OPTIMAL REDUNDANCY RESOLUTION FOR KINEMATICALLY REDUNDANT PARALLEL MANIPULATORS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY
TANSEL SITKI TUNÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

SEPTEMBER 2014

Approval of the thesis:

OPTIMAL REDUNDANCY RESOLUTION FOR KINEMATICALLY REDUNDANT PARALLEL MANIPULATORS

submitted by **TANSEL SITKI TUNÇ** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen ____________________
Dean,Graduate School of **Natural and Applied Sciences**

Prof. Dr. Süha Oral ____________________
Head of Department, **Mechanical Engineering**

Prof. Dr. Kemal Özgören ____________________
Supervisor, **Mechanical Engineering Dept.,METU**

Examining Committee Members:

Assist. Prof. Dr. Buğra Koku ____________________
Mechanical Engineering Dept., METU

Prof. Dr. Kemal Özgören ____________________
Mechanical Engineering Dept., METU

Prof. Dr. Kemal İder ____________________
Mechanical Engineering Dept., Cankaya University

Prof. Dr. Eres Söylemez ____________________
Mechanical Engineering Dept., METU

Assist. Prof. Dr. Kıvanç Azgın ____________________
Mechanical Engineering Dept., METU

Date: 03.09.2014

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Tansel Sıtkı Tunç

Signature :

ABSTRACT

OPTIMAL REDUNDANCY RESOLUTION FOR KINEMATICALLY REDUNDANT PARALLEL MANIPULATORS

Tunç, Tansel Sıtkı
M.S.,Department of Mechanical Engineering
Supervisor: Prof. Dr. Kemal Özgören

September 2014, 120 pages

In this study, the redundancy resolution of kinematically redundant parallel manipulators has been investigated as an optimization problem. The emerging optimization problem has been solved globally using a hybrid genetic algorithm. This algorithm has been applied as an example to a planar parallel manipulator which has four degrees of freedom. It has been assumed that the manipulator is used so that only the tip point of its end-effector is controlled. Therefore, the rotation angle of the end effector has been let free. As a result, the redundancy degree of the manipulator has become two for the planar point positioning task which requires two degrees of freedom. In the definiton of the optimiziation problem, the limits of the prismatic joints have acted as inequality constraints and the kinematic relationships, which consist of the loop closure and input-output equations between the tip point position and the joint variables, have acted as equality constraints. It has been assumed that the revolute joints have no limit. Various performance functions such as potential energy, kinetic energy and total power have been used for the purpose of optimization. By minimizing each function separately, different optimal redunancy resolutions have been obtained at the position, velocity and acceleration levels.

Keywords: Paralel manipulators, kinemtic redundancy, optimal redundancy resolution, hybrid genetic algorithm

ÖZ

KİNEMATİKÇE ARTIKSIL PARALEL MANİPÜLATÖRLER İÇİN EN İYİ ARTIKSILLIK ÇÖZÜMLEMESİ

Tunç,Tansel
Yüksek Lisans, Makine Mühendisliği Bölümü
Tez Yöneticisi: Prof.Dr.Kemal Özgören

Eylül 2014, 120 sayfa

Bu çalışmada kinematikçe artıksıl olan paralel manipülatörlerin artıksıllık çözümlemesi bir eniyileştirme problemi olarak ele alınmıştır. Ortaya çıkan eniyileştirme problemi evrensel olarak melez genetik algoritma kullanılarak çözülmüştür. Bu algoritma örnek olarak dört serbestlik dereceli bir düzlemsel paralel manipülatöre uygulanmıştır. Bu manipülatörün yalnızca işlem aygıtının uç noktasının haraketini kontrol etmek üzere kullanılacağı varsayılmıştır. Dolayısıyla, işlem aygıtının yönelim açısı da serbest bırakılmıştır. Böylece, iki serbestlik derecesi gerektiren düzlemsel nokta konumlama işi için manipülatörün artıksıllık derecesi iki olmuştur.Eniyileştirme probleminin tanımında manipülatörün kayareklemlerinin limitleri eşitsizlik kısıtlamaları, döngü kapanım denklemleri ve uç nokta ile eklem değişkenleri arasındaki girdi-çıktı ilişkilerinden oluşan kinematik denklemler ise eşitlik kısıtlamaları olarak rol almışlardır. Döner eklemlerin limitsiz olduğu varsayılmıştır. Eniyileştirme amacıyla, potansiyel enerji, kinetik enerji ve toplam güç gibi değişik başarım işlevleri kullanılmıştır. Her bir işlev ayrı ayrı enküçültülerek konum, hız ve ivme düzeylerinde değişik amaçlı en iyi artıksıllık çözümlemeleri elde edilmiştir.

Anahtar Kelimeler: Paralel manipülatörler, kinematik artıksıllık, en iyi artıksıllık çözümlemesi, melez genetik algoritma.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

A kinematically redundant manipulator is a mechanism which has more degrees of freedom than the required degrees of freedom for its end-effector. The inverse kinematic problem of a redundant manipulator has an infinite number of solutions. Redundancy resolution methods handle the multiple solutions of the inverse kinematic problem of a redundant manipulator and find the best solution according to a desired criterion.

Kinematically redundant manipulators have been used so far in many studies presented in the literature. Kinematic redundancy has been used for purposes such as avoiding obstacles, joint limits and singularities as well as for various optimization purposes [4]. The resolution of kinematic redundancy has been done at the velocity level so far. In such a resolution, joint velocities are first determined; later joint positions are found by integration. Joint velocities are usually found by taking the generalized pseudoinverse of the Jacobian matrix so as to minimize a quadratic cost function of the joint velocities. [4]Another method is the augmented Jacobian matrix method. To be able to apply this method as many additional tasks are defined as the degree of redundancy of the manipulator. Additional tasks can also be defined as part time tasks. With the help of the additional tasks redundancy is removed and the joint velocities are easily determined for the nonsingular positions of the manipulator by taking the inverse of the augmented jacobian matrix.[4]

Redundancy resolutions at the velocity level have some disadvantages. First of all, since the joint positions are found by integration after finding the joint velocities, there is an accumulation of error. Secondly, in a position keeping task, i.e., if there is a no-motion requirement, the joint variables cannot be found by integration for the required position of the end-effector.

For such a task, it is necessary to do redundancy resolution directly in the position level. Though rarely encountered, redundancy resolution at the position level has also been done in some sources in the literature. For example in [10] a redundancy resolution method has been employed in which Lagrange multipliers are used. In this method a cost function has been chosen depending on the desired criterion and the kinematic equations have served as the equality constraints. In the solution process the Lagrange multipliers have first been eliminated. Then, the resulting nonlinear equations have been solved numerically and the joint variables have thus been found. In applying this solution method as in [10], the joint limits have been ignored.

Parallel manipulators which are the subject of this study are mechanisms which have one or more kinematic loops, i.e., closed kinematic chains [1]. In applications where high structural stiffness and high positional accuracy are important, parallel manipulators offer advantages over serial ones. However parallel manipulators have smaller workspaces, lower dexterities, i.e., lower ability to orient their end-effectors within their workspaces, and more complicated mathematical descriptions. To overcome these disadvantages redundant parallel manipulators have been introduced. Redundancy in parallel manipulators is used to remove some of the singularities, enlarge the workspace to some extend, and improve dexterity as well [2].

Kinematically redundant parallel manipulators have been the subject of some studies seen in the literature. In [7] a manipulator with $3-PR^3$ joint structure which has three redundant degrees of freedom has been studied. In [8] a redundant parallel manipulator has been studied which itself has nine degrees of freedom and whose end-effector has six degrees of freedom. In [2], three different manipulators with one redundant degree of freedom have been studied. One of these three manipulators has 3-RPR joint structure and four degrees of freedom. One other is a spherical mechanism which has four degrees of freedom again. The last one is a spatial mechanism which has seven degrees of freedom.

So far redundancy resolution methods have not been employed very frequently for redundant parallel manipulators. A redundancy resolution method for redundant parallel manipulators has been found in [9]. In this source, the redundancy resolution of a parallel manipulator with $3 - RPR^2$ joint structure has been done via a local optimization algorithm to avoid singularities. However the details of the employed algorithm have not been discussed.

In this study, redundancy resolution has been realized by employing a hybrid genetic algorithm which yields a global result. The key feature of the redundancy resolution method employed in this study is that it can be used for the redundacy resolution of parallel manipulators at the position level. Of course, it can be used at the velocity and acceleration levels, too. Another significant feature of this study is the particular hybrid genetic algorithm. The usage of this algorithm has been shown with the help of a planar redundant parallel manipulator for different cost functions such as potential energy, kinetic energy , and total power. These cost functions are the expressions of three typical criteria that require redundancy resolution respectively in the position, velocity, and acceleration levels. Here, the method has been demonstrated by means of a planar example for the sake of simplicity. However, the same method can similarly be applied to a spatial redundant parallel manipulator, too.

CHAPTER 2

PARALLEL MANIPULATORS

In this chapter open and closed kinematic chains, classification of robot manipulators, parallel manipulators and their classification are discussed.

2.1Mechanisms and Manipulators with Open and Closed Kinematic Chains

"A kinematic chain is an assemblage of links that are connected by joints. When every link in a kinematic chain is connected to every other link by at least two distinct paths the kinematic chain forms one or more closed loops and is called a closed-loop chain."[3] "If every link is connected to every other link by one and only one path the kinematic chain is called an open-loop chain."[3] A hybrid kinematic chain is made up of both closed and open loop chains.[3] "A kinematic chain is called a mechanism when one of its links is fixed to the ground."[3] The fixed link is generally called the base. [3] "A manipulator, on the other hand, is a mechanism that , grasps and moves objects with a number of degrees of freedom."[17]

Open and closed loop kinematic chains can also be classified based on the concept of the connection degree. "Connection degree is the number of rigid bodies attached to a link in a mechanism by a joint."[1] "Simple kinematic chains are those in which each member possesses a connection degree that is less than or equal to 2."[1] "Serial manipulators are simple kinematic chains for which all the connection degrees are 2".[1] Of course, the base and the end-effector are excepted. Their connection degree is 1. A simple kinematic chain is also called an open-loop kinematic chain. A closed-loop kinematic chain contains at least one link with a connection degree greater than or equal to 3. [1]

The degrees of freedom of a mechanism

"The degrees of freedom (DOF)of a mechanism are the number of independent parameters or inputs needed to specify the configuration of the mechanism completely."[3] The DOF value of a mechanism is equal to the DOFs of all the links minus the number of constraints imposed by the joints. Kutzbach-Grübler formula can be used provided that the constraints imposed by the joints are independent of one another and do not introduce redundancies.

$$F = \lambda(n - j - 1) + \sum_{i=1}^{j} f_i \qquad (2.1)$$

"In the above expression F is the number of DOF of the mechanism, f_i is the number of DOF permitted by the joint i , λ is DOF of the working space which is 3 for planar and spherical mechanisms and 6 for spatial mechanisms, n is the total number of links including the base, j is the total number of joints."[3]

"If constraints imposed by the joints introduce redundancies (redundant degrees of freedom) which is also called a passive degree of freedom the number of passive degrees of freedom must be subtracted from the Kutzbach-Grübler formula".[3]Thus, the following *modified* Kutzbach-Grübler formula is obtained:

$$F = \lambda(n - j - 1) + \sum_{i} f_i - f_p \qquad (2.2)$$

Here, f_p is the number of *passive degrees of freedom*, which have no significance in the intended operation of the mechanism. [3]

Example 1:6-SPS Stewart-Gough Platform

This example demonstrates the application of the DOF formula given above. The example involves a 6-SPS Stewart-Gough Platform, which is a parallel manipulator. It is shown in Figure 2.1. This manipulator isa spatial mechanism in which a moving platform is connected to a fixed base with six extensible limbs. Each limb is made up of two aligned links that are connected by a prismatic(P) joint. It also contains two spherical(S) joints at its lower and upper ends so that it is connected to the base and to the moving platform. Because of the S-P-S jointcombination there is a passive degree of freedom associated with each limb, which is the arbitrary spinning rotation about its centerline.

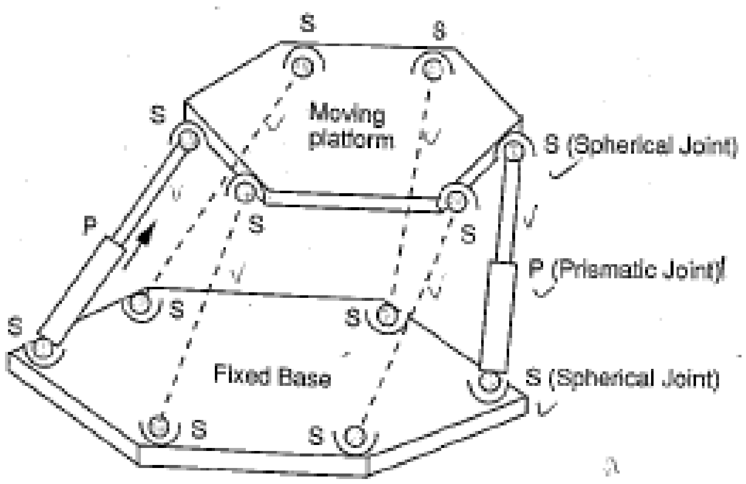


Figure 2.1: A 6-SPS Stewart-Gough Platform

For the mechanism shown in Figure 2.1,

$$\lambda = 6;\ n = 14;\ j_1 = 6;\ j_3 = 12;\ f_p = 6$$

Therefore,

$$F = 6(14 - 18 - 1) + (12 \times 3 + 6) - 6 = 6$$

2.2Classification of Robot Manipulators

In this section classification of robot manipulators based on degree of freedoms, workspace geometry, and motion characteristics is discussed.

<u>Classification by DOFs</u>

An ordinary manipulator possesses 6 degrees of freedom to move an object in the three-dimensional space. On the other hand, a redundant manipulator possesses more than 6 degrees of freedom and a deficient manipulator possesses less than 6 degrees of freedom. The Fanuc s-900W robot shown below in Figure 2.2 is a 6-DOF general purpose manipulator and the Adept-One robot shown further below in Figure 2.3 is a 4-DOF manipulator. [3]

Figure 2.2:The Fanuc s-900W

Figure 2.3:Adept-One

Classification by workspace geometry

"The workspace of a manipulator is the volume of space the end-effector can reach."[3] "*Reachable workspace* is the volume of space within which every point can be reached by the end-effector in at least one orientation."[3] "*Dextrous workspace* is the volume of space within which every point can be reached by the end-effector in all possible orientations." The dextrous workspace is a subset of the reachable workspace.[3]

Classification By Motion Characteristics

"A rigid body is said to perform a planar motion if all particles in the body describe plane curves that lie in parallel planes. A mechanism is said to be a planar mechanism if all the moving links in the mechanism perform planar motions that are parallel to one another."[3]

"A rigid body is said to be under a spherical motion if all particles in the body describe curves that lie on concentric spheres. Thus when a rigid body performs a spherical motion there exists at least one stationary point." [3] A rigid body rotating about a fixed axis can be considered as a special case of spherical motion since any point on the axis of revolution can be treated as the stationary point.[3] “A mechanism is said to be a spherical mechanism if all the moving links perform spherical motions about a common stationary point.”[3] “In addition all the joint axes of a spherical linkage must intersect at a common point.”[3] An example is the 3-DOF spherical parallel manipulator shown in Figure 2.4.It has only revolute joints with concurrent axes.

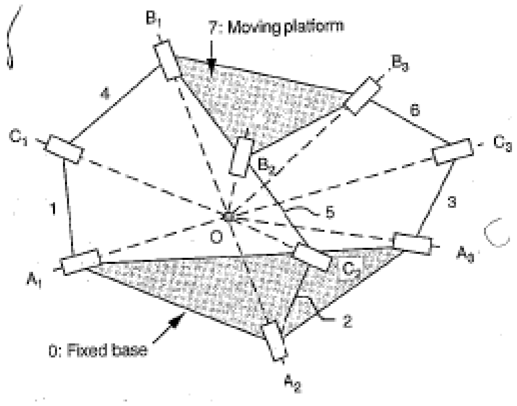


Figure 2.4: A spherical 3-DOF manipulator.

“A rigid body is said to perform a spatial motion if its motion cannot be characterized as planar or spherical motion. A manipulator is called a spatial manipulator if at least one of the moving links in the mechanism possesses a general spatial motion.” [3]

2.3 Parallel Manipulators

Parallel manipulators are mechanisms which consist of at least one closed kinematic chain. A four-bar linkage mechanism is an example of a closed loop kinematic chain. “Robots with parallel manipulators,also sometimes called parallel-kinematics machines,present very good performances in terms of accuracy, rigidity and ability to manipulate large loads.” [1] “They have been used in a large number of applications ranging from astronomy to flight simulators and are becoming increasingly popular in the machine-tool industry."[1] "Parallel manipulators offer advantages over serial ones where high structural stiffness and position accuracy are required."[2]

<u>Classification of Parallel Manipulators</u>

Planar Robots

3-DOF Manipulators

These have a moving a platform with two translational DOFs and 1 rotational DOF. In 3-DOF manipulators each chain contains 2 rigid bodies and 3 joints. The chains can present the following sequences as shown below in Figure 2.5: 3-RRR, 3-RPR, 3-RRP, 3-RPP, 3-PRR, 3-PPR, 3-PRP, 3-PPP. Even though in this classification all the kinematic chains have the same joint sequence, there can be mixed configurations as well, such as RRR-RPR-RRP, RRR-RRP-PPR, etc.The actuated joint can be any of the three joints of the relevant chain. "Generally placing the actuated joint on the end-effector should be avoided in order to lighten the weight of the moving equipment.[1]"

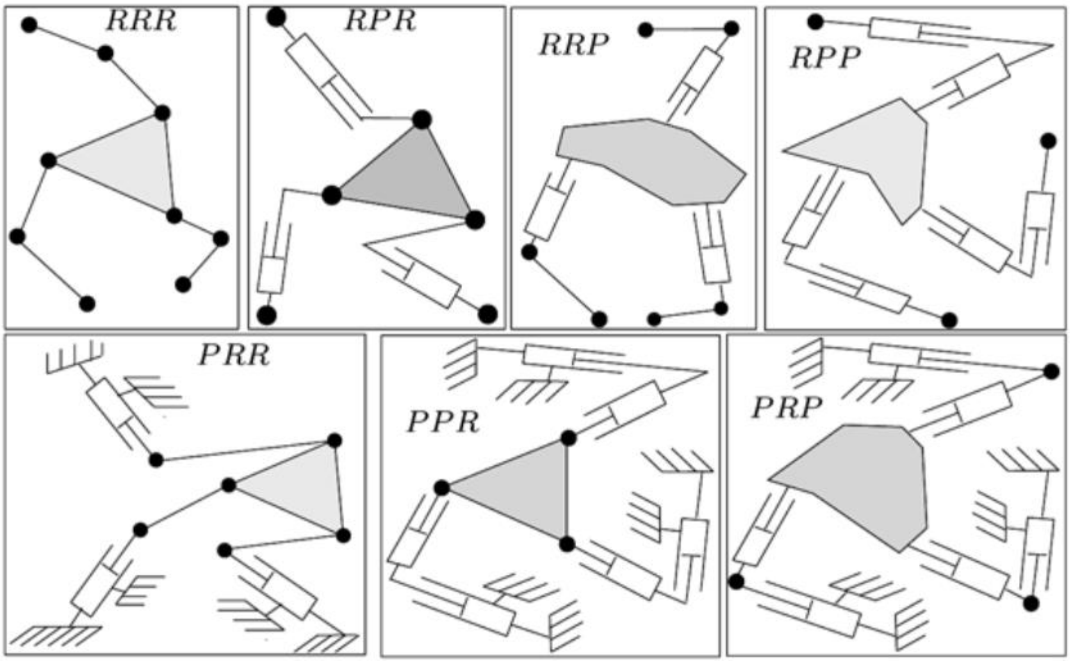


Figure 2.5: Chain sequences for 3-DOF planar parallel manipulators

Spatial motion robots

3-DOF MANIPULATORS

Translation Manipulators

These are manipulators with 3 translational degrees of freedom. Delta robot is the most famous example of translation manipulators.[1] It is illustrated in Figure 2.6.

> "All the kinematic chains of this robot are of the RRPaR type. A motor makes a revolute joint rotate about an axis w. On this joint is a lever at the end of which another joint of the R type is set with axis parallel to w. A parallelogram Pa is fixed to this joint, and allows translation in the directions parallel to w. At the end of this parallelogram is a joint of the R type with axis parallel to w and which is linked to the end-effector."[1]

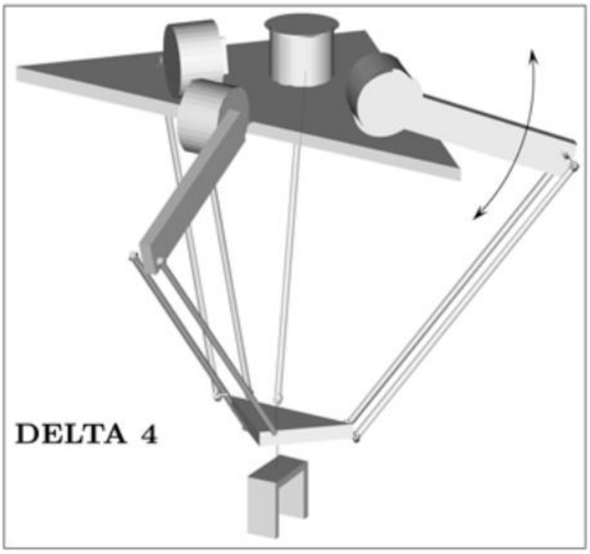


Figure 2.6: Delta Robot

"Delta ancestor is a mechanism described by Pollard intented to be used for car painting."[1] "This mechanism presents three revolute actuators that orientate three arms the ends of which are linked to the pod by three articulated links."[1] For the end-effector to have only translational degrees of freedom the three distal links must connect at ball-and-socket joints that share the same center.[1]

3-CRR, a robot with cylindrical joints proposed by Kong is another member of this family and is shown below in Figure 2.7.[1]

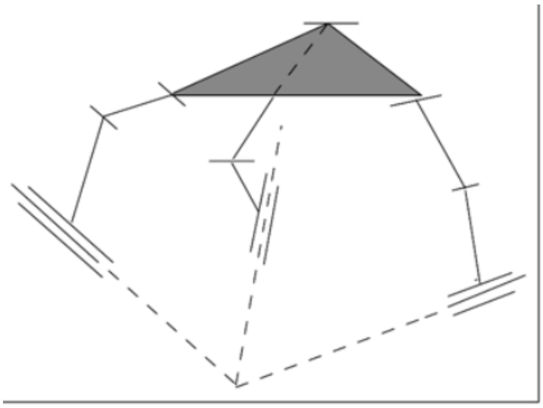


Figure 2.7: 3-CRR Robot

Tricept shown below in Figure 2.8 has an end-effector which has a stem which is free to translate along its axis.[1] “The stem is linked at its base by a universal joint, forbidding the stem to rotate around its axis;three chains of the RRPS type act on the end-effector.”[1]

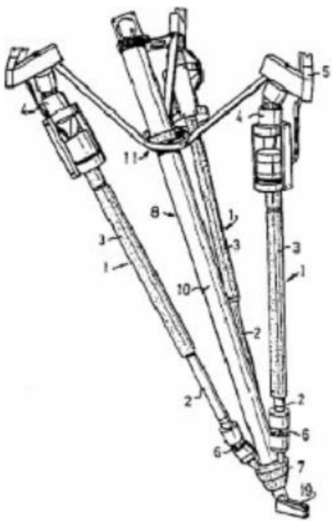


Figure 2.8: Tricept

3-UPU robot proposed by Tsai which is a special case of the family of 3-RRPRR mechanisms is the most academically studied 3DOF translational robot.[1] It is shown below in Figure 2.9.

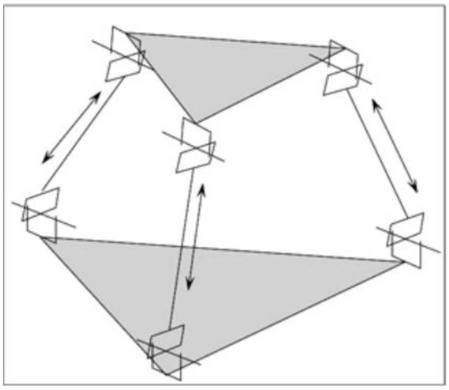


Figure 2.9: 3-UPU

Orientation Manipulators

“These are manipulators allowing three rotations about one point”[1]. “An example is made up of a moving platform a fixed base and extensible limbs.”[1]It is shown below in Figure 2.10. In this example both the end effector and the base take the form of a tetrahedron. The moving platform is directly connected to the fixed base by a spherical joint at point O. Three extensible limbs connect the moving platform to the fixed base. This is not a spherical mechanism because the three limbs and the moving platform do not have a common stationary point. Although the motion of the whole mechanism is not spherical, the moving platform possesses a spherical motion because of the existence of a fixed point O.

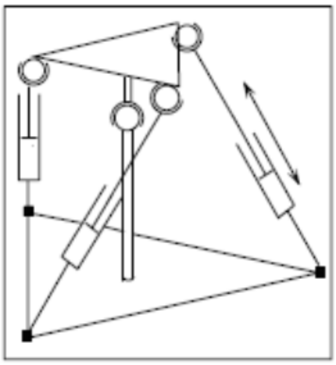
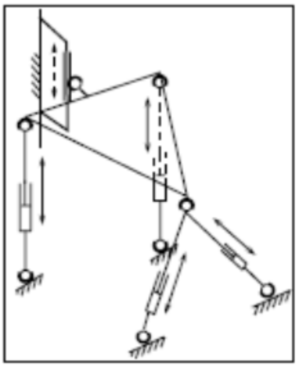


Figure 2.10: Orientation manipulator with a central mast

The 3RUU structure that has been proposed by Ti Gregorio is another example.[1]
Mixed Degrees of Freedom Maniupulators
“The 3 DOF RPS mechanism has a translational DOF along the vertical axis and rotation along the precession and nutation angles.”[1] “Three identical limbs connect to the moving platform by spherical joints and to the fixed base by revolute joints.”[1]

4-DOF MANIPULATORS
“It is theoretically impossible to design a 4 DOF spatial parallel manipulator with identical legs.”[1] ”Such a design has to rely either on a passive constraint mechanism, a specific geometry of the legs, different legs, less than 4 legs or a specific mechanical design.”[1]

“The flight simulator mechanism based on a passive constraint system was presented by koevermans”.[1] The DOFs are the three rotations and one translation. [1]It is shown below in Figure 2.11.



0

Figure 2.11: The flight simulator mechanism

“Specific arrangements to get 3T1R motion (also called schönflies motion) have been presented.”[1] As for a specific design the H4,I4 family of robots can be mentioned.[1]

5-DOF MANIPULATORS

“Robots with 5 DOF also have to rely on passive constraint mechanisms, specific geometries or design.”[1] “5 DOF parallel manipulator bears importance in the machine-tool field for so-called five-axis machining.”[1] “6 DOF are not strictly necessary for machining as the rotation of the spindle adds a DOF.”[1] “5 DOF spatial parallel robots can be constructed by employing a central mast between the moving platform and the fixed base to prohibit the rotation around the normal to the moving platform.”[1] Examples are shown below in Figure 2.12.

