SINGULARITY ANALYSIS AND WORKSPACE DETERMINATION OF PARALLEL MANIPULATORS: APPLICATIONS TO RECONFIGURABLE MANIPULATORS

POOYA MERAT

A THESIS

IN THE DEPARTMENT

OF

MECHANICAL AND INDUSTRIAL ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE

(MECHANICAL AND INDUSTRIAL ENGINEERING) AT

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

JUNE, 2014

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Pooya Merat

Entitled: **Singularity Analysis and Workspace Determination of Parallel Manipulators: Applications to Reconfigurable Manipulators**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Mechanical and Industrial Engineering)

complies with the regulations of the university and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

Chair	*Dr. Martin D. Pugh*
Examiner	*Dr. Chunyan Wang*
Examiner	*Dr. Wenfang Xie*
Supervisor	*Dr. Chun-Yi Su*
Co-supervisor	*Dr. Farhad Aghili*

Approved by:

Chair of Department or Graduate Program Director

Dean of Faculty

Date: June 9, 2014

ABSTRACT

Singularity Analysis and Workspace Determination

of Parallel Manipulators: Applications to Reconfigurable Manipulators

The reconfigurable manipulators with lockable cylindrical joints, proposed by Aghili [1], is a kind of reconfigurable mechanism that offers robust reconfigurability for serial manipulators while adding minimal weight to the system. Such design can be very advantageous in space applications where both dexterity and minimum weight are necessary. This design benefits from a specific design of cylindrical links which enables reconfiguration while the manipulator forms a closed chain. During reconfiguration; however, the system is a parallel mechanism and the singularities may limit its performance. For this reason, analysis of singularities and workspace for control purposes is necessary. In this thesis, first, we discuss the singularity analysis of parallel manipulators and a numerical method is proposed for singularity avoidance. Then, workspace analysis is discussed and another numerical method for workspace analysis is proposed. Both of these methods are applicable to a specific class of parallel manipulators including the reconfigurable manipulator with lockable cylindrical joints. The singularity avoidance technique is based on estimating closest pose of singular configuration in workspace. The results of this method are compared to two other related methods and the advantages and efficiency of the method are discussed. The second proposed method is towards the workspace determination of manipulators. Workspace determination can be used in the design stage or for the control of manipulators, such as high-level trajectory planning in a complex workspace. Results for several case studies are illustrated and the advantages and limitations of the method are discussed. As a complementary contribution to this thesis, a Matlab package is developed as a modeling and simulation test-bed for this type of reconfigurable manipulators.

ACKNOWLEDGEMENT

I would like to thank my supervisors Dr. C. Y. Su and Dr. F. Aghili for their support, patience, guidance and trust throughout this work. They have provided me with the opportunity to develop my raw ideas and have helped me to familiarize myself with scientific and research work.

Finally I would also like to express my gratitude to my beloved Rosa, and my family for their support during this research.

Table of Contents

List of Figures

List of Tables

1. Introduction

In this chapter, the current design of reconfigurable manipulators is reviewed along with the motivations for the presented work and the thesis outline.

1.1 Conceptual Design

Aghili [1] has first proposed a new reconfigurable manipulator design which is very similar to serial manipulators but is also able to reconfigure itself when it is not performing any other task. The robot has a reconfiguration phase that enables the change of length and twist angle of its arms without using extra joint actuators. To do so, first, the robot forms a closed kinematic chain by connecting its end-effector to a docking position. After the connection of the end-effector, the normally locked breaks implemented on its arms are released, and additional degrees of freedom will be added to the manipulator arms. If certain criteria are met, the active joints forces will be projected from the constraints exerted on end-effector onto the released degrees of freedom of arms, and thus reconfigurability of arms will be feasible. To exit reconfiguration phase the arms are locked and the end-effector is released from the docking position.

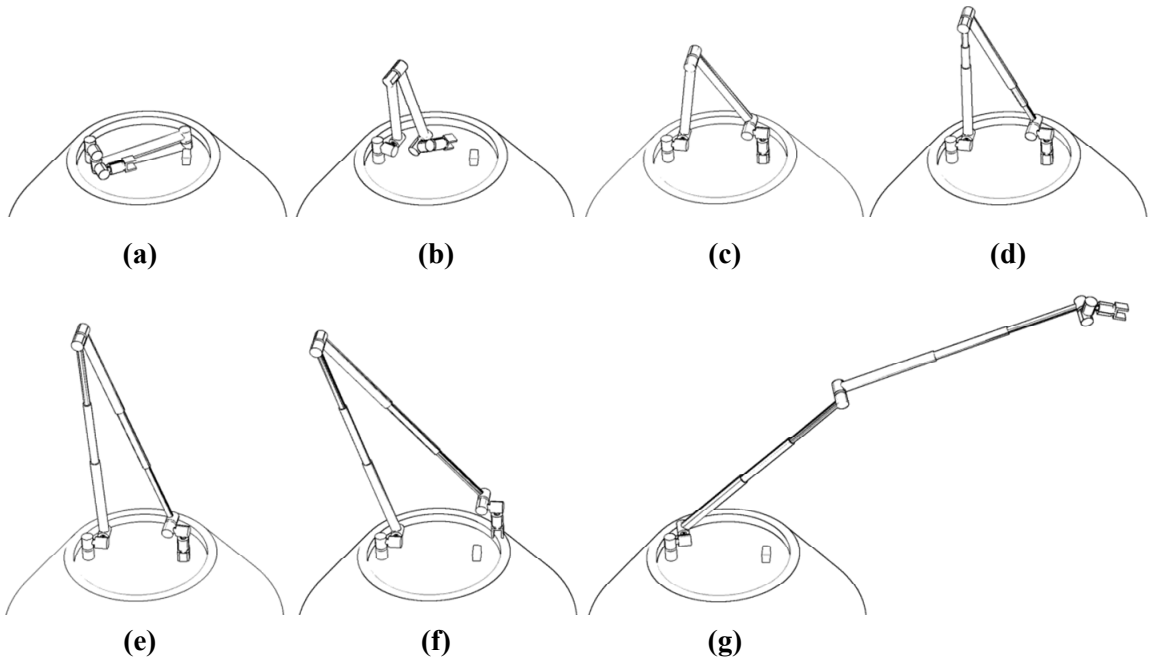
Such design is very useful for applications that require dexterity and minimum actuators at the same time such as space applications. Many space agencies agree that the reconfigurable space robots will be used in future for space exploration and in orbit servicing [2]. Reconfigurable manipulators with cylindrical links maintain similar structure and design with respect to conventional serial manipulators. This will facilitate replacement of a reconfigurable manipulator with a serial manipulator.

The space manipulators should be size efficient in order to be placed on lunch vehicle. For example Canadarm II, because of its large size needs to be folded from the hinges in the middle of each arm, to fit on the lunch vehicle. Then, it is manually unfolded by astronauts in space. One possible approach towards the automation of unfolding process is to use an extra actuator to unfold the links, however this would considerably add up to the weight of the system which is not desirable in space applications due to high lunch cost. The reconfigurable manipulator with cylindrical joints can automate this unfolding process without using any extra joint actuator. Instead it uses low-weight inactive locks. Moreover, the space manipulators are being used for several applications such as inspection or grasping free-floating satellites [3] and it is desirable to have a manipulator which can adapt itself for each particular task.

When released, each reconfigurable arm or cylindrical joint provides two degrees of freedom; one prismatic and one revolute. The cylindrical joint design is explained in more details in section **2.5**. A possible reconfiguration process is demonstrated in **Figure 1**.

1.2 Motivation

The control process of the mentioned reconfigurable manipulator, however, is not a trivial problem. During the reconfiguration, the manipulator resembles parallel manipulators which have a more complex workspace and control process. On the other hand, unlike most parallel mechanisms it has only two connection points to the ground which decreases the manipulability of the system. The main goal of this research is to address some of the complexities accompanied with the control process. A test-bed program in Matlab is also developed for further study and investigation of these reconfigurable manipulators.



(a) The initial folded state; (b) the end-effector moving towards its docking position; (c) constraining the end-effector motion at the docking position and releasing both link breaks; (d) reconfiguration process; (d) end of reconfiguration process, locking cylindrical joints; (f) releasing the docking position; (g) the reconfigured robot ready for next operation

Figure 1: Reconfiguration process of a 7-DoF reconfigurable manipulator

1.3 Thesis Outline

In Chapter **2**, the modeling and in In Chapter **3**, the control and simulation of reconfigurable manipulators are reviewed. Chapter **4** introduces a Matlab package developed as a test-bed for reconfigurable manipulators. In Chapter **5** the singularities of parallel manipulators are discussed and explained with a five-bar linkage case study in details. In Chapter **6**, a literature review of singularity handling techniques is presented along with a new proposed method. In Chapter **7**, a literature review on workspace determination as well as a new proposed method is presented and the results are discussed. In Chapter **8**, conclusions are derived and some future works are suggested.

2. Modeling

There are several methods to model a constrained multi-body system such as augmented Lagrangian formulation [4] and projection-based method [5]. In this context we review the latter which has been previously suggested by Aghili [6] for the reconfigurable manipulators. A thorough literature review on existing methods for constraint system modeling is presented by Merlet [7]. In projection-based modeling it is first required to model the open chain mechanism will be presented in the following.

2.1 Kinematics

Jacques Denavit and Richard Hartenberg in 1955 introduced a standard method for assigning coordinate systems (or frames) to serial chain multi-body mechanisms. Their method is being

Table 1: Frame assignment convention according to Denavit-Hartenberg

Step	Description
1	Select each two consecutive joint axes (*i* and *i+1*) and repeat step 2 to 4 for all of them.
2	Find common perpendicular or intersection point between the two axes. Assign link *i* frame origin at the point that intersection meets axis *i*.
3	Assign z_i axis pointing along *i*-th joint axis. (the direction is arbitrary and will determine the sign of joint variable).
4	Assign x_i axis pointing along the common perpendicular. (in case of intersection of axis z_i and z_{i+1}, x_i should point along the normal to the containing plane of both axes.)
5	Assign frame {0} to match {1} when the first joint variable is 0.

widely used for kinematic modeling of serial robotic manipulators. A complete description of the method could be found in [8], which is summarized here. The convention of assigning the frames is briefly explained in **Table 1**. Due to existence of some freedom in this convention, frame assignments may not be unique for a manipulator. But despite the different eventuality of frame assignment, kinematic model is always unique.

The Denavit-Hartenberg (DH) parameters are four parameters that represent the transformation from one frame to the next and are explained in Table 2.

Table 2: DH table parameters

Parameter	Name	Description
α_{i-1}	Link twist	Rotation from z_{i-1} to z_i along x_{i-1}
a_{i-1}	Link length	Translation from z_{i-1} to z_i along x_{i-1}
d_i	Joint length	Translation from x_{i-1} to x_i along z_i
θ_i	Joint angle	Rotation from x_{i-1} to x_i along z_i

The Denavit-Hartenberg table known also as the DH table is consisted of the mentioned four parameters of all links of a multi-body system ascending from the first link to the last link. It would make further calculations easier, if the frame assignments are chosen in a way that most possible parameters become zero (0). In the following, we present our approach towards the extraction of DH table.

The frame and dynamic properties assignment for a locked manipulator is illustrated in **Figure 2**. Each link is accompanied with a frame, named as the index of the link, for example frame $\{i\}$ is placed on link i. Frame $\{0\}$ however, is a fixed frame with a constant transformation to the base frame or $\{B\}$. For an n-DOF manipulator, we declare frames $\{n\}$ and $\{E\}$ to be attached to the last link and end-effector, respectively. Both moment of inertia of the link I or I_i and its center of mass or P_i are expressed in frame $\{i\}$.

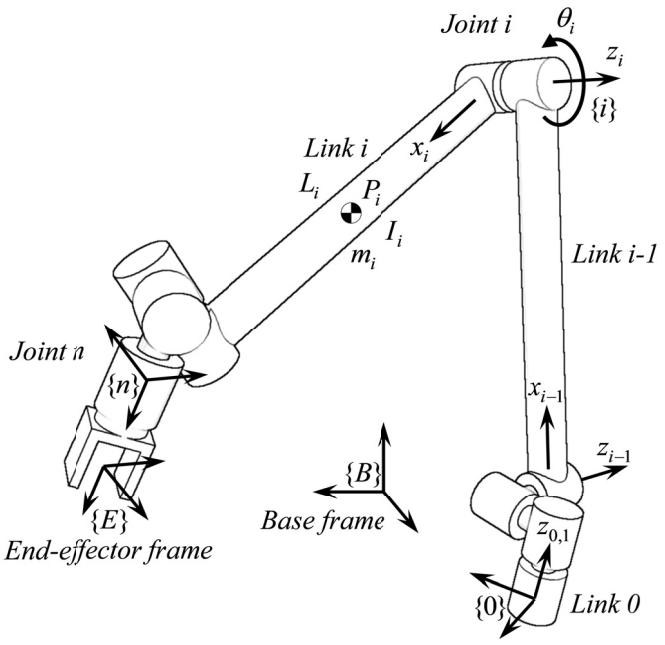


Figure 2: Frame assignment and dynamic properties of the locked manipulator

2.2 Dynamics

Among the two commonly used dynamic formulations; Newton-Euler and Lagrangian, the former was known to be more efficient. A comparison of the recursive Lagrangian formulation and recursive Newton-Euler formulation has been studied by Hollerbach [9]. Both Hollerbach [9] and Silver [10] suggested that with a proper Lagrangian formulation, the two methods are equivalent. A Matlab toolbox has been created by Corke [11] for modeling serial-link manipulators. In his implementation in order to reduce simulation calculations, the least significant terms will not be calculated. Fisette et al. [12, 13] has developed a powerful software named ROBOTRAN for formulation of a general multi-body system, that has a fast and optimized calculation of dynamic terms for a general multi-body. Khalil et al. [14] has proposed a simple and general closed form solution for the dynamic modeling of parallel robots using the formulation of serial manipulators. In this project, for the purpose of dynamic modeling of the closed-chain system, the Lagrangian

formulation has been used along with the projection operator [6] which will be discussed in section **2.4**.

As mentioned earlier, dynamics of the *open chain system* will be calculated in conjunction with projection operator for dynamic modeling of the constrained system. For a serial *n*-link manipulator, the dynamic equation could be written as [8]:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{2-1}$$

where $M(q) \in \mathrm{R}^{n\times n}$ is the inertia matrix, $C(q,\dot{q}) \in \mathrm{R}^{n\times n}$ the centrifugal mass matrix, $G(q) \in \mathrm{R}^{n}$ is the gravity term of the manipulator and $\tau \in \mathrm{R}^{n}$ the torques exerted on each joint.

We make the following assumptions:

1. The motor dynamics of the manipulator is neglected.
2. The motors axes coincide on the joint axis of each link.

By writing the jacobian of the center of the mass of link *i* in the global coordinates (base frame {B}) in the following form:

$$J_i = \begin{bmatrix} J_{\upsilon_i} \\ J_{\omega_i} \end{bmatrix} = \left[\begin{array}{ccc} J_{\upsilon_i}^1 & \cdots & J_{\upsilon_i}^n \\ \hline J_{\omega_i}^1 & \cdots & J_{\omega_i}^n \end{array}\right] \tag{2-2}$$

Where J_{υ_i} and J_{ω_i} are linear and rotational jacobians respectively, $J_{\upsilon_i}^j$ and $J_{\omega_i}^j$ are linear and rotational jacobians of center of mass of link *i* to *j*-th DoF.

The kinetic energy of the system (*K*) could be written using the inertia tensor (*M*):

$$K = \frac{1}{2}\dot{q}^T M(q)\dot{q} \tag{2-3}$$

On the other hand, the kinetic energy could be written by summing all links energy together:

$$K = \frac{1}{2}\sum_{i=1}^{n} \dot{q}^T m_i J_{\upsilon_i}^T J_{\upsilon_i} \dot{q} + J_{\omega_i}^T \dot{q}^T R_i I_i R_i^T J_{\omega_i} \dot{q} \tag{2-4}$$

Thus, *M* could be written explicitly in terms of jacobian and rotation matrix of each link:

$$M = \sum_{i=1}^{n} m_i J_{\upsilon_i}^T J_{\upsilon_i} + J_{\omega_i}^T R_i I_i R_i^T J_{\omega_i} \tag{2-5}$$

Centrifugal and Coriolis tensor (*C*) could be calculated using:

$$\begin{aligned} c_{i,j,k} &= \frac{1}{2}\left(\frac{\partial M_{\{i,j\}}}{\partial q_k} + \frac{\partial M_{\{i,k\}}}{\partial q_j} - \frac{\partial M_{\{j,k\}}}{\partial q_i} \right) \\ C_{\{i,j\}} &= \sum_{k=1}^{n} c_{i,j,k} \dot{q}_k \end{aligned} \tag{2-6}$$

where $M_{\{i,j\}}$ is the (*i, j*) element of *M*, $C_{\{i,j\}}$ is the (*i ,j*) element of *C* and $c_{i,j,k}$ is the Christoffel's symbol (i, j, k) element. Note that for every fixed *i,* $c_{i,j,k} = c_{i,k,j}$. The gravity tensor could be calculated using the following formula:

$$G_{\{i\}} = \sum_{j=1}^{n} \left(m_j g_0^T J_{\upsilon_j}^i \right) \tag{2-7}$$

Also, for checking validity of the generated model the following tests could be performed:

1. $\dot{M} - 2C$ must be skew symmetric
2. In a simulation run, in the absence of joint torques and gravitational force, the total kinetic energy of the system **(2-3)** should remain constant.

To reduce the calculation of tensors during simulation, a fast customized symbolic simplification (implemented in "*Model/MakeFunc.m*") is performed in Matlab. In this method, first all the parentheses in the final formula are selected as patterns. An algorithm searches for each pattern in the main formula and if more than one instance of the pattern is found, it is assigned to a temporary variable. As a result, it will be calculated only once in each simulation iteration. As a

comparison, the number of required operations for the tensors resulted from this method and non-simplified formula is presented in Table 3. An example of generated code of the algorithm could be found in Appendix A (Function Generation)**.**

Table 3: Efficiency of the developed fast simplification method

Tensor	Mode	Original		Number of Temporary Variables	Simplified		Reduction Percentage	
		+	×		+	×	+	×
M	1	15,124	44,776	332	986	1,496	93%	97%
	2	42,348	120,865	566	1,873	3,170	96%	97%
$\frac{\partial M}{\partial q}$	1	143,780	440,720	1,311	4,405	7,322	97%	98%
	2	559,456	1,636,672	3,094	12,865	22,801	98%	99%
Φ	1	525	1,463	54	103	187	80%	87%
	2	524	1,346	57	93	157	82%	88%
A	1	2,438	7,348	203	345	631	86%	91%
	2	3,169	8,856	258	416	734	87%	92%
$\frac{\partial A}{\partial q}$	1	2,073	6,344	174	301	526	85%	92%
	2	4,094	12,565	393	652	1,092	84%	91%

2.3 Constraints Formulation

Constraint equation is the relationship between all of the joint variables of a closed chain system which is in the form of $\Phi(q) = 0$. For the simulation purpose, calculation of constraint equation is necessary. The jacobian of constraint equations $A(q) = \frac{\partial \Phi(q)}{\partial q}$ will be used for control, simulation. In this section, calculation of both Φ and A will be discussed.

Since the reconfiguration process requires constraining of the reconfigurable manipulator end-effector, the constraint equation or Φ could be calculated from the end-effector transformation matrix to the base frame **(2-8)**.

$$\Phi(q) = f({}^{E}_{B}T(q)) = \begin{bmatrix} \Phi_P \\ \Phi_O \end{bmatrix} = 0 \tag{2-8}$$

The constraint equation Φ could be divided into a translational (Φ_P) and a rotational (Φ_O) equation. Extraction of Φ_P with respect to end-effector fixation point (or a surface) is trivial. However due to the complexity of orientation in space, Φ_O may be calculated in several ways which we will focus on here.

There are multiple conventions to express the orientation of a body in 3-dimensional space, such as rotation matrix, Euler angles, angle-axis and Quaternions (or Euler parameters). In this work we use Euler angles since it is the most commonly used representation and unlike many other representations has the minimum 3 parameters which is equal to the degrees of rotation of a body in 3-D space.

Consider $[\alpha \quad \beta \quad \gamma]^T$ to be the end-effector $Z'Y'X'$ Euler angles. Then the rotation matrix could be written in terms of α, β and γ:

$$R_{Z'Y'X'}(\alpha,\beta,\gamma) = R_z^{(\alpha)} R_y^{(\beta)} R_x^{(\gamma)} = \begin{bmatrix} c\alpha\, c\beta & c\alpha\, s\beta\, s\gamma - s\alpha\, c\gamma & c\alpha\, s\beta\, c\gamma + s\alpha\, s\gamma \\ s\alpha\, c\beta & s\alpha\, s\beta\, s\gamma + c\alpha\, c\gamma & s\alpha\, s\beta\, c\gamma - c\alpha\, s\gamma \\ -s\beta & c\beta\, s\gamma & c\beta\, c\gamma \end{bmatrix} \tag{2-9}$$

where $sx \equiv \sin x$ and $cx \equiv \cos x$. Consider the transformation matrix from end-effector frame $\{E\}$ to base frame $\{B\}$ to be:

$${}^{E}_{B}T(q) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-10}$$

By equating (2-9) and rotation matrix from (2-10) the Euler angles may be extracted in terms of joint angles. However there is not always a unique solution for such equation. Using equations

(2-11), Φ_O can be expressed in terms of Euler angles which is valid only when $\beta \in (-\pi/2 \quad \pi/2)$

.

$$\begin{aligned} \alpha(q) &= \arctan2\left(r_{21}(q), r_{11}(q)\right) \\ \beta(q) &= -\arcsin(r_{31}(q)) \\ \gamma(q) &= \arctan2\left(r_{32}(q), r_{33}(q)\right) \end{aligned} \tag{2-11}$$

The general constraint equation could be written in the following form:

$$\Phi(x, y, z, \alpha, \beta, \gamma) = \begin{bmatrix} \Phi_P(x, y, z) \\ \Phi_O(\alpha, \beta, \gamma) \end{bmatrix} \tag{2-12}$$

Consider n_P and n_O to be the number of DoF of position and orientation constraints respectively. In equation **(2-12)**, $\Phi_P(x, y, z)$ is the $n_P \times 1$ position constraint equation and $\Phi_O(\alpha, \beta, \gamma)$ is the $n_O \times 1$ orientation constraint equation.

Calculation of Φ is not always necessary for simulation, since it is only used to correct the error caused by integration during simulation, which will be discussed in section **3.3**. However for more robust simulation where the joints speed are larger or during testing an un-tuned controller, it is necessary to correct the generalized coordinates in order to satisfy the constraint equation. Also for *random sweep of workspace* method (Chapter 7) correction of state is a crucial step of the algorithm. In iterative correction algorithm Φ and A must be consistent i.e. they should have the same number of rows.

If correction is not required, A could be calculated independently using the end-effector jacobian. For example, the jacobian matrix corresponding to full 6-DoF constraint on a manipulator end-effector (A) could be considered equal to the $6 \times n$ jacobian, n being the number of DoF of the open chain manipulator.

In order to calculate $A(q)=\dfrac{\partial\Phi(q)}{\partial q}$ from **(2-12)**, arctan2 may be written in the form of $\text{arctan2}(y,\ x) = 2\arctan\left(\dfrac{\sqrt{x^2+y^2}+x}{y}\right)$ in order to simplify calculation of its derivative.

Equation (2-11) **is** not always valid and the solution may diverge during the correction process of simulation, for example when β exceeds the range $(-\pi/2 \quad \pi/2)$. Depending on the orientation constraint, sometimes it is possible to write Φ_O directly in terms of the rotation matrix ($^{E}_{B}R_{Z'Y'X'}$ in equation **(2-9)**). For example for the angular constraints $\begin{cases}\alpha-\alpha_0=0\\ \beta-\beta_0=0\\ \gamma-\gamma_0=0\end{cases}$, one could write $\Phi_O(q)=\begin{bmatrix} r_{11}-c\alpha_0\ c\beta_0 \\ r_{31}+s\beta_0 \\ r_{32}-c\beta_0\ s\gamma_0\end{bmatrix}$. Due to simplicity of this formulation the computation cost of A will be reduced considerably. However it may not be always possible to write the constraint equation in terms of rotation matrix. For this reason we will introduce the *redundant constraint equation* in the following to avoid this problem.

The existence of redundancy in constraint equation will be handled during the iterative correction, since the projector operator is based on pseudo-inversion of the constraint jacobian; a process which is not conditioned upon the rank of the jacobian [6]. Thus when a full constraint is exerted on the end-effector, a candidate for a rotational constraint equation could be $\Phi_O=\begin{bmatrix} r_{11} & \cdots & r_{33}\end{bmatrix}^T=0$ which would be a 9×1 equation with 6 degrees of redundancy. Without the need to compute any derivative of inverse trigonometric functions, A could also be calculated from $\dfrac{\partial\begin{bmatrix} r_{11} & \cdots & r_{33}\end{bmatrix}^T}{\partial q}$. Aside from simplicity of the resulted expression, this method is more robust especially when the constraint orientation is close to singularities of Euler angles, such as when $\beta_0\approx\pi/2$.

