

**Kinematic Isotropy and Robot Design Optimization using
A Genetic Algorithm Method**

By

Shabnam Khatami

B. Sc, University of Tehran, Iran, 1998

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF**

MASTER OF APPLIED SCIENCE

in

**THE FACULTY OF GRADUATE STUDIES
DEPARTMENT OF MECHANICAL ENGINEERING**

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

October 2001

© Shabnam Khatami, 2001

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

(Shabnam Khatami)

Department of Mechanical Engineering
The University of British Columbia
2324 Main Mall
Vancouver, BC, V6T 1Z4
Canada

Date: Oct 12, 2001

Abstract

The optimal kinematic design of robots is an interesting problem in contemporary robotics. It is important to have measures for determining the precision of the mechanism and size of the robot manipulator at the design phase. This all has been done before mostly on the basis of experience. The most essential issue for setting up any measures seems to be the ease of changing arbitrarily the position and orientation of the end-effector at the tip of the manipulator.

In the majority of recent works on optimal robot design, one of the most important criteria is that the robot can achieve isotropic configurations. The operation near isotropic configuration is considered as a high performance for robotic manipulators. At these configurations, the best servo accuracy can be achieved, the likelihood of error is equal in all directions, and equal forces may be exerted in all directions [29].

A measure of isotropy called the Global Isotropy Index or GII [49] has been used in this work, which is based on the robot behavior in the entire workspace. The GII is computed as the ratio of the minimum singular value of a robot's Jacobian matrix to the maximum one throughout its workspace. In search for finding the optimum design parameters that provide the most isotropic performance, the positions that offer the minimum ratio of singular values for each set of design parameters are compared to each other to find the maximum one. This strategy illustrates in fact a minimax optimization problem.

A "Genetic Algorithm" has been developed to optimize the minimax problem in order to find optimal design parameters such as link lengths of the best isotropic robot configurations at optimal working points of the end-effector and later, it has been implemented to optimize globally throughout the whole robot workspace. The method has been demonstrated for two types of robotic manipulators.

Table of Contents

Abstract	ii
Table of Contents.....	iii
List of Tables	vi
List of Figures	vii
Acknowledgements.....	viii
1 Introduction	1
2 Background	6
2.1 Condition Numbers.....	6
2.2 Global Isotropy Measures.....	11
2.3 Optimization algorithms.....	12
3 Minimax Problem.....	14
3.1 Singularities.....	14
3.3.1 Effect of Singularities.....	15
3.2 Singular Value Decomposition.....	16
3.2.1 Mathematical Background.....	16
3.3 The Global Isotropy Index.....	21
3.4 The Minimax Robot Design Optimization.....	23
4 Genetic Algorithms.....	25
4.1 Evolutionary Algorithms.....	25
4.2 Genetic Algorithms.....	26
4.2.1 GA Advantages.....	27
4.2.2 GA Coding.....	28
4.3 GA Major Operators.....	29
4.4 Reproduction.....	29
4.4.1 Roulette Wheel Selection.....	30
4.4.2 Proportional Fitness Assignment.....	30
4.4.3 Rank Based Fitness Assignment.....	31
4.4.4 Stochastic Universal Sampling.....	33
4.4.5 Truncation Selection.....	33
4.4.6 Tournament Selection.....	33

4.4.7	Local Selection.....	33
4.4.8	Comparison of Selection Schemes.....	34
4.5	Recombination.....	35
4.5.1	Crossover.....	35
4.5.2	Discrete Recombination.....	37
4.5.3	Intermediate Recombination.....	38
4.5.4	Line and Extended Line Recombination.....	38
4.6	Mutation.....	39
4.6.1	Binary Mutation.....	39
4.6.2	Real Valued Mutation.....	40
4.7	Other GA Operators.....	40
4.7.1	Reinsertion.....	40
4.7.2	Population Models.....	41
4.8	GA Termination.....	43
5	Minimax Genetic Algorithms.....	44
5.1	Genetic and Evolutionary Algorithm Toolbox in MATLAB.....	44
5.1.1	Features of the GEATbx.....	45
4.9	Minimax Genetic Algorithm	47
6	Implementation of the minimax GA for Robotic Manipulators...53	
6.1	Design of a Locally Isotropic 2-DOF Planar Manipulator.....	53
6.2	Design of a Globally Isotropic 2-DOF Planar Manipulator.....	59
6.3	Design of a 3-DOF Planar Manipulator.....	63
7	Conclusions.....	67
7.1	Overview of the research.....	67
7.2	Future Work.....	68
	Bibliography.....	69
	Appendix 1.....	74
	Appendix 2.....	77

Appendix 3.....	79
Appendix 4.....	81

List of Tables

Table 4.1:	Binary Representation of Individuals in a GA.....	28
Table 4.2:	Single Crossover.....	35
Table 4.3:	Multi-point Crossover.....	36
Table 4.4:	Uniform Crossover.....	37
Table 4.5:	Discrete Recombination.....	38
Table 4.6:	Binary Mutation.....	40
Table 6.1:	The Results of the Last Generation of the GA for Local Isotropic Optimization of an Elbow Manipulator.....	57
Table 6.2:	The Results of the Last Generation of the GA for Global Isotropic Optimization of an Elbow Manipulator.....	61
Table 6.3:	The Results of the Last Generation of the GA for Local Isotropic Optimization of a 3-DOF Manipulator.....	64

List of Figures

Figure 3.1:	Constrained Planar Elbow Manipulator.....	20
Figure 3.2:	Torque Ellipse at $x=5$	20
Figure 3.3:	Force/Torque Transformation.....	22
Figure 3.4:	Force/Torque Ellipse and GII.....	22
Figure 4.1:	Roulette Wheel Selection.....	30
Figure 4.2:	Proportional and Rank Based Fitness Assignment.....	32
Figure 6.1:	A 2-DOF Planar Elbow Manipulator.....	54
Figure 6.2:	Surface and Contour Plots of Variation of the Local Isotropic Measures for a 2-DOF Planar Manipulator.....	55
Figure 6.3	Color Classification.....	56
Figure 6.4	The Results of the GA for Local Isotropic Optimization of a 2-DOF Elbow Manipulator.....	58
Figure 6.5:	Surface and Contour Plots of Variation of the Global Isotropic Measures for a 2-DOF Planar manipulator with $l_1=5.5$	60
Figure 6.6	The Results of the GA for Global Isotropic Optimization of a 2-DOF Elbow Manipulator.....	62
Figure 6.7:	A 3-DOF Planar Manipulator.....	63
Figure 6.8:	The Results of the GA for a 3-DOF Manipulator.....	65
Figure A.1:	A 3-DOF RRR Serial Manipulator.....	74
Figure A.2:	A 2-DOF Robot.....	77
Figure A.3:	A 3-DOF Robot.....	79
Figure A.4:	A 2-DOF Planar Manipulator.....	81

Acknowledgements

I wish to thank my supervisor Dr. Farrokh Sassani for providing me with the opportunity to carry out this research. I gratefully acknowledge the National Center of Excellence-Institute for Robotics and Intelligent Systems (NCE-IRIS) for funding this work.

I would like to thank the people in the Computer Aided Manufacturing and Robotics Laboratory for providing me with a friendly atmosphere to work in. Thanks are also extended to Daniel Wong for his assistant with creating the figures of this work.

I would also like to express my deepest love and gratitude to my family for their continuous encouragement and support.

Chapter 1

Introduction

In modern robotics, the need for intrinsically accurate kinematic structures of robotic manipulators has become a very important issue. With the increasing application of precise manipulators such as mechanical fingers and articulated hands especially in vital real-life functions such as surgery, the need for mechanical design optimization of robotic manipulators is quite apparent. For these applications, accurate manipulation is highly demanded. Therefore, a high degree of dexterity has turned out to be one of the most important characteristics of robotic manipulators.

A number of different descriptions of the concept of dexterity for robotic manipulators have led to many different performance indices for the quantification. Dexterity has been interpreted to mean different physical concepts such as the kinematic extent to which a manipulator can reach all the orientations, which is in fact the description of the dexterous workspace [34]. It has also been interpreted as a specification of the dynamic response of a manipulator [55], and joint range availability [35].

Local kinematic measures of dexterity are mostly based on the Jacobian matrix of the manipulator. The basic idea of a local kinematic isotropy is that it may be desirable to design or configure a robot so that manipulation can be performed very accurately at a work point [35]. Various mathematical measures have been proposed for quantification of local kinematic accuracy. They include the determinant of robot's Jacobian matrix, manipulability, minimum singular values, and condition number which is defined as the ratio of the minimum singular value of manipulator's Jacobian matrix to the maximum one.

When the condition number is at its optimal value of one, the resulting robot has been described as being perfectly isotropic. Isotropic configuration has a number of advantages including good servo accuracy and singularity avoidance [35].

The term singularity is used when referring to infinite condition number of either square singular matrices or rectangular rank deficient matrices, and hence with regard to matrix invariability, isotropy and singularity are in fact at the opposite sides of the scale. Because an infinite condition number characterizes a singularity, isotropy can be considered as being the furthest possible distance from singularities.

The “Global Isotropy Index” or GII [49] is one of the proposed isotropy measures, which is based on the robot behavior in the entire workspace. It is defined as the ratio of the minimum singular value of robot’s Jacobian matrix to the maximum one obtained throughout the whole workspace not just at a single point. Since the GII is the ratio of the minimum over the maximum singular value, it allocates a value of 1 to perfect isotropy and a value of 0 instead of ∞ to a singularity. Furthermore, as a workspace inclusive measure, it takes into account both positional and directional isotropy.

The application of the GII as an isotropy measure leads to a minimax optimization problem. In the local optimization algorithm, the workspace positions are searched to find the positions that provide the minimum ratio of singular values for each set of design parameters. Then all those minimum values are compared to each other and the maximum of them is considered as the best GII. The position at which the best GII is obtained is considered the optimal working point in local isotropy measure and the associated set of design parameters providing that best performance at that optimal point is the optimal design set. This leads to the concept of local isotropy optimization of robotic manipulators.

The GII is also employed in this work to solve the problem for global isotropy, thus designing a manipulator that performs optimally over the entire workspace. In this case, for each set of particular design parameters, the minimum ratio of the minimum singular value of Jacobian to the maximum one is derived by comparing minimum and maximum singular values in the entire workspace and not just at a single position.

A minimax optimization algorithm called the “Culling Algorithm” is applied in [49] to solve for the GII optimization problem. It is designed to specifically optimize the minimax problems by descritizing the search space, identifying non-optimal parameters, and culling them from the search space until the optimal parameter remains. However, since all the existing points of the search space are examined in each iteration, the algorithm becomes significantly demanding as the size of the workspace and design parameter space increase. In addition, because of its discrete nature, the culling algorithm cannot be actually applied when a very high resolution is needed.

Most of the other optimization algorithms especially the traditional ones are limited to continuous and differentiable systems based on deriving some auxiliary functions. For example in gradient methods, the derivative information of the function should be defined. Such methods usually perform well on functions with only one optima (unimodal functions) and in case of functions with many peaks (multimodal functions), they occasionally become trapped in local optima.

Like most of the real world problems, the robot design optimization problem is a multi-modal, non-differentiable, and highly non-linear problem that cannot be solved by traditional optimization methods. Therefore, there is a need for an efficient optimization technique that can deal with complex functions such as the GII. One of the most powerful optimization algorithms that can solve for such functions is a “Genetic Algorithm”.

Genetic algorithms (GAs) are global search methods that are based on the procedure of natural evolution to guide their exploration in the search space. They require little knowledge of the problem itself and do not require the search space to be necessarily discrete.

A genetic algorithm models Darwin’s theory by simulating the evolution of a population of possible solutions to a particular problem. The variables of a problem are represented as coded strings, which are called chromosomes. By breeding these chromosomes, a population of new solutions is created. These new individuals are closer to the optimal result of the problem than the ones in previous generation.

Genetic algorithms are the selected optimization methods in this work. To solve for a minimax problem, two genetic algorithms run simultaneously. The program starts with creating a population of individual candidates for the first GA by randomly selecting points from the design parameter space. The first member of the population enters the second GA as the input value of design parameter and a population is initialized for the second GA by randomly selecting position points through robot workspace. The second GA runs until the optimal working point is found that gives the minimum ratio of singular values for that particular input design parameter. The fitness score of that best position is now assigned to the design parameter and the program returns to the first GA. The same procedure is repeated for the rest of the population until the maximization process of the first GA is ended and the best design parameter vector that provides the best isotropic configuration of the robot at a workspace position is found.

In the algorithm for global isotropy, the singular values for all workspace positions are compared during the second GA to find the minimum ratio of the minimum to maximum singular values of the entire workspace.

The applied tool for demonstrating the two GAs of our approach is the "Genetic and Evolutionary Toolbox in MATLAB" [44], which includes a variety of genetic algorithm features as explained in chapter 5.

This thesis is organized as follows. Chapter 2 presents several local and global condition indices that have been proposed before as well as different optimization algorithms to solve for kinematic design optimization of robotic manipulators. In chapter 3, the robot singularities, the "singular value decomposition" and the concept of manipulability ellipsoid are described and the global isotropy index [49] has been explained as the applied isotropic measure in this work. Chapter 4 includes an introduction to genetic algorithms and their various operators. In chapter 5, the "Genetic and Evolutionary Toolbox in MATLAB" is introduced and an algorithm is proposed to solve for the kinematic design of a robot as a minimax optimization problem. In chapter 6, the proposed algorithm is applied to design two types of robotic manipulators and some results are compared to the ones after applying a culling algorithm in [49]. In the end, the work is concluded in chapter 7.

Appendix 1 contains the equations of inverse kinematics for a 3-DOF RRR serial manipulator. Appendices 2 and 3 present the Jacobian matrices of a 2-DOF and a 3-DOF planar manipulator and in appendix 4, singular values are derived mathematically for a 2-DOF and a 3-DOF planar manipulator.

Chapter 2

Background

This chapter describes past proposals on local and global condition indices as a measure of robotic isotropy and several optimization methods that have been used to find the best isotropic robot configuration. In section 2.1, past proposals and results related to local and global condition numbers have been discussed. In section 2.2 some recent global isotropy measures have been stated and in section 2.3 different optimization algorithms are compared for solving a minimax problem such as a culling algorithm.

2.1 Condition Numbers

Condition numbers are used as a tool to measure the degree of dexterity in a robot manipulator. In numerical analysis, the condition number is interpreted as an indicator of the sensitivity of the solution of a linear system. If we consider the system $Ax=y$, then the following relation exists

$$\frac{\|\delta x\|}{\|x\|} \leq \text{con}(A) \frac{\|\delta y\|}{\|y\|} \quad (2.1)$$

δx is the uncertainty in finding x resulting from an uncertainty δy in y , and $\text{con}(A)$ stands for condition number of A [35].

We recall that the Jacobian matrix of a serial-type manipulator is defined as the transformation mapping the joint rates into Cartesian velocities at an operation point x of the end-effector. This transformation is shown in expression (2.2) and used when finding the joint velocities given end-effector velocities.

$$v = J(x)\dot{q} \quad (2.2)$$

The transpose of the Jacobian matrix can be interpreted as a linear transformation that maps the forces applied at a workspace point x into the torque vector. This transformation is shown in expression (2.3) and used when finding end-effector forces given joint torques

$$\tau = J^T(x)f \quad (2.3)$$

Thus when the condition number of Jacobian has its optimal value of one, minimum sensitivity in velocity exists. It means that the variation in relationship between the robot's actuators' motion and its end-effector velocity is at a minimum as the position or/and orientation of the end-effector is/are changed. Therefore, the manipulator can control equally well in all directions at that particular workspace point. The same strategy is applied for relationship (2.3) when the optimal condition number of J^T is interpreted as the minimum variation in relationship between the end-effector forces and actuator torques.

Typically, the condition number of the Jacobian matrix is considered as a measure of the kinematic accuracy of the manipulator. As the condition number of Jacobian matrix gets closer to unity, the resulting robot becomes more kinematically isotropic.

There are some particular points in the robot workspace where the determinant of square Jacobian matrices becomes zero or the rank of rectangular matrices defects. In this case the manipulator is expressed to be at singular configuration. When manipulator configuration gets near singular positions, very large joint rates are required to create small end-effector displacements. At a singularity, the required joint actions become unbounded and the manipulator loses at least one degree of freedom. This means the manipulator cannot move or exert forces (or torques) along (or around) some directions.

The condition numbers are derived from Jacobian matrix as explicit functions of joint coordinates and link lengths. With the manipulator Jacobian being a function of its joint variables, the manipulator condition number is configuration dependant. Within the concept of kinematic dexterity, several measures have been proposed as Jacobian

condition number of a manipulator such as determinant, manipulability and minimum singular values.

One of the first measures for determining of the mechanism was proposed by Yoshikawa [54]. He proposed a quantitative measure of manipulability of robotic mechanisms in positioning and orienting end-effectors. This measure of manipulability is given in equation (2.4) as a scalar value of w in terms of Jacobian matrix J at state θ .

$$w = \sqrt{\det(J(\theta)J^T(\theta))} \quad (2.4)$$

In Yoshikawa [55], this index for various robotic mechanisms has been applied to find the best posture that gives the greatest manipulability measure at a given position of end-effector in the workspace. However, the calculation of the matrix product JJ^T squares the condition number, which reduces the accuracy of the result. Furthermore, this measure represents an average mobility over all direction at the end-effector and has an analytical expression, but it depends on the scale of a manipulator [33].

The manipulability index can be applied for both redundant and nonredundant manipulators. When nonredundant manipulators are considered, the manipulability measure shrinks to

$$w = |\det J(\theta)| \quad (2.5)$$

Although a determinant going to zero marks the presence of a singularity, the value of the determinant cannot be considered as a useful measure of the isotropy. The determinant of a square matrix, or that of the product of a rectangular matrix by its transpose –under the assumption that the matrix has more columns than rows- does not measure how far from singularity a matrix is. In fact the determinant of a square matrix tells only when a matrix is invertible, but it does not quantify the invariability of the matrix.

Instead, the matrix condition number has been recommended by numerical analysts because of its quantitative accuracy estimates. The condition number indicates the uniformity of the Jacobian transformation [34]. It measures the round-off error

amplification on solving a system of linear algebraic equations associated with that matrix, whereas the determinant never comes into the picture when assigning the amount of that error in equation solving [1].

In Klein and Blaho [34], several measures of dexterity have been examined and compared to find an optimal configuration of a three-link planar manipulator for a given end-effector position for two-dimensional positioning tasks. Also, an optimal work point in which the manipulator has the greatest dexterity has been derived. One of the applied dexterity measures is the minimum singular value of Jacobian.

