A Framework for Identifying Solvable Joint Reactions in Over-constrained Mechanisms

A Thesis

Submitted for the Degree of

Master of Science

in the **Faculty of Engineering**

by

Shivam Sharma

Mechanical Engineering
Indian Institute of Science
Bangalore – 560 012 (INDIA)

November 2018

DEDICATED TO

Mom

CERTIFICATE

I hereby certify that the content embodied in this thesis titled **A Framework for Identifying Solvable Joint Reactions in Over-constrained Mechanisms** has been carried out by Mr. Shivam Sharma at the Indian Institute of Science, Bangalore under my supervision and that it has not been submitted elsewhere for the award of any degree or diploma.

Signature of the Thesis Supervisor: .....

Professor Ashitava Ghosal
Dept. of Mechanical Engineering
Indian Institute of Science, Bangalore

DECLARATION

I hereby declare that the content embodied in this thesis titled **A Framework for Identifying Solvable Joint Reactions in Over-constrained Mechanisms** is the research carried out by me at the Department of Mechanical Engineering, Indian Institute of Science, Bangalore under the supervision of Prof. Ashitava Ghosal, Department of Mechanical Engineering, IISc. In keeping with the general practice in reporting scientific observations, due acknowledgment has been made wherever the work described is based on the findings of other investigations.

Signature of the Author: .....

Shivam Sharma
Dept. of Mechanical Engineering
Indian Institute of Science, Bangalore

Acknowledgements

I am greatly indebted to my advisor Prof. Ashitava Ghosal for his excellent guidance, teaching and support throughout the program. His course on Robotics was the one which enabled me for this problem statement and provided me with all the tools needed for this thesis. I would also like to thank Prof. Ananthasuresh for his insights into my problem, and also the brief discussions I had with him about English Literature. I am also indebted to Prof. Rudra Pratap for his excellent course on Rigid-body Dynamics, which laid the foundation for me to venture into the field of multi-body dynamics. Finally, I am indebted to my friends in IISc – Pranay, Mahesh and Eshan – for the amazing discussions on Mathematics, Mechanics and Philosophy. I would also like to thank my friends Karthik, Rishant and Shailendra, who without being in IISc, remained a constant motivation and support for me throughout these years.

Finally, I will thank my Mother and my friend Anu, without whom, this thesis would never have been possible. They stood tall when I did fall.

Abstract

Over-constrained mechanisms have mobility over a finite range of motion even though they should be structures according to the Grübler- Kutzbach mobility criterion. This happens because of the unique geometry of over-constrained mechanisms which results in motion of the links of the mechanism. In over-constrained mechanisms, some of the constraints are redundant and not all reactions at the joints can be solved for uniquely. Solving for such reactions would require us to give up the rigid-body assumptions and consider the links as flexible. The usual strategy that is employed by many multi-body simulation packages is to either ask the user to remove such redundant constraints, or eliminate the constraints automatically, both of which can lead to erroneous joint reactions. Other methods use purely mathematical operations and do not take into account the physical properties of the mechanism. Further, methods have been developed which add weighing factors, reflecting the elasticity of the links/joints, into these mathematical methods. Finally, the most reliable way is to bring the material constitutive equations into the picture and solving the problem using finite element analysis (FEA) solvers.

In this thesis, we propose a different strategy based on the observation that some of the joint reactions may be solved for uniquely in over-constrained mechanisms solely on the basis of rigid-body assumptions. We propose a method to find such a set of uniquely solvable joint reactions. If the desired constraint reactions are in this set, then there is no need to pass the problem over to the FEA solver and, hence the proposed approach can hugely benefit us in terms of computational efficiency. The method to obtain the uniquely solvable joint reactions is named as "Reaction Solvability Analysis (RSA)." In this thesis, we present an algorithm for RSA based on the investigation of the Jacobian matrix of the over-constrained mechanism. The developed RSA methodology is tested on various mechanisms using different coordinate formulations and a comparison is made of their efficiency. It is shown that the well-known natural coordinates can be modified to make them very useful for RSA. The RSA approach is illustrated on several planar and spatial over-constrained mechanisms. The constrained dynamics equations are also derived and from the simulations, the joint reactions for the uniquely solvable joints are obtained as a function of time.

Contents

Acknowledgements i

Abstract ii

Contents iii

List of Figures vi

List of Tables vii

1 Introduction 1

1.1 Motivation . . . 1
1.2 Understanding Redundant Constraints . . . 2
1.2.1 The Problem of Joint Reaction Determinacy . . . 3
1.2.2 Redundant Constraints Handling Strategies . . . 3
1.2.3 Status of RCH in Multi-body Dynamics Solvers . . . 4
1.2.4 An Alternative RCH Strategy . . . 4
1.3 Coordinate Formulations for RSA . . . 5
1.4 Scope and Contribution of the Thesis . . . 5
1.5 Outline of Thesis . . . 6

2 Kinematics and Dynamics of Robotic Mechanisms 8

2.1 Representation of Rotations . . . 8
2.1.1 Rotation Matrices . . . 9
2.1.2 Quaternions . . . 10
2.1.3 Composition of Rotations . . . 11
2.2 Equations of Motion of Multi-body Systems . . . 11
2.3 Coordinates for Kinematic Modeling . . . 12

