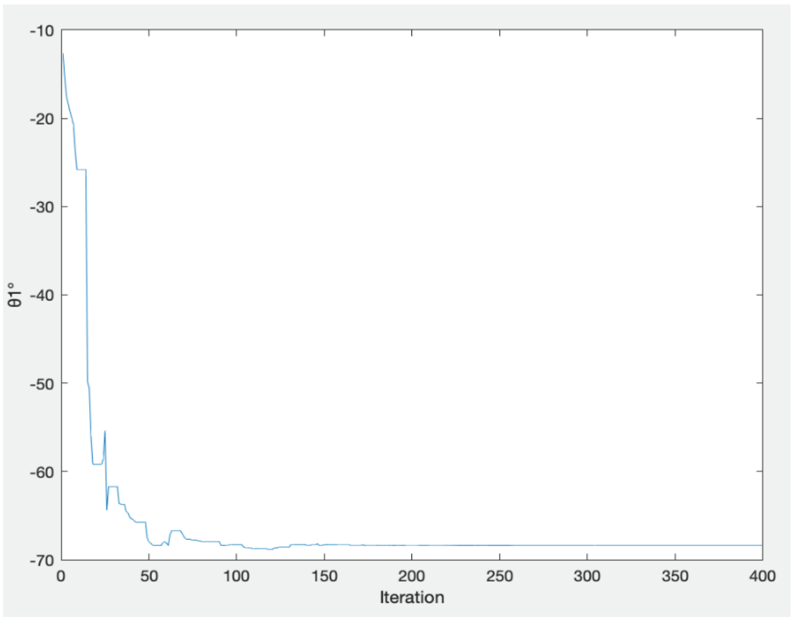


Figure 13:
Joint Angle (θ_1) Graph in Scenario 2 - Experiment 1

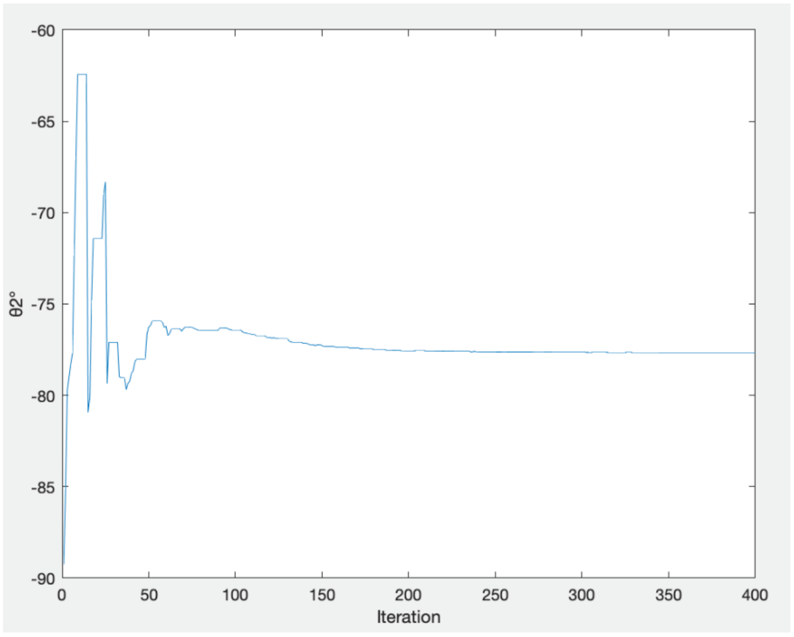


Figure 14:
Joint Angle (θ_2) Graph in Scenario 2 - Experiment 1

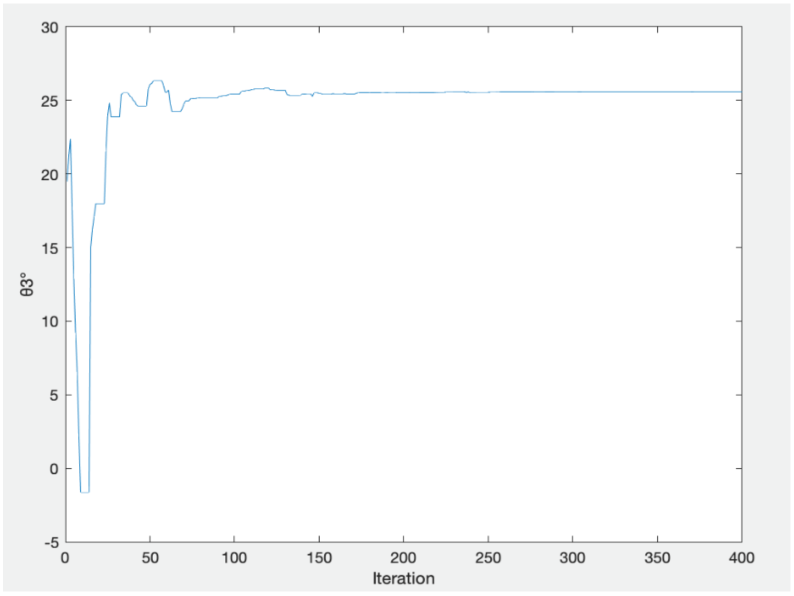


Figure 15:
Joint Angle (θ_3) Graph in Scenario 2 - Experiment 1

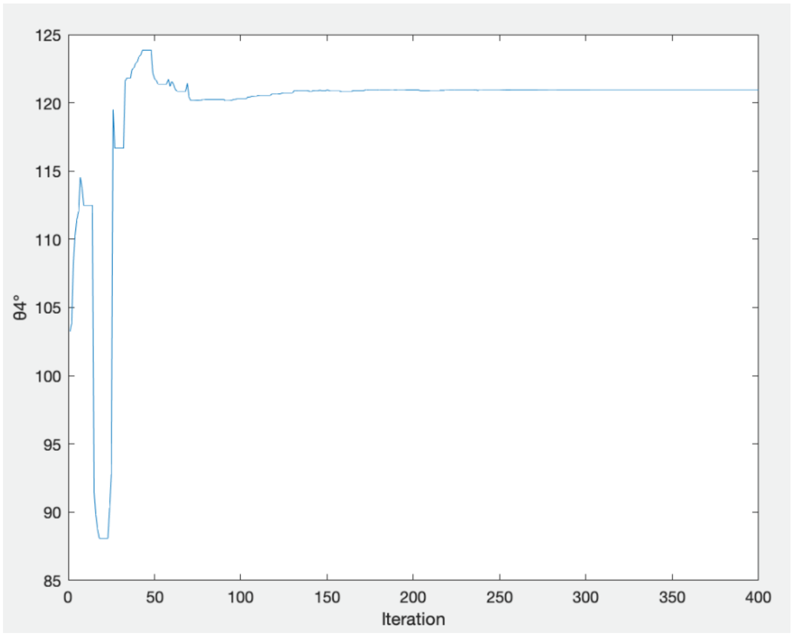


Figure 16:
Joint Angle (θ_4) Graph in Scenario 2 - Experiment 1

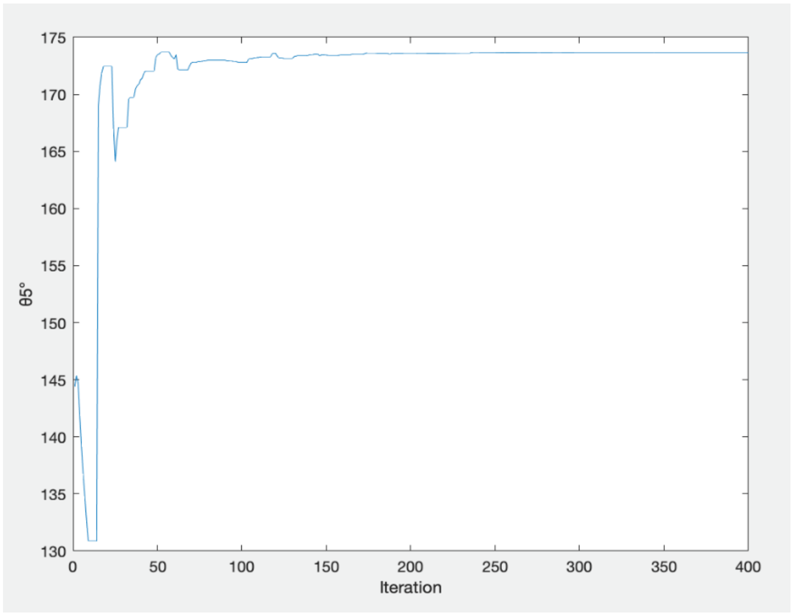


Figure 17:
Joint Angle (θ_5) Graph in Scenario 2 - Experiment 1

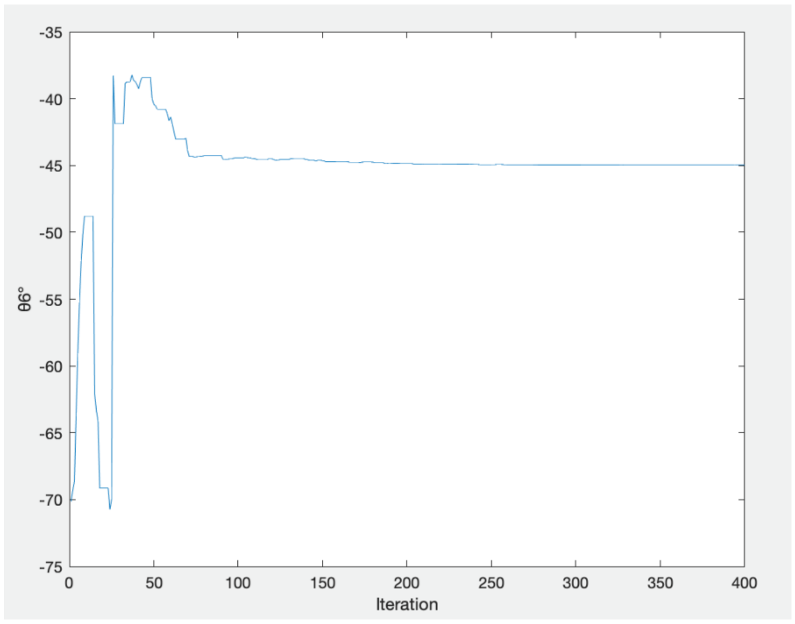


Figure 18:
Joint Angle (θ_6) Graph in Scenario 2 - Experiment 1

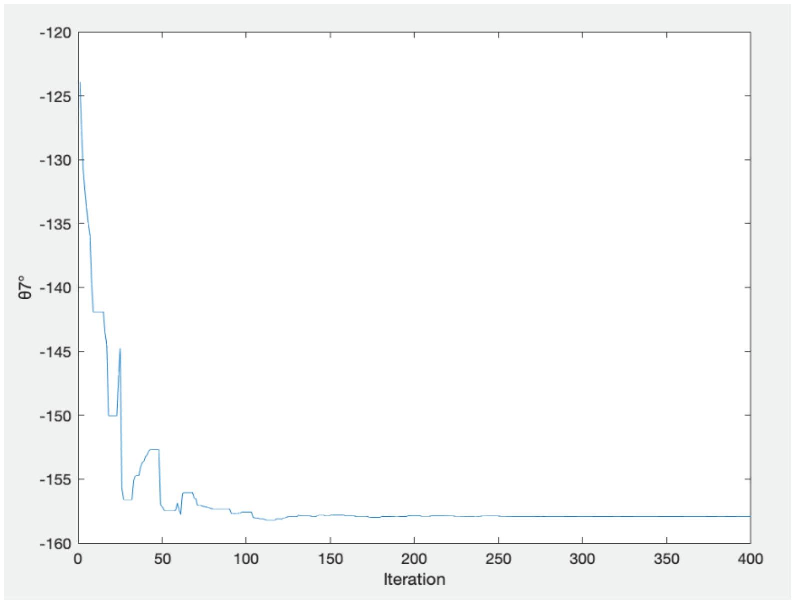


Figure 19:
Joint Angle (θ_7) Graph in Scenario 2 - Experiment 1

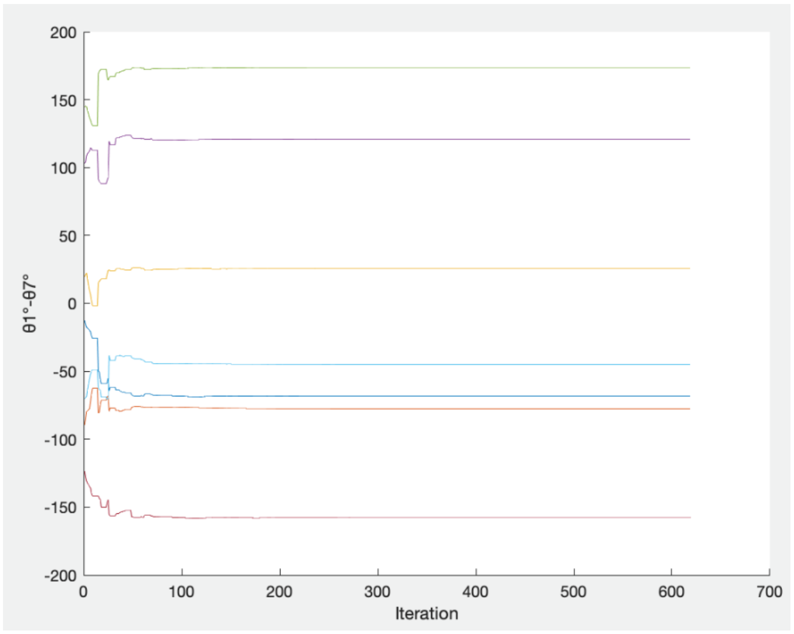


Figure 20:
All Joint Angle Graph in Scenario 2 - Experiment 1

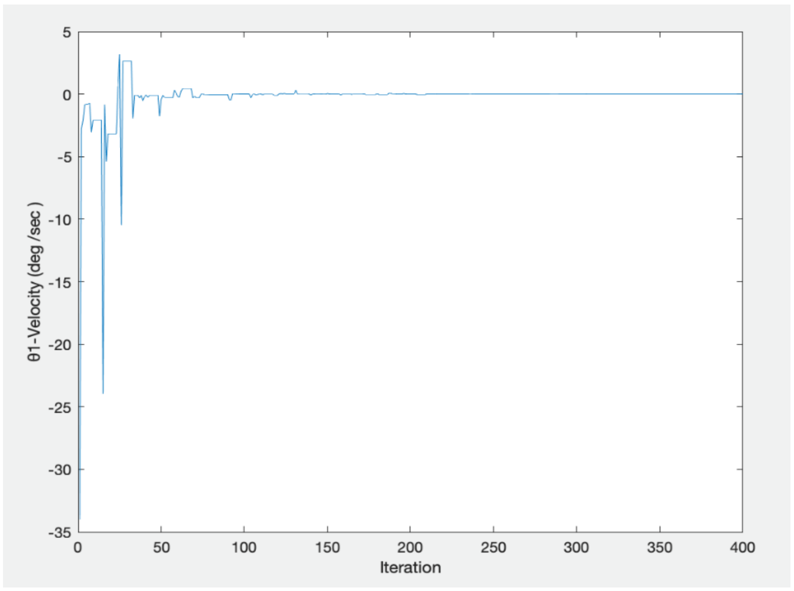


Figure 21:
Joint Velocity (deg/sec) for (θ_1) in Scenario 2 - Experiment 1

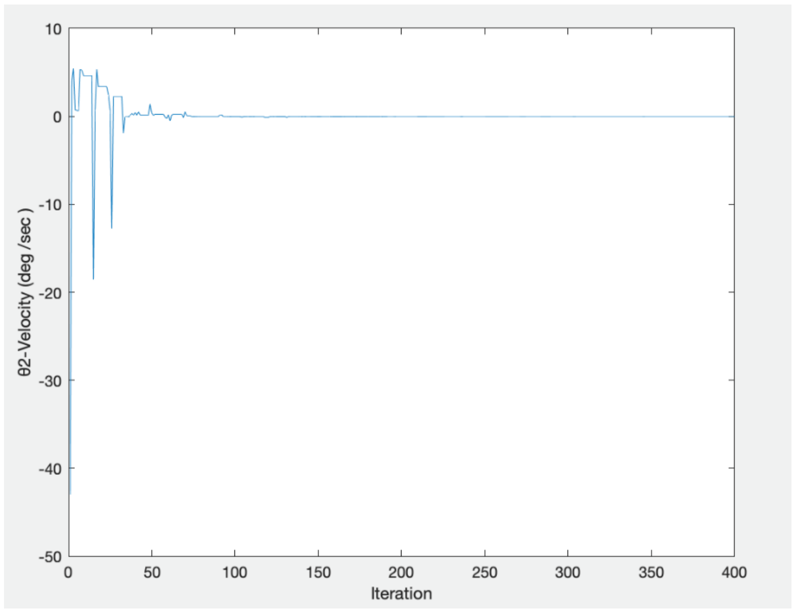


Figure 22:
Joint Velocity (deg/sec) for (θ_2) in Scenario 2 - Experiment 1

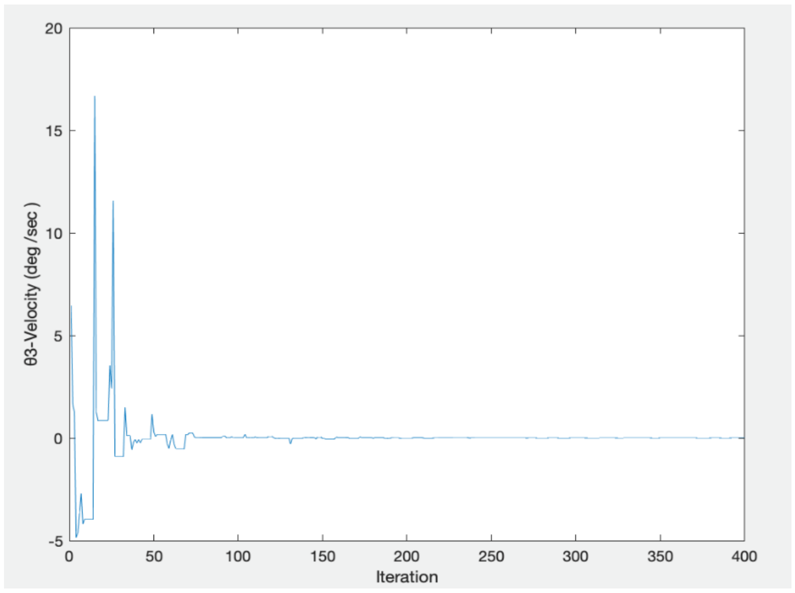


Figure 23:
Joint Velocity (deg/sec) for (θ_3) in Scenario 2 - Experiment 1

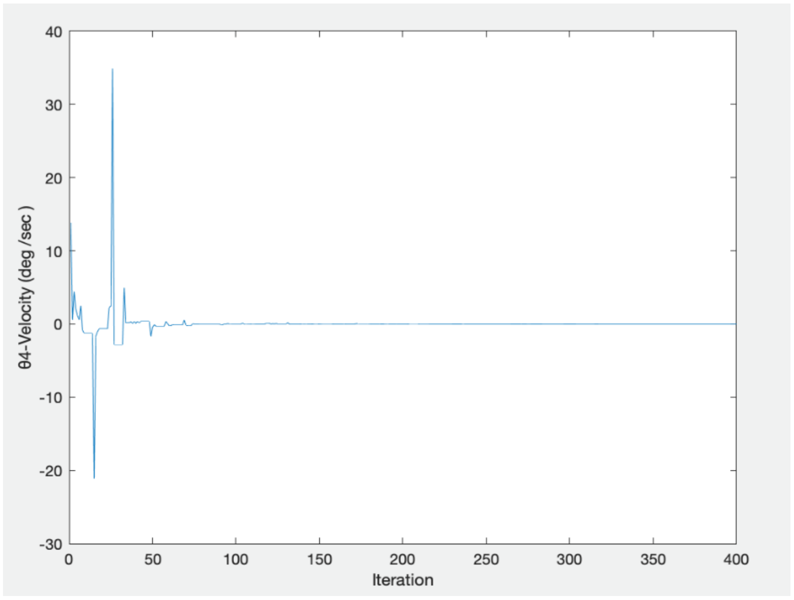


Figure 24:
Joint Velocity (deg/sec) for (θ_4) in Scenario 2 - Experiment 1

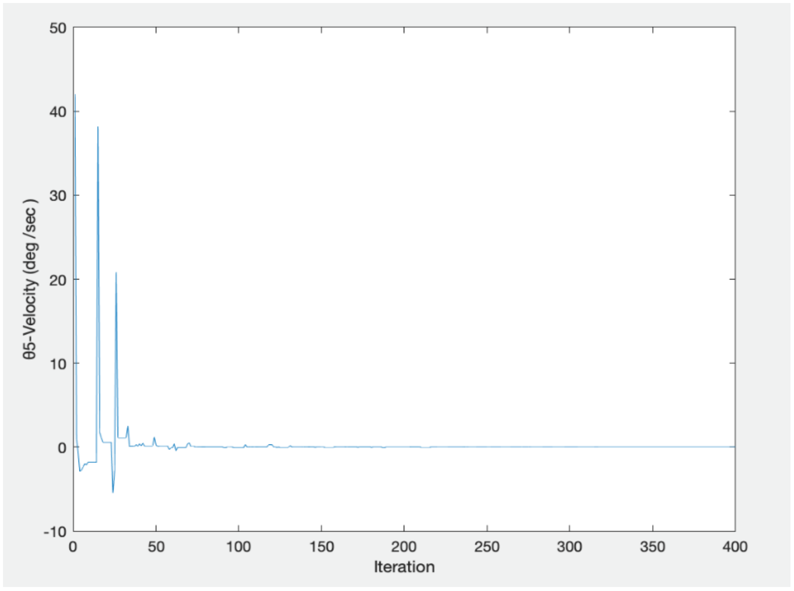


Figure 25:
Joint Velocity (deg/sec) for (θ_5) in Scenario 2 - Experiment 1

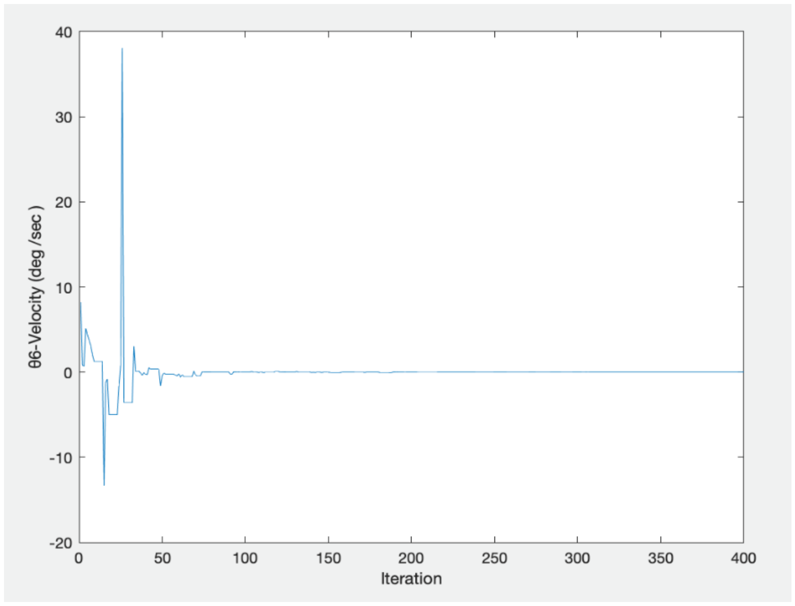


Figure 26:
Joint Velocity (deg/sec) for (θ_6) in Scenario 2 - Experiment 1

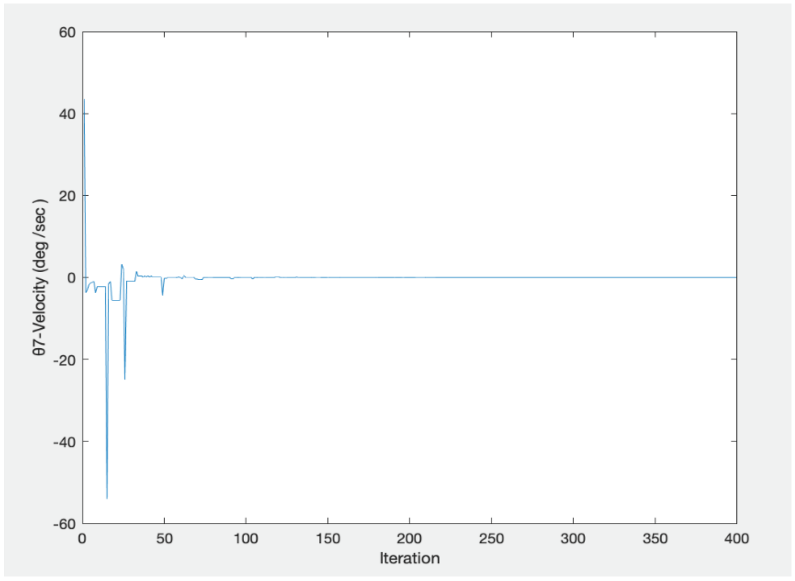


Figure 27:
Joint Velocity (deg/sec) for (θ_7) in Scenario 2 - Experiment 1

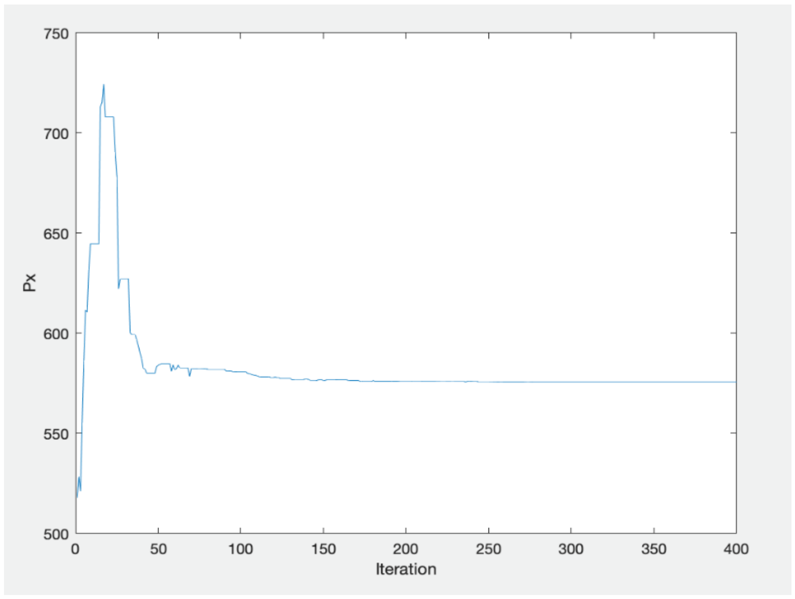


Figure 28:
Graph for Px (mm) in Scenario 2 - Experiment 1

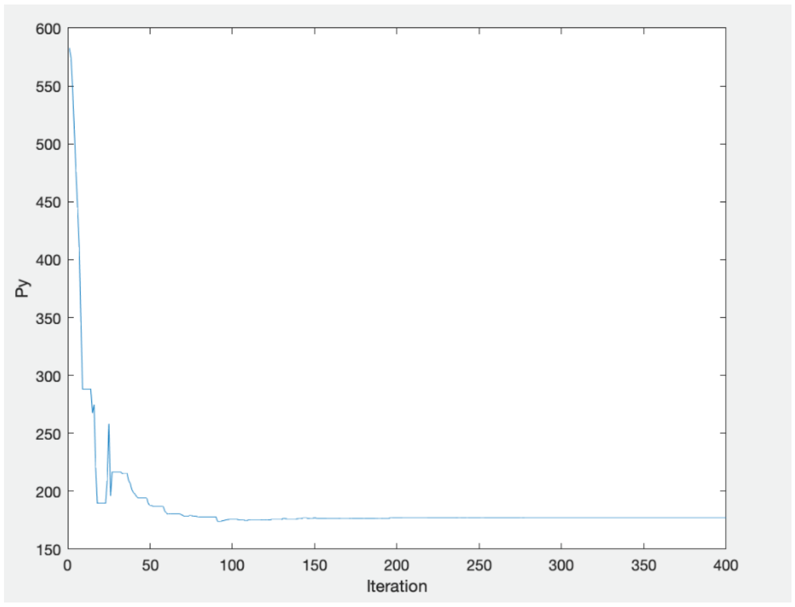


Figure 29:
Graph for Py (mm) in Scenario 2 - Experiment 1

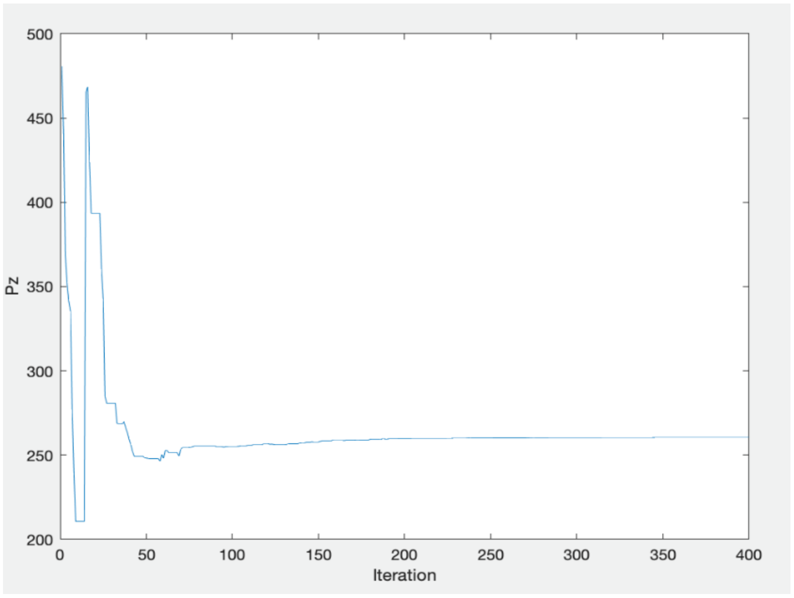


Figure 30:
Graph for Pz (mm) in Scenario 2 - Experiment 1

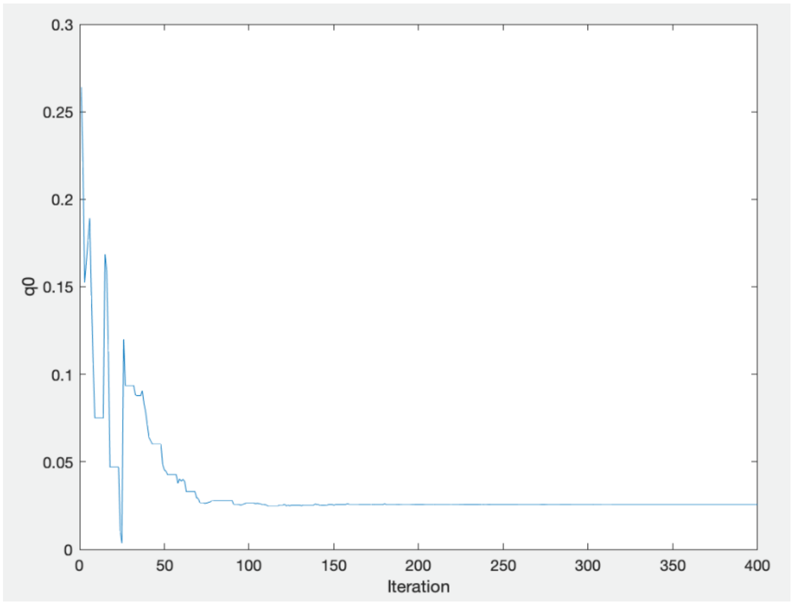


Figure 31:
Graph for q0 in Scenario 2 - Experiment 1

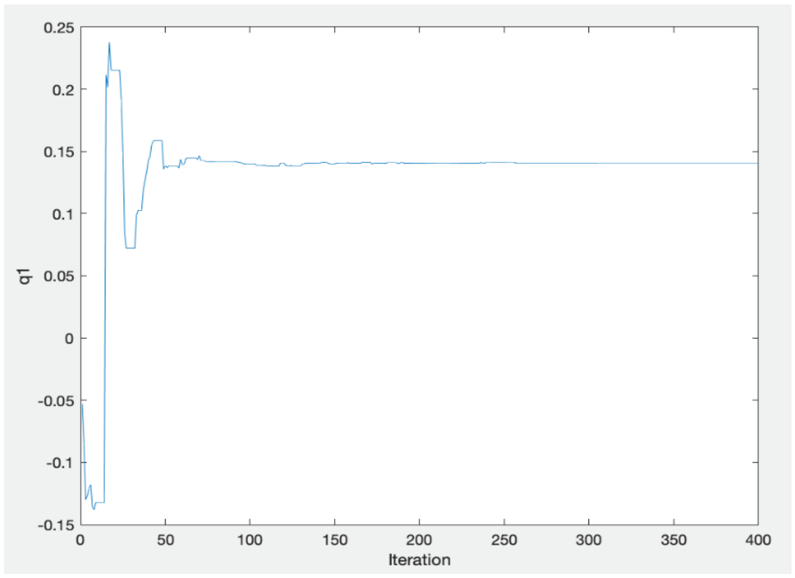


Figure 32:
Graph for q1 in Scenario 2 - Experiment 1

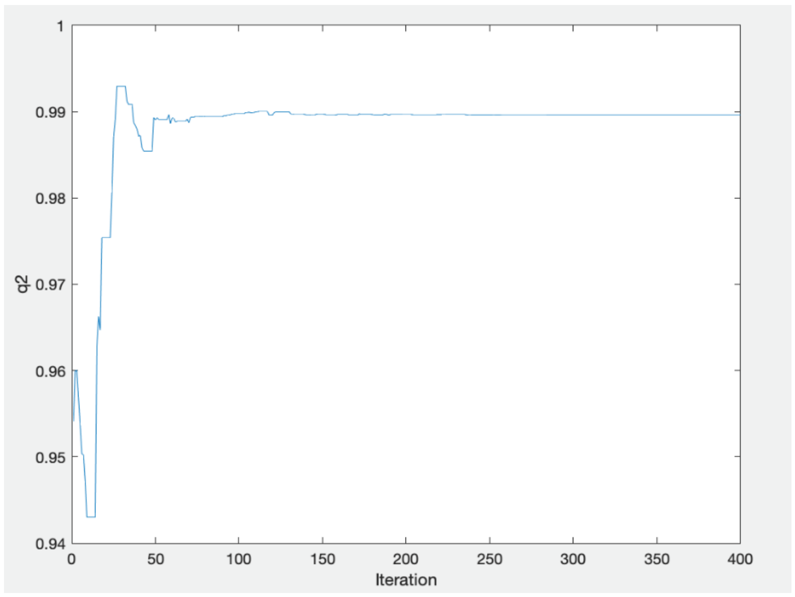


Figure 33:
Graph for q2 in Scenario 2 - Experiment 1

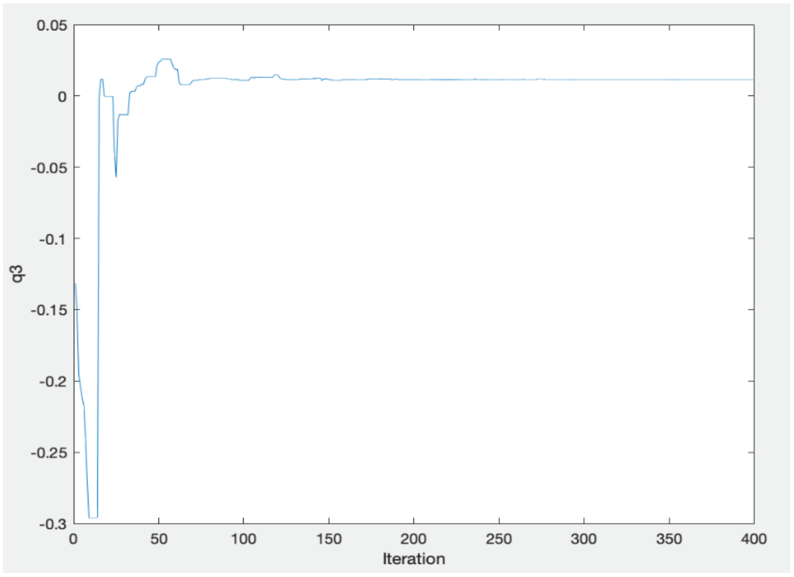
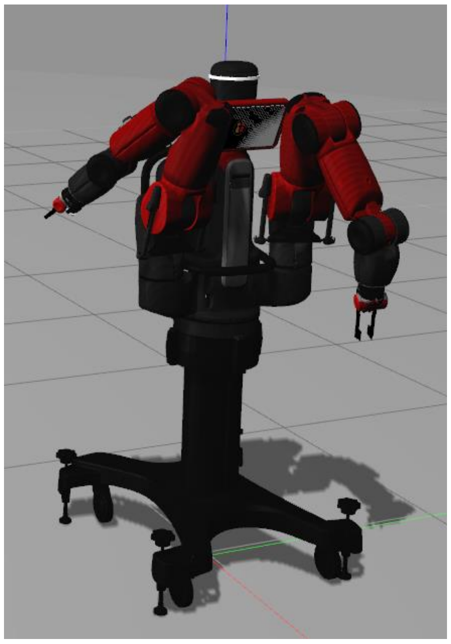