Defining the condition number as the ratio of the largest singular values of Jacobian to the smallest one, Klein and Blaho briefly discussed manipulator design problem with an isotropic configuration at a fixed working point for a fixed total arm length. However, this condition number has not been expressed analytically. It has been used with regard to the fact that for a nonredundant system to be isotropic, the rows or alternatively the columns of the Jacobian must have the same vector magnitude and be orthogonal. The above condition on the columns can be interpreted as the end-effector being an equal distance from the joint axes, and the incremental rectilinear motions of the end-effector caused by each joint must be orthogonal to the motion caused by the other joints.

Kim and Khosla [32] indicate two problems of Yoshikawa's manipulability measure when it is used for a design problem, Scale dependency and order dependency. These two problems create difficulties for design when needed to compare manipulators with different sizes. The scale dependency prevents a fair comparison between a longer manipulator and a shorter one and the order dependency makes it impossible to derive the physical meaning of the manipulability. Thus they overcome this shortcoming by defining a new measure of dexterity termed the measure of isotropy, which is the ratio of the geometric mean and the arithmetic mean of the eigenvalues of JJ^T [32].

$$\Delta = \frac{W}{\Psi} \quad , \quad \Psi = \frac{\text{trace}(JJ^T)}{\text{order } m} \quad (2.6)$$

Δ stands for the measure of isotropy and W is the manipulability measure proposed by Yoshikawa. This measure is nondimensional and thus independent of the scale of a manipulator.

In their late work [33], Kim and Khosla employ their measure as an optimization criterion for task based design of a 3-DOF manipulator. The optimization method used is a multi-population genetic algorithm and has the same number of optimization function as task points.

In Gosselin [22], dexterity indices for planar and spatial manipulators are presented and the approach used is to describe the velocity of the end-effector as the velocity of some points on it instead of using the velocity of only one point together with the angular velocity. He proposes to redefine the manipulator Jacobian as that matrix mapping joint rates into the velocities of a set of points on the end-effector, thereby eliminating the end-effector angular velocity and hence producing a dimensionally homogeneous Jacobian.

In Klein and Miklos [35], a technique called “Spatial Isotropy” is demonstrated that uses both positional and orientational isotropy. They extend the concept of isotropy to include simultaneous positioning and orienting of a work piece in three-dimensional space which is termed as spatial isotropy”. They believe that if the Jacobian is considered as a measure of whole isotropy just in one term, it would be a relatively weak and artificial condition.

In Ma and Angeles [39], the dynamic performance of manipulator is considered when optimizing to find the best robot architecture. The “Dynamic Conditioning Index” (DCI) is defined which measures the dynamical coupling and numerical stability of the generalized inertia matrix of a manipulator in its dynamic model. They applied DCI in design of serial and parallel manipulators. The results are well suited for a particular position, however, satisfactory performance is not guaranteed as the robot moves to other points. Therefore, it is not appropriate for optimal global performance when the manipulator contains a large workspace dimension.

In Angeles [1], the manipulator conditioning index (CI) is defined. It is defined in terms of the minimum condition number on proper choice of the joint variables. It is regarded

as a local property, for it does not take into account the distribution of the manipulator condition number over the whole workspace. He proposes an alternative approach to Gosselin's work that consists writing the point-velocity manipulator equations in nondimensional form by suitably dividing both sides of the velocity equation by a natural length of the manipulator.

As the ratio of minimum singular value of Jacobian to the maximum one, the condition number has been used as a measure of dexterity in many previous works for several reasons. First, when robot is at its isotropic configuration, the optimal condition number has a fixed value of 1 and that happens when all the singular value of Jacobian are the same at that particular working point. Secondly, a simple numerical procedure can be employed to find the best robot posture that moves the condition number towards unity for given mechanism and end-effector at a given working point. However, all the condition indices mentioned so far do depend on the operating point and the location of the point is critical in obtaining a good kinematic manipulation performance. Hence, they fail to give a reliable measure of the global performance of the manipulator throughout the whole workspace.

Even though in most occasions accurate end-effector hand motion at a workpiece is much more important than the accuracy along the trajectory leading there, global kinematic criteria should be considered to ensure the accuracy of manipulator performance over the whole workspace. Thus, several other measures have been proposed to overcome this problem as mentioned in section 2.2.

2.2 Global Isotropy Measures

Most of the global measures that have been proposed so far are derived as an overall of the calculated local measures. They generalize the local isotropy at all positions to propose a global measure such as the best average performance value of all local measures. These measures are not reliable since the best average performance might contain some poor performances at some points that result in very weak behaviors as the robot reaches those points in practice.

In Gossolin and Angeles [23], The “Global Conditioning Index” (GCI) is proposed as the integral of the condition number k of Jacobian matrix over the workspace scaled by the size of the workspace as shown in equation (2.7).

$$GCI = \frac{\int_W \frac{dW}{k}}{\int_W dW} \quad (2.7)$$

This global measure encounters the same deficiency as the average value. Some poor behaviors might be ignored at some intermediate workspace positions. Besides, computing the integral is quite demanding as the condition number is a highly non-linear function.

Stocco [49] introduces a measure of global isotropy called the “Global Isotropy Index” (GII). GII is used in his work to optimize the design of a number of manipulators such as a haptic pen. GII is the ratio of the smallest singular value of Jacobian matrix to the largest singular value of the Jacobian matrix in the entire workspace as shown in expression (2.8)

$$GII(p) = \min_{x_0, x_1 \in W} \frac{\sigma_{\min}(J(p, x_0))}{\sigma_{\max}(J(p, x_1))} \quad (2.8)$$

GII is the selected measure of isotropic performance in this work. More about the concept of the GII is mentioned in the next chapter.

2.3 Optimization Algorithms

The optimization of the GII introduced in previous section leads to a minimax optimization problem.

In Kim and Khosla [33], the “Multi-population Genetic Algorithm” is introduced to overcome the huge increase in size of the search space as the number of variables of a genetic Algorithm is increased. This algorithm is used to optimize the task based design of a 3-DOF manipulator. It has the same number of optimization function as task points.

Each objective function represents a manipulator (which are all just one) and all these objective functions have been connected by “connecting constraints”. It is actually an assumption of designing m manipulators for m task points. In this parallel implementation of GAs, the number of variables for each optimization function is fixed. The total fitness of an individual is obtained from its objective function and the adjusting value from connecting constraints by comparison with individuals in other populations.

In Buckley [7], genetic algorithms have been applied to solve for inverse kinematics of a kinematically redundant manipulator. The final results were awkward due to lack of diversity of the population. To overcome this problem, they later improved the results by enhancing the GA with heuristic based knowledge of manipulator kinematics. However, the final results were still not accurate.

In [48], “Culling Algorithm” is proposed as an approach to optimize the design of robotic manipulators. This algorithm is defined to solve a minmax optimization of the global isotropy index as a measure of isotropic performance of manipulators. This method is described in more details in the following chapter.

Chapter 3

Minimax Problems

This chapter discusses the robot singular configurations and minimax optimization problem in more details. In section 3.1, the nature of singularities and their effect on robot performance are explained. Section 3.2 goes through the analytical calculation of singular values and leads to the concept of the manipulability ellipsoid, which gives more insight into the concept of condition number as the ratio of singular values of Jacobian matrix. In section 3.3, the global isotropy index is described based on the concept of the manipulability ellipsoid, and finally in section 3.4 the minimax robot design problem is introduced.

3.1 Singularities

The Jacobian matrix operates as a bridge that relates variations in joint velocities to the corresponding variations in Cartesian velocities. If the matrix is invertible for all values of joint angles, then knowing Cartesian velocities, joint velocities can be determined. The Jacobian is not invertible when its determinant is equal to zero. In this case, variations in Cartesian velocities cannot be related to joint velocities. That's when the Jacobian matrix is termed to be singular. Many manipulators have configurations where the Jacobian becomes singular. In singularities, there might not be a solution to the inverse kinematics problem or there might be an infinite number of solutions. When a manipulator is in a singular configuration, it has lost rank. In other words, it has lost one or more degrees of freedom in Cartesian space.

It is common to classify singularities in two groups: workspace boundary singularities and workspace interior singularities. Workspace boundary singularities occur when the end-effector is at or near the boundary of the manipulator's workspace. In this state usually the manipulator is fully stretched out or folded back on itself. Workspace interior

singularities occur at configurations inside the workspace usually when two or more joint axes line up.

3.1.1 Effects of Singularities

Singularities cause many problems in robot performance, both for nonredundant and for redundant robots. Although redundant robots may be able to avoid singularities inside the workspace boundaries by the use of their redundant degrees of freedom, nonredundant robots have to find methods to pass through the singular configurations unless the Cartesian trajectory is modified. However, singularities that lie at the workspace boundaries are not avoidable even by redundant manipulators.

At singularities the Jacobian is not invertible for a six degree of freedom manipulator and thus regardless of the joint rates, the manipulator is unable to move in certain directions. Sometimes a manipulator may exhibit a couple of different singularities at a particular configuration causing the manipulator to be unable to move in different directions. For manipulators with less than six degrees of freedom, a singularity causes the manipulator to lose one or more of its degrees of freedom. Whenever a manipulator loses a degree of freedom, there are directions along or around which it cannot move or exert forces.

At manipulator configurations near singular positions, very large joint motions are required to produce relatively small end-effector displacements. In velocity domain, near a singularity the joint velocities that are required to maintain a desired end-effector velocity in certain directions can be extremely large. When a singularity is encountered, the required joint velocities become boundless and the manipulator loses at least one degree of freedom. This implies that excessive output demands will be placed on the joint actuators in the neighborhood of a singularity. As one or more actuators saturate, the end-effector will deviate from the prescribed trajectory. Furthermore, when a desired end-effector placement corresponds to a singular position, there may be an infinite number of possible joint configurations, a situation that complicates task planning [53].

The singularities can affect the manipulator in the force domain as well as in the velocity domain. They can have an effect on the size of forces that the manipulator can apply at the end-effector. In the force domain, the Jacobian transpose relates the Cartesian forces

applying at the end-effector into equivalent joint torques. When the Jacobian transpose loses one or more degrees of freedom, there are some directions in which the static forces cannot be applied by the end-effector.

3.2 Singular Value Decomposition

Singular value decomposition (SVD) of the Jacobian Matrix is employed to represent the “character” of the transformation from joint space to Cartesian space expressed by the Jacobian and used in solving the problems of robot operation such as inverse velocity near singular configurations. It has been extensively used as a tool for the analysis of the kinematic and dynamic characteristics of robotic manipulators. Most of the proposed dexterity measures are some functions of the singular values of the Jacobian matrix and closely linked to the singular values and vectors of Jacobian. For example, it will be shown that the manipulability measure proposed by Yoshikawa in 1984 is basically the product of the singular values of Jacobian. Some other measures in this category include the trace of the Jacobian matrix and the global isotropy index (GII).

In general, SVD has been known as being numerically expensive to calculate. This matter resulted in the popularity of the manipulability measure in the past, since it is numerically simple to compute and that its zeros coincide with the singularities of the Jacobian. However, as indicated in the previous chapter, it is not a suitable measure of robot performance.

The most popular technique for computing the SVD is commonly referred to as the Golub-Reinsch Algorithms (Golub and Reinsch 1970). It is available in many linear algebra software tools such as MATLAB. This algorithm is generally regarded as one of the most efficient and numerically established technique for computing the singular values of an arbitrary matrix.

3.2.1 Mathematical Background

For a manipulator with n degrees of freedom whose joint variables are defined by q_i , $i=1,2,\dots,n$, a class of tasks are described by m variables x_j , $j=1,2,\dots,m$. The relation between q_i and x_j is given by

$$X=f(Q) \quad (3.1)$$

Where $Q=[q_1, q_2, \dots, q_n]^T$ is the joint vector, $X=[x_1, x_2, \dots, x_n]^T$ is the manipulation vector, and the superscript T denotes the transpose. The aim is to solve the linear system of equations (3.2) which is the relation between the manipulation velocity \dot{x} and the joint velocity \dot{q}

$$\dot{X} = J(q)\dot{q} \quad \dot{q} \in \dot{Q} \subset R^n \quad \dot{x} \in \dot{X} \subset R^m \quad (3.2)$$

Where $J(q) \in R^{m \times n}$ is the manipulator Jacobian matrix, $m=n$ for non redundant robots and $m < n$ for redundant robots where degree of redundancy is $(n-m)$.

The Jacobian matrix J is non-singular if the relation (3.3) exists.

$$\text{Max}_{q} \text{rank } J(q) = m \quad (3.3)$$

If for some q^* ,

$$\text{rank } J(q^*) < m \quad (3.4)$$

then the manipulator is in a singular state. The state q^* is not desirable because the manipulation vector X can not move in a certain direction, meaning that the manipulability is seriously deteriorated. However, it is not only the singular state which is undesirable. A neighboring region of any singular state is also undesirable because in that region, the manipulation vector X can only move very slowly in a certain direction. In other words, the manipulability of the robot in this region is very limited.

It is known from linear algebra that if J is a real $m \times n$ matrix, then there exist orthogonal matrices $U = [u_1, \dots, u_m] \in R^{m \times m}$, $V = [v_1, \dots, v_n] \in R^{n \times n}$ such that $J = U \Sigma V^T$

where

$$\Sigma = [\text{diag}(\sigma_1, \dots, \sigma_m) 0] \in R^{m \times m} \quad (3.5)$$

with

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_m \geq 0 \quad (3.6)$$

The σ_i are the singular values of J , and the vectors u_i and v_i are the i^{th} left singular vector and the i^{th} right singular vector of J , respectively. This decomposition is called the singular value decomposition.

Another important theorem from linear algebra, the ‘‘Symmetric Real Schur Decomposition Theorem’’ [31], says that if A is a real $n \times n$ symmetric matrix, then there exists a real orthogonal $Q \in R^{n \times n}$ such that

$$Q^T A Q = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n) \quad (3.7)$$

Where λ_i are the eigenvalues of A . The columns of matrix Q are the eigenvectors of A . There are important relationships between the singular value decomposition of a matrix J and the Real Schur decomposition of symmetric matrices $J^T J$ and $J J^T$. Indeed, if $U^T J V = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_m)$

$$(3.8)$$

is the SVD of $J \in R^{m \times n}$, ($m \leq n$), then

$$V^T (J^T J) V = \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_m^2, 0, \dots, 0) \in R^{n \times n} \quad (3.9)$$

and

$$U^T (J J^T) U = \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_m^2) \in R^{m \times m} \quad (3.10)$$

Therefore, the eigenvectors of $J^T J$ are the right singular vectors of J (matrix V), whereas the eigenvectors of $J J^T$ are the left singular vectors of J (matrix U). Moreover, the eigenvalues of $J J^T$ are equal to the squared singular values of J .

It has been proved in [31] that the singular values of a Jacobian matrix do not depend on the coordinate frames in which the Jacobian is expressed. This allows us to choose the best coordinate frame that simplifies the Jacobian matrix as much as possible. The derivation of Jacobian matrix and SVD for a 2-DOF and a 3-DOF planar manipulator are shown in Appendices 2, 3, and 4.

If we map a hypersphere of joint velocities \dot{q} in the space R^n into the Cartesian space R^m such that

$$\|\dot{q}\|^2 = \dot{q}_1^2 + \dot{q}_2^2 + \dots + \dot{q}_n^2 \leq 1 \quad (3.11)$$

then, the result is an ellipsoid with principal axis $\sigma_1 u_1, \sigma_2 u_2, \dots, \sigma_m u_m$ where $u_i \in R^m$ is the i^{th} column vector of U [20]. This ellipsoid is called the manipulability ellipsoid and was one of the first measures for the analysis, design and control of robot manipulators. Therefore, the “Principal Kinematic Axes” (PKAs) $\{\sigma_1 u_1, \sigma_2 u_2, \dots, \sigma_m u_m\}$ may be identified by examining the transformation from joint space to Cartesian space expressed by Jacobian J . The singular value decomposition is employed to represent the character of the transformation [20].

The manipulability index is proportional to the ellipsoid volume spanned by the PKAs. This volume is approximated in practice by the product of the singular values. Since the volume of an ellipsoid increases as it becomes more spherical, there is a correlation between high manipulability indices and isotropic conditioning [20]. The volume of the manipulability ellipsoid is given by:

$$d \sigma_1 \cdot \sigma_2 \dots \sigma_m \quad (3.12)$$

where the constant d in equation (3.12) is given by

$$d = \begin{cases} (2\pi)^{m/2} / (2 \cdot 4 \cdot 6 \dots (m-2) \cdot m) & \text{when } m \text{ is even} \\ 2(2\pi)^{(m-1)/2} / (1 \cdot 3 \cdot 5 \dots (m-2) \cdot m) & \text{when } m \text{ is odd} \end{cases} \quad (3.13)$$

Therefore, manipulability measure w in (2.4) is equal to the volume of the kinematic manipulability ellipsoid except for the constant coefficient d . It is shown in (3.14) in terms of singular values of Jacobian matrix J .

$$w = \sigma_1 \cdot \sigma_2 \dots \sigma_m = \sqrt{\lambda_1 \lambda_2 \dots \lambda_m} \quad (3.13)$$

Where $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_m \geq 0$ are eigenvalues of JJ^T and $\sigma_i = \sqrt{\lambda_i}$ are singular values of Jacobian matrix.

[20] points out that the singular vectors for the force relationship $\tau = J^T f$ are the same as those describing the velocity domain, but that the singular values are reciprocal. This suggests that the direction for favorable velocity amplification is orthogonal to the direction for favorable force amplification. Moreover, since accuracy is inversely

proportional to amplification, a reciprocal relationship also exists between accuracy and amplification in both the velocity and force domains.

In Figure (3.2), reproduced from [49], the resulted manipulability ellipse is shown from plotting actuator torques that produce an end-effector force of unit magnitude and arbitrary direction for the manipulator of Figure (3.1) [49].

