OPTIMIZATION OF ENERGY IN ROBOTIC ARM USING GA

Thesis submitted in partial fulfillment of the requirement for the award of degree of

Master of Engineering

In

Electronics Instrumentation and Control Engineering

By:

Gouri Shanker Sharma

(80751010)

Under the supervision of:

Mr. M.D.Singh

Sr. Lecturer, EIED

ELECTRICAL AND INSTRUMENTATION ENGG. DEPARTMENT

THAPAR UNIVERSITY

July 2009

CERTIFICATE

This is to certify that the work presented in this thesis entitled "**Optimization of Energy In Robotic Arm Using GA**" in partial fulfillment of the requirement for the award of the degree of **Master of Engineering in Electronics Instrumentation and Control Engineering at Thapar University, Patiala**, is an original record under supervision and guidance of **Mr. M.D. Singh.** The matter embodied in this report has not been submitted anywhere for the award of any degree.

Gouri

Date: - 8-07-2009

Gouri Shanker

Roll No 80751010

It is certified that the above statement made by the student is correct to the best of our knowledge and belief.

Mr. M.D. Singh
Lecturer, EIED
(Supervisor)
Thapar University, Patiala

S. Ghosh 10/7/09

Dr. Smarajit Ghosh
Professor& Head, EIED
Thapar University, Patiala

Dr. R.K Sharma 21/7
Dean of Academic Affairs
Thapar University, Patiala

13/8/09

(Dr Dilbag Singh)
External Examiner

ACKNOWLEDGEMENT

First of all I would like to thank the Almighty, who has always guided me to work on the right path of the life.

I am very thankful to the Head of the Department, **Dr. Smarajit Ghosh**, for his encouragement, support and for providing the facilities for the completion of this thesis.

This work would not have been possible without the encouragement and able guidance of my supervisor **Mr. M.D. Singh**. Their enthusiasm and optimism made this experience both rewarding and enjoyable. Most of the novel ideas and solutions found in this thesis are the result of our numerous stimulating discussions. Their feedback and editorial comments were also valuable for writing this thesis.

I am also very thankful to the entire faculty and staff members of Electrical - Instrumentation Department for their direct–indirect help, cooperation, love and affection, which made my stay at TIET memorable.

I am deeply indebted to my parents for their inspiration and ever encouraging moral support, which enabled me to pursue my studies.

Dated: 13-08-09 Gouri Shanker

(80751010)

ABSTRACT

This presentation proposes genetic algorithm (GA) to optimize the point-to-point trajectory planning for a 3-link robot arm. The objective function for the proposed GA is to minimizing traveling time and space, while not exceeding a maximum pre-defined torque, without collision with any obstacle in the robot workspace. Fourth and fifth polynomials are used to describe the segments that connect initial, intermediate, and final point at joint-space. Direct kinematics has been used for avoiding the singular configurations of the robot arm.

The algorithm finds the shortest path for the end effector. This is done by choosing two points one initial point and final point in X-Y coordinate. Now, we find the minimum path for travelling of the end factor, and the distance between them. All the distance covered by the end effector is divided in ten equal parts. Due to which, the error is minimized and the accuracy is increased due to divide ten equal parts.

Now by using GA for ten equal parts find out the suitable angle using inverse kinematic. The GA is applied for all the intermediate points between the initial and final point. Since the GA is used for all the ten points hence the accuracy is increased. Now, this problem is formulated by the equal distance travel by the robot hand with reference to previous position.

The objective function is to minimize the energy consumed by each motor using GA. We assume that the first angle moved is very small than the second & the second angle moved is less than the third one. This selection was due to the reason that for the first motor the power requirement is greatest, lesser for the second motor and least for the third motor, because the load associated with the motors is decreasing respectively.

Above results have been compared with manual movements of robotic arm. Results show that after Using GA, for the same set of location. The energy consumed is lesser than in manual control of robotic arm.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1. GA	Genetic Algorithm
2. DOF	Degree Of Freedom
3. SCARA	Selective Compliance Assembly Robot Arm
4. I/O	Input & Output
5. FVK	Forward Velocity Kinematics
6. FFK	Forward Force Kinematics
7. IPK	Inverse Position Kinematics
8. IVK	Inverse Velocity Kinematics
9. IFK	Inverse Force Kinematics
10. D-H	Denavit - Hartenberg

LITERATURE SURVEY

Jorge Angeles et.al [1], (1988) studied that Trajectory planning of robot motions for continuous-path operations is formulated in configuration space resorting to the intrinsic properties of the path traced by one point of the end effector. It is shown that, by referring the orientation of the end effector to a unique orthogonal frame defined at every point of the fore mentioned path. A systematic procedure for trajectory planning in configuration space was derived. The computations required determining the angular velocity and angular acceleration of the path frame reduced to computing the Darboux vector of the path and its time derivative.

Nearchou and Aspragathos [2] (1996) solved the problem of point-to point motion of redundant robot manipulators working in environments with obstacles considered. The problem is formulated as a constrained optimization problem and is solved using a method based on Genetic Algorithm (GA).

Doyle and Jones [3] (1996) proposed a path-planning scheme that uses a GA to search the manipulator configuration space for the optimum path. The GA generates good path solutions but it is not sufficiently robust.

HeCn and Zalzala [4] (1997) proposed a GA method to generate the position and the configuration of a mobile manipulator. The authors study the optimization of the least torque normal, the manipulability, the torque distribution and the obstacle avoidance, through the inverse kinematics scheme.

For generating smooth trajectory planning for specified path, Z. Zoller and P. Zentan [5] (1999) focused on the problem of the trajectory planning with constant kinetic energy motion planning. The author using equation of dynamic of robot motion with constant kinetic energy. This method produced trajectory characteristics smoother and better than which did obtained from time optimal method. Nevertheless, it can be implemented only for pre-specified path. Various methods for trajectory planning schemes based on GAs have been.

Pires and Machado [6](2000) propose a path planning method based on a GA while adopting

the direct kinematics and the inverse dynamics. The optimal trajectory is the one that minimize the path length, the ripple in the time evolution and the energy requirements, without any collision with the obstacle in the workspace

P. Garg and M. Kumar [7] (2002) use GA techniques for robot arm to identify the optimal trajectory based on minimum joint torque requirements. The authors use polynomial of 4th degree in time for trajectory representation to joint space variables. Garg and Kumar presented the formulation and application of a strategy for the determination of an optimal trajectory for a multiple robotic configuration. Simulated annealing (SA) and GA have been used as the optimization techniques

S. G. Yue et al. [8] (2002) focused on the problem of point-to-point trajectory planning of flexible redundant robot manipulator (FRM) in joint space. The proposed trajectory to minimize vibration of FRMs is based on GA. The authors use quadrinomial and quintic polynomials to describe the segment, which connects the initial, intermediate, and final points in joint space.

E.J. Pires et al. [9] (2002) optimized robot structure while optimizing the required manipulating trajectories using GA. The objective is to minimize the space/time ripple in the trajectory without colliding with any obstacles in the workspace, while optimizing the mechanical structure. focused on the problem of point-to point trajectory planning of flexible redundant robot manipulator (FRM) in joint space. The proposed trajectory to minimize vibration of FRMs is based on GA. The authors use quadrinomial and quintic polynomials to describe the segment, which connects the initial, intermediate, and final points in joint space.
Trajectory planning schemes based on GAs have been proposed. P. Garg and M. Kumar [11] use GA techniques for robot arm to identify the optimal trajectory based on minimum joint torque requirements. The authors use polynomial of 4th degree in time for trajectory representation to joint space variables

S.G. Yue, S.K. Tos[10] (2002) use genetic algorithm to optimize a planar robot manipulator trajectory. The GA objective is to minimize the trajectory space/time ripple without exceeding the maximum pre-defined torque. The authors use direct kinematics to avoid the singularities.

In this line of through, this paper, propose a point-to point trajectory planning method based on GA while adopting the direct kinematics and the inverse dynamics. The optimum trajectory is the one that minimize both traveling time and traveling space, while not exceeding the maximum pre-defined torque, without collision with any obstacle in the workspace. Robots is bigger they get, the more servos and I/O are required to enable them to fulfill their duties

About on line trajectory planning, Chwa et al. [11] (2003) proposed "Missile Guidance Algorithm" to generate on- line trajectory planning of robot arms of the interception of a fast maneuvering object. The authors employed the guidance law throughout the tracking phase, and dynamic constraints such as torque and velocity constraints and satisfied the matching condition of the position and velocity at the time of the interception altogether. This was carried out by introducing body axis (as well as joint and inertia axis) as trajectory planning coordinates and separating the trajectory-planning problem into direction planning and speed planning of robot arm. Various methods for

Zhe Tang et al. [12] (2003) proposed a third–order spline interpolation based trajectory-planning method to plan a smooth biped swing leg trajectory by reducing the instant velocity change, which occurs at the time of collision of the biped swing leg with the ground. The authors demonstrate that the impact effects can be avoided at the time of the swing foot's heel touching with the ground

Tian and Collins [13] (2003) studied the trajectory planning problem for a two-degree-of freedom robot in a workspace with point obstacles. The goal is to move the end-effector of the robot from a given starting point to a target while avoiding the point obstacles. They applied GA to search interior points between the starting point and the target.

Dongkyoung Chwa et.al [14] (2005) presented a novel approach to an online trajectory planning of robot arms for the interception of a fast maneuvering object under torque and velocity constraints. A body axis was newly introduced as a trajectory-planning coordinate in order to meet the position and the velocity matching conditions for a smooth grasp of the fast-maneuvering object. Using the position of the object and the end-effector in the inertia axis,

the acceleration commands were generated in the X-, Y -, and Z-directions of the body axis and the acceleration commands were modified considering the torque and the velocity constraints. The trajectory planning in the X-direction became the speed planning to achieve the maximum speed, whereas the trajectory planning in the Y- and Z-directions became the direction planning where a missile-guidance algorithm was employed to intercept the maneuvering object. Finally, the acceleration commands in the body axis were transformed into the angle commands of the end-effector in the joint axis, which was used as the actual trajectory commands in robot arms.

Kian Hsiang Low et.al [15] (2006) described a distributed layered architecture for resource-constrained multi robot cooperation, which was utilized in autonomic mobile sensor network coverage. In the upper layer, a dynamic task allocation scheme self-organized the robot coalitions to track efficiently across regions. It used concepts of ant behavior to self-regulate the regional distributions of robots in proportion to that of the moving targets to be tracked in a non-stationary environment. As a result, the adverse effects of task interference between robots were minimized and network coverage was improved. The layers employed self-organization techniques, which exhibit autonomic properties such as self-configuring, self-optimizing, self-healing, and self-protecting. Quantitative comparisons with tracking strategies such as static sensor placements, potential fields, and auction-based negotiation showed that our layered approach could provide better coverage, greater robustness to sensor failures, and greater flexibility to respond to environmental changes.

Emmanuel A. Merchán et.al [16] (2006) presented a novel fuzzy genetic algorithm (GA) approach to tackling the problem of trajectory planning of two collaborative robot manipulators sharing a common workspace, where the manipulators had to consider each other as a moving obstacle whose trajectory or behavior was unknown and unpredictable, as each manipulator had individual goals and where both had the same priority. The goals were not restricted to a given set of joint values, but were specified in the workspace as coordinates at which it is desired to place the end-effector of the manipulator. By not constraining the goal to the joint space, the number of possible solutions that satisfies the goal increases according to the number of degrees of freedom of the manipulators.

Xiaobu Yuan et.al [17] (2007) presented a novel approach of automated multi robot nano assembly planning. This approach used an improved self-organizing map to coordinate assembly tasks of nano robots while generating optimized motion paths at run time with a modified shunting neural network. It was capable of synchronizing multiple nano robots working simultaneously and efficiently on the assembly of swarms of objects in the presence of obstacles and environmental uncertainty. Operation of the presented approach was demonstrated with experiments at the end of the paper.

Foudil [18] (2007) gave the generalized trajectory of a mobile Manipulator. The objective was to allow the end effector track a given trajectory in a fixed world frame. The motion of the platform and that of the manipulator were coordinated by a pseudo neural network designed from the kinematics model of the system. A learning paradigm was used to produce the required reference variables for each of the mobile platform and the robot manipulator for an overall coordinate behavior. Simulation results were presented to show the effectiveness of the proposed scheme.

Mike Peasgood et.al [19] (2008) addressed the challenging problem of finding collision-free trajectories for many robots moving toward individual goals within a common environment. Most popular algorithms for multi-robot planning manage the complexity of the problem by planning trajectories for robots individually; such decoupled methods were not guaranteed to find a solution if one exists. In contrast, this paper described a multiphase approach to the planning problem that used a graph and spanning tree representation to create and maintain obstacle-free paths through the environment for each robot to reach its goal. The resulting algorithm guarantees a solution for a well-defined number of robots in a common environment. The computational cost was shown to be scalable with complexity linear in the number of the robots, and demonstrated by solving the planning problem for 100 robots, simulated in an underground mine environment, in less than 1.5 s with a 1.5 GHz processor. The practicality of the algorithm was demonstrated in a real-world application requiring coordinated motion planning of multiple physical robots.

In the last decade, genetic algorithms (GAs) have been applied in large number of fields such as in control, parameter, and system identification, robotics, planning and scheduling, image processing, pattern recognition, speech recognition. This paper addresses the area of robotics,

namely the point-to-point trajectory planning for mechanical manipulators. At start, some of conventional methods have been used for trajectory planning. For generating smooth trajectory planning for specified path.

From the above survey, it has been found that there are numerous techniques to optimize the path of a robot/robotic arm between two points. But GA based method are mostly used.

CHAPTER 1
INTRODUCTION

Robot is derived from a Czech word meaning "menial labor". A robot is a mechanical or virtual, artificial agent. It is usually an electromechanical system, which, by its appearance or movements, conveys a sense that it has intent or agency of its own. They are capable of performing many different tasks and operations precisely and do not require any common safety and comfort elements. They possess a number of links attached serially to each other with joints, where each joint can be moved by some type of actuator.

Today, robots are used in many ways, from lawn mowing to auto manufacturing. Scientists see practical uses for robots in performing socially undesirable, hazardous or even "impossible" tasks: trash collection, toxic waste cleanup, desert and space exploration, and more. AI researchers are also interested in robots as a way to understand human (and not just human) intelligence in its primary function.

1.1Robotics

Robotics is the science and technology of robots, their design, manufacture, and application. Robotics requires a working knowledge of electronics, mechanics and software, and is usually accompanied by a large working knowledge of many subjects. A person working in the field is a robotics. Robotics systems consist of not just robots, but also other devices and systems that are used together with the robots to perform the necessary tasks. Robotics is an inter disciplinary subject that benefits from mechanical engineering, electrical and electronic engineering, computer science, biology, and many other disciplines. Robotics is used in industrial, military, exploration, home making and academic and research applications.

1.2 Robot Components

A Robotic system consists of various elements like:

1.2.1 Manipulator:- This is the main body of Robot and consists of the links, the joints,

and other structural elements of the Robot. The robot manipulator can be divided into two sections, each with a different function:

(**i) Arm and Body** - The arm and body of a robot are used to move and position parts or tools within a work envelope. They are formed from three joints connected by large links.

(ii)Wrist - The wrist is used to orient the parts or tools at the work location. It consists of two or three compact joints.

1.2.2 End-effector: - In robotics, an end effector is a device or tool connected to the end of a robot arm. The structure of an end effector, and the nature of the programming and hardware that drives it, depends on the intended task. .

1.2.3 Actuators: - The actuators are the 'muscles' of a robot; the parts which convert stored energy into movement. By far the most popular actuators are electric motors, but there are many others, some of which are powered by electricity, while others use chemicals, or compressed air. Eg. are Motors, Stepper Motors, Piezo Motors, Air Muscles, Electroactive Polymers

1.2.4 Sensors: -Sensors are used to collect information about the internal state of the robot or to communicate with the outside world .As in humans the Robot controllers needs to know where each link of the robot is in order to know the robot's configuration. Choosing a sensor for a particular application depends on various characteristics like cost of the sensor, size and weight of the sensor, output of the sensor that may be digital or

analog, how a sensor can interface with other devices, its resolution, reliability, accuracy, sensitivity, linearity etc.

Sensors are broadly classified as:

(i) Position sensors: Used to measure displacements, both rotary and linear, as well as movements.

(**ii) Velocity sensors**: Very much similar to position sensors, can measure rotational and linear velocity.

(iii) Force and pressure sensors: They are used for dynamic purposes and measure forces.

1.2.5 Controllers: - The controller is a major component of a robot, which directs the end effector to move in a desired sequence, and to pass through desired points. It also functions to store position and sequence data in memory so that program can be repeated. Robot

controllers range in complexity from simple stepping switches through pneumatic logic sequencers, diode matrix boards, electronic sequencers, and microprocessor to minicomputers.

1.2.6 Processor: - The processor is the brain of the robot. It calculates the motions of the robot's joints, determine how much and how fast each joint must move to achieve the desired location and speed, and oversees the coordinated actions of the controller and the sensors. The processor is generally a computer, which works like all other computers, but is dedicated to a single purpose. It requires an operating system, programs, and peripheral equipment such as monitors and has many of the same limitations and capabilities of a PC processor.