2.3.1 Absolute Coordinates (Reference Point Coordinates) . . . 13
2.3.2 Natural Coordinates (Fully Cartesian Coordinates) . . . 14
2.3.3 Relative Coordinates (Joint Coordinates) . . . 15
2.4 Constraints and Joint Reactions . . . 15
2.4.1 Joint Reactions from Absolute Coordinates . . . 16
2.4.2 Joint Reactions from Natural Coordinates . . . 16
2.4.3 Joint-Augmented Natural Coordinate Formulation . . . 17
2.5 Comparison of Different Coordinate Formulations . . . 18

3 Methodology Development **19**
3.1 Over-constrained Mechanisms . . . 19
3.2 Dynamics Equations Formulation . . . 20
3.2.1 Integration of ODE's . . . 22
3.3 Solving Redundantly Constrained Equations . . . 23
3.4 Development of RSA Methodology . . . 25
3.4.1 Methodology 1 . . . 25
3.4.2 Methodology 2 . . . 25
3.5 Comparison of RSA Implementation using Natural and Absolute Coordinates . . . 26
3.6 Summary . . . 27

4 Implementation of Reaction Solvability Analysis **28**
4.1 Parallelogram Mechanism . . . 28
4.1.1 Kinematic Modeling (Absolute Coordinates) . . . 28
4.1.2 Apply RSA Methodology-2 . . . 31
4.2 Kempe Burmester Mechanism . . . 31
4.2.1 Kinematic Modeling (Absolute Coordinates) . . . 32
4.2.2 Applying RSA Methodology-2 . . . 34
4.3 Bennett Mechanism (Absolute Coordinates) . . . 34
4.3.1 Kinematic Modeling . . . 35
4.3.2 Applying RSA Methodology-2 . . . 37
4.4 Bennett Mechanism (Natural Coordinates) . . . 37
4.4.1 Kinematic Modeling . . . 37
4.4.2 Applying RSA Methodology-2 . . . 39
4.5 A Planar Over-Constrained Mechanism . . . 39
4.5.1 Modeling using Natural Coordinates . . . 39

4.5.2 Applying RSA Methodology-2 . . . 41
4.6 Results and Discussion . . . 41

5 Implementation : Finding Joint Reactions 43
5.1 Kinematic Modeling of the Planar Mechanism . . . 43
5.2 Dynamics of the Planar Mechanism . . . 45
5.2.1 Step-1 : Finding Jacobian . . . 45
5.2.2 Step-2 : Finding mass matrix . . . 45
5.2.3 Step-3 : Integrating Dynamics Equations . . . 46
5.3 Summary . . . 47

6 Conclusions 50
6.1 Concluding Remarks . . . 50
6.2 Future Directions . . . 51

Appendices 52

A Mathematica® Notebooks 53
A.1 Kempe Burmester . . . 54
A.2 Bennett Mechanism RSA using Absolute Coordinates . . . 59
A.3 Bennett Mechanism RSA using Natural Coordinates . . . 64
A.4 Dynamics of a Planar Mechanism . . . 69

B Tools Developed 78
B.1 Quaternion Based Loop-closure . . . 78
B.2 Automatic Constraint Equations Generation . . . 80
B.3 Generating CAD Models . . . 81

C SymPy based Quaternion Library 84

References 89

List of Figures

1.1 Hinged Door 2

2.1 Rotation around X-Axis 9
2.2 General transformation of a vector in 3D 10
2.3 Planar 4-Bar Modelled with different Coordinates 13
2.4 4-Bar Modelled with Natural Coordinates 16
2.5 4-Bar Modelled with Modified Natural Coordinates 17

3.1 Bennett Mechanism 20
3.2 Using NDSolve in Mathematica® 22
3.3 Proposition 1 and 2 Explained 24

4.1 Parallelogram Mechanism 30
4.2 Kempe Burmester Mechanism 32
4.3 Bennett Mechanism 35
4.4 Bennett Mechanism modeled using Natural Coordinates 38
4.5 A Planar over-constrained mechanism 40

5.1 Absolute Coordinates Formulation of a planar mechanism 44
5.2 Reaction Force vs Time 47
5.3 Reaction Force vs Time 47
5.4 Reaction Force vs Time 48
5.5 Reaction Force vs Time 48
5.6 Reaction Force vs Time 48
5.7 Reaction Force vs Time 49

B.1 Automatically generated Bennett Mechanism, rendered in SolidWorks 83

List of Tables

4.1 Methodology-2 Results for Parallelogram Mechanism 31
4.2 Methodology-2 Results for Kempe-Burmester Mechanism 34
4.3 Methodology-2 Results for Bennett Mechanism 37
4.4 Methodology-2 Results for the Planar Mechanism 41

List of Code Samples

B.1 Bennett Forward Kinematics using Quaternions 78
B.2 Generating Constraint Equations using SymPy for Bennett Mechanism 80
B.3 Automatic CAD Model Generation 81
C.1 Quaternion Module for Sympy 84

Chapter 1

Introduction

In this chapter, we introduce the problem studied in this work, discuss the motivation behind the problem, and the related work that has been done in this area. The terminology used throughout the work is also introduced in this chapter.