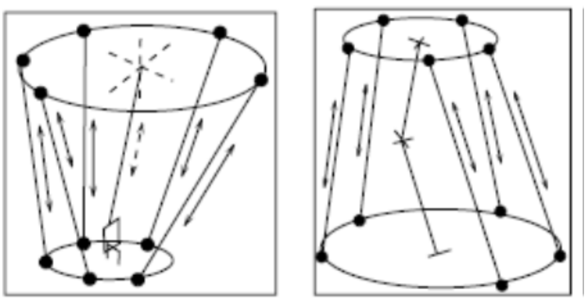


Figure 2.12: 5-DOF spatial motion parallel manipulators

6-DOF MANIPULATORS

Gough Platform has 6-UPS architecture with a hexagonal moving platform as shown below in Figure 2.13. It was originally used to test tire wear and tear. “The moving platform vertices are connected to a link by a ball-and-socket joint. The other end of the link is attached to the base by a universal joint. A linear actuator allows the modification of the total length of the link.”[1] This is the most commonly used parallel robot architecture. “This type of manipulator is usually called a Gough platform, 6-6 robot or hexapod.”[1]

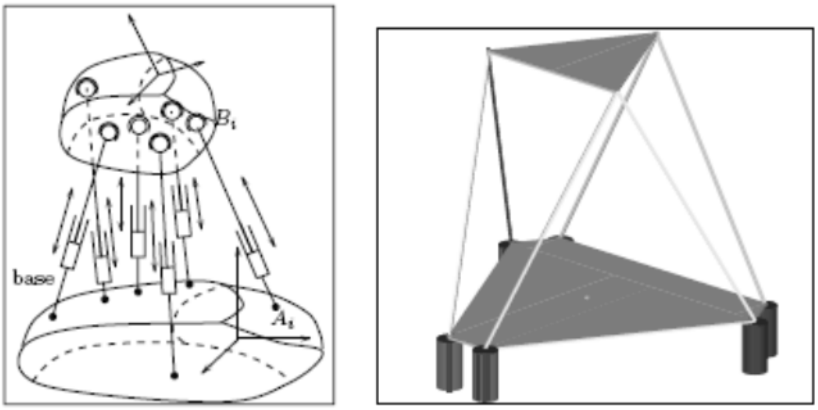


Figure 2.13: A general Stewart-Gough Platform

The first example of PUS-Chain Robot is the INRIA active wrist as shown below in Figure 2.14.[1]

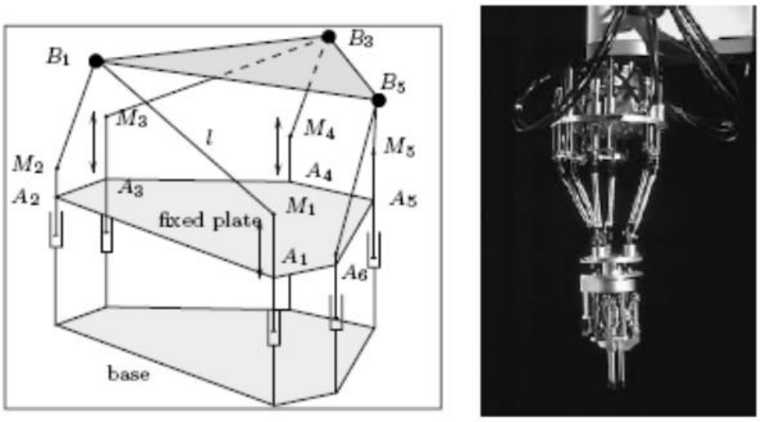


Figure 2.14: INRIA active wrist

"It has a vertical actuated prismatic joint that is connected to a fixed length link by a universal joint. The other end of the link is attached to the moving platform by a spherical joint. This structure has been used for the manufacture of lenses."[1] "Such a structure possesses the advantages of having a very low center of mass, a very light moving mass and reduced risk of collision between the links compared to the 6 UPS".[1] "The direction of the motion of the prismatic actuators may vary. It is tilted in the Hexa-M Machine-tool of Toyota Machine Works , horizontal and parallel in the Hexaglide robot or vertical with only 3 guide ways in the Linapod."[1] They are shown below in Figures 2.15 and 2.16

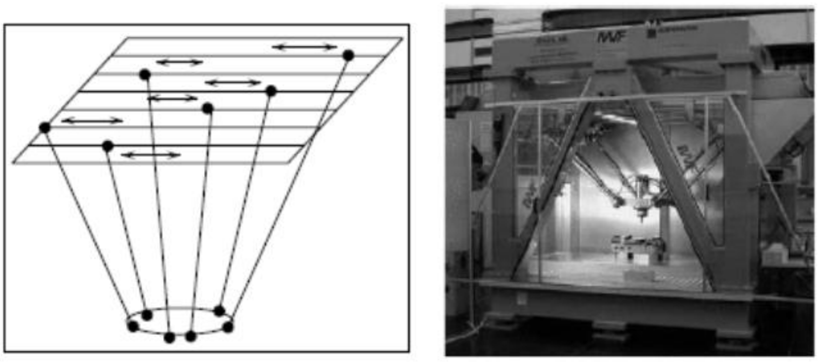


Figure 2.15: Hexaglide robot

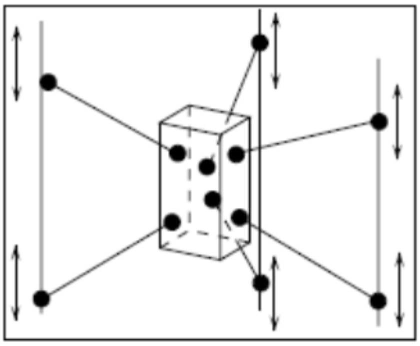


Figure 2.16: Linapod

Another example of PUS-Chain Robot is Nabla-6. “Nabla-6 with horizontal prismatic axis has only three distinct prismatic joint axes with two points sliding on the same axis.”[1] It is shown below in Figure 2.17. “Three ends of prismatic links are articulated on a triple ball-and-socket joint. The position of this common point can be controlled with the help of three associated actuators while the other three control the platform orientation. The result is a decoupled robot.” [1]

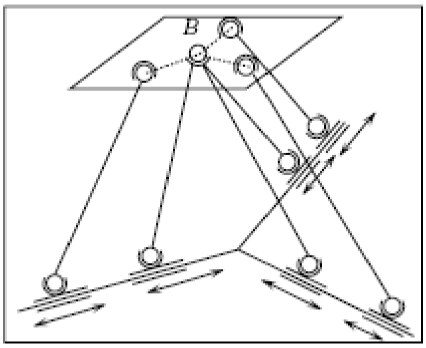


Figure 2.17: Nabla-6

CHAPTER 3

KINEMATICALLY REDUNDANT PARALLEL MANIPULATORS

Kinematically redundant parallel manipulators are parallel manipulators which have more degrees of freedom than the degrees of freedom required for its end-effector.

4-DOF Kinematically Redundant Parallel Mechanism

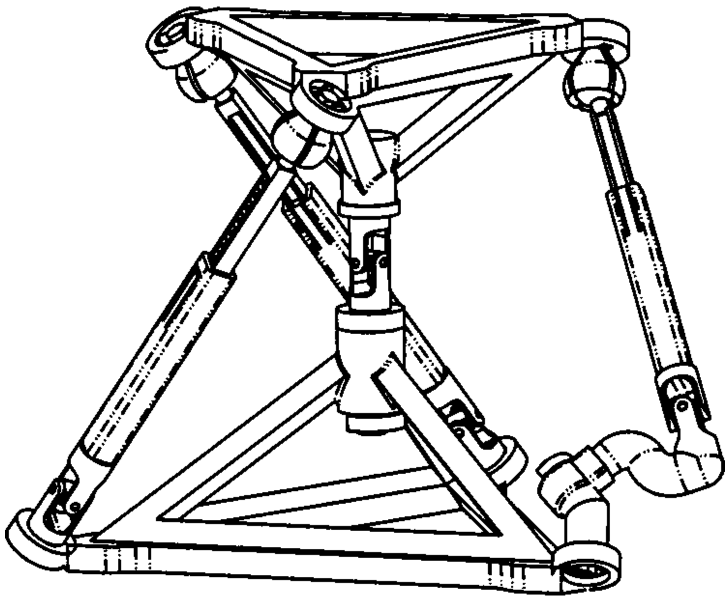


Figure 3.1: 4-DOF Kinematically Redundant Parallel Mechanism

As shown in Figure 3.1 above this mechanism [2]consists of a platform connected to a fixed base via four kinematic sub-chains.

> "Among these chains, one is a S sub-chain which is a spherical joint located at point O, which belongs to the base, two are the normal UPS sub-chains which comprise an actuated prismatic (P) attached to the base by a universal joint (U) and to the platform by a spherical joint (S). The other one is a redundant chain

which is obtained by adding one additional revolute joint and one link in the normal UPS subchains.” [2]

“The platform of this mechanism can be oriented arbitrarily around point O”. The orientation is controlled by adjusting the length of the three input links and the revolute actuator angle. The nonredundant counterpart of this redundant mechanism lacks the additional revolute joint and the additional link in the redundant chain. The conditions for the singularity of the redundant mechanism are reduced relative to the nonredundant mechanism. [2]

<u>Spatial 7-DOF Redundant Parallel Mechanism</u>

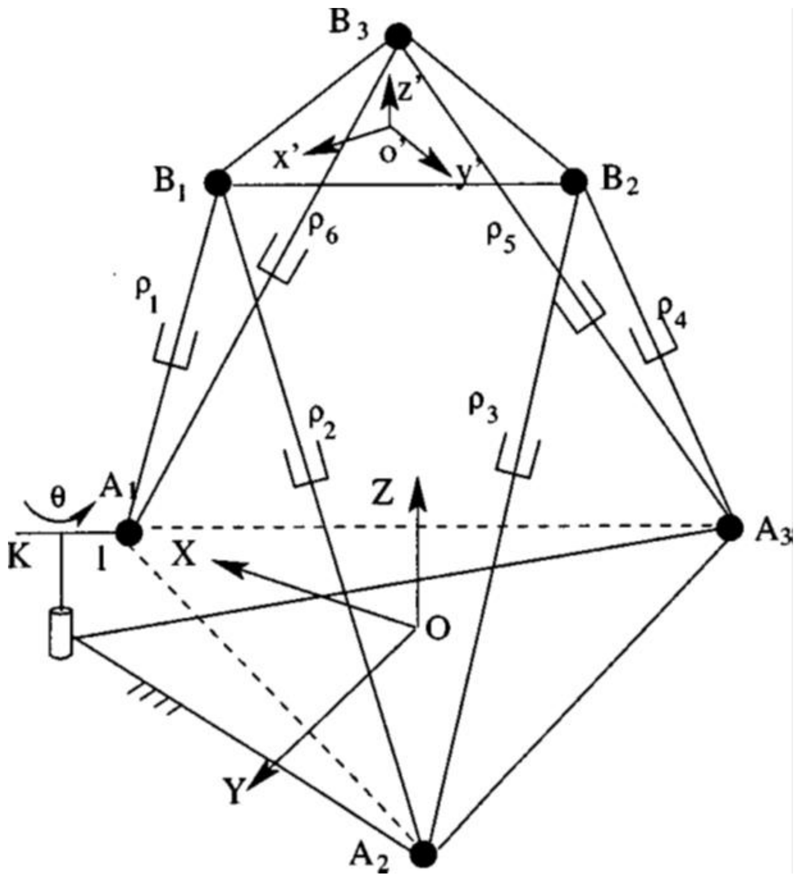


Figure 3.2: Spatial 7-DOF Redundant Parallel Mechanism

This kind of manipulator is obtained by adding one additional revolute joint to the simplest case of Gough-Stewart platform. It is illustrated in Figure 3.2. This structure, similar to the nonredundant Gough-Stewart platform, is composed of a triangular mobile platform connected to a triangular base through six prismatic pairs. The particularity of the architecture of the mechanism is that one of the three points on the base can be rotated around the vertical axis which is perpendicular to the plane of the base. Since an extra revolute joint is added to the Gough-Stewart platform, the original mechanism becomes a seven-degree-of-freedom kinematically redundant parallel platform. “The position and orientation of the platform in space are controlled by adjusting the length of the six legs and the input angle of the revolute pair.”[2] The conditions for the singularity of this redundant mechanism are reduced relative to the nonredundant mechanism. [2]

A new Family of three 6-DOF redundant planar parallel manipulators

Here, 1-degree of kinematic redundancy (1-DOKR) is added to each limb of the 3-RRR manipulator shown below in figure 2.3 producing manipulators with a total 3-DOKR. “Therefore the family of redundant parallel manipulators proposed here has 6 actuated degrees of freedom for a planar task three of which are redundant. The added kinematic redundancies enable the manipulators to avoid kinematic singularities, improve their maneuverability and enlarge their reachable and dexterous workspaces.”[7] Figures 3.3,3.4,3.5,3.6 below illustrate this new family of three 6-ADOF redundant planar parallel manipulators.

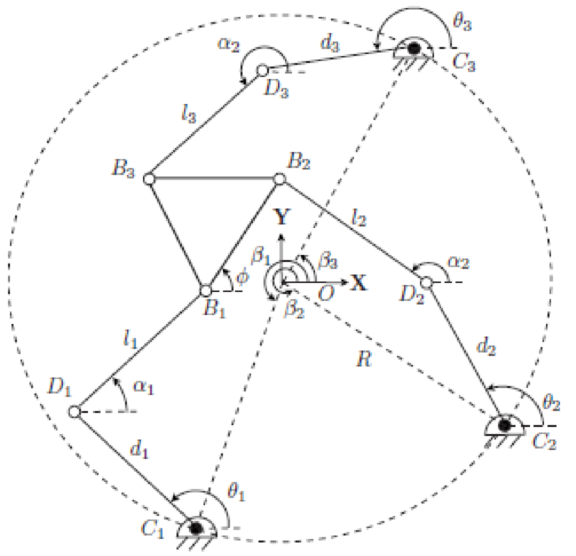


Figure 3.3: 3-RRR manipulator

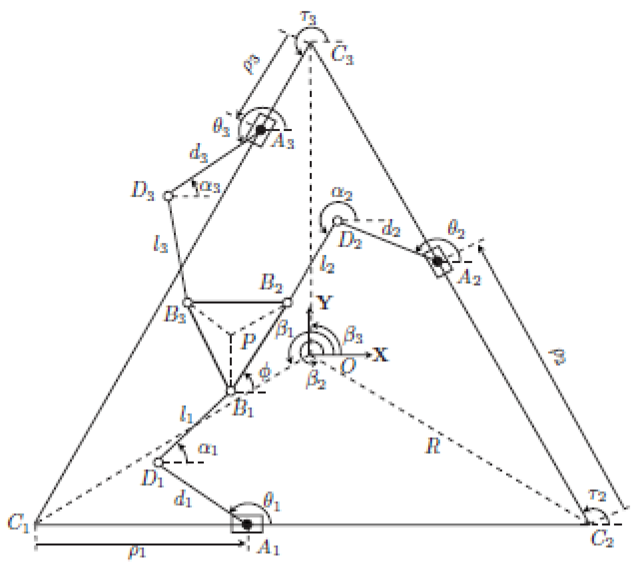


Figure 3.4: 3-PRRR triangle planar 6-ADOF kinematically redundant parallel manipulator

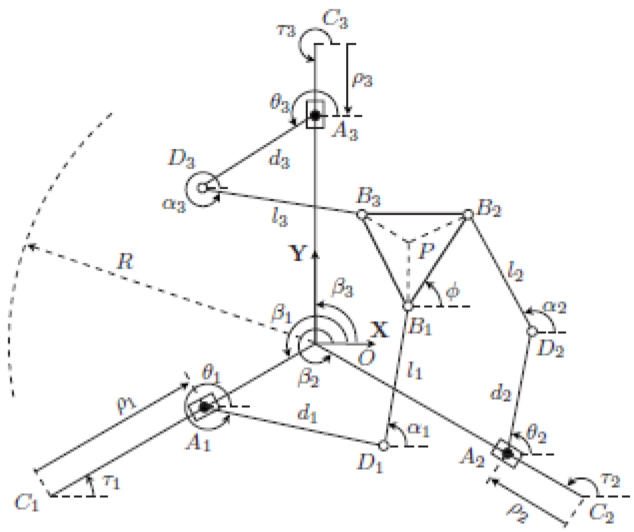


Figure 3.5: 3-PRRR star planar 6-ADOF kinematically redundant parallel manipulator

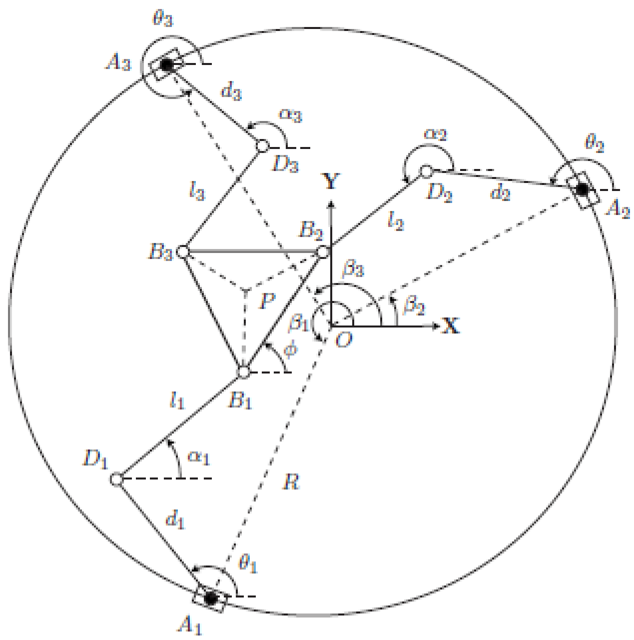


Figure 3.6: 3-PRRR circle planar 6-ADOF kinematically redundant parallel manipulator

Each limb of the 3-PRRR manipulators has one prismatic actuator at its base. “The redundant prismatic actuators slide on their respective guides that can take the shape of a triangle, a star and a circle.” An actuated revolute joint is mounted on the prismatic actuator at Point A_i. Note that the solid circles in all the figures represent active revolute joints, whereas the empty circles represent passive ones. Two passive revolute joints are at D_i and B_i, where point B_i is attached to the end-effector.[7]

<u>9-DOF Redundant Parallel Manipulator</u>

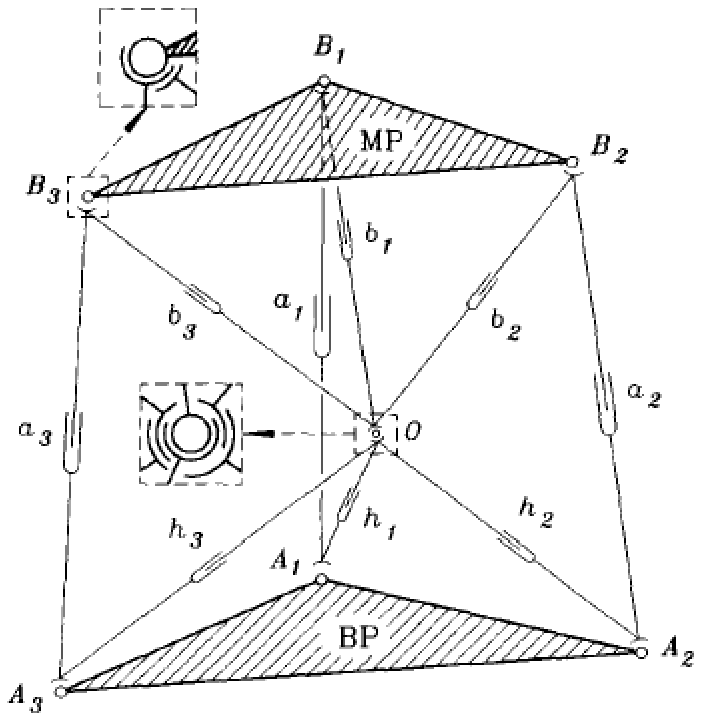


Figure 3.7: 9-DOF Redundant Parallel Manipulator

The model shown in figure 3.7 above consists of nine prismatic in-parallel actuators $a_{i,} b_i, and\ h_i$ for i = 1,2,3. “The three actuators $a_1$through a_3 ,called external legs, connect the moving platform directly to the base platform by double spherical joints at A_i and B_i.”[8] “The remaining six actuators $b_1$through b_3 and $h_1$through h_3 are called the upper internal and the lower internal legs respectively.”[8] “The internal legs are coupled pairwise by three concentric spherical joints at point O.”[8]

CHAPTER 4

MECHANICS OF PARALLEL MANIPULATORS

In this chapter kinetic energy and potential energy definitions for a rigid body, Lagrange's equations, and jacobian analysis and singularity conditions of parallel manipulators are discussed.

4.1 Review of Kinematics

Representation of vectors in different frames of references, vector operations with matrix representations, transformation matrices, rotation matrices, expression of transformation matrix as a rotation matrix and differentiation of vectors can be found in [18].

Kinetic Energy For a Rigid Body Doing Planar Motion

The kinetic energy of a rigid body doing planar motion consists of two parts; namely the translational kinetic energy of the mass center and the body's rotational kinetic energy:

$$K = \frac{1}{2} m v_g^2 + \frac{1}{2} I_z \omega_z^2$$

In the above expression m is the total mass of the rigid body, v_g is ther linear velocity of the center of mass of the rigid body, I_z is the moment of inertia about the center of mass of the rigid body, $\omega_z$is the angular velocity of the rigid body.[11]

Potential Energy For a Rigid Body

Potential Energy For a Rigid Body is given as

$$U = mgh_c$$

In the above expression h_c is the height of the center of mass of the rigid body above the ground.

Lagrange's Equations

Using Lagrange's Equations the actuation torques and forces at the active joints of a mechanism can be obtained [16].

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q_k}}\right) - \frac{\partial K}{\partial q_k} + \frac{\partial U}{\partial q_k} = Q_k + D_k \tag{4.1}$$

Where

$$\delta W = \delta W_{act} + \delta W_{dist} \tag{4.2}$$

$$\delta W_{act} = \sum_{k=1}^{m} Q_k \delta q_k \tag{4.3}$$

$$\delta W_{dist} = \sum_{k=1}^{m} D_k \delta q_k = \vec{F}_{ep} \cdot \delta \vec{r}_p + \vec{M}_{ep} \cdot \delta \vec{\psi}_p \tag{4.4}$$

In the above equations,K is the total kinetic energy of the mechanism, U is the total potential energy of the mechanism, q_k is the *k*th jointvariable, δW is the total virtual work, D_k is the disturbance torque or force and Q_k is the actuation torque or force of the *k*th active joint. $\vec{F}_{ep}$ and $\vec{M}_{ep}$ are the force and moment vectors applied by the environment on the platform, whose linear and angular virtual displacements are denoted by $\delta \vec{r}_p$ and $\delta \vec{\psi}_p$.

4.2 Jacobian Analysis and Singularity Conditions of Parallel Manipulators

In [9] kinematic redundancy of a redundant parallel manipulator is used to avoid direct kinematic singularities of the mechanism. Therefore Jacobian Analysis and Singularity Conditions of Parallel Manipulators must be explained here.

"If the actuated joint variables are denoted collectively by a vector $\bar{q}$ and the position (location and orientation)of the moving platform is described by a vector $\bar{x}$, the kinematic relationship between $\bar{q}$ and $\bar{x}$ can be written as:"[3]

$$\bar{f}(\bar{x},\bar{q}) = \bar{0}$$

"Then after differentiating this equation with respect to time a relationship between the input joint rates and the end-effector output velocity can be obtained as"[3]:

$$J_x\dot{\bar{x}} = J_q\dot{\bar{q}} \tag{4.5}$$

Where

$$J_x = \frac{\partial \bar{f}}{\partial \bar{x}}; J_q = -\frac{\partial \bar{f}}{\partial \bar{q}}$$

This derivation leads to two separate Jacobian matrices . "Due to the existence of two jacobian matrices, a parallel manipulator is said to be at a singular configuration when either J_x or J_q or both are singular. Three different types of singularities can be identified."[3]

Inverse Kinematic Singularities

"An inverse kinematic singularity occurs when the determinant of J_q goes to zero namely"[3]

$$Det(J_q) = 0$$

"When J_q is singular and the null space of J_q is not empty ,there exist some nonzero $\dot{\bar{q}}$ vectors that result in zero $\dot{\bar{x}}$ vectors. Infinitesimal motion of the moving platform along certain directions can not be accomplished."[3] "The manipulator loses one or more degrees of freedom."[3] "At an inverse kinematic singular configuration a parallel manipulator can resist forces or moments in some directions with zero actuator forces or torques."[3] Inverse kinematic singularities usually occur at the workspace boundary.[3]

Direct Kinematic Singularities

"Direct kinematic singularity occurs when the determinant of J_x is equal to zero."[3]

$$Det(J_x) = 0$$

"When J_x is singular if the null space of J_x is not empty there exist some nonzero $\dot{\bar{x}}$

vectors that result in zero $\dot{q}$ vectors."[3] "That is the moving platform can possess infinitesimal motion in some directions while all the actuators are completely locked."[3] "The moving platform gains 1 or more degrees of freedom."[3] "The manipulator can not resist forces or moments in some directions."[3]

Combined Singularities

"This occurs when the determinants of J_x and J_q are both zero."[3]

CHAPTER 5

REDUNDANCY RESOLUTION METHODS FOR REDUNDANT MANIPULATORS

In this chapter first redundancy in manipulators is discussed. Later mainstream redundancy resolution methods at the position and velocity levels are discussed in detail.

5.1 Redundancy in Manipulators

"For a redundant manipulator the number of independent parameters or inputs needed to specify the configuration of the mechanism completely, namely the number of DOFs of the mechanism, is greater than the number of degrees of freedom of the end-effector."[4] "For a manipulator the task space is the space that defines the pose (position and orientation) of the end-effector." The joint space consists of all the joint variables that completely define the configuration of the mechanism. For a redundant manipulator the dimension of the joint space (n) is greater than the dimension of the task space (m). "Regular manipulators which have equal degrees of freedom to their end-effector, may have limited workspace due to mechanical constraints on joints and obstacles that may be present in the work area."[4] "Redundant manipulators have more DOFs than the minimum DOFs required for reaching their task space."[4] "This allows the redundant manipulators to carry out tasks that require high dexterity. They can use extra DOFs to avoid joint limits and the obstacles in the workspace.[4]" The dexterity of redundant manipulators can also be used to satisfy any desirable kinematic or dynamic characteristic.

> "The mathematical methods developed for non-redundant manipulators are not applicable to a redundant one. The inverse kinematic problem for a redundant manipulator has generally infinitely many solutions. Methods that deal with the multiple solutions of the inverse kinematic problem for redundant manipulators and can find the best solution that satisfies a desired criterion are known as

redundancy resolution methods."[4]

5.2 Redundancy Resolution At The position Level

In this study two redundancy resolution methods at the position level have been encountered in the literature one being the lagrange multiplier method and the other singularity avoidance method. In what follows these two methods are explained in detail.