1.2.7 Software: - There are perhaps three groups of software that are used in a robot. One is the operating system, which operate the computer. The second is the robotic software, which calculates the necessary motions of each joint based on the kinematic equation of the robot. The third group is the collection of routines and application programs that are developed in order to use the peripheral devices of the robots, such as vision routines, or to perform specific task.

1.3 Degrees of Freedom

In Robotics, degrees of freedom (DOF) are the set of independent displacements and/or rotations that specify completely the displaced or deformed position and orientation of the body or system. The number of DOF that a manipulator possesses is the number of independent position variables that would have to be specified in order to locate all parts of the mechanism. In other words, it refers to the number of different ways in which a robot arm can move. In the case of typical industrial robots, because a manipulator is usually an open kinematic chain, and because each joint position is usually defined with a single variable, the number of joints equals the number of degrees of freedom.

In total, our arm has seven degrees of freedom: three in the shoulder, one in the elbow, and three in the arm below the elbow. Three degrees of freedom are sufficient to bring the end of a robot arm to any point within its workspace, or work envelope, in three dimensions. Thus, in theory, a robot should never need more than three degrees of freedom. But the extra

possible motions, provided by multiple joints, give a robot arm versatility that it could not have with just three degrees of freedom.

1.4 Robot Joints and Coordinates

Most robots have either a Prismatic joint or a Revolute joint.

(i) Prismatic joints are linear; there is no rotation involved. They are either hydraulic or pneumatic cylinders or, they are linear electric actuators. These joints are used in gantry, cylindrical or, similar joint configuration.

(ii) Revolute joints are rotary, and although hydraulic and pneumatic rotary joints are common, most rotary joints are electrically driven, either by stepper motors or, more commonly by servomotors.

Robot configurations generally follow the coordinate frames with which they are defined. The following configurations are common.

(i) Cartesian: - These robots are made of three linear joints that position the end effector, which are usually followed by additional revolute joint that orientate the end effector.

(ii) Cylindrical: - They have two prismatic joints and one revolute joint for positioning the part, plus revolute joints for orienting the part.

(iii) Spherical: - They follow a spherical coordinate system, which has one prismatic and two revolute joints for positioning the part, plus addit ional revolute jo ints for orientation.

(iv) Articulated: - They are all revolute, similar to a human arm. They are perhaps the most common configuration for industrial robots.

(v) Selective Compliance Assembly Robot Arm (SCARA): - They have two revolute joints that are parallel and allow the robot to move in horizontal plane, plus an additional prismatic joint that moves vertically.

1.5 Robot Programming Modes

Robots may be programmed in number of different modes, depending on the robot. The following programming modes are common.

(i) Physical setup: - In this mode, an operator sets up switches and hard stops that control the motion of the robot. This mode is usually used along with other devices, such as Programmable Logic Controllers (PLC).

(ii) Teach mode: - Here robot joints are moved with teach pendant. When the desired location is achieved, the location is entered into the controller. During Play back, the controller will move the joints to the same locations and orientations.

(iii) Continuous Walk Trough Mode: - All robot joints are moved simultaneously, while the motion is continuously sampled and recorded by the controller. During Playback, The exact motion that was recorded is executed.

(iv) Software mode: - A program is written offline or online and is executed by the controller to control the motions. This mode is sophisticated and versatile mode and can include sensory information, conditional statements and branching.

1.6 Advantages

A) Increase Productivity with Shorter Cycle Time
B) Increase Quality of Product, Process and Work Environment
C) Increase Manufacturing Flexibility
D) Reduce Scrap and Manufacturing Costs
E) Compete Better By Reducing Undesirable Tasks
F) Improved Worker Safety
G) Decrease Floor Space
H) GUI Setup for Fast & Easy Operation
I) Cluster Machining in One Cell
J) Automatic Changeover for Different Products
K) Re-Program Equipment for Different Process
L) Stabilizes Production

1.7 Disadvantages

A) High initial cost

B) Need for extra space, and new technology

C) Need highly skilled and technical engineers, programmers

D) Possible injuries during working with robot

E) Limited functions

F) New systems bring out defects

G) Intellectual or physical limitations of employees

H) Replace certain workers causing economic losses

I) Certain military robots are pro-war

J) Making companies and human beings more dependent

1.8 Parallax Servo Controller

In implanting the result, a robotic arm has been selected. This robotic arm is made by parallax. The Parallax Servo Controller, PSC, controls up to 16 servos, and may be networked together so that two PSCs can control 32 servos using a single I/O line. The PSC manages all of the servo pulses so your host controller doesn't have to. Additionally, the ramping function allows you to individually set the speed of each servo. With ramping, we can tell the servo where to go, and just how fast to get there. The result is true, "set-it and forget-it" functionality. The processor used on the PSC requires a supply 5 VDC. Use a separate power supply for your servos. In general, servos require 4-7.5 VDC. Be sure that the servo power source can supply ample current at the proper voltage and will not damage the servos. Locate and remove the Parallax Servo Controller from its protective anti-static bag. It should closely resemble the image shown in Figure 1.1.

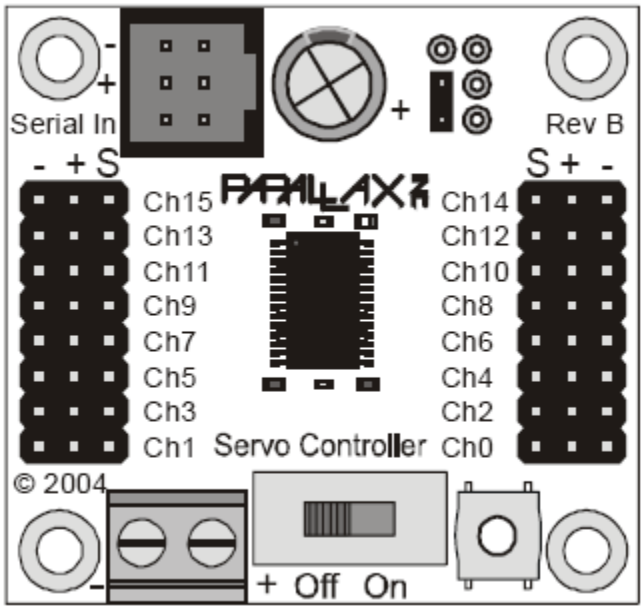


Figure 1.1: Parallax Servo Controller - Rev B (http://www.parallax.com).

Mount the Parallax Servo Controller using 4-40 screws (or equivalent). Figure 1.2 details the mounting-hole spacing and other dimensions of the PSC Rev A, which are the same for the Rev B board as well. Ensure the servo power switch on the Parallax Servo Controller is off, and then connect the power source for the servos to the screw terminals observing proper polarity. The power source for the servos must be a separate power source from that of the Board of Education Rev C. This document assumes the host system is a BS2 module plugged into a Board of Education Rev C from Parallax, though those items are not required for proper operation.

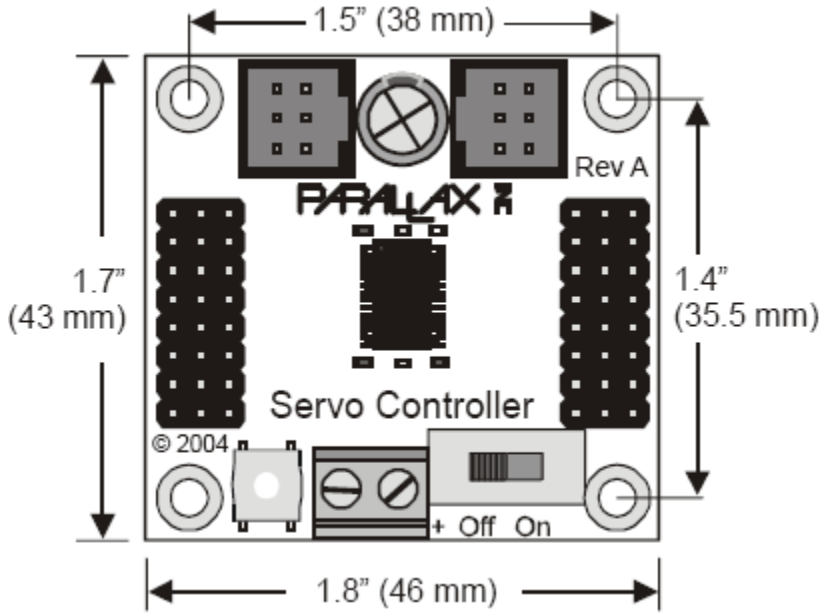


Figure 1.2: Parallax Servo Controller Mounting Holes (http://www.parallax.com).

A three-conductor cable (included) connects the Ground, +5VDC, and serial I/O line to the host system. If you are not using the Rev C BOE, connect the wires of the three-conductor cable to VSS (black), +5VDC (red), and the I/O pin of your choice on your host system. Note the I/O pin number on the BOE that is connected as the serial I/O line on the PSC. This document assumes you have connected this cable into X4, slot 15. Ensure that the V servo jumper selector, (between X4 and X5 on the BOE Rev C), is set to the VDD position. Connect your servos to the 3 pin terminals provided. Figure 1.3 is an example showing the Parallax Servo Controller connected to two servos and the Board of Education.

Follow the instructions in this document when making connections to the PSC. To power up your PSC, slide the power switch on the BOE to the "2" position. You should see the red LED illuminates on the PSC as well as the green LED light on the BOE.

1.9 Serial Command Format

The PSC supports several commands that are sent to it via RS-232 serial protocol. The voltage swing of this serial line is 0-5 VDC (TTL level). Each serial command must be preceded with an exclamation point, "!", and the pair of letters, "SC".

When your PSC starts up, the default baud rate is 2400 N 8 2, (no parity, eight data bits, two stop bits) true data, open-drain (driven low, pulled high). The exclamation point is used in some App Mods to determine the incoming baud rate, thereby supporting a feature called Auto-Baud. The PSC does not support Auto-Baud. The "SC" portion is an identifier that pertains to the PSC. Together, the "!" and the "SC" form a preamble, "ISC". The preamble serves to distinguish commands for the PSC from other messages on the serial I/O line, and allow different types of App Mods to use the same serial line.

After the preamble is sent, the command and associated parameters are sent. The eighth and final character sent is a $0D, (CR), used to terminate the string. If the command causes the PSC to reply, a three-byte reply is sent after a 1.5 mS delay

A BASIC Stamp microcontroller is a single-board computer that runs the PBASIC language interpreter in its microcontroller. It is called a “Stamp” simply because it is close to the size of an average postage stamp. The developer's code is stored in an EEPROM, which can also be used for data storage. The PBASIC language has easy-to-use commands for basic I/O, like turning devices on or off, interfacing with sensors, etc. More advanced commands let the BASIC Stamp module interface with other integrated circuits, communicate with each other, and operate in networks. This module is rated for the Commercial Temperature Range - Get the benefits of the BS2p with lower power consumption (resulting in prolonged battery life for your mobile robots, embedded application etc). The BS2p24 has a program execution speed of 12,000 instructions/second. New I/O instructions handle parallel LCDs, Philips I2C (master), and Dallas/Maxim 1-Wire (master) communications. The combination of impressive processing speed, memory, and special commands really sets the BS2p on a higher level. The total number of PBASIC commands on the BS2p is now at 61. These additional commands provide you with the ability to get the job done with fewer command lines, resulting in more efficient code.

The four premier additions to the PBASIC command options are listed below:

I2CIN and I2COUT - Allows you to use one I/O pin to communicate with I2C devices. o LCDIN and LCDOUT - Connecting to a Parallel LCD has never been easier. OWIN and OWOUT-Interface with Dallas Semiconductor 1-Wire parts. POLLIN, POLLOUT, POLLMODE, POLLRUN, POLLWAIT - Polled interrupt capability allows you to monitor I/O pins in between your PBASIC code.

1.10 Features

2.1 mm center-positive plug and 9-volt battery power supply connections (mechanically interlocked to prevent dual connection). Note: The Board of Education USB is not powered from the USB connection. Three-position power switch allows BASIC Stamp IC programming without providing power to servo connectors.Jumper selection of servo power: regulated (Vdd) or unregulated (Vin). Mini USB connector for BASIC Stamp IC programming and serial communication during run-time. On-board regulator delivers up to 1 amp of power for larger projects. P0 - P15 I/O pins, Vdd, Vin, and Vss connections brought adjacent to 2" x 1 3/8" breadboard area. Female 10-pin dual row connector for optional AppMods.

Power Supply Clip and Barrel Jack (top left): The 9 V battery clip and the barrel jack are positioned so that they cannot be accidentally used at the same time. For the jack we recommend 9 VDC V 300 mA supply.

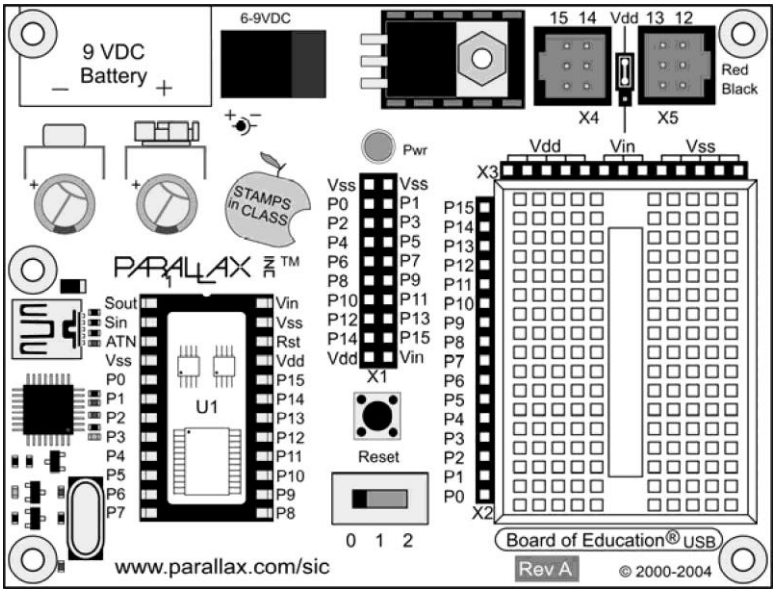


Fig. 1.3 Development / Education Platform for the BASIC Stamp® and Javelin Stamp Microcontrollers

1.10.1 Power Switch (bottom center): The leftmost position (0) is OFF – all power is disconnected. Always place the switch in position-0 when changing components on the

breadboard, and when disconnecting or reconnecting to the PC. The middle position (1) provides Vin to the regulator, the BASIC Stamp IC socket, and to the connectors marked Vin. In this position, Vdd will also be available on the breadboard and AppMod connectors. The rightmost position (2) provides power to the servo connectors X4 and X5 (see servo power selection below). The three-position position power switch is convenient when using the BOE-USB on small robots, like the Parallax Boe-Bot. Use position 1 to edit and test code while power is removed from the servos.

1.10.2 Servo Ports (X4 and X5), and Power Selection (top right): Select the power provided to servo sockets X4 and X5 by the jumper located between them; the default position is Vdd (+5 V regulated). When using a six-volt battery pack (as on a Boe-Bot robot), you may wish to move this jumper to the Vin position to provide extra power to the servos. When using Vin in the servo ports, always check to make sure the voltage supplied does not exceed the specifications of the particular brand of servos you are using. X4 and X5 connect to the BASIC Stamp I/O pins labeled above the sockets; do not build incompatible circuits connected to the same I/O pins on the breadboard when using X4 or X5. Always check the pinout and voltage requirements of 3-pin devices before connecting them to X4 or X5.

1.10.3 Reset Button: The reset button can be used to restart your BASIC Stamp IC without having to cycle the power. This saves wear-and-tear on the power switch for simple program restarts.

1.10.4 Breadboard Access for BASIC Stamp I/O (X2), Vdd, Vin, and Vss (X3): The BASIC Stamp IC's 16 I/O pins are brought to the X2 female socket left of the breadboard. I/O pins are accessed by plugging wires into the header, then into the breadboard sockets. The X3 socket provides four connection points for a +5V (Vdd) connection, unregulated input voltage (Vin), and ground (Vss).

1.10.5 AppMod Header (X1): provides connection and signal routing for 20-pin AppMods, the eb500 EmbeddedBlue Transceiver and the LCD Terminal . All I/O pins, Vdd, Vin, and Vss are routed through the AppMod connector. The BASIC Stamp I/O pins used by a device in the AppMod Header should not also be used with conflicting circuits on the breadboard

area. Please refer to the individual AppMod product documentation for device pin maps.

1.10.6 The training board USB Breadboar. The breadboard has many strips of copper which run underneath the board These strips connect the sockets to each other horizontally, in groups of 5. This makes it easy to connect components together to build circuits. Each metal strip and its five sockets forms a node.

A node is a point in a circuit where two components are connected. Connections between different components are formed by putting their legs in a common node. To use the breadboard, the legs of components or wires are placed in the sockets. The sockets are made so that they will hold the component in place. For chips with many legs (ICs), place them in the middle of the board so that half of the legs are on the left side and half are on the right side. Nodes on the left side are not connected to nodes on the right side.

1.11 Features

Each Servo is individually controllable: Position, rate, offset, and post delay. Speed Control. Each servo may have any of 63 run-time changeable rates. 128 Frames supported though most sequences require far fewer frames. The processor used on the PSCusb requires 5 VDC which is supplied by the USB port. Servos require far more power than the USB port can supply. Therefore you must use a separate power supply for your servos. In general, servos require 4-7.5 VDC. Be sure that the servo power source can supply ample current at the proper voltage and will not damage the servos. Locate and remove the PSCusb from its protective anti-static bag. It should closely resemble the image shown in Figure 1.