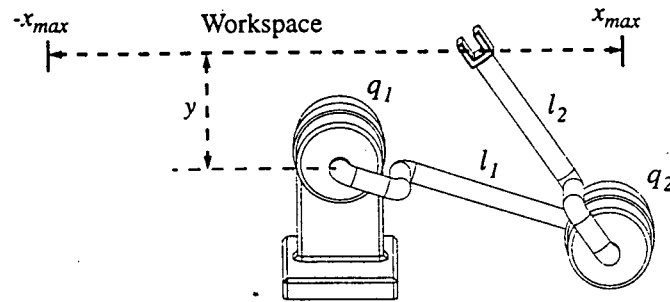


Figure 3.1: Constrained Planar Elbow Manipulator

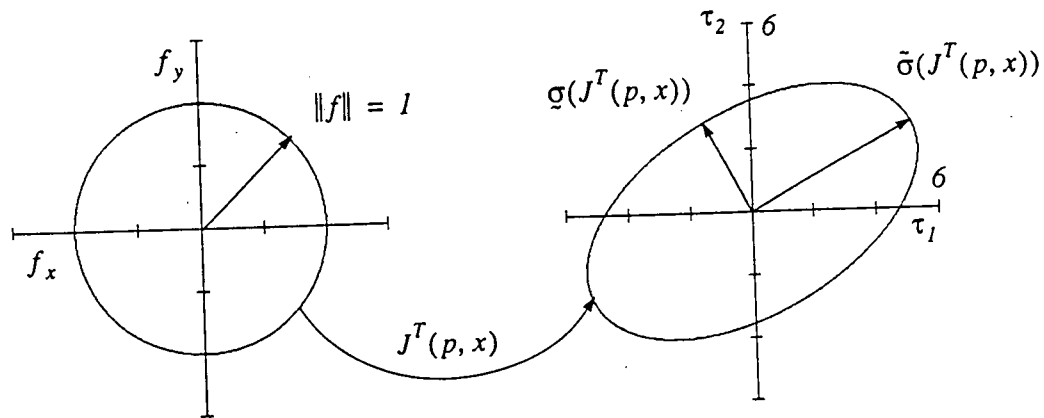


Figure 3.2: Torque Ellipse at $x=5$

The lengths of the major and minor axis of the ellipse shown in Figure (3.2) indicate the highest and lowest effective transmission ratios occurring in all directions which are related to maximum $\sigma_{max}(J(l,x))$ and minimum $\sigma_{min}(J(l,x))$ singular values of the of $J^T(x)$. The Jacobian is a function of both position x and geometry p . For minimizing the variation in relationship between robot actuators and its end-effector, these transmission ratios should be as close to each other as possible. Perfect isotropy exists when those ratios are equal. It means that the singular values of Jacobian are the same and the ellipse shifts to a circle. Therefore, the manipulator is isotropic if all its singular values are identical and nonzero. This is equivalent to saying that, if J is isotropic, then the product JJ^T is proportional to the identity matrix, i.e. a positive scalar α exists in this case for which

$$JJ^T = \alpha I \quad (3.14)$$

3.3 Global Isotropy index (GII)

The global isotropy index is proposed by [49], which is computed from the singular values of a design matrix in the entire workspace. GII gives an insight of global performance of a robot throughout the workspace by a scalar value.

As mentioned before, most condition numbers defined in the past are calculated from a design matrix such as Jacobian evaluated at a single position of the workspace as shown in (3.15).

$$K(l,x) = \sigma_{max}(J(l,x)) / \sigma_{min}(J(l,x)) \quad (3.15)$$

Because the Jacobian is computed at a position x , the condition number is a local measure of isotropy and the manipulator is isotropic at that individual position. However, the manipulator may not show similar levels of isotropy as it moves from that point to other points in the workspace.

In [49], torque ellipses are computed at three different positions for the manipulator of Figure (3.1) and the results are shown in Figure (3.3).

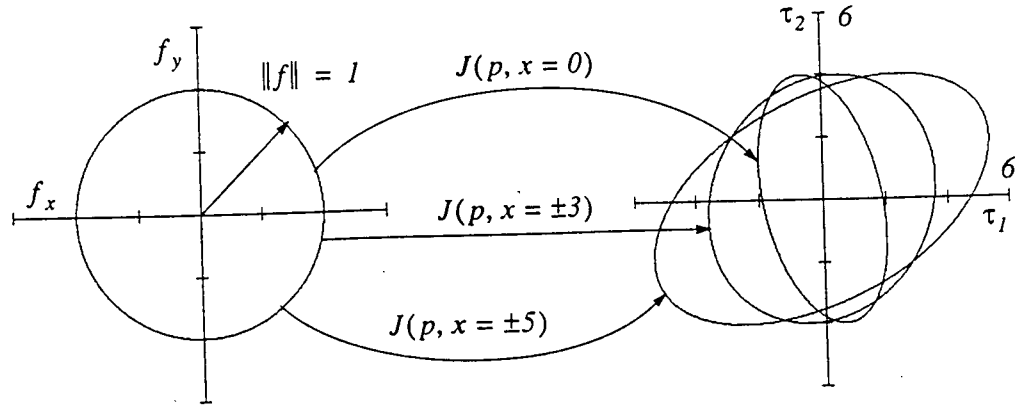


Figure 3.3: Force/Torque Transformation

The shape of each ellipse relates to directional isotropy and roundness presents positional isotropy. As it can be seen in the above figure, the condition number only measures the roundness of the ellipse, but ignores information related to its size. So directional isotropy is lacking in definition of the condition number in (3.15). For example, the ellipses at $x=0$ and $x=\pm 5$ have the same shape, but their size are quite different. The condition number, which is the ratio of the largest singular values to the smallest one is the same at both positions. However, the average of singular values is different at each position. To overcome this deficiency, [49] proposes the global isotropy index (GII).

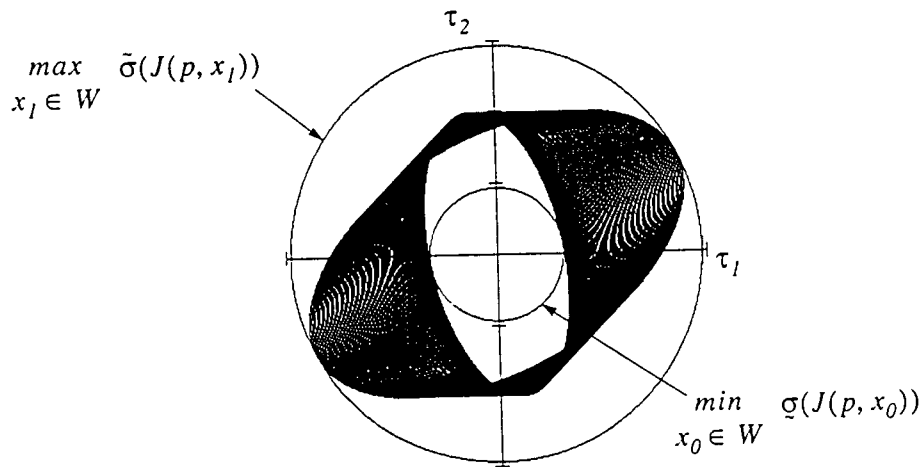


Figure 3.4: Force/Torque Ellipse and GII

In Figure (3.4), local actuator torque ellipses are computed and plotted for the previous manipulator at all values of x ranging from -5 to 5 [49]. The GII is the ratio of the radius of the largest circle contained within all of these ellipses to the radius of the smallest circle containing all of these ellipses. When all singular values are identical throughout the workspace, the mechanism is both directionally and positionally isotropic at each position of its workspace and the GII has a value of unity. Therefore a manipulator behaves consistently in all directions and that behavior doesn't change as the mechanism moves in its workspace.

An optimally isotropic robot design parameter p^* is the one that maximizes the value of the GII.

$$p^* = \arg \max_{p \in P} GII(p) \quad (3.16)$$

3.4 The Minimax Robot Design Optimization

The culling algorithm is used in [48] to optimize for GII to find the optimal design parameter that provides the best isotropic design for the manipulator. It is a branch and bound optimization method that culls non-optimal parameters from the search space until the best parameter remains. The method discretizes both the end-effector position space and the design parameter space to finite number of parameters and discovers the worst parameters in each search and culls them off.

At the beginning of the algorithm, a random point P_i is chosen from the set of all design parameter candidates. The singular values of all positions in the workspace are computed for that particular design parameter and two positions x_i and x_j are found that provide the minimum singular value and maximum singular value for P_i . The GII is measured as the ratio of minimum singular value to the maximum one obtained at those positions. The new GII is compared to the old one and if it is greater than the old one, the best GII found so far is updated to the new one.

In the next step, all the singular values at positions x_i and x_j are measured for all points in the design parameter space. For each design parameter, the minimum singular values found at those two positions are compared to each other and also compared to the best

minimum singular value from previous iterations. The minimum of these three numbers is set as the upper bound for the minimum singular values. The same strategy is applied to find the maximum singular values lower bound.

All the design parameters that their ratio of minimum singular value upper bound to maximum singular value lower bound is less than the best GII found so far are culled from the design parameter space. The design parameter that has the maximum ratio of minimum singular value upper bound to maximum singular value lower bound is chosen as the next candidate. The same steps are applied to the next chosen parameter and more parameters are culled until eventually only one parameter is left in the search space that is considered as the best result.

The culling algorithm reaches a global optimum with less objective value calculations than a global search since many function evaluations are avoided. However, it is limited to solving only for minimax problems. The algorithm becomes significantly less efficient when the value obtained from the objective function does not change through large parts of the workspace. Also because the search space is discretized, the resolution of the result weakens since in case of real numbers, adding even one more resolution makes a major increase in the size of search space. Even though in each iteration, at least one parameter is culled from the search space, the computational effort is still huge. Since an average of one or two positions are checked for each design parameter, the parameter space search doesn't improve the optimization approach. Also if better results are not found in some iterations, the algorithm turns into an exhaustive search. Therefore an alternative method, a genetic algorithm, is proposed in this work to overcome the deficiencies of the culling algorithm in solving for kinematic design optimization of manipulators.

Chapter 4

Genetic Algorithms

This chapter provides an introduction to evolutionary and genetic algorithms. In section 4.1, the general concept of evolutionary algorithms is described. In section 4.2, the genetic algorithms are introduced and their operators are discussed in section 4.3. In section 4.4, a number of selection schemes are describes and in section 4.5, various algorithms for recombination are presented. Mutation algorithms are defined in section 4.6 and in section 4.7, other GA steps and operators are briefly described. Finally in section 4.8, the GA termination criteria are discussed.

4.1 Evolutionary Algorithms

Evolutionary algorithms are random search techniques that take their inspiration from natural selection and survival of the fittest in the biological world. They differ from more traditional optimization techniques in the way that they involve a search from a population of solutions, not from a single point. At each generation, a new set of estimations is created by selecting individuals based on their level of fitness and breeding them using natural genetics operators. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

Evolutionary algorithms model natural processes, such as selection, recombination, mutation, and migration. The solutions with high fitness scores are recombined with other solutions by swapping parts of a solution with another. Solutions are also mutated by making a small change to a single element of the solution.

The idea of evolutionary computing was first introduced in the 1960s by I. Rechenberg from Germany in his work on "Evolution strategies". His idea was then developed by other researchers to introduce different approaches of evolutionary algorithms.

As evolutionary algorithms require no derivative information or formal initial estimates of the solution, and are stochastic in nature, they are capable of searching the entire solution space with more chance of finding the global optimum than traditional optimization algorithms. They perform well on functions with multiple local optima and can find a globally optimal solution.

4.2 Genetic Algorithms

Genetic algorithms (GAs) are part of the evolutionary computing that were first introduced by John Holland and developed by him, his students and colleagues at the University of Michigan. GAs are adaptive evolutionary methods, which are inspired by the genetic processes of biological organisms. Over many generations, natural populations evolve according to the principles of Darwinian theory of survival of the fittest. By mimicking this process, genetic algorithms are able to evolve solutions to real world problems.

GA's have received a considerable attention regarding their potential in dealing with constrained problems. They are robust, adaptive and very different from conventional search and optimization methods. They are very powerful tools in solving complex, multi-variable, real-world search and optimization problems and therefore, they have become a common optimization technique in the field of Artificial Intelligence.

They work with a population of individuals, each representing a possible solution. These individuals can be represented as coded strings containing a set of parameters. These parameters are known as genes and the whole string of variables for each possible solution is referred to as a chromosome. Each chromosome is assigned a fitness score according to how good it is as a solution to the problem. The fitness score can be thought as a measure of profit, utility, or goodness. The highly fit individuals are given opportunities to reproduce, by breeding with other individuals in the population. This produces new individuals as offspring, which share some features taken from each parent. The least fit members of the population are less likely to be selected for reproduction, and so they die off.

A whole new population of possible solutions is thus produced by selecting the best individuals from the current generation, and mating them to produce a new set of individuals. If the GA is well designed, the population will converge to an optimal solution to the problem.

The applications of GAs are numerous. They include optimization problems, scheduling, planning, games, image processing, adaptive fuzzy controllers and neural network weight learning. Also, GAs can be used directly to predict the stock market, make buy/sell decisions or to schedule airline flights.

The genetic algorithm approach is the result of the search for robustness. Natural systems are robust and efficient as they adapt to a wide variety of environments. By simulating nature's adaptation algorithm, we can reach similar extent of performance.

4.2.1 GA Advantages

Genetic Algorithms differ from more traditional optimization and search procedures in various ways. They usually work with a coding of the parameter set, not the parameters themselves. The coding discretizes the search space of the problem. Thus, GAs are able to work with discrete or discontinuous functions. The objective function in a GA can be numerical or logical because the variables are coded. This gives GAs a great flexibility to be applied to a wide range of systems.

GAs evaluate a population of possible solutions instead of a single point. Hence, the probability of getting trapped in a local optimum is reduced over single point search methods.

Most of the traditional techniques need to measure some particular characteristics of the problem such as gradient, Hessian or so to determine the next search point whereas in a GA, the next search is directed based on stochastic rules and there is no need for deterministic transition rules.

Another characteristic of GAs that makes them more flexible for a variety of real world problems is that they deal with the objective function itself to explore the search space and there is no need for derivative computation or other secondary functions.

GAs are a suitable method for multi-objective optimization and scheduling problems since they can provide a number of potential solutions. Therefore, the final solution is based on the user's choice from all the alternative solutions.

In general, genetic algorithms are applicable in optimization of many systems because no restrictions for the definition of the objective function exist. Simplicity of operation and the precision of the results are the main appeals of genetic algorithms.

4.2.2. GA Coding

In GAs coding, each individual of the population is represented as a sequence of genes from a certain alphabet. This whole string of the individual representation is called a chromosome that could consist of binary digits, floating point numbers, integers or real numbers. The coding scheme in a GA determines how the problem is structured and what type of genetic operators should be used. The common way in the past for coding the parameters of a GA was a binary string representation. For example for two individuals with real values of 26 and 17, the chromosomes would look like this:

Chromosome 1	11010
Chromosome 2	10001

Table 4.1: Binary Representation of Individuals in a GA

As table 4.1 indicates, each chromosome has one binary string. Each bit in this string represents a gene that contains some characteristics of the individual.

The most suitable way of coding depends mainly on the nature of the problem. However, it has been shown that the real-valued GA is an order of magnitude more efficient in terms of CPU time. In addition, a real-valued representation offers higher precision with more consistent results across replications [25].

In [27], some special encoding techniques have been introduced that are claimed to result in faster convergence of Genetic Algorithms.

4.3 GA Major Operators

A simple genetic algorithms is basically composed of three major operators:

1. Reproduction
2. Recombination
3. Mutation

At the beginning of the algorithm, a number of individuals are randomly initialized to produce the first population or generation. The objective function is then evaluated for every single individual. If the optimization criteria are not met, another new generation is created. For this purpose, individuals of old generation are selected as parents according to their fitness and recombined to produce offspring. All offspring will be mutated with a certain probability and then the fitness of this new generation is measured to evaluate the generation. The process is repeated until the optimization criteria are reached.

When choosing the size of the population, attention should be paid because if there are too few chromosomes selected, the GA has a few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, the GA slows down. Research shows that after some limits, which depend mainly on the nature of the problem and the method of encoding, it is not useful to increase population size because it drops the speed of the convergence of the problem

In the following sections, various operators of a genetic algorithm are described in more details.

4.4 Reproduction

Reproduction is a process in which individuals are selected according to their fitness function for mating. In reproduction, each string in the selection pool is assigned a reproduction probability according to its own objective value and the objective value of all other individuals in the selection pool. Individuals with higher reproduction probability are more likely to produce offspring for the next generation. If reproduction is not performed correctly, weak members are selected for producing the next generation. This will result in a divergence in GA operation since the fit individuals that are not

selected for reproduction are caused to die off and have no chance to transmit their genes to the next generation.

There are several selection methods for determining which individuals should be chosen for recombination. Some of the most common ones have been described in next parts briefly.

4.4.1 Roulette Wheel Selection

Roulette wheel selection or “stochastic sampling with replacement” is one of the simplest and most common selection methods. It is a stochastic scheme in which individuals are mapped to adjacent slices of a roulette wheel. The mapping is shown in Figure (4.1) for a population of four individuals. Each individual's segment size is proportional to its fitness score. Individuals with higher fitness score have more chances to be selected as mating population. The type of fitness assignment is considered the “proportional fitness assignment” which has been explained in section 4.4.2.

For selecting the mating population, a random number is uniformly generated between 0.0 and 1.0 and the individual whose slice spans the random number is selected. The process is repeated until the desired number of individuals for recombination is obtained.

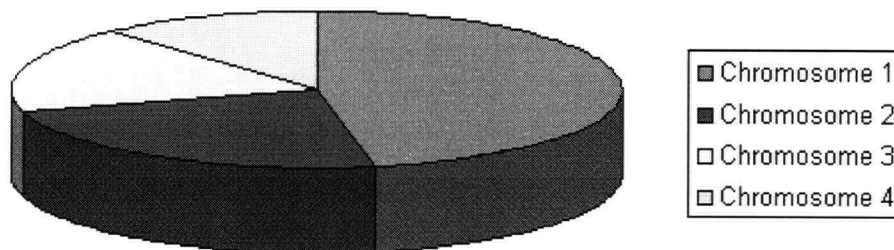


Figure 4.1: Roulette Wheel Selection

4.4.2 Proportional Fitness Assignment

In proportional fitness assignment, the individuals are selected with a probability proportional to their fitness values. For a fixed population size, the probability P_i of selecting the i^{th} member of the population is shown in (4.1).

$$P_i = f_i / \sum_j^{N_{ind}} f_j \quad (4.1)$$

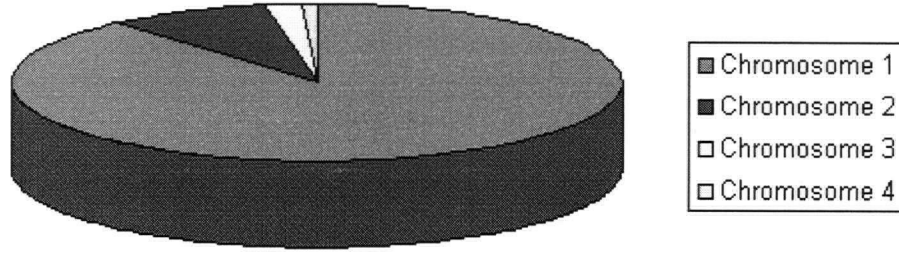
Where f_i is the fitness value of the i^{th} individual and N_{ind} is the number of individuals in population. As the expression (4.1) indicates, this method gives a greater chance to above-average individuals to be selected as parents than below-average members.