<u>Lagrange Multiplier Method</u>

In this method first the functional relation between q and x is rewritten as follows:

$$F(q) = f(q) - x = 0 \tag{5.1}$$

"Let $H(q)$be some criteria function with continuous first-order partial derivatives which represents the desired performance such as singularity avoidance or obstacle avoidance."[10] The Lagrangian function $L(q)$is defined as the following:

$$L(q) = \lambda^T F(q) + H(q) \tag{5.2}$$

Where λ is an m-dimensional Lagrangian multiplier vector. At the stationary points of L,

$$\frac{\partial L}{\partial q} = \lambda^T \frac{\partial F}{\partial q} + \frac{\partial H}{\partial q} = 0 \tag{5.3}$$

where the mxn matrix $\frac{\partial F}{\partial q}$ is the jacobian matrix J. The second term on the right side of the above equation is the transpose of the gradient vector h such that

$$h = (h_1, h_2, \dots, h_n) \quad ; \quad h_i = \frac{\partial H}{\partial q_i}, \qquad i = 1,2,\dots,n$$

Thus the above equation becomes the following:

$$\lambda^T J = -h^T \tag{5.4}$$

Transposing we get

$$J^T \lambda = -h \tag{5.5}$$

Or

$$\begin{bmatrix} (J^1)^T \\ (J^2)^T \\ . \\ . \\ . \\ (J^n)^T \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ . \\ . \\ . \\ \lambda_m \end{bmatrix} = - \begin{bmatrix} h_1 \\ h_2 \\ . \\ . \\ . \\ h_n \end{bmatrix} \tag{5.6}$$

where $(J^1)^T$ denotes the transpose of ith column vector of the Jacobian matrix. In the above equation there are n linear equations with m unknowns $\lambda_{1,}\lambda_2, \ldots, \lambda_m$. Selecting m linearly independent equations from the above equation which may be chosen to be the first m equations we have

$$\begin{bmatrix} (J^1)^T \\ (J^2)^T \\ . \\ . \\ . \\ (J^m)^T \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ . \\ . \\ . \\ \lambda_m \end{bmatrix} = - \begin{bmatrix} h_1 \\ h_2 \\ . \\ . \\ . \\ h_m \end{bmatrix} \tag{5.7}$$

Inverting, λ is obtained as

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ . \\ . \\ . \\ \lambda_m \end{bmatrix} = - \begin{bmatrix} (J^1)^T \\ (J^2)^T \\ . \\ . \\ . \\ (J^m)^T \end{bmatrix}^{-1} \begin{bmatrix} h_1 \\ h_2 \\ . \\ . \\ . \\ h_m \end{bmatrix} \tag{5.8}$$

Substituting this into the remaining n-m equations

$$\begin{bmatrix} (J^{m+1})^T \\ (J^{m+2})^T \\ . \\ . \\ . \\ (J^n)^T \end{bmatrix} \begin{bmatrix} (J^1)^T \\ (J^2)^T \\ . \\ . \\ . \\ (J^m)^T \end{bmatrix}^{-1} \begin{bmatrix} h_1 \\ h_2 \\ . \\ . \\ . \\ h_m \end{bmatrix} = \begin{bmatrix} h_{m+1} \\ h_{m+2} \\ . \\ . \\ . \\ h_n \end{bmatrix} \tag{5.9}$$

For brevity let us denote

$$J_m = \begin{bmatrix} (J^1)^T \\ (J^2)^T \\ . \\ . \\ . \\ (J^m)^T \end{bmatrix} \; ; \; J_{n-m} = \begin{bmatrix} (J^{m+1})^T \\ (J^{m+2})^T \\ . \\ . \\ . \\ (J^n)^T \end{bmatrix} \; ; h_m = \begin{bmatrix} h_1 \\ h_2 \\ . \\ . \\ . \\ h_m \end{bmatrix} ; \; h_{n-m} = \begin{bmatrix} h_{m+1} \\ h_{m+2} \\ . \\ . \\ . \\ h_n \end{bmatrix}$$

Adding $h_{n-m}$and multiplying both sides of the above equation by -1 the following is obtained,

$$J_{n-m} {J_m}^{-1} h_m - h_{n-m} = 0 \qquad (5.10)$$

Which may be alternatively expressed as

$$\left[J_{n-m} {J_m}^{-1} : -I_{n-m}\right] \begin{bmatrix} h_m \\ h_{n-m} \end{bmatrix} = 0$$

where $I_{n-m}$is an identity matrix of rank (n-m). If we denote

$$Z = \left[J_{n-m} {J_m}^{-1} : -I_{n-m}\right]$$

Then

$$Zh = 0 \qquad (5.11)$$

Since Z is an (n-m) x n matrix and h is an n-dimensional vector the above expression consists of n-m scalar equations. Combined with the original m kinematic equations there are n independent nonlinear equations which now fully specify the n unknowns. This set of n equations has to be solved numerically.[10]

Singularity Avoidance

In this example the redundancy resolution of a 3-RPRR mechanism is realized through local optimization by employing singularity avoidance. The schematic diagram of the 3-RPRR mechanism is given below in Figure 5.1.

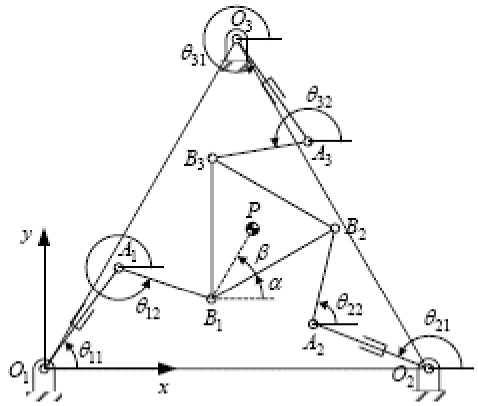


Figure 5.1: A Planar 3-RPRR Manipulator

The kinematic relationships for this mechanism are given as follows:

$$[x - l_{11}cos\theta_{11} - l_{13}\cos(\alpha+\beta)]^2 + [y - l_{11}sin\theta_{11} - l_{13}\sin(\alpha+\beta)]^2 = l_{12}^2 \quad (5.12)$$

$$[x - L - l_{21}cos\theta_{21} + l_{23}\cos(\beta-\alpha)]^2 + [y - l_{21}sin\theta_{21} - l_{23}\sin(\beta-\alpha)]^2 = l_{22}^2 \quad (5.13)$$

$$[x - L/2 - l_{31}cos\theta_{31} + l_{33}\cos(\alpha+3\beta)]^2 + \left[y - \frac{\sqrt{3}L}{2} - l_{31}sin\theta_{31} + l_{33}\sin(\alpha+3\beta)\right]^2 = l_{32}^2 \quad (5.14)$$

Here, $l_{i1}$and l_{i2} denote the lengths of link $O_iA_i$and A_iB_i respectively while l_{i3} and L represent the lengths of B_iP and O_1O_2 respectively.

“Differentiating the above equations with respect to time, the kinematic relationship between p and q is obtained with”[9]:

$$A\dot{p} + B\dot{q} = 0 \quad (5.15)$$

$$p = [x\ y\ \alpha]^T\ ; q = [l_{11}\theta_{11}l_{21}\theta_{21}l_{31}\theta_{31}]^T$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} b_{12}\ 0\ 0\ 0\ 0 \\ 0\ 0\ b_{23} b_{24}\ 0\ 0 \\ 0\ 0\ 0\ 0\ b_{35} b_{36} \end{bmatrix}$$

The expressions for the elements of the A and B matrices are given in Appendix A.

"When the determinant of A is equal to zero, the second type of singularity of the 3-RPRR mechanism occurs."[9] The local optimization criterion in the proposed algorithm is to avoid det(A) = 0. Assuming that a desired task space variable vector p_k is given at time index k the proposed kinematic redundancy resolution algorithm is summarized as follows. When the value of det(A) at the initial configuration is negative (positive) minimizing (maximizing) the cost function

$$H = \det(A_k)$$

Subject to the 3 kinematic constraint equations given above and the incremental limitation

$$l_{i1,k-1} - \Delta l_{i1} \leq l_{i1,k} \leq l_{i1,k-1} + \Delta l_{i1}, i = 1,2,3$$

where Δl_{i1} is determined based on the feasible maximum velocity $\overline{v_i}$ of the prismatic joints, i.e. $\Delta l_{i1} = \overline{v_i}T$. T is the sampling period.[9]

5.3 Redundancy Resolution at the Velocity Level

"Let *x* denote the *m*x1 task space vector. Let *q* denote the *n*x1 joint space vector. The degree of redundancy is defined as *n-m*."[4] The functional relation between *q* and *x* can be written as :

$$x = f(q) \tag{5.16}$$

"This relation is known as the forward kinematics relation." The linear and angular velocity components for the end-effector can be related to the rate of change of the joint variables as follows:

$$\dot{x} = J_e(q)\dot{q} \tag{5.17}$$

Where $J_e(q)$is the *mxn* Jacobian matrix of the end-effector.

> "All the possible joint velocities form an nx1 dimensional mathematical space that is a subset of R^n. All the possible end-effector velocity vectors form an mx1 dimensional mathematical space that is a subset of R^m. At any fixed q the Jacobian matrix can be interpreted as a linear transformation that maps vectors from the space R^n into the space R^m. The input space R^n of the Jacobian matrix has two important associated subspaces. These two subspaces are called the range and the null space. The range of the Jacobian matrix is the subspace of R^n that is covered by the transformation. Physically these are joint velocities that are mechanically possible to be generated by the manipulator's drive mechanism. The null space of the Jacobian matrix is a subset of the input space that is mapped to a zero vector in the output space R^m by the Jacobian matrix. Physically these are the achievable joint velocities that do not generate any velocity at the end-effector."[4]

That is,

$$J_e\dot{q}_{null} = 0 \tag{5.18}$$

"Although the velocities do not generate any motion at the end-effector they generate internal joint motions. Therefore these velocities can be used to satisfy any requirement that the redundant manipulator must meet while the end-effector is performing its main task without being disturbed."[4] Consider a desired end-effector velocity $\dot{x}_d$that can be generated by applying the joint rates $\dot{q}_d$.

$$\dot{x}_d = J_e\dot{q}_d \tag{5.19}$$

If the joint velocities $\dot{q}_{null}$ are selected from the null space and added to $\dot{q}_d$the combined joint velocities $\dot{q}_d + \alpha\dot{q}_{null}$ still generate the desired end-effector velocity.[4]

$$J_e(\dot{q}_d + \alpha\dot{q}_{null}) = J_e\dot{q}_d + 0 = \dot{x}_d \tag{5.20}$$

"If the Jacobian matrix $J_e(q)$ has full column rank at a given joint position q then the dimension of the null space is equal to the degree of redundancy. If the jacobian matrix has a rank of m' < m , the dimension of the null space is equal to n-m'."[4]

"Since the choice of velocities that belong to the null space is not unique there are several ways in which the desired main task $\dot{x}_d$ can be achieved. In other words there are multiple solutions to the inverse kinematics problem for a redundant manipulator."[4]

> "To wisely use these multiple solutions useful additional constraints can be defined. There are two approaches for defining additional constraints: global and local. Global approaches achieve optimal behavior along the whole trajectory which ensures superior performance over local methods. However their computational burden makes them unsuitable for real time sensor based manipulator control applications. In local approaches additional constraints are defined as part time jobs and these part time jobs are not active along the whole trajectory of the end-effector."[4]

To obtain the Jacobian matrix for parallel manipulators first the input-output and loop closure equations must be written out. Now let x denote the mx1 task space vector, q denote the nx1 active joint space vector and p denote the passive joint space vector. The input-output equations are given as:

$$x = f(q,p) \tag{5.21}$$

The loop closure equations are given as:

$$\bar{0} = \phi(q,p) \tag{5.22}$$

Carrying out the following mathematical manipulations

$$\dot{x} = \frac{\partial f}{\partial q}\dot{q} + \frac{\partial f}{\partial p}\dot{p} \tag{5.23}$$

$$\bar{0} = \frac{\partial \phi}{\partial q}\dot{q} + \frac{\partial \phi}{\partial p}\dot{p} \rightarrow \dot{p} = -\left(\frac{\partial \phi}{\partial p}\right)^{-1}\left(\frac{\partial \phi}{\partial q}\right)\dot{q} \tag{5.24}$$

$$\dot{x} = \frac{\partial f}{\partial q}\dot{q} - \frac{\partial f}{\partial p}\left[\left(\frac{\partial \phi}{\partial p}\right)^{-1}\left(\frac{\partial \phi}{\partial q}\right)\dot{q}\right] \tag{5.25}$$

$$\dot{x} = \left[\frac{\partial f}{\partial q} - \frac{\partial f}{\partial p}\left(\frac{\partial \phi}{\partial p}\right)^{-1}\left(\frac{\partial \phi}{\partial q}\right)\right]\dot{q} \tag{5.26}$$

$$\dot{x} = J(q,p)\dot{q} \tag{5.27}$$

And

$$J(q,p) = \left[\frac{\partial f}{\partial q} - \frac{\partial f}{\partial p}\left(\frac{\partial \phi}{\partial p}\right)^{-1}\left(\frac{\partial \phi}{\partial q}\right)\right]$$

<u>Exact Solutions</u>

In this section the pseudo-inverse method and the auugmented jacobian method are presented as the two exact redundancy resolution methods at the velocity level.

Pseudo-Inverse Method

One of the methods used for obtaining the exact solution to the velocity equation is finding the pseudo-inverse of the matrix J_e denoted by ${J_e}^{\#}$. and using it as :

$$\dot{q}_p = {J_e}^{\#}\dot{x}_d \tag{5.28}$$

This is a primary solution to the velocity equation. This solution is not in the null space of the Jacobian J_e.[4] "The pseudo-inverse of J_e can be written as

$${J_e}^{\#} = v\sigma^* u^T \tag{5.29}$$

where $v, \sigma\ and\ u^T$are obtained from the singular-value decomposition (SVT) of J_e and$\sigma^*$is the transpose of σ with all the non-zero values reciprocated."[4] If $J_e$has full row rank its pseudo-inverse is given by

$${J_e}^{\#} = J_e^T(J_e J_e^T)^{-1} \tag{5.30}$$

With the particular solution alone obtained from the pseudo-inverse method, the redundancy of the manipulator can not be exploited for any useful purpose. A joint velocity vector $\dot{q}_{null}$that belongs to the null space of the Jacobian matrix $\mathrm{J_e}$ can be added to the primary solution.

$$\dot{q} = \dot{q}_p + \dot{q}_{null} \tag{5.31}$$

$\dot{q}_{null}$ can be selected as :

$$\dot{q}_{null} = \left(I - J_e{}^{\#} J_e\right) v \tag{5.32}$$

where v is an arbitrary n-dimensional vector. If the arbitrary vector is chosen such that

$$v = -\nabla\phi(q) == -[\frac{\partial\phi}{\partial q_1} \;\; \dots \;\; \dots \;\; \frac{\partial\phi}{\partial q_n}]^T \tag{5.33}$$

where $\phi(q)$is a cost function, a desired minimization task can be satisfied.

The Pseudo-Inverse method can be applied to perform joint limit avoidance. "Since the goal is to keep the joints far from their limits, a representative of the difference of the position of a joint i to the center q_{ci} of the joint range Δq_i is defined as a cost function to be minimized."[4]

$$\phi(q) = \sum_{i=1}^{n} \left[\frac{q_i - q_{ci}}{\Delta q_i}\right]^2$$

$$v = -\nabla\phi(q)$$

"One also may decide to focus only on the joint that is farthest from its center of the range compared to all other joints. This can be expressed as the following mathematical relation."[4]

$$\phi(q) = max \frac{|q_i - q_{ci}|}{\Delta q_i} = \left\| \frac{\bar{q} - \bar{q}_c}{\overline{\Delta q}} \right\|_\infty$$

In the above expression the vector $\frac{\bar{q}-\bar{q}_c}{\overline{\Delta q}} = \bar{z}$ consists of elements z_i where

$$z_i = \frac{q_i - q_{ci}}{\Delta q_i}$$

The infinity norm is not differentiable. The p-norm defined by

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

for a vector x is an acceptable approximation for the infinity norm. Using the p-norm, a proper cost function for the joint limit avoidance can be defined as

$$\phi(q) = \left\| \frac{\bar{q} - \bar{q}_c}{\overline{\Delta q}} \right\|_p$$

“The higher p is,the closer the cost function is to the infinity norm.”[4] “P=6 is sufficient for most practical cases.”[4]

“Another problem with the primary solutions provided by the pseudo inverse method is that they may lead to singular configurations for the manipulator at which the Jacobian matrix does not have full rank.”[4]

Augmented Jacobian Matrix Method

“In this method, for a redundant manipulator with the degree of redundancy of r = n-m , r additional tasks are defined.”[4] “Since the additional task z is a function of the joint variables q, Jacobian of the additional task can be defined that relates their rate of change as :”[4]

$$\dot{z} = J_c\dot{q} \tag{5.34}$$

Now the number of equations and unknowns are balanced in the velocity equation. The augmented task vector can be expressed as :

$$y = \begin{bmatrix} x \\ z \end{bmatrix}$$

and

$$\dot{y} = \begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = J_a\dot{q} \tag{5.35}$$

where $J_a$is the nxn augmented Jacobian matrix

$$J_a = \begin{bmatrix} J_e \\ J_c \end{bmatrix}$$

“The solution for the joint rates $\dot{q}$ can be simply found by using the inverse of J_a.”[4]

Problems associated with this method: “For the inverse of the augmented Jacobian matrix to exist at all times the additional tasks must be defined at all times.” “Part time additional tasks such as obstacle avoidance or joint limit avoidance that are defined based on some conditions that may not exist at all times cannot be used as additional tasks."[4] "This method is not suitable for part-time tasks."[4] "Also, extra singularities can be introduced into the kinematics of the redundant manipulator by defining the additional task."[4] At certain postures the additional task Jacobian $J_c$may have possible rank deficiencies. Or at certain postures the rows of J_e or J_c may become linearly dependent. "This linear dependency which leads to singularity in the matrix $J_a$is task dependent and very hard to predict."[4]

Approximate Solution Methods

In this section singularity avoidance and configuration control methods are presented as the two approximate redundancy resolution methods at the velocity level.

Singularity Avoidance

"Close to a singular posture generating a velocity component in certain directions at the end-effector of a manipulator requires very high joint rates which are not physically possible for the joints to afford."[4] "A redundant manipulator can avoid singular postures by exploiting its extra DOFs than that required for a given main task."[4] If a cost function is defined as

$$F = \|J_e\dot{q} - \dot{x}_d\|^2 + \|\lambda\dot{q}\|^2$$

the partial derivative of the cost function with respect to $\dot{q}$ vanishes for $\dot{q}$ that minimizes F.

$$\frac{\partial F}{\partial \dot{q}^T} = 2(J_e^T J_e \dot{q} + \lambda^2 \dot{q} - J_e^T \dot{x}_d) = 0 \qquad (5.36)$$

Solving the partial derivative of the cost function F for the unknown $\dot{q}$ results in

$$\dot{q} = (J_e^T J_e + \lambda^2 I)^{-1} J_e^T \dot{x}_d \qquad (5.37)$$

"This way high joint rates are penalized causing the manipulator not to move close to the singularity posture."[4] The solution is unique and closely approximates the exact solution.[4]

Configuration Control

"In addition to the main task $\dot{x}_d$an additional task $\dot{z}_d$ and a singularity avoidance task are considered."[4]

$$\dot{x} = J_e\dot{q}$$

$$\dot{z} = J_c\dot{q}$$

"There is no restriction on the dimension of the additional tasks unlike for the augmented Jacobian method."[4] "The joint rates $\dot{q}$ are found such that the error for the main and the additional tasks are minimized while high joint rates are penalized."[4] A cost function is defined as follows:

$$F = (J_e\dot{q} - \dot{x}_d)^T W_e (J_e\dot{q} - \dot{x}_d) + (J_c\dot{q} - \dot{z}_d)^T W_c (J_c\dot{q} - \dot{z}_d) + \dot{q}^T W_v \dot{q}$$

"Where We, Wc and Wv are diagonal positive-definite weighting matrices that assign priority to the main, additional and singularity avoidance tasks."[4] "The joint rates that minimize the cost function can be found by equating the derivative of F to zero."[4]

$$\frac{\partial F}{\partial \dot{q}^T} = 2\left({J_e}^T W_e J_e + {J_c}^T W_c J_c + W_v\right)\dot{q} - 2\left({J_e}^T W_e \dot{x}_d + {J_c}^T W_c \dot{z}_d\right) = 0 \tag{5.38}$$

$$\dot{q} = \left({J_e}^T W_e J_e + {J_c}^T W_c J_c + W_v\right)^{-1}\left({J_e}^T W_e \dot{x}_d + {J_c}^T W_c \dot{z}_d\right) \tag{5.39}$$

> "Since there is no restriction on the dimension of the additional task unlike for the augmented jacobian method, the disadvantages of the augmented jacobian method do not exist for the configuration control method. Any part-time additional task for example joint limit avoidance or obstacle avoidance can be defined as the additional task. When the additional task is not active, for example, when the joints are not close to their limits, there are not as many active tasks as the degree of redundancy. In these situations a solution similar to that of the singularity avoidance method is yielded. When the additional task is active, when some of the joints are close to their limits the number of active additional tasks can be larger than the degree of redundancy. In those cases the best solution that minimizes the cost function F is yielded."[4]

"In configuration control if joint limit avoidance is chosen as the additonal task then the limits for the joints are defined by part-time constraints as additional tasks. These part-time additional tasks are active for a joint when the joint position is close to the joint limit. When a joint position is far from the joint limit, the joint limit avoidance additional task becomes inactive for that joint."[4] "A joint limit avoidance additional task is activated and deactivated by wisely selecting its corresponding weight matrix in the configuration control formulation W_c."[4] Usually a continuous weight for each joint is defined to ensure a smooth joint trajectory. "In a region of the joint motion around the center of the joint range the weight of the joint limit avoidance task for that joint is selected to be zero."[4] "A buffer region is assumed with a width τ_i. When the joint position enters this region, the weight of the joint limit avoidance task is increased from zero to a maximum at the lower or upper limit."[4]

$$W_{c_{ii}} = \begin{cases} W_0 & if\ q_i < q_{min} \\ \frac{W_0}{2}\left[1 + \cos\left(\pi\left(\frac{q_i - q_{imin}}{\tau_i}\right)\right)\right] & if\ q_{imin} \le q_i \le q_{imin} + \tau_i \\ 0 & if\ q_{imin} + \tau_i < q_i < q_{imax} - \tau_i \\ \frac{W_0}{2}\left[1 + \cos\left(\pi\left(\frac{q_{imax} - q_i}{\tau_i}\right)\right)\right] & if\ q_{imax} - \tau_i \le q_i \le q_{imax} \\ W_0 & if\ q_i > q_{imax} \end{cases}$$

"Since all the joints need to be monitored for the limits the additional task is defined as a one-to-one function of the joint positions."That is,

$$z = q \tag{5.40}$$

The corresponding Jacobian for the additional task Jc is defined by

$$J_c = \frac{\partial z}{\partial q} = I \tag{5.41}$$

"Also since the joint rates must vanish when the joint limits are reached, the desired joint rates when the joint limit avoidance additional task is active must be selected to be zero."

$$\dot{z}_d = 0 \tag{5.42}$$

The jacobian for the additional task and the weight matrix Wc are used with the configuration control method.[4]

"Similar to Joint Limit Avoidance, obstacle avoidance is a part-time task which is only activated when a possibility of collision is detected. The distance of a link to an obstacle is calculated. Obstacles are enclosed in circles with diameters larger than the largest obstacle dimension. The thickness of the manipulator links should also be added to the radii of these circles. These circles are called the Surface of Influence."[4]

The location of the potential point of collision a.k.a the critical point is calculated as follows and it is shown below in Figure 5.2:

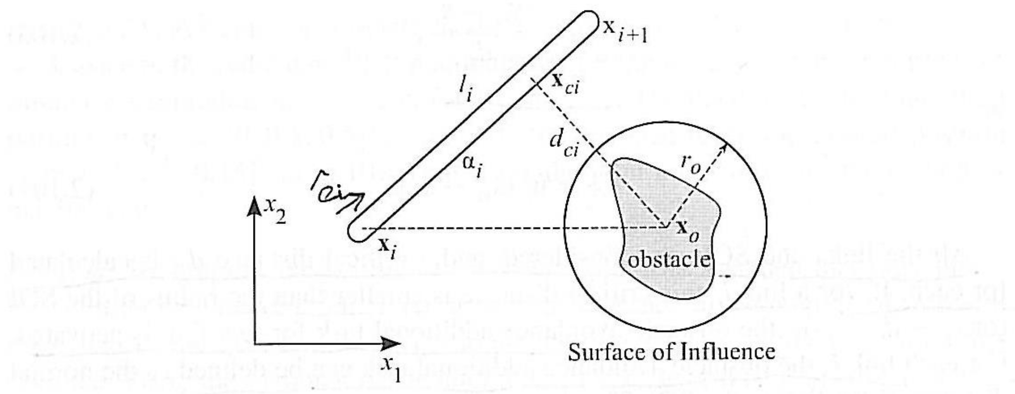


Figure 5.2: The critical point

The unit vector representing the direction of the link with length l_i is:

$$\widehat{e_i} = \frac{x_{i+1} - x_i}{l_i} \tag{5.43}$$

where x_{i+1} and $x_i$are the cartesian coordinates of the joints. If the center of the SOI(Surface of Influence) is at the Cartesian coordinates x_o, the projection of a line from joint i to the center of the SOI on the link i is

$$\alpha_i = \widehat{e_i}^T (x_o - x_i) \tag{5.44}$$

"The critical point is the closest point on the link i to the center of the SOI. The Cartesian coordinates of the critical point can be calculated as:"[4]

$$x_{c_i} = x_i + \alpha_i \widehat{e_i} \tag{5.45}$$

The distance of the critical point with the center of the SOI is:

$$d_{c_i} = \left\| x_{c_i} - x_o \right\| \tag{5.46}$$

“The unit vector pointing from the critical point to the center of the obstacle is determined as:”[4]

$$\hat{u}_i = \frac{x_{c_i} - x_o}{d_{c_i}} \tag{5.47}$$

“If for link i the critical distance is smaller than the radius of the SOI, the obstacle avoidance additional task for that link is activated. For each link i the obstacle avoidance task can be defined as the normal distance of the link to the SOI.”[4] That is

$$z_i = fi(q,t) = r_o - d_{c_i} \tag{5.48}$$

The derivative of the task can be written as

$$\dot{z}_i = -\frac{d}{dt}\left(d_{c_i}\right) = -\hat{u}_i^T\left(\frac{\partial x_{c_i}}{\partial q}\dot{q} - \dot{x}_o\right) \tag{5.49}$$

Where $\dot{x}_o$is the velocity of the center of the SOI or the obstacle.