Ensure the servo power switch on the Parallax Servo Controller is off, and then connect the power source for the servos to the screw terminals observing proper polarity. Servos require more power than the USB port can supply. For this reason, you must connect a separate power supply for the servos.

1.12 Controls Description

To control the servos in real-time mode, connect your servos to the PSCusb and simply slide the power switch on the PSCusb to the ON position. Now, moving the corresponding servo position slide bar in the PSCI software positions the servo.

1.12.1 Offset

Not all servos are alike so the offset control allows you to vary the center point of each servo. This may not matter too much for standard servos, but is very helpful when controlling servos modified to rotate 360 degrees.

1.12.2 Position

You may use either the slide control or the up/down arrows, or enter the servo position numerically to set the servo position. Please note that by clicking on the numbers 2500, 1500, or 750, the servo position will be set to that number. These numbers correspond to the width of the servo command pulse, in microseconds.

1.12.3 Rate

Each servo may be set to a rate of rotation. Furthermore, each servo may have a different speed for each frame. A servo rate of 0 will cause the servo to move as quickly as possible; it's fastest rate. A servo position of 63 will cause the servo to move to it's destination position at its slowest rate. At speed 63, the servo takes about 45 seconds to complete a 180 degree motion.

1.12.4 Delay

When a frame is executed, as in animate mode, a servo command is sent to the PSCusb for each servo in the order 0,1,2,3... After each servo command is sent, the delay time is observed before sending the next servo command. The Delay parameter has no effect when operating in real-time mode.

1.12.5 Frame

The number to the right of the word "Frame" indicates the current frame.

1.12.6 Go to

The number to the right of the phrase "Go to" shows the frame number that will be jumped to when all servo commands have been sent and all delays have been observed for the current frame. Note: no servo commands are sent for servos that appear to be unused.

1.12.7 Animation Controls

Controls are provided to facilitate the creation, controlling, and editing of animation

sequences. By hovering the mouse over each control button (without clicking) a hint will pop up to help clarify the purpose of the button.

1.13 BASIC Stamp2p Microcontroller Information

A BASIC Stamp microcontroller is a single-board computer that runs the PBASIC language interpreter in its microcontroller. It is called a “Stamp” simply because it is close to the size of an average postage stamp. The developer's code is stored in an EEPROM, which can also be used for data storage. The PBASIC language has easy-to-use commands for basic I/O, like turning devices on or off, interfacing with sensors, etc. More advanced commands let the BASIC Stamp module interface with other integrated circuits, communicate with each other, and operate in networks. This module is rated for the Commercial Temperature Range - Get the benefits of the BS2p with lower power consumption (resulting in prolonged battery life for your mobile robots, embedded application etc). The BS2p24 has a program execution speed of 12,000 instructions/second. New I/O instructions handle parallel LCDs, Philips I2C (master), and Dallas/Maxim 1-Wire (master) communications. The combination of impressive processing speed, memory, and special commands really sets the BS2p on a higher level. The total number of PBASIC commands on the BS2p is now at 61. These additional commands provide you with the ability to get the job done with fewer command lines, resulting in more efficient code.

The four premier additions to the PBASIC command options are listed below:

I2CIN and I2COUT - Allows you to use one I/O pin to communicate with I2C devices. LCDIN and LCDOUT - Connecting to a Parallel LCD has never been easier. OWIN and OWOUT - Interface with Dallas Semiconductor 1-Wire parts. POLLIN, POLLOUT, POLLMODE, POLLRUN, POLLWAIT - Polled interrupt capability allows you to monitor I/O pins in between your PBASIC code. For complete descriptions and examples of the highlighted commands above, please read the BASIC Stamp Command Reference section.

Table 1.1 Technical Specifications

Processor Speed	20 MHz Turbo
Program Execution Speed	~12,000 instructions/sec.
RAM Size	38 Bytes (12 I/O, 26 Variable)
Scratch Pad RAM	128 Bytes
EEPROM (Program) Size	8 x 2K Bytes, ~4,000 instructions
I/O Pins	16 +2 Dedicated Serial
Voltage Requirements	5 - 12 vdc
Current Draw at 5V	40 mA Run / 350 µA Sleep
PBASIC Commands	61
Environment	0-70 deg C.
Size	1.2"x0.6"x0.4"

CHAPTER 2
ROBOT MODELING

2.1 Robot Kinematics

In order to adjust the robot's controllers, a kinematic and dynamic mathematical model of the robot (include the robot's physical characteristics) are needed. The kinematics is the study of the robot's movements with regard to a reference system. It is an analytic description of the special movement of the robot like a function of time and a relationship, between the position and the orientation of the robot's external link and the values of their joint co-ordinates.

Robot kinematics is mainly of the following two types

(i) Forward kinematics

(ii) Inverse kinematics.

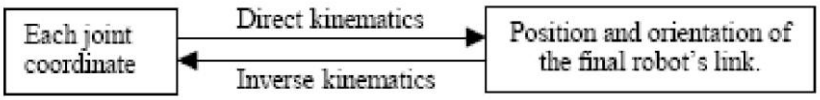


Figure 2.1: Forward and Inverse kinematics

2.1.1 Forward Kinematics

In Forward Kinematics, Each Joint angle is given, the position and orientation of end effector is calculated. It is also known as direct kinematics. In forward kinematics, the length of each

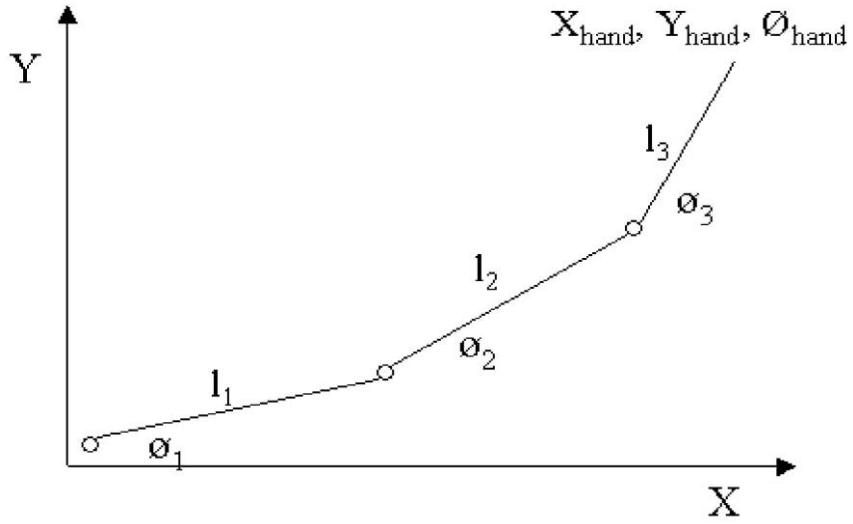


Figure 2.2: Forward kinematic plane

link and the angle of each joint is given and we have to calculate the position of any point in the work volume of the robot.

(i) Forward position kinematics

The forward position kinematics (FPK) solves the following problem: "Given the joint positions, what is the corresponding end effector's pose?". One given joint position vector always corresponds to only one single end effector pose. The FK problem is not difficult to solve, even for a completely arbitrary kinematic structure. In this following possibilites are considered:

(a) Cartesian (gantry, rectangular) coordinates.
(b) Cylindrical coordinates.
(c) Spherical coordinates.
(d) Articulated coordinates.

(ii) Forward velocity kinematics

The forward velocity kinematics (FVK) solves the following problem: "Given the vectors of joint positions and joint velocities, what is the resulting end effecter twist?" One given set of joint positions and joint velocities always corresponds to only one single end effecter twist. Its possible configurations are:

(a) Roll, Pitch, Yaw
(b) Euler Angles
(c) Articulated joints

(iii) Forward force kinematics

The forward force kinematics (FFK) solves the following problem: "Given the vectors of joint force/torques, what is the resulting static wrench that the end effector exerts on the environment?"

2.1.2 Inverse Kinematics

In inverse kinematics, the length of each link and position of the point in work volume is given and we have to calculate the angle of each joint. As shown below:

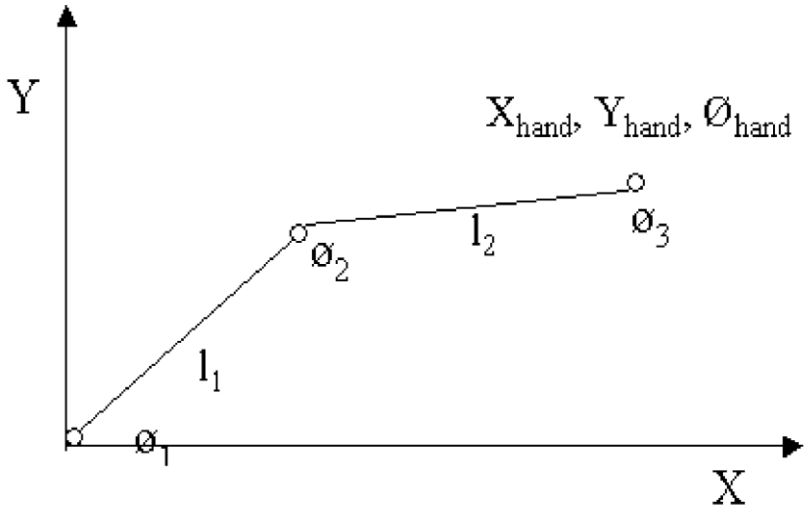


Figure 2.3: Inverse kinematic plane

(i) Inverse Position Kinematics

The Inverse Position Kinematics (IPK) solves the following problem: "Given the actual End effecter poses, what are the corresponding joint positions?" In contrast to the forward Problem, the solution of the inverse problem is not always unique: the same end effecter Pose can be reached in several configurations, corresponding to distinct joint position Vectors. Its possibilities are same as in the case of Forward Kinematics

(ii) Inverse Velocity Kinematics

Assuming that the Inverse Position Kinematics problem has been solved for the current End effecter pose, the Inverse Velocity Kinematics (IVK) then solves the following Problem: "Given the end effecter twist, what is the corresponding vector of joint Velocities?" Again, configurations are same.

(iii) Inverse Force Kinematics

Assuming that the Inverse Position Kinematics problem has been solved for the current end effector pose, the Inverse Force Kinematics (IFK) then solves the following problem: "Given the wrench that acts on the end effector, what is the corresponding vector of joint Forces/torques?"

2.1.3 D-H Representation of Forward Kinematic Equations

The D-H method allows the step from a link to the following link by 4 basic transformations that depends only on the robot's constructive characteristics.

These are basic transformations that relate the reference system of the element n+1 with the reference system of the element n.

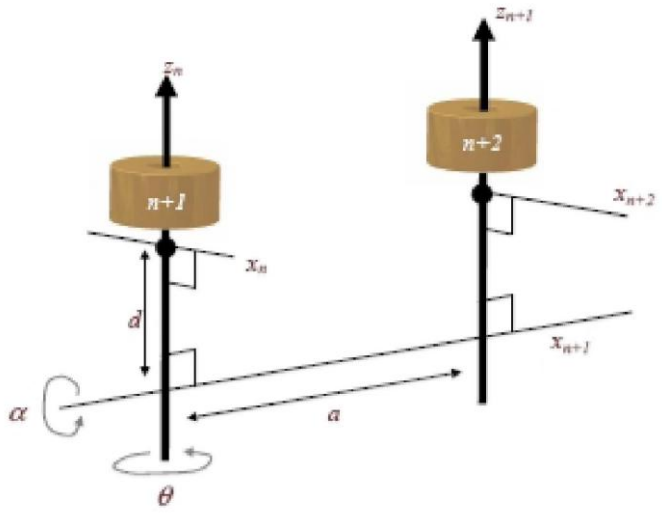


Figure 2.4: D-H representation of general joint link combination

1. A rotation qn+i about the Zn axis (to bring Xn parallel with Xn+1)
2. A translation dn+i along the Zn axis (to make the x-axes collinear)
3. A translation an+i along the X-axis (to make the z-axes coincide)
4. A rotation an+i about the Xn axis (to bring Zn parallel with Zn+1)

2.2 Trajectory Planning

Path and trajectory planning relates to the way a robot is moved from one location to another in a controlled manner. A path is defined as a sequence of robot configurations in a particular order without regard to the timing of these configurations. However, a trajectory is concerned about when each part of the path must be attained, thus Specifying timing. A trajectory depends upon the velocities and the accelerations at different times where as path is independent of it. The trajectory planning can be done in two ways:

(i)Joint Space Trajectory

(ii) Cartesian Space Trajectory

2.2.1 Joint Space

A trajectory planning approach for controlling flexible robots is described here. It is demonstrated that choosing actual joint angles as the generalized rigid coordinates is the key to applying the proposed approach. From the observation of the special structure of the input matrix, the concepts of motion-induced vibration and inverse dynamics under a specified motion history of the joints are formed naturally. Based on the above concepts, trajectory planning in joint space is proposed by using the optimization technique to determine the motion of joints along a specified path in joint space or workspace and for general point-to-point motion. The motion for each joint is assumed to be in a class consisting of a fifth-order polynomial and finite terms of Fourier series. This parameterization of motion allows the optimal trajectory planning to be formulated as a standard nonlinear programming problem, which avoids the necessity of solving a two-point-boundary-value problem and using dynamic programming. Setting the accelerations to zero at the initial and the final times is used to obtain smoother motion to reduce the spillover energy into unmoduleted high-frequency dynamics.

2.2.2 Cartesian Space

The Robot is to track a specified path for the purpose of accomplishing the task satisfactorily and avoiding collisions with obstacles in the workspace. In reality, typical robotic tasks include moving the end-effector to specified Cartesian positions or along a specified Cartesian path. Continuous- Path (CP) robotics applications appear in operations such as arc welding, flame cutting, and routing etc. Robot trajectory planning refers to the development of time history of position, velocity, and acceleration for each degree of freedom. To achieve a smooth motion, not only the position but also the velocity must be prescribed at every intermediate point along the motion path. Trajectory Planning can be conducted either in the joint-variable space or in the Cartesian space. However, for trajectory planning in Cartesian space, the transformation from Cartesian to joint coordinates in real-time is required. As the control of robot motion is carried out at the joint level, the computational complexities involved in trajectory planning and ordinate transformation have hindered the on-line

implementation of Cartesian based path planning. The path is divided into m segments. It was assumed that the traveling path could be specified by a group of parameter equations in Cartesian coordinates. The method described here has the merits as follows:

(1) The co-efficient of the polynomial of each segment of the path are obtained in recursive form, so the algorithm can be easily converted into computer programs

(2) The method needs less computation and obtains approximate minimum-time trajectories.

CHAPTER -3
MOTION PLANNING STRATEGY

3.1 Motion Planning Strategy

The supposed point-to-point trajectory is connected by several segments with continuous acceleration at the intermediate via point as shown in figure 1. The intermediate points can be given as particular points that should be passed through.

For a robot, the number of degrees of freedom of a manipulator is n and the number of end-effectors degree of freedom is m. If one wishes to be able to specify the position, velocity, and acceleration at the beginning and the end of a path segment, a fourth and a fifth polynomial can be used. Let us assume that there is mp intermediate via points between the initial and final points.