4.4.3 Rank Based Fitness Assignment

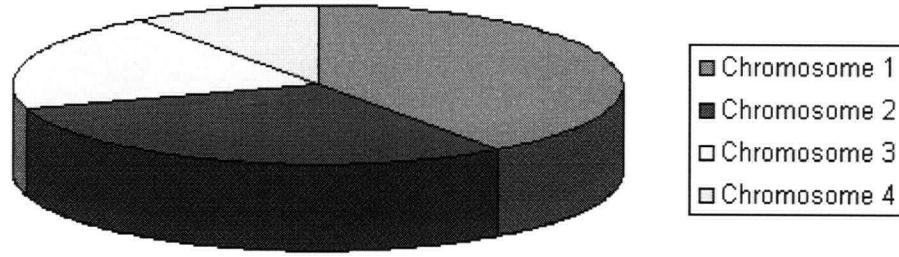
In this algorithm, the individuals are sorted according to their objective values. Then, each individual is assigned a fitness score based on its sorted position and not the objective value. The worst individual will have position 1, second worst 2, and the best will set in position N_{ind} where N_{ind} is the number of individuals in population. After ranking, each individual's fitness is normalized by the total fitness of the population to assign the probability of selection to that individual.

The method overcomes the weakness of the proportional fitness assignment when the objective values of the individuals of the population significantly differ. For example, if the best individual's fitness has result in the occupation of 90% of the roulette wheel, then the rest of the individuals will have very few chance to be selected for mating. This results in premature convergence since the gap has caused the search to narrow down too quickly.

Figure (4.2) shows the distribution of individuals in roulette wheel selection with proportional fitness assignment and rank based fitness assignment.



Situation Before Ranking (Graph of Fitness Scores)



Situation After Ranking (Graph of Order Numbers)

Figure 4.2: Proportional and Rank Based Fitness Assignment

Ranking can be performed in two different forms of “linear ranking” and “non-linear ranking”. In [45], selective pressure is defined as the probability of the best individual being selected compared to the average probability of selection of all individuals. If we consider P_{os} as the position of an individual in the population, least fit individual has $P_{os}=1$, and the fittest individual has $P_{os}=N_{ind}$. For Linear ranking, the fitness value for an individual is calculated as [45]:

$$Fitness(P_{os}) = 2 - SP + 2(SP-1)(P_{os}-1)/(N_{ind}-1) \quad (4.2)$$

SP stands for the selective pressure. Linear ranking allows a range $[1.0, 2.0]$ for values of selective pressure.

The expression for non-linear ranking is shown in (4.3).

$$Fitness(Pos) = N_{ind} \cdot X^{pos-1} / \sum_{i=1}^{N_{ind}} X^{i-1} \quad (4.3)$$

Where X is computed as the root of the polynomial (4.4).

$$(SP-1).X_{ind}^{N-1} + SP.X_{ind}^{N-2} + + SP.X + SP = 0 \quad (4.4)$$

In non-linear ranking higher values of selective pressures in $[1, N_{ind}-2]$ can be considered.

4.4.4 Stochastic Universal Sampling

In this algorithm, the individuals are mapped to contiguous segments of a roulette wheel according to their fitness function as in roulette wheel selection. However, instead of random selection of individuals, equally spaced pointers are placed over the wheel as many as the desired number of individuals. If we consider N_{ind} as the number of individuals to be selected, then the angle between each two adjacent pointers is $360^\circ/N_{ind}$ and the position of the first pointer is given by a randomly generated number in the range $[0, 360^\circ/N_{ind}]$. This method offers a selection of offspring which is more efficient than the roulette wheel selection.

4.4.5 Truncation Selection

Truncation selection is an artificial selection method that is usually used for large populations. In this method, individuals are ranked according to their fitness and the best individuals are selected for parents to produce offspring.

4.4.6 Tournament Selection

In tournament selection, a group of individuals is chosen randomly from the population and the fittest individual from this group is selected as a parent. This process is repeated until enough mating individuals are selected. The parameter for tournament selection is the tournament size. It takes values ranging from 2 to the number of individuals in population.

4.4.7 Local Selection

In this method, individuals are dwelled in several constrained environments called the local neighborhoods and each individual is considered only with regard to the individuals inside its neighborhood. The structure of the neighborhood can be linear in forms of full ring, half ring, two-dimensional in forms of full cross, half cross or full star/half star and also three-dimensional with any combination of the above structures [45].

At the beginning of the selection, first half of the mating population is selected uniformly at random. Each of the pre-defined selection methods can be used for this purpose. Then for each selected individual, a local neighborhood is considered and the best member of that region is chosen for mating with that individual. The selection of the mating individual from local neighborhood can be made in a number of different ways such as random selection.

The size of the neighborhood is determined by the distance from other neighbors and the structure of the neighborhood. The speed of transmission of genes between the individuals is related to the size of the neighborhood. Local selection in a small neighborhood usually gives better results than local selection in a bigger neighborhood. However, this might lead to less diversity in interaction between the individuals and premature convergence. Therefore, a precise decision should be made to meet both rapid propagation and maintenance of diversity in the population.

4.4.8 Comparison of Selection Schemes

As mentioned before, rank-based fitness assignment overcomes the scaling problems of the proportional fitness assignment. Ranking introduces a uniform scaling across the population and provides a simple and effective way of controlling selective pressure. The reproductive range is limited, so that no individuals generate an excessive number of offspring [45].

In truncation selection, all individuals below a certain fitness level do not have a probability to be selected as parents to produce offspring. Therefore, this selection scheme is more likely to eventually replace less fit individuals with fitter offspring.

Ranking and tournament selection seem to behave similarly. However, in tournament selection only discrete values can be assigned. Hence, ranking selection can be applied in a greater variety of problems where tournament selection cannot.

In general, rank-based fitness assignment behaves in a more robust manner than other selection methods and thus, is the chosen method of selection for this work.

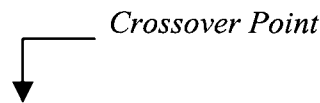
4.5 Recombination

After reproduction, recombination happens by mating each two individuals that have been selected as members of mating population during reproduction procedure. The variable values or genes of the parents are combined and the new individuals are produced.

Several methods of recombination are established based on the representation of the variables. The term “Crossover” is usually used when referring to the recombination of binary valued variables. Crossover is one of the first methods of recombination in which, variables of the individuals are represented as binary digits of 0 and 1. Crossover operates by choosing randomly some crossover points in the binary strings of parents. Two new individuals are created then by swapping the strings of the parents from the crossover points. Crossover is a special case of the discrete recombination and can be applied to integer valued and real valued variables as well by converting them to binary format.

4.5.1 Crossover

For the recombination of binary valued variables, the name “crossover” is used. In crossover, the binary string individual is divided into one or more segments at crossover points. The number of cross points differentiates between different crossover schemes. In single-point crossover, one crossover position is selected randomly from the range of $[1, 2, \dots, N_{var}-1]$ where N_{var} is number of variables of an individual. Then, the variable segments are swapped from this point and hence two new offspring are produced. Table (4.2) shows two offspring that are produced from two individuals by exchanging their genes at a random crossover point of 5.




Individual 1	10011	00100110110
Individual 2	01011	11000011110
Offspring 1	10011	11000011110
Offspring 2	01011	00100110110

Table 4.2: Single Crossover

In double-point crossover, two crossover positions are selected randomly and the variables of individuals are exchanged at these points to produce two new offspring. In multi-point crossover, m different crossover positions are chosen at random from the range $[1, 2, \dots, Nvar-1]$. The first segments of the parents that are chosen this way remain unchanged and the variables are exchanged between each successive crossover points of the two parents from the second segment.

Table (4.3) shows two individuals that are divided at crossover points 5, 9, and 16 and the two new children that are produced by exchanging the information between two parents.



Individual 1	10011	0010	0110110	001
Individual 2	01011	1100	0011110	100
Offspring 1	10011	1100	0110110	100
Offspring 2	01011	0010	0011110	001

Table 4.3: Multi-point Crossover

The multi-point crossover improves robustness of the GA by exploring the search space and avoids premature convergence early in the search.

Uniform crossover is another method of crossover in that a mask is created with the same length as the string of individuals. If we consider the following table, the offspring 1 is produced by using mask one. If the corresponding bit in the mask is 1, that gene is taken from individual 1 and if the corresponding mask bit is 0, that bit is taken from individual 2.

Individual 1	01110011010
Individual 2	10101100101
Mask 1	01100011010
Mask 2	10011100101
Offspring 1	11101111111
Offspring 2	00110000000

Table 4.4: Uniform Crossover

In shuffle crossover, a single crossover position is selected like single-point crossover. But before the variables are exchanged, they are randomly shuffled in both parents. After recombination, the variables in the offspring are unshuffled in reverse. This removes positional bias as the variables are randomly reassigned each time crossover is performed [45].

Crossover probability indicates how often will the crossover be performed. If crossover probability is 100%, then all offspring are made by crossover. If it is 0%, whole new generation is made from exact copies of individuals from old population, however this does not mean that the new generation is the same.

Crossover is made with the hope that new chromosomes will have good parts of old chromosomes and maybe the new chromosomes will be better. However it is useful to leave some part of population survive to next generation.

4.5.2 Discrete Recombination

Discrete recombination can be applied to binary, integer, or real variables. In discrete recombination, each individual is chosen randomly from the mating pool and the values of variables are exchanged between each two parents to produce individuals that inherit characteristics from their parents.

If we consider the following two individuals in table (4.5) with 5 variables each, there would be two new children produced by randomly choosing each of their variables from the corresponding variable of those two individuals.

Individual 1	67 87 23 29 94
Individual 2	20 95 34 55 74
Offspring 1	20 95 23 29 74
Offspring 2	20 87 23 55 74

Table 4.5: Discrete Recombination

It is obvious that this method is not appropriate when the individuals consist of just one variable.

4.5.3 Intermediate Recombination

Intermediate recombination is applied only to real value variables. In this method, the variable values of the offspring are chosen somewhere around and between the variable values of the parents from expression (4.5) [45].

$$Var_i^O = Var_i^{P1} + Var_i^{P2}(1-a_i) \quad i \in (1, 2, \dots, Nvar) \quad (4.5)$$

$a_i \in [-d, 1+d]$ uniform at random, $d=0.25$, a_i for each new i

Where Var_i^O stands for the i^{th} variable of offspring and Var_i^P for the i^{th} variable of the parents. For each new variable, a is chosen uniformly at random over an interval $[-d, 1+d]$. The variable d defines the size of the area for possible offspring, which is usually set to 0.25. This value is driven statistically to make sure that the range for choosing the offspring variables always remains the same as the one owned by the parents and doesn't decrease over the generations.

4.5.4 Line and Extended Line Recombination

Line recombination is the same as the intermediate recombination except that in Line recombination, only one value 0 is used for a_i .

In Extended line recombination, the parents define a line based on their variable values and the offspring are created on this line. The domain of variables defines the size of the area for possible offspring. The selection of offspring is not distributed at random. The probability of creating offspring near the parents is high and the probability that offspring

are created far away from the parents is low. Moreover, if the fitness of the parents is available, then offspring are mostly created around the fitter parents.

4.6 Mutation

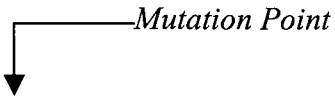
Mutation is another operator of genetic algorithms that normally takes place after recombination. Mutation is needed because, despite the fact that reproduction and crossover search and recombine the individuals, they may occasionally lose some potentially useful genetic characteristics. Mutation prevents the solutions to fall into a local optimum, but it should not occur very often, because then it turns the GA into a random search.

Mutation usually happens with small mutation step and low probability or mutation rate. Mutation step refers to the size of the changes for each mutated variable and mutation probability says how often parts of chromosome are mutated. If mutation probability is 100%, the whole chromosome is changed, if it is 0%, nothing is changed. Usually mutation steps and the mutation rate remain constant during a genetic algorithm run. However, in some cases, they change during the run according to the information obtained from the previous mutation.

Different mutation methods have been described in the following sections. In the past, real values were decoded to binary forms before mutation. Now with powerful real value techniques explained in section 4.6.2., there is no need for decoding and mutation can directly happen to real values.

4.6.1 Binary Mutation

In binary representation of variables, each string contains only two state of 0 or 1. In binary mutation, a variable value is chosen at random. Then, the value of that gene is switched from 1 to 0 or from 0 to 1. Table 4.6 shows the switching of the chosen gene from 1 to 0 after applying mutation.



Individual Before Mutation	1110011010
Individual After Mutation	1111011010

Table 4.6: Binary Mutation

The size of the mutation step is always 1 in binary mutation.

4.6.2 Real Value Mutation

In real value mutation, randomly created values are added to the real value variables with a low probability. The probability of mutating a variable is inversely proportional to the number of variables or dimension of the problem. The more dimensions one individual has, the smaller is the mutation probability. [45]

It has been discussed that a mutation rate of $1/N_{ind}$ produces good results for a wide variety of functions [45]. With this assumption, only one variable per individual is changed in each mutation.

When the individual is well adapted, small mutation steps often work better to reach the optimum result. However, small steps decrease the speed of convergence. Usually, small step size with a high mutation rate and large step sizes with a low mutation rate works as an efficient operator.

4.7 Other GA Operators

In this section, other effective genetic algorithm operators are mentioned. They usually play a secondary role in GA.

4.7.1 Reinsertion

When size of the new generation differs from size of the old one, reinsertion operates to maintain the size of the original population. Therefore, if less offspring are generated than the parents, some of the offspring are reinserted into the old population to produce another offspring until the required number of individuals is attained. Also, if more

offspring are produced than needed, then reinsertion selects the individuals that must be omitted from the new population.

There are two forms of reinsertion scheme, local reinsertion and global reinsertion. Local reinsertion is used along with local selection and global reinsertion is used for all other selection methods.

Local reinsertion happens in the same bounded neighborhood with the same structure as in the local neighborhood to save the locality of the information. Local reinsertion happens by inserting every offspring and replacing individuals in the neighborhood uniformly at random or replacing the weakest individuals in the neighborhood. It can also happen by inserting offspring fitter than weakest individual in the neighborhood and replacing weakest individuals or replacing parents. Another form of local reinsertion can happen by inserting offspring fitter than weakest individual in the neighborhood and replacing parents or replacing individuals in the neighborhood uniformly at random.

As well, there are several schemes of global reinsertion. The simplest scheme is called “pure reinsertion” in which the number of produced offspring is the same as parents and all parents are replaced by the offspring. In this scheme, every individual lives only one generation and therefore, it is possible that very good individuals are replaced without producing better offspring. In “uniform reinsertion”, less offspring are produced than parents and parents are replaced uniformly at random. In “elitist reinsertion”, less offspring than parents are produced and the worst parents are replaced by offspring whereas in “fitness-based reinsertion”, more offspring are produced than needed and only the best offspring are reinserted.

The elitist combined with fitness-based reinsertion prevents the information loss as it happens in pure reinsertion and therefore is the recommended method.

4.7.2 Population Models

Based on the definition of the selection pool, three population models can be defined. They are local model, global model and regional model.

The local model restricts the selection of parents to a local neighborhood. The local model considers every individual separately and uses the local selection to get the mating partner in a local neighborhood.

In global model, there are no limitations and the selection takes place inside the entire population. Two terms of “master” and “slave” are usually used in this model. Master handles the calculations where the whole population is needed such as fitness assignment or selection and a number of slaves do the rest of the calculations that are basically defined for one or two individuals such as recombination and mutation. This is known as synchronous master-slave-structure [45].

The slave calculations can be done in parallel. When evaluation of the objective function is the most time consuming part, the whole evolutionary algorithm is usually calculated by the master and only the objective function evaluation is done by the slaves. This defined distribution of objective function evaluation can be employed for any other population model as well to reduce long computation times.

In the regional model, parents are selected from isolated parts of the population called subpopulations. Inside each subpopulation, no limitations exist on selection the same as the global model. The subpopulations evolve separately for a certain number of generations, which is termed as the isolation time. After the isolation time, some of individuals are exchanged between the subpopulation. This process is called migration and the number of exchanged individuals is considered as the migration rate. The individuals for migration can be selected uniformly at random or best individuals are can be selected. Many possibilities exist for the structure of the migration of individuals between subpopulation. In the most general migration strategy, the individuals may migrate from any subpopulation to another. A pool of potential immigrants from other subpopulations is created in each subpopulation and the individuals are then selected uniformly at random from this pool.

In comparison to the single population algorithm, the parallel implementation of the regional model in forms of various subpopulations decreases the computation time, as it needs less objective function evaluations to find the global optimum. It has been proved

that the multi population evolutionary algorithm reaches better results for many systems than a single population algorithm with the same number of individuals.

4.8 GA Termination

The GAs move forward from one generation to the next until a termination criterion is met. One termination criterion involves the detection of the population convergence. In [4], Convergence is defined as the progression towards increasing uniformity. A gene is said in [4] to have converged when %95 of the population share the same value and the population said to have converged when all the genes have converged. In [25], when the sum of the deviations among individuals becomes smaller than some specified threshold, the algorithm is terminated.

The most frequently used stopping criterion in most of the GA software packages and tools is the completion of a pre-defined number of generations. This simple strategy is applied in Genetic and Evolutionary Toolbox for MATLAB that has been used in this work.

Chapter 5

Minimax Genetic Algorithms

This chapter describes genetic algorithms that have been developed to solve the minimax problem for both local and global isotropy. In section 5.1, the genetic and evolutionary toolbox in MATLAB that has been used in this work, is described and section 5.2 demonstrates the developed genetic algorithms.

5.1 Genetic and Evolutionary Algorithm Toolbox in MATLAB

MATLAB is a powerful language for technical computing and simulation in a broad range of areas. It has an open design that allows the users to develop additional tools for their own applications.

The Genetic and Evolutionary Algorithm Toolbox (GEATbx) is an application in MATLAB for evolutionary algorithms to optimize a wide variety of systems. The toolbox was developed by Hartmut Pohlheim in 1995 [44]. The first release has been updated several times since then and some new features and functions have been added. For this work, version 1.92 is used which has been last updated in October 1999.

The GEATbx contains m-file implementation of the GA steps that are executed throughout the completion of the algorithm. For a variety of the genetic algorithm features, associated parameters have been set internally to appropriate default values. Some of these values are calculated based on the number of variables and size of the problem. However, all these default values can be set to a desired value by the user inside the program.

Some particular algorithms such as multi-population genetic algorithms are produced as specific commands in the toolbox. These functions add some specific features to the simple genetic algorithm and thus define a special algorithm. In multi-population genetic algorithm, each population is divided into several subpopulations that are stored in the

population matrix in order. For a few generations, every subpopulation evolves isolated the same as a simple genetic algorithm. Then one or more individuals are exchanged between the subpopulations. This produces better results and models the evolutionary algorithm in a way more similar to nature than the simple genetic algorithm.

Choosing the appropriate evolutionary algorithm is mostly dependent on the format of the variables in the problem. In the toolbox, real, binary, and integer representation of the variables are supported. There exist functions for conversion between these representations. If the variables of the objective function are real, the real value representation for the genetic algorithm implementation is recommended.

The implementation of the problem's objective function is the most demanding issue while using the toolbox. Inside the objective function m-file, all the problem specific variables and default values for domain of variables should be defined. The objective function is called with a randomly created matrix representing the initial population of the first generation. This matrix contains as many rows as the predefined number of individuals and as many columns as the number of variables of the objective function or in other words, the dimension of the problem.