“The obstacle avoidance additional task must be defined such that a link does not enter the SOI of its corresponding obstacle.”[4]That is,

$$z_i^d = 0 \tag{5.50}$$

Also

$$\dot{z}_i^d = \ddot{z}_i^d = 0 \tag{5.51}$$

“Since each link i has its own unique condition regarding the obstacles, each row of the jacobian matrix for the additional task is calculated separately based on the position of the critical point on link i.”[4] The ith row of the jacobian matrix is derived as:[4]

$$J_{c_i} = -\hat{u}_i^T J_{x_{c_i}} \tag{5.52}$$

where

$$J_{x_{c_i}} = \frac{\partial x_{c_i}}{\partial q}$$

CHAPTER 6

DESCRIPTION AND KINEMATICS OF THE CASE-STUDY MANIPULATOR

The redundant planar parallel manipulator that is used as an example in this study is shown below in Figure 6.1. Its purpose of use is to make desired patterns or carvings on a material with planar surface. The end-effector can be a pen, a carver, a spray gun,a milling cutter or a laser head. The tip point of the end-effector which is desired to be positioned depending on the task is placed at the center of the moving triangular platform of the manipulator.

As seen, the manipulator consists of eight moving links and seven revolute and three prismatic joints. According to the Kutzbach-Grübler formula we have $n = 9, j = 10, \lambda = 3$ and $f_i = 1$ for the prismatic and revolute joints. Applying the Kutzbach-Grübler formula to the manipulator yields $F = 3 \times (9 - 10 - 1) + 10 = 4$

Hence, the manipulator is a 4-DOF mechanism. The active joints of the manipulator are three prismatic joints and the revolute joint at point O. The other joints are passive.

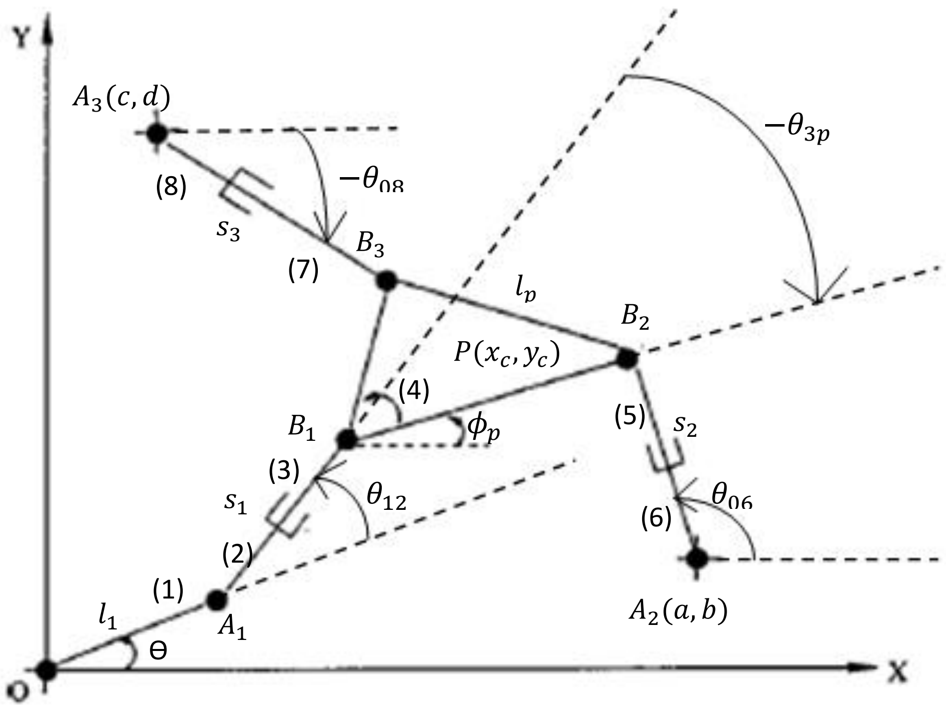


Figure 6.1:Case Study Manipulator

The input-output equations of the manipulator can be written as:

$$\bar{x} = \bar{f}(\bar{q}, \bar{p})$$

In this equation $\bar{x}$ is the output vector showing the position of the end-effector(its orientation and the location of its tip point P). $\bar{q}$ is the input vector consisting of the active joint variables.$\bar{p}$ is the vector including the passive joint variables. The details of these vectors are shown below.

$$\bar{x} = \begin{bmatrix} \phi_p \\ x_c \\ y_c \end{bmatrix}, \bar{q} = \begin{bmatrix} \theta \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}, \bar{p} = \begin{bmatrix} \theta_{12} \\ \theta_{3p} \\ \theta_{06} \\ \theta_{08} \end{bmatrix}$$

$$f_1 = \theta + \theta_{12} + \theta_{3p} - \emptyset_p = 0 \qquad (6.1)$$

$$f_2 = l_1\cos(\theta) + s_1\cos(\theta+\theta_{12}) + l_p\frac{\sqrt{3}}{3}\cos\left(\theta+\theta_{12}+\theta_{3p}+\frac{\pi}{6}\right) - x_c = 0 \tag{6.2}$$

$$f_3 = l_1\sin(\theta) + s_1\sin(\theta+\theta_{12}) + l_p\frac{\sqrt{3}}{3}\sin\left(\theta+\theta_{12}+\theta_{3p}+\frac{\pi}{6}\right) - y_c = 0 \tag{6.3}$$

The manipulator has two independent loops. Four loop closure equations belonging to these loops can be written as follows:

$$g_1 = l_1\cos(\theta) + s_1\cos(\theta+\theta_{12}) + l_p\cos(\theta+\theta_{12}+\theta_{3p}) - a - s_2\cos(\theta_{06}) = 0 \tag{6.4}$$

$$g_2 = l_1\sin(\theta) + s_1\sin(\theta+\theta_{12}) + l_p\sin(\theta+\theta_{12}+\theta_{3p}) - b - s_2\sin(\theta_{06}) = 0 \tag{6.5}$$

$$g_3 = l_1\cos(\theta) + s_1\cos(\theta+\theta_{12}) + l_p\cos(\theta+\theta_{12}+\theta_{3p}+\pi/3) - c - s_3\cos(\theta_{08}) = 0 \tag{6.6}$$

$$g_4 = l_1\sin(\theta) + s_1\sin(\theta+\theta_{12}) + l_p\sin(\theta+\theta_{12}+\theta_{3p}+\pi/3) - d - s_3\sin(\theta_{08}) = 0 \tag{6.7}$$

It has been assumed that the revolute joints of this manipulator have no limits. However its prismatic joints have lower and upper limits, which are given below for $i = 1,2,3$:

$$s_i - s_i^{max} \leq 0 \tag{6.8}$$

$$s_i^{min} - s_i \leq 0 \tag{6.9}$$

For the usage purpose of the manipulator(which is point positioning as mentioned before)it is enough to control only the motion of the tip point of the end-effector. Therefore, the angle $\emptyset_p$ of the platform, which also shows the orientation of the end-effector, has been let run free. As a result, for a *planar point positioning task*, which requires only two degrees of freedom, the degree of redundancy of the manipulator becomes two.

CHAPTER 7

OPTIMAL REDUNDANCY RESOLUTION

The task requiring two degrees of freedom that the redundant manipulator with four degrees freedom will do has been presented in chapter 5. The redundancy resolution for this task has been considered as an optimization problem in this study. The optimization problem has been defined as the minimization of a suitable cost function which is chosen based on a desired criterion. This function can be the required total power for the task to be carried out by the manipulator as well as the potential and/or kinetic energy of the manipulator.

7.1 Optimal Redundancy Resolution At The Position Level

Optimal Redundancy Resolution At The Position Level has been done by minimizing the potential energy of the manipulator. In this case the solution is obtained directly at the position level because potential energy is only a function of the positions of the links. The total potential energy of the manipulator is found as:

$$U = \sum_{i=1}^{8} m_i g y_{ci} \tag{7.1}$$

In the above equation m_i and y_{ci} are the mass and the mass center height of the*i*th link above the ground. The mass center heights of the links are obtained from the position vectors of the links. The position vectors of the mass centers of the links are found as:

$$\bar{r}_{1c}^{(1)} = \begin{bmatrix} \frac{l_1}{2} \\ 0 \\ 0 \end{bmatrix} \tag{7.2}$$

$$\bar{r}_{1c}^{(0)} = e^{\tilde{u}_3\theta}\bar{r}_{1c}^{(0)} = \bar{r}_{1c} = \begin{bmatrix} (l_1 \times cos(\theta))/2 \\ (l_1 \times sin(\theta))/2 \\ 0 \end{bmatrix} \tag{7.3}$$

$$\bar{r}_{2c} = \bar{r}_{2c}^{(0)} = \bar{r}_{OA1}^{(0)} + \bar{r}_{A12c}^{(0)} \tag{7.4}$$

$$\bar{r}_{A12c}^{(0)} = \hat{C}^{(0,1)}\bar{r}_{A12c}^{(1)} = e^{\tilde{u}_3\theta}\bar{r}_{A12c}^{(1)} \tag{7.5}$$

$$\bar{r}_{A12c}^{(1)} = \hat{C}^{(1,2)}\bar{r}_{A12c}^{(2)} = e^{\tilde{u}_3\theta_{12}}\bar{r}_{A12c}^{(2)} \tag{7.6}$$

$$\bar{r}_{A12c}^{(2)} = \begin{bmatrix} LC1 \\ 0 \\ 0 \end{bmatrix} \tag{7.7}$$

$$\bar{r}_{OA1}^{(0)} = \begin{bmatrix} l_1 \times \cos(\theta) \\ l_1 \times \sin(\theta) \\ 0 \end{bmatrix} \tag{7.8}$$

$$\bar{r}_{2c} = \begin{bmatrix} l_1 \times cos(\theta) + LC1 \times cos(\theta) \times cos(\theta_{12}) \ldots \\ \ldots - LC1 \times sin(\theta) \times sin(\theta_{12}) \\ l_1 \times sin(\theta) + LC1 \times cos(\theta) \times sin(\theta_{12}) \ldots \\ \ldots + LC1 \times cos(\theta_{12}) \times sin(\theta) \\ 0 \end{bmatrix} \tag{7.9}$$

$$\bar{r}_{3c} = \bar{r}_{3c}^{(0)} = \bar{r}_{OA1}^{(0)} + \bar{r}_{A13c}^{(0)} \tag{7.10}$$

$$\bar{r}_{A13c}^{(1)} = \hat{C}^{(1,2)}\bar{r}_{A13c}^{(2)} = e^{\tilde{u}_3\theta_{12}}\bar{r}_{A13c}^{(2)} \tag{7.11}$$

$$\bar{r}_{A13c}^{(0)} = \hat{C}^{(0,1)}\bar{r}_{A13c}^{(1)} = e^{\tilde{u}_3\theta}\bar{r}_{A13c}^{(1)} \tag{7.12}$$

$$\bar{r}_{A13c}^{(2)} = \begin{bmatrix} s_1 - LP1 \\ 0 \\ 0 \end{bmatrix} \tag{7.13}$$

$$\bar{r}_{3c} = \begin{bmatrix} l_1 \times cos(\theta) - \ cos(\theta) \times cos(\theta_{12}) \times (LP1 - \ s_1) + \cdots \\ \ldots sin(\theta) \times sin(\theta_{12}) \times (LP1 - \ s_1) \\ l_1 \times sin(\theta) - \ cos(\theta) \times sin(\theta_{12}) \times (LP1 - \ s_1) - \ldots \\ \ldots cos(\theta_{12}) \times sin(\theta) \times (LP1 - \ s_1) \\ 0 \end{bmatrix} \tag{7.14}$$

$$\bar{r}_{4c} = \bar{r}_{4c}^{(0)} = \begin{bmatrix} x_c \\ y_c \\ 0 \end{bmatrix} \tag{7.15}$$

$$\bar{r}_{6c} = \bar{r}_{6c}^{(0)} = \bar{r}_{OA2}^{(0)} + \bar{r}_{A26c}^{(0)} \tag{7.16}$$

$$\bar{r}_{A26c}^{(0)} = \ \hat{C}^{(0,6)} \bar{r}_{A26c}^{(6)} = \ e^{\tilde{u}_3 \theta_{06}} \bar{r}_{A26c}^{(6)} \tag{7.17}$$

$$\bar{r}_{A26c}^{(6)} = \begin{bmatrix} LC2 \\ 0 \\ 0 \end{bmatrix} \tag{7.18}$$

$$\bar{r}_{OA2}^{(0)} = \begin{bmatrix} a \\ b \\ 0 \end{bmatrix} \tag{7.19}$$

$$\bar{r}_{6c} = \begin{bmatrix} a \ + \ LC2 \times cos(\theta_{06}) \\ b \ + \ LC2 \times sin(\theta_{06}) \\ 0 \end{bmatrix} \tag{7.20}$$

$$\bar{r}_{5c} = \bar{r}_{5c}^{(0)} = \bar{r}_{OA2}^{(0)} + \bar{r}_{A25c}^{(0)} \tag{7.21}$$

$$\bar{r}_{A25c}^{(0)} = \ \hat{C}^{(0,6)} \bar{r}_{A25c}^{(6)} = \ e^{\tilde{u}_3 \theta_{06}} \bar{r}_{A25c}^{(6)} \tag{7.22}$$

$$\bar{r}_{A25c}{}^{(6)} = \begin{bmatrix} s_2 - LP2 \\ 0 \\ 0 \end{bmatrix} \tag{7.23}$$

$$\bar{r}_{5c} = \begin{bmatrix} a - cos(\theta_{06}) \times (LP2 - s_2) \\ b - sin(\theta_{06}) \times (LP2 - s_2) \\ 0 \end{bmatrix} \tag{7.24}$$

$$\bar{r}_{8c} = \bar{r}_{8c}^{(0)} = \bar{r}_{OA3}^{(0)} + \bar{r}_{A38c}^{(0)} \tag{7.25}$$

$$\bar{r}_{A38c}^{(0)} = \hat{C}^{(0,8)}\bar{r}_{A38c}^{(8)} = e^{\tilde{u}_3\theta_{08}}\bar{r}_{A38c}^{(8)} \tag{7.26}$$

$$\bar{r}_{A38c}{}^{(8)} = \begin{bmatrix} LC3 \\ 0 \\ 0 \end{bmatrix} \tag{7.27}$$

$$\bar{r}_{OA3}{}^{(0)} = \begin{bmatrix} c \\ d \\ 0 \end{bmatrix} \tag{7.28}$$

$$\bar{r}_{8c} = \begin{bmatrix} c + LC3 \times cos(\theta_{08}) \\ d + LC3 \times sin(\theta_{08}) \\ 0 \end{bmatrix} \tag{7.29}$$

$$\bar{r}_{7c} = \bar{r}_{7c}^{(0)} = \bar{r}_{OA3}^{(0)} + \bar{r}_{A37c}^{(0)} \tag{7.30}$$

$$\bar{r}_{A37c}^{(0)} = \hat{C}^{(0,8)}\bar{r}_{A37c}^{(8)} = e^{\tilde{u}_3\theta_{08}}\bar{r}_{A37c}^{(8)} \tag{7.31}$$

$$\bar{r}_{A37c}{}^{(8)} = \begin{bmatrix} s_3 - LP3 \\ 0 \\ 0 \end{bmatrix} \tag{7.32}$$

$$\bar{r}_{7c} = \begin{bmatrix} c - cos(\theta_{06}) * (LP3 - s_3) \\ d - sin(\theta_{06}) * (LP3 - s_3) \\ 0 \end{bmatrix} \tag{7.33}$$

In the above equations LC_i *and* LP_i are the half lengths of the ith cylinder and the ith piston respectively. Let

$$\bar{r}_{ic} = \begin{bmatrix} r_{icx} \\ r_{icy} \\ r_{icz} \end{bmatrix} \tag{7.34}$$

y_{ci} can be found as :

$$y_{ci} = r_{icy} \tag{7.35}$$

To obtain the masses of the links first we need to define a density. We will assume that the redundant link all the pistons and the cylinders and the platform are made of the same material. Let

$$\mu = 1kg/m \tag{7.36}$$

In the above identity μrepresents themass per length of the redundant link and all the pistons and cylinders. Now let the radii of the cross sections of all these links be .1 meter. The density can be found as:

$$\rho = \frac{\mu}{\pi *. 1^2}\frac{kg}{m^3} = \frac{1}{\pi *. 1^2} = 100/\pi\frac{kg}{m^3} \tag{7.37}$$

Now the masses of the redundant link and all the pistons and cylinders can be found as :

$$m_1 = l_1 \times \mu = l_1\ kg \tag{7.38}$$

$$m_2 = 2 \times LC1 \times \mu = 2LC1\ kg \tag{7.39}$$

$$m_3 = 2 \times LP1 \times \mu = 2LP1\ kg \tag{7.40}$$

$$m_5 = 2 \times LP2 \times \mu = 2LP2\ kg \tag{7.41}$$

$$m_6 = 2 \times LC2 \times \mu = 2LC2\ kg \tag{7.42}$$

$$m_7 = 2 \times LP3 \times \mu = 2LP3\ kg \tag{7.43}$$

$$m_8 = 2 \times LC3 \times \mu = 2LC3\ kg \tag{7.44}$$

To find the mass of the platform first its volume must be found:

$$A_{pl} = \sqrt{3}/4{l_p}^2 \tag{7.45}$$

$$t_{pl} = .02m \tag{7.46}$$

$$V_{pl} = A_{pl} * t_{pl} = .02 * \sqrt{3}/4{l_p}^2 m^3 \tag{7.47}$$

In the above equations A_{pl}, t_{pl}, and V_{pl} are the area, thickness and volume of the platform respectively.

$$m_4 = \frac{100}{\pi} * .02 * \frac{\sqrt{3}}{4}{l_p}^2 kg = \frac{\sqrt{3}}{2\pi}{l_p}^2 kg \tag{7.48}$$

g, acceleration of gravity is taken as 9.8 m/sec^2

In this optimization problem, the kinematic equations given before act as the equality constraints. They are given here again for the sake of convenience:

$$l_1 \cos(\theta) + s_1 \cos(\theta + \theta_{12}) + l_p \frac{\sqrt{3}}{3} \cos\left(\theta + \theta_{12} + \theta_{3p} + \frac{\pi}{6}\right) - x_c = 0 \tag{7.49}$$

$$l_1 \sin(\theta) + s_1 \sin(\theta + \theta_{12}) + l_p \frac{\sqrt{3}}{3} \sin\left(\theta + \theta_{12} + \theta_{3p} + \frac{\pi}{6}\right) - y_c = 0 \tag{7.50}$$

$$l_1 \cos(\theta) + s_1 \cos(\theta + \theta_{12}) + l_p \cos\left(\theta + \theta_{12} + \theta_{3p}\right) - a - s_2 \cos(\theta_{06}) = 0 \tag{7.51}$$

$$l_1 \sin(\theta) + s_1 \sin(\theta + \theta_{12}) + l_p \sin\left(\theta + \theta_{12} + \theta_{3p}\right) - b - s_2 \sin(\theta_{06}) = 0 \tag{7.52}$$

$$l_1\cos(\theta)+s_1\cos(\theta+\theta_{ij})+l_p\cos(\theta+\theta_{12}+\theta_{3p}+\pi/3)-c-s_3\cos(\theta_{08})=0 \quad (7.53)$$

$$l_1\sin(\theta)+s_1\sin(\theta+\theta_{12})+l_p\sin(\theta+\theta_{12}+\theta_{3p}+\pi/3)-d-s_3\sin(\theta_{08})=0 \quad (7.54)$$

The inequality constraints are the limits on the joint variables. The inequality constraints are given as:

$$s_1-6.8\le 0 \quad (7.55)$$

$$3.2-s_1\le 0 \quad (7.56)$$

$$s_2-8.8\le 0 \quad (7.57)$$

$$4.2-s_2\le 0 \quad (7.58)$$

$$s_3-6.8\le 0 \quad (7.59)$$

$$3.2-s_3\le 0 \quad (7.60)$$

$$0\le\theta\le 2\pi \quad (7.61)$$

$$0\le\theta_{12}\le 2\pi \quad (7.62)$$

$$0\le\theta_{3p}\le 2\pi \quad (7.63)$$

$$0\le\theta_{06}\le 2\pi \quad (7.64)$$

$$0\le\theta_{08}\le 2\pi \quad (7.65)$$

The results of the redundancy resolution done at the position level are given in the results chapter.

7.2 Optimal Redundancy Resolution At The Velocity Level

Optimal Redundancy Resolution At The Velocity level has been done by minimizing the kinetic energy of the manipulator. Since kinetic energy depends on the velocities of the links the solution is first obtained at the velocity level. The solution at the position level is later found by integration.The total kinetic energy of this planar manipulator is found as:

$$KE = \sum_{i=1}^{8} \frac{1}{2} m_i {v_{ic}}^2 + \frac{1}{2} I_{gi} {\omega_i}^2 \tag{7.66}$$

In the above expression v_{ic} and I_g are the velocity of the center of mass and the moment of inertia about the z axis passing through the center of mass for the ith link respectively.ω_i is the angular velocity of the ith link with respect to the base frame. The velocities of the center of masses of the links 1 through 8 can be found by taking the time derivative of the position vectors. Let

$$\bar{v}_{ic} = \begin{bmatrix} v_{icx} \\ v_{icy} \\ v_{icz} \end{bmatrix} \tag{7.67}$$

Then

$$\bar{v}_{1c} = \begin{bmatrix} -l_1 \dot{\theta} sin(\theta)/2 \\ l_1 * \dot{\theta} * cos(\theta)/2 \\ 0 \end{bmatrix} \tag{7.68}$$

$$\begin{aligned} v_{2cx} = \; & -\dot{\theta}_{12} \times (LC1cos(\theta)sin(\theta_{12}) + LC1cos(\theta_{12})sin(\theta)) - \dot{\theta} \\ & \times (l_1 sin(\theta) + LC1cos(\theta)sin(\theta_{12}) \\ & + LC1cos(\theta_{12})sin(\theta)) \end{aligned} \tag{7.69}$$

$$v_{2cy} = \dot{\theta}_{12} \times (LC1cos(\theta)cos(\theta_{12}) - LC1sin(\theta)sin(\theta_{12})) + \dot{\theta} \times (l_1cos(teta) + LC1cos(\theta)cos(\theta_{12}) - LC1sin(\theta)sin(\theta_{12})) \tag{7.70}$$

$$v_{2cz} = 0 \tag{7.71}$$

$$v_{3cx} = \dot{\theta}_{12} \times (cos(\theta)sin(\theta_{12})(LP1 - s_1) + cos(\theta_{12})sin(\theta)(LP1 - s_1)) + \dot{s}_1 \times (cos(\theta)cos(\theta_{12}) - sin(\theta)sin(\theta_{12})) + \dot{\theta} \times (cos(\theta)sin(\theta_{12}) \times (LP1 - s_1) - l_1sin(\theta) + cos(\theta_{12})sin(\theta)(LP1 - s_1)) \tag{7.72}$$

$$v_{3cy} = \dot{s}_1 \times (cos(\theta)sin(\theta_{12}) + cos(\theta_{12})sin(\theta)) - \dot{\theta}_{12} \times (cos(\theta)cos(\theta_{12}) \times (LP1 - s_1) - sin(\theta)sin(\theta_{12}) \times (LP1 - s_1)) + \dot{\theta} \times (l_1cos(\theta) - cos(\theta)cos(\theta_{12})(LP1 - s_1) + sin(\theta)sin(\theta_{12}) \times (LP1 - s_1)) \tag{7.73}$$

$$v_{3cz} = 0 \tag{7.74}$$

$$\bar{v}_{4c} = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ 0 \end{bmatrix} \tag{7.75}$$

$$\bar{v}_{5c} = \begin{bmatrix} \dot{s}_2 * cos(\theta_{06}) + \dot{\theta}_{06}sin(\theta_{06})(LP2 - s_2) \\ \dot{s}_2 * sin(\theta_{06}) - \dot{\theta}_{06}cos(\theta_{06})(LP2 - s_2) \\ 0 \end{bmatrix} \tag{7.76}$$

$$\bar{v}_{6c} = \begin{bmatrix} -LC2\dot{\theta}_{06}sin(\theta_{06}) \\ LC2\dot{\theta}_{06}cos(\theta_{06}) \\ 0 \end{bmatrix} \tag{7.77}$$

$$\bar{v}_{7c} = \begin{bmatrix} \dot{s}_3cos(\theta_{06}) \\ \dot{s}_3sin(\theta_{06}) \\ 0 \end{bmatrix} \tag{7.78}$$

$$\bar{v}_{8c} = \begin{bmatrix} -LC3\dot{\theta}_{08}sin(\theta_{08}) \\ LC3\dot{\theta}_{08}cos(\theta_{08}) \\ 0 \end{bmatrix} \tag{7.79}$$

The angular velocities of the links 1 through 8 are:

$$\bar{w}_i = \begin{bmatrix} 0 \\ 0 \\ w_{iz} \end{bmatrix} \tag{7.80}$$

where

$$w_{1z} = \dot{\theta} \tag{7.81}$$

$$w_{2z} = \dot{\theta} + \dot{\theta}_{12} \tag{7.82}$$

$$w_{3z} = \dot{\theta} + \dot{\theta}_{12} \tag{7.83}$$

$$w_{4z} = \dot{\phi} \tag{7.84}$$

$$w_{5z} = \dot{\theta}_{06} \tag{7.85}$$

$$w_{6z} = \dot{\theta}_{06} \tag{7.86}$$

$$w_{7z} = \dot{\theta}_{08} \tag{7.87}$$

$$w_{8z} = \dot{\theta}_{08} \tag{7.88}$$

Now moments of inertias about the centers of masses need to be found. If the redundant link and all the pistons and cylinders are considered as thin rods the moments of inertia can be found from the following:

$$I_g = \frac{1}{12} m_{rod} {l_{rod}}^2 \tag{7.89}$$

$$I_{g1} = \frac{1}{12} m_1 L1^2 \tag{7.90}$$

$$I_{g2} = \frac{1}{12} m_2 4LC1^2 \tag{7.91}$$

$$I_{g3} = \frac{1}{12} m_3 4LP1^2 \tag{7.92}$$

$$I_{g5} = \frac{1}{12} m_5 4LP2^2 \tag{7.93}$$

$$I_{g6} = \frac{1}{12} m_6 4LC2^2 \tag{7.94}$$

$$I_{g7} = \frac{1}{12} m_7 4LP3^2 \tag{7.95}$$

$$I_{g8} = \frac{1}{12} m_8 4LC3^2 \tag{7.96}$$

To find the moment of inertia of the platform first its polar moment of inertia about the center of mass must be found. For the triangle shown below in Figure 7.1

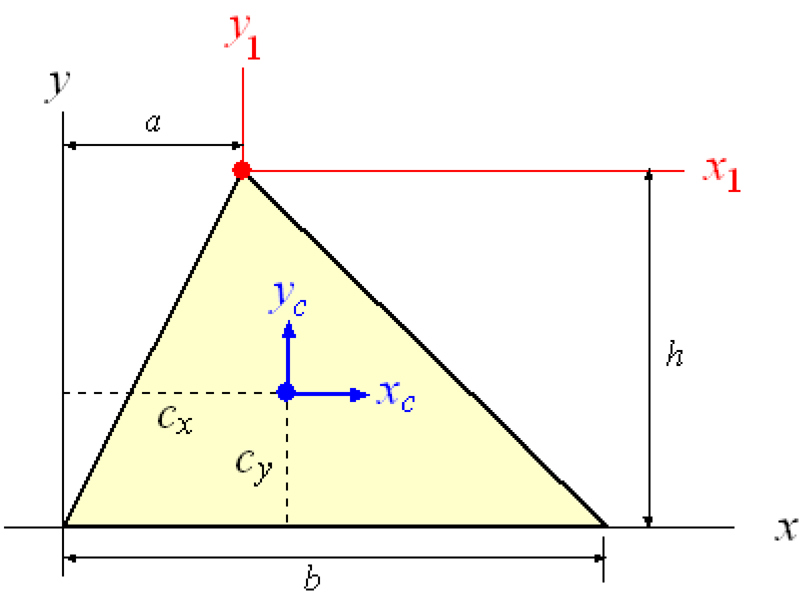


Figure 7.1: General Triangle for finding polar moment of inertia

the polar moment of inertia is given in [19] as

$$J_g = \frac{hb^3 + hab^2 + ha^2b + bh^3}{12} \tag{7.97}$$

For the platform which is in the shape of an equilateral triangle

$$h = l_p \frac{\sqrt{3}}{2} \tag{7.98}$$

$$a = \frac{l_p}{2} b = l_p \tag{7.99}$$

$$J_{g4} = \frac{l_p \frac{\sqrt{3}}{2} {l_p}^3 + l_p \frac{\sqrt{3}}{2} \frac{l_p}{2} {l_p}^2 + l_p \frac{\sqrt{3}}{2} \frac{{l_p}^2}{2} l_p + l_p (l_p \frac{\sqrt{3}}{2})^3}{12} \tag{7.100}$$

Now using the found polar moment of inertia, the mass moment of inertia can be found as follows:

$$t_{pl} = .02m$$

$$massperarea = \rho * t_{pl} \tag{7.101}$$

$$I_{g4} = J_{g4} * massperarea \tag{7.102}$$

In the above equations massperarea is the mass per unit area for the platform. The equality constraints for this optimization problem are obtained by taking the derivative of kinematic constraint equations. Because these kinematic constraint equations are too long they are given in Appendix B.