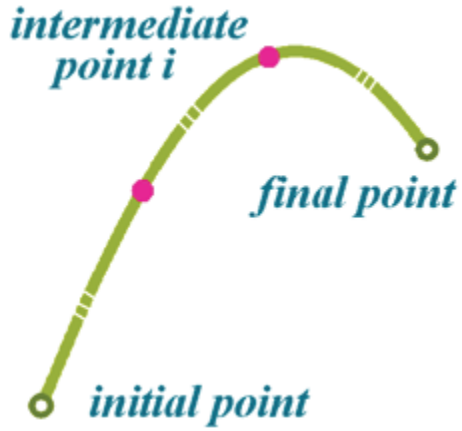


Figure 3.1: Intermediate points on the point-to-point trajectory

Between the initial points to mp (Intermediate points) intermediate via points, a fourth polynomial is used to describe these segments as [7]:

$$\theta_{i,i+1} = a_{i0} + a_{i1}t_i + a_{i2}t_i^2 + a_{i3}t_i^3 + a_{i4}t_i^4 \qquad (i = 0, \ldots\ldots mp - 1) \qquad (1)$$

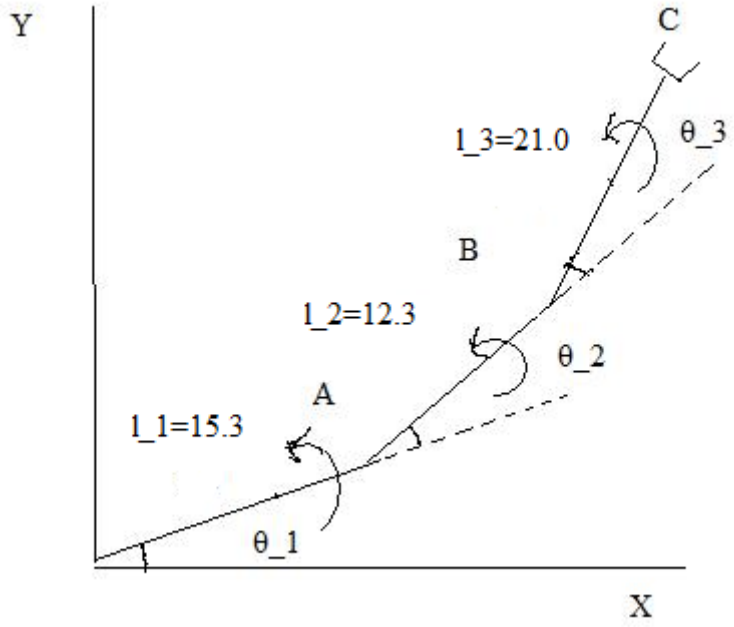


Figure 3.2 Three degree of freedom planar mechanism

Where (a_{i0} to a_{i4}) be constants and the constraint are given as:

$$\theta = a_{i0} \quad (2)$$

$$\theta_{i+1} = a_{i0} + a_{i1}t_i + a_{i2}t_i^2 + a_{i3}t_i^3 + a_{i4}t_i^4 \quad (3)$$

$$\dot{\theta} = a_{i1} \quad (4)$$

$$\dot{\theta_{i+1}} = a_{i1} + 2a_{i2}t_i + 3a_{i3}t_i^2 + 4a_{i4}t_i^3 \quad (5)$$

$$\ddot{\theta}_i = 2a_{i2} \quad (6)$$

Where Ti is the execution time from point i to point i+1. The five unknowns can be solved as:

$$a_{i0} = \theta_i \quad (7)$$

$$a_{i1} = \dot{\theta}_i \quad (8)$$

$$a_{i2} = \frac{\dot{\theta}_i}{2} \quad (9)$$

$$a_{i3} = (4\theta_{i+1} - \dot{\theta}_{i+1}t_i - 4\theta_i - 3\ddot{\theta}_i t_i^2\ )t_i^3 \quad (10)$$

$$a_{i4} = \left(\dot{\theta}_{i+1} - 3\theta_{i+1} + 3\theta_1 + 2\dot{\theta}_1 t_i + \frac{\ddot{\theta}_i t_i^2}{2}\right) t_i^4 \quad (11)$$

The intermediate point (i+1)'s acceleration can be obtained as:

$$\ddot{\theta}_{i+1} = 2a_{i2} + 6a_{i3}t_i + 12a_{i4}t_i^2 \quad (12)$$

The segment between the number mp of intermediate points and the final point can be described by quintal polynomial as:

$$\theta_{i,i+1}(t) = b_{i0} + b_{i1}t_i + b_{i2}t_i^2 + b_{i3}t_i^3 + b_{i4}t_i^4 + b_{i5}t_i^5 \quad (13)$$

Where the constants are given as:

$$\theta_i = b_{i0} \quad (14)$$

$$\theta_{i+1} = b_{i0} + b_{i1}t_i + b_{i2}t_i^2 + b_{i3}t_i^3 + b_{i4}t_i^4 + b_{i5}t_i^5 \quad (15)$$

$$\theta_1 = b_{i1} \quad (16)$$

$$\dot{\theta}_{i+1} = b_{i1} + 2b_{i2}t_i + 3b_{i3}t_i^2 + 4b_{i4}t_i^3 + b_{i5}t_i^4 \quad (17)$$

$$\ddot{\theta} = 2b_{i2} \quad (18)$$

$$\theta_{i+1} = 2b_{i2} + 6b_{i3}t_i + 12b_{i4}t_i^2 + 20b_{i5}t_i^3 \quad (19)$$

In addition, these constraints specify a linear set of six equations with six unknowns whose solution is:

$$bi_0 = \theta_i \quad (20)$$

$$b_{i\,1} = \dot{\theta}_i \quad (21)$$

$$bi_2 = \ddot{\theta}_i / 2 \quad (22)$$

$$b_{i\,3} = (20\theta_{i+1} - 20\theta_i - (8\dot{\theta}_{i+1} + 12\dot{\theta}_i)ti - (3\ddot{\theta}_i - \ddot{\theta}_{i+1})t_i^2)/2t_i^3 \quad (23)$$

$$bi_5 = (12\theta_{i+1} — 12\theta_i - (6\dot{\theta}_{i+1} + 6\dot{\theta}_i)\,ti - (\ddot{\theta}_i - \ddot{\theta}_{i+1})ti^2)/2ti^5 \quad (24)$$

As formulated above, the total parameters to be determined are the joint angles of each intermediate via point (n×mp parameters), the joint angular velocities of each intermediate point (n×mp parameters), the execution time for each segment (mp+1 parameters), and the posture of the final configuration (n-m).

Therefore, for 3-link robot case, it used mp= 1, n =3 and one degree of freedom of redundancy for the final point, there are nine parameters to be determined.

It should be point out that joint angular acceleration at each intermediate point could be obtained via equation (12). If all the intermediate points are connected by quintic polynomial, there will be eight parameters to be determined. This would be more time-consuming, which is why we choose both quadrinomial and quintic polynomial to generate the segments.

3.2 The GA Motion Planning Scheme

The GA planning scheme renders an optimized trajectory having minimum space, minimum time, while not exceeding a maximum pre-defined torque, without colliding with any obstacle in the workspace. The motion planning adopts direct kinematics to avoid singularity problems. The trajectory parameters are encoded directly, using real codification, as strings (chromosomes) to be used by GA.

For 3R, redundant robot there are nine parameters should be optimized as shown in the following chromosome:

$$[q_1, q_2, q_3, \dot{q}_1, \dot{q}_2, \dot{q}_3, t_1, t_2] \quad (25)$$

Where i q and i q & are intermediate joint angle and velocity for I th joint respectively,t1 is execution time from initial to intermediate via point, and t2 is execution time from intermediate to final point.

A GA starts with a random creation of a population of strings and there after generates successive populations of strings that improve over time. The processes involved in the generation of new populations mainly consist of the following operations that are illustrated

1. Reproduction: Reproduction is a process in which individual strings are copied according to their objective function values, ‘f’ (also called fitness function), which measures profit, utility or goodness that needs to be maximized. Strings with a higher fitness value have a probability of contributing one or more offspring in the next generation. The reproduction operator may be implemented in an algorithmic form in a number of ways such as Roulette wheel selection, rank selection, or steady state selection.

Once a string has been selected for reproduction, an exact replica of the string is made. This string is then entered into the mating pool, a tentative new population for further genetic operator action.

2. Crossover: After reproduction, simple crossover may proceed in two steps. First, members of newly reproduced strings in the mating pool are mated at random. Second, each pair of

strings undergoes crossing over as follows: an integer position ‘k’ along the string is selected uniformly at random between 1 and string length l minus one i.e., (1,l-1) Two new strings are created by swapping all the characters between positions (K+1) and l inclusively.

3. Mutation: Mutation is a random alteration of the value of a string position. In binary coding, this means changing a 1 to 0 and vice versa. In GA, its probability of occurrence is generally kept small, as a higher occurrence rate would lead to a loss of important data. GA, with 100 % mutation rate becomes random search in the solution space.

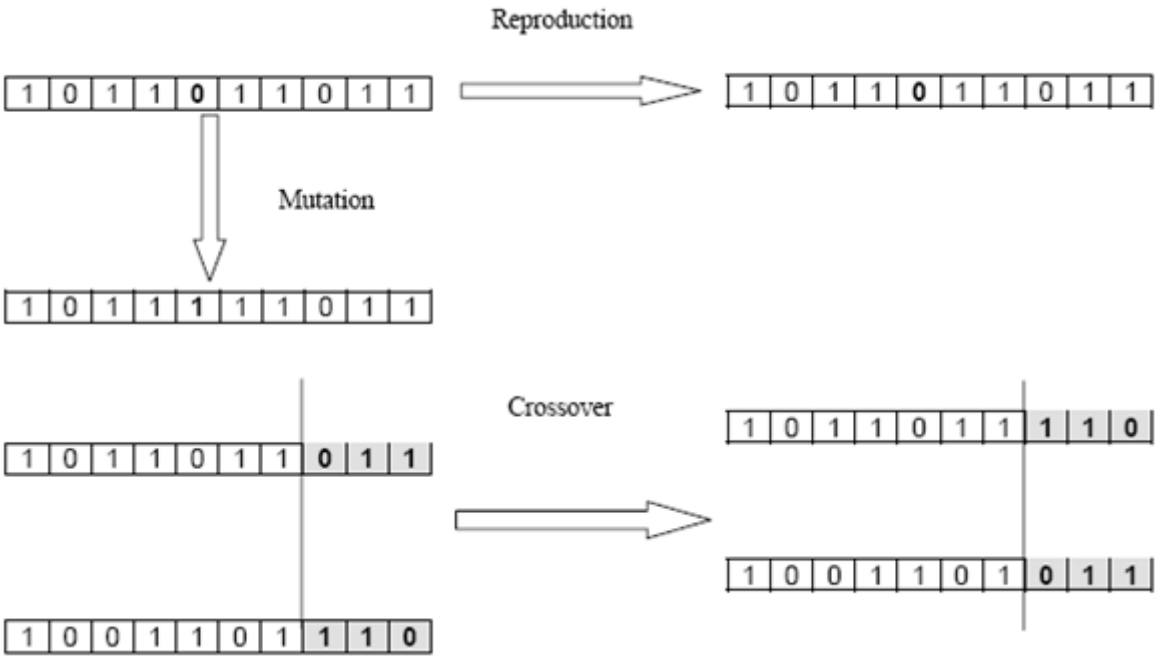


Fig.3.3; Schematic representation of basic genetic algorithm operation

GAs have proven their robustness and usefulness over other search techniques because of their unique procedures that differ from other normal search and optimization techniques in four distinct ways:

1. GAs work with coding of a parameter set, not the parameters themselves.
2. GAs search from a population of points, not a single point.
3. GAs use payoff (objective function) information, not derivative or other auxiliary knowledge.
4. GAs use probabilistic transition rules, not deterministic rules.

In the context of determining a suitable manipulator trajectory, a number of methods have been proposed and researched in recent years. For example, Monteiro and Madrid (1999) have used GA to plan the stages of the trajectory of a robot arm called Jeca III. They have proposed the use of GA to plan a trajectory with obstacle avoidance and implement joint space using classical GA. This is achieved in two stages: initial positioning, which locates the end effector of robot arm in first point of trajectory, and incremental positioning which moves the end effector to the next point of trajectory. Pires and Machado have used GA to generate collision free trajectories for robotic manipulators with the objective to minimize the path length and ripple in time evolution of robot positions and velocities. They have used direct kinematics for this purpose and have presented results for several redundant and non-redundant robot manipulators.

3.3 Problem formulation

For the given the initial and final positions of a robot end effector, the problem of finding an optimal path to be followed is considered in this paper. The problem consists of locating a specific path that requires the least amount of energy amongst several possible paths. It is evident that the end effector, in moving between any two specified end points, can follow a variety of paths. All such paths require different amounts of energy depending upon the distance covered the velocity and the acceleration achieved, and the payload carried. So the objective of this work is to find the path has minimum energy consumption for to move, at desired location, using GA.

3.4 Approach

we will consider a simple mechanism the with three degree of freedom as shown in fig. 3.2 .Where each link can independently rotate. The rotation of the first link θ_1 is measure relative to the reference frame, whereas the rotation of second link $\theta_2$is measure relative to first link, and whereas the rotation of third link $\theta_3$is measure relative to second link.

This would be similar to a robot, where each link's movement is measured relative to a current frame attached to previous link (niku [20]).

Equation that describe the position of point B, as follows;

$$x_C = l_1 cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \quad (26)$$

$$y_C = l_1 sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3) \quad (27)$$

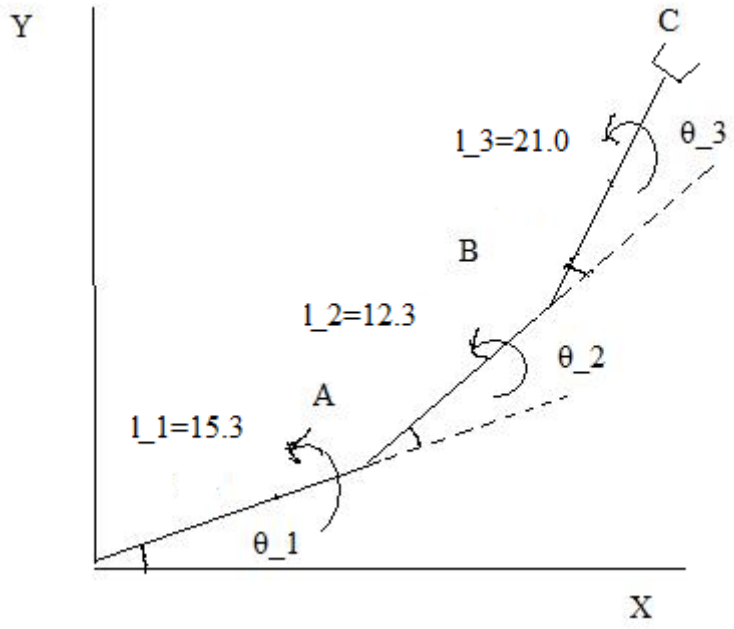


Figure 3.4 Motion planning mechanism

3.5 Operators in genetic algorithm

The initial population of strings is generated at random and the search is then carried out among this population. The evolution of the population elements is non generational, meaning that the new replace the worst elements. The main different operators adopted in the GA are reproduction, crossover, and mutation. In what concerns the reproduction operator, the successive generations of new strings are generated based on their fitness values. In this case, a 5-tournament is used to select the strings for reproduction. With a given probability Pc, the crossover operator adopted the single point technique and, therefore, the crossover point is only allowed between genes or, in other words, the crossover operator cannot disrupt genes. The mutation operator replaces one gene value x (t) with another one generated randomly with a specified range by a given probability Pm.

CHAPTER - 4
INTRODUCTION TO GA

4.1 Introduction to GA

Genetic Algorithms are a family of computational models inspired by evolution. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to these structures so as to preserve critical information. Genetic algorithms are often viewed as function optimizers, although the range of problems to which genetic algorithms have been applied is quite broad.

An implementation of a genetic algorithm begins with a population of (typically random) chromosomes. One then evaluates these structures and allocates reproductive opportunities in such a way that those chromosomes which represent a better solution to the target problem are given more chances to "reproduce" than those chromosomes which are poorer solutions. The "goodness" of a solution is typically defined with respect to the current population.

This particular description of a genetic algorithm is intentionally abstract because in some sense, the term genetic algorithm has two meanings. In a strict interpretation, the genetic algorithm refers to a model introduced and investigated .It is still the case that most of the existing theory for genetic algorithms applies either solely.

In a broader usage of the term, a genetic algorithm is any population-based model that uses selection and recombination operators to generate new sample points in a search space. Many genetic algorithm models have been introduced by researchers largely working from an experimental perspective. Many of these researchers are application oriented and are typically interested in genetic algorithms as optimization tools.

The goal of this presentation is to present genetic algorithms in such a way that students new to this field can grasp the basic concepts behind genetic algorithms as they work through the presentation. It should allow the more sophisticated reader to absorb this material with relative ease. The presentation also covers topics, such as inversion, which have sometimes been misunderstood and misused by researchers new to the field.