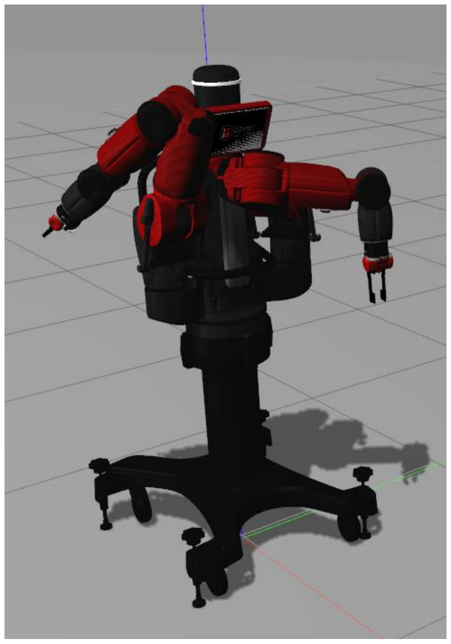


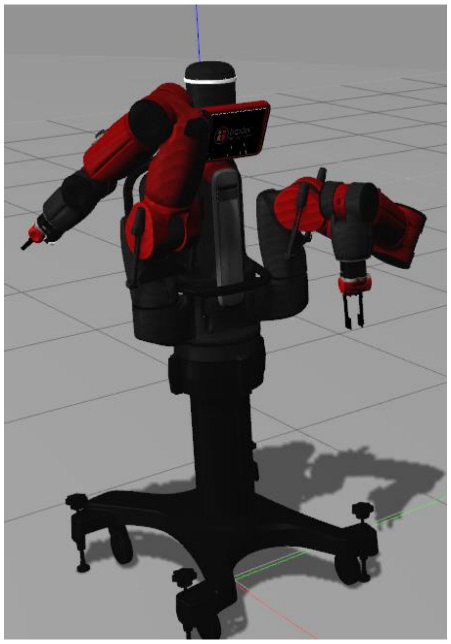
Figure 34:
Graph for q3 in Scenario 2 - Experiment 1



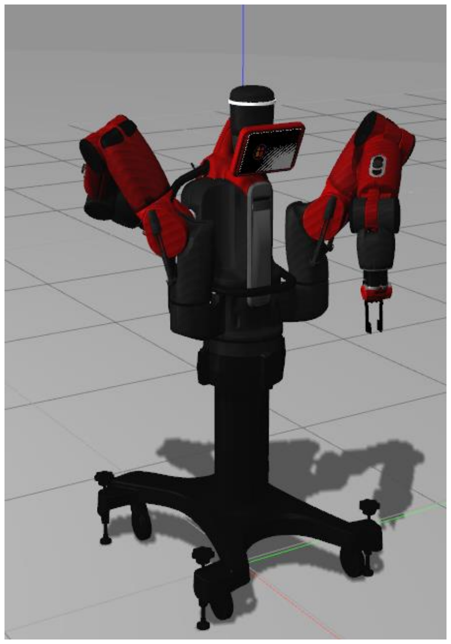
(a)



(b)



(c)



(d)

Figure 35:

Baxter with (a) first (b) second (c) third obtained angles and (d) first target joint angles

As shown in figure 13, we have the same pose, although each configuration is conspicuously different.

Experiment – 2

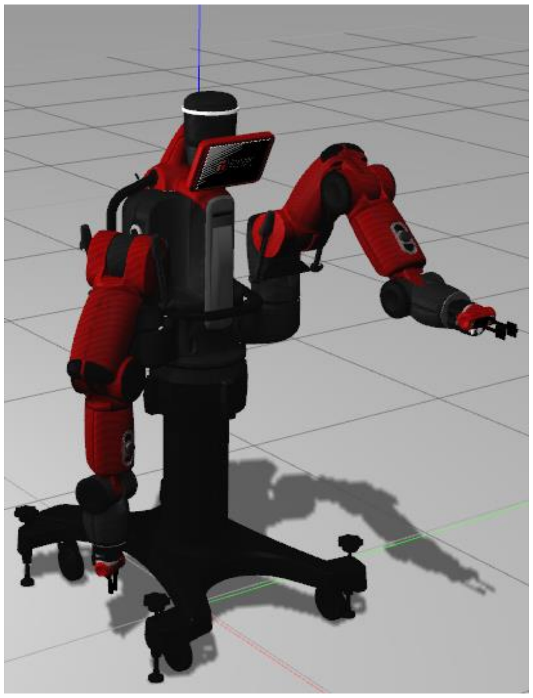


Figure 36:

How Baxter looks like with 2nd Setup Joint Angles

We used the same procedure with different target joint angles to validate the exactness and accuracy of the PSO method $i = 60,\ w_{damp} = 0.973,\ \ c1 = 1,\ \ c2 = 1,\ \ V_{initial} = 0.00001\, rad/sec\ \ \ w = 0.90.$

Joint angle values for three alternative solutions are shown in Table 10. The cost of derived solutions and the number of iterations are shown in Table 11. Figure 15 depicts the pose error graph of the three best options. As shown in Fig. 16, the proposed PSO approach efficiently searched for three unique robotic manipulator setups (a, b, c). The disparities between goal Fig.16(d) and achieved solutions are apparent.