2.4 Projection Operator

In this section a review of projection-based modeling, previously suggested by Aghili [6] for reconfigurable manipulators, will be presented. Projection-operator-based modeling of constraint systems, unlike many other methods, is immune to redundancy of the system and in the existence of singular configurations which makes it a suitable method for modeling of parallel manipulators. Projection operator is defined as:

$$P = I - A^{+}A \tag{2-13}$$

P is the null-space orthogonal projector of A and A is the constraint Jacobian matrix:

$$A = \frac{\partial \Phi}{\partial q} \tag{2-14}$$

Considering the open chain dynamics to be $M(q)\ddot{q} = \tau - V(q,\dot{q})$, the closed-chain dynamic formulation would be:

$$\ddot{q} = M_C{}^{+}\left(P(\tau - V) + C_C\dot{q}\right) \tag{2-15}$$

M_C and C_C could be calculated from:

$$\begin{aligned} M_C &= M + PM - (PM)^T \\ C_C &= -MA^{+}\frac{dA}{dt} \end{aligned} \tag{2-16}$$

where $\frac{dA}{dt} = \frac{\partial A}{\partial q}\dot{q}$. Equation **(2-15)** is the inverse dynamic formulation which is used for simulation.

2.5 Cylindrical Joint Modeling

Lockable passive cylindrical joint is the underlying design innovation of the reconfigurable manipulator [1] which will be reviewed here.

In robotics the cylindrical joint is a joint with two degrees of freedom; a linear and a rotational freedom, both along the cylinder axis. It could be considered as one prismatic joint and one revolute joint serially connected to each other. The cross section of a cylindrical joint is circular.

The passive cylindrical joints used in such manipulator are un-instrumented and passive, or in other words, they are not equipped with any sensors or actuators. This is the major benefit of this kind of reconfigurable manipulators, as they are able to change their reconfiguration without using any extra actuator or sensor except for the brake mechanism. Note that the brake mechanism could be designed in a more compact design in comparison to two extra actuators. The implemented normally locked *brake mechanism* can constrain both degrees of freedom of the cylindrical joint, which means that the link before and after the cylindrical joint will be considered as one solid link. A possible design for a brake mechanism on the cylindrical joints is suggested by Aghili in [15]. When the brake is released, the cylindrical joint can translate and rotate freely. Figure 3 shows schematic movement of a cylindrical joint.

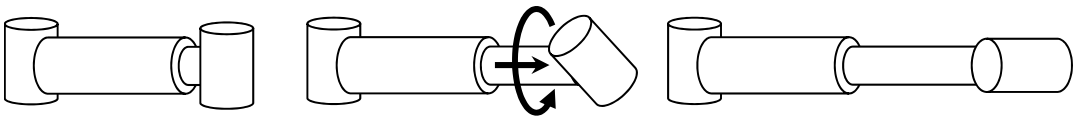


Figure 3: Extension and rotation of a cylindrical link

2.5.1 Cylindrical Joint Decoupling

For modeling of the cylindrical joints, its two degrees of freedom are decoupled into one revolute and one prismatic degree of freedom. Decoupling the kinematic properties is trivial. In the following we focus on decoupling the dynamic properties of a cylindrical joint.

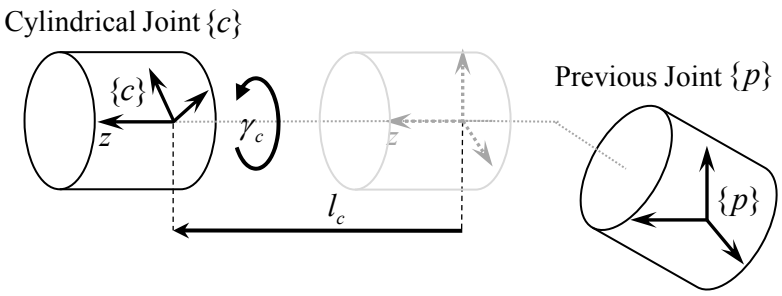


Figure 4: Frame assignment for cylindrical joint

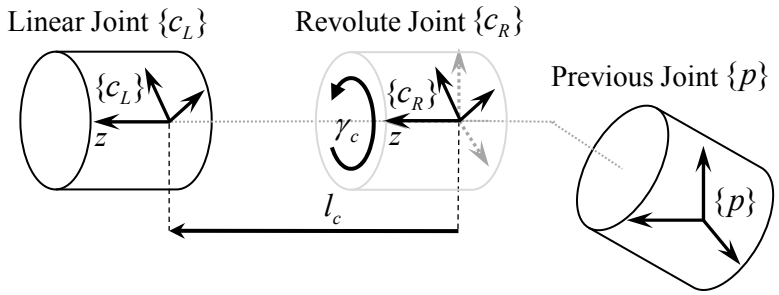


Figure 5: Frame assignment for decoupled cylindrical joint

Let's consider $\{c\}$ to be the cylindrical joint, $\{c_R\}$ the decoupled revolute joint and $\{c_L\}$ as the decoupled linear joint. **Figure 4** and **Figure 5** show the frame assignments for a cylindrical link. As could be seen from the figures frame $\{c\}$ coincides with frame $\{c_L\}$.

Table 4: Parameters of cylindrical links and decoupled links

Description	$\{c\}$	**Decoupled joints**	
		$\{c_R\}$	$\{c_L\}$
Mass	m_c	$m_{c_R} = 0$	$m_{c_L} = m_c$
Moment of inertia	I_c	$I_{c_R} = 0$	$I_{c_L} = I_c$
Total torque	N_c	N_{c_R}	N_{c_L}
Total force	F_c	F_{c_R}	F_{c_L}
Angular velocity (acceleration)	$\omega_c(\dot{\omega}_c)$	$\omega_{c_R}(\dot{\omega}_{c_R})$	$\omega_{c_L}(\dot{\omega}_{c_L})$
Velocity (acceleration) of center of mass	$\upsilon_c(\dot{\upsilon}_c)$	$\upsilon_{c_R}(\dot{\upsilon}_{c_R})$	$\upsilon_{c_L}(\dot{\upsilon}_{c_L})$

The parameters of the cylindrical links and decoupled links are defined in Table 4.

The Euler-Newton formulation of a cylindrical joint is shown in equation (2-17).

$$\begin{aligned} N_c &= I_c\dot{\omega}_c + \omega_c \times I_c\omega_c \\ F_c &= m_c\dot{\upsilon}_c \end{aligned} \tag{2-17}$$

By writing the same equations for revolute and linear links and knowing $m_{c_R} = 0$ and $I_{c_R} = 0$ from Table 4, one could write equation **(2-18)**.

$$\begin{cases} N_{c_R} = 0 \\ F_{c_R} = 0 \end{cases}$$
$$\begin{cases} N_{c_L} = I_{c_L}\dot{\omega}_{c_L} + \omega_{c_L} \times I_{c_L}\omega_{c_L} = I_c\dot{\omega}_{c_R} + \omega_{c_R} \times I_c\omega_{c_R} \\ F_{c_L} = m_{c_L}\dot{\upsilon}_{c_L} = m_{c_L}\dot{\upsilon}_c \end{cases} \tag{2-18}$$

By comparing **Figure 4** and **Figure 5** the following relations could be extracted:

$$\begin{aligned} \omega_{c_R} &= \omega_c \\ \upsilon_{c_L} &= \upsilon_c \end{aligned} \tag{2-19}$$

From equations **(2-18)** and **(2-19)**, equation (2-20) could be concluded.

$$\begin{aligned} N_c &= N_{c_R} + N_{c_L} \\ F_c &= F_{c_R} + F_{c_L} \end{aligned} \tag{2-20}$$

Thus the dynamic properties of the released cylindrical joint could be inherited only to the last degree of freedom.

2.5.2 Locked and Released Mode Properties

The reconfigurable manipulator has multiple modes of operation, as addressed in Section **2**. The dynamics and kinematics characteristics of a reconfigurable manipulator differ in each mode, yet they all originate from the same manipulator. For example when a two-part cylindrical joint is unlocked, the mass and coordinate system of the two new links will be different than those of the

original locked cylinder. In this section the relation of kinematic and dynamic properties between a general unlocked mode and locked mode will be explained.

The locked robot DH table could be defined as Table 5.

Table 5: DH table of locked manipulator

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	α_0	L_0	D_1	θ_1
2	α_1	L_1	D_2	θ_2
...	…	…	…	…
n	α_{n-1}	L_{n-1}	D_n	θ_n

where α_i and L_i, are the twist angle and length of *i*-th link. D_i and θ_i are the length and angle of joint *i*, respectively.

Let's denote ${}^{0}_{B}T$ to be the constant transformation matrix from {0} or the fixed frame on joint 0 to the base frame or {*B*}, and ${}^{N}_{E}T$ the constant transformation from the last joint frame or {*N*} to the end-effector frame or {*E*}. When a passive joint is released, the manipulator gains two additional degrees of freedom and the DH table will grow by two rows. Table 6 shows the DH table of the manipulator when a passive joint implemented on L_k (row *k+1* of Table) is released.

Table 6: DH table of manipulator when passive joint k is released

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	α_0	L_0	D_1	θ_1
2	α_1	L_1	D_2	θ_2
…	…	…	…	…
j+1	$\alpha_j + \gamma_j$	$L_j + l_j$	D_{j+1}	θ_{j+1}
…	…	…	…	…
k+1	$\pi/2$	0	L_k	$\alpha_k + \gamma_k$
	0	0	l_k	0
	$-\pi/2$	0	D_{k+1}	θ_{k+1}
…	…	…	…	…
n	α_{n-1}	L_{n-1}	D_n	θ_n

The rotational and linear degrees of freedom of the cylindrical joint on link k (L_k) are named γ_k and l_k respectively. Consider link k to be equipped with the j-th cylindrical joint. The DH table of the locked manipulator is demonstrated in **Table 7**. The *combined center of mass* or $P_{k\{1,2\}}$ and combined inertia matrix or $I_{k\{1,2\}}$ could be calculated from the equation **(2-21)** and equations **(2-22)** respectively**.**

$$P_{k\{1,2\}} = \frac{m_{k\{1\}} P_{3\{1\}} + m_{k\{2\}} (R_P P_{k\{2\}} + l_j \hat{x})}{m_{k\{1\}} + m_{k\{2\}}} \tag{2-21}$$

Table 7: DH table of a locked cylindrical joint and its neighbour joints

i	α_{i-1}	a_{i-1}	d_i	θ_i	I_i	m_i	P_i
k	α_{k-1}	L_{k-1}	D_k	θ_k	$I_{k\{1,2\}}$	$m_{k\{1\}} + m_{k\{2\}}$	$P_{k\{1,2\}}$
$k+1$	$\alpha_k + \gamma_j$	$L_k + l_j$	D_{k+1}	θ_{k+1}	I_{k+1}	m_{k+1}	P_{k+1}
$k+2$	α_{k+1}	L_{k+1}	D_{k+2}	θ_{k+2}	I_{k+2}	m_{k+2}	P_{k+2}

$$\begin{aligned}
dP_{k\{1\}} &= P_{k\{1,2\}} - P_{k\{1\}} \\
dP_{k\{2\}} &= P_{k\{1,2\}} - P_{k\{2\}} \\
dI_{k\{1\}} &= m_{k\{1\}} \left(dP_{k\{1\}}^T dP_{k\{1\}} I_{3\times 3} - dP_{k\{1\}} dP_{k\{1\}}^T \right) \\
dI_{k\{2\}} &= m_{k\{2\}} \left(dP_{k\{2\}}^T dP_{k\{2\}} I_{3\times 3} - dP_{k\{2\}} dP_{k\{2\}}^T \right) \\
I_{k\{1,2\}} &= I_{k\{1\}} + dI_{k\{1\}} + R_x^{-(\alpha_k + \gamma_j)} I_{k\{2\}} + dI_{k\{2\}}
\end{aligned} \tag{2-22}$$

where $R_P = R_x^{-(\alpha_k + \gamma_j)}$, is the rotation matrix from the second part of the cylindrical joint to the first part. The DH parameters and dynamic properties of a released cylindrical link are demonstrated in **Figure 6**.

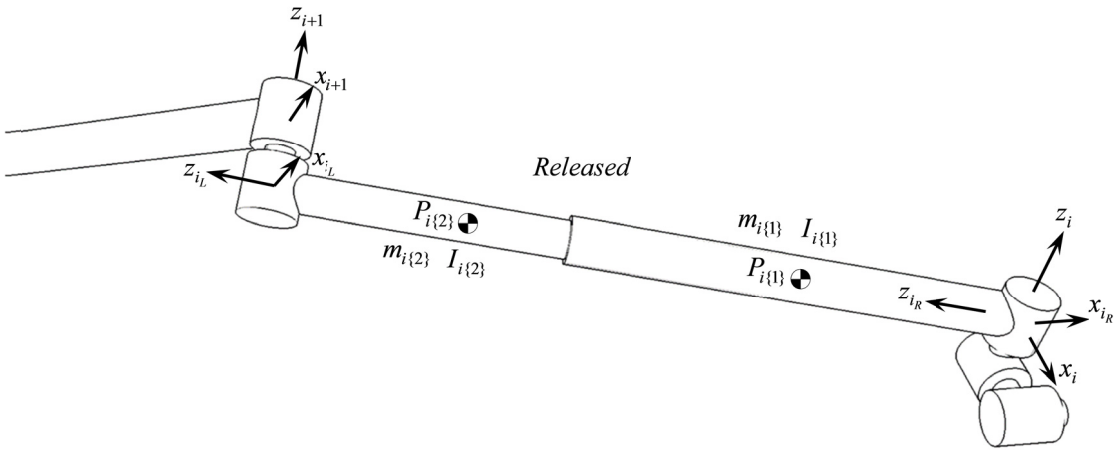


Figure 6: Kinematic and Dynamic properties of a released cylindrical joint

According to **Figure 6**, the DH-table of released cylindrical joints corresponding to link *k* could be derived as in Table 8, where $R_{Cyl_{pre}} = R_z^{(-\pi/2)}$ and $R_{Cyl} = R_x^{(-\pi/2)} R_z^{(-\pi/2)}$.

Table 8: DH table of a released cylindrical joint and its neighbour joints

link		α'_{i-1}	a'_{i-1}	d'_i	θ'_i	I'_i	m'_i	P'_i
k		α_{k-1}	L_{k-1}	D_k	$\theta_k + \pi/2$	$R_{Cyl_{pre}} I_{k\{1\}}$	$m_{k\{1\}}$	$R_{Cyl_{pre}} P_{k\{1\}}$
k+1	Rot_{cyl}	$\pi/2$	0	0	$\alpha_k + \gamma_j$	0	0	0
	Lin_{cyl}	0	0	$L_k + l_j$	0	$R_{Cyl} I_{k\{2\}}$	$m_{k\{2\}}$	$R_{Cyl} P_{k\{2\}}$
	Joint	$-\pi/2$	0	D_{k+1}	$\theta_{k+1} - \pi/2$	I_{k+1}	m_{k+1}	P_{k+1}
k+2		α_{k+1}	L_{k+1}	D_{k+2}	θ_{k+2}	I_{k+2}	m_{k+2}	P_{k+2}

Table 9: DH table of a locked 7-DoF reconfigurable manipulator

i	α_{i-1}	a_{i-1}	d_i	θ_i	I_i	m_i	P_i
1	α_0	L_0	D_1	θ_1	I_1	m_1	P_1
2	α_1	L_1	D_2	θ_2	I_2	m_2	P_2
3	α_2	L_2	D_3	θ_3	$I_{3\{1,2\}}$	$m_{3\{1\}} + m_{3\{2\}}$	$P_{3\{1,2\}}$
4	$\alpha_3 + \gamma_1$	$L_3 + l_1$	D_4	θ_4	$I_{4\{1,2\}}$	$m_{4\{1\}} + m_{4\{2\}}$	$P_{4\{1,2\}}$
5	$\alpha_4 + \gamma_2$	$L_4 + l_2$	D_5	θ_5	I_5	m_5	P_5
6	α_5	L_5	D_6	θ_6	I_6	m_6	P_6
7	α_6	L_6	D_7	θ_7	I_7	m_7	P_7

The DH parameters corresponding to the locked and unlocked configurations of a 7-DOF manipulator are respectively listed in Table 9 and Error! Not a valid bookmark self-reference.. In Appendix B (Conversion Algorithm) a conversion algorithm is presented for reconfigurable manipulators.

Table 10: DH table of a released 7-DoF reconfigurable manipulator

i'		α'_{i-1}	a'_{i-1}	d'_i	θ'_i	I'_i	m'_i	P'_i
1		α_0	L_0	D_1	θ_1	I_1	m_1	P_1
2		α_1	L_1	D_2	θ_2	I_2	m_2	P_2
3		α_2	L_2	D_3	$\theta_3 + \pi/2$	$R_{Cyl_{pre}} I_{3\{1\}}$	$m_{3\{1\}}$	$R_{Cyl_{pre}} P_{3\{1\}}$
4	Rot_{cyl}	$\pi/2$	0	0	$\alpha_3 + \gamma_1$	0	0	0
	Lin_{cyl}	0	0	$L_3 + l_1$	0	$R_{Cyl} I_{3\{2\}}$	$m_{3\{2\}}$	$R_{Cyl} P_{3\{2\}}$
	$Joint$	$-\pi/2$	0	D_4	$\theta_4 - \pi/2 + \pi/2$	$R_{Cyl_{pre}} I_{4\{1\}}$	$m_{4\{1\}}$	$R_{Cyl_{pre}} P_{4\{1\}}$
5	Rot_{cyl}	$\pi/2$	0	0	$\alpha_4 + \gamma_2$	0	0	0
	Lin_{cyl}	0	0	$L_4 + l_2$	0	$R_{Cyl} I_{4\{2\}}$	$m_{4\{2\}}$	$R_{Cyl} P_{4\{2\}}$
	$Joint$	$-\pi/2$	0	D_5	$\theta_5 - \pi/2$	I_5	m_5	P_5
6		α_5	L_5	D_6	θ_6	I_6	m_6	P_6
7		α_6	L_6	D_7	θ_7	I_7	m_7	P_7

3. Control and Simulation

The reconfigurable robot has two general phase of control: reconfiguration control and motion control. Reconfiguration control may itself contain several modes depending on the number of cylindrical joints and other design parameters.

3.1 Design

A simple scenario of the reconfiguration could be as follows: before reconfiguration starts the end-effector grasps a fixed point then the cylindrical joints are released and the reconfiguration is performed, finally the cylindrical joints are locked and end-effector releases the fixed point (Figure 1). In another words, during reconfiguration the actuation performs passive joint control rather than end-effector control.

When a cylindrical joint is released, the number of degrees of freedom of system will increase. However, the total number of actuators remains the same, since there are no actuators associated with a cylindrical joint. On the other hand the end-effector is constrained to a *fixation point* which reduces the total freedom of the system. To avoid having an under-actuated system and to be able to control all the released passive DoFs to the desired values, the following design parameters should be consistent:

1. Number of constrained degrees of freedom at end-effector (r)
2. Number of actuated joints or active degrees of freedom (n).
3. Number of released passive joints (m)

In order to avoid the system to be over-constrained and under-actuated the following inequality must hold:

$$2m \leq r \leq \min(n,6) \tag{3-1}$$

When a manipulator is equipped with more than one cylindrical joint, it may not be possible to release all the passive joints simultaneously, since according to the inequality (3-1) there is a limit for number of released joints for each manipulator. Thus, for some manipulators in some cases the reconfiguration is performed through multiple phases in which, different passive joints are released. In this case the manipulator model would be different for each phase so as its kinematics and dynamics. We define the different configurations that a reconfigurable manipulator acquires during the reconfiguration, as *reconfiguration mode* and assign the following design information to it:

1. The constrains that are being exerted on end-effector
2. Released passive joints and locked passive joints (if any)

Each manipulator may need to enter one or more reconfiguration modes in order to achieve the desired DH parameters. It should be noted that there may be infinite possible scenarios leading to same configurations. For example, a 6-DoF reconfigurable manipulator $(n = 6)$ with two cylindrical joints $(2m = 2 \text{ or } 4)$ and with a full constrained end-effector $(r = 6)$, such manipulator can acquire three different modes; ***mode 1*** is when both cylindrical links are released, ***mode 2*** and ***mode 3*** are when only one of the cylindrical links is released. The manipulator may achieve a desired configuration for both cylindrical joints using *mode 1* without entering *mode 2* and *3*. It is also possible to achieve a desired configuration with acquiring any combination of modes consecutively. The former scenario may be useful to deal with existence of obstacles and singularities.

In the previous example *mode 2 and 3* have four degrees of redundancy while *mode 1* has two degrees of redundancy. Thus, the manipulability and controllability of the manipulator in *mode 2 and 3* is expected to be higher than *mode 1*, since the ratio of number of actuators to outputs is higher.

3.2 Control

Parallel manipulators have been extensively studied by Merlet [7]. Many parallel manipulators are redundant so as reconfigurable manipulator in some of its modes. The redundant actuation could be used to achieve secondary goals other than controlling the end-effector. Such as obstacle avoidance [16], minimizing the joint forces [17] or keeping the joints in their desired limit [18]. An overall study on redundancy resolution could be found in work of Chiaverini et al. [19].

Constraint equation of a general parallel manipulator is of the form $\Phi(u,o,p)=0$ which is a function of inputs or u, outputs or o and passive joints or p. For some parallel robots it is possible to write the constraint equation excluding the passive joints (which are not output) i.e. $\Phi(u,o)=0$. For example in the reconfigurable manipulator the constraint equation could be written only as a function of inputs and outputs as described in section **2.3**. Considering $q=\begin{bmatrix} q_a \\ q_p \end{bmatrix}$ as the vector of joints position, q_a to be the active joints position (or system input) and q_p the cylindrical joints position (or system output); by differentiating the constraint equation we get:

$$\frac{\partial \Phi(q_a,q_p)}{\partial q}\dot{q}=\frac{\partial \Phi}{\partial q_a}\dot{q}_a+\frac{\partial \Phi}{\partial q_p}\dot{q}_p=0 \qquad (3\text{-}2)$$

By defining $A_a=\frac{\partial \Phi}{\partial q_a}$ and $A_p=\frac{\partial \Phi}{\partial q_p}$, the equation of jacobian could be rewritten as:

$$A_a\dot{q}_a+A_p\dot{q}_p=0 \qquad (3\text{-}3)$$

A_a (A_p) is an instantaneous mapping from active (passive) joint space to *constraint space* in the velocity level. Equation (3-3) is the original relation between input and output velocity. One solution for mapping between the input and output velocity space is:

$$\begin{aligned} \dot{q}_a &= -inv(A_a)A_p\dot{q}_p \\ \dot{q}_p &= -inv(A_p)A_a\dot{q}_a \end{aligned} \tag{3-4}$$

Since A_a and A_p may be non-square matrices, function inv is used as a general inverse which will be discussed later. Let's assume that there are *r* degrees of freedom constraints being exerted on end-effector, a is the number of active joints and p is the number of released passive joints. Dimension of A_a and A_p would be $r \times a$ and $r \times p$ respectively. To satisfy the controllability condition, as mentioned in the previous section, $p \le r \le a$ must be true. It should be remarked that whenever $p < r$ or $r < a$ is true, according to inequality (3-1), $p < a$ would be true and thus the system would be redundant.

When $r < a$, there are infinite solutions for $\dot{q}_a$ in equation (3-3), since the number of variables are more than number of equations. A general solution could be given using the *null-space operator* of A_a :

$$\dot{q}_a = -inv(A_a)A_p\dot{q}_p + (I - inv(A_a)A_a)\dot{X}_a \tag{3-5}$$

In (3-5), $(I - inv(A_a)A_a)$ is the null-space operator which maps the arbitrary $1 \times a$ vector $\dot{X}_a$ to the null-space of A_a. It should be mentioned that when $r = a$, A_a would be a square matrix and only one solution will exist for active joints velocity i.e. $\dot{q}_a = A_a^{-1}A_p\dot{q}_p$.

If $p < r$, to extract $\dot{q}_p$ from equation (3-3) number of equation will be more than number of variables, thus in case of consistency of equations, there exist only one solution for $\dot{q}_p$, however there will be infinite number of equations resulting in the same answer:

$$J = -(inv(A_p) + (I - A_p inv(A_p))X_r)A_a$$
$$\dot{q}_p = J\dot{q}_a \tag{3-6}$$

where J is the jacobian matrix of the system, $(I - A_p inv(A_p))$ is the projector to null-space of $inv(A_p)$ and X_r is an arbitrary $1 \times r$ vector. When $p = r$, A_p would be a square matrix and $\dot{q}_p = A_p^{-1} A_a \dot{q}_a$. The general inverse function (inv) in equations (3-5) and (3-6), could be left inverse: $A_x^{-1} = (A_x^T A)^{-1} A^T$, Moore-Penrose pseudo-inverse A_x^+ or any other general inverse with the property of $inv(A_x)A_x = I$. However in formula (3-6) since the solution is singular, an equation with the minimum computation may be used such as left inverse. The formulas proposed in (3-5) and (3-6) could be used for trajectory generation, control and estimation purposes respectively.

The redundancy of the system could be used for obstacle avoidance or minimizing norm of torque or constraint force. Aghili et al. [1] has proposed a controller based on null-space projection operator of A, which minimizes the constraints force:

$$\tau_\theta = P_{11}^+ P_{12} \left(K_p (q_p - q_{p_{des}}) + K_d \dot{q}_p \right) \tag{3-7}$$

τ_θ is the torque applied to active joints, and P_{11} is the active rows and columns of projection operator of A as in (2-13) thus a $a \times a$ matrix, and P_{12} is the active rows and passive columns of P, thus a $a \times p$ matrix. K_p and K_d are PD control gains.

Since it is possible to estimate the passive joints position and velocity with respect to those of active joints which are instrumented with sensors, the passive joints are not instrumented with sensors in this design.