Generate a set of random solutions Repeat Test each solution in the set (rank them)

remove some bad solutions from set Duplicate some good solutions make small changes to some of them Until best solution is good enough

4.2 Encode a solution

• Obviously this depends on the problem! GA's *often* encode solutions as fixed length "bit strings" (e.g. 101110, 111111, and 000101). Each bit represents some aspect of the proposed solution to the problem for GA's to work, we need to be able to "test" any string and get a "score" indicating how "good" that solution is

4.3 Example Drilling for Oil

• Imagine you had to drill for oil somewhere along a single 1km desert road

• Problem: choose the best place on the road that produces the most oil per day

• We could represent each solution as a position on the road

• Say, a whole number between [0...1000]

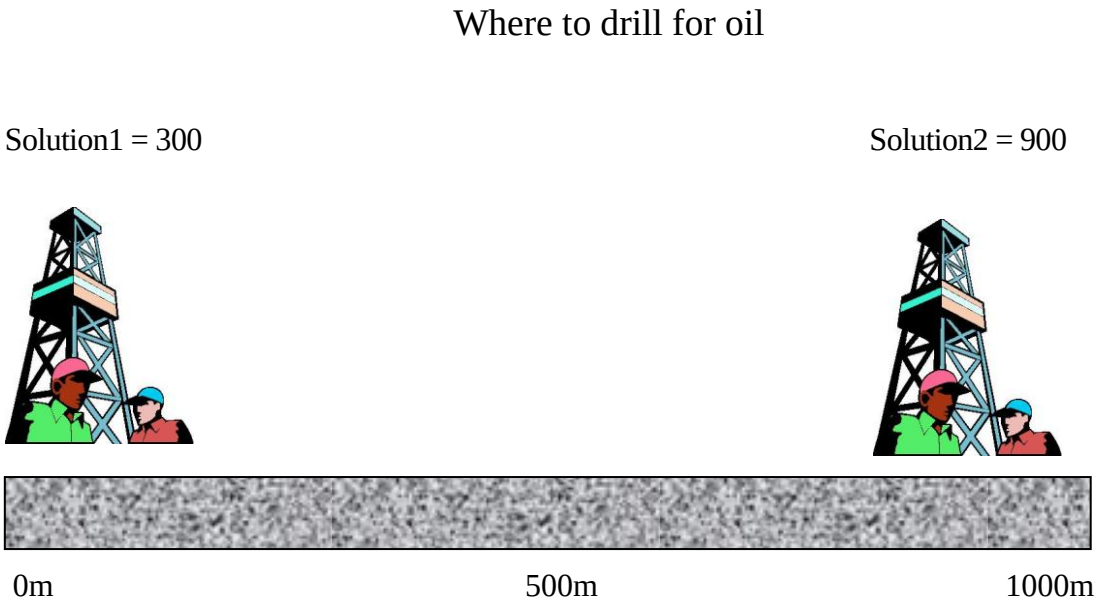


Fig. 4.1: Problem formulation of GA

• The set of all possible solutions [0...1000] is called the *search space* or *state space*

• Often GA's code numbers in binary producing a bit string representing a solution

• In our example we choose 10 bits which is enough to represent 0...1000

Table 4.1 Convert to binary string

	512	256	128	64	32	16	8	4	2	1
900	1	1	1	0	0	0	0	1	0	0
300	0	1	0	0	1	0	1	1	0	0
1023	1	1	1	1	1	1	1	1	1	1

In GA's these encoded strings are sometimes called "genotypes" or "chromosomes" and the individual bits are sometimes called "genes"

Drilling for oil

Solution1 = 300 (0100101100) Location Solution2 = 900(1110000100)

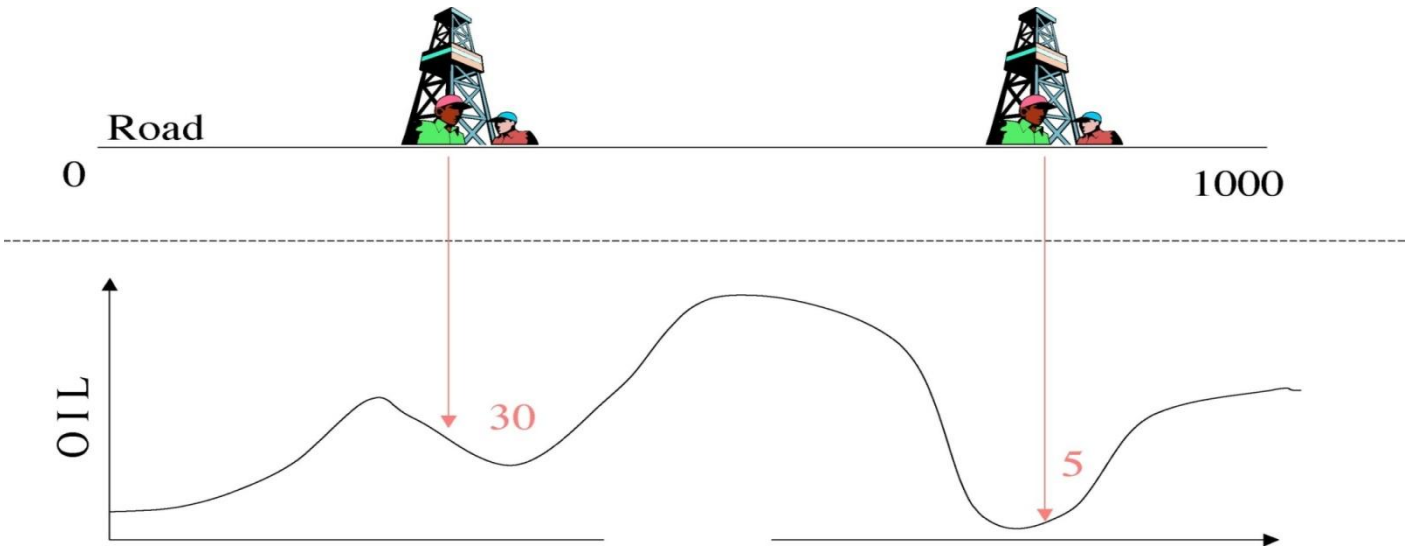


Fig. 4.2: Search problem solution in search space.

Summary

Represent possible solutions as number encoded a number into a binary a string generate a score for each number given a function of "how good" each solution is - this is often called a fitness function .Our oil example is really optimization over a function f(x) where we adapt the parameter x

4.4 Search Space

• For a simple function f(x) the search space is one dimensional.

• But by encoding several values into the chromosome many dimensions can be searched e.g. two dimensions f(x,y)

• Search space can be visualized as a surface or *fitness landscape* in which fitness dictates height

• Each possible genotype is a point in the space

• A GA tries to move the points to better places (higher fitness) in the space

• Obviously, the nature of the search space dictates how a GA will perform

• A completely random space would be bad for a GA

• Also GA's can get stuck in local maxima if search spaces contain lots of these

• Generally, spaces in which small improvements get closer to the global optimum are good

4.4.1 Adding Sex – Crossover

• Although it may work for simple search spaces our algorithm is still very simple

• It relies on random mutation to find a good solution

• It has been found that by introducing "sex" into the algorithm better results are obtained

• This is done by selecting two parents during reproduction and combining their genes to produce offspring

• Two high scoring "parent" bit strings (*chromosomes)* are selected and with some probability (crossover rate) combined

• Producing two new *offspring* (bit strings)

• Each offspring may then be changed randomly (*mutation*)

4.4.2 Selecting Parents

Many schemes are possible so long as better scoring chromosomes more likely selected score is often termed the *fitness* "Roulette Wheel" selection can be used:

– Add up the fitness's of all chromosomes

– Generate a random number R in that range

– Select the first chromosome in the population that - when all previous fitness's are added - gives you at least the value

Table 4.2 Population for Selecting Parents

No.	Chromosome	Fitness
1	1010011010	1
2	1111100001	2
3	1011001100	3
4	1010000000	1
5	0000010000	3
6	1001011111	5
7	0101010101	1
8	1011100111	2

4.4.3 Roulette wheel selection

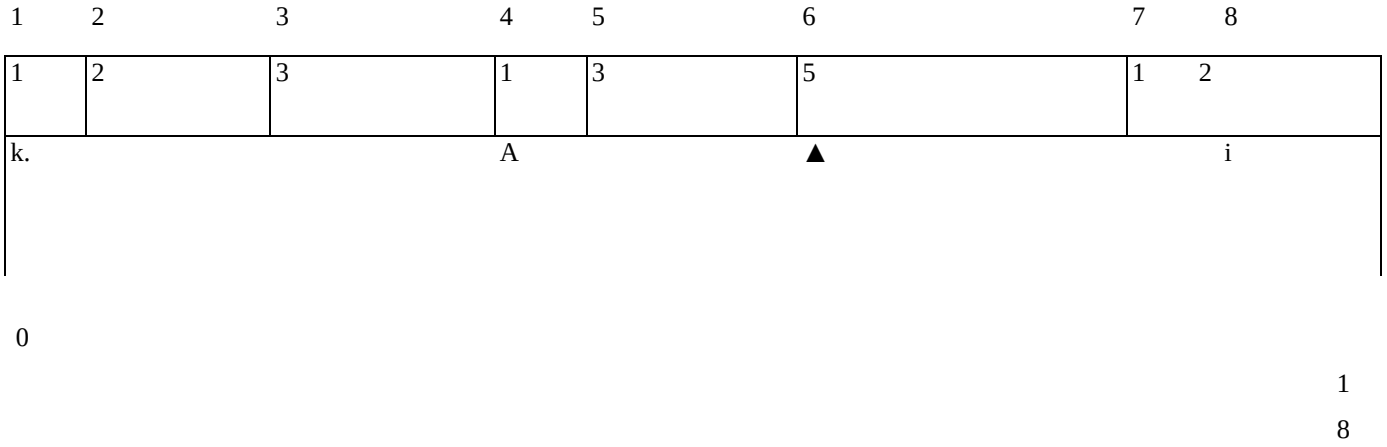


Fig. 4.3 Roulette wheel selection

4.4.4 Crossover – Recombination

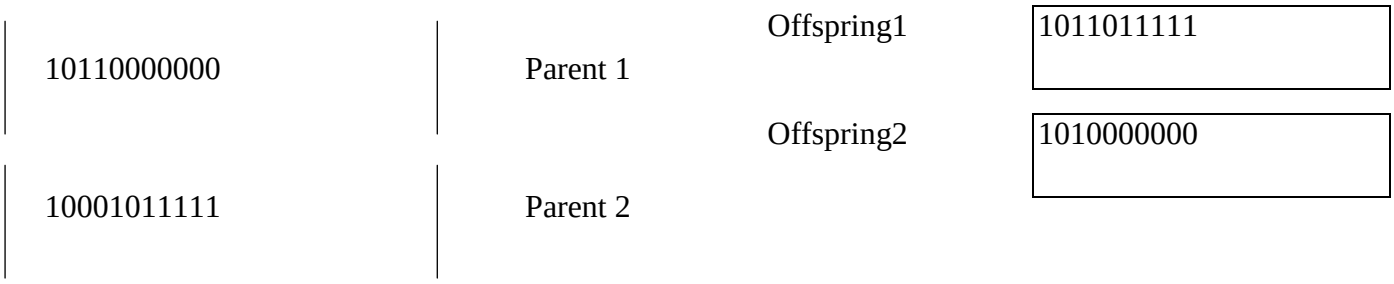


Fig. 4.4: Crossover- recombination

Single point Crossover random with some high probability (crossover rate) apply crossover to the parents. (Typical values are 0.8 to 0.95)

4.4.5 Mutation

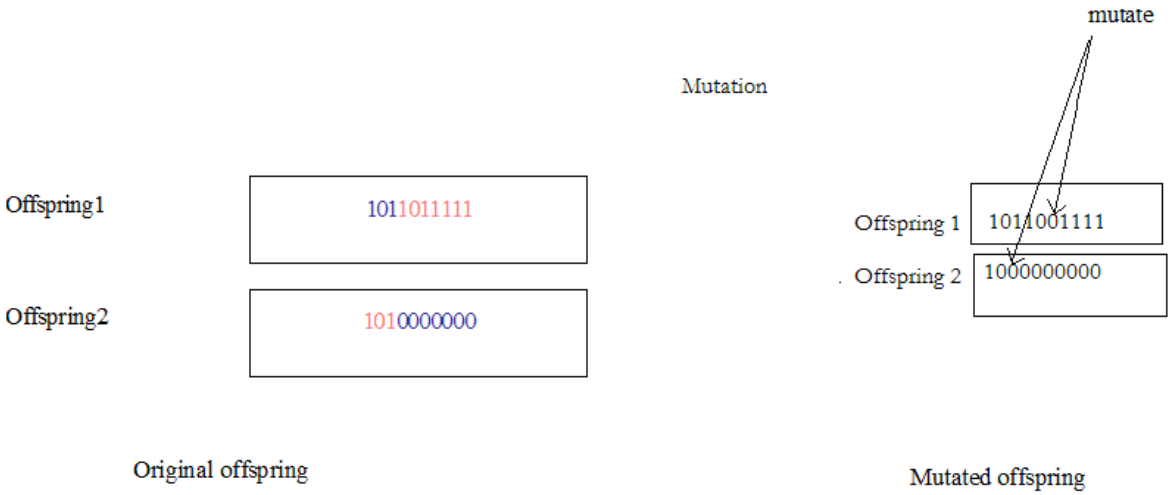


Fig. 4.5: Mutation

4.4.6 Algorithm

Generate a *population* of random chromosomes Repeat (each generation) Calculate fitness of each chromosome Repeat Use roulette selection to select pairs of parents Generate offspring with crossover and mutation until a new population has been produced

Until best solution is good enough

4.4.7 Many parameters to set

Any GA implementation needs to decide on a number of parameters: Population size (N), mutation rate (m), crossover rate (c), Often these have to be "tuned" based on results obtained - no general theory to deduce good values. Typical values might be: N = 20, m = 0.01, c = 0.8.

4.4.8 Crossover work

• A lot of theory about this and some controversy

• Holland introduced “Schema” theory

• The idea is that crossover preserves “good bits” from different parents, combining them to produce better solutions

• A good encoding scheme would therefore try to preserve “good bits” during crossover and mutation.

4.4.9 Genetic Programming

• When the chromosome encodes an entire program or function itself this is called genetic programming (GP)

• In order to make this work encoding is often done in the form of a tree representation

• Crossover entails swapping sub trees between parents

4.5 Genetic Programming

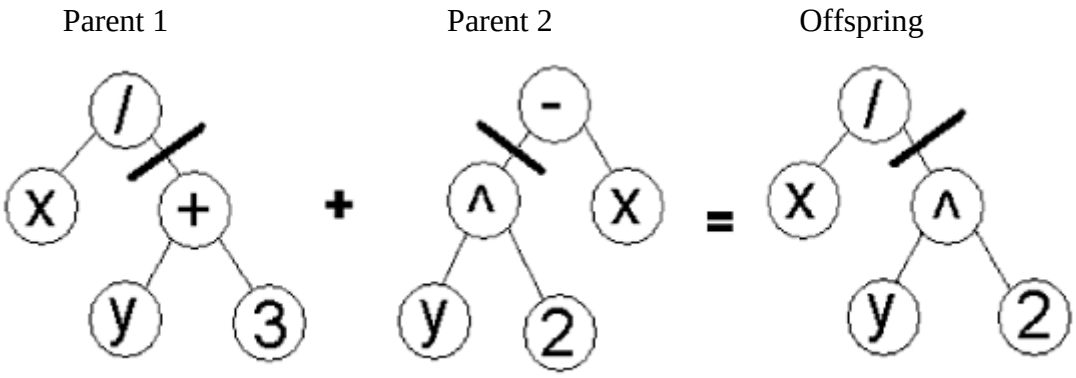


Fig. 4.6: Genetic Programming.

It is possible to evolve whole programs like this but only small ones. Large programs with complex functions present big problems

4.6 Fitness functions

• Most GA's use explicit and static fitness function (as in our "oil" example)

• Some GA's (such as in Artificial Life or Evolutionary Robotics) use dynamic and implicit fitness functions - like *"how many obstacles did I avoid"*

• In these latter examples other chromosomes (robots) effect the fitness function

4.7 Applications

Genetic algorithms have wide application areas but are more useful in problems where the search space is large, complex or poorly understood, also when traditional search methods fail or no mathematical analysis is available. A few examples are

- Chemistry
- Engineering
- Game playing
- Molecular Biology
- Robotics

In the address below we have a study done on the travelling salesman problem/ant colony by using genetic algorithms and relating to other optimization algorithms

4.8 Advantages and Disadvantages

One of the advantages of genetic algorithms is that it is parallel because they have multiple offspring thus making it ideal for large problems where evaluation of all possible solutions in serial would be too time taking, if not impossible.

They perform well in problems where the fitness landscape is complex, where the fitness function is discontinuous, noisy, changes over time or has many local optima.

They are able to solve problems knowing nothing about the problem from the start.

Despite all advantages of genetic algorithms they have to be well written and thought out for them to properly work and a few things to take in consideration are:

- A proper representation language for the possible input solutions
- The fitness function has to be written so that higher fitness is achieved and does result in a

better solution for the given problem

- All parameters such as the size of population, rate of mutation and crossover among others.

A problem that can also occur with genetic algorithms known premature convergence which means that an individual that is more fit than others at early stages may dominate on the reproduction process leading to a local optimum convergence rather than a more thorough search that could lead to a global optimum.

CHAPTER 5
RESULT AND DISCUSSION

5.1: Result

We will consider a simple mechanism with three degree of freedom as shown in fig. 5.1 where each link can independently rotate. The rotation of the first link θ_1 is measured relative to the reference frame, the rotation of second link $\theta_2$is measured relative to first link, and the rotation of third link $\theta_3$is measure relative to second link.

This would be similar to a robot, where each link's movement is measured relative to a current frame attached to previous link.