Table 10:

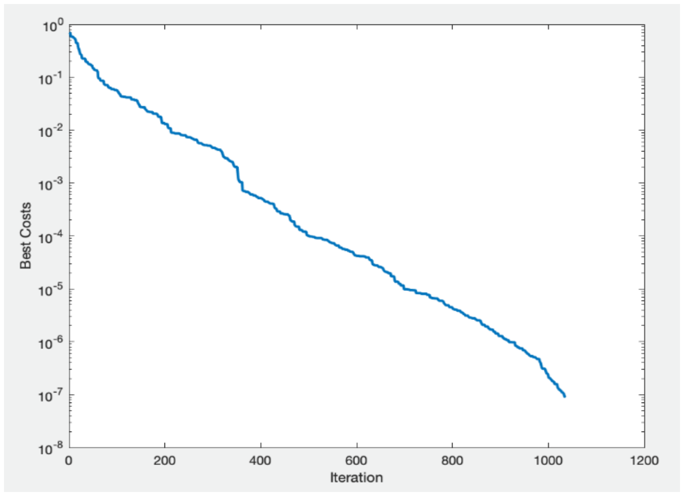
Target and Obtained Angle Values for first setup joint angles

Angle	Second Setup Joint Angles	First Obtained Angles	Second Obtained Angles	Third Obtained Angles
θ_1(rad)	- 0.7850	-1.0380	-0.6763	-1.0380
θ_2(rad)	- 0.7850	-0.7300	-0.7753	-0.7300
θ_3(rad)	0.0	0.3989	--0.1691	0.3989
θ_4(rad)	+ 1.5710	1.5510	1.5673	1.5510
θ_5(rad)	0.0	0.0287	-0.0145	0.0287
θ_6(rad)	- 0.7850	-0.7856	-0.7851	-0.7856
θ_7(rad)	0.0	-0.3140	0.1308	-0.3140

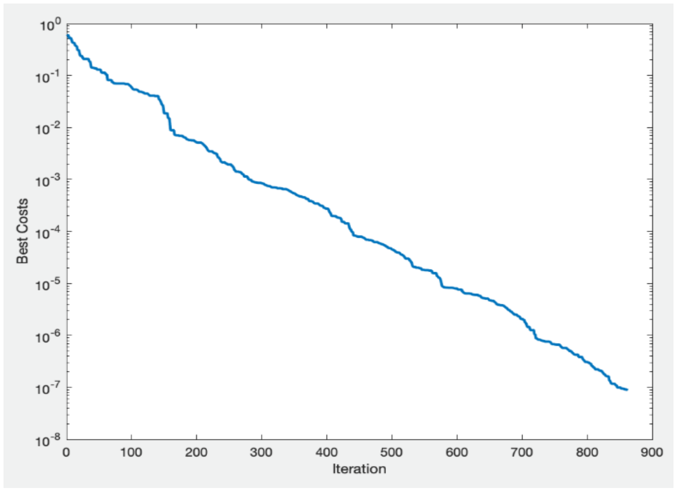
Table 11:

Simulation Results for 7 DOF first setup joint angles

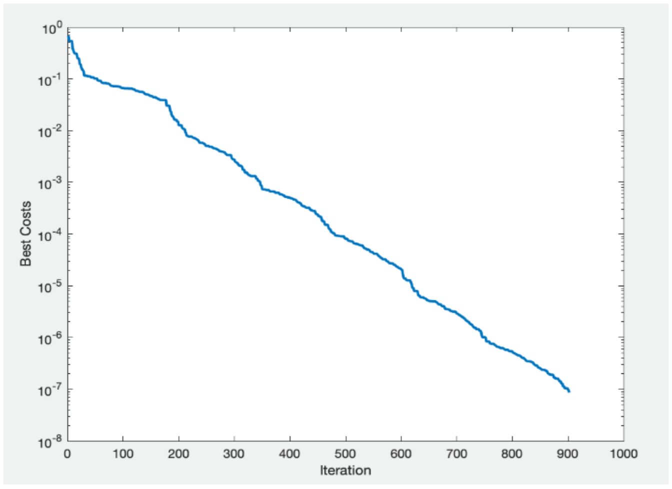
Solution	Euclidian Distance (Meter)	Number Of Iteration	Number of Particles
Solution 1	8.9193e-08	1036	60
Solution 2	8.7761e-08	862	60
Solution 3	8.6224e-08	902	60



(a)

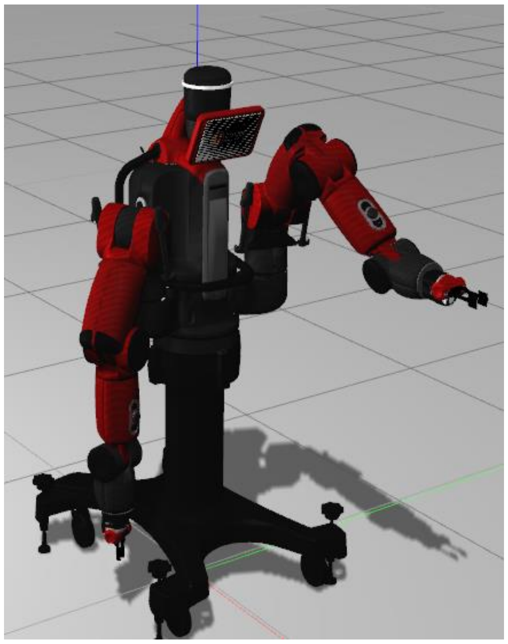


(b)

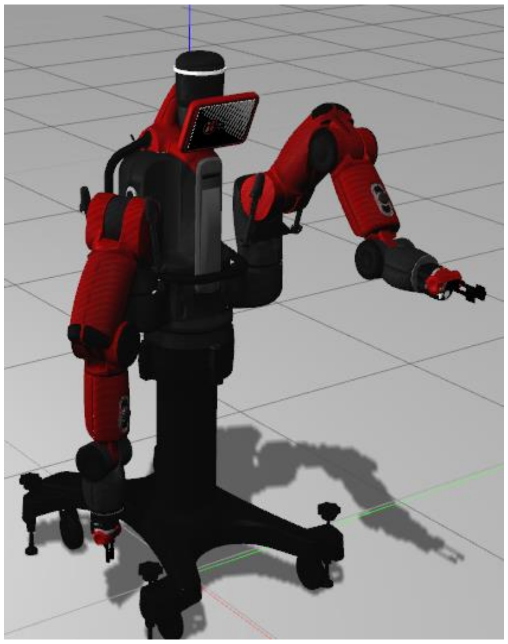


(c)

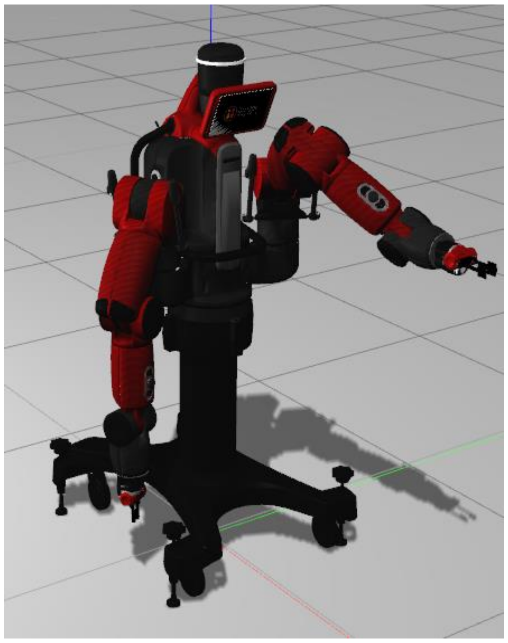
Figure 37:
Error's graphic of (a) first (b) second (c) third obtained angles for first setup joint angles



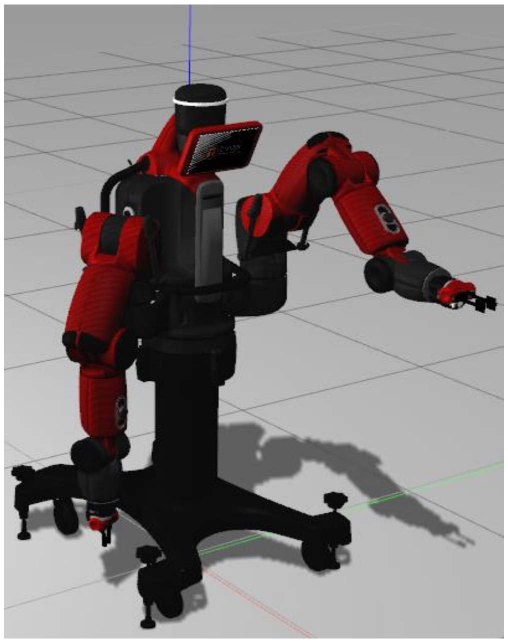
(a)



(b)



(c)



(d)

Figure 38:
Baxter with (a) first (b) second (c) third obtained angles and (d) second target joint angles

As shown in figure 16, we obtained three configurations that satisfy pose and lead the end effector to the exact location.

Obtained Result For Scenario 3

In this scenario, we added another joint to the continuation of Baxter's second wrist to demonstrate the efficacy of our method in solving the IK of an 8-DOF robot manipulator, as shown in Figure 17.

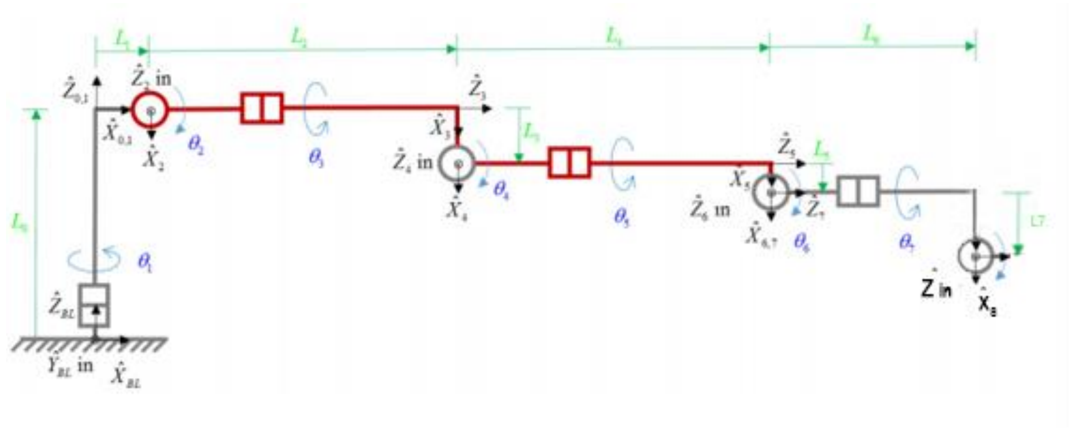


Figure 39:
8-DOF Baxter's Left Arm Cartesian Frame Assignment

According to the cartesian frame assignment (figure 17), we derive the DH table as follows.

Table 12*:*

DH Table for 8-DOF robot

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	-90	L_1	0	θ_2 +90
3	90	0	L_2	θ_3
4	-90	L_3	0	θ_4
5	90	0	L_4	θ_5
6	-90	L_5	*0*	θ_6
7	90	0	L_6	θ_7
8	-90	L_7	*0*	θ_8

Hence, we derived the DH table as follows. First, the length of the added joint is defined as 15mm.

Table 13:

8-DOF Baxter's Arm Length

Length	Value (mm)
L0	281.35
L1	125.00
L2	364.35
L3	69.00
L4	374.29
L5	10.00
L6	229.5
L7	15.00

The joint limit for the last attached joint is shown in table 12.

Table 14:

8-DOF left and right arm joint limits

Joint Name	Joint Variable	θ_i min	θ_i max	θ_i range
S_0	θ_1	+51°	-141°	192°
S_1	θ_2	+60°	-123°	183°
E_0	θ_3	+173°	-173°	346°
E_1	θ_4	+150°	-3°	153°
W_0	θ_5	+175°	-175°	350°
W_1	θ_6	+120°	-90°	210°
W_2	θ_7	+175°	-175°	350°
W_3	θ_8	+120°	-90°	210°

We practiced the algorithm with two different start settings to validate the exactness and accuracy in solving 8-DOF robot manipulator inverse kinematics $i = 60$, $\omega_{damp=0.973},\ c1=1,\ c2=1,\ V_{initial}=0.00001\left(\frac{m}{rad}\right),\ w=0.90$.

Obtained joint angle values for two setup angles are shown in Table 13 and Table 15. The cost of derived solutions and the number of iterations are shown in Table 14 and Table 16. Figures 18 and 19 depict the pose error graph of the experiments.

Experiment-1

Table 15:

Target and Obtained Angle Values for the first experiment

Angle	First Experiment Setup Joint Angles	Obtained Angles
θ_1(rad)	-1.5497	0.6483
θ_2 (rad)	-0.2269	0.2798
θ_3(rad)	1.3677	0.0859
θ_4(rad)	1.3381	1.6575
θ_5(rad)	-0.6322	2.3326
θ_6(rad)	1.9947	-1.5090
θ_7(rad)	0.9017	-2.2079
θ_8(rad)	0.1212	1.1826

Once we derive the pose for both angles, we get the same solution as [0.1607 0.3693 -0.0854 0.2302 -0.2514 -0.8489 0.4040**]**

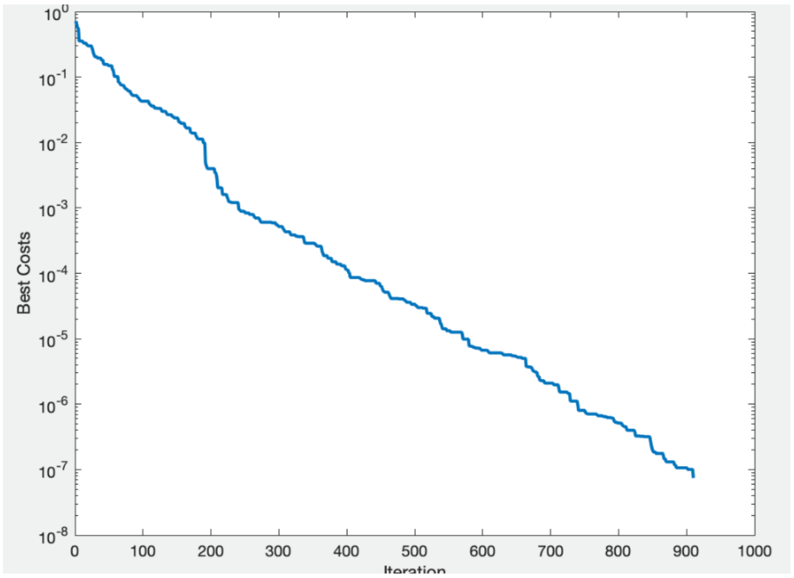


Figure 40:
Error graphic of the first experiment

Table 16:

Simulation result for the first experiment

Solution		Euclidian Distance (Meter)	Number Of Iteration	Number of Particles
Solution		7.4475-08	910	80

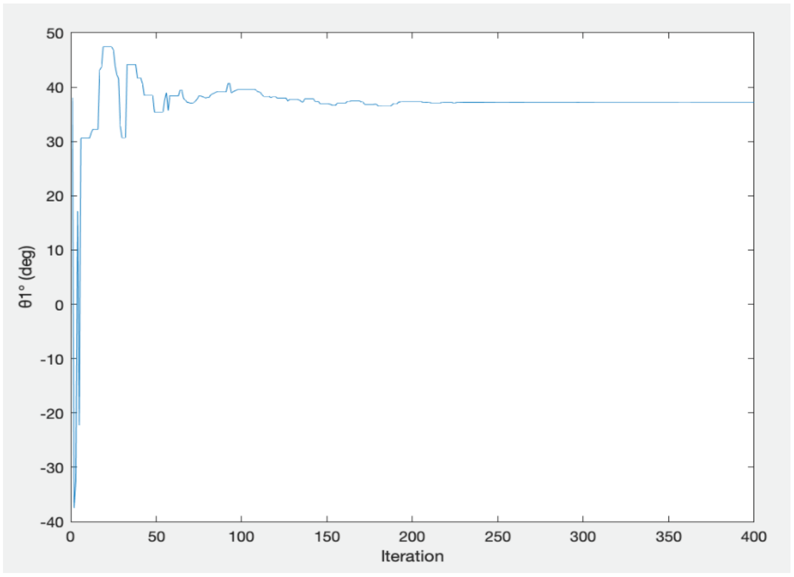


Figure 41:
Joint Angle (θ_1) Graph in Scenario 3 - Experiment 1

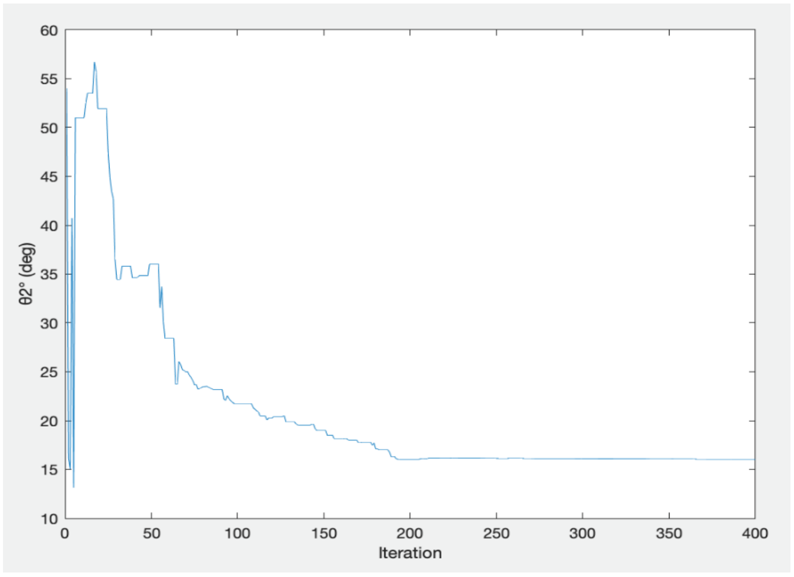


Figure 42:
Joint Angle (θ_2) Graph in Scenario 3 - Experiment 1

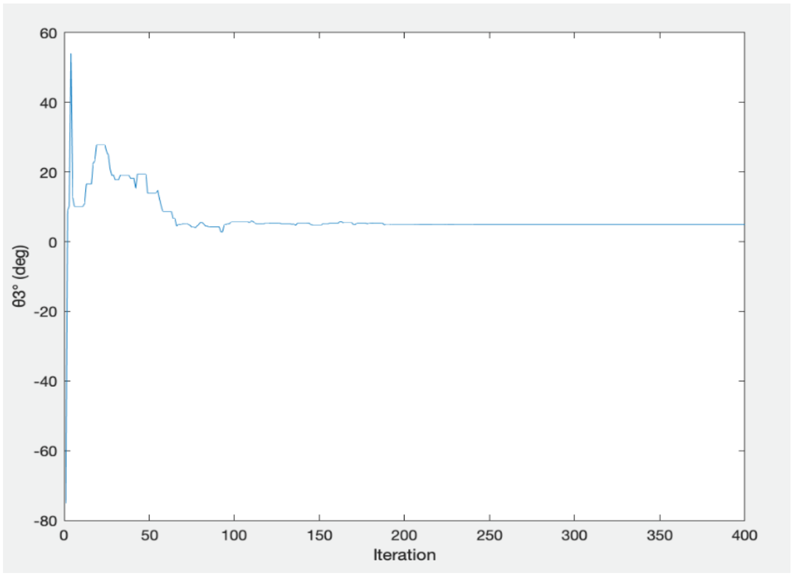


Figure 43:
Joint Angle (θ_3) Graph in Scenario 3 - Experiment 1

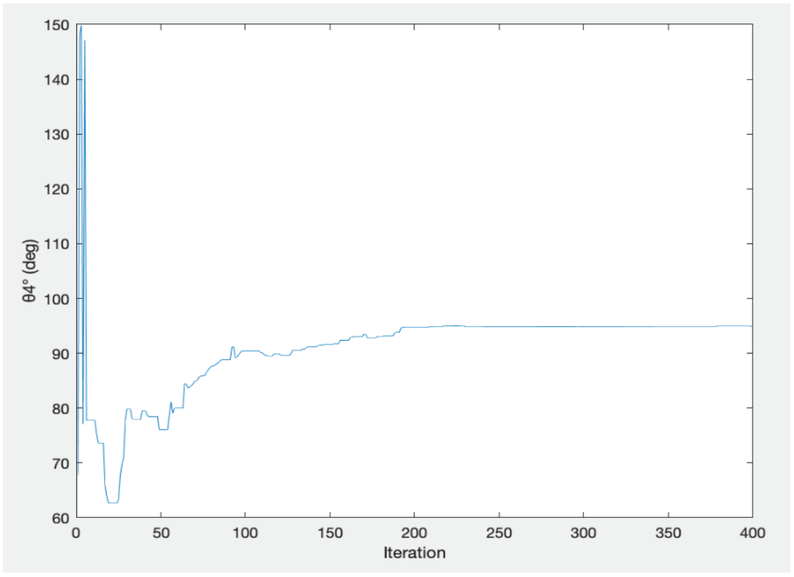


Figure 44:
Joint Angle (θ_4) Graph in Scenario 3 - Experiment 1

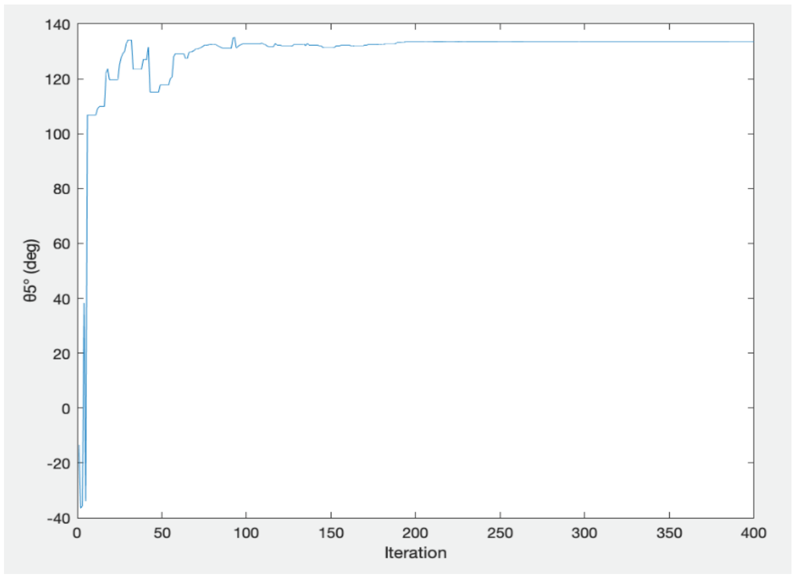


Figure 45:
Joint Angle (θ_5) Graph in Scenario 3 - Experiment 1

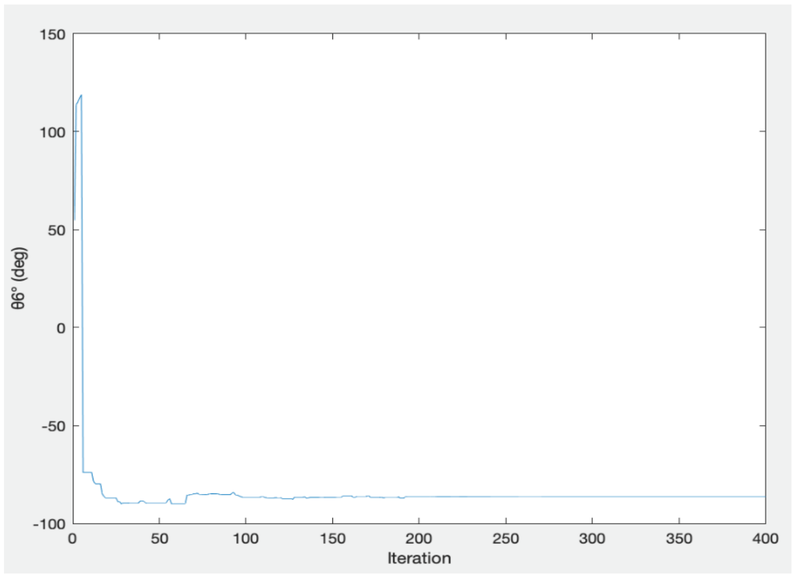


Figure 46:
Joint Angle (θ_6) Graph in Scenario 3 - Experiment 1

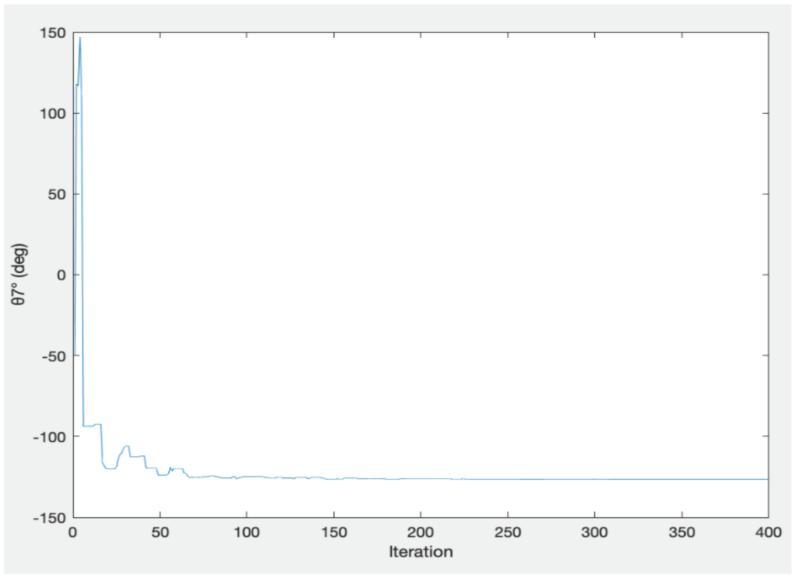


Figure 47:
Joint Angle (θ_7) Graph in Scenario 3 - Experiment 1

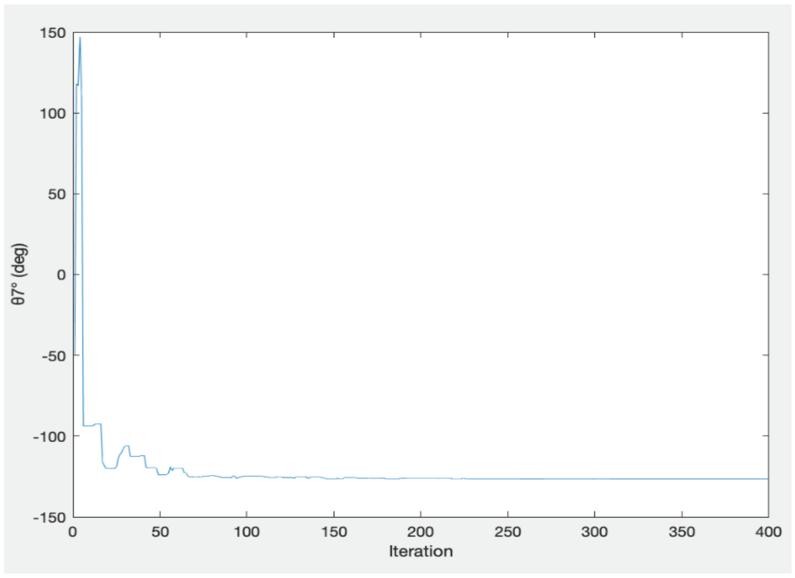


Figure 48:
Joint Angle (θ_8) Graph in Scenario 3 - Experiment 1

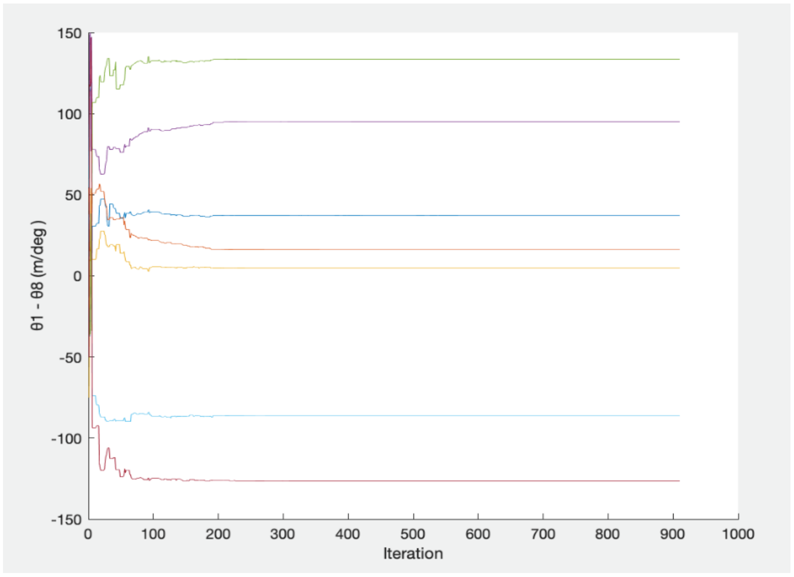


Figure 49:
All Joint Angle Graph in Scenario 3 - Experiment 1

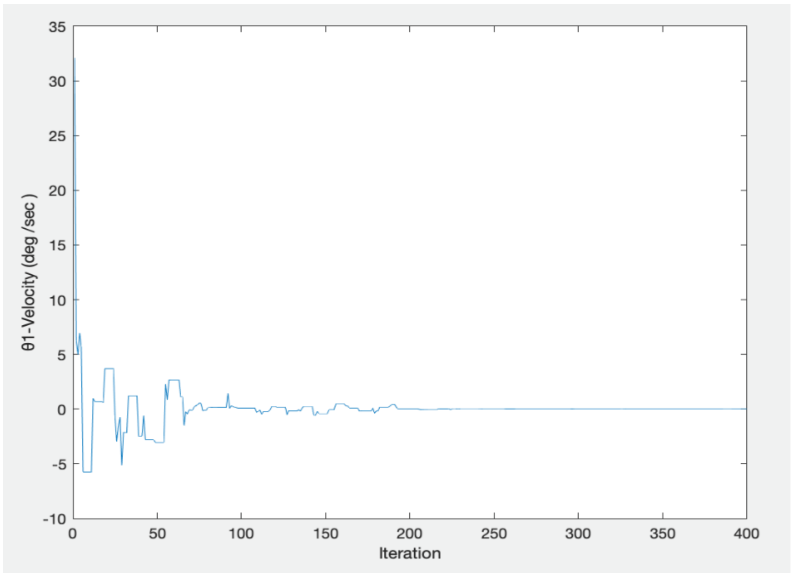