The inequality constraints for this optimization problem consist of the limits on the velocities of the joint variables. These limits for the prismatic joint variables are obtained as follows. Let s_{i_j} be the value of the ith prismatic joint variable at time step j and Δt be the time interval. Then

$$s_{i_{j+1}} = s_{i_j} + \dot{s}_i \Delta t \tag{7.103}$$

$$s_i^{min} \leq s_{i_{j+1}} \leq s_i^{max} \tag{7.104}$$

$$s_i^{min} \leq s_{i_j} + \dot{s}_i \Delta t \leq s_i^{max} \tag{7.105}$$

$$\dot{s}_i \leq \frac{s_i^{max} - s_{i_j}}{\Delta t} \tag{7.106}$$

$$\dot{s}_i \geq \frac{s_i^{min} - s_{i_j}}{\Delta t} \tag{7.107}$$

There are no limits on the velocities of the revolute joints.

The results of the redundancy resolution done at the velocity level are given in the results chapter.

7.3 Optimal Redundancy Resolution At The Acceleration Level

Optimal Redundancy Resolution at the acceleration level has been done by minimizing the total power spent by the manipulator. Since total power depends on the accelerations of the links the solution is first obtained at the acceleration level. The solutions at the velocity and position levels are later found by sucessive integration. Power is spent at the active joint variables of the manipulator. So the total power of the manipulator is found using the following equation:

$$P = \sum_{i=1}^{3} F_i \dot{s}_i + T\dot{\theta} \tag{7.108}$$

In the above equation F_i and T are the forces and torque along the degrees of freedom of the active prismatic joints and the revolute joint. These forces and the torque are found through Lagrange's equation as follows:

$$F_i + D_k = \frac{d(\frac{dKE}{d\dot{p_i}})}{dt} - \frac{dKE}{dp_i} + \frac{dPE}{dp_i} \quad ; \; i = 1,2,3; k = 1,2,3 \tag{7.109}$$

$$T + D_k = \frac{d(\frac{dKE}{d\dot{\theta}})}{dt} - \frac{dKE}{d\theta} + \frac{dPE}{d\theta}; k = 4 \tag{7.110}$$

Since $D_k = 0 \; for \; k = 1,2,3,4$

$$F_i = \frac{d(\frac{dKE}{d\dot{p_i}})}{dt} - \frac{dKE}{dp_i} + \frac{dPE}{dp_i} \quad ; \; i = 1,2,3 \tag{7.111}$$

$$T = \frac{d(\frac{dKE}{d\dot{\theta}})}{dt} - \frac{dKE}{d\theta} + \frac{dPE}{d\theta} \tag{7.112}$$

The accelerations of the joint variables appear in the terms involving the time derivatives. The expressions for $F_i\dot{s}_i$ and $T\dot{\theta}$ are too long and are therefore given in Appendix C.

The equality constraints for this problem are obtained by taking the second time derivative of the kinematic constraint equations. Since the expressions for these equations are too long they are given in Appendix D.

Limits on the accelerations of the prismatic joints are found as follows. Let s_{i_j}be the value of the ith prismatic joint variable at time step j and Δt be the time interval. Then:

$$s_{i_{j+1}} = s_{i_j} + \dot{s}_{i_j}\Delta t \tag{7.113}$$

$$s_{i_{j+2}} = s_{i_{j+1}} + \dot{s}_{i_{j+1}}\Delta t \tag{7.114}$$

$$\dot{s}_{i_{j+1}} = \dot{s}_{i_j} + \ddot{s}_{i_j}\Delta t \tag{7.115}$$

$$s_{i_{j+2}} = s_{i_j} + \dot{s}_{i_j}\Delta t + \left(\dot{s}_{i_j} + \ddot{s}_{i_j}\Delta t\right)\Delta t = s_{i_j} + \dot{s}_{i_j}\Delta t + \dot{s}_{i_j}\Delta t + \ddot{s}_{i_j}\Delta t^2 \qquad (7.116)$$
$$= s_{i_j} + 2\dot{s}_{i_j}\Delta t + \ddot{s}_{i_j}\Delta t^2$$

$$s_i^{min} \leq s_{i_{j+2}} \leq s_i^{max} \qquad (7.117)$$

$$s_i^{min} \leq s_{i_j} + 2\dot{s}_{i_j}\Delta t + \ddot{s}_{i_j}\Delta t^2 \leq s_i^{max} \qquad (7.118)$$

$$\ddot{s}_{i_j} \leq \frac{s_i^{max} - s_{i_j} - 2\dot{s}_{i_j}\Delta t}{\Delta t^2} \qquad (7.119)$$

$$\ddot{s}_{i_j} \geq \frac{s_i^{min} - s_{i_j} - 2\dot{s}_{i_j}\Delta t}{\Delta t^2} \qquad (7.120)$$

There are no limits on the accelerations of the revolute joints.

The results of the redundancy resolution done at the acceleration level are given in the results chapter.

In this study, a hybrid genetic algorithm method, which is discussed with further detail in the next chapter, has been employed to be able to find the global optimum of the optimization problems without getting stuck with the local solutions.

CHAPTER 8

OPTIMIZATION

Optimization is concerned with finding the best solution to some problem . Mathematically it is finding the largest or the smallest value of some well-defined mathematical expression or the objective function, involving all the variables or factors or aspects that have been selected for study and that can be expressed in mathematical form. In an unconstrained optimization problem there are no constraints on the variables. In a constrained optimization problem there may be equality and inequality constraints on the variables. If the objective function and all the equality and inequality constraints are linear functions of the variables then it is called a linear problemming problem. (If the objective function is a quadratic function and the equality and inequality constraints are linear functions then it is a quadratic programming problem) In a non-linear programming problem the objective function and some of the equality and inequality constraints are nonlinear functions of the variables. [5]

Consider the following nonlinear programming problem

$$Minimize\ f(\bar{x})$$

$$Subject\ to\ g_i(\bar{x}) < 0\ for\ i = 1, \dots, m$$

$$h_i(\bar{x}) = 0\ for\ i = 1, \dots, l$$

> "A vector $\bar{x}$ satisfying all the constraints is called a feasible solution to the problem. The collection of all such solutions forms the feasible region. The nonlinear programming is to find a feasible point $\bar{x}'$such that $f(\bar{x}) > f(\bar{x}')$ for each feasible point $\bar{x}$. Such a point $\bar{x}'$ is called an optimal solution to the problem. If more than one optimum exists they are referred to as alternative optimal solutions." [5]

In [6] a semi analytical method is given to find the global minimum of a nonlinear programming problem with nonnegative variables.

8.1 Genetic algorithms to find the global minimum of a nonlinear programming problem

According to [12] most methods called "Genetic Algorithms (GAs)" have at least the following elements in common:populations of chromosomes, selection according to fitness, crossover to produce new offspring, and random mutation of new offspring.

> "Each chromosome can be thought of as a point in the search space of candidate solutions. The GA processes populations of chromosomes, successively replacing one such population with another. The GA most often requires a fitness function that assigns a score (fitness) to each chromosome in the current population. The fitness of a chromosome depends on how well that chromosome solves the problem at hand."[12]

GA Operators

> "The simplest form of genetic algorithms involves three types of operators: selection, crossover and mutation. Selection operator selects chromosomes in the population for reproduction. The fitter the chromosome, the more times it is likely to be selected to reproduce. Crossover operator randomly chooses a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring."[12]

A simple Genetic Algorithm

"1. Start with a randomly generated population of n chromosomes
2. Calculate the fitness f(x) of each chromosome x in the population.
3.Repeat the following steps until n offspring have been created:

a.Select a pair of parent chromosomes from the current population, the probability of selection being an increasing function of fitness. Selection is done "with replacement", meaning that the same chromosome can be selected more than once to become a parent.

b.With probability p_c (the "crossover probability") cross over the pair at a randomly chosen point to form two offspring. If no crossover takes place, form two offspring that are exact copies of their respective parents.

c.Mutate the two offspring at each locus with probability p_m (the mutation probability) and place the resulting chromosomes in the new population.

4.Replace the current population with the new population.
5.Go to Step 2."[12]

"Each iteration of this process is called a generation. A GA is typically iterated for anywhere from 50 to 500 or more generations."[12]

Classification of Encodings

"How to encode a solution of the problem into a chromosome is a key issue when using GAs."[15] According to [15] , according to what kind of symbol is used as the alleles of a gene, the encoding methods can be classified as follows:

- Binary Encoding
- Real-number Encoding
- Integer or literal permutation encoding
- General data structure encoding

Binary encoding is not suitable for global optimization problems with too many variables. For example if a there are 8 variables in a global optimization problem and 10 bit long chromosomes are used to represent each variable it is necessary to deal with 80 bit long chromosomes. In such situations it is advised in [15] to use real-number encoding. In the present study Real-number encoding is used for the genetic algorithm since there are plenty of variables in the global optimization problems.

Selection Probability

> "This issue concerns how to determine selection probability for each chromosome.In proportional selection procedure, the selection probability of a chromosome is proportional to its fitness. This simple scheme exhibits some undesirable properties. For example, in early generations there is a tendency for a few super chromosomes to dominate the selection process; in later generations when population is largely converged, competition among chromosomes is less strong and a random search behaviour will emerge. Scaling and ranking mechanisms are proposed to mitigate these problems. Scaling method maps raw objective function values to some positive real values and the survival probability for each chromosome is determined according to these values. Ranking method ignores the actual objective function values and uses a ranking of chromosomes instead to determine survival probability." [13]

In general, the scaled fitness $\acute{f}_k$ from the raw fitness f_k for chromosome k can be expressed as follows:

$$\acute{f}_k = g(f_k) \tag{8.1}$$

where the function g(.) transforms the raw fitness into scaled fitness. The function g(.) may take different forms to yield different scaling methods. These methods can be roughly classified into two categories:

- Static Scaling
- Dynamic Scaling

"The mapping relation between the scaled fitness and raw fitness can be constant to yield static scaling methods, or it can vary according to some factors to yield dynamic scaling methods."[13] In [13] several different scaling methods are discussed in further detail. These are:

- Linear Scaling
- Dynamic Linear Scaling
- Sigma Truncation
- Power Law Scaling
- Logarithmic Scaling
- Normalizing
- Boltzmann Selection

"In the Rank Scaling method the idea is sorting the population from the best to the worst and assigning the selection probability of each chromosome according to the ranking but not its raw fitness."[13] "Two methods are in common use: linear ranking and exponential ranking. Let p_k be the selection probability for the kth chromosome in the ranking of population; the linear ranking takes the following form:"[13]

$$p_k = q - (k-1)\ x\ r \tag{8.2}$$

where q is the probability for the best chromosome. "Let q_o be the probability for the worst chromosome; the parameter r can be determined as follows:"[13]

$$r = \frac{q - q_o}{pop_{size} - 1} \tag{8.3}$$

In this study Rank scaling is employed as the selection probability method.

Selection

"The principle behind genetic algorithms is essentially Darwinian natural selection.Selection provides the driving force in a genetic algorithm. With too much force, genetic search will terminate prematurely; with too little force, evolutionary progress will be slower than necessary. The selection directs the genetic search toward promising regions in the search space."[15]

"In the Roulette Wheel Selection method the basic idea is to determine survival probability for each chromosome proportional to the fitness value."[15] "Then a model roulette wheel can be made displaying these probabilities. The selection process is based on spinning the wheel the number of times equal to population size, each time selecting a single chromosome for the new population."[15]

Other selection methods are discussed with further detail in [15]. These are:

- $(\mu + \lambda)$ Selection
- Tournament Selection

In this study roulette wheel selection method is employed as the selection method.

Crossover

"With the bit string representation a random cut-point is chosen and the offspring is generated by combining the segment of one parent to the left of the cut-point with the segment of the other parent to the right of the cut-point." [13]

With real number encoding Arithmetical Crossover is used. In this method to form a combination of two chromosomes the weighted average of two vectors x_1 and x_2 , x_1 and x_2 being two chromosomes, is calculated. Arithmetic crossover is defined as the combination of two vectors as follows:

$$x_1' = \lambda_1 x_1 + \lambda_2 x_2 \quad (8.6)$$

$$x_2' = \lambda_1 x_2 + \lambda_2 x_1 \quad (8.7)$$

According to the restriction on multipliers, it yields to three types of crossovers: convex crossover, affine crossover and linear crossover.[15] When $\lambda_1 = \lambda_2$ it is called average crossover in [15]. In this study arithmetical average convex crossover is employed as the crossover method since reel-numbered encoding is used.

Mutation

With the bit string representation some of the bits in a chromosome are flipped[12]. "Mutation can occur at each bit position in a string with some probability, usually very small."[12]

Nonuniform Mutation is given for real-number encoding. "For a given parent $\bar{x}$ if the element x_k of it is selected for mutation, the resultant offspring is $\acute{\bar{x}} = (x_1, \dots, \acute{x_k}, \dots, x_n)$, where $\acute{x_k}$ is randomly selected from the following two possible choices:"[12]

$$\acute{x_k} = x_k + \Delta(t, x_k^U - x_k) \tag{8.8}$$

$$\acute{x_k} = x_k - \Delta(t, x_k - x_k^L) \tag{8.9}$$

"The function $\Delta(\mathrm{t}, \mathrm{y})$ returns a value in the range $[0, y]$ such that the value of $\Delta(\mathrm{t}, \mathrm{y})$ approaches 0 as t increases(t is the generation number).The function $\Delta(\mathrm{t}, \mathrm{y})$ is given as follows:"[12]

$$\Delta(\mathrm{t}, \mathrm{y}) = \mathrm{yr}(1 - \frac{\mathrm{t}}{\mathrm{T}})^{\mathrm{b}} \tag{8.10}$$

"Where r is a random number from [0,1], T the maximum generation number, and b a parameter determining the degree of nonuniformity."[15]

In this study nonuniform mutation is employed as the mutation method.

Constrained Optimization

"The central problem for applying genetic algorithms to the constrained optimization is how to handle constraints because genetic operators used to manipulate the chromosomes often yield infeasible offspring." There are several techniques to handle constraints with genetic algorithms. The existing techniques can be roughly classified as follows:[13]

- Rejecting Strategy
- Repairing Strategy
- Modifying Genetic Operators Strategy
- Penalizing Strategy

Rejecting Strategy

> "This strategy discards all infeasible chromosomes created throughout the evolutionary process. This method may work reasonably well when the feasible search space is convex and it constitutes a reasonable part of the whole search space. However such an approach has serious limitations. For example when the initial consists of infeasible chromosomes only, it might be essential to improve them. Moreover, quite often the system can reach the optimum more easily if it is possible to cross an infeasible region."[13]

Repairing Strategy

"Repairing a chromosome involves taking an infeasible chromosome and generating a feasible one through some repairing procedure." This strategy depends on the existence of a deterministic repair procedure to convert an infeasible offspring into a feasible one. "The weakness of the method is in its problem dependence. For each particular problem a specific repair algorithm should be designed. Also for some problems the process of repairing infeasible chromosomes might be as complex as solving the original problem."[13]

Modifying Genetic Operator Strategy

"In this strategy problem-specific representation and specialized genetic operators are invented to maintain the feasibility of chromosomes."[13]

Penalty Strategy

"This technique transforms the constrained problem into an unconstrained problem by penalizing the infeasible solutions, in which a penalty term is added to the objective function for any violation of the constraints."[13] "Most penalty techniques belong to the class of problem-dependent approach."[13]

Hybrid Genetic Algorithms

"In hybrid genetic algorithms local optimization is incorporated as an add-on extra to the simple genetic algorithm loop of recombination and selection."[13] "With this approach local optimization is applied to each newly generated offspring to move it to a local optimum before injecting it into the population."[13] "Genetic algorithms are used to perform global exploration among a population, while local optimization methods are used to perform local exploitation around chromosomes."[13] "Because of the complementary properties of genetic algorithms and conventional optimization methods the hybrid approach often outperforms either method operating alone."[13]

In this study, hybrid genetic algorithm approach is adopted as the global optimization method.

THE PROPOSED GLOBAL OPTIMIZATION METHOD

The following operations are carried out at each iteration step of the hybrid genetic algorithm:

(i) A few different solution guesses are made at step $k = 1$ in the beginning of the procedure. A raw population is formed by defining these guesses as initial chromosomes. Later, the mature population of step $k = 1$ is obtained by maturizing the chromosomes of this raw population. Later, the procedure proceeds to step $k = 2$.

(ii) At an intermediate step k ($k>1$) of the procedure, a new raw population is obtained by applying crossing-over and mutation operations to the chromosomes in the mature population of step $k-1$. Later, the mature population of step k is obtained by maturizing the chromosomes of this raw population. Later, the procedure proceeds to step $k+1$.

(iii) Maturization is done as follows: A raw chromosome is considered as an initial solution guess and maturized by applying a gradient based local optimization method upon it. The applied gradient based method may give the global solution as well as a local solution. In other words, a maturized chromosome emerges as the representative of a global or local optimum.

(iv) The iteration is continued until k = N for a chosen N. The best member of the last maturized population is accepted as the global optimum.

HOW TO DETERMINE THE VALUES OF THE GENETIC ALGORITHM PARAMETERS

Mutation rate is supposed to be low that is how the value for that parameter was determined as .001.

To be able to apply optimal redundancy resolution for a single trajectory 400 different global optimizations are done. For different steps of a given trajectory and for different trajectories it is not practical to use different values for b (degree of nonuniformity in nonuniform mutation) and pc (probability of crossover) parameters. The values for b and pc have been obtained according to the potential energy optimization results. A tip point position was determined and genetic algorithm was run with different b and pc values for that position and b and pc values were searched that would let the global optimum for that position to be found. When b was 3 and pc was .8 global optimum was caught. The typical value for pc was already given as .7 in [15]. So different values around .7 were tried for pc. As for b, integer values were tried.

This is how it was made sure that the genetic algorithm found the global optimum. All the local optima obtained by the hybrid genetic algorithm and the potential energy values at these local optima were recorded. It was seen that the global minimum produced by the hybrid genetic algorithm was equal to the minimum of the potential energies of the obtained local optima. Later it was seen that the hybrid genetic algorithm yielded the global minimum for different tip point positions with these values of b and pc. The reason for this was that by changing the tip point position the global optimization problem did not change by much.

8.2 Local Optimization

In what follows Karush-Kuhn-Tucker Optimality Conditions, Successive Quadratic Programming Approach, One Dimensional Unconstrained Optimization, Multi Dimensional Constrained Local Optimization, and Multi Dimensional unconstrained local optimization are discussed.

Karush-Kuhn-Tucker Optimality Conditions

Consider the Problem P given below:

$$Minimize\ f(\bar{x})$$
$$subject\ to\ g_i(\bar{x}) \leq 0 \qquad for\ i = 1, \dots, m$$
$$h_i(\bar{x}) = 0 \qquad for\ i = 1, \dots, l$$
$$\bar{x} \in X$$

Let $\bar{x}$ solve the problem stated above locally. Then the following conditions must be satisfied:

$$\nabla f(\bar{x}) + \sum_{i=1}^{m} u_i \nabla g_i(\bar{x}) + \sum_{i=1}^{l} v_i \nabla h_i(\bar{x}) = \bar{0} \tag{8.11}$$

$$u_i g_i(\bar{x}) = 0 \qquad for\ i = 1, \dots, m \tag{8.12}$$

$$u_i \geq 0 \qquad for\ i = 1, \dots, m \tag{8.13}$$

In the above conditions ∇ stands for gradient. There is no condition on v_i.[5]

The conditions at optimality given above are known as the Karush-Kuhn-Tucker Optimality Conditions. There is a globally convergent numerical algorithm called as MSQP(Merit Sequential Quadratic Programming) which solves the above conditions together with the original equality and inequality constraints in the nonlinear programming problem starting from any given initial guess. For the logic behind this algorithm to be better understood first sucessive quadratic programming must be explained.

Successive Quadratic Programming Approach:

To present the concept of this method, consider the equality-constrained nonlinear problem , where $\bar{x} \epsilon R^n$, and all functions are assumed to be continuously twice differentiable.

$$P: \quad Minimize \quad f(\bar{x})$$

$$subject\ to\ h_i(\bar{x}) = 0, \qquad i = 1, \dots, l.$$

The extension for including inequality constraints is motivated by the following analysis for the equality-constrained case and is considered subsequently.

The KKT optimality conditions for Problem P require a primal solution $\bar{x}\ \epsilon\ R^n$ and a Lagrange multiplier vector $\bar{v}\ \epsilon\ R^l$ such that

$$\nabla f(\bar{x}) + \sum_{i=1}^{l} v_i \nabla h_i(\bar{x}) = 0 \tag{8.14}$$

$$h_i(\bar{x}) = 0, \quad i = 1, \dots, l.$$

"Let us write this system of equations more compactly as $W(\bar{x}, \bar{v}) = 0.$ We now use the Newton-Raphson method to solve the above set of equations."[5] Hence we solve

$$W(\bar{x}_k, \bar{v}_k) + \nabla W(\bar{x}_k, \bar{v}_k) \begin{bmatrix} \bar{x} - \bar{x}_k \\ \bar{v} - \bar{v}_k \end{bmatrix} = 0 \tag{8.15}$$

to determine the next iterate $(\bar{x}, \bar{v}) = (\bar{x}_{k+1}, \bar{v}_{k+1})$, where ∇W denotes the Jacobian of W. "Defining $\nabla^2 L(\bar{x}_k) = \nabla^2 f(\bar{x}_k) + \sum_{i=1}^{l} v_{ki} \nabla^2 h_i(\bar{x}_k)$ to be the usual Hessian of the Lagrangian at $\bar{x}_k$with the Lagrange multiplier vector $\bar{v}_k$, and letting $\nabla \bar{h}$ denote the Jacobian of $\bar{h}$ comprised of rows $\nabla h_i(\bar{x})^{\mathrm{t}}$ for $\mathrm{i} = 1, \dots, \mathrm{l}$ we have the following equality"[5]

$$\nabla W(\bar{x}_k, \bar{v}_k) = \begin{bmatrix} \nabla^2 L(\bar{x}_k) & \nabla \bar{h}(\bar{x}_k)^t \\ \nabla \bar{h}(\bar{x}_k) & 0 \end{bmatrix} \tag{8.16}$$

Using (8.14) and (8.16) we can rewrite (8.15) as

$$\nabla^2 L(\bar{x}_k)(\bar{x} - \bar{x}_k) + \nabla\bar{h}(\bar{x}_k)^t(\bar{v} - \bar{v}_k) = -\nabla f(\bar{x}_k) - \nabla\bar{\mathrm{h}}(\bar{x}_k)^t\bar{v}_k$$

$$\nabla\bar{h}(\bar{x}_k)(\bar{x} - \bar{x}_k) = -\bar{h}(\bar{x}_k)$$

Substituting $\bar{d} = \bar{x} - \bar{x}_k$, this in turn can be rewritten as

$$\nabla^2 L(\bar{x}_k)\bar{d} + \nabla\bar{h}(\bar{x}_k)^t\bar{v} = -\nabla f(\bar{x}_k) \qquad (8.17)$$

$$\nabla\bar{h}(\bar{x}_k)\bar{d} = -\bar{h}(\bar{x}_k)$$

We can now solve for $(\bar{d}, \bar{v}) = (\bar{d}_k, \bar{v}_{k+1})$ using this system. “Setting $\bar{x}_{k+1} = \bar{x}_k + \bar{d}_k$, we then increment k by 1 and repeat this process until $\bar{d} = \bar{0}$ happens to solve (8.17). When this occurs we shall have found a KKT solution for problem P.”[5]

“Now instead of adopting the foregoing process to find any KKT solution for P, we can instead employ a quadratic minimization subproblem whose optimality conditions duplicate (8.17).”[5] Such a quadratic program is stated below:

$$QP(\bar{x}_k, \overline{v_k}): \quad Minimize \quad f(\bar{x}_k) + \nabla f(\bar{x}_k)^t\bar{d} + \frac{1}{2}\bar{d}^t\nabla^2 L(\bar{x}_k)\bar{d} \qquad (8.18)$$

$$subject\ to \ \ h_i(\bar{x}_k) + \nabla h_i(\bar{x}_k)^{\mathrm{t}}\bar{d} = 0, \qquad i = 1, \dots, l$$

“We now consider the inclusion of inequality constraints $g_i(\bar{x}) \leq 0, i = 1, \dots., m$ in Problem P, where g_i are continuously twice differentiable for i = 1,…,m. This revised problem is restated below.”[5]

$$P: \quad Minimize \quad f(\bar{x})$$

$$subject\ to \ \ g_i(\bar{x}) \leq 0, \qquad i = 1, \dots, m$$

$$h_i(\bar{x}) = 0, \ \ i = 1, \dots, l$$

“For this instance, given an iterate $(\bar{x}_k, \bar{u}_k\bar{v}_k), where\ \bar{u}_k \geq \bar{0}$ and $\bar{v}_k$ are, respectively the Lagrange multiplier estimates for the inequality and the equality constraints, we consider the following quadratic programming subproblem as a direct extension of (8.18).”[5]

$$QP(\bar{x}_k, \bar{u}_k\bar{v}_k): \quad Minimize \quad f(\bar{x}_k) + \nabla f(\bar{x}_k)^t\bar{d} + \frac{1}{2}\bar{d}^t\nabla^2 L(\bar{x}_k)\bar{d} \qquad (8.19)$$

$$subject\ to\ g_i(\bar{x}_k) + \nabla g_i(\bar{x}_k)^{\mathrm{t}}\bar{d} \leq 0, \qquad i = 1, \dots, m$$

$$h_i(\bar{x}_k) + \nabla h_i(\bar{x}_k)^{\mathrm{t}}\bar{d} = 0, \qquad i = 1, \dots, l$$

where $\nabla^2 L(\bar{x}_k) = \nabla^2 f(\bar{x}_k) + \sum_{i=1}^{m} u_{ki}\nabla^2 g_i(\bar{x}_k) + \sum_{i=1}^{l} v_{ki}\nabla^2 h_i(\bar{x}_k)$

> "If $\bar{d}_k$ solves $QP(\bar{x}_k, \bar{u}_k, \bar{v}_k)$ with Lagrange multipliers $\bar{u}_{k+1}$ and $\bar{v}_{k+1}$and if $\bar{d}_k = \bar{0}$, then $\bar{x}_k$along with $(\bar{u}_{k+1}, \bar{v}_{k+1})$yields a KKT solution for the original Problem P. Otherwise we set $\bar{x}_{k+1} = \bar{x}_k + \bar{d}_k$, increment k by 1 and repeat the process. It can be shown that if $\bar{x}$ is a regular KKT solution which together with $(\bar{u}, \bar{v})$ satisfies the KKT optimality conditions and if $(\bar{x}_k, \bar{u}_k, \bar{v}_k)$ is initialized sufficiently close to $(\bar{x}, \bar{u}, \bar{v})$ the foregoing iterative process will converge quadratically to $(\bar{x}, \bar{u}, \bar{v})$."[5]

"A principal disadvantage of the SQP method described thus far is that convergence is guaranteed only when the algorithm is initialized sufficiently close to a desirable solution, whereas, in practice this condition is usually difficult to realize."[5] The following merit function SQP algorithm is a globally convergent variant of the SQP algorithm.