The mathematical expression of objective function is calculated for every single individual of the population. Each individual is assigned a fitness function based on its objective function value and the process moves towards the recombination of the best individuals, which are selected as parents for the next generation. Other steps of the GA such as recombination and mutation are taken place and therefore, the second generation of individuals is generated. The same evaluations are made for the new generation until the best result is derived.

The mathematical representation of the fitness function can be calculated for every single individual of the population in a for-loop. However, the execution time can be greatly reduced by vectorization.

5.1.1 Features of the GEATbx

The general frame for all types of data in the GEATbx is a 2-D matrix. This allows the toolbox to be compatible with the older versions of MATLAB such as MATLAB 4.

Chromosomes are one type of data that are used in the toolbox. They are also called individuals and classified as genotypes. The entire population is stored in chromosome data structure in the form of a matrix with as many rows as the individuals of the population and as many columns as the variables of each individual. Each row represents the genotypic representation of that individual that might be binary, integer or real values.

Decision variables or phenotypes are another type of data in the toolbox. It is possible to have integer, real and binary variables in the same phenotype data structure. In many occasions, the phenotypes and genotypes are basically the same and there is no mapping necessary between these two sets of data. For example, if variables are real and the algorithm has been defined to work with real valued variables, phenotypes and genotypes are identical and no mapping is necessary.

Objective function is a scalar number that assigns a performance evaluation to each phenotype. In case of multiple objectives, objective function is in fact a vector containing components that correspond to each objective. Each row of the objective function matrix represents a particular individual's objective vector. The number of columns of the objective function matrix is equal to the number of individuals of the current population.

Fitness values are another sort of data that are derived from the objective values. They are non-negative values that are not necessarily the same as objective values. Fitness values are stored in a matrix with same dimension as objective function's matrix. In case of multi-objective functions, the fitness of each individual is derived as a function of the vector of objective function values.

The central point in the GA is a main function that calls all the necessary GA operators and does most of the administrative works such as population initialization, displaying and saving the results. When the population of individuals is initialized, the GA starts. New populations are generated and are evaluated by GA operators for predefined number of generations.

Selection operator has been supported in toolbox in several forms such as linear/non-linear ranking, stochastic universal sampling, truncation, tournament, roulette wheel, and local selection. The desired method can be chosen by the user and set internally within the program. Recombination has been defined in a number of forms as well such as

discrete, intermediate, line, extended line and also as crossover for binary representation of variables in single/double point, shuffle and uniform crossover. For mutation, toolbox supports binary and real valued mutation depending on the variable representation. For real value representation, several mutation functions are represented.

There are also a number of different population models that have been defined in the toolbox. Global model, regional model (multiple subpopulations) and local model (local selection, global/regional, local reinsertion, and different neighborhood structures) can be mentioned as some of them. The migration operator is defined in regional model in the forms of unrestricted, ring, and neighborhood.

The results can be shown in a classified form during the execution of the algorithm. The format of the visualization can be set to have the number of objective function calls, best objective value and rank of subpopulations on the screen as well as the graphical results while the program is running.

5.2 Minimax Genetic Algorithm

In this section, the algorithm for solving the isotropic minimax problem is given. It is based on the implementation of a GA that runs another GA while executing to assign the fitness score to each individual of its current population.

At the beginning of the program, all the necessary input parameters of both GAs such as the number of generations, number of individuals in each generation, number of subpopulations of each population and the lower and upper bounds for each variable are defined. However, these are not the only default parameters that need to be set. Other options such recombination, crossover, and mutation methods should be set internally as well as the generation gap and selective pressure.

If the user does not define the above inputs, the toolbox automatically considers $200\sqrt{n}$ generations where n is the number of independent variables of the objective function. Each population consists of $2\sqrt{n}$ subpopulations of $20+n/10$ individuals. The selection algorithm is set to stochastic universal sampling, the recombination algorithm to discrete recombination and the mutation algorithm to real mutation and the variables are treated

as real values. Furthermore, the generation gap and migration rate are set to 1 and 0.2, respectively.

Generally, the GA termination should be detected automatically by the program as it converges towards the optimum result. This feature is not yet implemented in the genetic and evolutionary algorithm toolbox and the GA ends after evaluating the predefined number of generations. Therefore, attention should be paid when choosing the number of generations to find the optimal result. The adequate number of generations to get the best solution can be figured out after a few trials.

The algorithm is shown step by step for both local and global isotropy through equations (5.1) - (5.22).

List of Symbols

i, j, m, n	= looping indices
l	= design parameter
x	= end-effector position
σ_{min}	= minimum singular value at a position
σ_{max}	= maximum singular value at a position
N_{gen}	= number of generations of a genetic algorithm
N_{ind}	= number of individuals of a population
N_{sub}	= number of subpopulations of a genetic algorithms
F	= fitness score of an individual
F^*	= best fitness score obtained through all generations
b_{max}	= upper bound of variables
b_{min}	= lower bound of variables
P	= current population of the genetic algorithm
p	= an individual of the current population
GA_1	= a minimization genetic algorithm
GA_2	= a maximization genetic algorithm

Prefixes 1 and 2 stand for the parameter of the GA_1 and GA_2 , respectively.

Minimax Genetic Algorithm Optimization Approach

$$\text{Set} \quad i = 0 \quad (5.1)$$

$$\text{Random} \quad {}^1P_0(l) \quad b_{\min}(l) \leq p \leq b_{\max}(l) ; \forall p \in {}^1P_0(l) \quad (5.2)$$

REPEAT

$$\text{Set} \quad j = 1 \quad (5.3)$$

REPEAT

$$\text{Choose} \quad l_j = {}^1p_j \quad (5.4)$$

$$\text{Set} \quad m = 0 \quad (5.5)$$

$$\text{Random} \quad {}^2P_0(x) \quad b_{\min}(x) \leq x \leq b_{\max}(x) ; \forall x \in {}^2P_0(x) \quad (5.6)$$

REPEAT

$$\text{Set} \quad n = 1 \quad (5.7)$$

REPEAT

$$\text{Choose} \quad x_n = {}^2p_n \quad (5.8)$$

$$\text{Find} \quad F(x_n) = \frac{\sigma_{\min}(l_j, x_n)}{\sigma_{\max}(l_j, x_n)} \quad (5.9)$$

$$\text{UNTIL} \quad n = {}^2N_{ind} + 1 \quad (5.10)$$

$$\text{Apply GA}_2 \text{ operators} \quad (5.11)$$

$$\text{if} \quad m \leq {}^2N_{gen} + 1 \quad (5.12)$$

$$\text{Set} \quad {}^2P_{m+1}(x) \quad (5.13)$$

$$\text{UNTIL} \quad m = {}^2N_{gen} + 1 \quad (5.14)$$

$$\text{Set} \quad F(l_j) = {}^2F^* \quad (5.15)$$

$$\text{UNTIL} \quad j = {}^l N_{ind} \quad (5.16)$$

$$\text{Apply GA}_1 \text{ operators} \quad (5.17)$$

$$\text{if} \quad i \leq {}^l N_{gen} + 1 \quad (5.18)$$

$$\text{Set} \quad {}^1 P_{i+1}(l) \quad (5.19)$$

$$\text{UNTIL} \quad i = {}^l N_{gen} + 1 \quad (5.20)$$

The minimax genetic solution is the parallel implementation of two GAs. GA₂ develops a minimization search to find the position that gives the minimum ratio of singular values for a given design parameter and GA₁ searches through all design parameters to find the optimum one that provides the maximum value of that minimum ratio at a position.

The algorithm starts in (5.1) with a loop to create the generations of the GA₁. The first population ${}^1 P_0(l)$ is randomly chosen in (5.2) from all the candidates in the design parameter space. Next, the first chromosome of the population enters into GA₂ as the known design parameter so that the isotropy can be evaluated for that particular link length at different workspace positions. Another loop starts in (5.5) to create the generations of GA₂ and the first population of the current generation is randomly chosen from the robot workspace points in (5.6). The measurement of the performance index takes place in (5.9) as a function of input design parameter l_j at a position x_n obtained from (5.8). The computation of the objective function is completed during the execution of the third loop for all positions of the current population of GA₁ and the next generation is created in (5.13) after applying the GA operators in (5.11).

After the GA₂ completes, the best performance index obtained during all its generations is assigned in (5.15) to the current design candidate as its fitness score for GA₁ maximization development. Similarly, the next individuals of the current population of GA₁ are chosen one by one and go through all the previous stages until the termination criteria are met.

Inside the loop, the objective function calculates the same mathematical expressions for each individual separately and since the algorithm contains several loops, it takes a considerably long time for the program to converge to the final result. This matter

aggravates as the dimension of the program increases since more generation and individuals are needed to converge to the optimal answer. Therefore, instead of defining the repeating nature of the objective function in for-loops, the program is vectorized so that each mathematical expression is calculated for the entire members of the population simultaneously. This action significantly reduces the execution time with the same final results as before.

The previous algorithm has been demonstrated for local isotropic optimization in which the minimum and maximum singular values of the performance index are both obtained at the same position. The algorithm can be modified for a global measure of isotropy by replacing expression (5.9) with (5.21) and (5.22).

$$\begin{aligned} \text{Find} \quad & x_{n1} = \arg \min \sigma_{\min}(l_i, x_n), \quad x_{n2} = \arg \max \sigma_{\max}(l_i, x_n) \\ & x_{n1}, x_{n2} \in x_n \end{aligned} \quad (5.21)$$

$$\text{Find} \quad F(x_{n1}, x_{n2}) = \frac{\sigma_{\min}(l_j, x_{n1})}{\sigma_{\max}(l_j, x_{n2})} \quad (5.22)$$

Here the comparison of the singular values happens at two different positions of the workspace. The smallest “minimum singular value” and the greatest “maximum singular value” of positions x_{n1} and x_{n2} are considered in the performance measure (5.22). Therefore the final result assures the global isotropy of robot throughout the entire workspace.

Chapter 6

Implementation of the Minimax GA for Robotic Manipulators

This chapter illustrates the application of genetic algorithms in kinematic design optimization for two typical robotic manipulators. In section 6.1, the method has been applied to a two-DOF planar elbow manipulator and the results have been compared to those obtained using the culling algorithm in [49]. In section 6.2, the algorithm is further developed to globally optimize the same manipulator. Finally, the method has been used to optimize the design of a 3-DOF planar manipulator in section 6.3.

6.1 Design of a Locally Isotropic 2-DOF Planar Manipulator

In [49], the minimax culling algorithm has been applied to find the local isotropic design of a planar elbow manipulator shown in Figure (6.1). The condition index is considered as a measure of the local isotropic performance, which is a non-linear, non-differentiable ratio of the minimum singular value of Jacobian matrix of the robot to the maximum one at a workspace position. The position space is reduced to one dimension x by setting y to a constant value of 2 unit of length. The minimum forearm length l_2 is calculated from (6.1) to ensure that the boundaries of the usable and reachable workspace shown in Figure (6.1) are separated by a minimum safety margin of length K such that $\{k_1, k_2\} \geq K$ [49]. Therefore, the parameter space and the position space are reduced to one dimension l_1 and x , respectively.

$$l_2 = \max\left(\left\|\sqrt{x_{\max}^2 + y^2} - l_1\right\|, \|y - l_1\|\right) + K \quad (6.1)$$

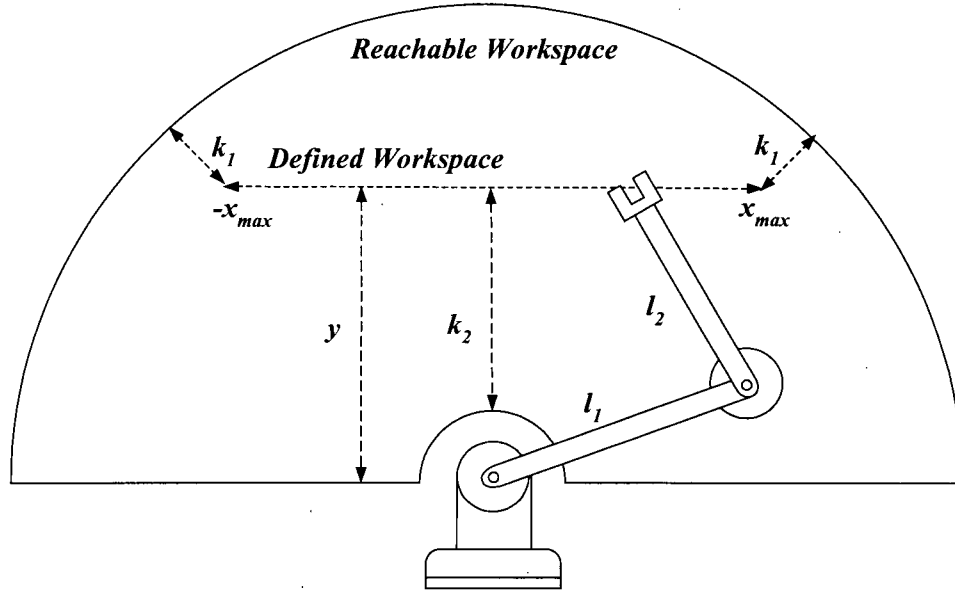


Figure 6.1: A 2-DOF Planar Elbow Manipulator

Applying the culling algorithm in [49], a resolution of one decimal point is considered for discretizing the parameter space. Assuming $x_{max}=5$ and $2 \leq l_1 \leq 8$, the defined workspace has been divided into 11 discrete positions and the parameter space has been divided into 61 discrete design parameters. With $K=0.4$ and an intermediate initial value of link length $l_1=6$, the culling algorithm starts by searching all the positions in workspace for that particular design parameter. The worse parameters are culled throughout the search in the parameter space until only $l_1=4.5$ remains to be the optimal link length that provides the best isotropic state for the manipulator at the optimal working point of $x=0$.

The 2-D and 3-D mesh plots of the objective function for the manipulator of Figure (6.1) are shown in Figure (6.2) with respect to the link length l_1 and position x .

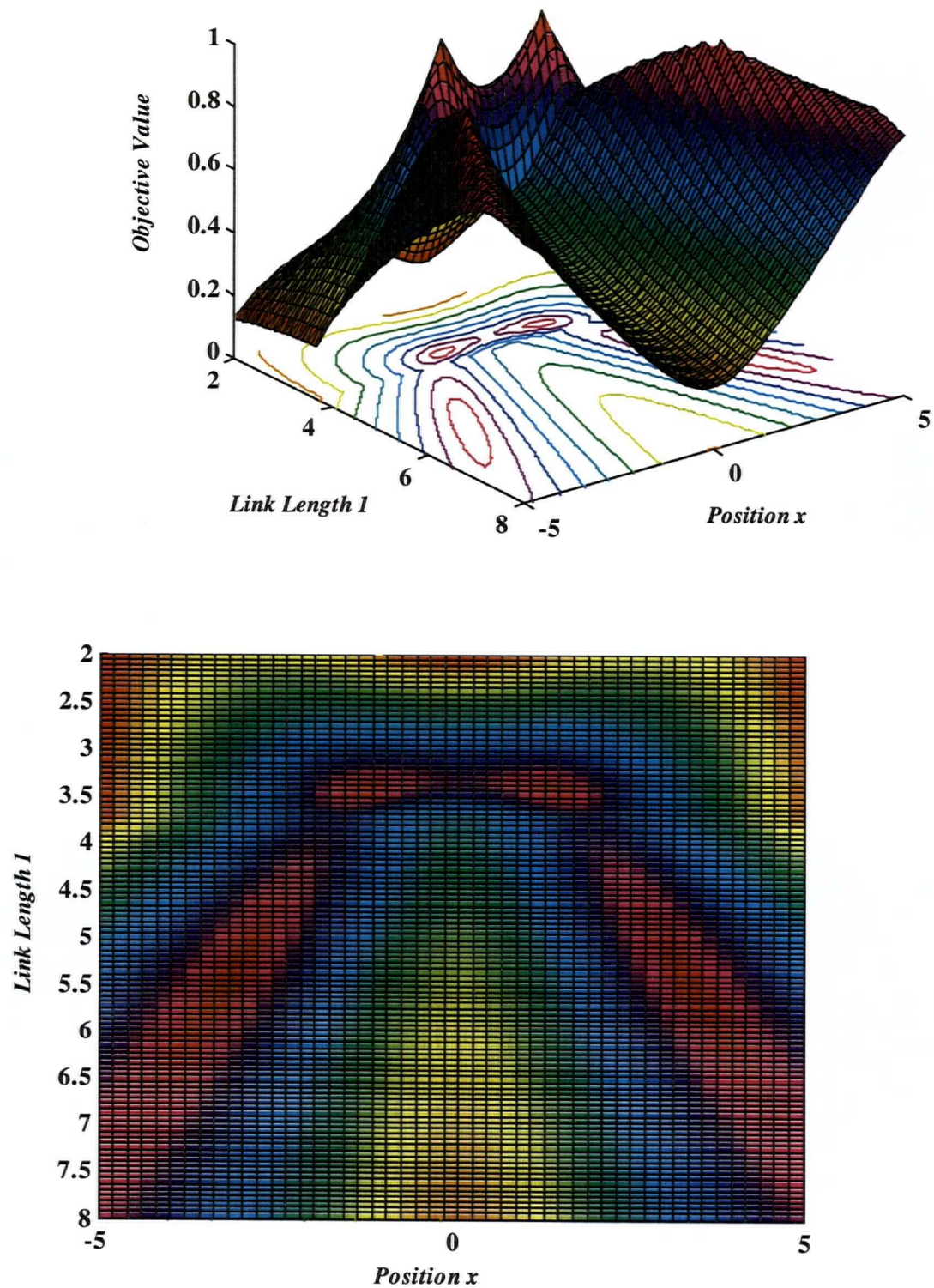


Figure 6.2: Surface and Contour Plots of Variation of the Local Isotropic Measures for a 2-DOF Planar Manipulator

According to Figure (6.2), if we take a move from the minimum objective value to the peak of the graph, the colors shift from dark orange to red as shown in Figure (6.3)



Figure 6.3: Color Classification

As the Figure (6.2) shows, the maximum “minimum value of objective function” is around $l_1=4.5$ at positions $x=0$ and $x=\pm 5$ and is associated with color green. In other words, the minimum condition number at $l_1=4.5$ has a greater value in comparison to all other link lengths and this minimum occurs at three end-effector positions.

For all other link lengths, the colors associated with the minimum objective function are all below green at all positions. Furthermore, for $l_1 \approx 4.5$, other positions offer a higher objective function value than those three positions because the colors related to the minimum objective function at other positions appear above the green area in Figure (6.3). Therefore, the optimal link length is located around $l_1=4.5$ that provides the most possible isotropic performance of the manipulator in Figure (6.1) at $x=0$ and the boundary points of $x = \pm 5$.