Figure 50:
Joint Velocity (deg/sec) for $\theta_1$in Scenario 3 - Experiment 1

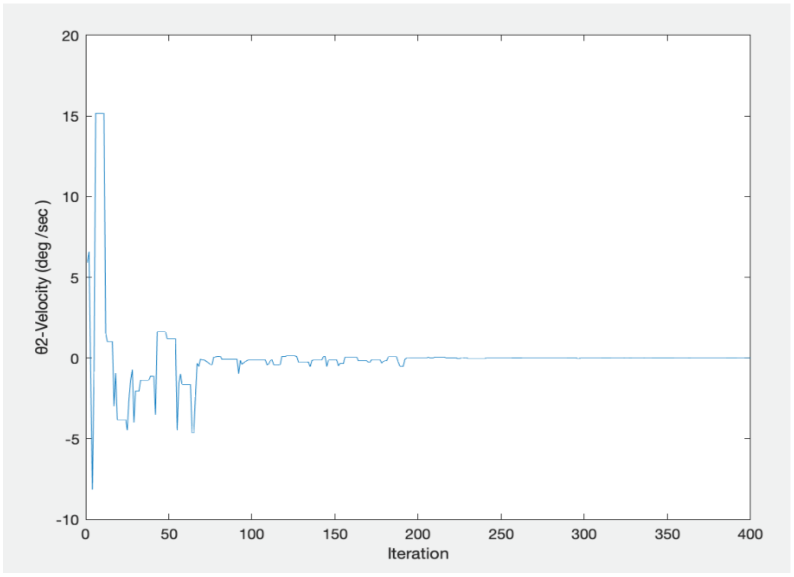


Figure 51:
Joint Velocity (deg/sec) for θ_2 in Scenario 3 - Experiment 1

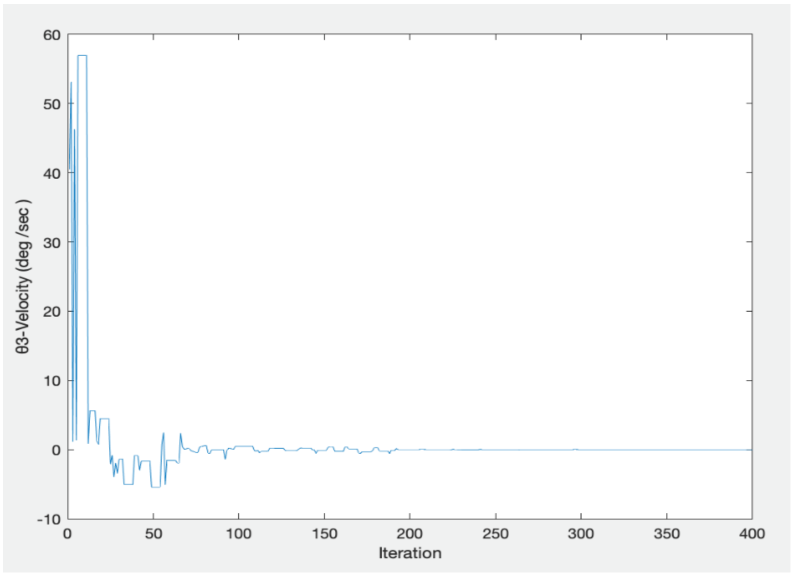


Figure 52:
Joint Velocity (deg/sec) for θ_3 in Scenario 3 - Experiment 1

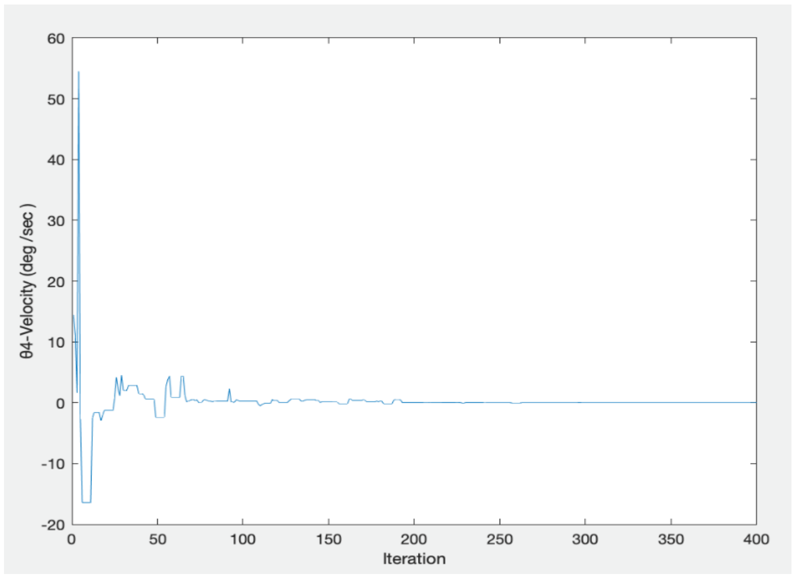


Figure 53:
Joint Velocity (deg/sec) for θ_4 in Scenario 3 - Experiment 1

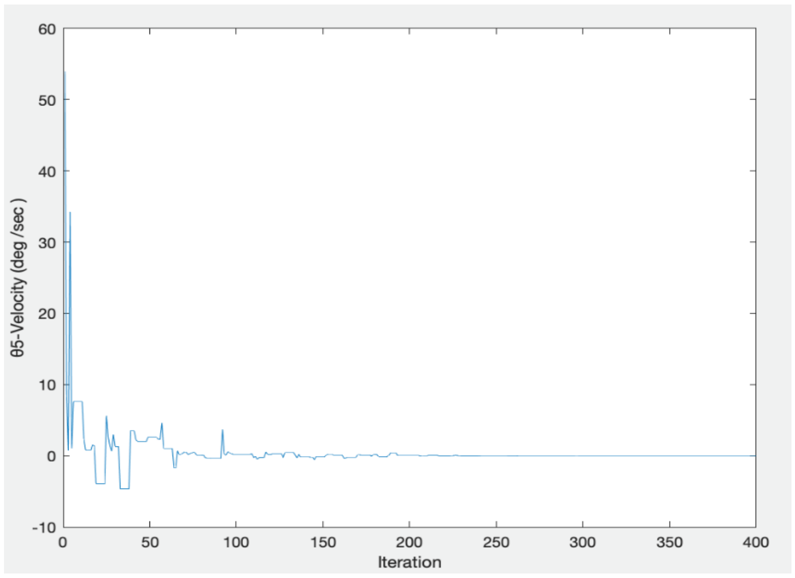


Figure 54:
Joint Velocity (deg/sec) for θ_5 in Scenario 3 - Experiment 1

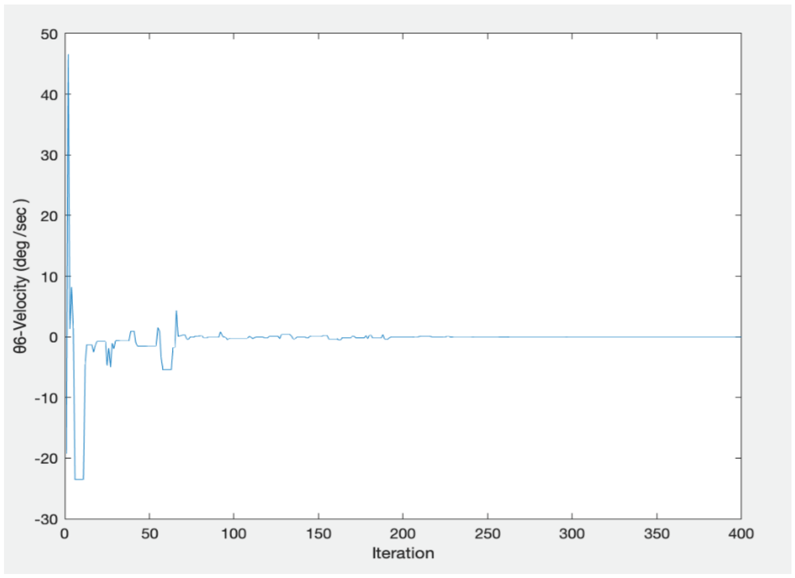


Figure 55:
Joint Velocity (deg/sec) for θ_6 in Scenario 3 - Experiment 1

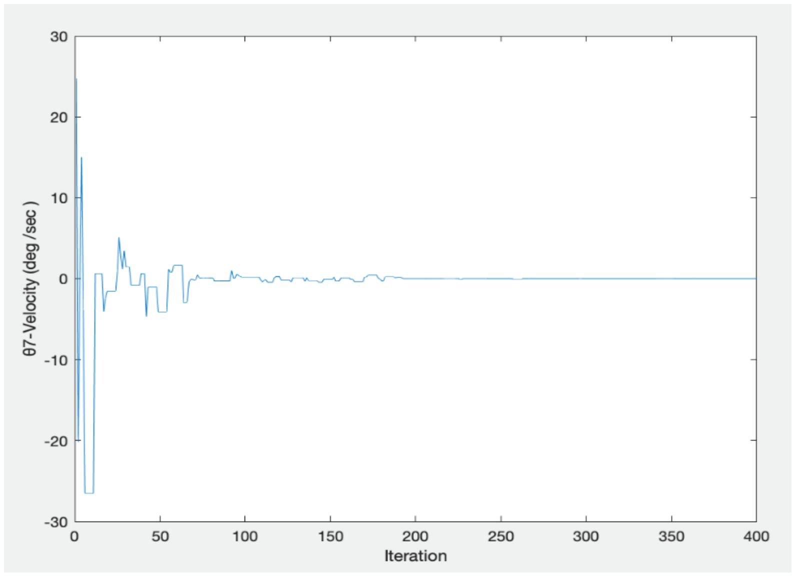


Figure 56:
Joint Velocity (deg/sec) for θ_7 in Scenario 3 - Experiment 1

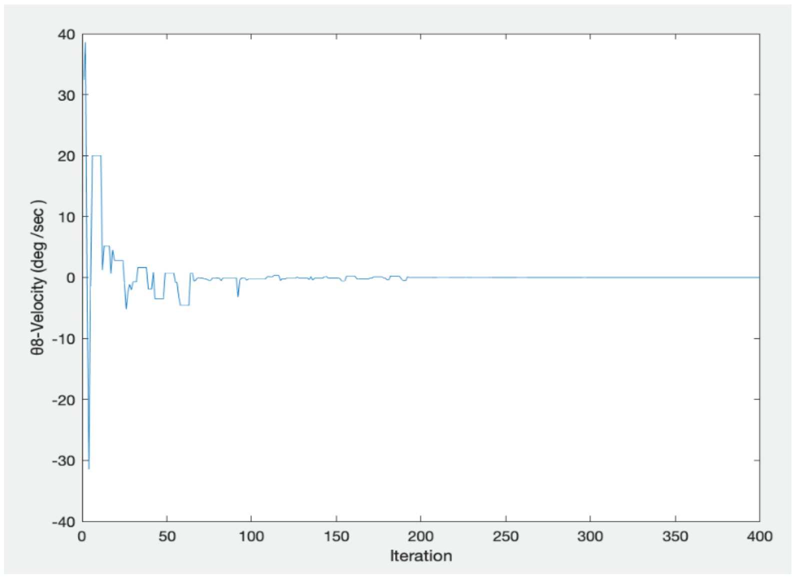


Figure 57:
Joint Velocity (deg/sec) for θ_8 in Scenario 3 - Experiment 1

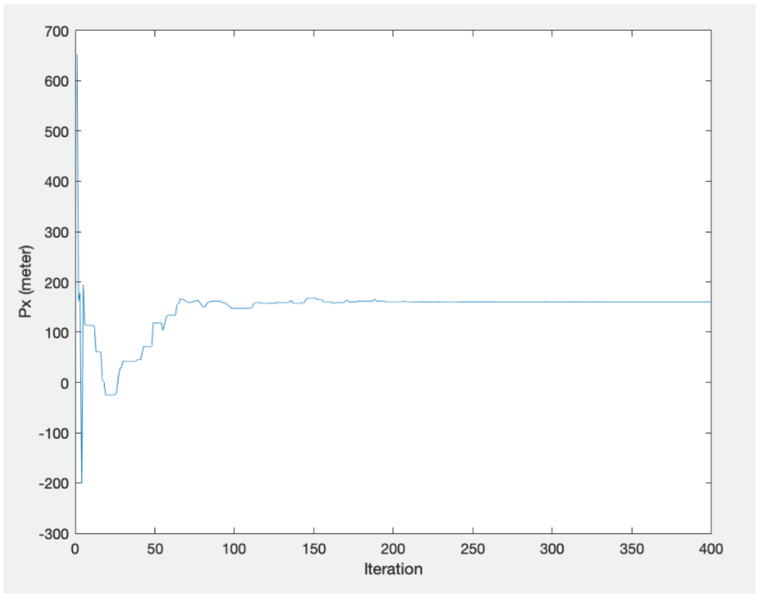


Figure 58:
Graph for Px (mm) in Scenario 3 - Experiment 1

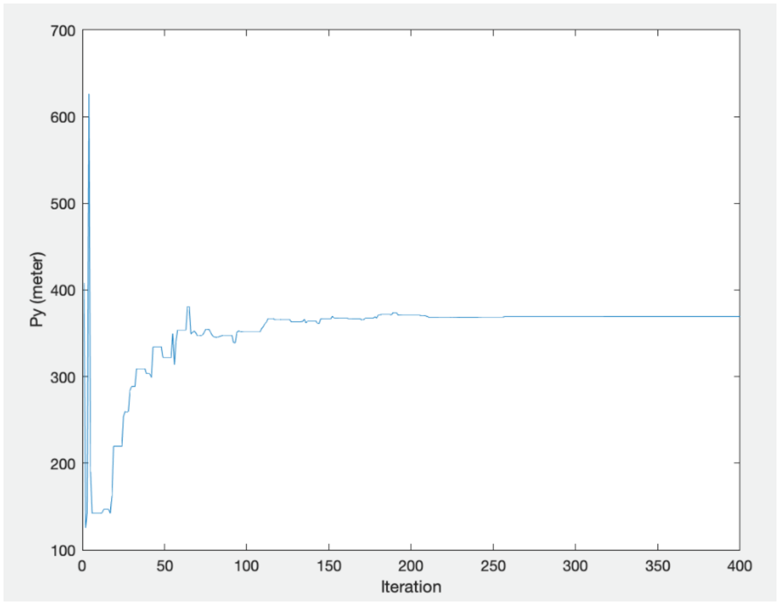


Figure 59:
Graph for Py (mm) in Scenario 3 - Experiment 1

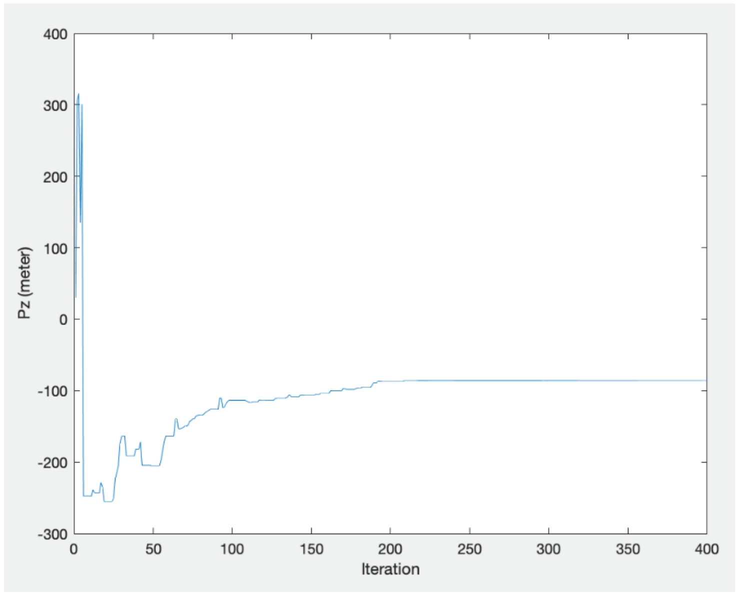


Figure 60:
Graph for Pz (mm) in Scenario 3 - Experiment 1

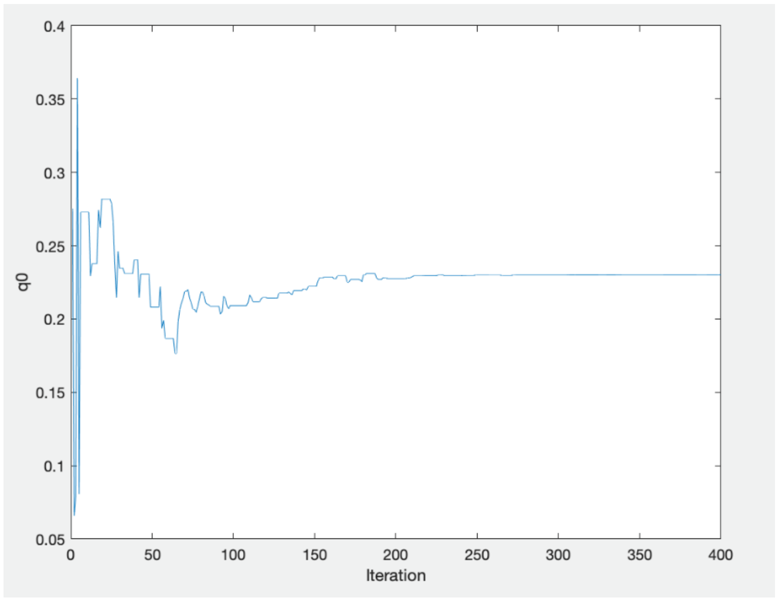


Figure 61:
Graph for q0 in Scenario 3 - Experiment 1

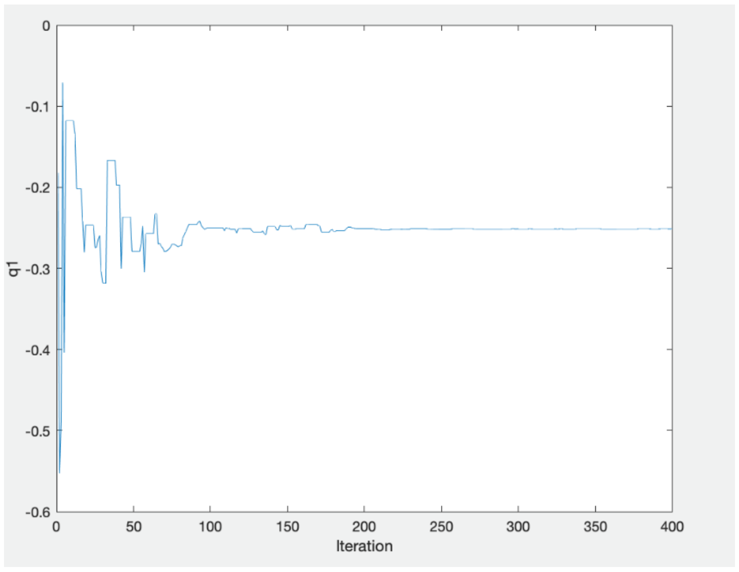


Figure 62:
Graph for q1 in Scenario 3 - Experiment 1

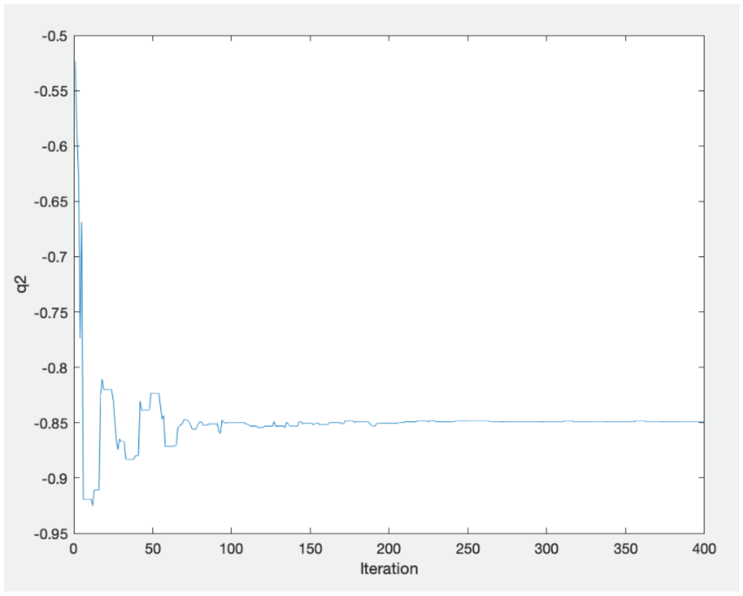


Figure 63:
Graph for q2 in Scenario 3 - Experiment 1

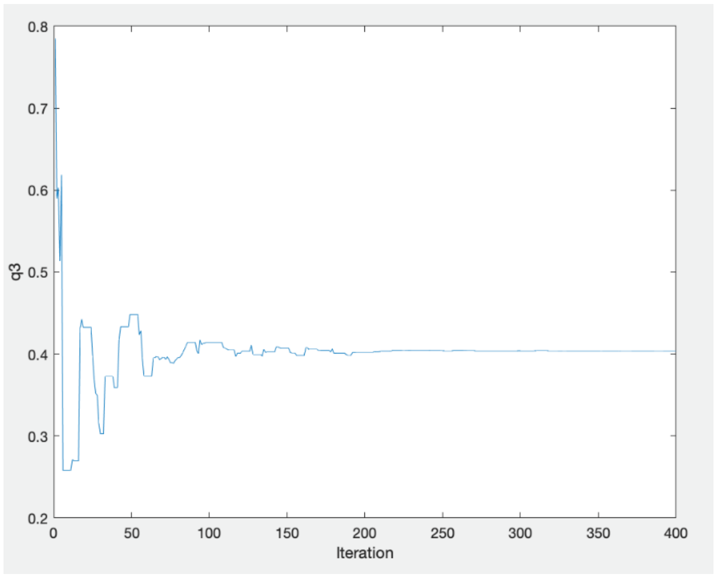


Figure 64:
Graph for q3 in Scenario 3 - Experiment 1

Experiment-2

Table 17:

Target and Obtained Angle Values for the second experiment

Angle	First Experiment Setup Joint Angles	Obtained Angles
θ_1(rad)	-0.08	-1.5497
θ_2 (rad)	-1.00	-0.2269
θ_3(rad)	-1.19	1.3677
θ_4(rad)	+1.94	1.3381
θ_5(rad)	+0.67	-0.6322
θ_6(rad)	+1.03	1.9947
θ_7(rad)	- 0.50	0.9017
θ_8(rad)	-0.08	0.1212

Once we derive pose for both angles by using forward kinematics, we get the same solution as [0.5614 0.1812 0.2598 0.2464 -0.4116 -0.5617 -0.6741]

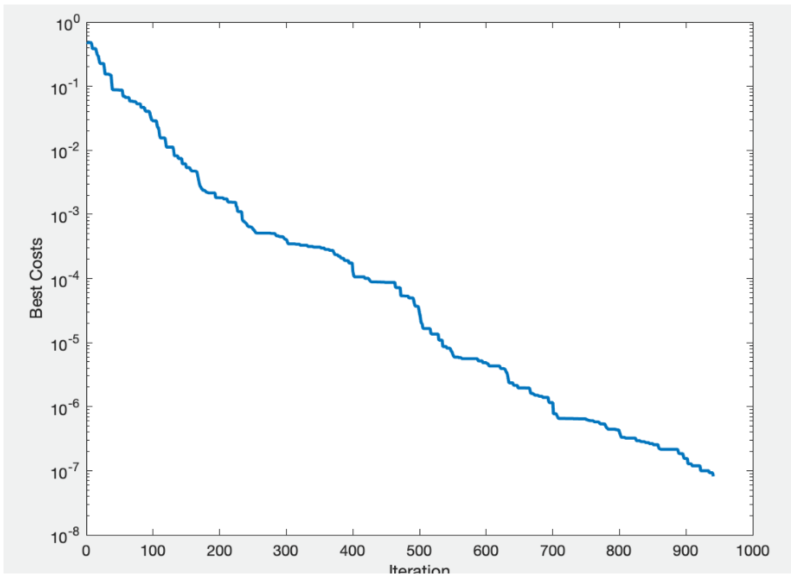


Figure 65:
Error Graphic of the second experiment

Table 18*:*

Simulation result for the second experiment

Solution	Euclidian Distance (Meter)	Number Of Iteration	Number of Particles
Solution	8.1868e-08	941	80