Equation that describe the position of point C;

$$x_C = l_1 cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \quad (28)$$

$$y_C = l_1 sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3) \quad (29)$$

The Jacobian matrix for a planar robot with two degrees-of-freedom is

$$J(\theta_1 + \theta_2 + \theta_3) = \begin{bmatrix} l_1 sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3) \\ l_1 cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \end{bmatrix} \quad (30)$$

Giving value of l_1 , l_2 &l_3 i.e. 15.3, 12.3 & 21.0 cm respectively;

Initial Angle $\theta_1, \theta_2, \theta_3$. 225.3470, 209.6582, 217.2948 degree respectively;

x_{iC} =15.3 cos225.3470 + 12.3 cos(225.3470 + 209.6582) + 21 cos(225.3470 + 209.6582 +217.2948*)*

y_{iC} = 15.3sin225.3470 + 12.3 sin(225.3470 + 209.6582) + 21 sin(225.3470 + 209.6582 +217.2948*)*

x_{iC} =19.000;

y_{iC}=25.000;

Final Angle $\theta_1, \theta_2, \theta_3$ 338.9899, 232.5404, 234.4757;

x_{fC} =15.3 $cos338.9899 + 12.3\cos(338.9899 + 232.5404) + 21\cos(338.9899 + 232.5404 + 234.4757)$

$y_{fC} = 15.3sin338.9899 + 12.3\sin(338.9899 + 232.5404) + 21\sin(338.9899 + 232.5404 + 234.4757)$

x_{fC} =11.000;

y_{fC} =6.000;

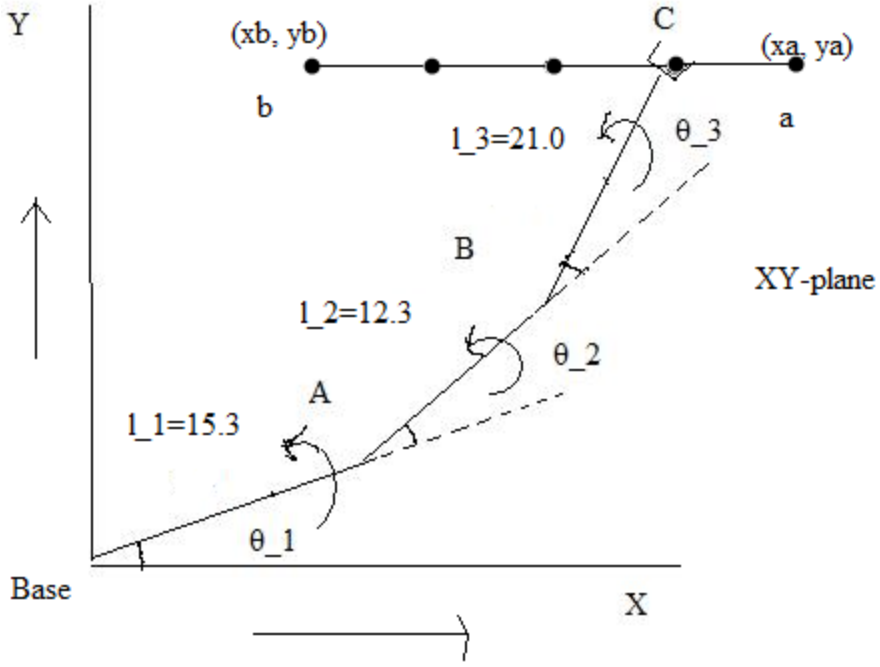
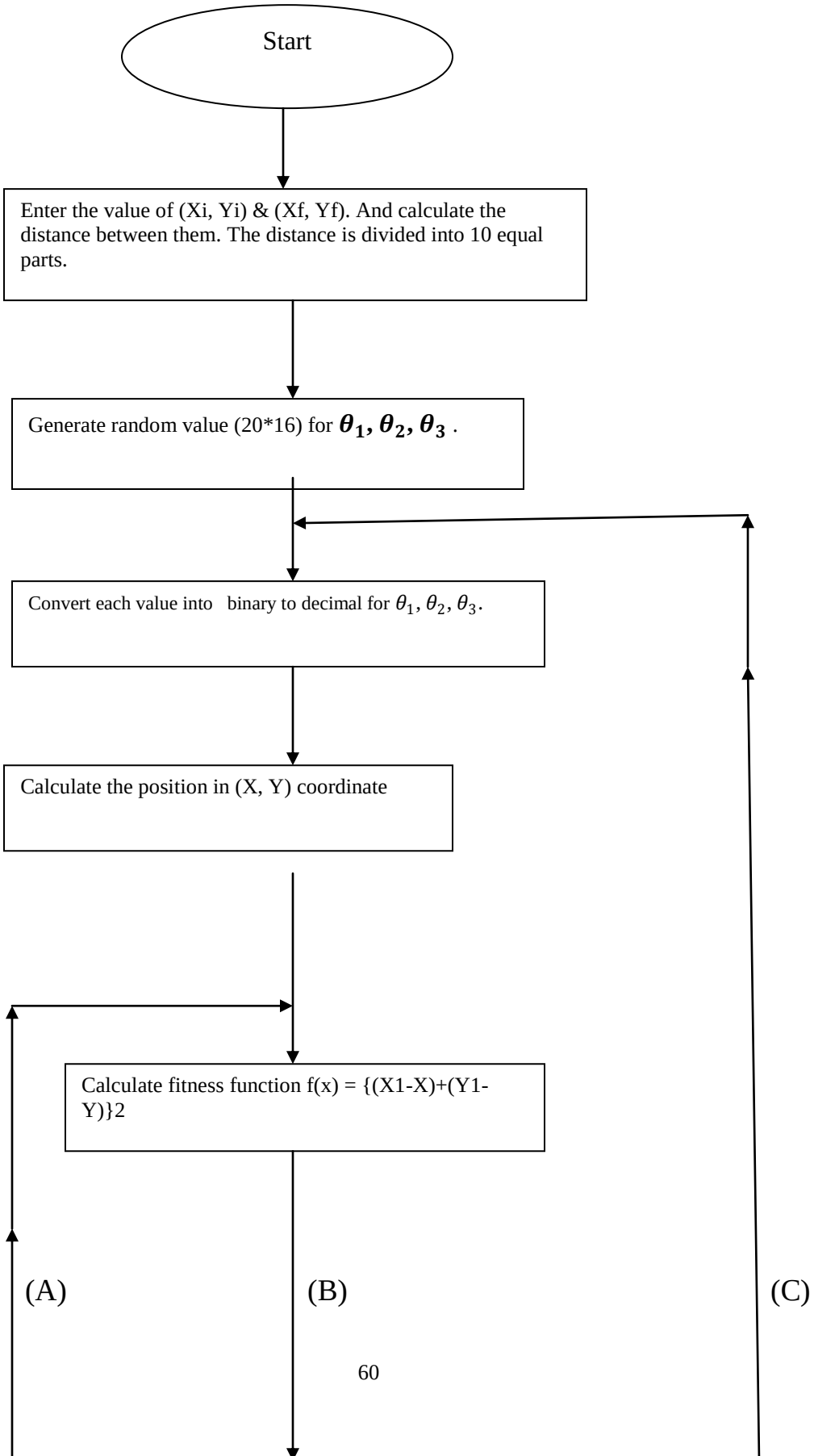
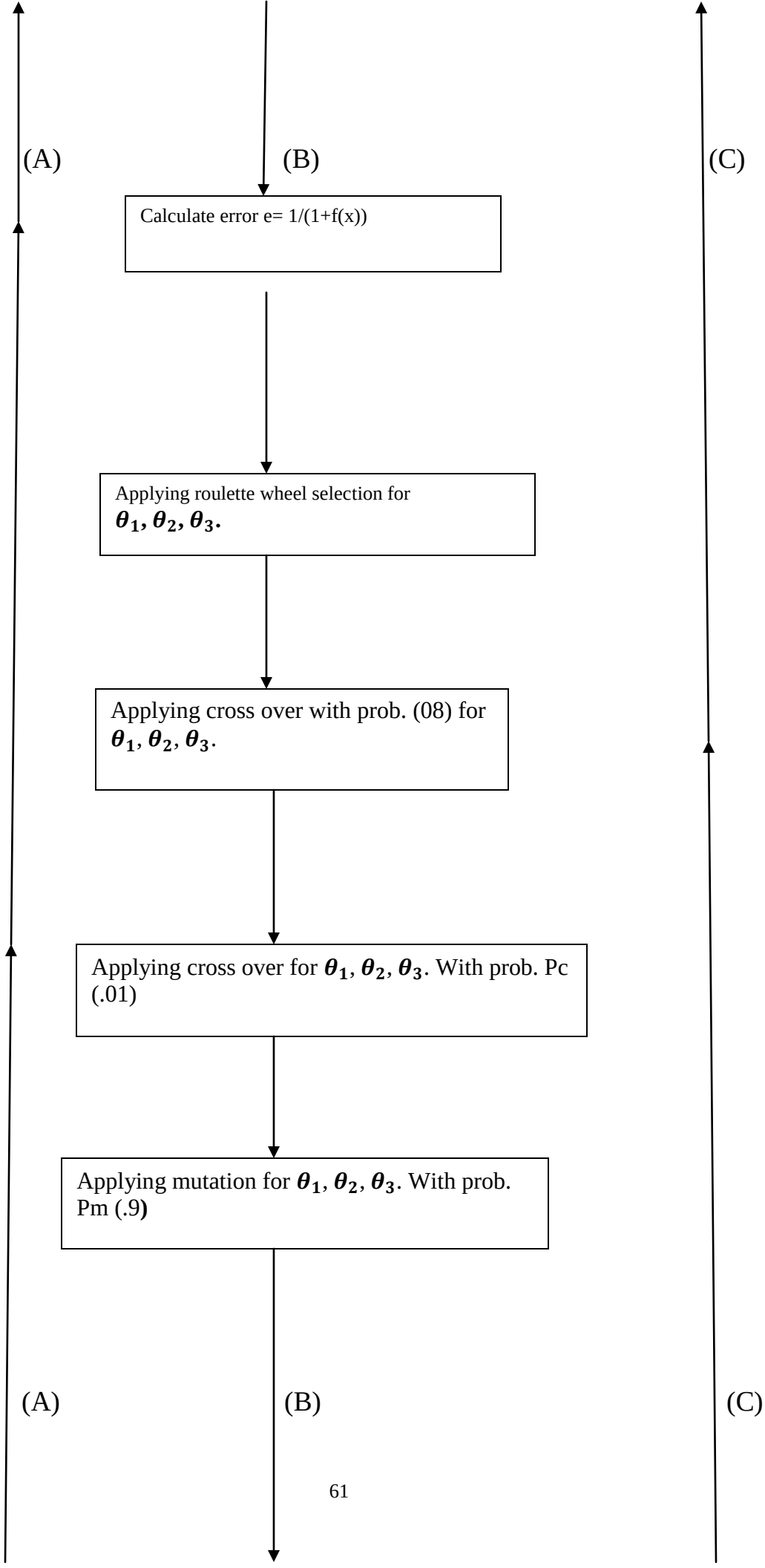


Figure 5.1: Three-degrees-freedom planar robot in Cartesian space

5.2 Methodology




(A)
(B)
(C)
Calculate error e= 1/(1+f(x))
Applying roulette wheel selection for $\boldsymbol{\theta_1, \theta_2, \theta_3}$.
Applying cross over with prob. (08) for $\boldsymbol{\theta_1}$, $\boldsymbol{\theta_2}$, $\boldsymbol{\theta_3}$.
Applying cross over for $\boldsymbol{\theta_1}$, $\boldsymbol{\theta_2}$, $\boldsymbol{\theta_3}$. With prob. Pc (.01)
Applying mutation for $\boldsymbol{\theta_1}$, $\boldsymbol{\theta_2}$, $\boldsymbol{\theta_3}$. With prob. Pm (.9)
(A)
(B)
(C)

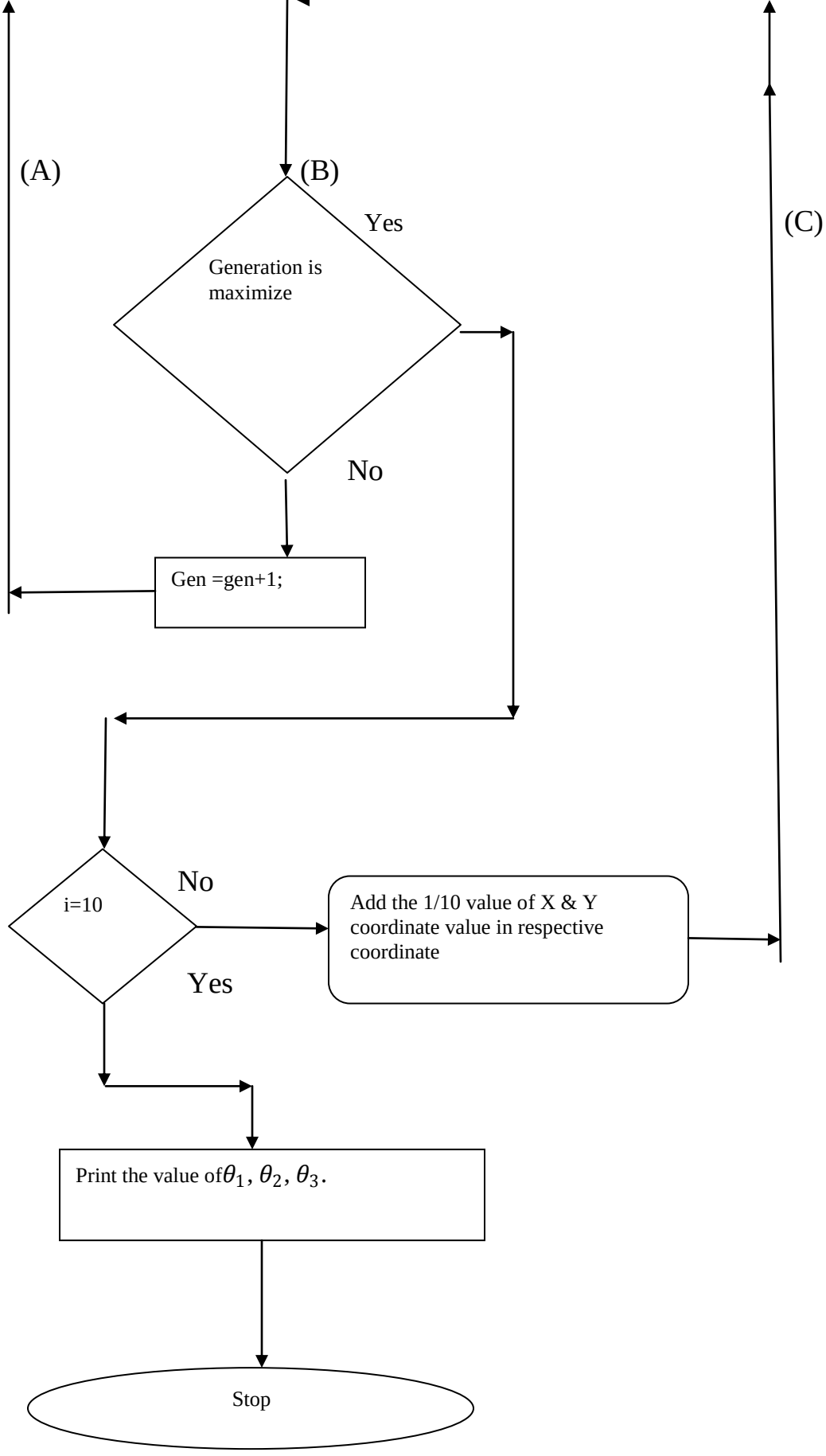


Fig. 5.2: Flow chart of program

5.3: Observation & Calculations

$N = 3*(\theta_1) + 2*(\theta_2) + 1*(\theta_3)$ (31)

We assume that the first angle moved is very small than the second & the second angle moved is less than the third one. This selection was due to the reason that for the first motor the power requirement is greatest, lesser for the second motor and least for the third motor, because the load associated with the motors is decreasing respectively.

Table 5.1 Value of angles for position with error and energy {(19, 25) to (11, 6)}.

S. No.	Weightage Angle(θ_1)	Weightage Angle(θ_2)	Weightage Angle(θ_3)	Angle(θ_1)	Angle(θ_2)	Angle(θ_3)	Error	N(Total summation)
1.	3	2	1	225.3470	209.6582	217.2948	0.0594	1312.652
2.	3	2	1	101.4989	184.1073	122.8752	0.0749	795.5865
3.	3	2	1	80.8388	336.9717	63.0903	0.0757	979.5501
4.	3	2	1	258.6703	122.5045	132.9318	1.3218	1153.952
5.	3	2	1	176.4810	38.0379	153.6990	0.0858	759.2178
6.	3	2	1	173.0053	322.0415	30.4428	0.3147	1193.542
7.	3	2	1	214.5881	208.6646	315.3812	1.6802	1376.475
8.	3	2	1	11.2022	70.2733	259.6397	0.4950	433.7929
9.	3	2	1	348.9809	209.9736	176.4569	0.1599	1643.347
10.	3	2	1	338.9899	232.5404	234.4757	0	1716.526

Table 5.2: Value of angles for position with error and energy {(25, 19) to (6, 11)}

S. No.	Weightage Angle(θ_1)	Weightage Angle(θ_2)	Weightage Angle(θ_3)	Angle(θ_1)	Angle(θ_2)	Angle(θ_3)	Error	N(Total summation)
1.	3	2	1	36.0439	85.4152	190.2691	0.1178	469.2312
2.	3	2	1	331.6376	255.1462	153.3802	0.5720	1658.585
3.	3	2	1	171.1563	160.3732	332.2432	0.2683	1166.459
4.	3	2	1	49.4094	147.6616	160.4492	1.7245	604.0006
5.	3	2	1	289.3307	290.6674	47.3013	0.5639	1496.628
6.	3	2	1	15.0130	29.4354	225.2883	0.1208	329.1981
7.	3	2	1	238.3602	114.7263	344.4666	0.7416	1289
8.	3	2	1	135.9022	257.6664	109.5442	1.2593	1032.584
9.	3	2	1	282.6221	70.7373	300.8034	0.1627	1290.144
10.	3	2	1	101.1282	114.2899	254.0317	0	785.9961

Table 5.3: Value of angles for position with error & energy {(12, 25) to (25, 12)}

S. No.	Weightage Angle(θ_1)	Weightage Angle(θ_2)	Weightage Angle(θ_3)	Angle(θ_1)	Angle(θ_2)	Angle(θ_3)	Error	N(Total summation)
1.	3	2	1	54.4118	130.3875	269.8207	0.0594	693.8311
2.	3	2	1	175.1962	228.2517	207.1760	0.0749	1189.268
3.	3	2	1	35.4970	181.7529	63.4921	0.0757	533.4889
4.	3	2	1	41.2190	286.6309	55.9142	1.3218	752.833
5.	3	2	1	150.6251	311.4311	209.6167	0.0858	1284.354
6.	3	2	1	160.3006	159.3796	85.3185	0.3147	884.9795
7.	3	2	1	162.3465	226.7044	5.0888	1.6802	945.5371
8.	3	2	1	277.4746	205.0610	59.0849	0.4950	1301.631
9.	3	2	1	106.0719	282.5530	353.4744	1.4528	1236.796
10.	3	2	1	69.5054	333.7008	139.9905	0	1015.908

Table 5.4: Value of angles for position with error & energy {(12, 12) to (25, 25)}

S. No.	Weightage Angle(θ_1)	Weightage Angle(θ_2)	Weightage Angle(θ_3)	Angle(θ_1)	Angle(θ_2)	Angle(θ_3)	Error	N(Total summation)
1.	3	2	1	16.9587	164.0355	230.7961	0.1178	609.7432
2.	3	2	1	280.6729	66.0676	301.4643	0.5720	1275.618
3.	3	2	1	342.8606	258.8534	338.9312	0.2683	1885.22
4.	3	2	1	82.8950	13.3690	358.5563	1.7245	633.9793
5.	3	2	1	341.3339	163.8524	192.7478	0.5639	1544.454
6.	3	2	1	293.3568	4.0641	286.2740	0.1208	1174.473
7.	3	2	1	189.1995	0.0311	337.8340	0.7416	905.4947
8.	3	2	1	278.6788	260.8993	323.7996	1.2593	1681.635
9.	3	2	1	336.2728	289.1511	69.7057	0.1627	1656.826
10.	3	2	1	93.8693	220.7706	49.4957	0	772.6448

Table 5 .5: Value of angles for position with error and energy {(15, 28) to (25, 18)}.