Shall the controller perform reliably and the system remain controllable, the manipulator should not encounter singularities along its trajectory. The singularities of a specific type of parallel

manipulator which includes reconfigurable manipulators will be studied in Chapter **5**. However for the reconfigurable manipulator specifically, singularities of A_p has been studied by Aghili et al. [2], and it is concluded that if and only if two cylindrical joints are parallel, A_p would be rank deficient and thus the system state would be at an *undermobility* singularity (see Chapter **5**). Singularities of A_a for the special case of full position lock on the end-effector (full position end-effector constraint) has been also studied by Aghili et al. in [20]. It is also pointed out in [20] that the condition $R(A_p) \in R(A_a)$ must always hold for the system to be controllable, where $R(A_x)$ is the range of matrix A_x. If A_a is of full rank then the range of A_a would cover the constraint space so the condition would hold. If A_a is rank deficient, although the system is at an *undermobility* singularity (see Chapter **5**), the system can remain controllable only if A_p has no component in the direction(s) of deficiencies of A_a. The former situation is very unlikely to happen and it would be very difficult to study; thus it is not taken into account in this work. As a result the controllability of the system is only considered to be a function of singularities of A_a and A_p which will be discussed in more details in Chapter **5**.

3.3 Simulation

It is possible to simulate reconfiguration process as well as pre and post-reconfiguration phases along with real-time visual feedback, in the Matlab package developed for reconfigurable manipulators (see Chapter **4**). The simulation can be run in variable time-step mode to decrease the simulation computation. The variable time-step is designed to be a function of the joints acceleration.

During simulation, the integration will lead to a drift error of q. It is possible that the constraint equation be no longer satisfied i.e. $\Phi(\tilde{q}) \neq 0$, with $\tilde{q}$ being the drifted q. This can lead to instability of the system during simulation with un-tuned controller. An iterative solution for correction of

this drift has been proposed by Aghili [6] which will be reviewed here. By defining the drift error as $\delta q = q - \tilde{q}$ and using the first two terms of the Taylor series for the constraint equation $\Phi = 0$ around $\tilde{q}$ we would have:

$$\Phi(q) = \Phi(\tilde{q} + \delta q) \cong \Phi(\tilde{q}) + \frac{\partial \Phi(\tilde{q})}{\partial q} \delta q \tag{3-8}$$

where $\frac{\partial \Phi(\tilde{q})}{\partial q} = A(\tilde{q})$, therefore:

$$\Phi(\tilde{q}) + A(\tilde{q})\delta q \cong \Phi(q) = 0 \tag{3-9}$$

From **(3-9)** it can be inferred that if $\tilde{q}$ is close enough to q, using the following formula iteratively $\tilde{q}$ will approach to q.

$$\tilde{q}_{k+1} \cong \tilde{q}_k - A^+(\tilde{q}_k)\Phi(\tilde{q}_k) \tag{3-10}$$

where using pseudo inverse least square solution of δq will be acquired in each iteration.

The error compensation of q does not necessarily lead to compensation of the error of $\dot{q}$ and so using the similar formulation to correction of q it is possible to compensate the drift error of $\dot{q}$ by using the following formula iteratively until a desired accuracy is reached:

$$\dot{\tilde{q}}_{k+1} \cong \dot{\tilde{q}}_k - A^+(q)A(q)\dot{q}_k \tag{3-11}$$

The two formulas (3-10) and (3-11) are core formulas for the *iterative correction algorithm* of joint positions and velocities respectively. The two algorithms are shown in Figure 7 and Figure 8. Note that to decrease the computation time of the iterative loop in the first algorithm A^+ is calculated once outside the loop and is considered to be constant. Such assumption is only valid when the joints errors are very small, but when it is not, A^+ may need to be recalculated in the first few iterations, until it is smaller than a specific threshold so that the algorithm converges. Such method is also used in random sweep of workspace (Section **7.2**).

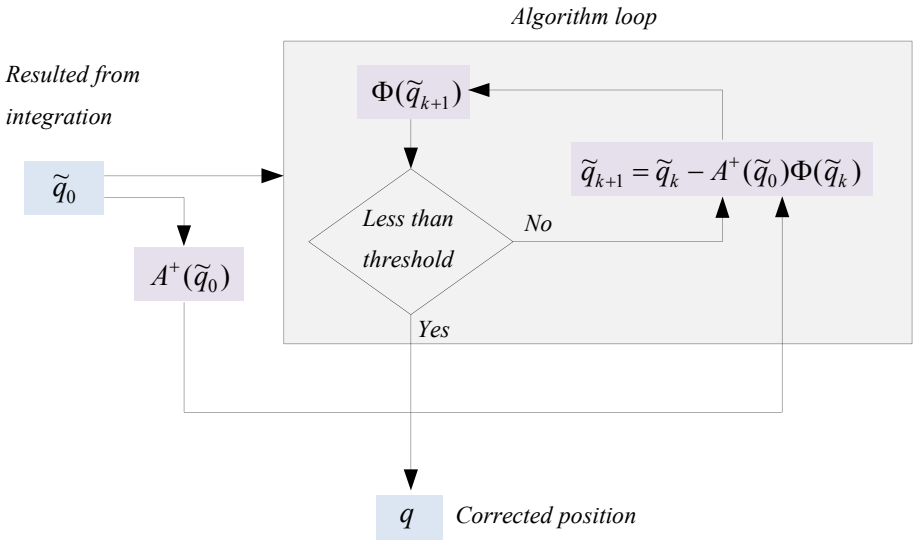


Figure 7: Joints position iterative correction algorithm

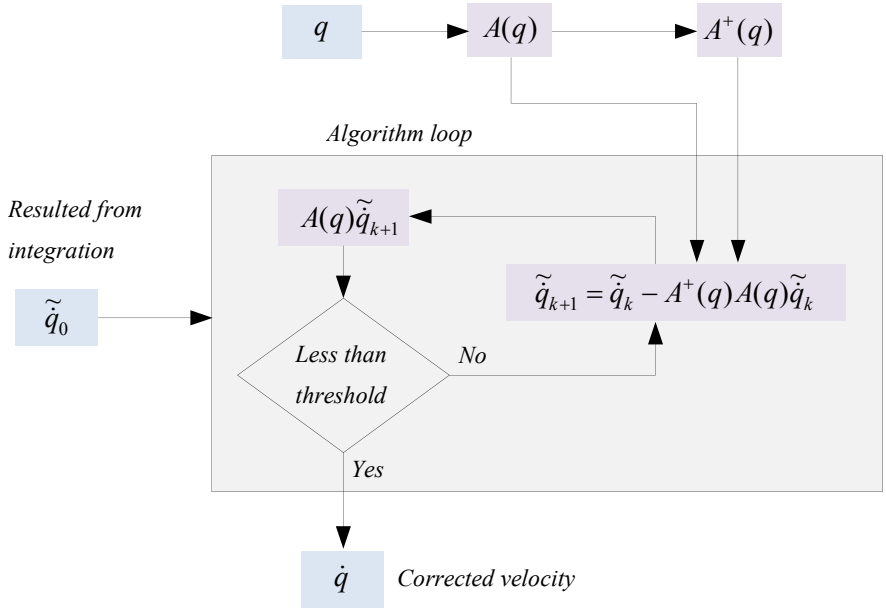


Figure 8: Joints velocity iterative correction algorithm

To check the validity of the generated model, in a simulation run and in the absence of joint torques and gravitational force, the total kinetic energy of the system could be monitored, knowing that it must remain constant.

4. Matlab Package

A Matlab package is developed for control, modeling and simulation of the reconfigurable manipulators with lockable cylindrical joints. A tutorial video is also prepared as a getting-started at [21]. The user can evaluate his/her controller for different manipulators with different conditions during both massless or semi-static and dynamic simulations. A snapshot of the main user-interface of the package could be seen in **Figure 9**.

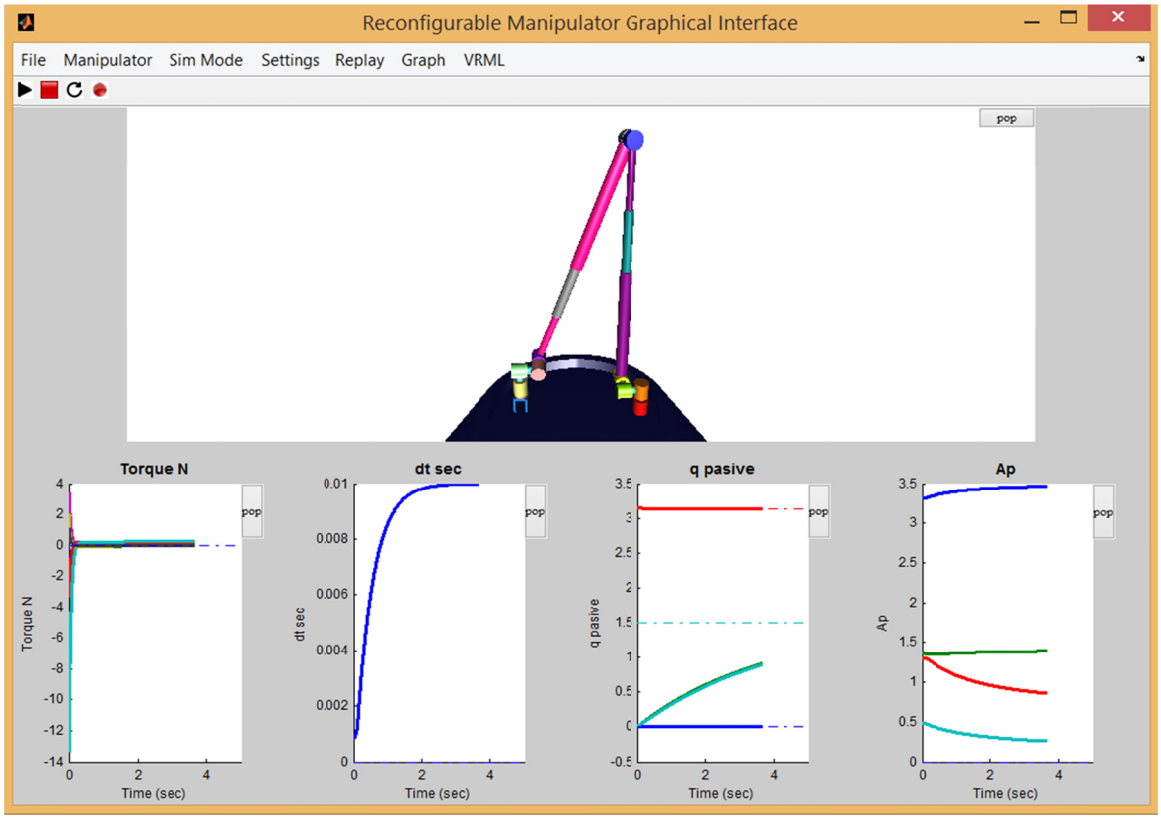


Figure 9: A view of main GUI of Matlab package

Properties of the manipulator could be inserted through an interactive interface. Both mathematical and 3D models are generated automatically and are accessible into the main user

interface. Simulations could be run directly from the main interface. The class-based structure of the program allows the package features to be used in the user's custom Matlab code, without the interference of user-interface (Section **4.3**).

4.1 Structure

There are five classes included in the package:

1. "RecMan" is the main class and it includes all other classes as well as manipulator properties, simulation, control and all other high level functions.
2. "Model" is the responsible class for modeling of manipulator and includes the conversion explained in Section **2.5.2** and simplification in Section **2.2**.
3. "GUI" class includes all the graphics handling.
4. "VR" performs the 3-D model auto-generation and its handling.
5. "Ang" is the class containing all the required angular conversion functions.

When a manipulator is built using the **Model.Build**, a folder with the name of the manipulator is created and all the related files, functions and tensor matrices of the manipulator are placed within it. Some files are generated separately for each mode. Tensor functions are listed in Table 11, and other files in Table 12.

Table 11: Tensor functions generated by Matlab package

Function	Description
Phi	Constraint equation: Φ
A	Jacobian of Φ : A
A2q	Jacobian of A: $\frac{\partial A}{\partial q}$
EEB	Transformation matrix from end-effector to base: T_{EE}^{B}

JEEB	Jacobian of T_{EE}^{B}: $\frac{\partial T_{EE}^{B}}{\partial q}$
ELB	Transformation matrix from elbow of manipulator to base: T_{EL}^{B}
JELB	Jacobian of T_{EL}^{B}: $\frac{\partial T_{EL}^{B}}{\partial q}$
IM	Inertia Matrix: M
IM2q	Derivative of Inertia matrix (M) with respect to q : $\frac{\partial M}{\partial q}$
G	Gravity Matrix: G

Table 12: Manipulator files generated by Matlab package

File/Function	Description
Props.mat	Manipulator properties
Mode.mat	matfile containing information about joints including information about number of DoF and which of them are active or passive. It is dependent on the mode of manipulator
List.mat	List of symbolic data of all manipulator tensors
Read.m	Calculates a tensor
VR_World.wrl	VRML (*.wrl) file of 3D model
VR_Update	Updates 3D model world
Comp.mat	Matfile containing the information about how computation cost is effected using the simplification implemented by **Model.MakeFunc**
Settings.mat	Contains the control, states and replay camera trajectory information

4.2 Graphical Interface

An interactive error handling GUI is implemented which notifies user of input errors (**Figure 10**).

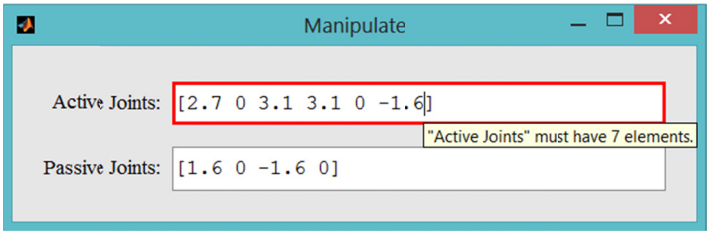


Figure 10: Error handling of GUI input dialogs

Manual control and model generation windows are demonstrated in **Figure 11** and **Figure 12** respectively.

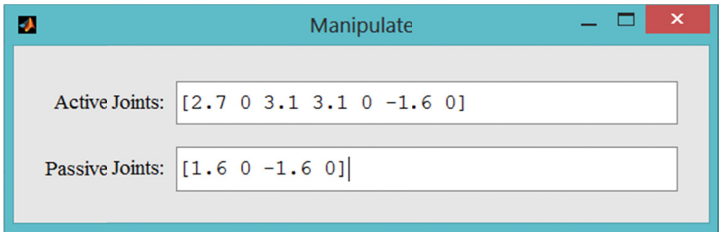


Figure 11: Manual control of the manipulator

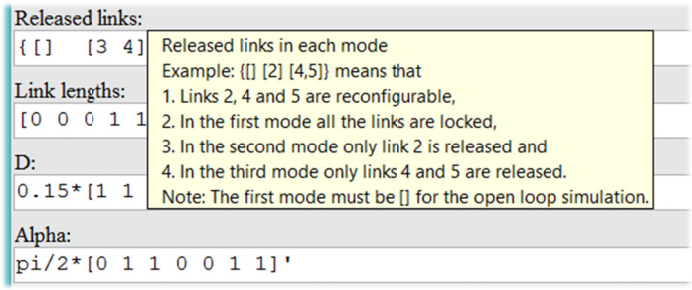


Figure 12: GUI help tooltip

4.3 Command-Line Functionality

All the methods of the package are accessible through the main class (**RecMan**) by bypassing the graphical user-interface. A list of Matlab package methods and their descriptions are presented in **Table 13**.

Table 13: Matlab package classes and their methods

Class	Method	Description
RecMan	**Simulate**	Reconfiguration simulation
	SimOpen	Open chain simulation
Model	**Build**	Creates the model of the manipulator, all the tensors functions will be placed into the manipulator folder
	MakeFunc	Converts a symbolic formula into an m-function which is simplified using temporary variables, (used within **Model.Build**)
	MultHalf	Multiplies two matrices efficiently to get only the upper half of the result (used within **Model.Build**)
	Transf	Returns transformation matrix based on screw theory inputs (DH parameters), (used within **Model.Build**)
	Reader	Creates a **read.m** function which provides the access to all generated functions by **Model.Build** specific for each manipulator
GUI	**Open**	Opens the main GUI figure containing the access to all other GUI windows and 3D model
	Control_Settings	Opens a window to input the control parameters
	Control_Settings	Opens a window to input the control parameters
	Build_Manip	Opens a window to input manipulator parameters
	Manipulate	Opens a window which enables the manual control of the manipulator
	Replay	Replays the animation of the most recent simulation
	GraphUpdate	Updates the graphs in the main interface
	Get_CamTrj	Opens a GUI for camera trajectory input
VR	**Build**	Creates a ***.wrl** (VRML) file associated with the manipulator and contains all the 3D model information
	SetCamera	Sets the camera position and orientation in 3D model world
	Updater	Writes the **VR_Update** function which updates the 3D model with respect to current state
Ang	**rotx, roty, rotz**	Rotation matrix of rotation around x, y and z axis
	euler2r, r2euler	Conversion from Euler angles to rotation matrix and vice versa
	r2aa, aa2r	Conversion from rotation matrix to angle-axis and vice versa

To calculate a tensor of a previously generated model, in a given state the **Read** function can be used. The function prototype has the form **Read(IN, Name)**, where **Name** is the name of the tensor to be calculated and **IN** must be a structure with the following fields:

Table 14: Necessary fields for Read function data input

Field	Description
Mode	A Mode of the manipulator (scalar)
EP	End-effector position (3 by 1 vector)
EO	End-Effector Euler orientation (3 by 1 vector)
State	State of the manipulator (position of all DoFs followed by velocity of them)
plus	Locked passive joints position

5. Singularity Analysis

In this chapter, first, an introduction to singularities of parallel robots is presented. Then, we focus on the analysis and classification of singularities of a specific group of parallel manipulators. Singularities of a simple parallel manipulator are analysed in details as a case study.

5.1 Literature Review

Merlet and Gosselin [22] define the singularity analysis as:

1. Characterizing singularities (Chapter **5**)
2. Introducing performance indices or measure of closeness to singularity (Chapter **6**)
3. Detecting existence of singularities in workspace or along a given trajectory (Chapter **7**)

In this chapter we will focus on introducing and categorizing singularities. Singularities have been extensively studied in the literature due to their importance in control and design of systems. Singular configuration of a serial manipulator is a configuration in which the end-effector of the system loses one or more degrees of freedom. However, in parallel manipulators there exists another type of singularity where the end-effector or passive joints gain additional degrees of freedom or in other words the system "loses its inherent infinite rigidity" [7].

Analysis of singularities is based on studying the constraint equation of a mechanism which is not unique. Constraint equation of a general parallel manipulator is of the form $\Phi(q_i, q_o, q_p) = 0$ which is a function of inputs (q_i), outputs (q_o) and passive joints (q_p). Constraint equation

$\Phi = 0$ is a *smooth nonlinear function*. Singularities of parallel manipulators has been first studied by Gosselin and Angeles [23]. They considered the system to be non-redundant and it was assumed that the constraint equation could be formulated as $\Phi(q_i, q_o) = 0$. Later, Zlatanov et al. [24] has extended the singularity classification for a general mechanism and introduced new singularity types that could only be studied by taking the passive joints (other than outputs) into account. In a later work [25], they studied the consistency of different types of singularities.

For some parallel robots, it is possible to eliminate the passive joints from the constraint equation i.e. $\Phi(q_i, q_o) = 0$. In this work we only focus on the specific type of manipulators which can be formulated in this way and are non-redundant or redundant. We will call the formulation as *reduced constraint equation*, and the corresponding parallel manipulator as *IO-state manipulator.*

The definitions made by Gosselin and Angeles [23] would cover the singularity types of an IO-state manipulator, which will be reviewed here.

Consider the time derivative of the constraint equation $\left(\Phi(q_i, q_o) = 0\right)$:

$$A_i \dot{q}_i + A_o \dot{q}_o = 0 \tag{5-1}$$

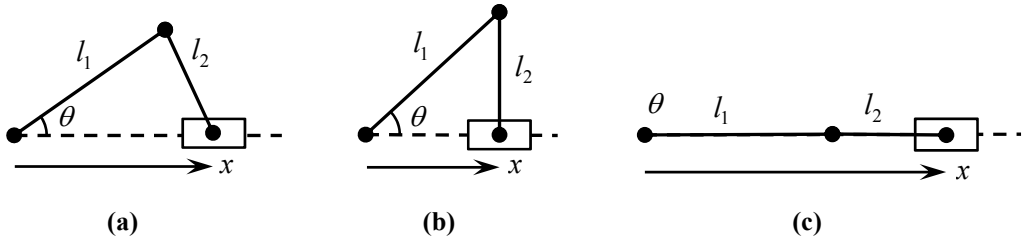
where $A_i = \dfrac{\partial \Phi}{\partial q_i}$ and $A_o = \dfrac{\partial \Phi}{\partial q_o}$ are the partial derivatives of Φ with respect to inputs and outputs respectively. Based on this formulation three types of singularities are defined [23]:

1. According to equation (5-1), if A_i becomes singular or rank deficient, when solving for $\dot{q}_o$ the equation of velocity would become linearly dependent for $\dot{q}_o$ and thus $\dot{q}_o$ would be linearly dependant (with an order equal to the order of rank deficiency of A_i). In such configurations the output loses instantaneous motion in one or more direction and it might be at a singularity barrier. The force (respectively motion) amplification factor from input to output would become infinity (zero).

2. Similarly, when A_o becomes singular or rank deficient, $\dot{q}_i$ would become instantaneously linearly dependant. In this type of singularity the force amplification factor between input and output would become zero and motion amplification factor would become infinity. As a result the system would gain some degrees of freedom and no load could be supported by the outputs. It is also reported that the risk of breakdown will be increased near such singularities due to small force amplification factor [7], or it may be impossible to initiate the motion from such configuration without any external help [26].
3. The third type of singularity happens when first and second types of singularities coalesce. Existence of such singularity is dependent on the design of manipulator and thus may always be absent in some designs.

Pierrot in [27] names first and second types of singularities as *undermobility* and *overmobility* respectively. We will use his notation from now on. The term undermobility is taken from the fact that the output loses some degrees of freedom. While in overmobility singularity the output gains additional freedom and would become loose. Since the third type of singularities (named *configuration singularity* by Gosselin et al. [23]) is not an independent kind of singularity, it is not explicitly studied in this work; however it could be easily inferred. Refer to section **5.2** for a visualized example of all types of singularities.

As an example, two singular configurations are shown for a crank-slider system in Figure 13.



(a): a non-singular configuration, (b, c): two singular configurations

Figure 13: Crank and slider system in a few configurations

Apparently, the definitions of singularities are dependent on the selection of actuated joints and outputs. For example in the crank-slider mechanism in Figure 13, if θ is considered as input and x as output, then the singular configuration (b) would be an overmobility singularity where no force could be maintained at x (and along x). Furthermore no movement can be initiated starting using actuators at this configuration and system would be momentarily loose. Considering the same input-output declaration (θ : input, x: output), Figure 13 (c) shows an undermobility singularity where instantaneous motion of x is 0 and input to output force gain is infinity. However, if x is considered as input and θ as output then the singularity types would be the opposite of the last case i.e. (b) an undermobility and (c) an overmobility singularity.

Both serial and parallel manipulators may have multiple inverse kinematic solutions for a single output. Singularities may be a kinematic branch in either input or output space. A kinematic branch is the intersection of two or more kinematic solutions. For example in case of θ : input, x: output in Figure 13, configuration (c) is a kinematic branch since the system can change configuration between elbow-up and elbow down series of solutions. Relation of singularities to kinematic solutions will be addressed more specifically in Chapter **7**.

In platform-type parallel manipulators such as Stewart, there exist only undermobility singularities [7]; however, for the reconfigurable manipulator both types of overmobility and undermobility could exist in the workspace.

It is worth mentioning that redundancy will decrease the overmobility singularities of a parallel mechanism, since more actuators exist to control the outputs. A_i has equal or more columns than rows since number of constraints are less than or equal to the number of actuators, and as a result its rank is equal to rank of its rows. Redundancy would increase the number of columns while maintaining the number of rows. Since more number of vectors has to become linearly dependant

(in the view of column rank) for A_i to lose rank, possibility of rank deficiency of A_i will decrease, in contrast to a non-redundant case.

5.2 A Five-bar Linkage Case Study

This section provides a case study for the singularity analysis discussed in Section 5.1. It will also be used as a case study for singularity avoidance and workspace analysis in Chapters **6** and **7**.

Four and five-bar mechanisms, due to their simplicity has been used to demonstrate singularities types of close loop mechanisms [24, 28]. Fallahi et al. [29] has studied the workspace of a specific kind of five-bar linkage. In this section, we study the singularities of a similar mechanism.

We consider a 5-*R* manipulator where the two joints connected to ground are actuated and the middle joint position is output. The system has two degrees of freedom thus the workspace is 2-D and the singularity contours and boundaries are 1-D. The system schematic is shown in Figure 14.

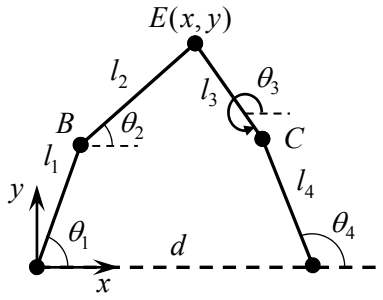


Figure 14: Schematic of five–bar linkage

The two actuated joint angles are θ_1 and θ_4, and (x, y) is the output which is the end-effector position. Although for such mechanism, an explicit relation between output and input kinematics could be extracted; however, we write the constraint equations in the format of a general parallel robot to include all system variables:

$$\Phi = \begin{bmatrix} l_1c_1 + l_2c_2 + l_3c_3 - l_4c_4 - d \\ l_1s_1 + l_2s_2 + l_3s_3 - l_4s_4 \\ l_1c_1 + l_2c_2 - x \\ l_1s_1 + l_2s_2 - y \end{bmatrix} = 0 \qquad (5\text{-}2)$$

where $s_i = \sin(\theta_i)$, $c_i = \cos(\theta_i)$, Φ is the constraint equation with 6 variables and 4 independent equations and [x y] is the coordinates of the end-effector.