Summary of the Merit Function SQP algorithm(MSQP)

"Initialization Put the iteration counter at k = 1 and select a starting solution$\bar{x}_k$. Also, select a positive definite approximation B_k to the Hessian $\nabla^2 L(\bar{x}_k)$ defined with respect to some Lagrange multipliers $\bar{u}_k \geq 0\ and\ \bar{v}_k$ associated with the inequality and equality constraints , respectively of Problem P."[5]

"Main Step Solve the quadratic programming subproblem QP given by (6) with $\nabla^2 L(\bar{x}_k)$ replaced by B_k and obtain a solution $\bar{d}_k$ along with Lagrange multipliers $(\bar{u}_{k+1}, \bar{v}_{k+1})$. If $\bar{d}_k = 0$ then stop with $\bar{x}_k$ as a KKT solution for problem P having Lagrange multipliers $(\bar{u}_{k+1}, \bar{v}_{k+1})$. Otherwise, find $\bar{x}_{k+1} = \bar{x}_k + \lambda_k\bar{d}_k$,where$\lambda_k$ minimizes $F_E(\bar{x}_k + \lambda\bar{d}_k)$over $\lambda \epsilon R, \lambda \geq 0.$ Update B_k to a positive definite matrix B_{k+1}. Increment k by 1 and repeat the Main Step."[5] $F_E(\bar{x})$ is given as follows:

$$F_E(\bar{x}) = \mathrm{f}(\bar{x}) + \mu[\sum_{i=1}^{m} \max\{0, g_i(\bar{x})\} + \sum_{i=1}^{l} |h_i(\bar{x})|]$$

where $\mu \geq \max\{u_1, \dots, u_m, |v_1|, \dots, |v_l|\}$.[5]

A scheme for updating B_k to a positive definite matrix B_{k+1} is given in [14] as follows:

$$p = \bar{x}_{k+1} - \bar{x}_k$$

$$\nabla fobjlagr(\bar{x}_{k+1}) = \nabla f(\bar{x}_{k+1}) + \sum_{i=1}^{m} u_{k+1i} \nabla g_i(\bar{x}_{k+1}) + \sum_{i=1}^{l} v_{k+1i} \nabla h_i(\bar{x}_{k+1})$$

$$\nabla fobjlagr(\bar{x}_k) = \nabla f(\bar{x}_k) + \sum_{i=1}^{m} u_{k+1i} \nabla g_i(\bar{x}_k) + \sum_{i=1}^{l} v_{k+1i} \nabla h_i(\bar{x}_k)$$

$$q = \nabla fobjlagr(\bar{x}_{k+1}) - \nabla fobjlagr(\bar{x}_k)$$

$$If\ p^t q \ \geq\ .2 * p^t * B_k * p\ then\ \theta_{BFGS} = 1\ Else\ \theta_{BFGS} = .8 * p^t * B_k * p/(p^t * B_k * p - p^t * q)$$

$$N = \theta_{BFGS} * q + (1 - \theta_{BFGS}) * B_k * p$$

$$B_{k+1} = B_k + \frac{N * N^t}{N^t * p} - \frac{B_k * p * p^t * B_k}{p^t * B_k * p}$$

To solve the QP subproblem in the MSQP algorithm local constrained optimization must be employed. To be able to find λ_k which minimizes $F_E\left(\overline{x_k} + \lambda \bar{d}_k\right)$one dimensional unconstrained optimization must be employed. In the following, both of these two concepts are explained in detail.

<u>One Dimensional Unconstrained Optimization</u>

> “One dimensional search is the backbone of many algorithms for solving a nonlinear programming problem. Many nonlinear programming algorithms proceed as follows. Given a point $\bar{x}_k$, find a direction vector $\bar{d}_k$ and then a suitable step size λ_k , yielding a new point $\bar{x}_{k+1} = \bar{x}_k + \lambda_k \bar{d}_k$; the process is then repeated. Finding the step size λ_k involves solving the subproblem to minimize $f\left(\bar{x}_k + \lambda \bar{d}_k\right)$ which is a one-dimensional search problem in the variable λ. Consider a function θ of one variable λ to be minimized. One approach to minimizing θ is to set the derivative θ′ equal to 0 and then solve for λ. Note, however that θ is usually defined implicitly in terms of a function f of several variables. In particular, given the vectors $\bar{x}$ and $\bar{d}, \theta(\lambda) = f(\bar{x} + \lambda \bar{d})$. If f is not differentiable, then θwill not be differentiable. If f is differentiable then $\theta'(\lambda) = \bar{d}^t \nabla f(\bar{x} + \lambda \bar{d})$. Therefore to find a point λ with $\theta'(\lambda) = 0$ we have to solve the equation $\bar{d}^t \nabla f\left(\bar{x} + \lambda \bar{d}\right) = 0$, which is usually nonlinear in λ. Furthermore λ satisfying $\theta'(\lambda) = 0$ is not necessarily a minimum; it may be a local minimum, a

local maximum, or even a saddle point. For these reasons minimizing θ by letting its derivative be equal to zero is avoided. Instead, some numerical techniques are employed for minimizing the function θ."[5]

Line Search Without Using Derivatives

In what follows golden section method is presented as the only line search method without using derivatives.

GOLDEN SECTION METHOD

"Consider the line search problem to minimize $\theta(\lambda)$ subject to $a \leq \lambda \leq b$. Since the exact location of the minimum of θ over $[a, b]$ is not known , this interval is called the interval of uncertainty."[5]

Following is a summary of the golden section method for minimizing a function over the interval $[a_1, b_1]$.[5]

"Initialization Step: Choose an allowable final length of uncertainty $\ell > 0$. Let $[a_1, b_1]$ be the initial interval of uncertainty, and let $\rho_1 = a_1 + (1 - \alpha)(b_1 - a_1)$and $\mu_1 = a_1 + \alpha(b_1 - a_1)$, where $\alpha = .618$. Evaluate $\theta(\rho_1)$ and $\theta(\mu_1)$, let k = 1, and go to the Main Step.

Main Step:

1. If $b_k - a_k < l$, stop; the optimal solution lies in the interval $[a_k, b_k]$. Otherwise, if $\theta(\rho_k) > \theta(\mu_k)$, go to step 2; and if $\theta(\rho_k) \leq \theta(\mu_k)$, go to Step 3.
2. Let $a_{k+1} = \rho_k$ and$b_{k+1} = b_k$. Furthermore, let $\rho_{k+1} = \mu_k$ and let $\mu_{k+1} = a_{k+1} + \alpha(b_{k+1} - a_{k+1})$. Evaluate $\theta(\mu_{k+1})$ and go to Step 4.
3. Let $a_{k+1} = a_k$ and $b_{k+1} = \mu_k$. Furthermore, let $\mu_{k+1} = \rho_k$, and let $\rho_{k+1} = a_{k+1} + (1 - \alpha)(b_{k+1} - a_{k+1})$. Evaluate $\theta(\rho_{k+1})$ and go to Step 4.
4. Replace k by k + 1 and go to Step 1."[5]

In this study golden section method is employed as the one dimensional unconstrained optimization method since it yields a global solution.

Line Search Using Derivatives

In what follows, bisection search method and newton's method are presented as the two line search methods using derivatives.

BISECTION SEARCH METHOD

"Initialization Step: Let $[a_{1,}b_1]$be the initial interval of uncertainty, and let ℓ be the allowable final interval of uncertainty. Let n be the smallest positive integer such that $(\frac{1}{2})^n \leq \ell/(b_1 - a_1)$. Let k = 1 and go to the Main Step.

Main Step:

1. Let $\lambda_k = \left(\frac{1}{2}\right)(a_k + b_k)$and evaluate $\theta'(\lambda_k)$.If $\theta'(\lambda_k) = 0$, stop; λ_k is an optimal solution. Otherwise, go to Step 2 if$\theta'(\lambda_k) > 0$, and go to Step 3 if $\theta'(\lambda_k) < 0$.
2. Let $a_{k+1} = a_k$ and $b_{k+1} = \lambda_k$. Go to Step 4.
3. Let $a_{k+1} = \lambda_k$and $b_{k+1} = b_k. Go\ to\ Step$ 4.
4. If k = n, stop; the minimum lies in the interval $[a_{n+1,}b_{n+1}]$. Otherwise, replace k by k+1 and repeat Step1."[5]

This method is not employed in this study since it can yield a local solution or it may yield an interval rather than a single point.

NEWTON'S METHOD

"Newton's method is based on exploiting the quadratic approximation of the function θ at a given point λ_k."[5] This quadratic approximation q is given by

$$q(\lambda) = \theta(\lambda_k) + \theta'(\lambda_k)(\lambda - \lambda_k) + \frac{1}{2}\theta''(\lambda_k)(\lambda - \lambda_k)^2$$

"The point $\lambda_{k+1}$is taken to be the point where the derivative of q is equal to zero." [5] This yields $\theta'(\lambda_k) + \theta''(\lambda_k)(\lambda_{k+1} - \lambda_k) = 0, so\ that$

$$\lambda_{k+1} = \lambda_k - \frac{\theta'(\lambda_k)}{\theta''(\lambda_k)}$$

"The procedure is terminated when $|\lambda_{k+1} - \lambda_k| < \varepsilon$ or when $|\theta'(\lambda_k)| < \varepsilon$, where ε is a prespecified termination scalar."[5]

"Note that the above procedure can only be applied for twice differentiable functions. Furthermore, the procedure is well defined only if $\theta''(\lambda_k) \neq 0$ for each k."[5]

In this study this method is not employed since it may yield a local solution.

Multi Dimensional Constrained Local Optimization

In what follows penalty functions are presented as the only multi dimensional constrained local optimization method.

Penalty Functions

"Methods using penalty functions transform a constrained problem into a single unconstrained problem or into a sequence of unconstrained problems. The constraints are placed into the objective function via a penalty parameter in a way that penalizes any violation of the constraints."[5] "To motivate penalty functions, consider the following problem having the single constraint $h(\bar{d})$:"[5]

$$\text{Minimize} \quad f(\bar{d})$$
$$\text{subject to} \quad h(\bar{d}) = 0.$$

"Suppose that this problem is replaced by the following unconstrained problem, where $\mu > 0$ is a large number:"[5]

$$\text{Minimize} \quad f(\bar{d}) + \mu h^2(\bar{d})$$
$$\text{subject to} \quad \bar{d} \in R^n$$

"It can intuitively be seen that an optimal solution to the above problem must have $h^2(\bar{d})$ close to zero, because otherwise, a large penalty $\mu h^2(\bar{d})$ will be incurred."[5]

"Now consider the following problem having single inequality constraint g($\bar{d}$) $\leq$ 0:"[5]

$$\text{Minimize} \quad f(\bar{d})$$
$$\text{subject to} \quad g(\bar{d}) \leq 0.$$

"It is clear that the form $f(\bar{d}) + \mu g^2(\bar{d})$ is not appropriate, since a penalty will be incurred whether $g(\bar{d}) < 0$ or $g(\bar{d}) > 0$. A penalty is desired only if the point $\bar{d}$ is not feasible, that is , if $g(\bar{d}) > 0$."[5] A suitable unconstrained problem is therefore given by :

$$\text{Minimize} \quad f(\bar{d}) + \mu\, maximum\{0, g(\bar{d})\}$$
$$\text{subject to} \quad \bar{d} \epsilon R^n.$$

If $g(\bar{d}) \leq 0$, then $max\{0, g(\bar{d})\} = 0, and\ no\ penalty\ is\ incurred.$ On the other hand, if If $g(\bar{d}) > 0$ then $max\{0, g(\bar{d})\} > 0$ and the penalty term $\mu g(\bar{d})$ is realized.

“In general, a suitable penalty function must incur a positive penalty for infeasible points and no penalty for feasible points.”[5] “If the constraints are of the form $g_i(\bar{d}) \leq 0 for\ i = 1, \dots, m$ and $h_i(\bar{d}) = 0\ for\ i = 1, \dots, l$ a suitable penalty function α is defined by

$$\alpha(\bar{d}) = \sum_{i=1}^{m} \phi[g_i(\bar{d})] + \sum_{i=1}^{l} \psi[h_i(\bar{d})],$$

where ϕ and ψ are continuous functions satisfying the following:”[5]

$\phi(y) = 0$ if $y \leq 0$ and $\phi(y) > 0$ if $y > 0$

$\psi(y) = 0$ if $y = 0$ and $\psi(y) > 0$ if $y \neq 0$

Typically, ϕ and ψ are of the forms

$$\phi(y) = [max\{0, y\}]^p$$

$$\psi(y) = |y|^p$$

where p is a positive integer. Thus, the penalty function α is usually of the form

$$\alpha(\bar{d}) = \sum_{i=1}^{m} [max\{0, g_i(\bar{d})\}]^p + \sum_{i=1}^{l} |h_i(\bar{d})|^p.$$

The function $f(\bar{d}) + \mu\alpha(\bar{d})$is referred to as the auxiliary function.

“Returning to the primal problem:

Minimize	$f(\bar{d})$	
subject to	$g_i(\bar{d}) \leq 0$	for i = 1,…,m
	$h_i(\bar{d}) = 0$	for i = 1,…,l
	$\bar{d} \epsilon D,$	

where f,g_1,…,g_m, h,…,h_l are continuous functions on R^n and D is a nonempty set in R^n, suppose that the problem has a feasible solution, and let αbe a continuous function given by

$\alpha(\bar{d}) = \sum_{i=1}^{m} \phi[g_i(\bar{d})] + \sum_{i=1}^{l} \psi[h_i(\bar{d})],$

where ϕ and ψ are continuous functions satisfying the following:

$\phi(y) = 0 \quad \text{if } y \leq 0 \qquad \text{and} \qquad \phi(y) > 0 \quad \text{if } y > 0$

$\psi(y) = 0 \quad \text{if } y = 0 \qquad \text{and} \qquad \psi(y) > 0 \quad \text{if } y \neq 0$

Furthermore, suppose that for each μ there exists a solution $\bar{d}_\mu \epsilon\, D$ to the problem to minimize $f(\bar{d}) + \mu\alpha(\bar{d})$ subject to $\bar{d}\epsilon D$. The limit $\bar{d}$ of any convergent subsequence of $\{\bar{d}_\mu\}$ is an optimal to the original problem , and $\mu\alpha(\bar{d}_\mu) \to 0\ as\ \mu \to \infty$. The optimal solution $\bar{d}_\mu$ to the problem to minimize $f(\bar{d}) + \mu\alpha(\bar{d})$ subject to $\bar{d}\epsilon D$ can be made arbitrarily close to the feasible region by choosing μ large enough. The optimal points $\{d_\mu\}$ are generally infeasible but as the penalty parameter μ is made large, the points generated approach an optimal solution from outside the feasible region. Hence this technique is also referred to as an exterior penalty function method."[5]

"Most algorithms using penalty functions employ a sequence of increasing penalty parameters. With each new value of the penalty parameter an optimization technique is employed, starting with the optimal solution obtained for the parameter value chosen previously."[5]

Summary of penalty function methods:

"Initialization Step: Let $\varepsilon > 0$ be a termination scalar. Choose an initial point $\overline{d_1}$, a penalty parameter $\mu_1 > 0$, and a scalar $\beta > 1$. Let k = 1, and go to the Main Step.

Main Step:

1. Starting with $\bar{d}_k$, solve the following problem:
   Minimize $f(\bar{d}) + \mu_k\alpha(\bar{d})$
   subject to $\bar{d} \epsilon D$
   Let $\bar{d}_{k+1}$ be an optimal solution and go to Step 2.
2. If $\mu_k\alpha(\bar{d}_{k+1}) < \varepsilon$ stop; otherwise, let $\mu_{k+1} = \beta\mu_k$, replace k by k+1 and go to Step1."[5]

"For the types of penalty functions considered thus far, it is necessary to make the penalty parameter infinitely large in a limiting sense to recover an optimal solution."[5] However, it is also possible to design penalty functions which are capable of recovering an exact optimal solution for reasonable finite values of the penalty parameter μ without the need for μ to approach infinity. Below two penalty functions are presented that possess this property and are therefore known as exact penalty functions.[5]

THE ABSOLUTE VALUE PENALTY FUNCTION

"The absolute value penalty function conforms with the typical form

$$\alpha(\bar{d}) = \sum_{i=1}^{m}\left[max\{0, g_i(\bar{d})\}\right]^p + \sum_{i=1}^{l}\left|h_i(\bar{d})\right|^p$$

with p = 1."[5] It is given as:

$$F_E(\bar{d}) = f(\bar{d}) + \mu[\sum_{i=1}^{m} max\{0, g_i(\bar{d})\} + \sum_{i=1}^{l}\left|h_i(\bar{d})\right|]$$

AUGMENTED LAGRANGIAN PENALTY FUNCTION

"Augmented Lagrangian (ALAG) penalty function not only recovers an exact optimum for finite penalty parameter values but also enjoys the property of being differentiable."[5] "ALAG penalty function is given as:

$$F_{ALAG}(\bar{d}, \bar{u}, \bar{v}) = f(\bar{d}) + \sum_{i=1}^{l} v_i\, h_i(\bar{d}) + \sum_{i=1}^{l} \mu eq_i h_i^2(\bar{d})$$
$$+ \sum_{i=1}^{m} \mu ineq_i max^2\left\{g_i(\bar{d}) + \frac{u_i}{2\mu ineq_i}, 0\right\} - \sum_{i=1}^{m} \frac{u_i^2}{4\mu ineq_i}$$

where $\bar{u}\ and\ \bar{v}$ are the lagrange multipliers corresponding to the inequality and equality constraints respectively at the optimal solution."[5] The fundamental schema of this type of algorithm is as follows:

"Initialization Step: Select some initial Lagrangian multiplier vector $\bar{u}\ and\ \bar{v}$ and positive values $\mu eq_1, \dots, \mu eq_l$ and $\mu ineq_1, \dots, \mu ineq_m$ for the penalty parameters. Let $\overline{d_0}$ be a null vector, and denote VIOL($\overline{d_0}$) $= \infty$, where for any $\bar{d} \epsilon R^n$, $VIOL(\bar{d}) \equiv$ max{$\left|h_i(\bar{d})\right|$, max{$0, g_j(\bar{d})$}: $i = 1, \dots, l\ \ j = 1, \dots, m$} is a measure of constraint violations. Put k = 1 and proceed to the inner loop of the algorithm.

Inner Loop: **Penalty Function Minimization** Solve the unconstrained problem to minimize$F_{ALAG}(\bar{d}, \bar{u}, \bar{v})$ subject to $\bar{d} \epsilon R^n$, and let $\bar{d}_k$ denote the optimal solution obtained. If $VIOL(\bar{d}_k)$is less than some tolerance ε stop, with $\bar{d}_k$ as a KKT point. Otherwise, if $VIOL(\bar{d}_k) \leq (\frac{1}{4})VIOL(\bar{d}_{k-1})$, proceed to the outer loop. On the other hand, if $VIOL(\bar{d}_k) > 1/4VIOL(\bar{d}_{k-1})$, then for each constraint i = 1,…,l and j = 1,…,m for which $\left|h_i(\bar{d})\right| > 1/4VIOL(\bar{d}_{k-1})$ and $max\{0, g_j(\bar{d})\} > 1/4VIOL(\bar{d}_{k-1})$, replace the corresponding penalty parameter μ by $10\,\mu$ and repeat this inner loop step.

Outer Loop: **Lagrange Multiplier Update**Replace $\bar{v}\ by\ \bar{v}_{new}$, and $\bar{u}\ by\ \bar{u}_{new}$ where

$(v_{new})_i = v_i + 2\mu eq_i h_i(\bar{d}_k)$ for i = 1,…,l.
$(u_{new})_i = u_i + \max\{2\mu ineq_i g_i(\bar{d}_k), -u_i\}$ for i = 1,…,m.
Increment k by 1, and return to the inner loop."[5]

In this study ALAG penalty function method is employed as the MultiDimensional Constrained Local Optimization method.

In order to solve the unconstrained problem emerging in the inner loop of the ALAG penalty function method a multi dimensional unconstrained local optimization algorithm must be employed.

<u>Multi Dimensional unconstrained local optimization</u>

In what follows method of steepest descent method ,method of newton and Levenberg-Marquardt method are presented as three methods for multi dimensional unconstrained local optimization.

Method of Steepest Descent

"A vector $\bar{s}$ is called a direction of descent of a function f at $\bar{d}$if there exists a $\delta > 0$ such that $f(\bar{d} + \lambda\bar{s}) < f(\bar{d})$for all $\lambda \epsilon (0, \delta)$."[5] "The method of steepest descent moves along the direction $\bar{s}$. If f is differentiable at $\bar{d}$ with a nonzero gradient, then $-\nabla f(\bar{d})$ is the direction of steepest descent. Given a point $\bar{d}$, the steepest descent algorithm proceeds by performing a line search along the direction $-\nabla f(\bar{d})$."[5]

"Initialization Step: Let $\varepsilon > 0$ be the termination scalar. Choose a starting point $\overline{d_1}$, let k = 1 , and go to the Main Step.
Main Step:
If $\|\nabla f(\bar{d}_k)\| < \varepsilon$, stop; otherwise, let $\bar{s_k} = -\nabla f(\bar{d}_k)$, and let $\lambda_k$be an optimal solution to the problem to minimize $f(\bar{d}_k + \lambda\bar{s_k})$ subject to $\lambda \geq 0$. Let $\bar{d}_{k+1} = \bar{d}_k + \lambda_k\bar{s_k}$, replace k by k+1 and repeat the Main Step."[5]

Method of Newton

"To motivate the procedure, consider the following approximation q at a given point $\overline{d}_k$:

$$q(\bar{d}) = f(\bar{d}_k) + \nabla f(\bar{d}_k)^t(\bar{d} - \bar{d}_k) + \frac{1}{2}(\bar{d} - \bar{d}_k)^t H(\bar{d}_k)(\bar{d} - \bar{d}_k)$$

where $H(\bar{d}_k)$ is the Hessian matrix of f at $\bar{d}_k$."[5] "A necessary condition for a minimum of the quadratic approximation q is that $\nabla q(\bar{d}) = \bar{0}$, or $\nabla f(\bar{d}_k) + H(\bar{d}_k)(\bar{d} - \bar{d}_k) = \bar{0}$."[5] Assuming that the inverse of H($\bar{d}_k$) exists, the successor point $\bar{d}_{k+1}$ is given by

$$\bar{d}_{k+1} = \bar{d}_k - H(\bar{d}_k)^{-1}\nabla f(\bar{d}_k)$$

This procedure is continued until $\|\nabla f(\bar{d}_k)\|$is smaller than a termination scalar ε .

"If Newton's method is initialized close enough to a local minimum $\bar{d}$ with a positive definite Hessian H($\bar{d}$), then it converges quadratically to this solution."[5] "The method may not be defined because of the singularity of $H(\bar{d}_k)$ at a given point $\bar{d}_k$ or the search direction $\bar{s_k} = H(\bar{d}_k)^{-1}\nabla f(\bar{d}_k)$ may not be a descent direction or even if it is a descent direction a unit step size might not give a descent in f."[5]

Modification of Newton's Method: Levenberg-Marquardt

"Through a modification of Newton's method it is possible to design a well-defined algorithm that converges to a point of zero gradient irrespective of the starting solution."[5]

> "Given $\bar{d}$ consider the direction $\bar{s} = -B\nabla f(\bar{d})$where B is a symmetric positive definite matrix. The matrix B is specified as $(\kappa I + H)^{-1}$ where $H = H(\bar{d})$. The algorithmic scheme of determining the new iterate $\bar{d}_{k+1}$ from an iterate $\bar{d}_k$ according to the solution of the system
>
> $$\left(\kappa_k I + H(\bar{d}_k)\right)(\bar{d}_{k+1} - \bar{d}_k) = -\nabla f(\bar{d}_k)$$
>
> is known as a Levenberg-Marquardt method."[5]

The rest of this algorithm is as follows.

> "Given an iterate $\bar{d}_k$ and a parameter $\kappa_k > 0$, first ascertain the positive definiteness of $\kappa_k I + H(\bar{d}_k)$ by attempting to construct its Cholesky factorization. If this is unsuccessful , then multiply κ_k by a factor of 4 and repeat until such a factorization is available. Then solve the system via $\left(\kappa_k I + H(\bar{d}_k)\right)(\bar{d}_{k+1} - \bar{d}_k) = -\nabla f(\bar{d}_k)$ to obtain $\bar{d}_{k+1}$. Compute f($\bar{d}_{k+1}$) and determine R_k as the ratio of the actual decrease f($\bar{d}_k$) - f($\bar{d}_{k+1}$) in f to its predicted decrease q($\bar{d}_k$) - q($\bar{d}_{k+1}$) as foretold by the quadratic approximation q

to f at $\bar{d}=\bar{d}_k$. If $R_k < .25$, put $\kappa_{k+1} = 4\kappa_k$; if $R_k > .75$ put $\kappa_{k+1} = \kappa_k/2$; otherwise put $\kappa_{k+1} = \kappa_k$. Furthermore, in case $R_k \leq 0$so that no improvement in f is realized, reset $\bar{d}_{k+1} = \bar{d}_k$; or else, retain the computed $\bar{d}_{k+1}$. Increment k by 1 and reiterate until convergence to a point of zero gradient is obtained."[5]

In this study Levenberg-Marquardt is employed as the Multi Dimensional unconstrained local optimization method ince it is a globally convergent method.