In minimax GA algorithms, the two multi-population genetic algorithms that are developed to optimize for the manipulator in Figure (6.1) run with 30 generations each. Every generation contains 2 subpopulations each of which includes 20 individuals. The program converges to the optimal solution of $GII=0.4014$ for link length $l_1=4.4798$ at workspace point of 0. The results from the last generation of one of the runs are shown in table 6.1. As the results indicate, the best isotropic configurations occur at either the intermediate point of $x=0$ or the end points of $x= \pm 5$ as was visually noticed before from Figure 6.2.

<i>Link Length</i>	<i>Position</i>	<i>Performance Measure</i>
4.3433	5.0000	0.3682
5.0878	-0.0000	0.3467
4.4879	0.0004	0.4006
4.4873	0.0000	0.4006
4.4719	-5.0000	0.3997
4.4849	-0.0000	0.4009
4.4983	-0.0000	0.3996
4.4737	-5.0000	0.4001
4.3282	5.0000	0.3644
4.7794	0.0001	0.3732
4.4909	0.0000	0.4003
4.4896	0.0000	0.4004
4.5381	0.0000	0.3957
4.4101	-5.0000	0.3847
4.4855	-0.0003	0.4008
5.0855	0.0001	0.3469
4.3532	5.0000	0.3707
4.4855	-0.0002	0.4008
4.0533	5.0000	0.2908
4.1857	-5.0000	0.3275
4.1855	-5.0000	0.3274
4.4914	-0.0000	0.4002
4.4098	5.0000	0.3847
4.5055	-0.0000	0.3989
4.4874	0.0001	0.4006
4.4410	5.0000	0.3923
4.5605	0.0001	0.3936
4.5581	0.0001	0.3938
4.5075	-0.0001	0.3987
5.6434	-0.0000	0.3056
4.5181	0.0004	0.3977
4.4799	0.0001	0.4014
4.4410	-5.0000	0.3923
4.5160	-0.0001	0.3979
4.4798	0.0000	0.4014
5.5872	-0.0000	0.3094
4.7715	-0.0004	0.3740
4.7645	-0.0001	0.3746
5.6185	0.0001	0.3072
4.1535	-5.0000	0.3188
5.6913	0.0000	0.3024
5.6913	0.0000	0.3024
3.2189	-5.0000	0.1630
3.5845	-5.0000	0.1671
4.2189	-5.0000	0.3363
4.6139	0.0001	0.3885

← Best Result

Table 6.1: The Results of the Last Generation of the GA for Local Isotropic Optimization of an Elbow Manipulator

The graphs in Figure (6.4) show the variation of best objective values and best individual candidates for l_1 throughout the 30 generations.

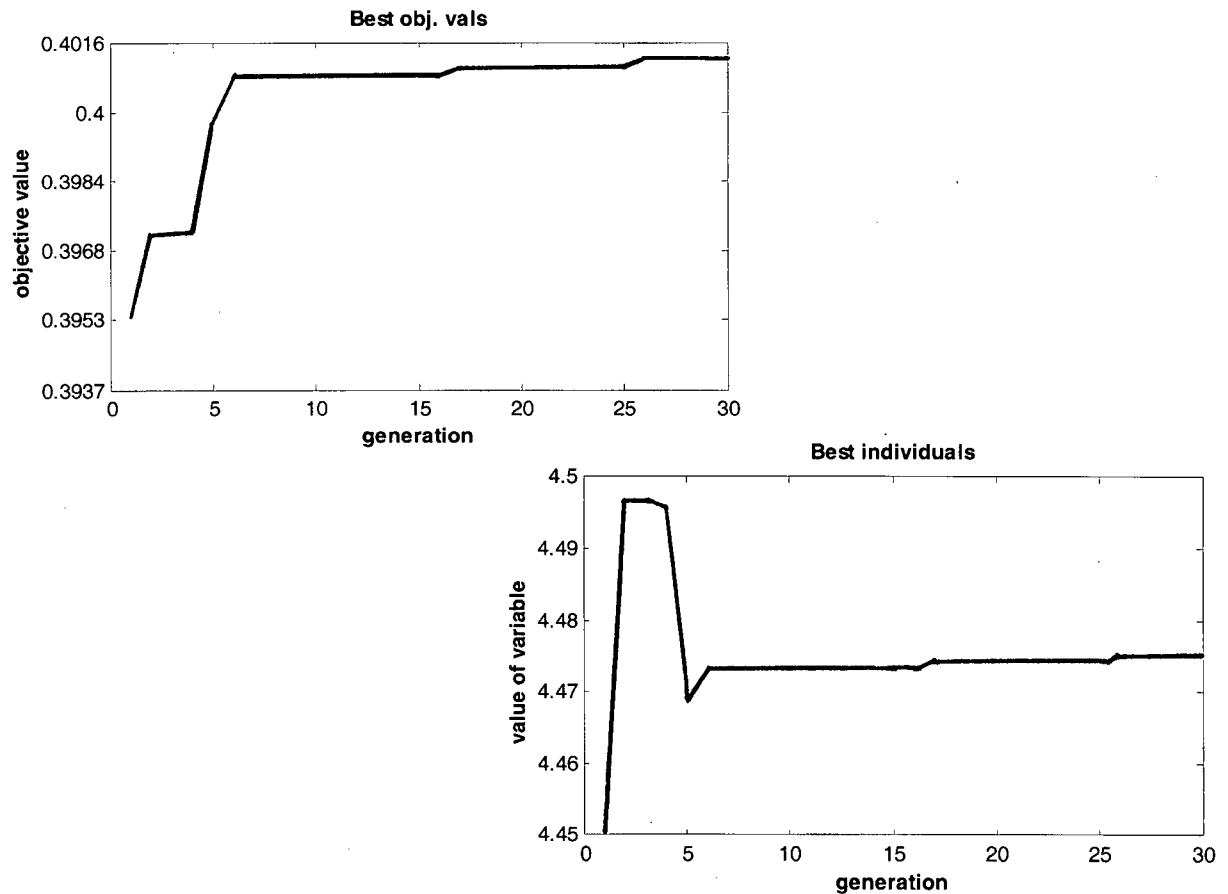


Figure 6.4: The Results of the GA for Local Isotropic Optimization of a 2-DOF Elbow Manipulator

In comparison to the culling algorithm, GA presents a resolution of 4 decimal points for both the parameter and position space. This results in more accurate results. If the culling algorithm runs to provide such a resolution, a huge number of objective function evaluations should take place since the search happens in a position space of 110001 points and a design parameter space of 60001 points.

As it is clear from Figure (6.2), the objective function plot has left/right symmetry for workspace positions. Therefore, future optimizations need to only consider half of the

workspace, $[-5,0]$ or $[0,5]$. This reduces the size of the workspace to 55001 points and results in a faster convergence.

6.2 Design of a Globally Isotropic 2-DOF Planar Manipulator

In this section, the minimax genetic algorithm is considered for a global optimization of the manipulator of Figure (6.1). For this purpose, the performance index is considered as the ratio of the minimum singular value of Jacobian matrix to the maximum one obtained through search in the entire workspace.

For each of the two GAs of the program, 30 generations complete. Each generation in turn includes 30 individuals in each of its 3 subpopulations. The surface and the contour plots of the global objective function are shown in Figure (6.5) and the results obtained from the last generation are shown in table (6.2). As the table indicates, the program converges to an optimal solution of $l_1=5.46$ and the most isotropic performance with $GII = 0.2334$ is obtained as the ratio of singular values of position sets $\{-5,0\}$ or $\{5,0\}$ which include the workspace middle point and the boundaries. It can also be seen from Figure (6.4) that at these sets, the minimum GII value is associated with the dark orange which has the lowest rank in color classification of Figure (6.3). This minimum is the maximum objective value in comparison to all other minimum ones of other link lengths at all workspace positions. Hence, the robot with a design of $l_1=5.46$ has the most isotropic behavior through its global performance in the entire workspace than any other feasible designs.

As the color transitions in both Figures (6.2) and (6.5) indicate, each point demonstrates an approximately similar behavior as its neighbors and the transition is gradual. Therefore, there is not a huge loss of information when discretizing the workspace for optimal search. This is one of the advantages of robot design optimization problem that makes the discrete optimization algorithms such as the culling algorithm to be applicable. The culling algorithm is not used in [49] for global isotropic design optimization of the manipulator of Figure (6.1). However, in case of 4 decimal point resolution, any discrete search algorithm should take place in a position space of 5,000,150,001 points and a parameter space of 60001 points.

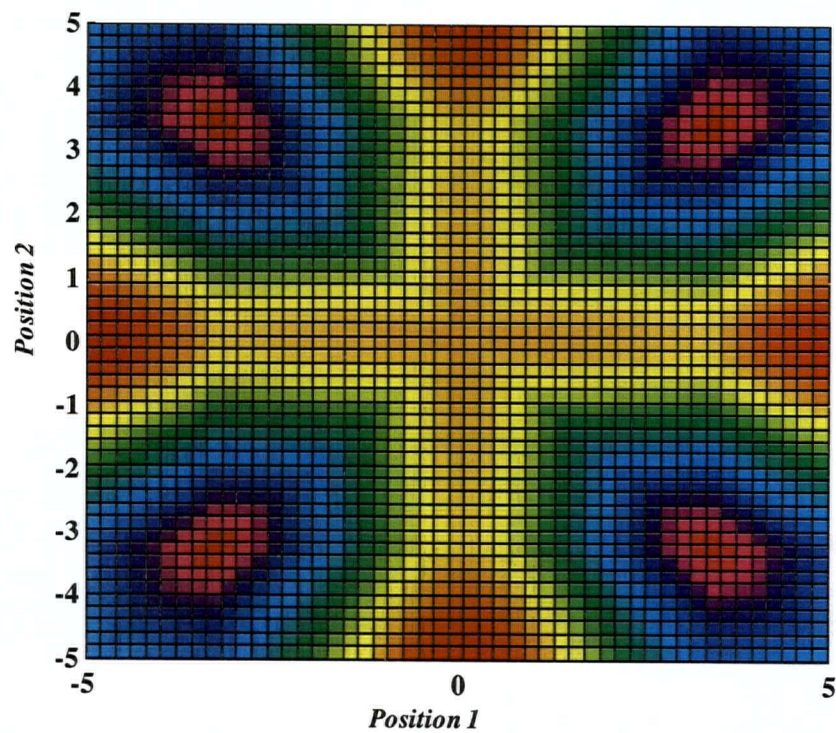
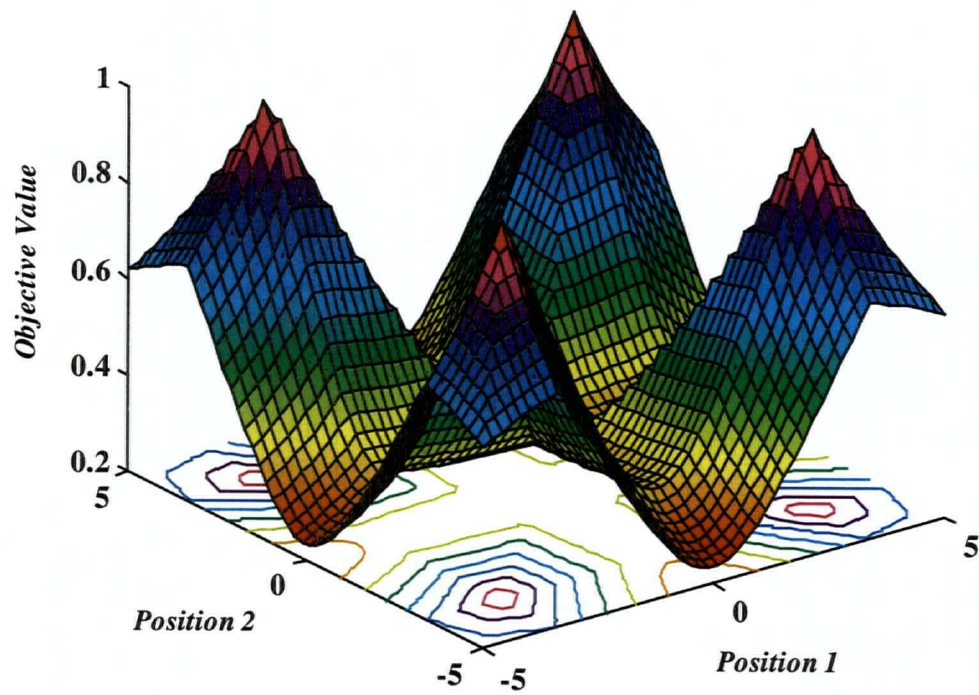


Figure 6.5: Surface and Contour Plots of Variation of the Local Isotropic Measures for a 2-DOF Planar Manipulator with $l_1=5.5$

Link Length	Position x_1	Position x_2	Performance Measure
6.0624	5.0000	0.0034	0.2325
5.4362	5.0000	0.0009	0.2334
5.4432	-0.0022	5.0000	0.2334
6.8130	0.0003	-5.0000	0.2280
6.8594	-5.0000	0.0012	0.2275
5.5384	-5.0000	0.0013	0.2334
5.1238	0.0001	5.0000	0.2331
5.4620	5.0000	0.0008	0.2334
6.6142	-0.0024	5.0000	0.2298
6.8514	-5.0000	-0.0025	0.2276
5.7618	-5.0000	0.0004	0.2332
5.6514	-5.0000	0.0001	0.2333
5.6545	0.0040	-5.0000	0.2333
5.8051	-0.0018	5.0000	0.2331
6.8185	-5.0000	-0.0051	0.2280
6.8091	-0.0041	-5.0000	0.2281
5.5717	-0.0024	-5.0000	0.2333
5.5429	-0.0019	-5.0000	0.2334
4.8722	-0.0001	5.0000	0.2325
5.1722	5.0000	0.0289	0.2332
4.9285	5.0000	-0.0032	0.2326
4.8535	5.0000	-0.0006	0.2324
5.3978	0.0005	5.0000	0.2334
5.2449	5.0000	-0.0008	0.2333
5.4042	-0.0016	5.0000	0.2334
5.3992	-0.0031	5.0000	0.2334
5.2809	-5.0000	-0.0025	0.2333
5.2812	-5.0000	0.0008	0.2333
5.3965	5.0000	0.0021	0.2334
5.3965	-0.0001	-5.0000	0.2334
5.4811	-0.0028	-5.0000	0.2334
5.4811	0.0027	5.0000	0.2334
5.4982	0.0000	5.0000	0.2334
5.8658	0.0006	-5.0000	0.2330
5.5472	0.0002	-5.0000	0.2334
5.4864	-5.0000	-0.0006	0.2334

Table 6.2: The Results of the Last Generation of the GA for Global Isotropic Optimization of an Elbow Manipulator

As Figure (6.5) indicates, the plot of the objective function has left/right and top/bottom symmetry for workspace points. Therefore, a quarter of the workspace would be sufficient for optimization. That is, the global search for minimum and maximum singular values of the entire workspace can be done in range of $[-5,0]$ or $[0,5]$.

The graphs in Figure (6.6) represent the variation the best individual candidates for link length l_1 throughout 30 generations.

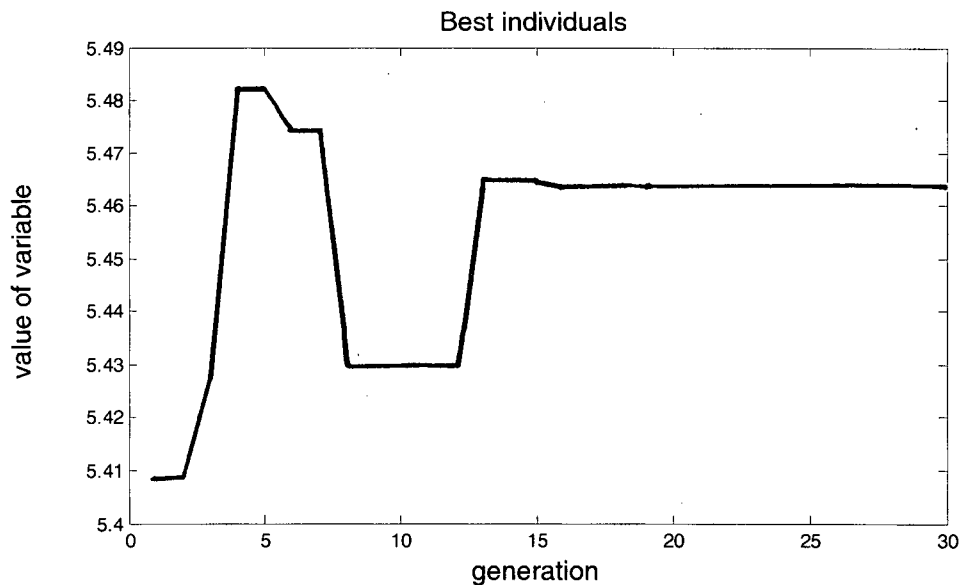


Figure 6.6: The Results of the GA for Global Isotropic Optimization of the Elbow Manipulator

6.3 Design of a 3-DOF Planar Manipulator

In this section, the minimax genetic algorithm is applied to solve for a 3-DOF planar manipulator shown in Figure (6.7).