Another possible formulation is to remove the passive joints (θ_2 and θ_3) from the constraint equation. By equating the distance of point E from point B and C with l_2 and l_3 the reduced-constraint equation could be written:

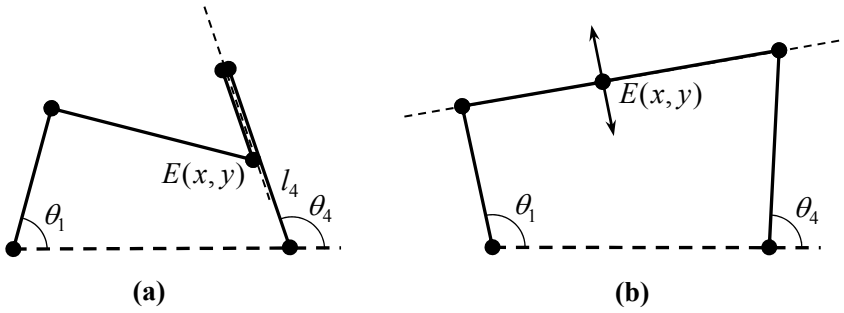
$$\Phi_r = \begin{bmatrix} (x - l_1c_1)^2 + (y - l_1s_1)^2 - {l_2}^2 \\ (x - l_4c_4 - d)^2 + (y - l_4s_4)^2 - {l_3}^2 \end{bmatrix} = 0 \qquad (5\text{-}3)$$

Note that Φ_r is only a function of inputs and outputs and includes no other variable. Φ_r could be used to calculate the measure of singularity and detect the singularity direction. This constraint formulation is possible only for some parallel manipulators such as the reconfigurable manipulators being discussed in this thesis. For singularities analysis purposes we only use Φ_r.

Let's calculate singularities of five-bar linkage for further singularity analysis. It could be proved that a singularity occurs when and only when the difference of two consecutive angles becomes a multiple of π [29], which is:

$$\theta_i = \theta_{i+1} + n\pi, \quad n \in Z, \quad i = 1, 2, 3 \qquad (5\text{-}4)$$

The geometrical interpretation is when two consecutive links are along each other. Two singular configurations are shown in Figure 15.



(a) undermobility singularity, (b) overmobility singularity

Figure 15: Two singular configuration of five-bar linkage

Figure 15 (a) illustrates an undermobility singularity because no movement of output (point *E*) is feasible along link 4. In (b) the end-effector is temporarily loose in the specified direction, thus in an overmobility singularity.

An undermobility singularity occurs when in solution set (5-4), i=1 or 3 and overmobility singularity happens when i=2.

Now we will calculate the loci of singularities in both input and output space. In all types of singularities two links have the same direction thus the system could be thought of as a four-link mechanism.

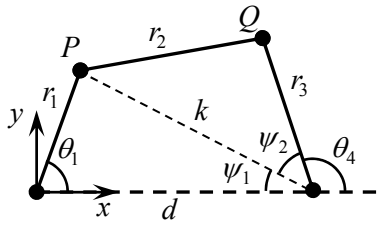


Figure 16: A five-bar linkage when it is in a singularity configuration

According to Figure 16, the relation between θ_1 and θ_4 is extracted using the following equations:

$$
\begin{aligned}
&\theta_4 = \pi - \psi_1 \pm \psi_2 \\
&\psi_1 = \text{atan2}(P_y, d - P_x) \\
&\psi_2 = \arccos\left(\frac{k^2 + r_3^2 - r_2^2}{2kr_3}\right) \\
&k = \sqrt{(P_x - d)^2 + P_y^2} \\
&P = r_1 \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{bmatrix}
\end{aligned}
\tag{5-5}
$$

Depending on which two consecutive links of five-bar linkage have the same direction, the end-effector position could be calculated as in (5-6):

$$
\begin{array}{l|l}
\bullet\ if\ \cos(\theta_1 - \theta_2) = 1 & \bullet\ if\ \cos(\theta_2 - \theta_3) = 1 \\
\quad E = (l_1 + l_2)\begin{bmatrix} c_1 \\ s_1 \end{bmatrix} & \quad B = l_1\begin{bmatrix} c_1 \\ s_1 \end{bmatrix},\ C = \begin{bmatrix} l_4 c_4 + d \\ l_4 c_4 \end{bmatrix} \\
\bullet\ if\ \cos(\theta_1 - \theta_2) = 0 & \quad \gamma = \text{atan2}(C_y - B_y, C_x - B_x) \\
\quad E = (l_1 - l_2)\begin{bmatrix} c_1 \\ s_1 \end{bmatrix} & \quad E = B + (l_2 + l_3)\begin{bmatrix} \cos(\gamma) \\ \sin(\gamma) \end{bmatrix} \\
\bullet\ if\ \cos(\theta_3 - \theta_4) = 1 & \bullet\ if\ \cos(\theta_2 - \theta_3) = 0 \\
\quad E = (l_3 + l_4)\begin{bmatrix} c_4 \\ s_4 \end{bmatrix} + \begin{bmatrix} d \\ 0 \end{bmatrix} & \quad B = l_1\begin{bmatrix} c_1 \\ s_1 \end{bmatrix},\ C = \begin{bmatrix} l_4 c_4 + d \\ l_4 c_4 \end{bmatrix} \\
\bullet\ if\ \cos(\theta_3 - \theta_4) = 0 & \quad \gamma = \text{atan2}(C_y - B_y, C_x - B_x) \\
\quad E = (l_3 - l_4)\begin{bmatrix} c_4 \\ s_4 \end{bmatrix} + \begin{bmatrix} d \\ 0 \end{bmatrix} & \quad E = B + (l_2 - l_3)\begin{bmatrix} \cos(\gamma) \\ \sin(\gamma) \end{bmatrix}
\end{array}
\tag{5-6}
$$

The following parameters were considered for the five-bar linkage case study in this thesis:

$$
\begin{aligned}
&l_1 = 1,\ \ l_2 = 1.3 \\
&l_3 = 1,\ \ l_4 = 1.6 \\
&d = 2
\end{aligned}
\tag{5-7}
$$

The undermobility singularity loci are presented in the output space (x vs. y) in Figure 17. The overmobility singularity loci in input space (θ_1 vs. θ_4) could be seen in Figure 18.

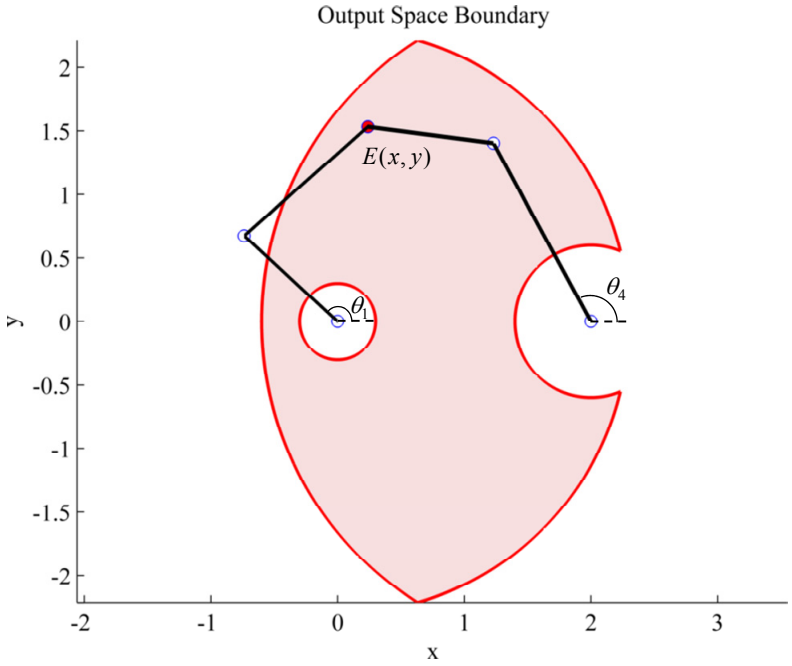


Figure 17: Workspace of a five-bar linkage

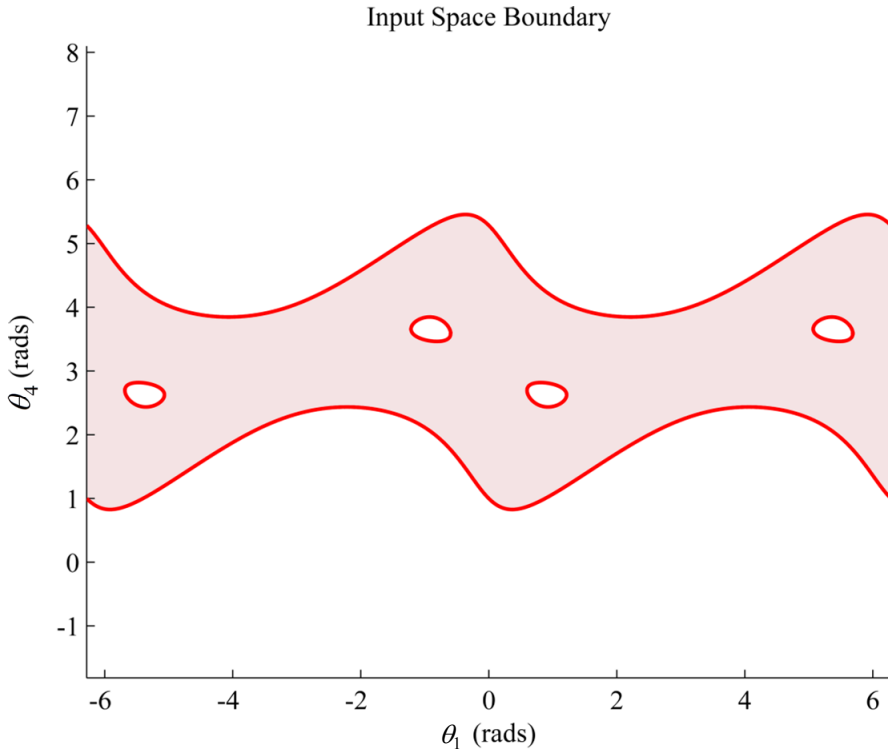
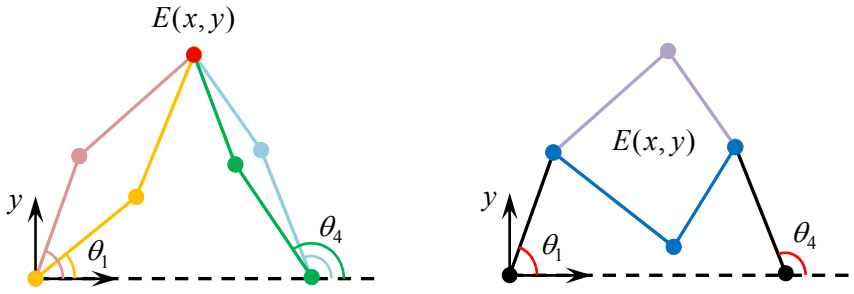


Figure 18: Input space of a five-bar linkage

Note that for a five-bar linkage there are four configurations for each output or (*x, y*) and two for each input or (θ_1, θ_4) (**Figure 19**).



(a) **Four possible configurations for a single output (x, y)**

(a) **Two possible configurations for a single input (θ_1, θ_4)**

Figure 19: Number of solutions for each input/output.

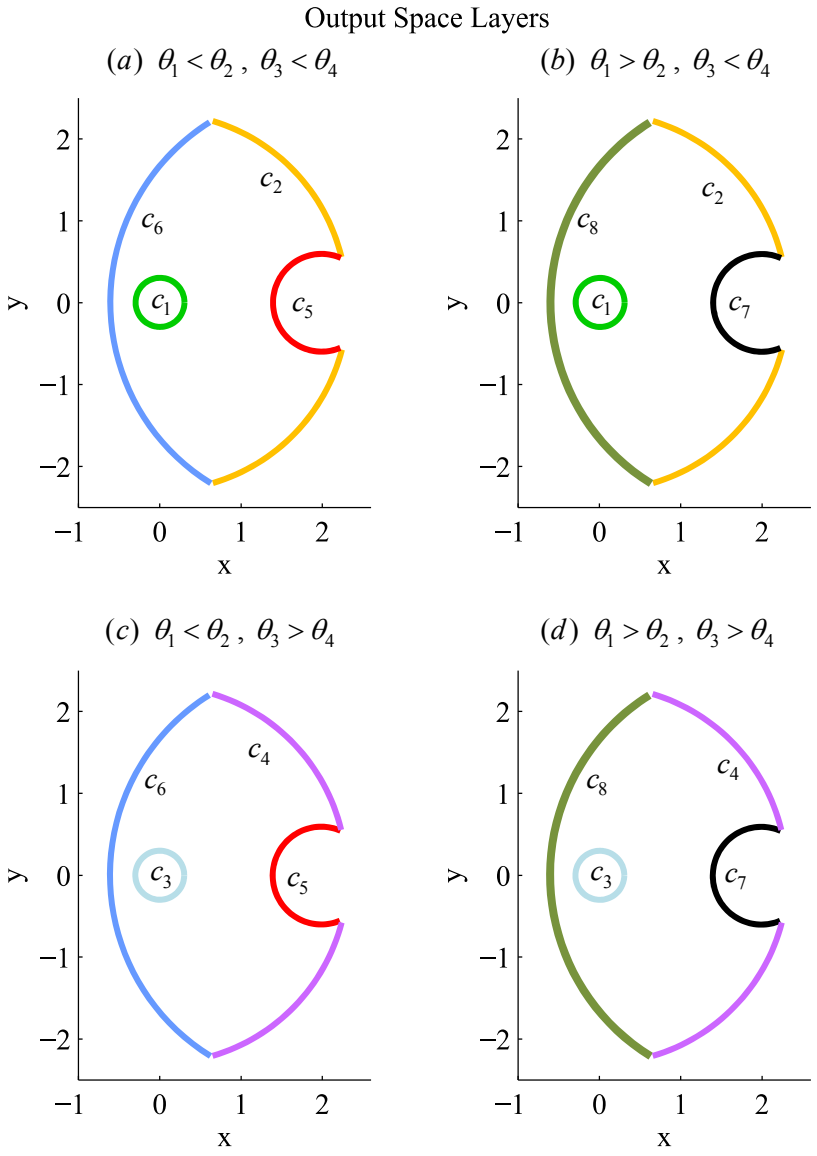


Figure 20: Connection of output space layers.

The boundaries of input and output space in this parallel manipulator are kinematic branches. For example when $\theta_1 = \theta_2$ the system could change configuration in output space. According to **Figure 19** there are four layers of output space which are connected through the workspace boundaries. The connectivity is illustrated via different colors and indices in Figure 20 (c_1 through c_8).

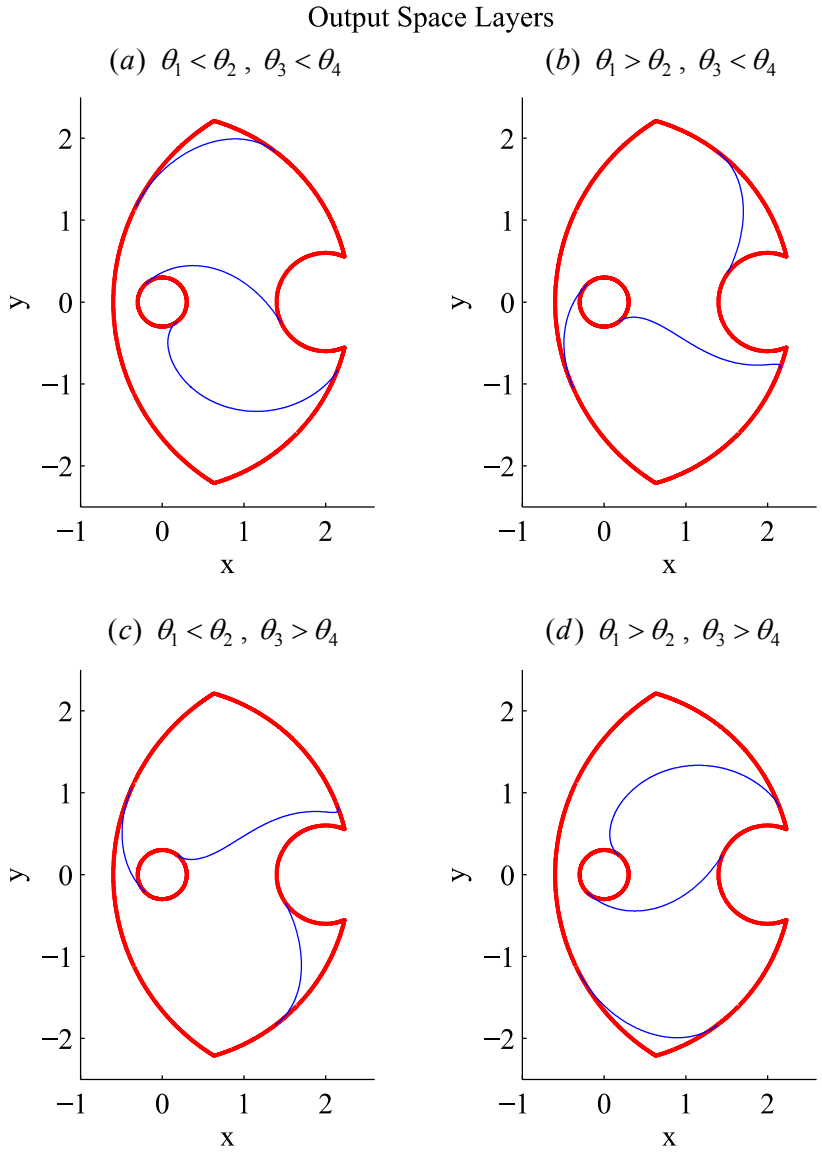


Figure 21: Undermobility singularity in the output space aspects

These layers are called *aspects* by Borrel et al. [30]. For this specific manipulator passing through boundary singularities are required to change configuration. However, for some manipulators it is possible to change configuration without passing through singularity (see Chapter **7** for more details). The undermobility singularities shown in Figure 18 could be mapped into these different output spaces as in Figure 21. Similarly the overmobility singularities could be mapped into the two layers of input space (Figure 22).

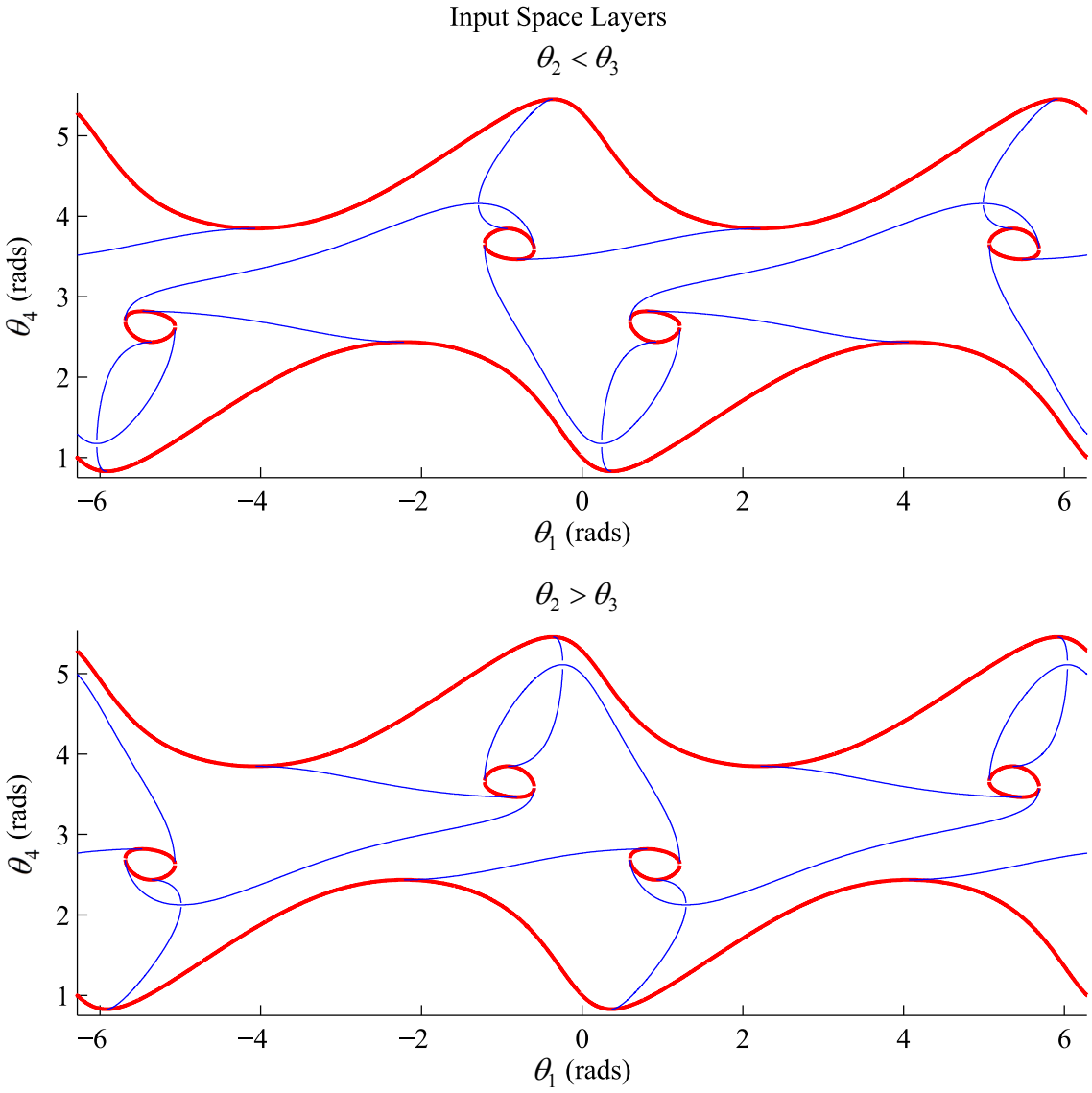


Figure 22: Overmobility singularity in the input space aspects

In Figure 22, all boundaries in two different aspects are connected to corresponding point in other aspect, and the system would change configuration whenever it passes through red lines specified in figure. Note that whenever the system passes through the singularities inside the output space (overmobility singularities) the trajectory reflects back on the boundary in input space. This has been previously pointed out by Wenger et al. [31] and it is illustrated in Figure 23.

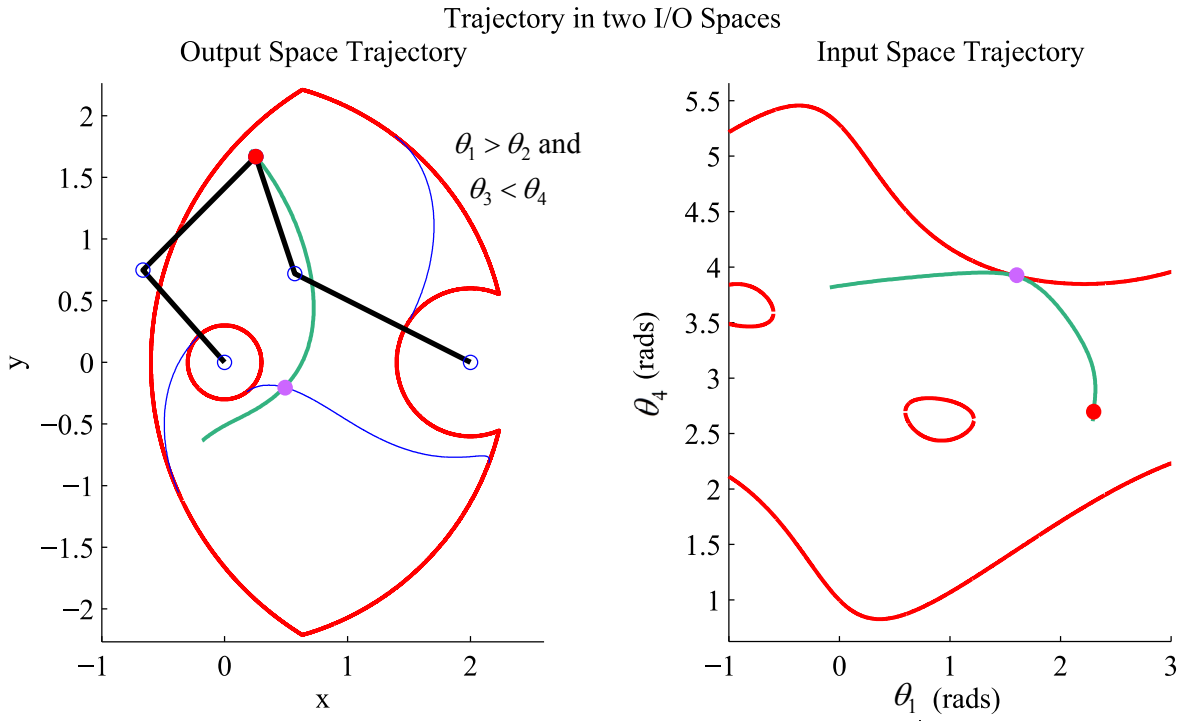


Figure 23: Trajectory reflection in the input space

The same interpretation could be made for undermobility singularities and output space boundaries. In Chapter **7**, it will be explained that for some manipulators, some undermobility (overmobility) singularities may not be boundaries or barriers in output (input) space.