CHAPTER 9

RESULTS

Firstly, the manipulator was made to follow trajectory 1. For this simulation the following values have been used for L1,Lp, LP1, LC1, LP2, LC2, LP3, LC3,a, b, c and d.

$$L1 = 2; Lp = 2; LP1 = 2; LC1 = 1.5; LP2 = 2.5; LC2 = 2; LP3 = 2; LC3 = 1.5; a = 7; b = 1; c = 1; d = 7$$

The following initial conditions have been used for Kinetic Energy and Total Power Optimizations:

$$p1 = 4.8204\ meters\ p2 = \ 5.3629\ meters\ p3 = \ 3.3358\ meters$$

$$\theta = \frac{pi}{3} ra$$

$$\theta_{12} = -0.0863\ ra;\ \theta_{3p} = -0.6991\ ra; \theta_{06} = 1.8170\ ra; \theta_{08} = 0.1855\ ra;$$

$$\phi = \frac{pi}{12}\ ra$$

$$\dot{p}1 = \dot{p}2 = \dot{p}3 = 0\ meter/sec$$

$$\dot{\theta} = \dot{\theta}_{12} = \dot{\theta}_{3p} = \dot{\theta}_{06} = \dot{\theta}_{08} = \dot{\phi} = 0\ ra/sec$$

The x and y coordinates of the tip point in trajectory 1 followed by the tip point are shown below in figures 9.1 and 9.2 respectively.

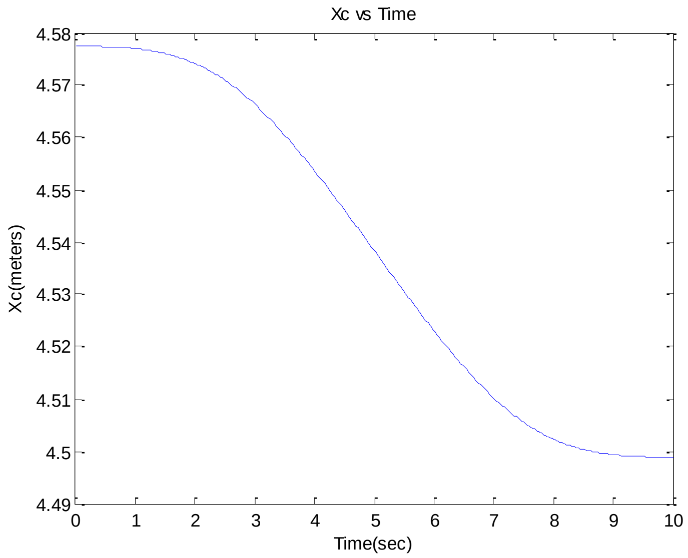


Figure 9.1: The x coordinate of the tip point in trajectory 1

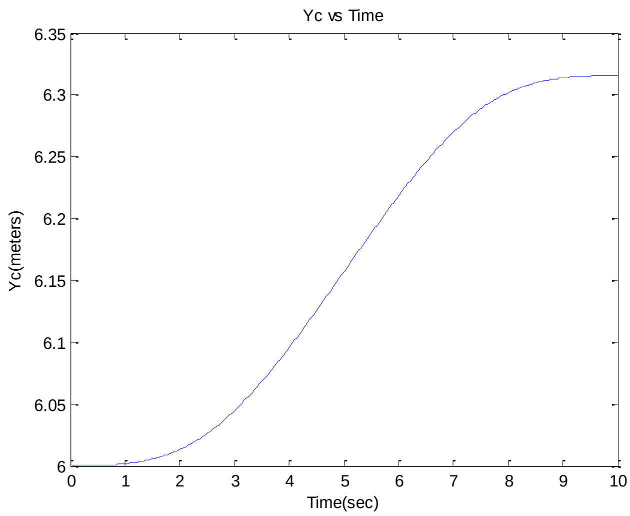


Figure 9.2: The y coordinate of the tip point in trajectory 1

The trajectories of the active joint variables that have been obtained after carrying out the optimal redundancy resolution for trajectory 1 are shown below in figures 9.3,9.4,9.5 and 9.6.

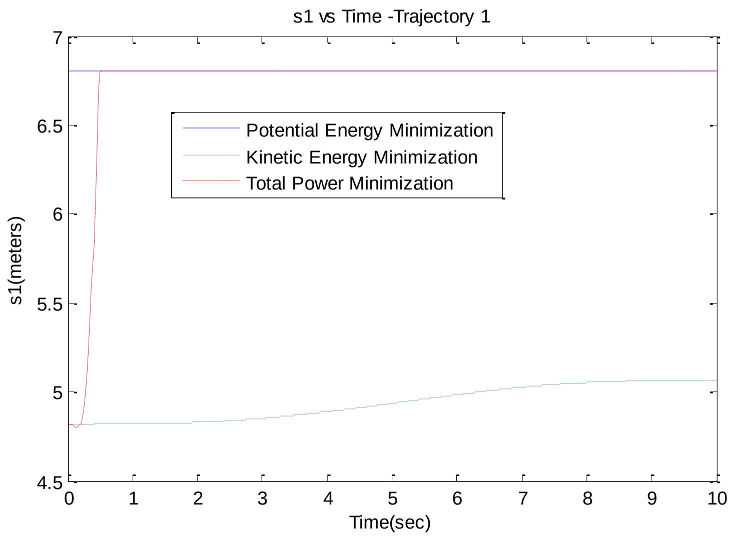


Figure 9.3: The trajectory obtained for s_1

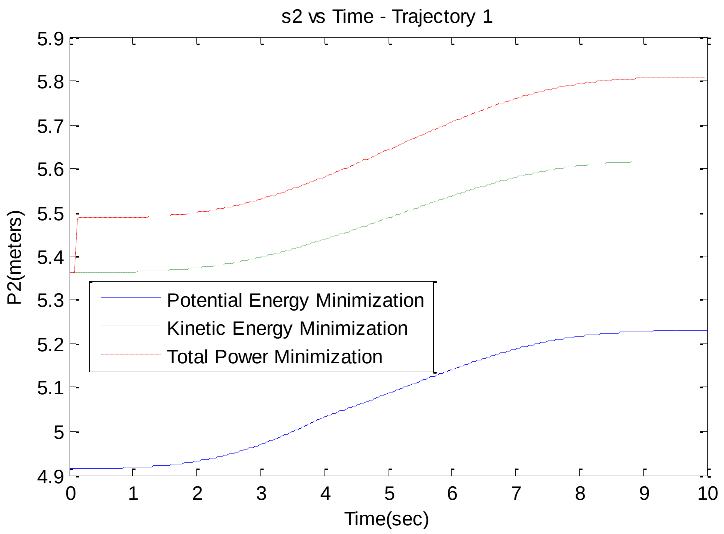


Figure 9.4: The trajectory obtained for s_2

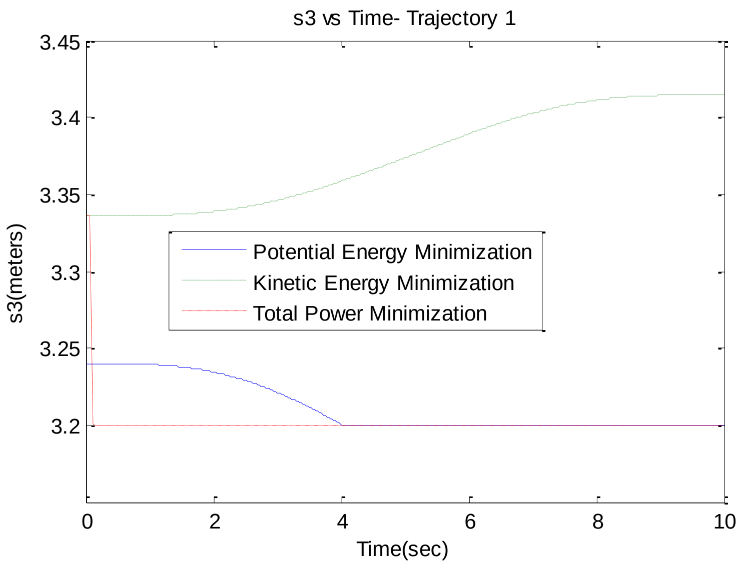


Figure 9.5: The trajectory obtained for s_3

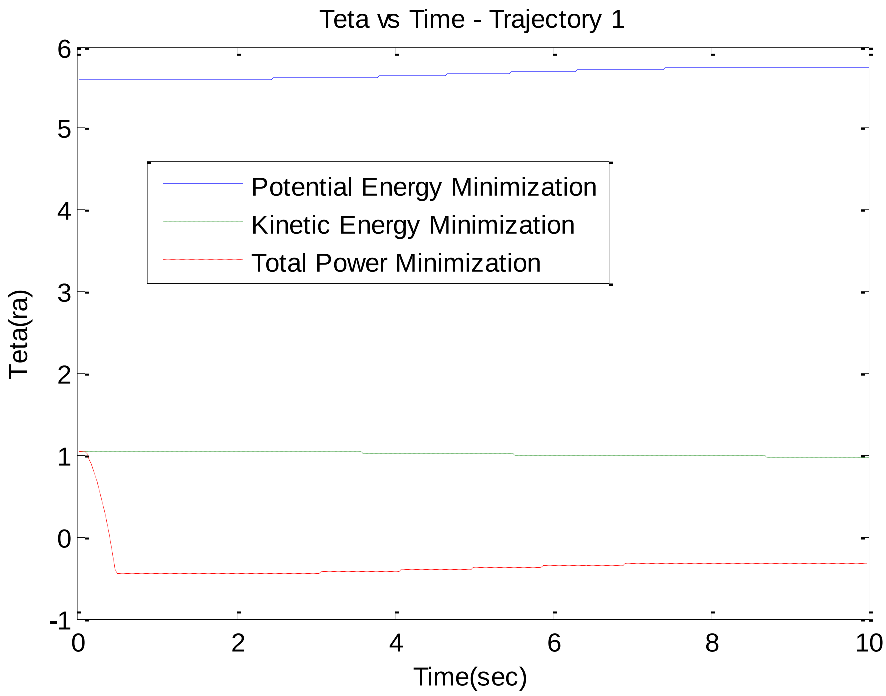


Figure 9.6: The trajectory obtained for θ

The graphs that have been obtained for the variations of potential energy, kinetic energy and total power against time are shown below in figures 9.7,9.8 and 9.9.

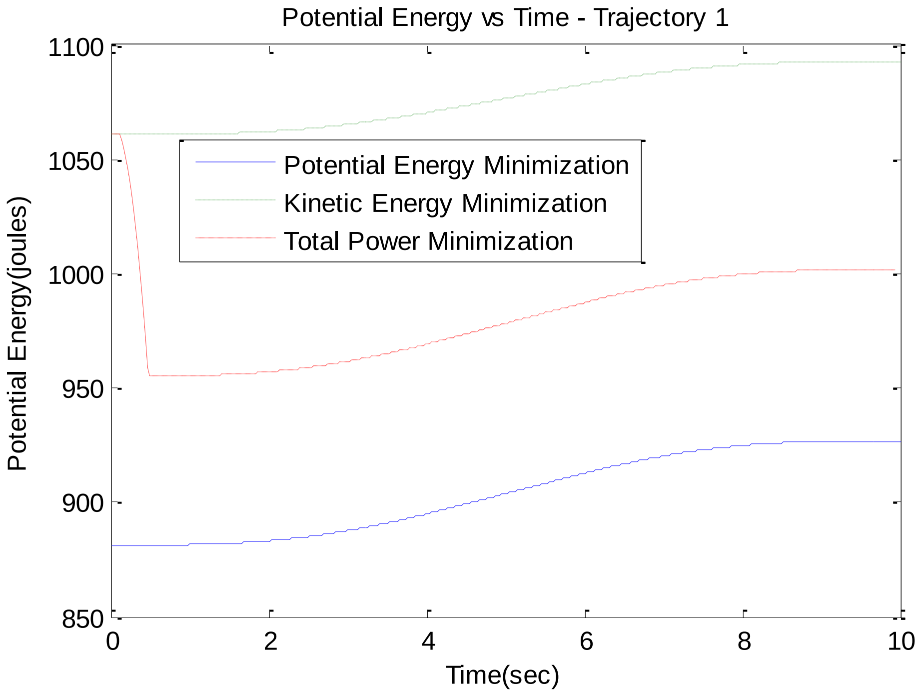


Figure 9.7: Variation of potential energy with time

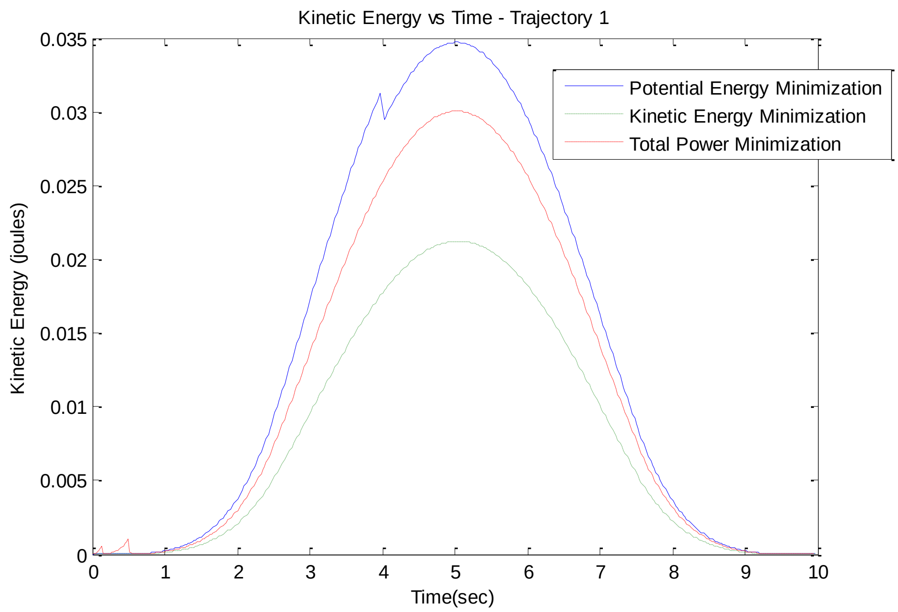


Figure 9.8: Variation of kinetic energy with time

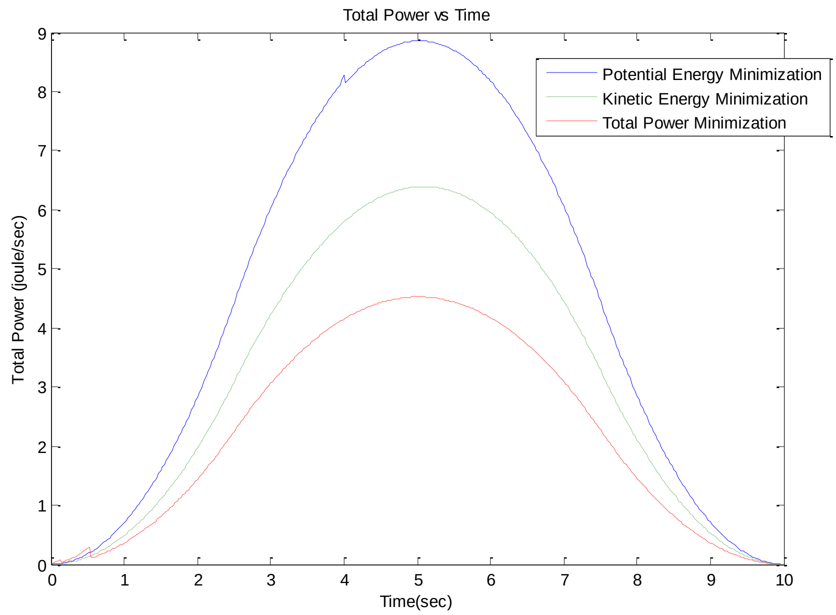


Figure 9.9: Variation of total power with time

CHAPTER 10

DISCUSSION AND CONCLUSIONS

With the pseudo inverse of the jacobian method redundancy resolution can be done only at the velocity level. With the application of this method to parallel manipulators a cost function of only the active joint variables themselves is minimized. Since redundancy resolution is done at the velocity level an error accumulation can occur. With the pseudo inverse method joint limit avoidance can be achieved by assigning a proper cost function. However with the method presented in this thesis both joint limit avoidance and the optimization (minimization)of a cost function can be achieved at the same time. With the augmented Jacobian method as many additional tasks as the number of degree of redundancy can be achieved while the tip point follows the desired trajectory. With the method of this thesis only two additonal tasks can be achieved, one being joint limit avoidance and the other minimization of a cost function, no matter what the degree of redundancy is. The configuration control method is an approximate method so the tip point may not follow the desired trajectory closely. It only approximates the desired trajectory. However with this method there is no restriction on the dimension of the additonal tasks. The redundancy resolution method at the position level with lagrange multipliers can be applied to parallel manipulators. Though with this method inequality constraints are ignored. And also it is not explained how to solve the emerging set of nonlinear equations. The redundancy resolution method applied at the position level for singularity avoidance to a parallel manipulator can be employed for other parallel manipulators. However in this method the details of the optimization procedure is not explained. Compared to the optimal redundancy resolution method presented in this study only singularity avoidance task can be realized. However with the method presented in this study many different optimization tasks can be achieved. Also the method of this study can be applied to serial manipulators, too. The disadvantage of the method of this thesis is that it sometimes yields unsmooth results and it may be necessary to use different genetic algorithm parameters for different types of manipulators. Unsmooth results sometimes appear in the potential energy, kinetic energy and total power graphs. It is always possible to use a suboptimal path for the joint positions at the time steps where unsmooth regions arise. Another disadvantage of the optimal redundancy resolution method of this thesis is that it can hardly be used online while a task is being performed. However, it can be used off-line in order to make an optimal trajectory planning for a desired task.

By inspecting the results of all three optimal redundancy resolutions that have been done along the trajectory which has been shown as trajectory 1, it has been concluded that the employed hybrid genetic algorithm finds the global minimum for all three redundancy resolutions at each step of this trajectory. In redundancy resolution at the position level, global optimization problems that emerge at about ten different steps along the trajectory have been seperately solved one by one with hybrid genetic algorithm. It has been seen that the hybrid genetic algorithm yields global optima at these steps. It has been concluded that global optima have been found at the remaining steps since the potential energy graph obtained for the redundancy resolution at the position level is smooth. In redundancy resolution at the velocity level also I have seperately solved the global optimization problems one by one that emerge at about ten different steps along the trajectory with hybrid genetic algorithm.At each one of these steps all the local optima found by the hybrid genetic algorithm and the kinetic energy values at these local optima have been recorded.It has been observed that there is a single local optimum at each one of these steps. Naturally, the hybrid genetic algorithm has converged to the global optimum at these steps. It has been concluded that global optima have been reached at the time steps corresponding to the smooth regions of the kinetic energy graph obtained in the redundancy resolution at the velocity level. The hybrid genetic algorithm has also been run seperately at the time steps corresponding to the unsmooth regions of the kinetic energy graph, and it has been observed that at these time steps there is also a single local optimum. In redundancy resolution at the acceleration level the global optimization problems have also been seperately solved with hybrid genetic algorithm that emerge at about ten different steps along the trajectory.At each one of these steps all the local optima found by the hybrid genetic algorithm and the total power values at these local optima have been recorded.It has been observed that there is a single local optimum at each one of these steps. Naturally, the hybrid genetic algorithm has converged to the global optimum at these steps. It has been concluded that global optima have been reached at the time steps corresponding to the smooth regions of the total power graph obtained in the redundancy resolution at the acceleration level. The hybrid genetic algorithm has also been run seperately at the time steps corresponding to the unsmooth regions of the total power graph, and it has been observed that at these time steps there is also a single local optimum.

If one or some of the prismatic joints enter or leave the joint limits at some time steps in the prismatic joint graphs obtained as a result of the optimization that have been done along a trajectory, unsmooth regions may arise in one or some of the potential energy, kinetic energy and total power graphs at these time steps. The reason for this is that the first and second time derivatives of the joint positions reach high values at points where prismatic joints enter and leave joint limits. As a second reason for this fact it has been thought that the genetic algorithm may have shifted from global optimum to local optimum at time steps corresponding to the points where prismatic joints enter and leave joint limits. However it has been observed that there is no shift from global minimum to local minimum at time steps corresponding to the unsmooth regions in potential energy, kinetic energy and totalpower graphs. Meaning that the genetic algorithm finds the global minimum also at these regions.

When the potential energy, kinetic energy and total power graphs obtained as a result of the three optimal redundancy resolutions are analyzed it can be seen that potential energy is at its minimum for potential energy optimization, kinetic energy is at its minimum for kinetic energy optimization and total power is at its minimum for total power optimization as expected.

In the future as an extension of the work done in this thesis, the optimal redundancy resolution method of this thesis can also be applied to kinematically redundant spatial parallel manipulators and kinematically redundant serial manipulators.

REFERENCES

1. Merlet, J. P., Paralel robots, Dor-drecht, Springer, 2006

2. Gosselin, C., “Kinematic analysis and Design of Kinematically redundant paralel mechanisms”, Transactions of the ASME, Vol.126, 109-118, 2004

3. Tsai, L. W., The mechanics of serial and paralel manipulators, John Wiley & Sons Inc., New York, 1999.

4. Fahimi, F., Autonomous robots modeling path planning and control, Springer, 978-0-387-09537-0, New York, 2009.

5. Bazaraa, M., Sherali, H. D., Shetty, C. M., Nonlinear Programming Theory and Algorithms, John Wiley & Sons Inc., 978-0-471-48600-8, New Jersey, 2006.

6. Cooper, L., Applied Non-Linear Programming, Aloray Inc., 0-913690-02-3, New Jersey, 1974.

7. Ebrahimi, I., Carretero, J., “A Family of Kinematically Redundant Paralel Manipulators”, Transactions of the ASME, Vol.130, 062306-1 - 062306-8, 2008.

8. Zanganeh, K. E., Angeles, J., “Instantaneous Kinematics and Design of a Novel Redundant Paralel Manipulator”, IEEE, 1050-4729/94, 3043-3048, 1994.

9. Cha, S. H., Lasky, T. A., Velinsky, S. A., “Singularity Avoidance for the 3-RRR Mechanism Using Kinematic Redundancy”, IEEE International Conference on Robotics and Automation, Roma, Italy, 10-14 April, 2007, 1195-1200.

10. Chang, P. H., "A closed-Form Solution for Inverse Kinematics of Robot Manipulators with Redundancy", IEEE Journal of Robotics and Automation, Vol.Ra-3, No.5, 393-403, 1987

11. R.C.HIBBELER,Engineering Mechanics Dynamics

12. Mitchell, M., An Introduction to Genetic Algorithms, The MIT Press, 0-262-13316-4, Cambridge, Massachusetts, 1999.

13. Gen, M., Cheng, R., Genetic Algorithms and Engineering Design, John Wiley & Sons Inc., 1997.

14. Powell, M. J. D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations", Lecture Notes in Mathematics, 144-157, 1978.

15.Genetic Algorithms and Engineering Optimization

16. R.J., Clifton, "Notes on derivation of Lagrange's Equations", http://www.brown.edu/Departments/Engineering/Courses/En137/Lagrange.pdf.11.10.2011

17. Wikimedia Foundation, Inc., "Manipulator", http://en.wikipedia.org/wiki/Manipulator.09.03.2014

18. Lecture notes of ME 502 and ME 522

19. eFunda, Inc., "Triangular Area", http://www.efunda.com/math/areas/triangle.cfm.18.02.2012

APPENDIX A

THE EXPRESSIONS FOR THE ELEMENTS OF THE A AND B MATRICES IN THE SINGULARITY AVOIDANCE REDUNDANCY RESOLUTION METHOD

$$a_{11} = x - l_{11}\cos\theta_{11} - l_{13}\cos(\alpha+\beta) = l_{12}\cos\theta_{12}\ ,$$

$$a_{12} = y - l_{11}\sin\theta_{11} - l_{13}\sin(\alpha+\beta) = l_{12}\sin\theta_{12}\ ,$$

$$a_{13} = l_{13}\left[a_{11}\sin(\alpha+\beta) - a_{12}\cos(\alpha+\beta)\right],$$

$$a_{21} = x - L - l_{21}\cos\theta_{21} + l_{23}\cos(\beta-\alpha) = l_{22}\cos\theta_{22}$$

$$a_{22} = y - l_{21}\sin\theta_{21} - l_{23}\sin(\beta-\alpha) = l_{22}\sin\theta_{22}\ ,$$

$$a_{23} = l_{23}\left[a_{21}\sin(\beta-\alpha) + a_{22}\cos(\beta-\alpha)\right],$$

$$a_{31} = x - L/2 - l_{31}\cos\theta_{31} + l_{33}\cos(\alpha+3\beta) =$$

$$l_{32}\cos\theta_{32}\ ,$$

$$a_{32} = y - \sqrt{3}L/2 - l_{31}\sin\theta_{31} + l_{33}\sin(\alpha+3\beta) =$$

$$l_{32}\sin\theta_{32}\ ,$$

$$a_{33} = -l_{33}\left[a_{31}\sin(\alpha+3\beta) - a_{32}\cos(\alpha+3\beta)\right],$$

and for $i = 1, 2, 3$ and $j = 2(i-1)+1$,

$$b_{ij} = -\left(a_{i1}\cos\theta_{i1} + a_{i2}\sin\theta_{i1}\right),$$

$$b_{i(j+1)} = l_{i1}\left(a_{i1}\sin\theta_{i1} - a_{i2}\cos\theta_{i1}\right).$$

APPENDIX B

THE EQUALITY CONSTRAINT EQUATIONS FOR OPTIMAL REDUNDANCY RESOLUTION AT THE VELOCITY LEVEL

Note that in the below equations the following are true:

$$\theta = teta;\ \theta_{12} = teta12; \theta_{3p} = teta3p;\ \theta_{06} = teta06;\ \theta_{08} = teta08$$

$$s_1 = p1; s_2 = p2; s_3 = p3;$$

$$\dot{\theta} = tetadot;\ \dot{\theta}_{12} = teta12dot; \dot{\theta}_{3p} = teta3pdot; \dot{\theta}_{06} = teta06dot; \dot{\theta}_{08} = teta08dot$$

$$\dot{s}_1 = p1dot; \dot{s}_2 = p2dot; \dot{s}_3 = p3dot$$

$$\dot{x}_c = xcdot; \dot{y}_c = ycdot; \dot{\phi} = phidot; l_p = Lp; l_1 = L_1$$

$\dot{f}_1 = 0 =$ tetadot - phidot + teta12dot + teta3pdot

$\dot{f}_2 = 0 =$ p1dot*cos(teta + teta12) - tetadot*(p1*sin(teta + teta12) + L1*sin(teta) + (3^(1/2)*Lp*sin(pi/6 + teta + teta12 + teta3p))/3) - xcdot - teta12dot*(p1*sin(teta + teta12) + (3^(1/2)*Lp*sin(pi/6 + teta + teta12 + teta3p))/3) - (3^(1/2)*Lp*teta3pdot*sin(pi/6 + teta + teta12 + teta3p))/3

$\dot{f}_3 = 0 =$ tetadot*(p1*cos(teta + teta12) + L1*cos(teta) + (3^(1/2)*Lp*cos(pi/6 + teta + teta12 + teta3p))/3) - ycdot + teta12dot*(p1*cos(teta + teta12) + (3^(1/2)*Lp*cos(pi/6 + teta + teta12 + teta3p))/3) + p1dot*sin(teta + teta12) + (3^(1/2)*Lp*teta3pdot*cos(pi/6 + teta + teta12 + teta3p))/3