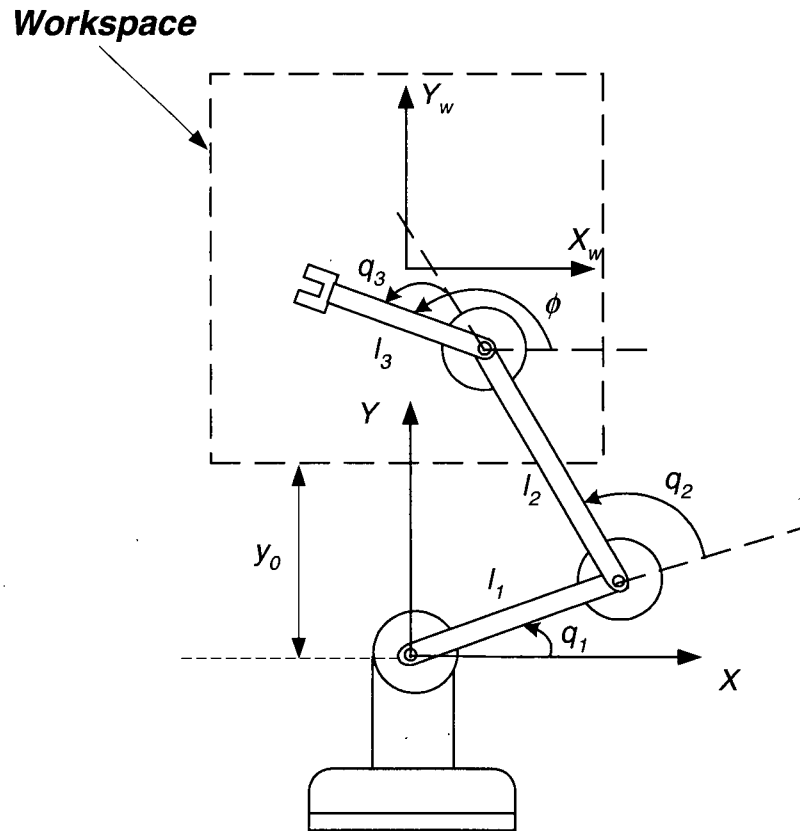


Figure 6.7: A 3-DOF Planar Manipulator

The robot moves in a square workspace ($x \in [-2, 2]$, $y \in [-2, 2]$) that is located at a fixed distance of $y_0 = 2$ cm above the base coordinate frame XY . Physical constraint of $l_i \in [2, 4]$, $i = 1, 2, 3$ is imposed on all link lengths. The program runs with 50 generations for each GA. Each generation includes 50 individuals in each of its 3 subpopulations.

l_1	l_2	l_3	X	Y	ϕ	GII	
4.0000	3.7045	4.0000	-2.0000	2.0000	-1.2489	0.0486	← Best Result
3.6938	3.7106	4.0000	2.0000	2.0000	-1.8925	0.0471	
2.6922	3.6604	3.9781	2.0000	2.0000	-1.8922	0.0428	
3.3956	3.2386	4.0000	-2.0000	2.0000	-1.2489	0.0376	
2.0000	3.4596	4.0000	-2.0000	2.0000	-1.2479	0.0377	
4.0000	3.4984	3.9875	-2.0000	2.0000	-1.2490	0.0450	
3.9499	3.7042	4.0000	-2.0000	2.0000	-1.2490	0.0483	
3.9939	2.6591	3.8052	2.0000	2.0000	-1.8922	0.0302	
4.0000	2.7445	3.7901	2.0000	2.0000	-1.8917	0.0318	
3.8996	3.4981	4.0000	-2.0000	2.0000	-1.2490	0.0444	
3.6000	2.9946	4.0000	-2.0000	2.0000	-1.2490	0.0341	
4.0000	2.7445	3.7901	-2.0000	2.0000	-1.2488	0.0318	
3.8996	3.4981	4.0000	-2.0000	2.0000	-1.2490	0.0444	
3.6000	2.9946	4.0000	-2.0000	2.0000	-1.2490	0.0341	
4.0000	2.7445	3.7901	-2.0000	2.0000	-1.2488	0.0318	
3.9000	2.9729	3.6999	-2.0000	2.0000	-1.2490	0.0358	
4.0000	3.6606	4.0000	-2.0000	2.0000	-1.2490	0.0478	
3.9898	2.8213	3.9737	-2.0000	2.0000	-1.2487	0.0328	
3.9000	2.9950	3.9983	2.0000	2.0000	-1.8924	0.0354	
3.9000	2.9950	4.0000	-2.0000	2.0000	-1.2485	0.0354	
2.8948	3.6312	4.0000	2.0000	2.0000	-1.8922	0.0427	
4.0000	3.4043	4.0000	-2.0000	2.0000	-1.2490	0.0433	
3.9891	2.7803	3.9967	-2.0000	2.0000	-1.2490	0.0320	
4.0000	3.4507	4.0000	-2.0000	2.0000	-1.2490	0.0441	
4.0000	3.2536	3.5871	-2.0000	2.0000	-1.2490	0.0418	
4.0000	3.2661	3.5746	2.0000	2.0000	-1.8921	0.0421	
3.6584	3.6839	4.0000	2.0000	2.0000	-1.8925	0.0465	
2.8780	3.6196	4.0000	2.0000	2.0000	-1.8925	0.0425	
2.8506	3.7596	4.0000	-2.0000	2.0000	-1.2488	0.0449	
4.0000	3.3388	3.9872	-2.0000	2.0000	-1.2489	0.0421	
3.8954	3.2562	4.0000	-2.0000	2.0000	-1.2487	0.0401	
4.0000	3.7595	4.0000	2.0000	-2.0000	-0.1719	0.0471	
2.3136	3.7171	4.0000	2.0000	2.0000	-1.8925	0.0428	
4.0000	3.5205	4.0000	-2.0000	2.0000	-1.2490	0.0453	
4.0000	3.2559	4.0000	-2.0000	2.0000	-1.2490	0.0406	

Table 6.3: The Results of the Last Generation of the GA for Local Isotropic Optimization of a 3-DOF Planar Manipulator

Table (6.3) presents the results of optimizing for the 3-DOF manipulator of Figure (6.6). The optimal result appear to be $l_1=4.0000$, $l_2= 3.7045$, $l_3= 4.0000$, and that configuration has the best isotropic performance at $X= -2.0000$ and $Y= 2.0000$ for $\phi= -1.2489$ rad with a $GII=0.0486$. However, other results shown in table (6.3) will provide a near optimum behavior of robot at their related optimal points.

The graph in Figure (6.8) presents the variation of best individual candidates for link lengths l_1, l_2 and l_3 throughout 40 generations.

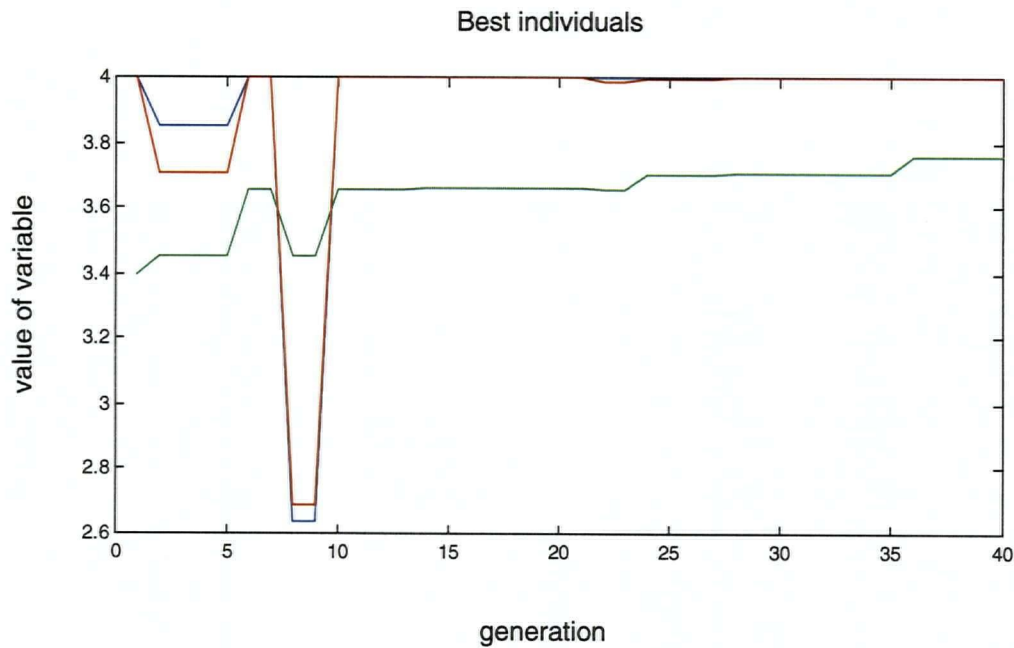


Figure 6.8: The Results of the GA for a 3-DOF Manipulator

In case of the 3-DOF manipulator, all three link lengths and the orientation angle are free design parameters. Using a minimax GA algorithm, the results with a resolution of 4 decimal points for X , Y and ϕ , and link lengths l_1 , l_2 and l_3 are obtained.

Chapter 7

Conclusions

This chapter summarizes the research detailed in this thesis and makes some suggestions for future work in this area.

7.1 Overview of the Research

The main objective of this research work was to identify the power of genetic algorithms in solving minimax problems such as robot design optimization. The objective function of the optimization problem was a highly non-linear, non-differentiable mathematical expression based on the singular values of a Jacobian matrix. The work was completed for two types of robotic manipulators. First, the isotropic design of a 2-DOF planar elbow manipulator was considered where the condition index was a local measure of isotropy. Describing the second link of the robot as a function of the first link and a 1-D workspace in form of a horizontal line, the optimal link lengths of the manipulator were derived as well as the optimal working point at which the robot had the most isotropic performance. The final results were compared to the results of a previous work on the same manipulator applying a culling algorithm and it was concluded that the genetic algorithm seems to converge faster to a solution. Furthermore, it should be noted that the genetic algorithm has a higher resolution since the search need not to be discretized.

The algorithm was also implemented to solve for global optimization of the same 2-DOF manipulator. In this approach, the comparison between the singular values of the Jacobian took place at all positions of the workspace and the optimal design resulted in a robot that would perform “most uniformly” throughout the entire workspace.

In the last step, the optimization was implemented to solve for a 3-DOF planar manipulator performing in a 2-D square workspace with four design variables, three link lengths and an orientation angle.

7.2 Future Work

The algorithm proposed in this work can be applied to all other minimax optimization algorithms. It works for all types of problems with logical, mathematical, continuous, discrete, non-linear, non-differentiable or multi-modal objective functions. The algorithm is not sensitive to the initial conditions or the nature of the objective function

For the specific application in robot design optimization, the work can be promoted by combining the culling algorithms and the genetic algorithms approaches. In this strategy, the computational expense of both algorithms can be greatly reduced and the optimal results can be obtained in a faster convergence with a higher resolution. In this combined algorithm, a generation of random points can be evaluated by a genetic algorithm through the search step of the culling algorithm instead of evaluating all of the points. The combined algorithm benefits the elimination nature of the culling algorithm for reducing the search space and the stochastic nature of the genetic algorithm for reducing the amount of objective function evaluations. The final result is guaranteed to be the same as a global search in a much shorter time.

Bibliography

- [1] J. Angeles, "The Design of Isotropic Manipulator Architectures in the Presence of Redundancies", *The International Journal of Robotics Research*, Vol. 11, No. 3, 196-201, June 1992.
- [2] J. Angeles, C Lopez-Cajun, "Kinematic Isotropy and the Conditioning Index for Serial Robotic Manipulators", *The International Journal of Robotics Research*, Vol. 11, No. 6, 560-571, December 1992.
- [3] J. Angeles, F. Ranjbaran, "On the Design of the Kinematic Structure of Seven-Axes Redundant Manipulators for Maximum Conditioning", *Proceeding of the 1992 IEEE International Conference on Robotics and Automation* (Nice, France), 494-499, May 1992.
- [4] D. Baesley, D. Bull, R. Martin, "An Overview of Genetic Algorithms: Part 1, Fundamentals", *University Computing*, Vol. 15, No. 2, 58-69, 1993.
- [5] A. Bowling, O. Khatib, "Robot Acceleration Capability: The Actuation Efficiency Measure", *Proceeding of the 2000 IEEE International Conference on Robotics and Automation*, Vol. 4, 3970 -3975, 2000.
- [6] A. Bowling, O. Khatib, "The Motion Isotropy Hypersurface: A Characterization of Acceleration Capability", *Proceeding of the 1998 IEEE International Conference on Robotics and Automation*, Vol. 2, 965 -971, 1998.
- [7] K. A. Buckley, S. H. Hopkins, B. C. H. Turton, "Solution of Inverse Kinematics Problems of a Highly Kinematically Redundant Manipulator using Genetic Algorithms", *Second International Conference On Genetic Algorithms in Engineering Systems: Innovations and Applications*, Conf. Pub., No. 446, 264-269, 1997.
- [8] N. Chaiyaratana, A. M. S. Zalzala, "Recent Developments in Evolutionary and Genetic Algorithms: Theory and Applications", *Second International Conference On Genetic Algorithms in Engineering Systems: Innovations and Applications*, Conf. Pub., No. 446, 270-277, 1997.
- [9] O. Chocron, P. Bidaud, "Evolutionary Algorithms in Kinematic Design of Robotic Systems" *Proceeding of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, 1111 -1117, 1997.
- [10] A. J. Chipperfield, P. J. Fleming, C. M. Fonseca, "Genetic Algorithm Tools for Control Systems Engineering", *Adaptive Computing in Engineering Design and Control*, Plymouth, UK, September 1994.

- [11] A. J. Chipperfield, P. J. Fleming, "The MATLAB Genetic algorithm Toolbox", *IEE Colloquium on Applied Control Techniques using MATLAB*, Digest No. 1995/014, 1995.
- [12] J. J. Craig, "Introduction to Robotics: Mechanics and Control" *Addison-Wesley Publishing Company, Inc.*, 1989.
- [13] K. Doel, D. K. Pai, "Performance Measures for Robot Manipulators: A Unified Approach", *The International Journal of Robotics Research*, Vol. 15, No. 1, 92-111, February 1996.
- [14] K.L. Doty, "Robot Manipulability", *Proceeding of the 2000 IEEE International Conference on Robotics and Automation*, Vol. 11, Issue 3, 462-468, June 1995.
- [15] K.L. Doty, C. Melchirri, C. Bonivento, "A Theory of Generalized Inverse Applied to Robotics", *The international Journal of Robotics Research*, Vol. 12, No. 1, 1-20, October 1998.
- [16] F. Gao, X. Liu, W. A. Gruver, "Performance Evaluation of Two Degree of Freedom Planar Parallel Robots", *Mech. Machine Theory*, Vol. 33, No. 6, 661-668, 1998.
- [17] F. Gao, X. Liu, W. A. Gruver, "The Global Conditioning Index in the Solution Space of Two Degree of Freedom Planar Parallel Manipulators", *Proceedings of the 1995 IEEE Conference on SMC*, Vol. 5, 4055-4059, 1995.
- [18] F. Gao, W. A. Gruver, "Performance Evaluation Criteria for Analysis and Design of Robotic Specimens", *Proceedings of the 8th International Conference on Advanced Robotics*, 879-884, 1997
- [19] G. M. Griner, "A Comparison of Simulated Evolution and Genetic Evolution Performance", *Proceedings of the First IEEE Conference on Computational Intelligence*, Vol. 1, 374 -378, 1994.
- [20] R. Grupen, K. Souccar, "Manipulability-Based Spatial Isotropy: A Kinematic Reflex", *International Workshop on Mechanical Computer Systems for Perception and Action* 1993.
- [21] D. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", *Addison-Wesley Publishing Company, Inc.*, 1989.
- [22] C. Gosselin, "Dexterity Indices for Planar and Spatial Robotic Manipulators", *Proceeding of IEEE International Conference on Robotics and Automation* (Cincinnati, Ohio), 650-655, May 1990.
- [23] C. Gosselin, J. Angeles, "A Global Performance Index for the Kinematic Optimization of Robot Manipulators", *Transaction of the ASME, Journal of Mechanical Design*, Vol. 113, 220-226, 1991.

- [24] J. Han, W. K. Chung, Y. Youm, S. h. Kim, "Task Based Design of Modular Robot Manipulator using Efficient Genetic Algorithm", *Proceeding of the 1997 IEEE International Conference on Robotics and Automation*, Vol.1, 507 –512, April 1997.
- [25] C. R. Houk, J. Joines, M. G. Kay, "A Genetic Algorithm for Function Optimization: A Matlab Implementation",
- [26] I. Jeong, J. Lee, "Adaptive Simulated Annealing Genetic Algorithm for System Identification", *Engng. Applic. Artif. Intel.*, Vol. 6, No. 5, 523-532, 1996.
- [27] M. Kircanski, "Inverse Kinematic Problem Near Singularities for Simple Manipulators: Symbolical Damped Least-Square Solution", *Proceeding of the 1993 IEEE International Conference on Robotics and Automation*, Vol.1, 974 - 979, May 1993.
- [28] M. Kircanski, "Symbolic Singular Value Decomposition for Simple Redundant Manipulator and Its Application to Robot Control", *The International Journal of Robotics Research*, Vol. 14, No. 4, 382-398, August 1995.
- [29] M. Kircanski, "Kinematic Isotropy and Optimal Kinematic Design of Planar Manipulators and a 3-DOF Spatial Manipulator", *The international Journal of Robotics Research*, Vol. 15, No. 1, 61-77, February 1996.
- [30] M. Kircanski, "Robotic Isotropy and Optimal Robot Design of Planar Manipulators", *Proceeding of the 1994 IEEE International Conference on Robotics and Automation*, Vol. 2, 1100-1105, 1998.
- [31] M. Kircanski, M. Boric, "Symbolic Singular Value Decomposition for a PUMA Robot and Its Application to a Robot Operation Near Singularities", *The International Journal of Robotics Research*, Vol. 12, No. 5, 460-472, October 1993.
- [32] J. Kim, P.Khosla, "Dexterity Measures for Design and Control of Manipulators", *IEEE/RSJ International Workshop on Intelligent Robots and Systems* (Osaka, Japan), 758-763, November 1991.
- [33] J. Kim, P. Khosla, "A Multi-population Genetic Algorithm and Its Application to Design of Manipulators", *Proceeding of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Raleigh, NC), 279-268, July 1992.
- [34] C. Klein, B. Blaho, "Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators", *The International Journal of Robotics Research*, Vol. 6, No. 2, 72-83, Summer 1987.
- [35] C. Klein, T. Miklos, "Spatial Robot Isotropy", *The International Journal of Robotics Research*, Vol. 10, No. 4, 426-437, August 1991.

- [36] S. Lee, J. M. Lee, "Multiple Task Point Control of a Redundant Manipulator", *Proceeding of the 199 IEEE International Conference on Robotics and Automation*, Vol. 2, 988 -993, 1990.
- [37] D. P. Kwok, F. Sheng, "Genetic Algorithms and Simulated Annealing for Optimal Robot Arm PID Control", *Proceedings of the First IEEE Conference on Evolutionary Computation*, Vol. 2, 707 -713, 1994.
- [38] W. Lee "An Evolutionary System for Automatic Robot Design", *Proceeding of the 2000 IEEE International Conference on Robotics and Automation*, Vol. 4, 3477 -3482, 1998.
- [39] O. Ma, J. Angeles "The Concept of Dynamic Isotropy and Its Application to Inverse Kinematic and Trajectory Planning", *Proceeding of IEEE International Conference on Robotics and Automation*, Vol.1, 481 -486, 1990.
- [40] O. Ma, J. Angeles, "Optimum Design of Manipulators Under Dynamic Isotropy Conditions", *Proceeding of the 1993 IEEE International Conference on Robotics and Automation*, Vol. 1, 470-475, 2-6 May 1993.
- [41] A. Maciejewski, C. Klein, "The Singular Value Decomposition: Computation and Applications to Robotics", *The International Journal of Robotics Research*, Vol. 8, No. 6, 63-79, December 1989.
- [42] Q.C. Meng, T.J. Feng, Z. Chen, C.J Zhou, J. H. BO, "Genetic Algorithms Encoding Study and A Sufficient Convergence Condition of GAS", *IEEE SMC '99 Conference Proceedings*, Vol. 1, 649- 652, 1999.
- [43] C. J. J. Paredis, P. K. Khosla, "Kinematic Design of Serial Link Manipulators From Task Specifications", *The international Journal of Robotics Research*, Vol. 12, No. 3, 274-287, 1993.
- [44] H. Pohlheim, "Tutorial, Genetic and Evolutionary Algorithm Toolbox for use with MATLAB", *Documentation for Genetic and Evolutionary Algorithm Toolbox for use with MATLAB*, Version 1.92, Documentation 1.92 a, October 1999.
- [45] H. Pohlheim, "Evolutionary Algorithms: Overview, Methods and Operators", *Documentation for Genetic and Evolutionary Algorithm Toolbox for use with MATLAB*, Version 1.92, Documentation 1.92 a, December 1999.
- [46] S. Saha, J. Angeles, J. Darcovich, "The Kinematic Design of a 3-dof Isotropic Mobile Robot", *Proceeding of the 1993 IEEE International Conference on Robotics and Automation*, Vol. 1, 283 - 288, May 1993.