6. Singularity Avoidance

In this chapter, the existing methods for detecting and handling singularities are reviewed. We focus on real-time singularity avoidance method for trajectory generation which was proposed by Marani et al. [32]. A new measure of singularity is then proposed which is an estimated distance of closest singularity point to the current pose in the workspace. The new measure is compared to Marani's method and *minimum singular value* (first suggested by Klein et al. [33]), on a five-bar mechanism as a case study. The results show that the new criterion for detection and avoiding the singular configurations is more accurate in terms of distance in the workspace than both mentioned methods.

6.1 Literature Review

As discussed in Chapter **5**, near the undermobility singularities the motion amplification factor of actuated joints to outputs decreases dramatically and as a result, the manipulability of the system will decrease. While in overmobility singularities the control over output is temporarily lost and constraint forces will increase which may cause the system to break down. For this reason it is usually desirable to avoid such *ill-conditioned* configurations. It should be noted that in some cases, due to existence of large amplification factors between the input and output motion, it may be useful for some mechanisms to operate close to singularities such as when accurate positioning of end-effector or increased sensitivity of a sensor is required [34]. Besides, Nenchev et al. [35, 36] introduce the *dynamic singularity* and suggest the possibility of passing through the overmobility singularities by taking dynamics of system and instantaneous motion of outputs into account. Similar method could be used to pass through some undermobility singularities;

however, since the undermobility singularity may be a workspace barrier, further knowledge about the type of singularity is required (see Chapter **7**).

Zein et al. [37] have divided the singularity handling methods into three main categories:

1. *Eliminating singularities in the design stage*: this method is best but may be difficult to apply, and restricts design possibilities. For the reconfigurable manipulator, the grasping point of end-effector could be considered as a design parameter, since it would affect the workspace and singularity surfaces. By studying the workspace, (see Chapter **7**) it is possible to eliminate the singularities and enhance the workspace of the mechanism.
2. *Determining the singularity-free regions of workspace*: this method involves study of the workspace which will be addressed in Chapter **7**. This method does not restrict the design, but it may be still difficult to implement. A high level real-time trajectory generator may be required to use the information of the workspace for singularity free paths.
3. *Trajectory generation of singularity-free paths in the workspace:* In this method a real-time singularity avoidance method tries to generate singularity-free paths given the initial and destination points in workspace. However these methods cannot guarantee to achieve a singularity-free path if it exists and they cannot assure that a singularity-free path does not exists.

In this chapter we focus on generation of singularity-free trajectories for singularity avoidance (third category of Zein et al. [37]). Although determining if a singularity-free trajectory exists between two points in workspace is still an open problem [7], depending on the mechanism, in practice using real-time singularity avoidance may be sufficient. For example in serial manipulators or in a limited workspace which excludes interior singularities these methods would

suffice. Besides, the real-time singularity avoidance techniques may be used in conjunction with trajectory planners to reduce their computation cost.

Near overmobility singularities, control torque of active joints is increased. Therefore, one method to avoid overmobility singularities is to limit the maximum required actuator torque. Bhattacharya et al. [26] has compared two real-time singularity avoidance techniques both designed to maintain the actuator forces within a given limit. The trajectory is modified real-time using an optimization method with respect to actuator torque limits. Dasgupta et al. [38] proposed a numerical trajectory generation between two points in workspace for a Stewart platform which requires the workspace segments information. Calculation of all workspace segments; however, is still an open problem for the Stewart 6-dimensional workspace [7].

Bertram et al. [39] proposed a numerical search method for trajectory planning in an unknown workspace using the so-called *rapidly-exploring random trees*. For redundant manipulators a redundancy resolution method is explored by Liu et al. [40]. They have proposed a trajectory generation method which takes the maximum allowable velocity and acceleration of joints into account and the timing of the trajectory is controlled based on the joint velocities. The algorithm generates a trajectory passing through some generated knots which their density is increased near singularities.

In most methods in the third category of Zein et al. [37] (singularity-free trajectories), a primary trajectory is modified until a singularity-free path is obtained, then using the trajectory tracking control the system will track the computed path.

However, some of these methods including the method of Marani et al. [32] which will be discussed in section **6.3** could be implemented for real-time control. In real-time singularity avoidance, only when the system gets close to singularities necessary corrections are made to the trajectory ahead to avoid the encountered singularity. For the real-time singularity avoidance

technique which will be discussed in the present work, it is first, necessary to introduce the measure of singularity (Section **6.2)**.

6.2 Singularity Index

As mentioned before many criterions have been introduced to quantify how close a system is to a singularity, or how *dexterous* a manipulator is in a given configuration. Dexterity or manipulability is measure of the system potential to manipulate objects in a given state. Yoshikawa [41] introduced *measure of manipulability* (*MoM*) which is proportional to the volume of *manipulability ellipsoid*. The manipulability ellipsoid is formed by the singular values of the jacobian as its principal axes. It is worth mentioning that for non-redundant manipulators this measure is equal to the determinant of the jacobian matrix. Isaacson [42] introduced *condition number* which is the ratio of minimum to maximum singular value. It could be considered as a measure of how close the ellipsoid is to a sphere. The condition number is positive and less than 1. The *minimum singular value* itself was suggested by Klein et al. [33] to be used as a measure of closeness to singularity. In this context this measure will also be shown in a case study and will be compared to the proposed method.

In many literatures (such as [7, 41]), when both translational and rotational velocities appear in the range space of the jacobian, an arbitrary weight matrix is suggested to make a balance between rotational and translational motion. In these cases measure of singularity does not have a metric unit and it is referred to as *singularity index* by Merlet [7]. However without loss of generality in the present work, weight matrix is not discussed but could be easily implemented in the presentation of workspace or the discussed measures of singularity.

For parallel redundant manipulators most measures of dexterity are based on calculation of singular values. For this reason a brief explanation of singular value decomposition (***SVD***) will be discussed here. Calculation of SVD has been first studied by Golub and Kahan [43]. Maciejewski

et al. [44] have extracted a faster method for calculation of SVD for robotics systems in an iterative task, which takes advantage of result of the SVD in the previous iteration. SVD is used in calculation of Moore-Penrose pseudo-inversion which provides a least square solution for a redundant system of equations [45].

SVD is an algorithm which decomposes any matrix into three matrices:

$$A = UDV^T \tag{6-1}$$

where A is an arbitrary real matrix, it could be square, non-square matrix or rank deficient. U and V, are square orthogonal matrices. D is a diagonal matrix with its diagonals being singular values of A, sorted from the highest to the lowest value. Singular values of A could be interpreted as how close A is to rank deficiency (or linear dependency) in directions specified by columns of U in range space of A. In other words the singular values are the underlying criteria of how close A is to all its possible singularities.

6.3 Marani's Methods

Marani et al. [32] have suggested a real-time singularity avoidance based on maintaining a minimum measure of manipulability (Yoshikawa's measure or *MoM*) near singularities. Their method is originally suggested for serial manipulators but it could also be used for parallel robots. The method is based on calculation of a gradient in workspace, along which the measure of manipulability would change the most. Then the trajectory is *reconstructed* [46] in real-time i.e. the component of path along the mentioned direction is removed in each iteration. However since the control is real-time and non-model-based, they introduce two zones with respect to *MoM*, which we call *alarm zone* and *restricted zone.* When the trajectory enters the first zone the algorithm starts to repel the trajectory from singularity with a variable gain. The gain is a function of *MoM* and the two limits of *MoM* distance and increases as system gets close to the restricted zone. The controller aims to prevent the trajectory enter the restricted zone. Figure 24 shows ideal

performance of the algorithm. It is worth mentioning that in all non-model-based real-time singularity avoidance techniques there is always a risk that the system enters the restricted zone of low dexterity, since the dynamics of the system is unknown, but with proper high-level control and tuning of control gains and with the help of simulation, a proper control scheme could be achieved.

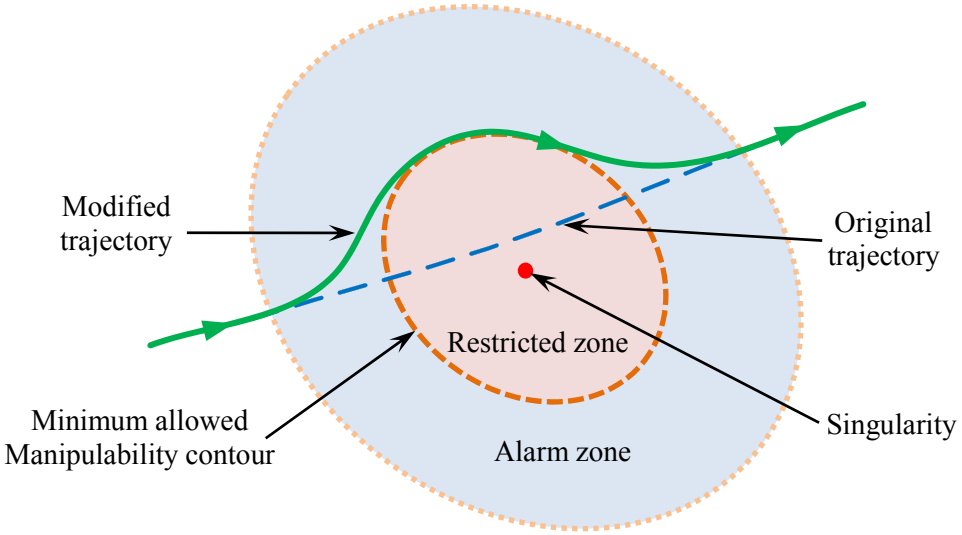


Figure 24: Marani's Singularity Avoidance

It should be also mentioned that depending on the singularity contour it is possible for the algorithm to stall at local minima especially for the complex workspace of parallel manipulators. An illustration of this phenomenon is shown in Figure 25. To avoid such singularities a high-level trajectory planer is required (such as methods that use search algorithms [39] or use prior information of workspace, see Chapter **7**).

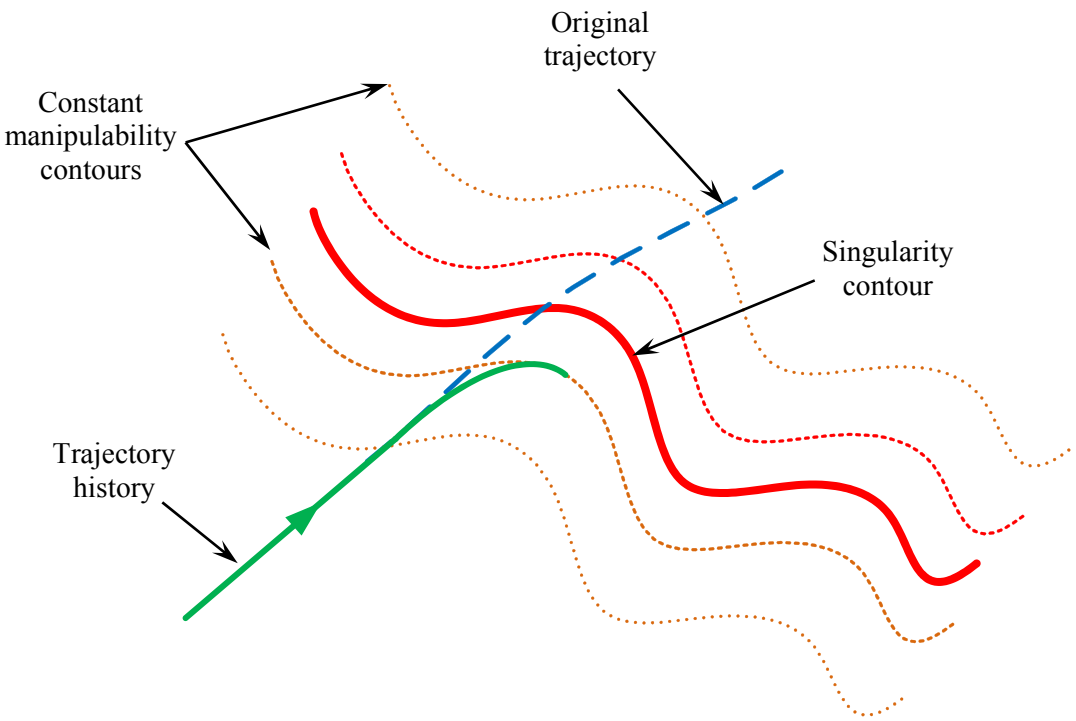


Figure 25: Failure of singularity avoidance technique at a singularity barrier

Measure of manipulability which is a scalar value, could be calculated using the following formula for a non-square (or square) jacobian [41]:

$$mom = \sqrt{\det(AA^T)} = \sqrt{\det(A^T A)} \tag{6-2}$$

For a square matrix *MoM* is equal to $\left|\det(A)\right|$. Considering $[d_1 \quad \cdots \quad d_n]^T$ to be singular values of A, *MoM* could be expressed in terms of singular values [47]:

$$mom = \prod_{i=1}^{n} d_i \tag{6-3}$$

It is possible to calculate the gradient of *MoM* with respect to generalized coordinates using the following formulation [48]:

$$\frac{\partial mom}{\partial q_k} = mom \cdot \text{trace}\left(\frac{\partial A}{\partial q_k} A^+\right)$$
$$N = \frac{\partial mom}{\partial q} \tag{6-4}$$

where N is unit vector of the gradient of manipulability (*MoM*) in state-space (q). In other words N is the normal vector to the constant manipulability contour in generalized coordinates (q). As could be seen, calculation of jacobian of A is also required to compute N. As mentioned before, an straightforward algorithm such as [32] could be implemented to use this normal vector and *MoM* as the singularity closeness measure to avoid the singularities. The implementation of such method will not be discussed here. We will now focus on parallel manipulators and provide examples on performance of this method.

As mentioned in Chapter **5**, in the present work we limit our study to *IO-state manipulators*. As a reminder, these parallel manipulators could be formulated using the constraint equation $\Phi(q_i, q_o) = 0$ where q_i and q_o are input and output states. Thus the time derivative of constraint equation could be written as $A_i\dot{q}_i + A_o\dot{q}_o = 0$ where A_i and A_o are input and output jacobians. Rank deficiencies of A_i will result in undermobility singularities while rank deficiencies of A_o will cause overmobility singularities.

Both types of undermobility and overmobility singularities could be mapped in both input and output space, as discussed in Chapter **5**. The term "measure of manipulability" will not be used as a measure of rank deficiency of A_i or A_o since the name comes from the jacobian of serial manipulators. Instead the term *measure of singularity* (*MoS*) will be used but with the same formulations as in equations (6-2), (6-3) and (6-4).

Consider *MoS* of A_x to be $\text{mos}(A_x)$ (A_x being A_i or A_o). Based on formulation in (6-4), we will introduce four possible normal vectors for singularity avoidance applications:

$$q = \begin{bmatrix} q_i \\ q_o \end{bmatrix}, \quad \frac{\partial q}{\partial q_i} = \begin{bmatrix} I_{i\times i} \\ A_o{}^{+} A_i \end{bmatrix}, \quad \frac{\partial q}{\partial q_o} = \begin{bmatrix} A_i{}^{+} A_o \\ I_{o\times o} \end{bmatrix} \tag{6-5}$$

$$
{}^{q}G_{mx} = \frac{\partial \text{mos}(A_x)}{\partial q}
$$

$$
{}^{i}G_{mx} = \frac{\partial \text{mos}(A_x)}{\partial q_i} = {}^{q}G_{mx}\,\frac{\partial q}{\partial q_i}
$$

$$
{}^{o}G_{mx} = \frac{\partial \text{mos}(A_x)}{\partial q_o} = {}^{q}G_{mx}\,\frac{\partial q}{\partial q_o}
$$

where ${}^{q}G_{mx}$, ${}^{i}G_{mx}$ and ${}^{o}G_{mx}$ are the gradient of $\text{mos}(A_x)$ in the generalized coordinates (q), input (q_i) and output (q_o) spaces respectively. For example to avoid undermobility singularities, a potential algorithm would be to monitor $\text{mos}(A_i)$ constantly and whenever it is reached to a minimum limit, the control force could be adjusted using its gradient; ${}^{i}G_{mx}$ or ${}^{o}G_{mx}$ in order to avoid singularity.

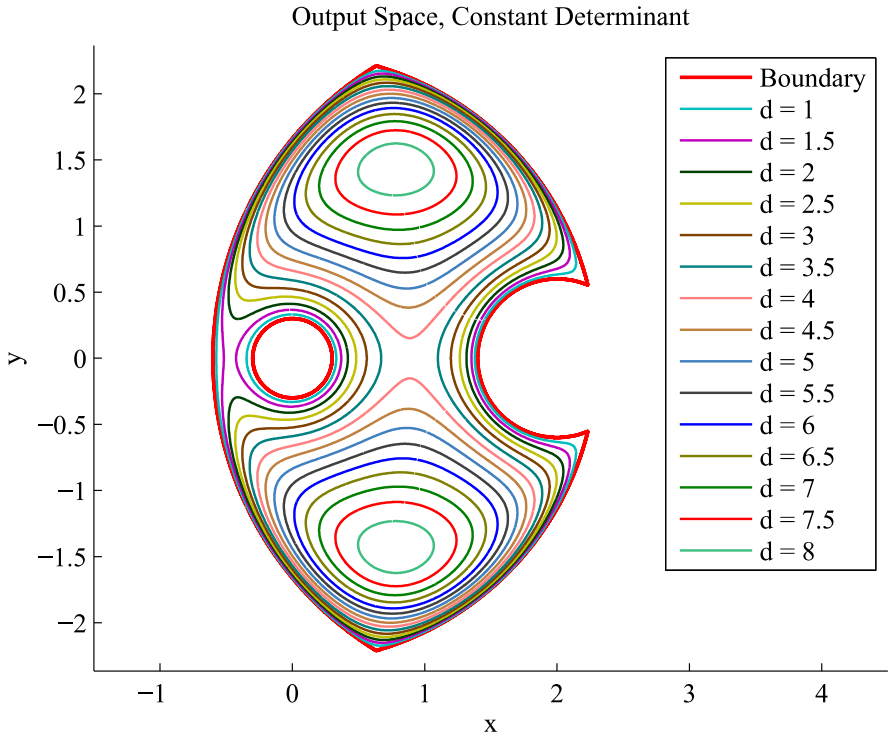


Figure 26: A_i measure of singularity contours

The constant $\text{mos}(A_i)$ contours in output space are plotted for the five-bar linkage that was studied earlier in Section **5.2**, in Figure 26. Note that for the studied non-redundant five-bar manipulator, $\text{mos}(A_i)$ is equal to $\det(A_i)$.

As could be seen in the Figure 26, the density of contours is not uniform in the sense of distance from workspace boundaries. Based on this intuition, in the next section a new method is proposed and will be compared to the discussed method.

6.4 Singularity Estimator

An ideal measure for closeness to singularity could be the distance of the current system pose to its nearest singular configuration [7]. To the best of our knowledge and according to [7] (p.204) no such method exist. In this section we proposed a new measure based on estimating the closest singular point either in input or output space. The estimation is based on first order derivative of minimum singular value with respect to generalized coordinates. Higher orders may be used but due to high order of computation, it is not recommended here.

According to Voglewede et al. [49] a *singularity index* (see Section **6.2**) should satisfy the following three criteria:

1. Singularity index of A or $S(A)$ should be zero if and only if A is singular.
2. If A is not singular then $S(A) > 0$
3. $S(A)$ should have a clear physical meaning

Voglewede et al. [49] introduced several singularity indices, both in the velocity or force domain. However in the present work the aim is to define a measure as a distance of the current pose to the nearest singularity either in the output or input space.

As a motivation for the suggestion of proposed singularity estimator, let's first consider one of the constant $mos(A_i)$ contours in Marani's method (from Figure 26), which is shown in Figure 27.

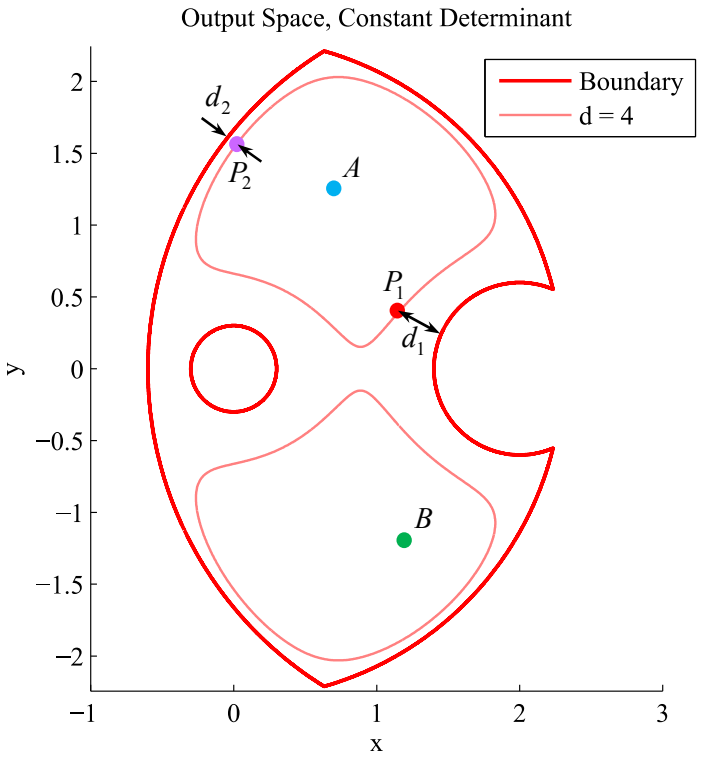


Figure 27: A_i measure of singularity contour

As we can see in Figure 27, the two points on the contour P_1 and P_2 have very different distances (d_1 and d_2) from the nearest singularity point. On the other hand, a successful singularity avoidance algorithm based on the shown contour ($mos(A_i) = 4$), would not allow a trajectory between point A and B, while a very close point to singularity such as P_1 is allowed to be reached. In the new method we try to estimate the closest singularity point and measure its direction and distance from the current pose.

For future calculations it is necessary to review the computation of jacobian of singular values which has been studied by Papadopoulo et al. [50]. Consider singular value decomposition of A in equation (6-1). Let a_{mn} , u_{mr} , v_{nr} be the (m,n) element of the matrix A, (m,r) element of U and

(n,r) element of V, respectively. Then the derivative of r-th singular value or d_r can be calculated from the following formula [50]:

$$\frac{\partial d_r}{\partial a_{mn}} = u_{mr}\, v_{nr} \tag{6-6}$$

By knowing the value of $\frac{\partial a_{mn}}{\partial q}$ for every (m,n), using the chain rule the jacobian of all singular values with respect to system generalized coordinates (q) could be calculated:

$$\frac{\partial d_r}{\partial q} = \sum_{m,n} \frac{\partial d_r}{\partial a_{mn}} \frac{\partial a_{mn}}{\partial q} \tag{6-7}$$

Consider the r-th singular value of A_x to be ${}^{x}d_r$, where A_x is either input jacobian (A_i) or output jacobian (A_o). Using the chain rule, the jacobian of ${}^{x}d_r$ with respect to input and output coordinates could be calculated:

$$q = \begin{bmatrix} q_i \\ q_o \end{bmatrix}, \quad \frac{\partial q}{\partial q_i} = \begin{bmatrix} I_{i\times i} \\ A_o^{+} A_i \end{bmatrix}, \quad \frac{\partial q}{\partial q_o} = \begin{bmatrix} A_i^{+} A_o \\ I_{o\times o} \end{bmatrix}$$

$$\begin{aligned} {}^{q}G_{d_r x} &= \frac{\partial {}^{x}d_r}{\partial q} \\ {}^{i}G_{d_r x} &= \frac{\partial {}^{x}d_r}{\partial q_i} = {}^{q}G_{d_r x} \frac{\partial q}{\partial q_i} \\ {}^{o}G_{d_r x} &= \frac{\partial {}^{x}d_r}{\partial q_o} = {}^{q}G_{d_r x} \frac{\partial q}{\partial q_o} \end{aligned} \tag{6-8}$$

where ${}^{q}G_{d_r x}$, ${}^{i}G_{d_r x}$ and ${}^{o}G_{d_r x}$ are the gradient of ${}^{x}d_r$ in the generalized coordinates (q), input (q_i) and output (q_o) spaces respectively. One possible singularity index could be selected as the minimum singular value itself [33]. Consider ${}^{x}d_m$ to be the minimum singular value of A_x. An algorithm could be implemented with the minimum singular value of A_x. Then using its gradient

with respect to inputs or outputs, a limit of minimum singular value may be maintained in real-time.

Similar to the approach in Section **6.3**, the contours of ${}^{i}d_{m}$ in output space is plotted for the five-bar linkage case from Section **5.2**, in Figure 28.