$\dot{g}_1 = 0 =$ p1dot*cos(teta + teta12) - tetadot*(p1*sin(teta + teta12) + L1*sin(teta) + Lp*sin(teta + teta12 + teta3p)) - teta12dot*(p1*sin(teta + teta12) + Lp*sin(teta + teta12 + teta3p)) - p2dot*cos(teta06) + p2*teta06dot*sin(teta06)-Lp*teta3pdot*sin(teta + teta12 + teta3p)

$\dot{g}_2 = 0 =$ teta12dot*(p1*cos(teta + teta12) + Lp*cos(teta + teta12 + teta3p)) + p1dot*sin(teta + teta12) - p2dot*sin(teta06) + tetadot*(p1*cos(teta + teta12) + L1*cos(teta) + Lp*cos(teta + teta12 + teta3p)) - p2*teta06dot*cos(teta06) +

Lp*teta3pdot*cos(teta + teta12 + teta3p)

$\dot{g}_3 = 0 =$ p1dot*cos(teta + teta12) - teta12dot*(p1*sin(teta + teta12) + Lp*sin(pi/3 + teta + teta12 + teta3p)) - p3dot*cos(teta08) - tetadot*(p1*sin(teta + teta12) + Lp*sin(pi/3 + teta + teta12 + teta3p) + L1*sin(teta)) - Lp*teta3pdot*sin(pi/3 + teta + teta12 + teta3p) + p3*teta08dot*sin(teta08)

$\dot{g}_4 = 0 =$ teta12dot*(p1*cos(teta + teta12) + Lp*cos(pi/3 + teta + teta12 + teta3p)) + p1dot*sin(teta + teta12) + tetadot*(p1*cos(teta + teta12) + Lp*cos(pi/3 + teta + teta12 + teta3p) + L1*cos(teta)) - p3dot*sin(teta08) - p3*teta08dot*cos(teta08) + Lp*teta3pdot*cos(pi/3 + teta + teta12 + teta3p)

APPENDIX C

TERMS APPEARING IN THE TOTAL POWER EXPRESSION

Note that in the below equations the following are true:

$$\theta = teta;\ \theta_{12} = teta12; \theta_{3p} = teta3p;\ \theta_{06} = teta06;\ \theta_{08} = teta08$$

$$s_1 = p1; s_2 = p2; s_3 = p3;$$

$$\dot{\theta} = tetadot;\ \dot{\theta}_{12} = teta12dot; \dot{\theta}_{3p} = teta3pdot; \dot{\theta}_{06} = teta06dot; \dot{\theta}_{08} = teta08dot$$

$$\dot{s}_1 = p1dot; \dot{s}_2 = p2dot; \dot{s}_3 = p3dot$$

$$\ddot{\theta} = tetaddot;\ \ddot{\theta}_{12} = teta12ddot; \ddot{\theta}_{3p} = teta3pddot; \ddot{\theta}_{06} = teta06ddot; \ddot{\theta}_{08} = teta08ddot$$

$$\ddot{s}_1 = p1ddot;\ \ddot{s}_2 = p2ddot;\ \ddot{s}_3 = p3ddot$$

$$\dot{x}_c = xcdot; \dot{y}_c = ycdot; \dot{\phi} = phidot; l_p = Lp; l_1 = L_1$$

$$\ddot{x}_c = xcddot;\ \ddot{y}_c = ycddot; \ddot{\phi} = phiddot;$$

$T \times \dot{\theta} =$ tetadot*(p1ddot*(2*LP1*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta))*(L1*cos(teta) - cos(teta)*cos(teta12)*(LP1 - p1) + sin(teta)*sin(teta12)*(LP1 - p1)) + 2*LP1*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12))*(cos(teta)*sin(teta12)*(LP1 - p1) - L1*sin(teta) + cos(teta12)*sin(teta)*(LP1 - p1))) + (49*LP1*(2*L1*cos(teta) - 2*cos(teta + teta12)*(LP1 - p1)))/5 + p1dot*(2*LP1*(tetadot*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12)) + teta12dot*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12)))*(L1*cos(teta) - cos(teta)*cos(teta12)*(LP1 - p1) + sin(teta)*sin(teta12)*(LP1 - p1)) - 2*LP1*(tetadot*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta)) + teta12dot*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta)))*(cos(teta)*sin(teta12)*(LP1 - p1) - L1*sin(teta) + cos(teta12)*sin(teta)*(LP1 - p1)) + 2*LP1*(cos(teta)*cos(teta12) –

sin(teta)*sin(teta12))*(p1dot*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta)) - teta12dot*(cos(teta)*cos(teta12)*(LP1 - p1) - sin(teta)*sin(teta12)*(LP1 - p1)) + tetadot*(L1*cos(teta) - cos(teta)*cos(teta12)*(LP1 - p1) + sin(teta)*sin(teta12)*(LP1 - p1))) - 2*LP1*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta))*(teta12dot*(cos(teta)*sin(teta12)*(LP1 - p1) + cos(teta12)*sin(teta)*(LP1 - p1)) + p1dot*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12)) + tetadot*(cos(teta)*sin(teta12)*(LP1 - p1) - L1*sin(teta) + cos(teta12)*sin(teta)*(LP1 - p1)))) + teta12ddot*((2*LC1^3)/3 + (2*LP1^3)/3 - 2*LP1*(cos(teta)*cos(teta12)*(LP1 - p1) - sin(teta)*sin(teta12)*(LP1 - p1))*(L1*cos(teta) - cos(teta)*cos(teta12)*(LP1 - p1) + sin(teta)*sin(teta12)*(LP1 - p1)) + 2*LP1*(cos(teta)*sin(teta12)*(LP1 - p1) + cos(teta12)*sin(teta)*(LP1 - p1))*(cos(teta)*sin(teta12)*(LP1 - p1) - L1*sin(teta) + cos(teta12)*sin(teta)*(LP1 - p1)) + 2*LC1*(LC1*cos(teta)*cos(teta12) - LC1*sin(teta)*sin(teta12))*(L1*cos(teta) + LC1*cos(teta)*cos(teta12) - LC1*sin(teta)*sin(teta12)) + 2*LC1*(LC1*cos(teta)*sin(teta12) + LC1*cos(teta12)*sin(teta))*(L1*sin(teta) + LC1*cos(teta)*sin(teta12) + LC1*cos(teta12)*sin(teta))) + teta12dot*(2*LP1*(L1*cos(teta) - cos(teta)*cos(teta12)*(LP1 - p1) + sin(teta)*sin(teta12)*(LP1 - p1))*(tetadot*(cos(teta)*sin(teta12)*(LP1 - p1) + cos(teta12)*sin(teta)*(LP1 - p1)) + teta12dot*(cos(teta)*sin(teta12)*(LP1 - p1) + cos(teta12)*sin(teta)*(LP1 - p1)) + p1dot*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12))) + 2*LP1*(cos(teta)*sin(teta12)*(LP1 - p1) + cos(teta12)*sin(teta)*(LP1 - p1))*(p1dot*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta)) - teta12dot*(cos(teta)*cos(teta12)*(LP1 - p1) - sin(teta)*sin(teta12)*(LP1 - p1)) + tetadot*(L1*cos(teta) - cos(teta)*cos(teta12)*(LP1 - p1) + sin(teta)*sin(teta12)*(LP1 - p1))) + 2*LP1*(cos(teta)*cos(teta12)*(LP1 - p1) - sin(teta)*sin(teta12)*(LP1 - p1))*(teta12dot*(cos(teta)*sin(teta12)*(LP1 - p1) + cos(teta12)*sin(teta)*(LP1 - p1)) + p1dot*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12)) + tetadot*(cos(teta)*sin(teta12)*(LP1 - p1) - L1*sin(teta) + cos(teta12)*sin(teta)*(LP1 - p1))) + 2*LP1*(cos(teta)*sin(teta12)*(LP1 - p1) - L1*sin(teta) + cos(teta12)*sin(teta)*(LP1 - p1))*(tetadot*(cos(teta)*cos(teta12)*(LP1 - p1) –

sin(teta)*sin(teta12)*(LP1 - p1)) + teta12dot*(cos(teta)*cos(teta12)*(LP1 - p1) - sin(teta)*sin(teta12)*(LP1 - p1)) - p1dot*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta))) - 2*LC1*(tetadot*(LC1*cos(teta)*sin(teta12) + LC1*cos(teta12)*sin(teta)) + teta12dot*(LC1*cos(teta)*sin(teta12) + LC1*cos(teta12)*sin(teta)))*(L1*cos(teta) + LC1*cos(teta)*cos(teta12) - LC1*sin(teta)*sin(teta12)) - 2*LC1*(LC1*cos(teta)*sin(teta12) + LC1*cos(teta12)*sin(teta))*(teta12dot*(LC1*cos(teta)*cos(teta12) - LC1*sin(teta)*sin(teta12)) + tetadot*(L1*cos(teta) + LC1*cos(teta)*cos(teta12) - LC1*sin(teta)*sin(teta12))) + 2*LC1*(teta12dot*(LC1*cos(teta)*sin(teta12) + LC1*cos(teta12)*sin(teta)) + tetadot*(L1*sin(teta) + LC1*cos(teta)*sin(teta12) + LC1*cos(teta12)*sin(teta)))*(LC1*cos(teta)*cos(teta12) - LC1*sin(teta)*sin(teta12)) + 2*LC1*(tetadot*(LC1*cos(teta)*cos(teta12) - LC1*sin(teta)*sin(teta12)) + teta12dot*(LC1*cos(teta)*cos(teta12) - LC1*sin(teta)*sin(teta12)))*(L1*sin(teta) + LC1*cos(teta)*sin(teta12) + LC1*cos(teta12)*sin(teta))) + tetaddot*(2*LP1*(L1*cos(teta) - cos(teta)*cos(teta12)*(LP1 - p1) + sin(teta)*sin(teta12)*(LP1 - p1))^2 + 2*LP1*(cos(teta)*sin(teta12)*(LP1 - p1) - L1*sin(teta) + cos(teta12)*sin(teta)*(LP1 - p1))^2 + (L1^3*cos(teta)^2)/4 + (L1^3*sin(teta)^2)/4 + L1^3/12 + (2*LC1^3)/3 + (2*LP1^3)/3 + 2*LC1*(L1*cos(teta) + LC1*cos(teta)*cos(teta12) - LC1*sin(teta)*sin(teta12))^2 + 2*LC1*(L1*sin(teta) + LC1*cos(teta)*sin(teta12) + LC1*cos(teta12)*sin(teta))^2) + (49*L1^2*cos(teta))/10 + (49*LC1*(2*LC1*cos(teta + teta12) + 2*L1*cos(teta)))/5)

$F_1 \times \dot{s}_1 =$ p1dot*(tetaddot*(2*LP1*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta))*(L1*cos(teta) - cos(teta)*cos(teta12)*(LP1 - p1) + sin(teta)*sin(teta12)*(LP1 - p1)) + 2*LP1*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12))*(cos(teta)*sin(teta12)*(LP1 - p1) - L1*sin(teta) + cos(teta12)*sin(teta)*(LP1 - p1))) - p1dot*(2*LP1*(tetadot*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta)) + teta12dot*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta)))*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12)) - 2*LP1*(tetadot*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12)) + teta12dot*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12)))*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta))) + (98*LP1*sin(teta + teta12))/5 +

p1ddot*(2*LP1*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta))^2 + 2*LP1*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12))^2) + teta12dot*(2*LP1*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12))*(p1dot*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta)) - teta12dot*(cos(teta)*cos(teta12)*(LP1 - p1) - sin(teta)*sin(teta12)*(LP1 - p1)) + tetadot*(L1*cos(teta) - cos(teta)*cos(teta12)*(LP1 - p1) + sin(teta)*sin(teta12)*(LP1 - p1))) - 2*LP1*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta))*(teta12dot*(cos(teta)*sin(teta12)*(LP1 - p1) + cos(teta12)*sin(teta)*(LP1 - p1)) + p1dot*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12)) + tetadot*(cos(teta)*sin(teta12)*(LP1 - p1) - L1*sin(teta) + cos(teta12)*sin(teta)*(LP1 - p1))) + 2*LP1*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta))*(tetadot*(cos(teta)*sin(teta12)*(LP1 - p1) + cos(teta12)*sin(teta)*(LP1 - p1)) + teta12dot*(cos(teta)*sin(teta12)*(LP1 - p1) + cos(teta12)*sin(teta)*(LP1 - p1)) + p1dot*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12))) + 2*LP1*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12))*(tetadot*(cos(teta)*cos(teta12)*(LP1 - p1) - sin(teta)*sin(teta12)*(LP1 - p1)) + teta12dot*(cos(teta)*cos(teta12)*(LP1 - p1) - sin(teta)*sin(teta12)*(LP1 - p1)) - p1dot*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta)))) + teta12ddot*(2*LP1*(cos(teta)*sin(teta12)*(LP1 - p1) + cos(teta12)*sin(teta)*(LP1 - p1))*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12)) - 2*LP1*(cos(teta)*cos(teta12)*(LP1 - p1) - sin(teta)*sin(teta12)*(LP1 - p1))*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta))) - 2*LP1*(tetadot*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12)) + teta12dot*(cos(teta)*cos(teta12) - sin(teta)*sin(teta12)))*(p1dot*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta)) - teta12dot*(cos(teta)*cos(teta12)*(LP1 - p1) - sin(teta)*sin(teta12)*(LP1 - p1)) + tetadot*(L1*cos(teta) - cos(teta)*cos(teta12)*(LP1 - p1) + sin(teta)*sin(teta12)*(LP1 - p1))) + 2*LP1*(tetadot*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta)) + teta12dot*(cos(teta)*sin(teta12) + cos(teta12)*sin(teta)))*(teta12dot*(cos(teta)*sin(teta12)*(LP1 - p1) + cos(teta12)*sin(teta)*(LP1 - p1)) + p1dot*(cos(teta)*cos(teta12) -

sin(teta)*sin(teta12)) + tetadot*(cos(teta)*sin(teta12)*(LP1 - p1) - L1*sin(teta) + cos(teta12)*sin(teta)*(LP1 - p1))))

$F_2 \times \dot{s}_2 =$ p2dot*(p2ddot*(2*LP2*cos(teta06)^2 + 2*LP2*sin(teta06)^2) + (98*LP2*sin(teta06))/5 - 2*LP2*teta06dot*cos(teta06)*(p2dot*sin(teta06) - teta06dot*cos(teta06)*(LP2 - p2)) + 2*LP2*teta06dot*sin(teta06)*(p2dot*cos(teta06) + teta06dot*sin(teta06)*(LP2 - p2)))

$F_3 \times \dot{s}_3 =$ p3dot*(p3ddot*(2*LP3*cos(teta06)^2 + 2*LP3*sin(teta06)^2) + (98*LP3*sin(teta08))/5)

APPENDIX D

THE EQUALITY CONSTRAINT EQUATIONS FOR OPTIMAL REDUNDANCY RESOLUTION AT THE ACCELERATION LEVEL

Note that in the below equations the following are true:

$$\theta = teta;\ \theta_{12} = teta12; \theta_{3p} = teta3p;\ \theta_{06} = teta06;\ \theta_{08} = teta08$$

$$s_1 = p1; s_2 = p2; s_3 = p3;$$

$$\dot{\theta} = tetadot;\ \dot{\theta}_{12} = teta12dot; \dot{\theta}_{3p} = teta3pdot; \dot{\theta}_{06} = teta06dot; \dot{\theta}_{08} = teta08dot$$

$$\dot{s}_1 = p1dot; \dot{s}_2 = p2dot; \dot{s}_3 = p3dot$$

$$\ddot{\theta} = tetaddot;\ \ddot{\theta}_{12} = teta12ddot; \ddot{\theta}_{3p} = teta3pddot; \ddot{\theta}_{06} = teta06ddot; \ddot{\theta}_{08} = teta08ddot$$

$$\ddot{s}_1 = p1ddot;\ \ddot{s}_2 = p2ddot;\ \ddot{s}_3 = p3ddot$$

$$\dot{x}_c = xcdot; \dot{y}_c = ycdot; \dot{\phi} = phidot; l_p = Lp; l_1 = L_1$$

$$\ddot{x}_c = xcddot;\ \ddot{y}_c = ycddot; \ddot{\phi} = phiddot;$$

$\ddot{f}_1 = 0 =$ tetaddot - phiddot + teta12ddot + teta3pddot

$\ddot{\boldsymbol{f}}_{\boldsymbol{2}} = 0 =$ p1ddot*cos(teta + teta12) - teta12dot*(tetadot*(p1*cos(teta + teta12) + (3^(1/2)*Lp*cos(pi/6 + teta + teta12 + teta3p))/3) + teta12dot*(p1*cos(teta + teta12) + (3^(1/2)*Lp*cos(pi/6 + teta + teta12 + teta3p))/3) + p1dot*sin(teta + teta12) + (3^(1/2)*Lp*teta3pdot*cos(pi/6 + teta + teta12 + teta3p))/3) - teta3pdot*((3^(1/2)*Lp*tetadot*cos(pi/6 + teta + teta12 + teta3p))/3 + (3^(1/2)*Lp*teta12dot*cos(pi/6 + teta + teta12 + teta3p))/3 + (3^(1/2)*Lp*teta3pdot*cos(pi/6 + teta + teta12 + teta3p))/3) - tetaddot*(p1*sin(teta + teta12) + L1*sin(teta) + (3^(1/2)*Lp*sin(pi/6 + teta + teta12 + teta3p))/3) - p1dot*(tetadot*sin(teta + teta12) + teta12dot*sin(teta + teta12)) - xcddot - tetadot*(tetadot*(p1*cos(teta + teta12) + L1*cos(teta) + (3^(1/2)*Lp*cos(pi/6 + teta + teta12 + teta3p))/3) + teta12dot*(p1*cos(teta + teta12) + (3^(1/2)*Lp*cos(pi/6 +

teta + teta12 + teta3p))/3) + p1dot*sin(teta + teta12) + (3^(1/2)*Lp*teta3pdot*cos(pi/6 + teta + teta12 + teta3p))/3) - teta12ddot*(p1*sin(teta + teta12) + (3^(1/2)*Lp*sin(pi/6 + teta + teta12 + teta3p))/3) - (3^(1/2)*Lp*teta3pddot*sin(pi/6 + teta + teta12 + teta3p))/3

$\ddot{\boldsymbol{f}}_{\boldsymbol{3}} = 0 =$ tetaddot*(p1*cos(teta + teta12) + L1*cos(teta) + (3^(1/2)*Lp*cos(pi/6 + teta + teta12 + teta3p))/3) - ycddot + p1dot*(tetadot*cos(teta + teta12) + teta12dot*cos(teta + teta12)) - teta3pdot*((3^(1/2)*Lp*tetadot*sin(pi/6 + teta + teta12 + teta3p))/3 + (3^(1/2)*Lp*teta12dot*sin(pi/6 + teta + teta12 + teta3p))/3 + (3^(1/2)*Lp*teta3pdot*sin(pi/6 + teta + teta12 + teta3p))/3) - teta12dot*(tetadot*(p1*sin(teta + teta12) + (3^(1/2)*Lp*sin(pi/6 + teta + teta12 + teta3p))/3) - p1dot*cos(teta + teta12) + teta12dot*(p1*sin(teta + teta12) + (3^(1/2)*Lp*sin(pi/6 + teta + teta12 + teta3p))/3) + (3^(1/2)*Lp*teta3pdot*sin(pi/6 + teta + teta12 + teta3p))/3) + teta12ddot*(p1*cos(teta + teta12) + (3^(1/2)*Lp*cos(pi/6 + teta + teta12 + teta3p))/3) + p1ddot*sin(teta + teta12) - tetadot*(tetadot*(p1*sin(teta + teta12) + L1*sin(teta) + (3^(1/2)*Lp*sin(pi/6 + teta + teta12 + teta3p))/3) - p1dot*cos(teta + teta12) + teta12dot*(p1*sin(teta + teta12) + (3^(1/2)*Lp*sin(pi/6 + teta + teta12 + teta3p))/3) + (3^(1/2)*Lp*teta3pdot*sin(pi/6 + teta + teta12 + teta3p))/3) + (3^(1/2)*Lp*teta3pddot*cos(pi/6 + teta + teta12 + teta3p))/3

$\ddot{\boldsymbol{g}}_{\boldsymbol{1}} = 0 =$ teta06dot*(p2dot*sin(teta06) + p2*teta06dot*cos(teta06)) - tetaddot*(p1*sin(teta + teta12) + L1*sin(teta) + Lp*sin(teta + teta12 + teta3p)) - p1dot*(tetadot*sin(teta + teta12) + teta12dot*sin(teta + teta12)) - tetadot*(teta12dot*(p1*cos(teta + teta12) + Lp*cos(teta + teta12 + teta3p)) + p1dot*sin(teta + teta12) + tetadot*(p1*cos(teta + teta12) + L1*cos(teta) + Lp*cos(teta + teta12 + teta3p)) + Lp*teta3pdot*cos(teta + teta12 + teta3p)) + p1ddot*cos(teta + teta12) - teta12ddot*(p1*sin(teta + teta12) + Lp*sin(teta + teta12 + teta3p)) - p2ddot*cos(teta06) - teta12dot*(tetadot*(p1*cos(teta + teta12) + Lp*cos(teta + teta12 + teta3p)) + teta12dot*(p1*cos(teta + teta12) + Lp*cos(teta + teta12 + teta3p)) + p1dot*sin(teta + teta12) + Lp*teta3pdot*cos(teta + teta12 + teta3p)) - teta3pdot*(Lp*tetadot*cos(teta + teta12 + teta3p) + Lp*teta12dot*cos(teta + teta12 + teta3p) + Lp*teta3pdot*cos(teta + teta12 + teta3p)) +

p2*teta06ddot*sin(teta06) + p2dot*teta06dot*sin(teta06) - Lp*teta3pddot*sin(teta + teta12 + teta3p)

$\ddot{g}_2 = 0 =$ p1dot*(tetadot*cos(teta + teta12) + teta12dot*cos(teta + teta12)) - teta3pdot*(Lp*tetadot*sin(teta + teta12 + teta3p) + Lp*teta12dot*sin(teta + teta12 + teta3p) + Lp*teta3pdot*sin(teta + teta12 + teta3p)) - teta06dot*(p2dot*cos(teta06) - p2*teta06dot*sin(teta06)) - teta12dot*(tetadot*(p1*sin(teta + teta12) + Lp*sin(teta + teta12 + teta3p)) - p1dot*cos(teta + teta12) + teta12dot*(p1*sin(teta + teta12) + Lp*sin(teta + teta12 + teta3p)) + Lp*teta3pdot*sin(teta + teta12 + teta3p)) + teta12ddot*(p1*cos(teta + teta12) + Lp*cos(teta + teta12 + teta3p)) + p1ddot*sin(teta + teta12) - p2ddot*sin(teta06) - tetadot*(tetadot*(p1*sin(teta + teta12) + L1*sin(teta) + Lp*sin(teta + teta12 + teta3p)) - p1dot*cos(teta + teta12) + teta12dot*(p1*sin(teta + teta12) + Lp*sin(teta + teta12 + teta3p)) + Lp*teta3pdot*sin(teta + teta12 + teta3p)) + tetaddot*(p1*cos(teta + teta12) + L1*cos(teta) + Lp*cos(teta + teta12 + teta3p)) - p2*teta06ddot*cos(teta06) - p2dot*teta06dot*cos(teta06) + Lp*teta3pddot*cos(teta + teta12 + teta3p)

$\ddot{g}_3 = 0 =$ teta08dot*(p3dot*sin(teta08) + p3*teta08dot*cos(teta08)) - teta3pdot*(Lp*tetadot*cos(pi/3 + teta + teta12 + teta3p) + Lp*teta12dot*cos(pi/3 + teta + teta12 + teta3p) + Lp*teta3pdot*cos(pi/3 + teta + teta12 + teta3p)) - teta12dot*(tetadot*(p1*cos(teta + teta12) + Lp*cos(pi/3 + teta + teta12 + teta3p)) + teta12dot*(p1*cos(teta + teta12) + Lp*cos(pi/3 + teta + teta12 + teta3p)) + p1dot*sin(teta + teta12) + Lp*teta3pdot*cos(pi/3 + teta + teta12 + teta3p)) - teta12ddot*(p1*sin(teta + teta12) + Lp*sin(pi/3 + teta + teta12 + teta3p)) - p1dot*(tetadot*sin(teta + teta12) + teta12dot*sin(teta + teta12)) + p1ddot*cos(teta + teta12) - tetadot*(teta12dot*(p1*cos(teta + teta12) + Lp*cos(pi/3 + teta + teta12 + teta3p)) + p1dot*sin(teta + teta12) + tetadot*(p1*cos(teta + teta12) + Lp*cos(pi/3 + teta + teta12 + teta3p) + L1*cos(teta)) + Lp*teta3pdot*cos(pi/3 + teta + teta12 + teta3p)) - p3ddot*cos(teta08) - tetaddot*(p1*sin(teta + teta12) + Lp*sin(pi/3 + teta + teta12 + teta3p) + L1*sin(teta)) - Lp*teta3pddot*sin(pi/3 + teta + teta12 + teta3p) + p3*teta08ddot*sin(teta08) + p3dot*teta08dot*sin(teta08)

$\ddot{g}_4 = 0 =$teta12ddot*(p1*cos(teta + teta12) + Lp*cos(pi/3 + teta + teta12 + teta3p)) - teta08dot*(p3dot*cos(teta08) - p3*teta08dot*sin(teta08)) + p1dot*(tetadot*cos(teta

+ teta12) + teta12dot*cos(teta + teta12)) - teta3pdot*(Lp*tetadot*sin(pi/3 + teta + teta12 + teta3p) + Lp*teta12dot*sin(pi/3 + teta + teta12 + teta3p) + Lp*teta3pdot*sin(pi/3 + teta + teta12 + teta3p)) - teta12dot*(tetadot*(p1*sin(teta + teta12) + Lp*sin(pi/3 + teta + teta12 + teta3p)) + teta12dot*(p1*sin(teta + teta12) + Lp*sin(pi/3 + teta + teta12 + teta3p)) - p1dot*cos(teta + teta12) + Lp*teta3pdot*sin(pi/3 + teta + teta12 + teta3p)) + p1ddot*sin(teta + teta12) + tetaddot*(p1*cos(teta + teta12) + Lp*cos(pi/3 + teta + teta12 + teta3p) + L1*cos(teta)) - p3ddot*sin(teta08) - tetadot*(teta12dot*(p1*sin(teta + teta12) + Lp*sin(pi/3 + teta + teta12 + teta3p)) - p1dot*cos(teta + teta12) + tetadot*(p1*sin(teta + teta12) + Lp*sin(pi/3 + teta + teta12 + teta3p) + L1*sin(teta)) + Lp*teta3pdot*sin(pi/3 + teta + teta12 + teta3p)) - p3*teta08ddot*cos(teta08) - p3dot*teta08dot*cos(teta08) + Lp*teta3pddot*cos(pi/3 + teta + teta12 + teta3p)