- [47] L. Stocco, S. E. Salcudean, F. Sassani, "Mechanical Design for Global Isotropy with Applications to Haptic interface", *Sixth Annual Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Dallas, Texas, November 1997.
- [48] L. Stocco, S. E. Salcudean, F. Sassani, "Matrix Normalization for Optimal Robot Design", *Proceeding of the 1998 IEEE International Conference on Robotics and Automation*, Vol. 2, 1346- 1351, May 1998.
- [49] L. Stocco, "Robot Design Optimization with Haptic Interface applications", *Ph.D Thesis, Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC*, October 1999.
- [50] L. Stocco, S. E. Salcudean, F. Sassani, "On the Use of Scaling Matrices for Task-Specific Robot Design" *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 5, October 1999.
- [51] L. Stocco, S. E. Salcudean, F. Sassani, "Isotropy and Actuator Optimization in Haptic Interface Design", *Proceeding of the 2000 IEEE International Conference on Robotics and Automation*, Vol. 1, 763- 769, April 2000.
- [52] C. Tischler, K. Hunt, A. Samuel, "On Optimizing the Kinematic Geometry of a Dexterous Robot Finger", *The International Journal of Robotics Research*, Vol. 17, No. 10, 1055-1067, October 1998.
- [53] V. Tourassis, M. Ang, " Identification and Analysis of Robot Manipulator Singularities", *The International Journal of Robotics Research*, Vol. 11, No. 3, 248-259, June 1992.
- [54] T. Yoshikawa, "Analysis and Control of Robot Manipulators with Redundancy", *Preprints First International Symposium on Robotics Research* (Bretton woods, N.H), 735-747, August-September 1983.
- [55] T. Yoshikawa, "Manipulability of Robotic Mechanisms", *The International Journal of Robotics Research*, Vol. 4, No. 2, 3-9, Summer 1985.
- [56] K. E. Zanganeh, J. Angeles, "Kinematic Isotropy and the Optimum Design of Parallel Manipulators", *The International Journal of Robotics Research*, Vol. 16, No. 2, 185-197, 1997.

Appendix 1

Inverse kinematics of a 3-DOF RRR Serial Manipulator

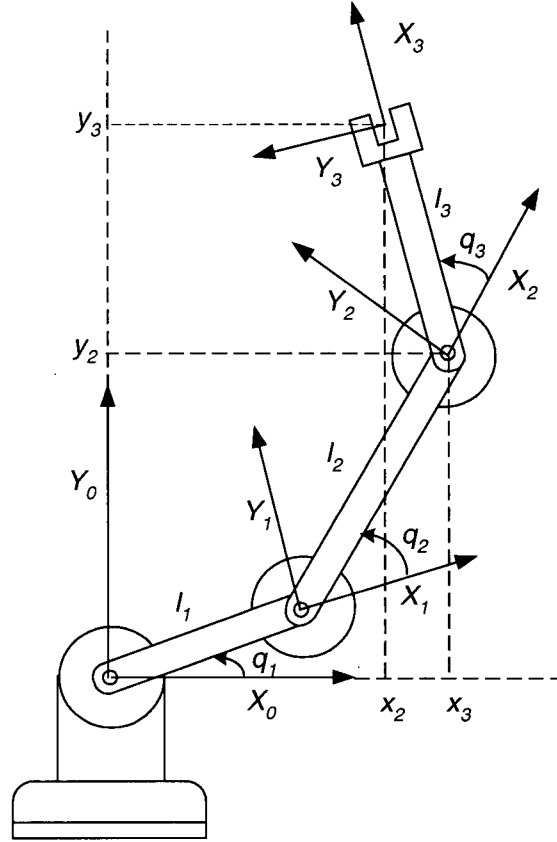


Figure A.1: A 3-DOF RRR Serial Manipulator

Consider the planar 3-DOF serial manipulator shown in Figure A.1. It is desired to find the joint variables q_1 , q_2 and q_3 corresponding to a given end-effector position and orientation. We define two coordinate x_3 , y_3 for the origin of wrist frame and the angle ϕ which is in fact the orientation angle of link 3 in the plane. Having specified that orientation, the following relation exists

$$\phi = q_1 + q_2 + q_3 \quad (\text{A.1})$$

The following relations can be easily derived from the figure.

$$x_2 = x_3 - l_3 c_\phi = l_1 c_1 + l_2 c_{12} \quad (\text{A.2})$$

$$y_2 = y_3 - l_3 s_\phi = l_1 s_1 + l_2 s_{12} \quad (\text{A.3})$$

Which describes the position of the origin of frame 2. This depends only on two angles q_1 and q_2 where

$$c_{12} = \cos(q_1 + q_2) = c_1 c_2 - s_1 s_2 \quad (\text{A.4})$$

$$s_{12} = \sin(q_1 + q_2) = c_1 s_2 + s_1 c_2 \quad (\text{A.5})$$

Squaring and summing Eqs. (A.2) and (A.3) and using relations (A.4) and (A.5) Yields:

$$x_2^2 + y_2^2 = l_1^2 + l_2^2 + 2l_1 l_2 c_2 \quad (\text{A.6})$$

Solving (A.6) for c_2 , we get

$$c_2 = \frac{x_2^2 + y_2^2 - l_1^2 - l_2^2}{2l_1 l_2} \quad (\text{A.7})$$

The existence of solution obviously imposes that $-1 \leq c_2 \leq 1$, otherwise the given point would be outside the arm reachable workspace. Assuming the point is inside the workspace, we can write expression (A.8) for s_2

$$s_2 = \pm \sqrt{1 - c_2^2} \quad (\text{A.8})$$

Where the positive sign is relative to the elbow-down posture and the negative sign to the elbow-up posture. Therefore, the choice of sign corresponds to multiple solutions. In solving for q_2 , both sine and cosine of the desired joint angle are determined to ensure that all solutions are found and the solved angle is in the proper quadrant. Hence applying the two-argument arctangent, the angle q_2 can be computed as

$$q_2 = \text{Atan2}(s_2, c_2) \quad (\text{A.9})$$

Substituting q_2 into (A.2) and (A.3) yields an algebraic system of two equations in the two unknowns s_1 and c_1 whose solution is

$$s_1 = \frac{(l_1 + l_2 c_2)y_2 - l_2 s_2 x_2}{x_2^2 + y_2^2} \quad (\text{A.10})$$

$$c_1 = \frac{(l_1 + l_2 c_2)x_2 + l_2 s_2 y_2}{x_2^2 + y_2^2} \quad (\text{A.11})$$

In analogy to the above, q_1 can be found

$$q_1 = \text{Atan2}(s_1, c_1) \quad (\text{A.12})$$

Finally the angle q_3 is found from (A.1) as

$$q_3 = \phi - q_1 - q_2 \quad (\text{A.13})$$

If ϕ is not specified, the arm is redundant and there exists infinite solutions to the inverse kinematics problem.

Appendix 2

Jacobian Matrix of a 2-DOF RR Serial Manipulator

The Jacobian of a 2-DOF serial manipulator in figure A.2 is written in frame $\{0\}$ in (A.14) with the origin and frame considerations shown in Fig A.2 where all joint angles are set to zero [50].

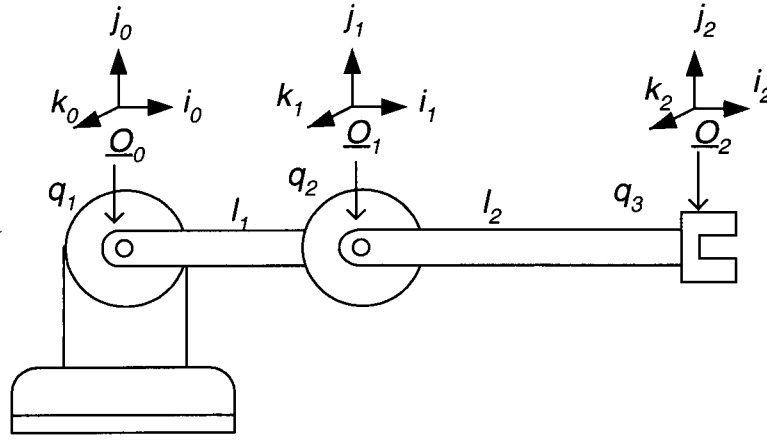


Figure A.2: A 2-DOF Robot

$${}^0J = \begin{bmatrix} j^T(\underline{O}_0 - \underline{O}_2) & j^T(\underline{O}_1 - \underline{O}_2) \\ i^T(\underline{O}_2 - \underline{O}_0) & i^T(\underline{O}_2 - \underline{O}_1) \end{bmatrix} \quad (\text{A.14})$$

where

$$j^T(\underline{O}_0 - \underline{O}_2) = -l_1 \sin(q_1) - l_2 \sin(q_1 + q_2) \quad (\text{A.15})$$

$$j^T(\underline{O}_1 - \underline{O}_2) = -l_2 \sin(q_1 + q_2) \quad (\text{A.16})$$

$$i^T(\underline{O}_2 - \underline{O}_0) = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \quad (\text{A.17})$$

$$i^T(\underline{O}_2 - \underline{O}_1) = l_2 \cos(q_1 + q_2) \quad (\text{A.17})$$

Therefore,

$${}^0J = \begin{bmatrix} -l_1 \sin(q_1) - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) \\ l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) & l_2 \cos(q_1 + q_2) \end{bmatrix} \quad (\text{A.18})$$

The Jacobian written in frame $\{3\}$ can be obtained as

$${}^3J = \begin{bmatrix} l_1 \sin(q_2) & 0 \\ l_1 \cos(q_2) + l_2 & l_2 \end{bmatrix} \quad (\text{A.19})$$

Appendix 3

Jacobian Matrix of a 3-DOF RRR Serial Manipulator

The Jacobian of the planar serial manipulator in figure A.3 is computed from (A.20) using the terms defined in A.21- A.26 and the origin and frame assessment shown in Fig A.3, where all joint angles are set to zero [50].

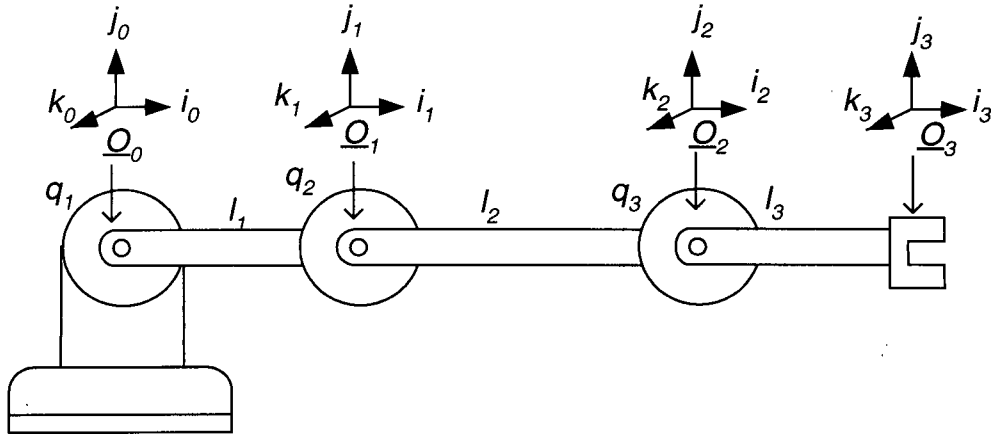


Figure A.3: A 3-DOF Robot

$${}^0j = \begin{bmatrix} j^T(\underline{O}_0 - \underline{O}_3) & j^T(\underline{O}_1 - \underline{O}_3) & j^T(\underline{O}_2 - \underline{O}_3) \\ i^T(\underline{O}_3 - \underline{O}_0) & i^T(\underline{O}_3 - \underline{O}_1) & i^T(\underline{O}_3 - \underline{O}_2) \\ 1 & 1 & 1 \end{bmatrix} \quad (\text{A.20})$$

where

$$j^T(\underline{O}_0 - \underline{O}_3) = -l_1 \sin(q_1) - l_2 \sin(q_1 + q_2) - l_3 \sin(q_1 + q_2 + q_3) \quad (\text{A.21})$$

$$j^T(\underline{O}_1 - \underline{O}_3) = -l_2 \sin(q_1 + q_2) - l_3 \sin(q_1 + q_2 + q_3) \quad (\text{A.22})$$

$$j^T(\underline{O}_2 - \underline{O}_3) = -l_3 \sin(q_1 + q_2 + q_3) \quad (\text{A.23})$$

$$i^T(\underline{O}_3 - \underline{O}_0) = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) + l_3 \cos(q_1 + q_2 + q_3) \quad (\text{A.24})$$

$$i^T(\underline{O}_3 - \underline{O}_1) = l_2 \cos(q_1 + q_2) + l_3 \cos(q_1 + q_2 + q_3) \quad (\text{A.25})$$

$$i^T(\underline{Q}_3 - \underline{Q}_2) = l_3 \cos(q_1 + q_2 + q_3) \quad (\text{A.26})$$

The Jacobian written in frame $\{3\}$ can be obtained as [39].

$${}^3_j = \begin{bmatrix} l_2 s_3 + l_1 s_{23} & l_2 s_3 & 0 \\ l_3 + l_2 c_3 + l_1 c_{23} & l_3 + l_2 c_3 & l_3 \end{bmatrix} \quad (\text{A.27})$$

Appendix 4

Singular Values of Jacobian Matrix for a 2-DOF and 3-DOF Planar Manipulator

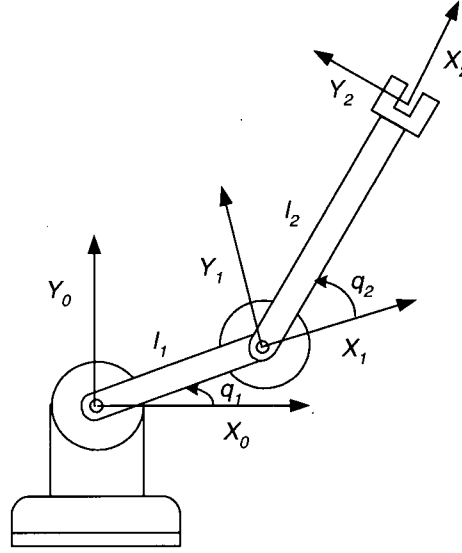


Figure A.4: A 2-DOF Planar Manipulator

Consider the nonredundant 2-DOF planar manipulator shown in Fig A.4. Since the singular values of Jacobian matrix are not dependent on coordinate frame in which Jacobian is expressed, we consider the Jacobian matrix that corresponds to task coordinates expressed in the coordinate frame which is located at the manipulator tip. That makes the elements of the Jacobian simpler than any other frames [29].

$$J = {}^2J = \begin{bmatrix} l_1 s_2 & 0 \\ l_2 + l_1 c_2 & l_2 \end{bmatrix} \quad (\text{A.28})$$

Where l_1 and l_2 are the lengths of the links 1 and 2. , $s_2 = \sin q_2$ and $c_2 = \cos q_2$ and q_1 and q_2 are joint angles.

To use the relationship between eigenvalues of $J^T J$ and singular values of J , we consider the following matrix

$$J^T J = \begin{bmatrix} l_3^2 & l_2(l_2 + l_1 c_2) \\ l_2(l_2 + l_1 c_2) & l_2^2 \end{bmatrix} \quad (\text{A.29})$$

Where l_3 is the length between the manipulator base and the tip.

$$l_3^2 = l_1^2 + l_2^2 + 2l_1 l_2 c_2 \quad (\text{A.30})$$

Considering the relations:

$$\det(J) = \pm \sigma_1 \cdot \sigma_2 \dots \sigma_m \quad (\text{A.31})$$

$$\text{trace}(J^T J) = \sigma_1^2 + \sigma_2^2 + \dots + \sigma_m^2 \quad (\text{A.32})$$

We obtain:

$$\sigma_1^2 + \sigma_2^2 = l_2^2 + l_3^2 \quad (\text{A.33})$$

$$\det(J) = l_1 l_2 s_2 = \sigma_1 \cdot \sigma_2 \det(UV) \quad (\text{A.34})$$

$$\sigma_1 \cdot \sigma_2 = l_1 l_2 |s_2| \quad (\text{A.35})$$

$$\det(UV) = \text{sgn}(s_2) \quad (\text{A.36})$$

$\sigma_1 \cdot \sigma_2 \geq 0$. Here $\text{sgn}()$ denotes the signum function.

The eigenvalues of $J^T J$ can be easily derived as the roots of the characteristic polynomial $\det(J^T J - \lambda I) = 0$. The square root of these eigenvalues (i.e., singular values of J) thus become

$$\sigma_{1,2} = \sqrt{\frac{l_2^2 + l_3^2 \pm \sqrt{(l_2^2 + l_3^2)^2 - 4l_1^2 l_2^2 s_2^2}}{2}} \quad (\text{A.37})$$

substituting l_3 in (1.36) results in

$$\sigma_{1,2} = \sqrt{\frac{l_1^2 + 2l_2^2 + 2l_1 l_2 c_2 \pm \sqrt{(l_1^2 + 2l_2^2 + 2l_1 l_2 c_2)^2 - 4l_1^2 l_2^2 s_2^2}}{2}} \quad (\text{A.38})$$

For a 3-DOF manipulator, considering the Jacobian in second frame in (A.38) is considered.

$${}^3j = \begin{bmatrix} l_2 s_3 + l_1 s_{23} & l_2 s_3 & 0 \\ l_3 + l_2 c_3 + l_1 c_{23} & l_3 + l_2 c_3 & l_3 \end{bmatrix} = \begin{bmatrix} j_{11} & j_{12} & 0 \\ j_{21} & j_{22} & j_{23} \end{bmatrix} \quad (\text{A.39})$$

The singular values of the above Jacobian are derived as shown in (A.40).

$$\sigma_{1,2} = \sqrt{\frac{h_1^2 + h_2^2 \pm \sqrt{(h_1^2 + h_2^2)^2 - 4(h_1^2 h_2^2 - h_{12}^2)}}{2}} \quad (\text{A.40})$$

Where

$$\begin{aligned} h_1^2 &= j_{11}^2 + j_{12}^2 \\ h_2^2 &= j_{21}^2 + j_{22}^2 + j_{23}^2 \\ h_{12} &= j_{11} j_{21} + j_{12} j_{22} \end{aligned} \quad (\text{A.41})$$