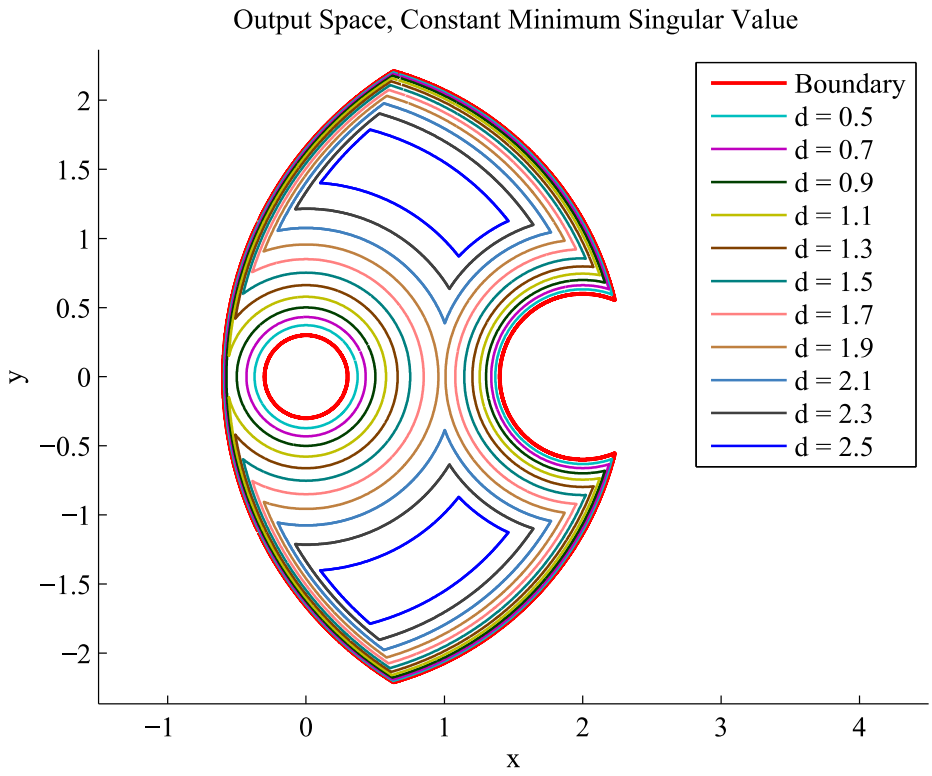


Figure 28: Minimum singular value contours in output space

One of the conclusions that could be drawn from Figure 28 is that unlike the measure of singularity (*MoS*), when using the minimum singular value only the nearest singularity is detected. But the singularity index of *MoS* (see Section **6.3**) is the product of all singular values.

Since the rank of A_i in the five-bar linkage case study is two, two of the closes singularity pints for A_i may be detected which belong to two different singularity loci (Figure 29**)**. However for avoidance only the closest one may be considered.

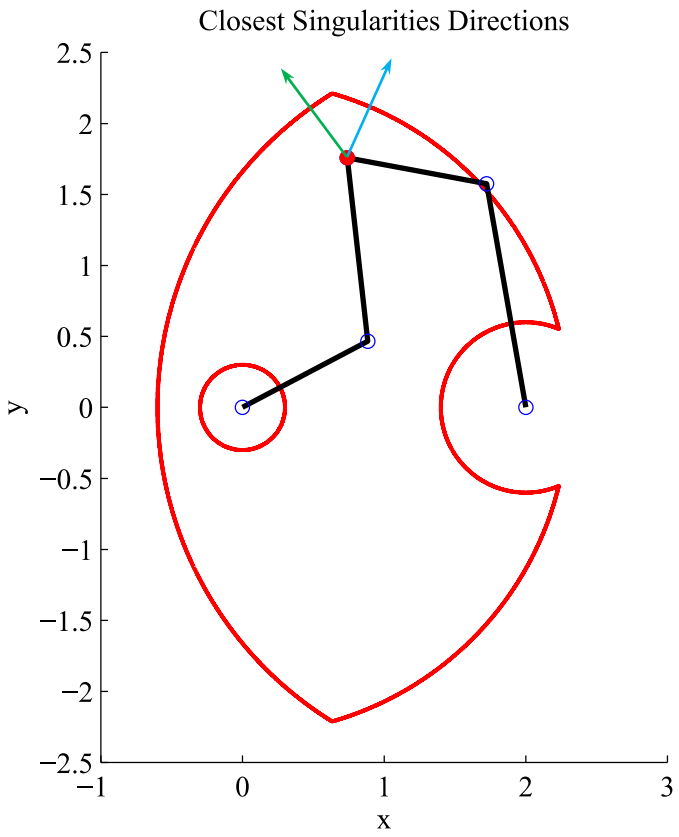


Figure 29: Directions of two (=rank(A_i)) closest undermobility singularities

As a review, using formulations in (6-6), (6-7) and (6-8) the jacobian of all the singular values of A_x with respect to q_i or q_o could be calculated i.e. ${}^{S}G_{d_r x}$, $\forall r,\ x \in \{i,o\},\ S \in \{i,o\}$ where S is the variable space that the gradient is calculated with respect to. Considering the gradient ${}^{S}G_{d_r x}$ to be constant from the current pose to the nearest singularity point, the following approximation could be made:

$$\begin{aligned} {}^{x}d_r &\cong {}^{i}G_{d_r x}\ {}^{x}\delta q_i \\ {}^{x}d_r &\cong {}^{o}G_{d_r x}\ {}^{x}\delta q_o \end{aligned} \qquad (6\text{-}9)$$

Where ${}^{x}\delta q_i$ (respectively ${}^{x}\delta q_o$) is the vector from current position to nearest approximate point in input (output) space where the r-th singular value of A_x would become 0. From (6-9) ${}^{x}\delta q_i$ and ${}^{x}\delta q_o$ could be solved as:

$$
\begin{aligned}
{}^{x}\delta q_i &\cong ({}^{i}G_{d_r x})^{-1}_{left}\ {}^{x}d_r \\
{}^{x}\delta q_o &\cong ({}^{o}G_{d_r x})^{-1}_{left}\ {}^{x}d_r
\end{aligned}
\tag{6-10}
$$

where $X^{-1}_{left} = (X^T X)^{-1} X^T$ is the left inverse of B. Whenever the systems is close enough to singularities the proposed approximation method would provide rather accurate distance of the singularities. Using this method, one singularity point is estimated for each singular value, however for control purposes only the smallest distance ($\min_r({}^{x}\delta q_i)$ or $\min_r({}^{x}\delta q_o)$) may be used for singularity avoidance scheme. Thus a candidate algorithm for avoiding singularities could be based on singularity index of $\min_r({}^{x}\delta q_i)$ or $\min_r({}^{x}\delta q_o)$ and its gradient ${}^{i}G_{d_r x}$ or ${}^{o}G_{d_r x}$ respectively. The contours of $\min_r({}^{x}\delta q_i)$ in output space is plotted for the five-bar linkage case from Section **5.2**, in Figure 30.

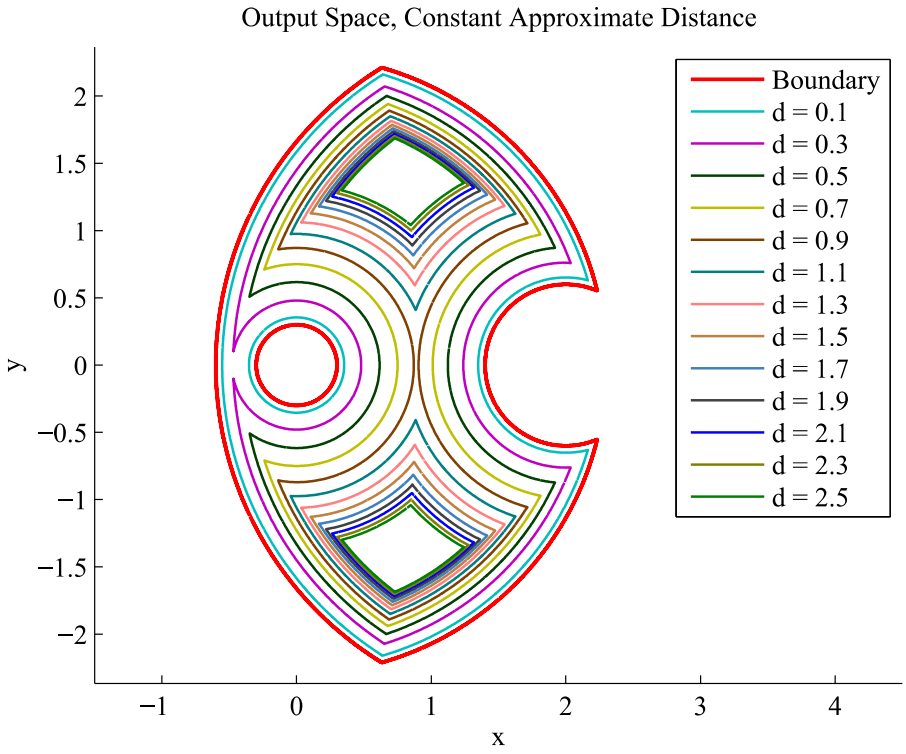


Figure 30: Approximate singularity distance contours in output space

It could be seen that the proposed measure of singularity is very uniform especially at the closed points near singularity. This accuracy will be decreased as the distance to singularity is increased. This is due to the fact that the method is based on assuming the gradient of singular values to be constant.

As final comparison the results of the three discussed measure of singularities contours could be seen in Figure 31. As could be seen the uniformity of the proposed method near singularities is more than the two other methods. Such measure provides good estimation of constant distance from singularities and may be more desirable from control point of view. It should be noted that the difference in computational cost of Marani's and proposed method is negligible. As mentioned before, to enhance the accuracy of estimation, it is possible to use higher order derivatives, but the acquired accuracy may not worth the high computational cost.

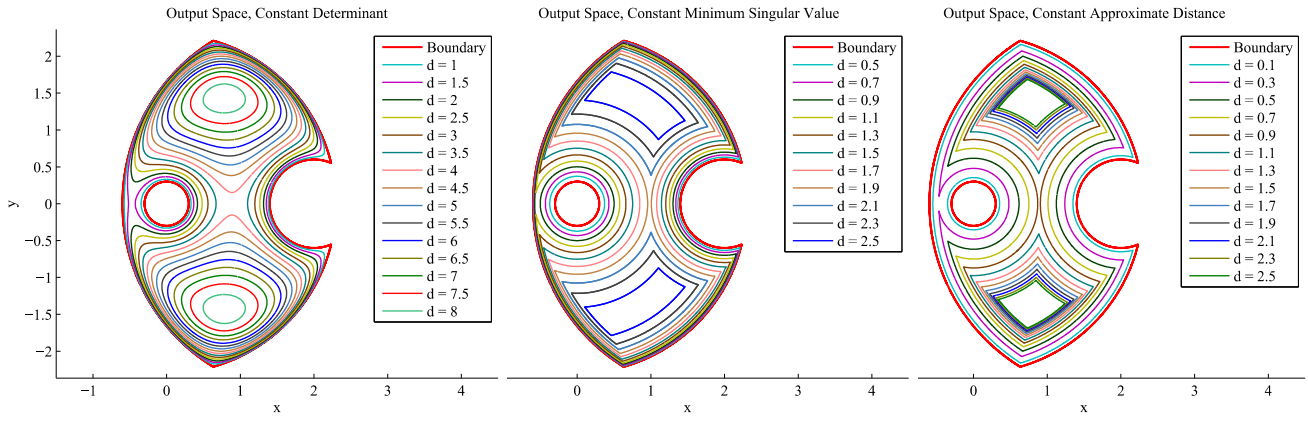


Figure 31: The comparison of the three singularity measure contours

7. Workspace Determination

In this chapter, first, a review of the works done on manipulator workspace analysis is presented. Then, a numerical method based on *swept volumes* is proposed and the results of the approach are illustrated for several cases. Finally, its efficiency and accuracy is discussed in comparison to some of the relevant works in the literature.

7.1 Literature Review

A parallel manipulator workspace may be restricted by many factors including [7]:

- Undermobility singularity that may split the workspace into separate components
- Mechanical limits on system variables
- Self-collision between the links of the robot

In this chapter we limit our study to the first two factors; mechanical limits and singularities that could limit the joints workspace.

There has been extensive study of workspace in the literature which can be categorized into the following groups:

- **Analytical methods**: Analytical methods such as works done in [51-55] which has a high computation time and sometimes only simplified models may be used and some methods are only applicable to square jacobians which would limit the usage to non-redundant manipulators. Besides it does not usually provide any geometrical insights on the singular configurations [22].

- **Geometrical approaches**: Such as works in [56-59], Merlet [7] has introduces a geometrical approach based on Grassmann line-geometry to identify the singular configuration.
- **Numerical methods**: There are two approaches in numerical methods:
    1. Numeric solution of constraint equation which includes works done by Bohigas et al. [60] and Porta et al. [61].
    2. Continuation-based methods, in which a search trajectory tries to trace boundary surfaces of a robot workspace, using the constraint formulation, such as the work done by Haug et al. [62] and Snyman et al. [63] and Kumar et al. [64].
    3. Workspace sweep methods: includes the *swept volumes* which will be discussed later. The method used in this chapter also lies in this category.

Due to momentary dependency of outputs (respectively inputs), only at undermobility (overmobility) singularities the configuration may be at a barrier singularity in output (input) space. For this reason, from now on the term singularity is only referred to singularities of corresponding jacobian matrix.

As seen in Section **5.2**, some manipulators may have several kinematic solutions for a given pose in workspace, Borrel et al. [30] first used the term *aspect* of workspace for these layers of solution. Wenger [65] defines *uniqueness domains* as the largest parts of workspace that are singularity free.

For non-redundant manipulators using the determinant sign of the square jacobian matrix, it is possible to distinguish some aspects of the workspace. For example for the five-bar linkage case study in Section **5.2**, the determinant of the A_i jacobian, is negative in two aspects (a) and (c) in Figure 21 and positive in two others (or positive in (a) and (c) and negative in two others, depending on the formulation of A_i). However for redundant parallel manipulators, no

interpretation about the regions of the workspace can be acquired using the jacobian matrices (A_i and A_o) since determinant is not defined for non-square matrices.

For representing the workspace of robots with more than three degrees of freedom one approach is to constrain some of the degrees of freedom of robot and plot the workspace over the two or three remaining degrees of freedom in two or three dimensional space respectively. Examples include *constant orientation workspace*, *constant position workspace*. Fixing one or a few active or passive joints at specific positions, allows the representation of the workspace with respect to remaining joints. The former would be a cross section of the complete workspace at some fixed joints positions.

Oblak et al. [66] have categorized different types of jacobian degeneracies or singularities into two categories: c*rossable* or *non-crossable* singularities, where non-crossable singularities are then categorized into *boundary* and *limit* singularities. Bohigas et al. [67] later used another notation but same concept for categorization: *barrier* and *non-barrier* (*traversable*) singularities, where traversable singularities are categorized into *boundary* or *interior* singularities.

See Figure 32 for a visualization of the mentioned categories.

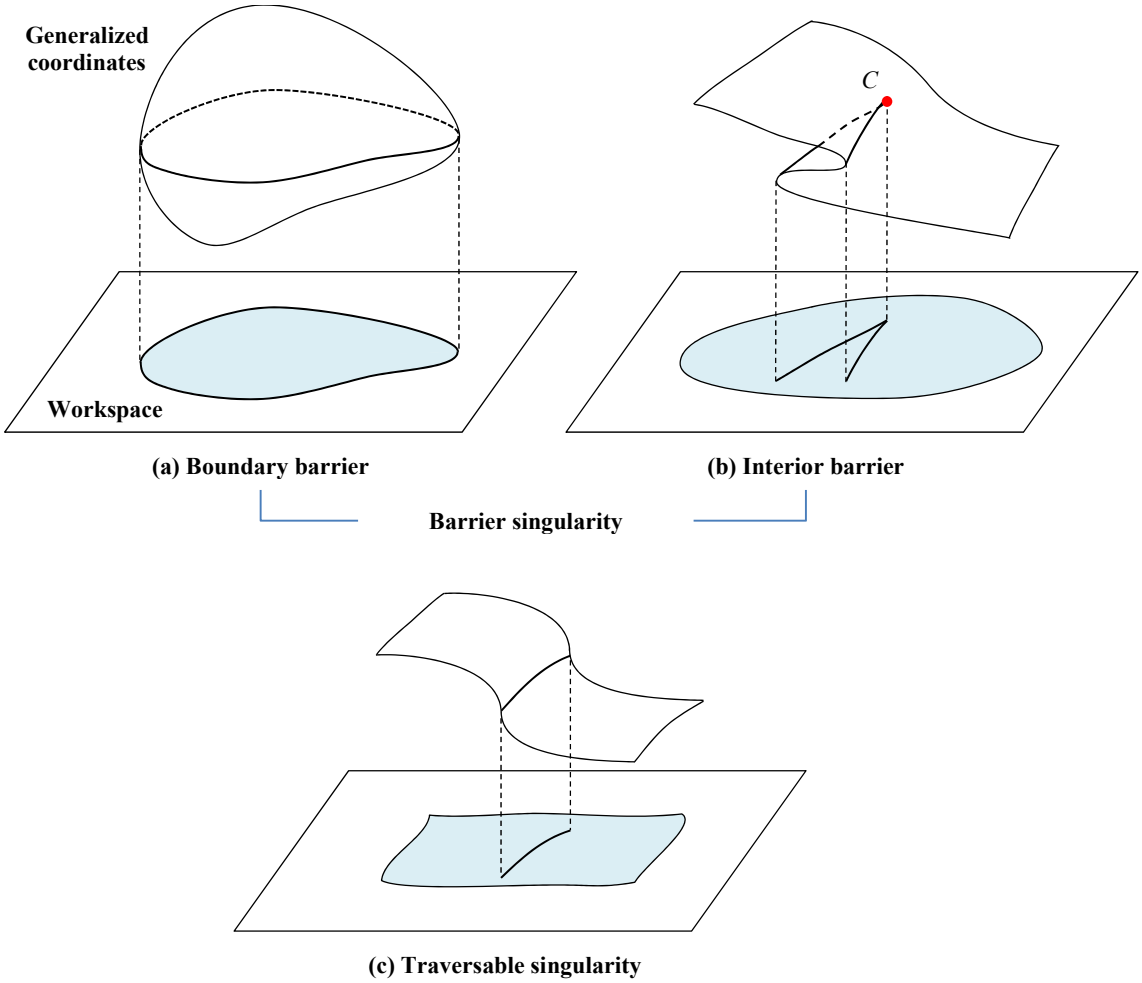


Figure 32: Singularities in workspace

As could be seen in the Figure 32, the workspace may be very complex especially for parallel manipulators; therefore a thorough prior knowledge of the workspace may be required for a successful trajectory generation.

Merlet et al. [68] has elaborately studied the workspaces of planar parallel manipulators. Decoupled parallel manipulators have been proposed to deal with the complexity of workspace determination [69, 70]. Agrawal [71] and Bonev et al. [72] have used *screw systems theory* to identify the workspace boundaries.

Bohigas et al. [67] have recently presented a complete method to identify the boundaries of a manipulator. They have later [60] implemented their method for general manipulator and have reported several case studies. The key benefit of this method is that it does not require any prior knowledge about the workspace and it guaranties to find all the boundaries. Their method is based on linear relaxation method which was previously outlined by Porta et al. [61]. In this method, first a set of constraint equations of the singularities containing generalized coordinates is transformed into canonical form. Then using the linear relaxation method, all the solution of the equations is isolated in boxes and the boxes are shrunk and split until the desired resolution of boundary is acquired. However this method has high computation cost for accurate information on the workspace boundaries and is not applicable to workspaces with dimensions greater than three [67].

Snyman et al. [63] have elaborately reviewed the methods for boundary identification, and has proposed a numerical method based on emitting several rays from a point in the workspace and finding the intersection of rays with the boundary by solving an optimization problem. The intersection points will be connected to each other to form the boundary. Dash et al. [73] have similar method for detection of workspace for parallel manipulators by emitting several radii from different points in the workspace.

Haug et al. [62] have suggested a method based on continuation method and following the boundary. Abdel-Malek et al. [52] have used an analytic method for deriving all singularities of a general 3-DoF manipulator. Abdel-Malek et al. [51] have compared their analytic method [52] with the numerical method presented in [62].

Another complexity of workspace is that it is possible in some cases for manipulator to change its assembly mode without crossing any singularities. This idea had been studied by Wenger et al. [74] for serial manipulators which are called cuspidal manipulators. A more general study of

cuspidal manipulators could be found in [75]. Changing configuration without passing singularity is due to existence of some *cusp points* in the workspace which are the intersection of two or more kinematic solutions. For example point *C* in Figure 32 shows a cusp point. Innocenti et al. [76] have first reported existence of cusp points for parallel manipulators. Later more studies have been conducted on such configuration change in parallel manipulators [37, 77-79]. Descriptive studies could be found in the works of McAree et al. [80] and Husty et al. [81].

As mentioned before a group of methods trace the boundary using numerical techniques. One of these methods is *swept volume.* In this method a swept volume (SV) is generated by sweeping an object over a smooth trajectory in a space. Wellman's work [82] was the first work related to swept volumes. Later the field started growing by development of computers and fast computational methods. Applications of swept volume include geometric modeling and workspace analysis of robots [83]. It has been used for path planning [84], workspace analysis [85], collision detection [86, 87] and manipulability study [88]. Swept volume formulation provides a practical solution for both visualization and analysis of workspace and boundaries. Elaborate study on applications of swept volumes in robotics can be found in [83]. The proposed method in this work could be considered as a swept volume method which will be explained in the next section.

Another similar numerical method is based on checking if a group of random points or gridded points satisfy the constraint equation which has been used in [31, 39, 89, 90]. If a point is consistent with the constraint equation then the corresponding generalized coordinates will be added to the workspace. But testing the consistency of a specific point in the workspace is not an easy task, since no knowledge of an initial starting point is available and also for some points there may exist several generalized coordinates solutions and as a result the efficiency of such algorithms are usually lower than the swept volume methods.

7.2 Random Sweep of Workspace

In this section an easy-to-implement method named Random Sweep of Workspace (*RSW*) will be presented here, which is used for determination of workspace. The method is especially fast and efficient over sweeping two dimensional workspaces. This method provides information both on boundary and interior points of the workspace. Here, the method is explained for parallel manipulators, but a similar algorithm may also be used for serial mechanisms.

The proposed method is based on a *simplified semi-static simulation* of a manipulator. From an initial random point inside the workspace, a straight trajectory will be followed towards an initial random direction in the generalized coordinates. During the simulations the generalized coordinates are constantly corrected in order to respect the constraint equation. After acquiring enough number of points, the generated trajectory is broken down into smaller but singularity-free trajectories. Note that any singularity-free trajectory would lie in a single aspect of workspace. All the intersecting trajectories are grouped together as one aspect and each group of points will be visualized using triangulation algorithms. Mechanical limits such as joints limits may be implemented in the algorithm.

The algorithm steps are explained below:

1. An initial point (q) inside the workspace should be provided to the algorithm. A random direction is selected.
2. The generalized coordinates (q) will be corrected using the method explained in Section **3.3** (Figure 7), and it is stored in the trajectory history.
3. A vector of generalized coordinates (δq) is calculated based on most recent direction of movement.
4. If q violates the joints limits, the respective component of δq will be mirrored.
5. δq is mapped into the null-space of A; the jacobian of general constraint equation.

6. Maximum norm of δq is limited with respect to the minimum singular value of A in order to avoid large steps especially near singularities which could result in failure of coordinate correction in step 2.
7. q of next iteration is calculated using δq.
8. Repeat step 2 through step 7 until a desired number of points is acquired.
9. The generated trajectory will be broken down into smaller singularity-free trajectories named T_B.
10. Trajectories in T_B are grouped into clusters (G) based on their intersections. Since the lines are discrete, the *k-Nearest Neighbours* algorithm [91] is used to indicate if two lines have intersection given a minimum distance limit.
11. Each cluster G contains points of a single aspect of the workspace. Using *Delaunay triangulation* [92], in 3D, patches are formed to visualize the aspects.

However note that the general constraint equation should contain all generalized coordinates (q) as opposed to a reduced-constraint equation set. Note that even if the provided initial point is not inside the workspace, the correction algorithm may be able to move the point into the workspace, but it is not guaranteed. To increase the speed of simulation algorithm (steps 2 through 7) parallel computing could be used to run several independent simulations with different starting points or directions in parallel.

Since the result is expressed in one or more cluster(s) of generalized coordinate points, it is possible to plot the workspace with respect to several parameters. For example it could be with respect to minimum singular value of jacobian or with respect to any joint variable.

In the following, three case studies will be studied using RSW. All computations are performed using parallel computing on a Quadcore i7 2.4GHz CPU in a Matlab program.

7.2.1 Case Study I: Five-bar Linkage

The five-bar linkage case study introduced in **5.2** is considered here. A trajectory containing around 200,000 points were gathered for this manipulator and according to minimum singular value of A_i the trajectories are broken into about 2500 lines, both steps in less than 20 seconds. Then in less than 2 seconds, the points are grouped into four clusters in output space, which is expected according to studies shown in Section **5.2**.

One of the main advantages of this method is providing information about the interior regions of the workspace. For example the one of the aspects of workspace is plotted with respect to four link angles in Figure 33, using the acquired data in RSW method.

Using the computed generalized coordinates inside the workspace, further information such as different singularity indices for each point could be acquired. The first aspect of workspace has been plotted according to product of singular values (*MoS*) and minimum singular value in Figure 34, which are consistent with the results in Chapter **6**. Two-dimensional workspace is also plotted over two other variables (θ_4 and a singularity index) in 3-D in Figure 35.

When using fewer numbers of points (such as 100,000 points in this case) the density in some regions of workspace may be less and some un-swept holes with sizes considerable to the real holes in the workspace may appear (Figure 36). Using the knowledge of minimum singular value they may be distinguished from outside the boundary which is not discussed in this context.