Test On Baxter in Real

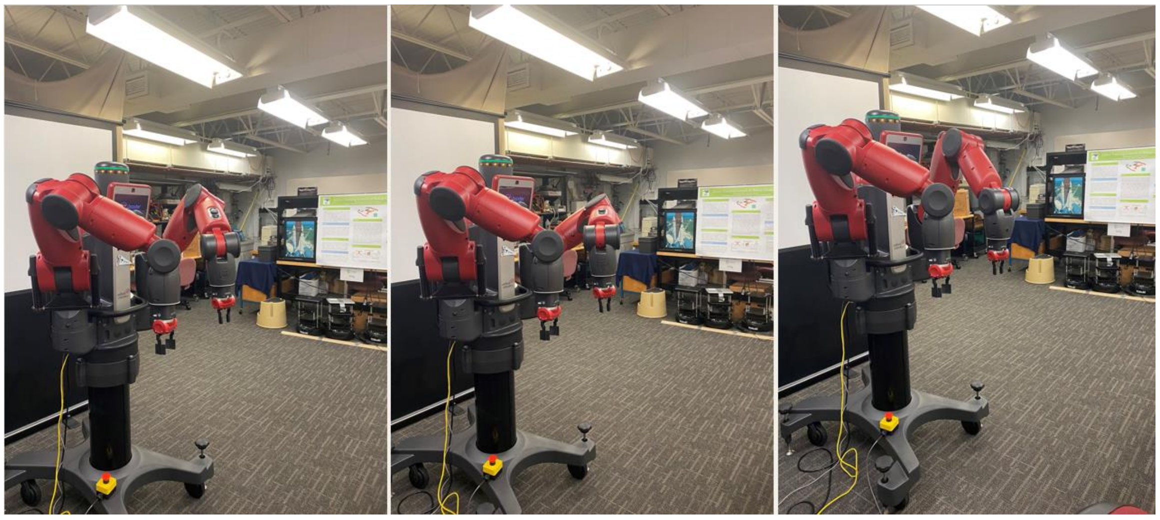


Figure 66:
Demonstration in real -1

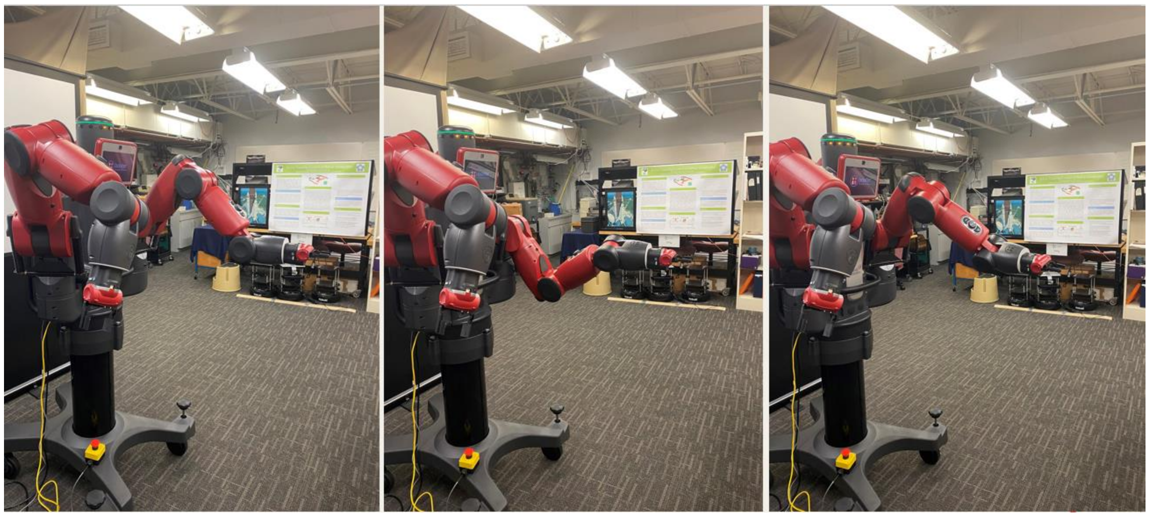


Figure 67:
Demonstration in real -2

Conclusion

In this study, the meta-heuristic method, the Particle Swarm Optimization algorithm, proved its effectiveness in solving the inverse kinematics of any serial robot manipulator. The algorithm has been tested under several different concepts and demonstrated on the actual Baxter robot and the simulation using gazebo robot simulation. Each run ended up with a solution that moves the end-effector accurately to any desired pose in cartesian space by converging position and orientation with the PSO algorithm. The utility of quaternion representation for manipulators demonstrates that solving IK using quaternion results in a definite pose solution with no ambiguity neither gimbal lock. Our approach to solving the IK problem for any serial manipulator always gives a possibility of attaining a different set of joint angles that satisfy error criteria. IK calculations of 6DOF, 7DOF, and 8DOF serial robot manipulators have been performed successfully. The result confirms that PSO is highly effective in getting the optimal IK solution for any serial robotic manipulators.

REFERENCES

[1] Paul, Richard (1981). Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators. Cambridge, MA: MIT Press. ISBN 978-0-262-16082-7. Archived from the original on 2017-02-15. Retrieved 2016-09-22.

[2] Çavdar T, Alavi M (2012) A new heuristic approach for inverse kinematics of robot arms. J Comput Theor Nanosci 19:329–333

[3] Momani SM, Abo-Hammour Z, Alsmadi O (2016) Solution of inverse kinematics problem using genetic algorithms. Appl Math Inf Sci 10:1–9

[4] Beni, G., Wang, J., 1993, Swarm Intelligence in Cellular Robotic Systems, Proceed, NATO Advanced Workshop on Robots and Biological Systems, Tuscany-Italy.

[5] Marco Dorigo and Mauro Birattari (2007), Scholarpedia, 2(9):1462.

[6] Golbon-Haghighi, Mohammad-Hossein; Saeidi-Manesh, Hadi; Zhang, Guifu; Zhang, Yan (2018). "Pattern Synthesis for the Cylindrical Polarimetric Phased Array Radar (CPPAR)" (PDF). Progress in Electromagnetics Research. ***66****: 87–98. doi:10.2528/PIERM18011016 (inactive 2021-01-15).*

[7] R. Eberhart & J. Kennedy, A New Optimizer Using Particle Swarm Theory, Sixth International Symposium on Micro Machine and Human Science.

[8] J. Kennedy, R.C. Eberhart, et al., "Particle swarm optimization," In Proceedings of IEEE international conference on neural networks, volume 4, pages 1942–1948. Perth, Australia, 1995.

[9] Zhang, Y. (2015). "A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications." Mathematical Problems in Engineering. 2015: 931256.

[10] Malaysian Technical Universities Conference on Engineering & Technology 2012, MUCET 2012 Part 4 Information And Communication Technology An Overview of Particle Swarm Optimization Variants Muhammad Imran[a]*, *, Rathiah Hashima and Noor Elaiza Abd Khali*

[11] Y. Shi and R. Eberhart., "A modified particle swarm optimizer," In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, pages 69–73. IEEE, 2002.

[12] J. Xin, G. Chen, and Y. Hai., "A Particle Swarm Optimizer with Multistage Linearly-Decreasing Inertia Weight," In Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on, volume 1, pages 505–508. IEEE, 2009.

[13] Bansal, J. C., Singh, P. K., Saraswat, M., Verma, A., Jadon, S. S., & Abraham, A. (2011). Inertia Weight strategies in Particle Swarm Optimization. 2011 Third World Congress on Nature and Biologically Inspired Computing.

[13] Bansal, J., Saraswat, M., Jadon, S., Abraham, A., 2011, Inertia Weight Strategies in PSO, 3rd World Congress on Nature and Biologically Inspired Computing (NABIC), 640, Salamanca-Spain.

[14] W. Al-Hassan, MB Fayek, and SI Shaheen, "Psosa: An optimized particle swarm technique for solving the urban planning problem," In Computer Engineering and Systems, The 2006 International Conference on, pages 401–405. IEEE, 2007

[15] R.F. Malik, T.A. Rahman, S.Z.M. Hashim, and R. Ngah, "New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight," International Journal of Computer Science and Security (IJCSS), 1(2):35, 2007.

[16] Masrom, S., Moser, I., Montgomery, J., Abidin, S. Z. Z., & Omar, N. (2013). Hybridization of Particle Swarm Optimization with adaptive genetic algorithm operators. 2013 13th International Conference on Intelligent Systems Design and Applications.

[17] Zhan, Z., Zhang, J., Li, Y., Chung, H., (2009), Adaptive Particle Swarm Optimization, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 39(6), 1380.

[18] Kennedy, J., (1997), "The Particle Swarm: Social Adaptation of Knowledge." Proceedings of IEEE Intl. Conference on Evolutionary Computation, p 305.

[19] Tu, Z., Lu, Y., 2008, Corrections to A Robust Stochastic Genetic Algorithm (SGA) for Global Numerical Optimization, IEEE Transactions on Evolutionary Computation, Vol. 12

[20] Kennedy, J., (2003). "Bare Bones Particle Swarms." Proceedings of the 2003 IEEE Swarm Intelligence Symposium

[21] "Gazebo" Gazebo Simulator. Archived from the original on 2018-01-16. Retrieved 2019-0

[22] "Gazebo Simulator for DARPA Virtual Robotics Challenge." YouTube. DARPA. 2016-02-04. Retrieved 2013-06-03.

[23] Kennedy, J., Eberhart, R., 1997. A discrete binary version of the particle swarm algorithm. In: Proceedings of 1997 IEEE International Conference on Systems, Man, and Cybernetics. Vol. 5. pp. 4104-4108.

[24] Cai, Q., Gong, M., Shen, B., Ma, L., & Jiao, L. (2014). Discrete particle swarm optimization for identifying community structures in signed social networks. Neural Networks, 58, 4–13.

[25] Marco, D., Thomas, S., Ant Colony Optimization, 2004, A Bradford Book, The MIT Press, Cambridge, London, England.

[26] Wiak, S., Krawczyk, A., & Dolezel, I. (Eds.). (2008). Intelligent Computer Techniques in Applied Electromagnetics. Studies in Computational Intelligence. doi:10.1007/978-3-540-78490-6

[27] Mitchell, Melanie (1996). An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press. ISBN 9780585030944.

[28] Whitley, Darrell (1994). "A genetic algorithm tutorial" (PDF). Statistics and Computing. ***4*** *(2): 65–85.*

[29] Genetic algorithm - Wikipedia. https://en.wikipedia.org/wiki/Genetic_algorithm

[30] Introduction to Genetic Algorithms –

https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code e396e98d8bf3#:~:text=A%20genetic%20algorithm%20is%20a,offspring%20of%20the%20next%20generation.

[31] Pham, DT, Ghanbarzadeh A, Koc E, Otri S, Rahim S and Zaidi M. The Bees Algorithm. Technical Note, Manufacturing Engineering Centre, Cardiff University, UK, 2005.

[32] Pham, D.T., Castellani, M. (2009), The Bees Algorithm – Modelling Foraging Behaviour to Solve Continuous Optimisation Problems. Proc. ImechE, Part C, 223(12), 2919-2938.

[33] Pham, D.T. and Castellani, M. (2013), Benchmarking and Comparison of Nature-Inspired Population-Based Continuous Optimisation Algorithms, Soft Computing, 1-33.

[34] Pham, D.T. and Castellani, M. (2015), A comparative study of the bees algorithm as a tool for function optimization, Cogent Engineering 2(1), 1091540.

[35] Whitaker, R., Hurley, S., An Agent-based Approach to Site Selection for Wireless Networks, Proc ACM Symposium on Applied Computing, 2002.

[36] Miller, P., 2010, The Smart Swarm: How Understanding Flocks, Schools, and Colonies can make us better at Communicating

[37] Rosenberg, L., Willcox, G., Pescetelli, N., October 2017, Artificial Swarm Intelligence amplifies Accuracy when Predicting Financial Markets, 2017 IEEE 8th Annual Ubiquitous Computing, Electronics, and Mobile Communication Conference

[38] Dorigo M.,Birattari M.(2007). Swarm intelligence. Scholarpedia 2:1462 10.4249 / scholarpedia . 1462

[39] Hamann H., Schmickl T. (2012). Modeling the swarm: analyzing biological and engineered swarm systems. Math. Comput. Modell. Dyn. Syst. 18, 1–12.

[40] Swarm Robotic Behaviors and Current Applications Melanie Schranz1, Martina Umlauft1, Micha Sende1, Wilfried Elmenreichs 2020, Decision Making, and Getting Things Done, NY: Avery.

[41] Sharkey A. J. (2007). Swarm robotics and minimalism. Connect. Sci. 19, 245–260.

[42] "The Kilobot Project." Harvard Self-organizing Systems Research Group. Retrieved October 29, 2014.

[43]snasa.gov/directorates/spacetech/niac/2018_Phase_I_Phase_II/Marsbee_Swarm_of _Flapping_Wing_Flyers_for_Enhanced_Mars_Exploration/

[44] http://www.navaldrones.com/CARACAS.html

[45] "DoD ramps micro-drones after successful 'swarm' test." www.defensesystems.com. Defense Systems. Archived from the original on 2017-09-04. Retrieved September 3, 2017.

[46] Jump up to [a b c d e f g] "Microsoft Word - Perdix Fact Sheet (01062017 Final)" (PDF). www.defense.gov. United States Department of Defense. Archived (PDF) from the original on 2017-01-10. Retrieved January 14, 2017.

[47] Jump up to [c] "US military tests swarm of mini-drones launched from jets." www.bbc.co.uk. BBC. Archived from the original on 2017-01-13. Retrieved January 14, 2017.

[48] Knight, W., 2012, This Robot Could Transform Manufacturing, MIT Technology Review.

[49] Denavit, Jacques; Hartenberg, Richard Scheunemann (1955). "A kinematic notation for lower-pair mechanisms based on matrices." Trans ASME J. Appl. Mech. 23: 215–221.

[50] Hartenberg, Richard Scheunemann; Denavit, Jacques (1965). Kinematic synthesis of linkages. McGraw-Hill series in mechanical engineering. New York: McGraw-Hill. p. 435. Archived from the original on 2013-09-28. Retrieved 2012-01-13.

[51] Forward Kinematics. https://www.rosroboticslearning.com/forward-kinematics

[52] Robot control part 2: Jacobians, velocity, and force https://studywolf.wordpress.com/2013/09/02/robot-control-jacobians-velocity-and-force/

[53] Jacobian. https://www.rosroboticslearning.com/jacobian

[54] Donald L. Pieper, The kinematics of manipulators under computer control. Ph.D. thesis, Stanford University, Department of Mechanical Engineering, October 24, 1968.

[55] https://rahulbhadani.github.io/docs/QueternionAndEuler.pdf

[56] Eberhart, R., Kennedy, J., 1995, Particle Swarm Optimization, Swarm Intelligence, EEE Intl Conference on Neural Networks, 1942, Perth, Australia.

[57] A Swarm of Insight, 2019 Radiology Today, retrieved from www.radiologytoday.net

[58] Dereli, S., & Köker, R. (2020). A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm. *Artificial Intelligence Review*, *53*(2), 949-964.

[59] Ayyıldız, M., & Çetinkaya, K. (2015). Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-DOF serial robot manipulator. Neural Computing and Applications, 27(4), 825–836. doi:10.1007/s00521-015-1898-8

[60] Sancaktar, I., Tuna, B., & Ulutas, M. (2018). Inverse kinematics application on the medical robot using adapted PSO method. Engineering Science and Technology, an International Journal. doi:10.1016/j.jestch.2018.06.011

[61] Rokbani, N., & Alimi, A. M. (2013). Inverse kinematics using particle swarm optimization, a statistical analysis. *Procedia Engineering*, *64*, 1602-1611.

[62] Collinsm, T. J., & Shen, W.-M. (2017). Particle Swarm Optimization for high-DOF inverse kinematics. 2017 3rd International Conference on Control, Automation and Robotics (ICCAR). doi:10.1109/iccar.2017.7942651

[63] Starke, S., Hendrich, N., Magg, S., & Zhang, J. (2016, December). An efficient hybridization of genetic algorithms and particle swarm optimization for inverse kinematics. In 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO) (pp. 1782-1789). IEEE.

[64] Huang, H.-C., Xu, S. S.-D., & Hsu, H.-S. (2014). Hybrid Taguchi DNA Swarm Intelligence for Optimal Inverse Kinematics Redundancy Resolution of Six-DOF Humanoid Robot Arms. Mathematical Problems in Engineering, 2014, 1–9. doi:10.1155/2014/358269

[65] Huang, H. C., Chen, C. P., & Wang, P. R. (2012, October). Particle swarm optimization for solving the inverse kinematics of 7-DOF robotic manipulators.

In *2012 IEEE international conference on systems, man, and cybernetics (SMC)* (pp. 3105-3110). IEEE.

[66] Durmuş, B., Temurtaş, H., & Gün, A. (2011, May). An inverse kinematics solution using particle swarm optimization. In International advanced technologies symposium (Vol. 4, pp. 193-197).