S. No.	Weightage Angle(θ_1)	Weightage Angle(θ_2)	Weightage Angle(θ_3)	Angle(θ_1)	Angle(θ_2)	Angle(θ_3)	Error	N(Total summation)
1.	3	2	1	222.6230	267.4456	1.7201	0.0594	1204.48
2.	3	2	1	173.0640	356.6279	326.6110	0.0749	1559.059
3.	3	2	1	37.0823	78.1908	275.6394	0.0757	543.2679
4.	3	2	1	241.0738	97.0330	340.8354	1.3218	1258.123
5.	3	2	1	81.7426	328.3415	235.4174	0.0858	1137.328
6.	3	2	1	214.6295	108.7994	124.0311	0.3147	985.5184
7.	3	2	1	3.8488	212.6976	112.1335	1.6802	549.0751
8.	3	2	1	124.4433	11.3093	62.1106	0.4950	458.0591
9.	3	2	1	65.9916	195.9358	153.2213	0.1599	743.0677
10.	3	2	1	255.0529	36.0542	115.9202	0	953.1873

Table 5.6: Theoretical value of angle and coordinate when the robot is moved manually.

S.No.	Position	Distance covered (unit)	Initial angles			Final angles			N(Total summation)
			Angle (θ_1)	Angle (θ_2)	Angle (θ_3)	Angle (θ_1)	Angle (θ_2)	Angle (θ_3)	
1.	(19, 25) (11, 6)	27	116.00	310.50	292.59	245	222.54	253.25	1347.46
2.	(25, 19) (6, 11)	27	123.15	228.39	8.46	215.92	247.28	256.80	1116.90
3.	(12, 25) (25, 12)	26	157.92	268.21	293.87	99.34	199.43	66.22	1033.57
4.	(12, 12) (25, 25)	26	209.81	220.84	289.35	120.96	257.43	341.61	1289.90
5.	(15, 28) (25, 18)	20	142.78	278.1	299.11	114.46	256.77	318.77	1229.67

Energy E=0.08*N (32)

Where 0.08 is voltage per degree

Energy E=VI cosØ (33)

Where cosØ =1 & I= 1 A, (Assume)

Table 5.7: Comparison of energy consumed using GA and without using GA

S. No.	Using GA		Without using GA	
	E (volt)	Distance (cm)	E (volt)	Distance (cm)
(19, 25) (11, 6)	75.12	20.615	107.79	27
(25, 19) (6, 11)	80.97	20.615	89.35	27
(12, 25) (25, 12)	78.99	18.385	82.64	26
(12, 12) (25, 25)	83.6	18.385	103.19	26
(15, 28) (25, 18)	75.12	14.142	98.32	20

Table 5.8 Fitness function and generation

S. No.	Fitness function	Generation
1.	0.85	0
2.	0.9	100
3.	1	200
4.	1	300
5.	1	400
6.	1	500

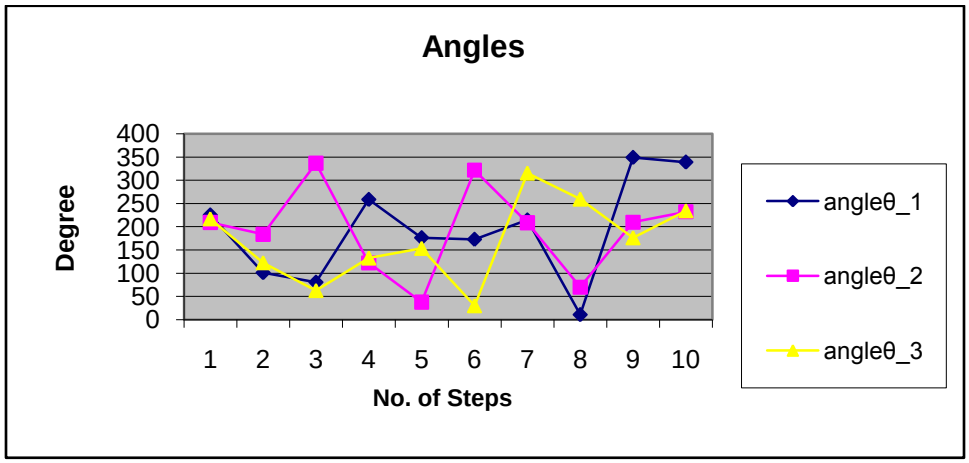


Figure 5.3: Angle of joints for {(19, 25) to (11, 6)}

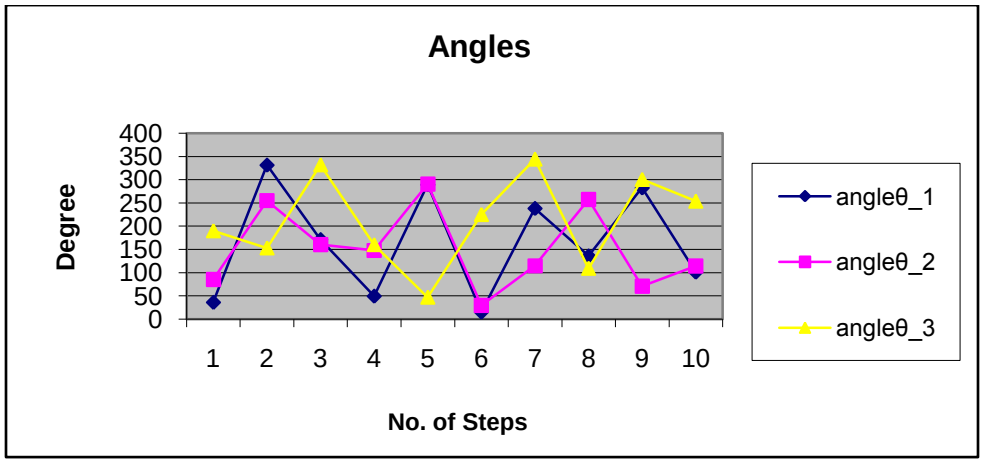


Figure 5.4: Angle of joints for {(25, 19) to (6, 11)}

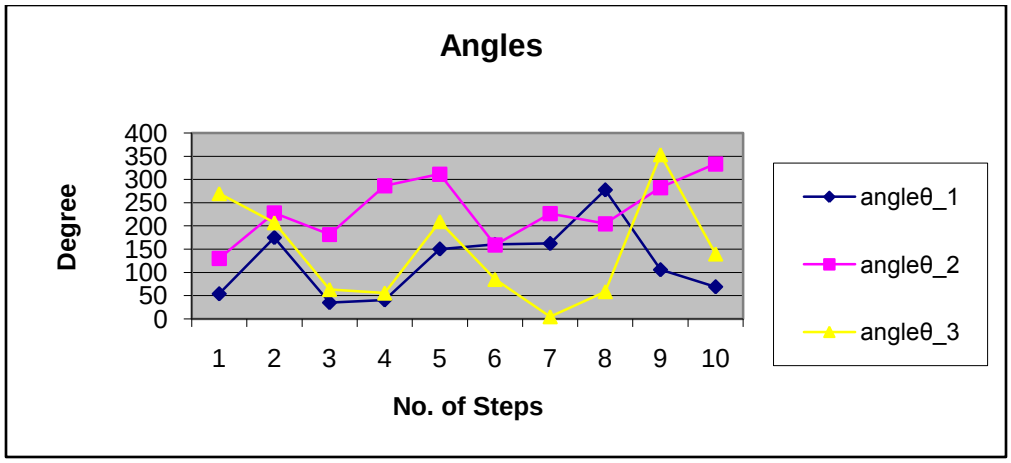


Figure 5.5: Angle of joints for {(12, 25) to (25, 12)}

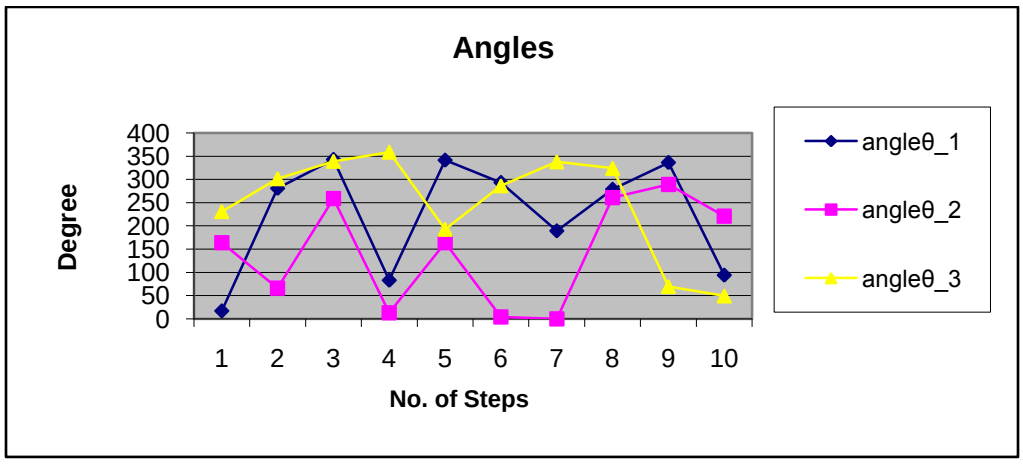


Figure 5.6: Angle of joints for {(12, 12) to (25, 25)}

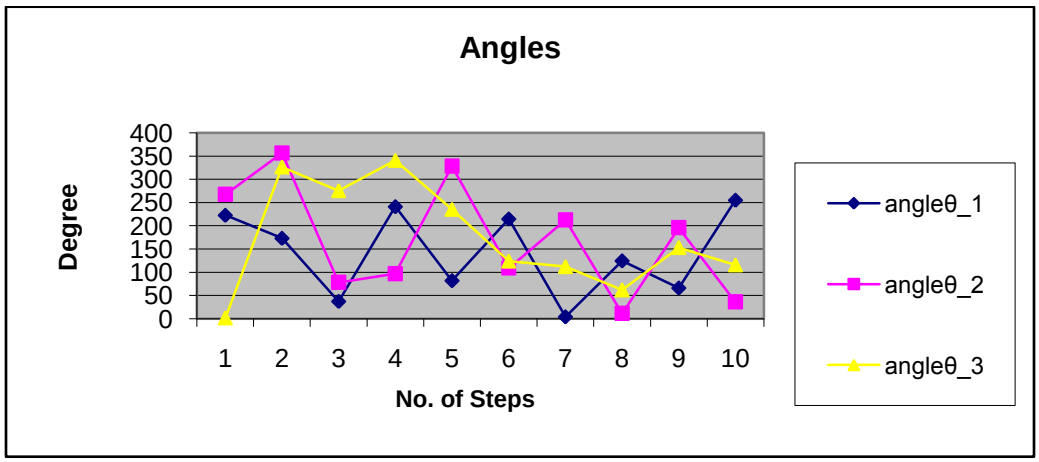


Figure 5.7: Angle of joints for {(15, 28) to (25, 18)}

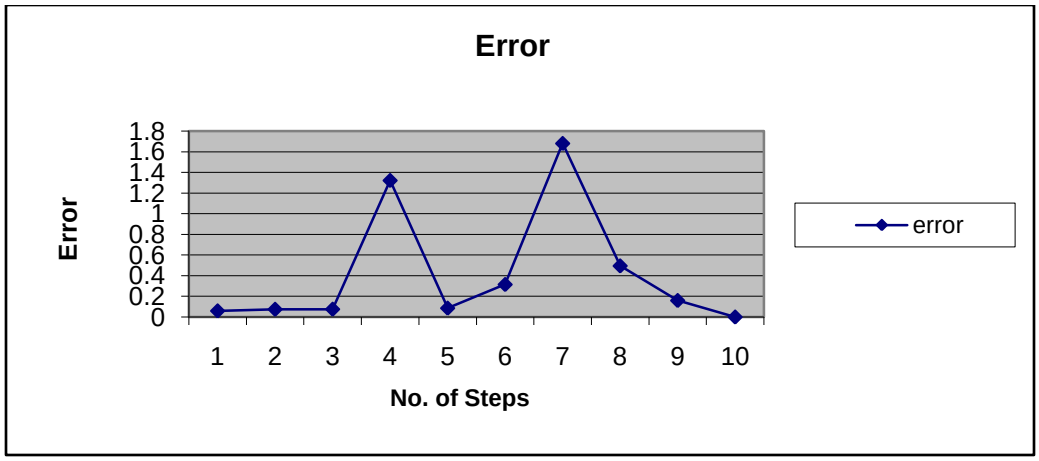


Figure 5.8: Graph between error and no. of step {(19, 25) to (11, 6)}

Figure 5.9: Graph between error and no. of step {(25, 19) to (6, 11)}

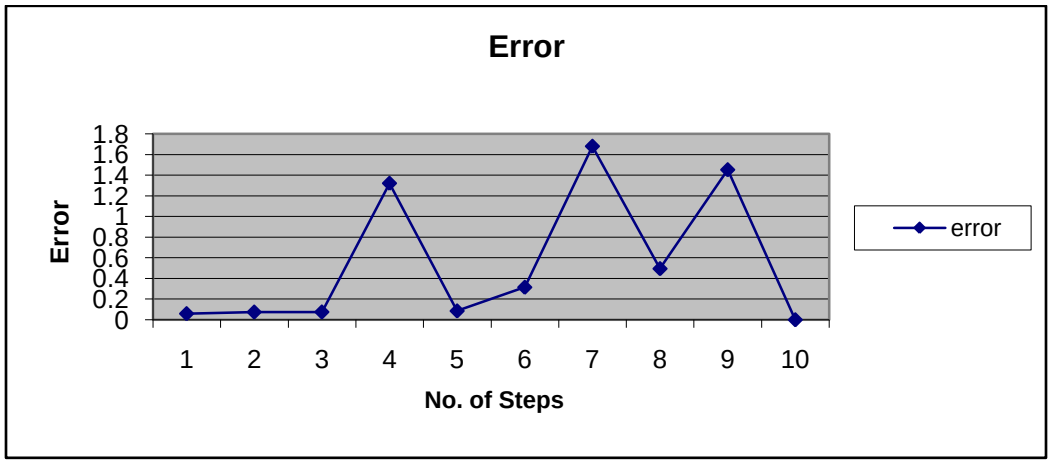


Figure 5.10: Graph between error and no. of step {(12, 25) to (25, 12)}

Figure 5.11: Graph between error and no. of step {(12, 12) to (25, 25)}

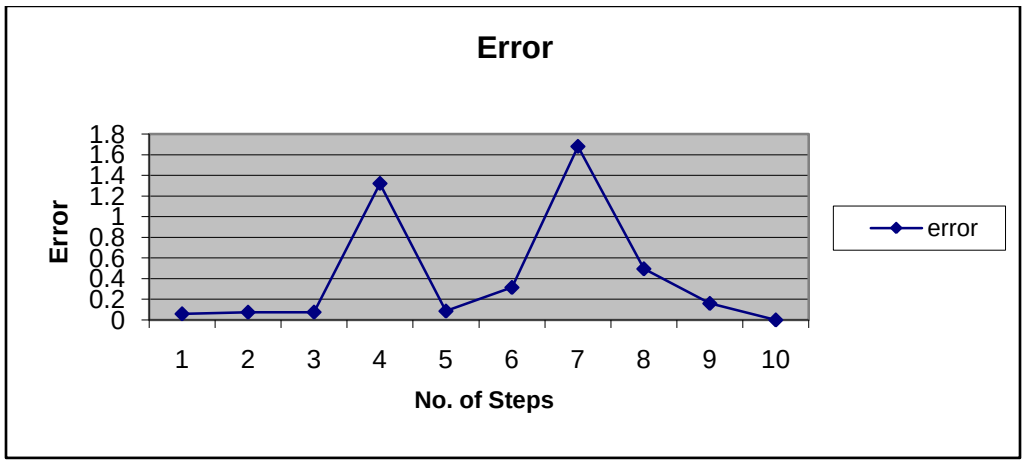


Figure 5.12: Graph between error and no. of step {(15, 28) to (25, 18)}

.