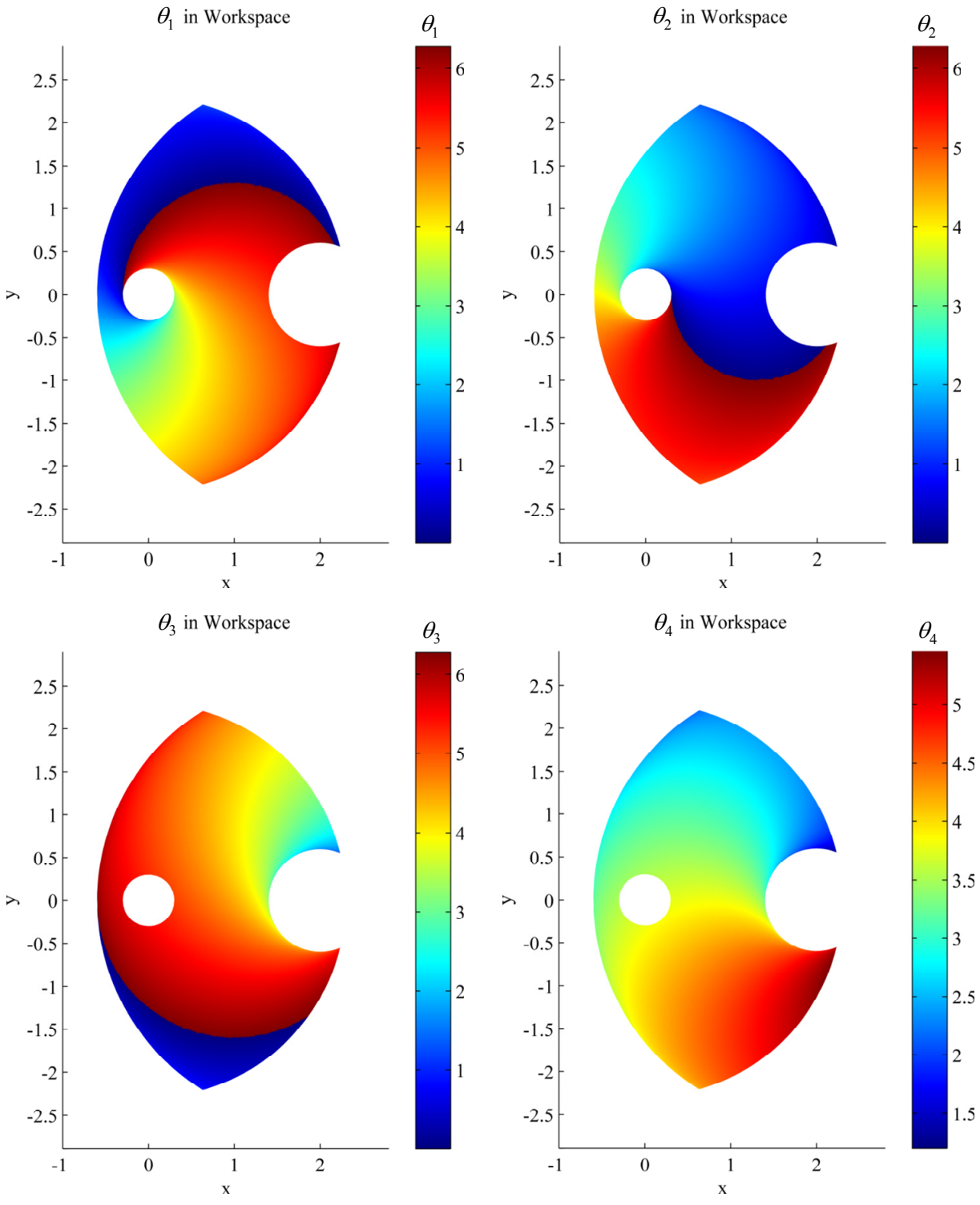


Figure 33: Link angles throughout workspace

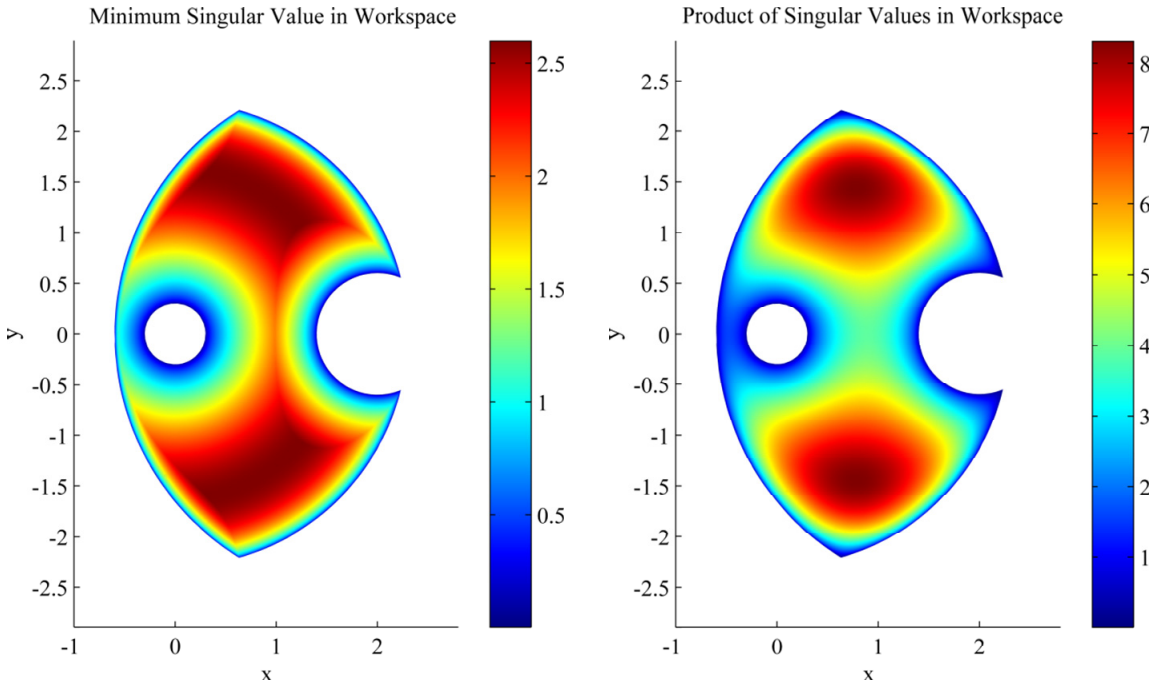


Figure 34: Two singularity measures throughout workspace

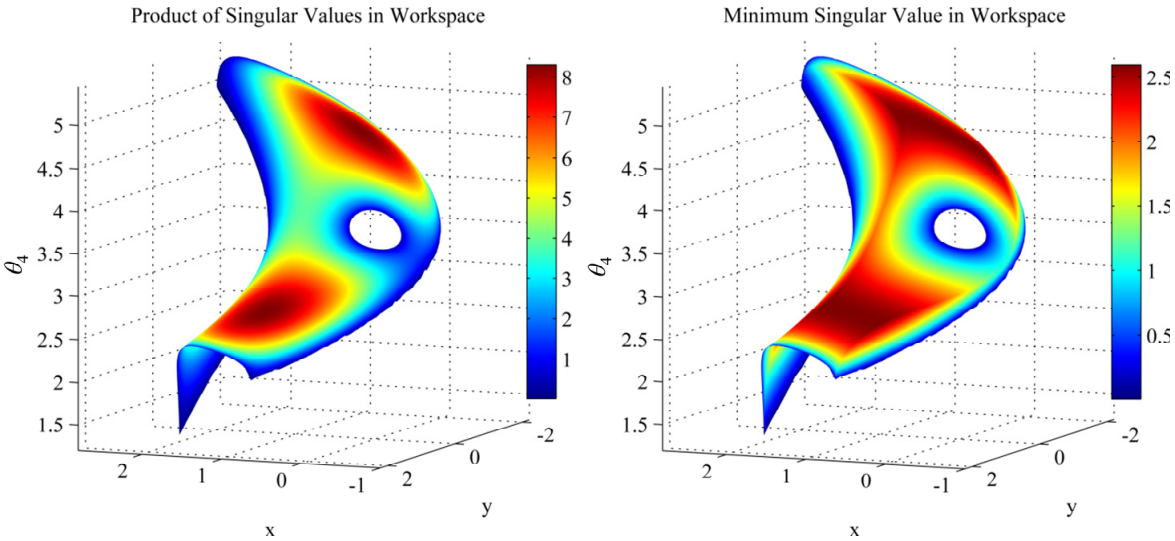


Figure 35: Workspace over two variables

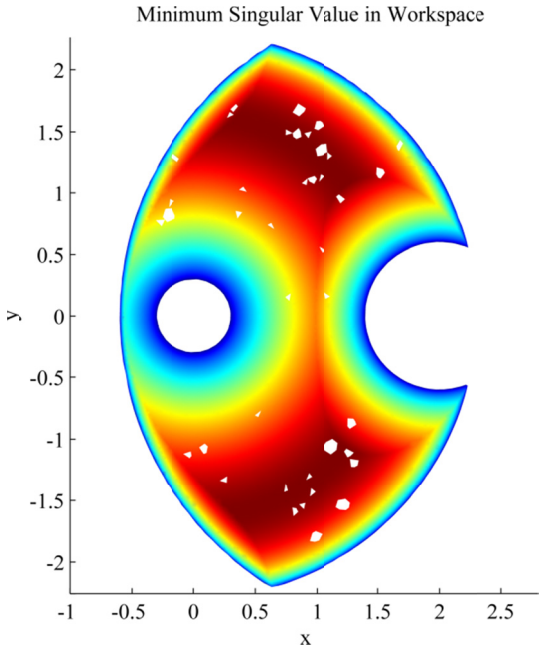


Figure 36: Existence of un-swept regions

7.2.2 Case Study II: 3DoF Reconfigurable Manipulator

A 3-DoF reconfigurable manipulator with a full position constraint on the end-effector and with one released cylindrical link is considered. The manipulator is shown in Figure 37.

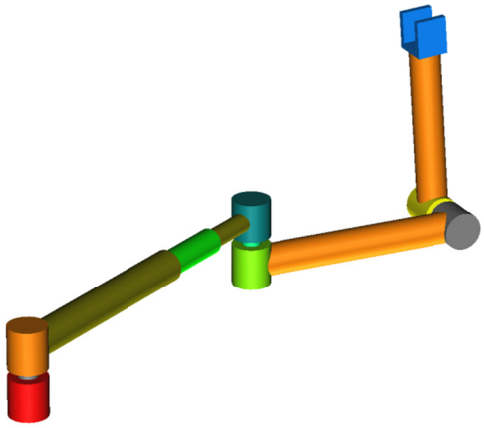


Figure 37: 3-DoF reconfigurable manipulator

A trajectory containing around 500,000 points were gathered for this manipulator and according to minimum singular value of A_i the trajectories have been broken into about 5000 lines, all in

less than 150 seconds. It has been automatically grouped into two clusters (aspects) in output space, in less than 5 seconds. Two different aspects are shown with respect to minimum singular value in Figure 39. As could be seen from the figures, workspace is in not symmetric along any point or line since the manipulator is not symmetric due to non-zero joint heights. Figure 38 shows half of the whole workspace which is plotted with respect to first joint variable and minimum singular value of jacobian.

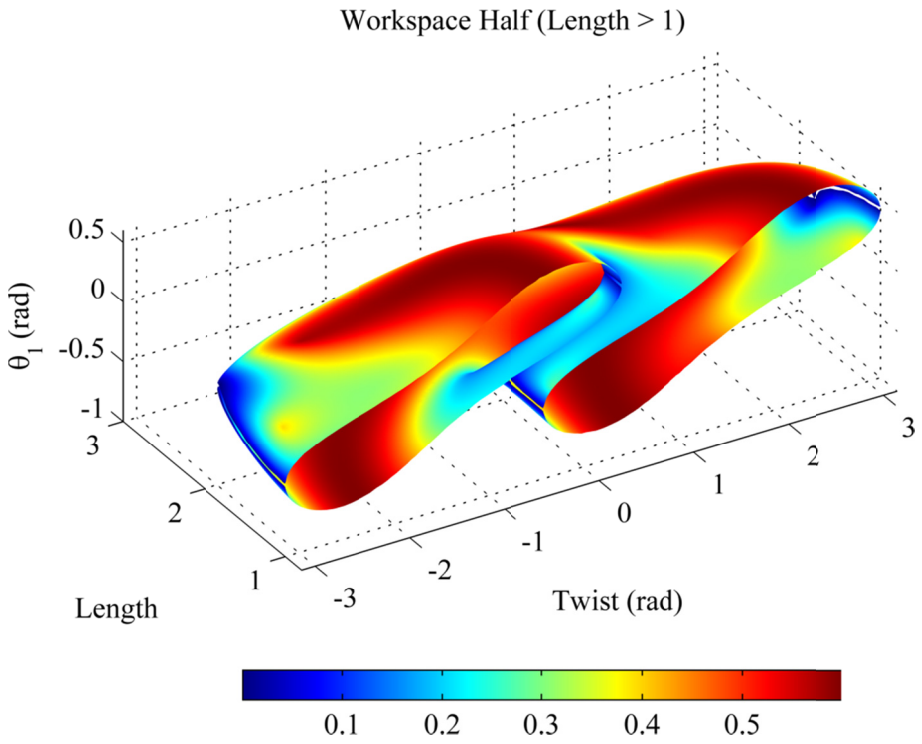


Figure 38: Both aspects of 3-Dof reconfigurable manipulator

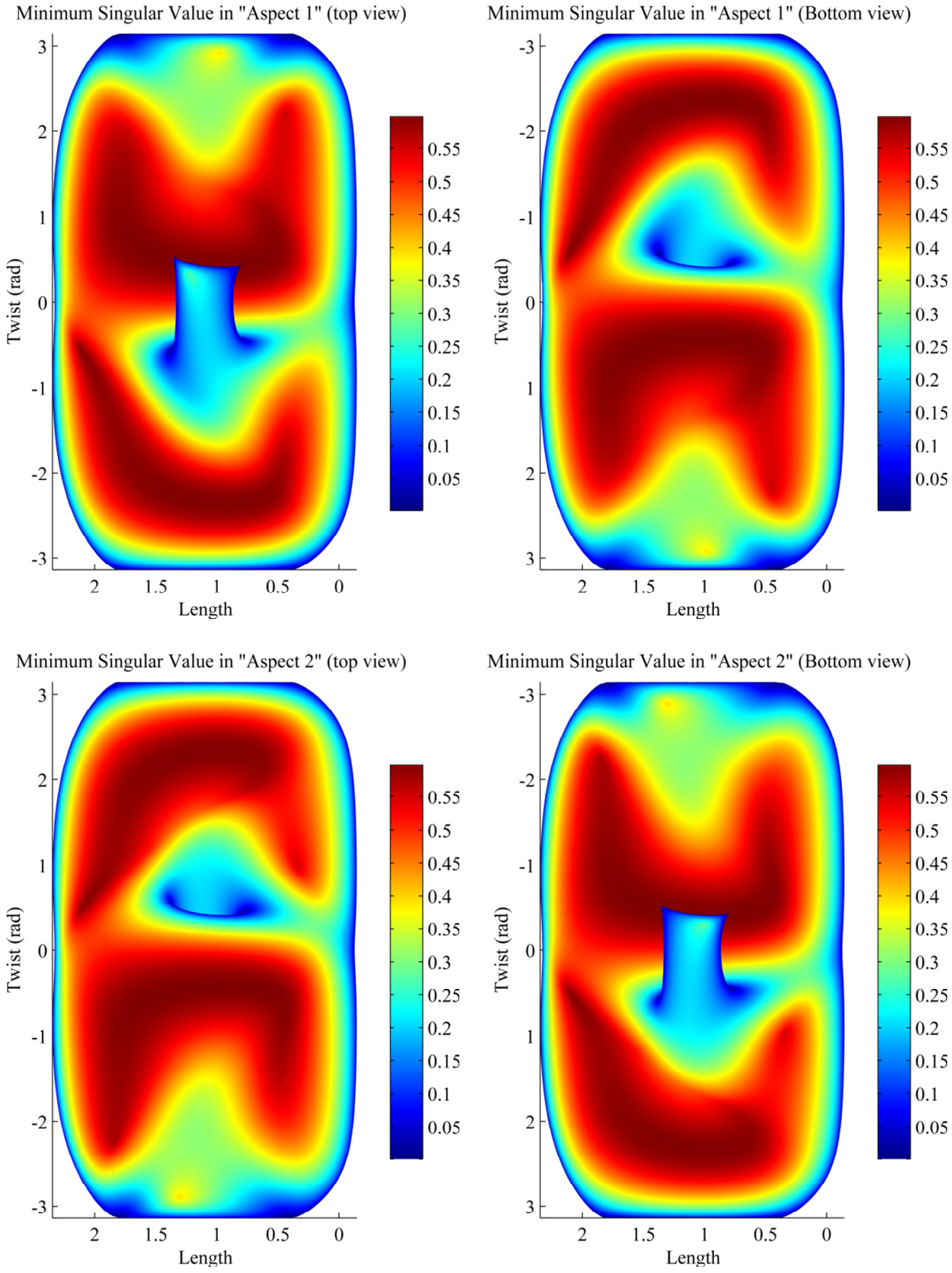


Figure 39: Two aspects of 3-DoF manipulator

7.2.3 Case Study III: 6-DoF Reconfigurable Manipulator

A 6-DoF reconfigurable manipulator with a full constraint on the end-effector and with one released cylindrical link is considered. Note that in such system both cylindrical joints may be released in a single mode, which would result in a 4-D workspace. But since the algorithm is not robust towards the sweeping of more than 2-DoF workspaces and also in order to be able to represent the workspace it is necessary to fix a few degrees of freedom. For this reason the second cylindrical joint is considered fix and the workspace is swept for the released DoF of first cylindrical joint. The manipulator is shown in Figure 40.

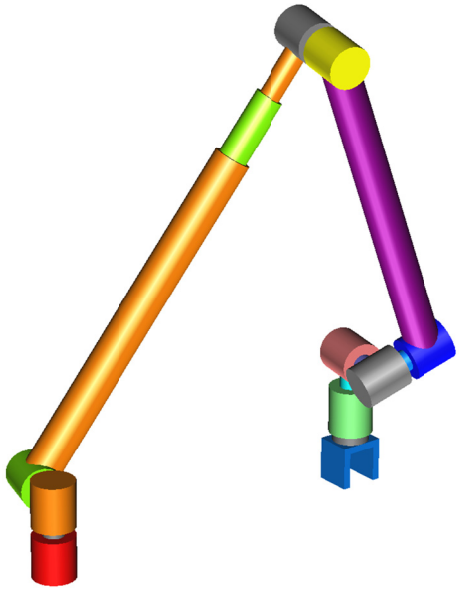


Figure 40: 6-DoF reconfigurable manipulator

For this system about 1,300,000 points were swept and the trajectories have been broken into about 6000 lines, all in less than 400 seconds. The whole workspace could be seen in Figure 41.

The workspace of this manipulator over other variables (i.e. $\theta_i, i = 2,3,4,5,6$) is reported in **0**Appendix C (6-DoF Workspace).

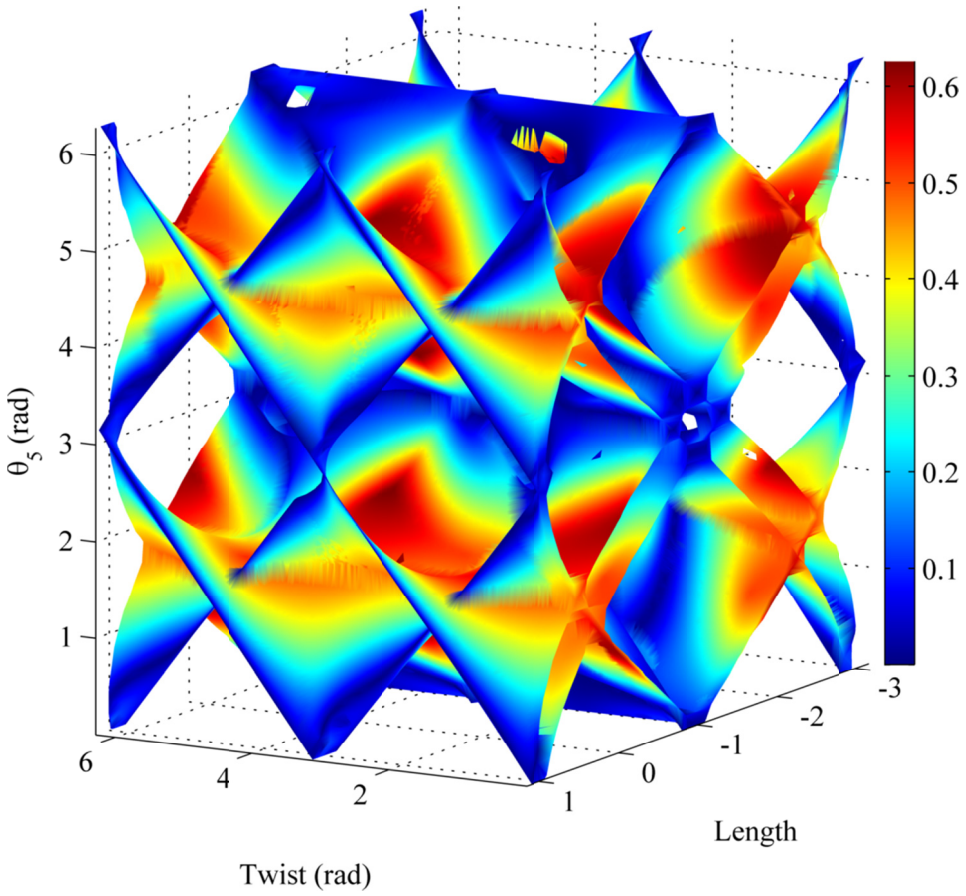


Figure 41: Workspace of 6-DoF reconfigurable manipulator with respect to fifth joint angle

8. Conclusions and Future Work

The kinematic and dynamic modeling process along with the simulation and control methods for the reconfigurable manipulator were reviewed. A developed Matlab package dedicated to reconfigurable manipulators was presented which could be used as a platform for future study of the reconfigurable manipulators.

A new algorithm for singularity avoidance namely the *singularity estimator* was also discussed. To the best of our knowledge this method is the only method in the literature that provides the distance index from singularity in the workspace (or input space). It provides accurate information on the nearest singularities in the workspace which could be used in a higher level controller such as trajectory planner or real-time control. Comparison of this method to two similar methods in the literature over a case study was also presented and the results confirm that the proposed method provides the best criterion in the sense of distance from singularity at workspace.

For workspace determination a new numerical approach named *random sweep of workspace* is proposed. Due to simplicity of the method, it is very fast in comparison to other analytical or numerical approaches. The workspaces of three different manipulators using this method were presented graphically. The results show that using this approach, high speed, desirable accuracy and acquiring information not only about the boundaries of the workspace, but also about the inside of the workspace could be achieved.

The proposed random sweep of workspace, although unlike other methods is both fast and provides information about the interior of workspace, it does not guarantee to sweep the whole workspace and human supervision may be required to obtain enough accurate result. If the simulation step is small with respect to workspace area, large number of points will be needed to sweep all parts of workspace with the desired accuracy. Simulation step could be maximized up to a limit if the size of workspace is large, the method could be very slow. As a future work a simple intelligent gain may be added to direct the randomness of the trajectory towards the un-swept positions in the workspace. However the calculation of the gain should be simple enough to maintain the high speed of the algorithm.

A possible improvement to the data representation of data gathered by RSW method is to use the information of minimum singular value in order to fill the un-swept holes in the workspace. The possibility of existence of un-swept holes are more at the higher manipulability parts of the workspace, since the algorithm increases the motion step when the minimum singular value is higher, for faster sweeping of workspace. Also to increase the efficiency of the algorithm a stop criterion may be added to stop the data sampling until a desired accuracy is obtained.

9. References

[1] F. Aghili and K. Parsa, "Design of a reconfigurable space robot with lockable telescopic joints," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006, pp. 4608-4614.

[2] F. Aghili and C.-Y. Su, "Reconfigurable Space Manipulators for In-orbit Servicing and Space Exploration," 2012.

[3] F. Aghili, "Pre-and post-grasping robot motion planning to capture and stabilize a tumbling/drifting free-floater with uncertain dynamics," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, pp. 5461-5468.

[4] E. Bayo, J. Garcia de Jalon, and M. A. Serna, "A modified Lagrangian formulation for the dynamic analysis of constrained mechanical systems," *Computer methods in applied mechanics and engineering,* vol. 71, pp. 183-195, 1988.

[5] G. H. Golub and C. F. Van Loan, *Matrix computations* vol. 3: JHU Press, 2012.

[6] F. Aghili, "A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: applications to control and simulation," *Robotics, IEEE Transactions on,* vol. 21, pp. 834-849, 2005.

[7] J.-P. Merlet, *Parallel robots*. Netherlands: Springer, 2006.

[8] J. J. Craig, *Introduction to robotics: mechanics and control*, Third Edition ed., 2004.

[9] J. M. Hollerbach, "A recursive lagrangian formulation of maniputator dynamics and a comparative study of dynamics formulation complexity," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 10, pp. 730-736, 1980.

[10] W. M. Silver, "On the equivalence of Lagrangian and Newton-Euler dynamics for manipulators," *The International Journal of Robotics Research,* vol. 1, pp. 60-70, 1982.

[11] P. I. Corke, "An automated symbolic and numeric procedure for manipulator rigid-body dynamic significance analysis and simplification," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, 1996, pp. 1018-1023.

[12] P. Fisette and J.-C. Samin, "Symbolic generation of large multibody system dynamic equations using a new semi-explicit Newton/Euler recursive scheme," *Archive of applied mechanics,* vol. 66, pp. 187-199, 1996.

[13] N. Docquier, A. Poncelet, and P. Fisette, "ROBOTRAN: a powerful symbolic gnerator of multibody models," *Mechanical Sciences,* vol. 4, pp. 199-219, 2013.

[14] W. Khalil and O. Ibrahim, "General solution for the dynamic modeling of parallel robots," *Journal of Intelligent and robotic systems,* vol. 49, pp. 19-37, 2007.