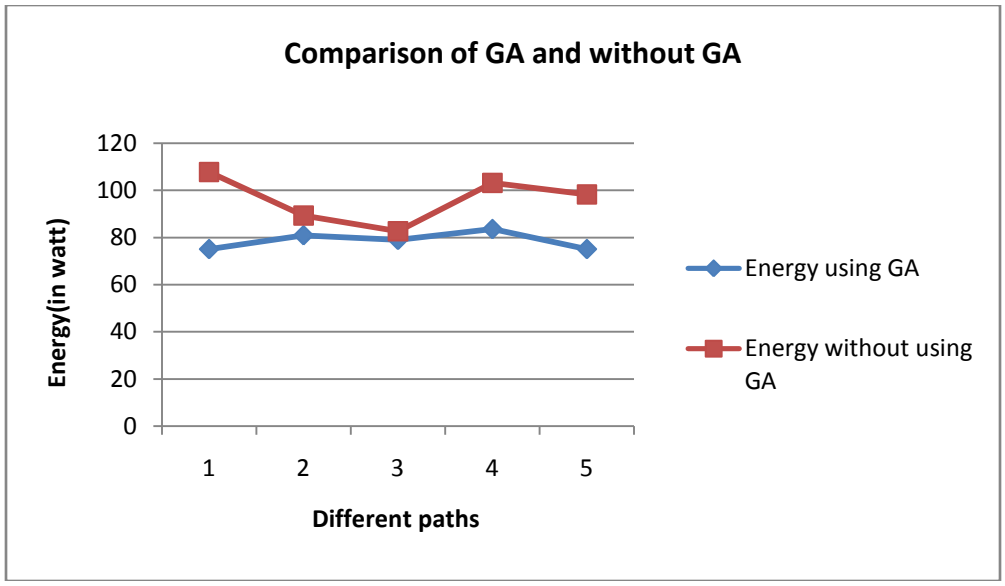


Figure 5.13: Energy comparison between using GA and without GA.

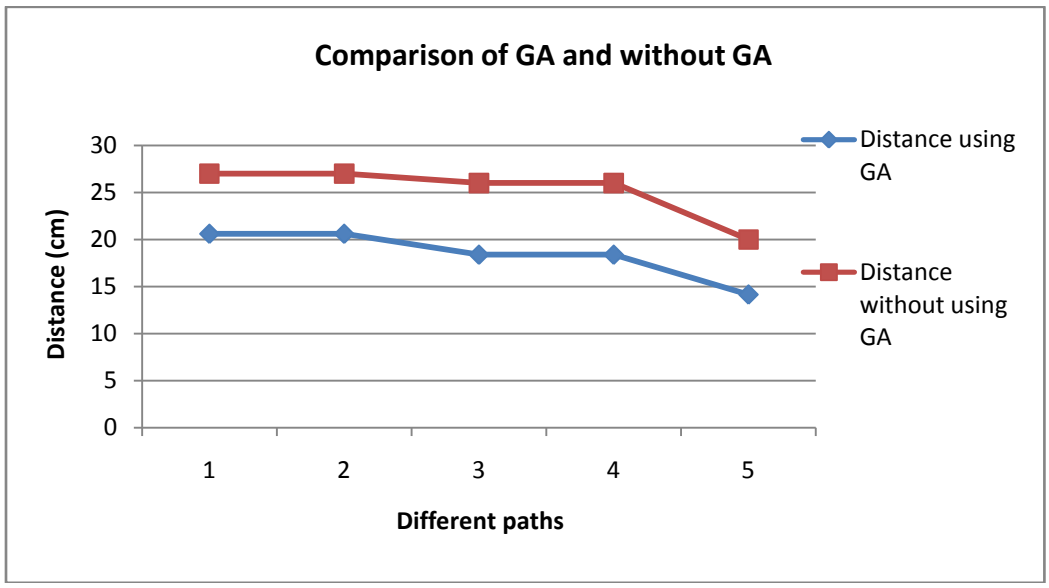


Figure 5.14: Distance comparison between using GA and without GA.

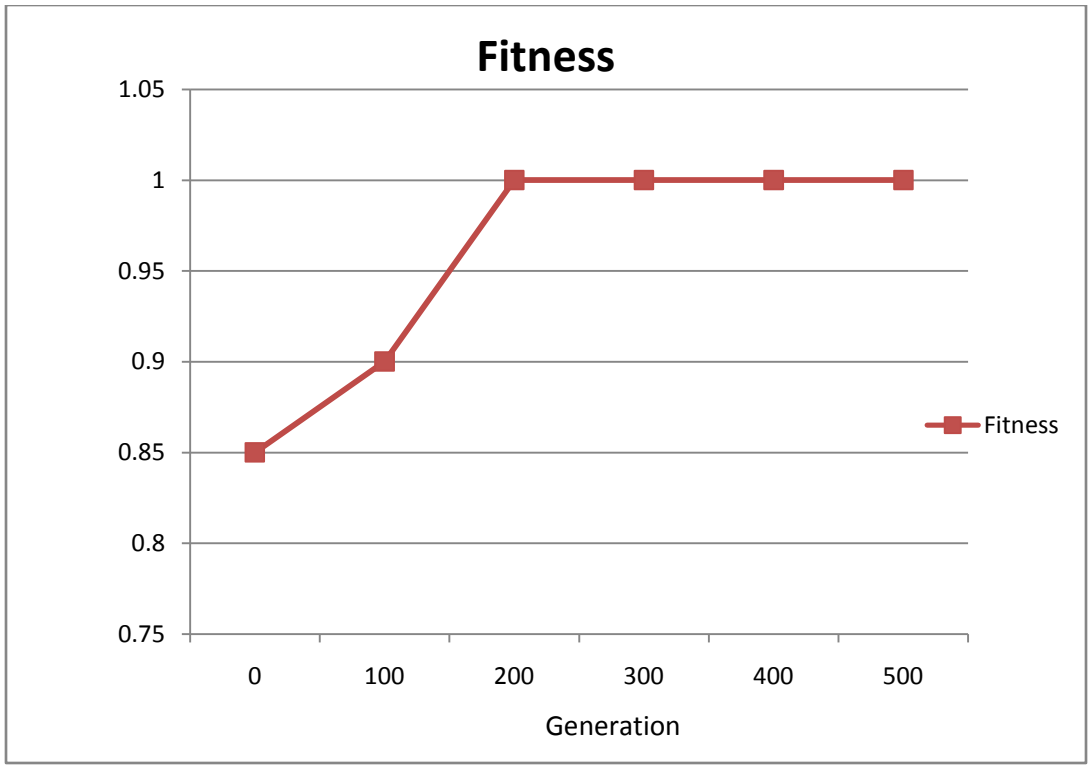


Figure 5.15: Graph b/w fitness function and generation

Calculate the value of E

From Table 5.1 calculate total movement of angle i.e. N for the position {(19, 25) to (11, 6)}.

Energy E=0.08*N

Where 0.08 is voltage per degree

Energy E=VI cosØ

E=sum/10 (average)

E=75.12 watt

From Table 5.2 calculate total movement of angle i.e. N for the position {(25, 19) to (6, 11)}

Energy E=0.08*N

Where 0.08 is voltage per degree

Energy E=VI cosØ

E=sum/10 (average)

E=80.97 watt

From Table 5.3 calculate total movement of angle i.e. N for the position {(12, 25) to (25, 12)}

Energy E=0.08*N

Where 0.08 is voltage per degree

Energy E=VI cosØ
E=sum/10 (average)

E=78.99 watt

From Table 5.4 calculate total movement of angle i.e. N for the position {(12, 12) to (25, 25)}

Energy E=0.08*N

Where 0.08 is voltage per degree

Energy E=VI cosØ
E=sum/10 (average)

E=83.6 watt

From Table 5.5 calculate total movement of angle i.e. N for the position {(15, 28) to (25, 18)}.

Energy E=0.08*N

Where 0.08 is voltage per degree

Energy E=VI cosØ
E=sum/10 (average)

E=75.12 watt

Calculate the value of Distance

Distance =$\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$ (34)

Distance between position {(19, 25) to (11, 6)}.

Distance = 20.615

Distance between position (25, 19) (6, 11)

Distance = 20.615

Distance between position (12, 25) (25, 12)

Distance = 18.385

Distance between position (12, 12) (25, 25)

Distance = 18.385

Distance between position (15, 28) (25, 18)

Distance = 14.142

CHAPTER-6
CONCLUSION & FUTURE SCOPE

From the Table 5.1, 5.2, 5.3, 5.4 and 5.5 find the value of Energy, Distance and error using GA for different coordinates. Table 5.6 calculates the theoretical value of Energy and Distance.

Table 5.7 Comparison of energy consumed using GA and without using GA. When we compare the Energy and Distance using GA and without using GA the Energy consume is less and the Distance cover is also less so we save the energy using GA.

Table 5.8 fitness function and generation. We find the value of fitness function with respect to generation.

Figure 5.3, Figure 5.4, Figure 5.5, Figure 5.6 and Figure 5.7 see the movement of Angle of joints $\theta_1, \theta_2, \theta_3$ {(19, 25) to (11, 6)}, {(25, 19) to (6, 11)}, {(12, 25) to (25, 12)}, {(12, 12) to (25, 25)},{(15, 28) to (25, 18)} respectively. We see the movement of angle for idea of how much each angle is move. The range of angle θ_1 is for {(19, 25) to (11, 6)} is 11.2022 to 338.9899, angle θ_2 is 38.0379 to 336.9717, angle θ_3 is for {(19, 25) to (11, 6)} is 30.4428 to 315.3812.

The range of angle θ_1 is for {(25, 19) to (6, 11)} is 15.0130 to331.6376, angle θ_2 is 29.4354 to 290.6616, angle θ_3 is 47.3013 to 344.4666.

The range of angle θ_1 is for {(12, 25) to (25, 12)} is 35.4970 to 277.4746, angle θ_2 is 130.3875 to 333.7008, angle θ_3 is 5.0888 to 353.9905.

The range of angle θ_1 is for {(12, 12) to (25, 25)} is 16.9587 to 342.8606, angle θ_2 is 0.0311 to 289.1511, angle θ_3 is for 49.4957 to 358.5563

The range of angle θ_1 is for {(15, 28) to (25, 18)} is 3.8488 to 255.0529, angle θ_2 is for {(15, 28) to (25, 18)} is 11.3093 to 328.3415, angle θ_3 is 1.7201 to 340.8354.

Figure 5.8, Figure 5.9, Figure 5.10, Figure 5.11 and Figure 5.12 see the Graph between error and no. of step {(19, 25) to (11, 6)}, {(25, 19) to (6, 11)}, {(12, 25) to (25, 12)}, {(12, 12) to (25, 25)} and{(15, 28) to (25, 18)} respectively. In this we see the variation of error with respect to the no. of steps at the end error is zero. That means the end effector is reach to the destination.

The range of error is 0 to 1.6802, 0 to 1.7245, 0 to 1.6802, 0 to 1.7245 and 0 to 1.6802 respectively.

In Figure 5.13 see the graph Energy comparison between using GA & without GA. From the graph we see that energy less use in using GA.
In Figure 5.14, Distance comparison between using GA & without GA. From the graph we see that Distance less use in using GA.
From the Figure 5.15, see the relation between fitness function & generation.

Two points are defined in X-Y plane and point to point trajectory of 3 links robotic Arm is modeled in terms of angle, Energy and Distance. Result of various point to point trajectories shows that optimization of part to part path is possible using GA. Optimization is done by minimizing Energy consumed to move robotic arm, Distance travelled form one point to anther and error in final location of end effector, using 4th order polynomial. The optimized result is compared with manual movement of possible paths and the result by GA method is better than without GA.

Future Scope: The present work can extend to point location in XYZ coordinate system that is 3-D system. Some more constraint like workspace limitation, payload variation, time taken etc. can be include into function parameter for GA. In future present GA technique is implemented in Matlab or computer, and simulated for energy distance and Error in location. In future this can be integrated with an embedded system using AI- processors on board or a ”c” program for GA technique can be added.

REFERENCES

[1] Jorge Angeles, Angel Rojas, and Carlos S. Lopez-Cajun, “Trajectory Planning in Robotics Continuous-Path Applications” IEEE JOURNAL OF ROBOTICS AND AUTOMATION, VOL. 4, NO. 4,PP 360-365, 1988.

[2] A.B. Doyle, D.I Jones, “Robot Path Planning with Genetic Algorithms”, 2ND PORTUGUESE CONF. ON AUTOMATIC CONTROL, PP. 312-218, PORTO, PORTUGAL, 1996.

[3] Nearchou,A, Aspragathos.N “A Collision free continuous path control of manipulators using Genetic Algorithm” .JOURNAL OF SYSTEM ENGINEERING, VOL. 6,22-32. 1996.

[4] Mingwu Chen, Ali M. S. Zalzala,” A Genetic Approach to Motion Planning of Redundant Mobile Manipulator Systems Considering Safety and Configuration”, J. ROBOTIC SYSTEM. VOL. 14, N. 7, PP. 529-544, 1997.

[5] Zoltan Zoller, & Peter Zentan " Constant Kinetic Energy Robot Trajectory Planning ", PERIDICA POLYTECHNICA SER. MECH. ENG, Vol. 43, No.2, 213 – 228, 1999.

[6] E.J. Solteiro Pires & J.A. Tenreiro Machado, " A Ga Perspective Of The Energy Requirement For Manipulators Maneuvering In A Workspace With Obstacles", CEC 2000- CONGRESS ON EVOLUTIONARY COMPUTATION , SANDIAGO, CALIFORNIA, USA. PP. 1110- 1116, 2000.

[7] Devendra P. Grag, & Manish Kumar, "Optimization Techniques Applied To Multiple Manipulators for Path Planning and Torque Minimization", ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE, VOLUME 15, PP. 241-252(12) 2002.

[8] S.G. Yue, D. Henrich, X. L. Xu, & S.K. Toss, "Point-to-Point Trajectory Planning of Flexible Redundant Robot Manipulators Using Genetic Algorithms", JOURNAL OF ROBOTICA, VOL 20, PP 269-280, 2002.

[9] E.J. Solteiro Pires, Tenreiro Machado JA & De Moura Oliveira CR, "An Evolutionary Approach to Robot Structure and Trajectory Optimization", 10TH IEEE-INTERNATIONAL CONFERENCE ON ADVANCED ROBOTICS, ICAR'01, BUDAPEST, BULGARIA PP. 333-338, 22-25, 2002.

[10] S.G. Yue, D. Henrich, X. L. Xu, & S.K. Toss, "Point-to-Point Trajectory Planning of Flexible Redundant Robot Manipulators Using Genetic Algorithms", JOURNAL OF ROBOTICA, VOL 20, PP 269-280, 2002.

[11] Dong Kyoung Chwa, Jungo Kang, Kihong Im, Jin Young Choi, " Robot Arm Trajectory Planning Using Missile Guidance Algorithm,", SICE ANNUAL CONFERENCE, VOL. 2, PP 2056-2061, 2003.

[12] Zhe Tang, Changjiu Zhou, Zenqi Sun,"Trajectory Planning For Smooth Transition Of A Biped Robot," IEEE INTERNATIONAL CONFERENCE ON ROBOTIC & AUTOMATION, VOL. 2, 2455-2460, 2003.

[13] Tian.L.; Collins C “An effective robot trajectory planning method using Genetic Algorithm”. JOURNAL OF MECHATRONICS, VOL. 14, 445-470, (2003).

[14] Dongkyoung Chwa, Junho Kang, and Jin Young Choi, “Online Trajectory Planning of Robot Arms for Interception of Fast Maneuvering Object Under Torque and Velocity Constraints”, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS- PART A: SYSTEMS AND HUMANS, VOL. 35, NO. 6, PP 831-843, 2005.

[15] Kian Hsiang Low, Wee Kheng Leow, Member, IEEE, and Marcelo H. Ang, Jr., Member,

IEEE, “Autonomic Mobile Sensor Network With Self-Coordinated Task Allocation and Execution”, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART C: APPLICATIONS AND REVIEWS, VOL. 36, NO. 3, PP 315- 327, 2006.

[16] Emmanuel A. Merchán-Cruz and Alan S. Morris,“Fuzzy-GA-Based Trajectory Planner for Robot Manipulators Sharing a Common Workspace”, IEEE TRANSACTIONS ON ROBOTICS, VOL. 22, NO. 4, PP 613-624, 2006.

[17] Xiaobu Yuan and Simon X. Yang, “Multir obot-Based Nano assembly Planning with Automated Path Generation”, IEEE/ASME TRANSACTIONS ON MECHATRONICS, VOL. 12, NO. 3, PP 352- 356,2007.

[18] Foudil, “Trajectory Generation for Mobile Manipulators Using a Learning Method”, MEDITERRANEAN CONFERENCE ON CONTROL AND AUTOMATION, PP 1-6, 2007.

[19] Mike Peasgood, Christopher Michael Clark, and John McPhee, “A Complete and Scalable Strategy for Coordinating Multiple Robots Within Roadmaps”, IEEE TRANSACTIONS ON ROBOTICS, VOL. 24, NO. 2, PP 283-292, 2008.

[20] Niku, “Introduction to Robotics Analysis, Systems and Applications”. Saeed B. Niku Mechanical Engineering Department, California Polytechnic State University, San Luis Obispo. Second edition,2007.