[15] F. Aghili, "A conceptual design for reconfigurable robots," *Proceedings of the Canadian Engineering Education Association,* 2011.

[16] A. A. Maciejewski and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *The international journal of robotics research,* vol. 4, pp. 109-117, 1985.

[17] J. M. Hollerbach and K. Suh, "Redundancy resolution of manipulators through torque optimization," *Robotics and Automation, IEEE Journal of,* vol. 3, pp. 308-316, 1987.

[18] A. Liegeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 7, pp. 868-871, 1977.

[19] S. Chiaverini, G. Oriolo, and I. Walker, "Kinematically Redundant Manipulators," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., ed: Springer Berlin Heidelberg, 2008, pp. 245-268.

[20] F. Aghili and K. Parsa, "A reconfigurable robot with lockable cylindrical joints," *Robotics, IEEE Transactions on,* vol. 25, pp. 785-797, 2009.

[21] P. Merat. *Matlab Package for a special Reconfigurable Manipulator Ideal for Space Application*. Available: http://youtu.be/28HhAUmrqnI

[22] J.-P. Merlet and C. Gosselin, "Parallel Mechanisms and Robots," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., ed: Springer Berlin Heidelberg, 2008, pp. 269-285.

[23] C. Gosselin and J. Angeles, "Singularity analysis of closed-loop kinematic chains," *Robotics and Automation, IEEE Transactions on,* vol. 6, pp. 281-290, 1990.

[24] D. Zlatanov, R. G. Fenton, and B. Benhabib, "Singularity analysis of mechanisms and robots via a motion-space model of the instantaneous kinematics," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, 1994, pp. 980-985.

[25] D. Zlatanov, R. Fenton, and B. Benhabib, "Identification and classification of the singular configurations of mechanisms," *Mechanism and Machine Theory,* vol. 33, pp. 743-760, 1998.

[26] S. Bhattacharya, H. Hatwal, and A. Ghosh, "Comparison of an exact and an approximate method of singularity avoidance in platform type parallel manipulators," *Mechanism and Machine Theory,* vol. 33, pp. 965-974, 1998.

[27] F. Pierrot, "A new design of a 6-DOF parallel robot," *J. of Robotics and Mechatronics,* vol. 2, pp. 308-315, 1990.

[28] F. Park and J. W. Kim, "Singularity analysis of closed kinematic chains," *Journal of mechanical design,* vol. 121, pp. 32-38, 1999.

[29] B. Fallahi, H. Lai, R. Naghibi, and Y. Wang, "A study of the workspace of five-bar closed loop manipulator," *Mechanism and Machine Theory,* vol. 29, pp. 759-765, 1994.

[30] P. Borrel and A. Liegeois, "A study of multiple manipulator inverse kinematic solutions with applications to trajectory planning and workspace determination," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, 1986, pp. 1180-1185.

[31] P. Wenger and D. Chablat, "Workspace and assembly modes in fully-parallel manipulators: A descriptive study," in *Advances in Robot kinematics: Analysis and control*, ed: Springer, 1998, pp. 117-126.

[32] G. Marani, J. Kim, J. Yuh, and W. K. Chung, "A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, 2002, pp. 1973-1978.

[33] C. A. Klein and B. E. Blaho, "Dexterity measures for the design and control of kinematically redundant manipulators," *The International Journal of Robotics Research,* vol. 6, pp. 72-83, 1987.

[34] R. Ranganath, P. Nair, T. Mruthyunjaya, and A. Ghosal, "A force–torque sensor based on a Stewart Platform in a near-singular configuration," *Mechanism and machine theory,* vol. 39, pp. 971-998, 2004.

[35] D. Nenchev and M. Uchiyama, "Dynamic analysis of parallel-link manipulators under the singularity-consistent formulation," in *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, 1996, pp. 1227-1233.

[36] D. Nenchev, S. Bhattacharya, and M. Uchiyama, "Dynamic analysis of parallel manipulators under the singularity-consistent parameterization," *Robotica,* vol. 15, pp. 375-384, 1997.

[37] M. Zein, P. Wenger, and D. Chablat, "Non-singular assembly-mode changing motions for 3-RPR parallel manipulators," *Mechanism and Machine Theory,* vol. 43, pp. 480-490, 2008.

[38] B. Dasgupta and T. Mruthyunjaya, "Singularity-free path planning for the Stewart platform manipulator," *Mechanism and Machine Theory,* vol. 33, pp. 711-725, 1998.

[39] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 2006, pp. 1874-1879.

[40] G. Liu, Y. Wu, X. Wu, Y. Kuen, and Z. Li, "Analysis and control of redundant parallel manipulators," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, 2001, pp. 3748-3754.

[41] T. Yoshikawa, "Manipulability of robotic mechanisms," *The international journal of Robotics Research,* vol. 4, pp. 3-9, 1985.

[42] E. Isaacson, *Analysis of numerical methods*: Courier Dover Publications, 1994.

[43] G. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix," *Journal of the Society for Industrial & Applied Mathematics, Series B: Numerical Analysis,* vol. 2, pp. 205-224, 1965.

[44] A. A. Maciejewski and C. A. Klein, "The singular value decomposition: Computation and applications to robotics," *The International Journal of Robotics Research,* vol. 8, pp. 63-79, 1989.

[45] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik,* vol. 14, pp. 403-420, 1970.

[46] J. Kim, G. Marani, W. K. Chung, and J. Yuh, "A general singularity avoidance framework for robot manipulators: task reconstruction method," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, 2004, pp. 4809-4814.

[47] T. Yoshikawa, "Analysis and control of robot manipulators with redundancy," in *Robotics research: the first international symposium*, 1984, pp. 735-747.

[48] J. Park, "Analysis and control of kinematically redundant manipulators: an approach based on kinematically decoupled joint space decomposition," PhD thesis, Pohang University of Science and Technology (POSTECH), 1999.

[49] P. A. Voglewede and I. Ebert-Uphoff, "Measuring" closeness" to singularities for parallel manipulators," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, 2004, pp. 4539-4544.

[50] T. Papadopoulo and M. I. Lourakis, "Estimating the jacobian of the singular value decomposition: Theory and applications," in *Computer Vision-ECCV 2000*, ed: Springer, 2000, pp. 554-570.

[51] K. Abdel-Malek, F. Adkins, H.-J. Yeh, and E. Haug, "On the determination of boundaries to manipulator workspaces," *Robotics and Computer-Integrated Manufacturing,* vol. 13, pp. 63-72, 1997.

[52] K. Abdel-Malek and H.-J. Yeh, "Analytical Boundary of the Workspace for General3-DOF Mechanisms," *The International Journal of Robotics Research,* vol. 16, pp. 198-213, 1997.

[53] B. M. St-Onge and C. M. Gosselin, "Singularity analysis and representation of the general Gough-Stewart platform," *The International Journal of Robotics Research,* vol. 19, pp. 271-288, 2000.

[54] E. Dupuis, E. Papadopoulos, and V. Hayward, "The singular vector algorithm for the computation of rank-deficiency loci of rectangular Jacobians," in *International Conference on Intelligent Robots and Systems, Maui, Hawai, USA*, 2001.

[55] H. Li, C. M. Gosselin, M. J. Richard, and B. M. St-Onge, "Analytic form of the six-dimensional singularity locus of the general Gough-Stewart platform," *Journal of Mechanical Design,* vol. 128, pp. 279-287, 2006.

[56] J.-P. Merlet, "Singular configurations of parallel manipulators and Grassmann geometry," *The International Journal of Robotics Research,* vol. 8, pp. 45-56, 1989.

[57] F. C. Park and J. W. Kim, "Manipulability and singularity analysis of multiple robot systems: A geometric approach," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, 1998, pp. 1032-1037.

[58] R. Di Gregorio, "Singularity-locus expression of a class of parallel mechanisms," *Robotica,* vol. 20, pp. 323-328, 2002.

[59] G. Liu, Y. Lou, and Z. Li, "Singularities of parallel manipulators: a geometric treatment," *Robotics and Automation, IEEE Transactions on,* vol. 19, pp. 579-594, 2003.

[60] O. Bohigas, M. Manubens, and L. Ros, "A complete method for workspace boundary determination on general structure manipulators," *Robotics, IEEE Transactions on,* vol. 28, pp. 993-1006, 2012.

[61] J. M. Porta, L. Ros, and F. Thomas, "A linear relaxation technique for the position analysis of multiloop linkages," *Robotics, IEEE Transactions on,* vol. 25, pp. 225-239, 2009.

[62] E. J. Haug, C.-M. Luh, F. A. Adkins, and J.-Y. Wang, "Numerical algorithms for mapping boundaries of manipulator workspaces," in *Proceedings of the 24th ASME Mechanisms Conference*, 1994.

[63] J. Snyman, J. Duffy, and L. du Plessis, "An optimization approach to the determination of the boundaries of manipulator workspaces," *Journal of Mechanical Design,* vol. 122, pp. 447-456, 2000.

[64] A. Kumar and K. Waldron, "The workspaces of a mechanical manipulator," *Journal of Mechanical Design,* vol. 103, pp. 665-672, 1981.

[65] P. Wenger, "Uniqueness domains and regions of feasible paths for cuspidal manipulators," *Robotics, IEEE Transactions on,* vol. 20, pp. 745-750, 2004.

[66] D. Oblak and D. Kohli, "Boundary surfaces, limit surfaces, crossable and noncrossable surfaces in workspace of mechanical manipulators," *Journal of Mechanisms, Transmissions and Automation in Design,* vol. 110, pp. 389-396, 1988.

[67] O. Bohigas Nadal, L. Ros Giralt, and M. Manubens, "A complete method for workspace boundary determination," 2010.

[68] J.-P. Merlet, C. M. Gosselin, and N. Mouly, "Workspaces of planar parallel manipulators," *Mechanism and Machine Theory,* vol. 33, pp. 7-20, 1998.

[69] C. Innocenti and V. Parenti-Castelli, "Direct kinematics of the 6-4 fully parallel manipulator with position and orientation uncoupled," in *Robotic Systems*, ed: Springer, 1992, pp. 3-10.

[70] I. Zabalza, J. Ros, J. J. Gil, J. M. Pintor, and J. M. Jimenez, "Tri-scott. a new kinematic structure for a 6-dof decoupled parallel manipulator," in *Proceedings of the workshop on fundamental issues and future research directions for parallel mechanisms and manipulators*, 2002, pp. 12-15.

[71] S. K. Agrawal, "Workspace boundaries of in-parallel manipulator systems," in *Advanced Robotics, 1991.'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, 1991, pp. 1147-1152.

[72] I. A. Bonev, D. Zlatanov, and C. M. Gosselin, "Singularity analysis of 3-DOF planar parallel mechanisms via screw theory," *Journal of Mechanical Design,* vol. 125, pp. 573-581, 2003.

[73] A. K. Dash, I. Chen, S. H. Yeo, and G. Yang, "Workspace generation and planning singularity-free path for parallel manipulators," *Mechanism and Machine Theory,* vol. 40, pp. 776-805, 2005.

[74] P. Wenger and J. El Omri, "Changing posture for cuspidal robot manipulators," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, 1996, pp. 3173-3178.

[75] P. Wenger, "Cuspidal and noncuspidal robot manipulators," *Robotica,* vol. 25, pp. 677-689, 2007.

[76] C. Innocenti and V. Parenti-Castelli, "Singularity-free evolution from one configuration to another in serial and fully-parallel manipulators," *Journal of Mechanical design,* vol. 120, pp. 73-79, 1998.

[77] M. Zein, P. Wenger, and D. Chablat, "Singular curves and cusp points in the joint space of 3-RPR parallel manipulators," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 2006, pp. 777-782.

[78] M. Baili, P. Wenger, and D. Chablat, "Classification of one family of 3R positioning manipulators," *arXiv preprint arXiv:0705.1344,* 2007.

[79] E. Macho, O. Altuzarra, C. Pinto, and A. Hernandez, "Singularity free change of assembly mode in parallel manipulators: Application to the 3RPR planar platform," *12th IFToMM World Congr., Besancon, France,* 2007.

[80] P. R. McAree and R. W. Daniel, "An explanation of never-special assembly changing motions for 3–3 parallel manipulators," *The International Journal of Robotics Research,* vol. 18, pp. 556-574, 1999.

[81] M. L. Husty, "Non-singular assembly mode change in 3-RPR-parallel manipulators," in *Computational Kinematics*, ed: Springer, 2009, pp. 51-60.

[82] B. L. Wellman, *Technical descriptive geometry*: McGraw-Hill Book Co., 1948.

[83] K. Abdel-Malek, J. Yang, D. Blackmore, and K. Joy, "Swept Volumes: Fundation, Perspectives, and Applications," *International Journal of Shape Modeling,* vol. 12, pp. 87-127, 2006.

[84] R. A. Brooks, "Planning collision-free motions for pick-and-place operations," *The International Journal of Robotics Research,* vol. 2, pp. 19-44, 1983.

[85] F. Freudenstein and E. Primrose, "On the analysis and synthesis of the workspace of a three-link, turning-pair connected robot arm," *Journal of Mechanisms, Transmissions and Automation in Design,* vol. 106, pp. 365-370, 1984.

[86] M. Herman, "Fast, three-dimensional, collision-free motion planning," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, 1986, pp. 1056-1063.

[87] S. Redon, Y. J. Kim, M. C. Lin, and D. Manocha, "Fast continuous collision detection for articulated models," *Journal of Computing and Information Science in Engineering,* vol. 5, pp. 126-137, 2005.

[88] C. Law, L. Sobierajski Avila, and W. Schroeder, "Application of path planning and visualization for industrial-design and maintainability-analysis," in *Reliability and Maintainability Symposium, 1998. Proceedings., Annual*, 1998, pp. 126-131.

[89] D. Alciatore and C. Ng, "Determining manipulator workspace boundaries using the Monte Carlo method and least squares segmentation," *ASME Robotics: Kinematics, Dynamics and Controls,* vol. 72, pp. 141-146, 1994.

[90] Y. Wang and G. S. Chirikjian, "Workspace generation of hyper-redundant manipulators as a diffusion process on SE (N)," *Robotics and Automation, IEEE Transactions on,* vol. 20, pp. 399-408, 2004.

[91] T. Seidl and H.-P. Kriegel, "Optimal multi-step k-nearest neighbor search," in *ACM SIGMOD Record*, 1998, pp. 154-165.

[92] D.-T. Lee and B. J. Schachter, "Two algorithms for constructing a Delaunay triangulation," *International Journal of Computer & Information Sciences,* vol. 9, pp. 219-242, 1980.

Appendices

Appendix A (Function Generation)

A sample function generated using the "*Model/MakeFunc.m*" function included in the Matlab package, is presented in the following:

1. Function prototype

```
function out = Phi1(a1, a2, a3, a4, a5, a6, a7, c1, c2, c3, c4, c5, c6, c7, e1, i1, e2, i2, m1, m2, m3, o1, o2, o3)
```

2. temporary variables

```
A1 = sin(a1);
B1 = cos(a1);
A2 = sin(a2);
B2 = cos(a2);
A3 = sin(a3);
B3 = cos(a3);
A4 = sin(a4);
B4 = cos(a4);
A5 = sin(a5);
B5 = cos(a5);
A6 = sin(a6);
B6 = cos(a6);
A7 = sin(a7);
B7 = cos(a7);
E1 = sin(e1);
F1 = cos(e1);
E2 = sin(e2);
F2 = cos(e2);
```

```
t15 = B4*F1*t22 - A4*t21 + A2*B1*B4*E1;
t14 = B4*t23 + A4*F1*t24 - A1*A2*A4*E1;
t13 = B4*t21 + A4*F1*t22 + A2*A4*B1*E1;
t12 = A4*t23 - B4*F1*t24 + A1*A2*B4*E1;
t56 = E2*t12;
t60 = F2*t20;
t11 = t60 - t56;
t57 = E2*t15;
t59 = F2*t19;
t10 = t59 + t57;
t31 = B5*E2*t25;
t34 = B5*F2*t17;
t41 = A5*t18;
t9 = t31 - t41 + t34;
t8 = B5*t18 + A5*E2*t25 + A5*F2*t17;
t7 = B5*t13 + A5*F2*t15 - A5*E2*t19;
t6 = A5*F2*t12 - B5*t14 + A5*E2*t20;
t30 = B5*E2*t20;
```

```
O1 = sin(o1);
P1 = cos(o1);
O2 = sin(o2);
P2 = cos(o2);
O3 = sin(o3);
P3 = cos(o3);

t28 = A2*A3*E1;
t25 = B2*F1 - t28;
t36 = A1*A3;
t24 = B1*B3 + t36*B2;
t38 = A3*B1;
t23 = t38 - A1*B2*B3;
t22 = A1*B3 - t38*B2;
t21 = t36 + B1*B2*B3;
t55 = E1*t24;
t20 = t55 + A1*A2*F1;
t54 = E1*t22;
t19 = t54 - A2*B1*F1;
t37 = A2*B3;
t18 = A4*B2*E1 - t37*B4 + A2*A3*A4*F1;
t17 = A2*A4*B3 + B2*B4*E1 + 2*A3*B4*F1;
t58 = E2*t17;
t61 = F2*t25;
t16 = t61 - t58;
```

```
t32 = B5*F2*t12;
t40 = A5*t14;
t5 = t40 + t32 + t30;
t29 = B5*E2*t19;
t33 = B5*F2*t15;
t39 = A5*t13;
t4 = t39 - t33 + t29;
t3 = B6*t8 + A6*t16;
t2 = B6*t7 - A6*t10;
t1 = B6*t6 + A6*t11;
t26 = 0.15*A1*A2;
t27 = 0.15*A2*B1;
t35 = 0.15*B2;
t42 = A6*t6;
t43 = A6*t7;
t44 = A6*t8;
t45 = A7*t4;
t46 = A7*t5;
t47 = A7*t9;
t48 = B6*t10;
t49 = B6*t11;
t50 = B6*t16;
t51 = B7*t1;
t52 = B7*t2;
t53 = B7*t3;
t62 = O3*P1;
t63 = P1*P3;
```

3. Function output based on temporary variables

```
out = reshape([0.15*A1 - m1 - 0.15*t54 - 0.15*t59 + 0.3*t43 + (i2 + 1)*t13 + t21*(i1 + 1) + 0.15*t39 - 0.15*t57 + t27 + 0.3*t48 - 0.15*t33 + t27*F1 + 0.15*t29,0.15*t55 - m2 - 0.15*B1 + 0.3*t42 + 0.15*t60 - (i2 + 1)*t14 - t23*(i1 + 1) - 0.15*t40 - 0.15*t56 + t26 - 0.3*t49 - 0.15*t32 + t26*F1 - 0.15*t30,0.15*t58 - m3 - 0.3*t44 - 0.15*t61 - (i2 + 1)*t18 - 0.15*t41 - t35 + 0.3*t50 - t35*F1 + 0.15*t28 + 0.15*t31 + t37*(i1 + 1) + 0.15*t34 + 0.15,0.00000000000000000061232*t52 + t43 + 0.00000000000000000061232*t45 + t48 - P1*P2,0.00000000000000000061232*t51 + t42 - 0.00000000000000000061232*t46 - t49 - O1*P2,O2 - t44 + 0.00000000000000000061232*t47 + t50 -
```

0.00000000000000000061232*t53,B7*t4 - A7*t2 + O1*P3 - O2*t62,- A7*t1 - B7*t5 - t63 - O1*O2*O3,B7*t9 + A7*t3 - O3*P2,0.00000000000000000061232*t43 - t52 - t45 + 0.00000000000000000061232*t48 - O1*O3 - O2*t63,0.00000000000000000061232*t42 - t51 + t46 - 0.00000000000000000061232*t49 + t62 - O1*O2*P3,0.00000000000000000061232*t50 - t47 - 0.00000000000000000061232*t44 + t53 - P2*P3,], [12 1]);

Appendix B (Conversion Algorithm)

In the following a conversion algorithm is presented for reconfigurable manipulator. The conversion algorithm is demonstrated in Table 16. The corresponding terminology could be found in Table 15.

Table 15: Conversion algorithm terminology

<table>
<tr><th>Structure</th><th colspan="4">Subfields</th></tr>
<tr><td rowspan="8">"locked"
(locked mode)</td><td rowspan="3">"DH.i"
(i-th row of DH table)</td><td colspan="3">"alpha" (α_i)</td></tr>
<tr><td colspan="3">"a" (a_i)</td></tr>
<tr><td colspan="3">"d" (d_i)</td></tr>
<tr><td rowspan="5">"Dyn.i"
(dynamic properties of link i)</td><td rowspan="3">link i:
regular</td><td colspan="3">"theta" (θ_i)</td></tr>
<tr><td colspan="3">"I" (I_i)</td></tr>
<tr><td colspan="3">"m" (m_i)</td></tr>
<tr><td rowspan="3">link i:
cylindrical</td><td colspan="3">"P" (P_i)</td></tr>
<tr><td>"p1"
(cylindrical joint part 1 connected to i-th joint)</td><td>"I" (I_i)</td></tr>
<tr><td>"p2"
(cylindrical joint part 2 connected to joint i+1)</td><td>"m" (m_i)
"P" (P_i)</td></tr>
</table>

<table>
<tr><td rowspan="8">"modeX"
(a released mode)</td><td rowspan="4">"DH.i"
(i-th row of DH table)</td><td rowspan="4">link i:
cylindrical</td><td colspan="2">"alpha" (α_i)</td></tr>
<tr><td colspan="2">"a" (a_i)</td></tr>
<tr><td colspan="2">"d" (d_i)</td></tr>
<tr><td colspan="2">"theta" (θ_i)</td></tr>
<tr><td rowspan="4">link i:
regular</td><td rowspan="4">"Rot" (Rotational DoF)
"Lin" (Linear DoF)
"Joint" (Joint DoF)</td><td>"alpha" (α_i)</td></tr>
<tr><td>"a" (a_i)</td></tr>
<tr><td>"d" (d_i)</td></tr>
<tr><td>"theta" (θ_i)</td></tr>
<tr><td colspan="2">"Dyn.i"
(dynamic properties of link i)</td><td colspan="2">"I" (I_i)
"m" (m_i)
"P" (P_i)</td></tr>
<tr><td rowspan="2">"plus.j"
(j-th cylindrical joint)</td><td>"lin" (l_j)
(linear DoF)</td><td colspan="2">-</td></tr>
<tr><td>"alpha" (γ_j)
(rotational DoF)</td><td colspan="2">-</td></tr>
</table>

Table 16: Property conversion algorithm between locked and unlocked mode

modeX: arbitrary mode

R_cyl_pre = rotz(-pi/2) = $R_{Cyl_{pre}}$

R_cyl = rotz(-pi/2)

for i from 1 to n (number of links), do:

if link i-1 is a cylindrical joint, do:

j = find(i-1, cylindircal_joints)

if cylindrical joint j is released, do:

(modify previous joint properties due to coordinate rotation)

modeX.DH.(i-1).theta = locked.DH.(i-1).theta + pi/2

modeX.Dyn.(i-1).I = R_cyl_pre*locked.Dyn.(i-1).I.p1

modeX.Dyn.(i-1).m = locked.Dyn.(i-1).m.p1

modeX.Dyn.(i-1).P = R_cyl_pre*locked.Dyn.(i-1).P.p1

(Rotational DoF of cylinder: Rot_{cyl})

modeX.DH.i.Rot = {a = 0, d = 0, I = $\bar{0}$, m = 0, P = 0}

modeX.DH.i.Rot.alpha = pi/2

modeX.DH.i.Rot.theta = locked.DH.i.alpha + plus.j.alpha

(Linear DoF of cylinder: Lin_{cyl})

modeX.DH.i.Lin = {alpha = 0, a = 0, theta = 0}

modeX.Dyn.i.Lin.d = locked.DH.i.a + plus.j.lin

modeX.Dyn.i.Lin.I = R_cyl*locked.Dyn.(i-1).I.p2

modeX.Dyn.i.Lin.m = locked.Dyn.(i-1).m.p2

modeX.Dyn.i.Lin.P = R_cyl*locked.Dyn.(i-1).P.p2

modeX.DH.i.Rot.alpha = locked.DH.i.alpha

(Linear DoF of cylinder: *Joint*)

modeX.{DH.i.d, Dyn.i.{I, m, P}} =

locked.{DH.i.d, Dyn.i.{I, m, P}}

modeX.DH.i.Joint.alpha = -pi/2

modeX.DH.i.Joint.a = 0

modeX.DH.i.Joint.theta = locked.DH.i.theta – pi/2

else (link i-1 is a locked cylindrical joint)

(Pre-cylindrical joint)

modeX.Dyn.(i-1).m = locked.Dyn.(i-1).m.p1 + locked.Dyn.(i-1).m.p1

% P.12 and I.12 are calculated by equations (2-21) and (2-22)

% and are not repeated here

modeX.Dyn.(i-1).{P, I} = {P.12, I.12}

(Cylindrical joint)

modeX.{DH.i.{d, theta}, Dyn.i.{I, m, P}} =

locked.{DH.i.{d, theta}, Dyn.i.{I, m, P}}

modeX.DH.i.alpha = locked.DH.i.alpha + plus.j.alpha

modeX.DH.i.a = locked.DH.i.a + plus.j.lin

end

else (link i-1 is a regular link)

modeX.DH.i = locked.DH.i

modeX.Dyn.i = locked.Dyn.i

end

end

Nomenclature

Term	**Description**
EE	End-effector
DoF(s)	Degree(s) of freedom
n-D	*n*-Dimensional
SVD	Singular value decomposition
MoM (*MoS*)	Measure of manipulability (singularity)
RSW	The proposed random sweep of workspace method