1.1 Motivation

In some mechanisms, the actual degree of freedom (DOF) is more than the number as computed using the well-known Grübler-Kutzbach criterion[1]. This happens when multiple joints are constraining the same degrees of freedom. Such joint constraints are redundant, and they give rise to what is called as redundantly constrained or **over-constrained mechanisms**. Over-constrained mechanisms are an important class of mechanisms since, due to redundant constraints, they have better load-bearing capabilities when the movement is stopped – once the motion of a link in the over-constrained mechanism is stopped they become structures (with DOF less than zero) and can withstand external loads. For this reason they are widely used as deployable mechanisms in space and other terrestrial applications[2].

However, over-constrained mechanisms have always been considered paradoxical and tough to analyze. This is because of several reasons:

1. It is difficult to calculate the correct mobility of the mechanism using simple counting-based criterion such as the Grübler-Kutzbach criterion.

2. The kinematics of the mechanism can be solved only after the redundant constraints have been identified and removed.

3. Not all the joint reactions in these mechanisms can be obtained as there are linearly-dependent constraint equations leading to singular Jacobian matrices.

It is the third paradoxical property (3) of over-constrained mechanisms that is the topic of this thesis.

Over-constrained mechanisms have been analysed extensively for their interesting properties. Waldron [3, 4] did rigorous mathematical analysis and gave the first comprehensive list of over-constrained linkages. Mavroidis [5] presented a new systematic method for dealing with over-constrained mechanisms using which they were able to verify all previously known over-constrained mechanisms. Mavroidis [6] then gave a new general method which allowed them to verify all known over-constrained mechanisms. While earlier works dealing with finding new over-constrained mechanisms were either not analytical or restricted to specialized groups, [6] presented a general analytical method.

1.2 Understanding Redundant Constraints

Redundant constraints are those that can be removed from the multi-body system without changing the kinematics of the system[7]. These constraints are often introduced due to construction or fabrication. For example, consider a door with 3 hinges. The motion of the door remains kinematically the same even if we remove 2 of the hinges. However, having 2 extra hinges allows for a more sturdy door and the door can withstand larger loads.

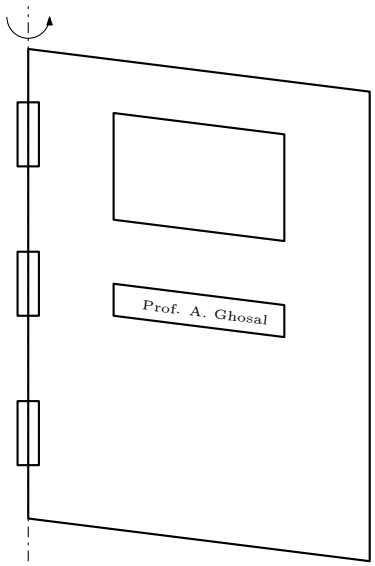


Figure 1.1: Hinged Door

Mathematically, redundant constraints are constraint equations which are linearly dependent, hence giving rise to a rank-deficient Jacobian matrix.

1.2.1 The Problem of Joint Reaction Determinacy

Redundant constraints can be eliminated from the mathematical model of a multi-body system to solve for its kinematics. The choice of constraints which are eliminated is usually arbitrary. Again using the hinged-door analogy, a hinged door will rotate the same if any 2 hinges are removed. However, situation is different in dynamic (or kinetostatic) analysis. Two bodies that form a kinematic pair act on each other by a reaction force and/or torque. These reaction forces/torques obtained from different sets of constraint equations will usually be different. Hence, one can not really solve for unique joint reactions in such a system. To solve for joint reactions in such a system, one needs to discard the rigid body assumption, introduce flexibility and constitutive relations of the material and resort to finite element analysis. This problem of indeterminacy of reactions solely on the basis of equilibrium equations is well known in statics since the time of Maxwell and is known as indeterminate structures[8].

1.2.2 Redundant Constraints Handling Strategies

There are various strategies employed for redundant constraints handling (termed as RCH, henceforth):

RCH Strategy 1 *The multi-body system is solved by eliminating any arbitrary set of dependent constraints equations.*

RCH Strategy 2 *The set of equations is not modified and algorithms capable of dealing with dependent equations are used. For example the minimum-norm solution and augmented Lagrangian [9] approach can be used.*

RCH Strategy 3 *Some of the methods in 2 are purely mathematical operations and do not reflect any physical properties of the system. To make methods in 2 resemble reality closely, penalty-based and weighing factors based methods are used.*

RCH Strategy 4 *The material constitutive equations are brought in and the problem is solved using finite element analysis (FEA) solvers.*

For the purpose of kinematic analysis, all RCH strategies yield correct results. Hence, RCH strategy 1 and 2 are used as they are the easiest to implement. RCH Strategies 2 and 3 have been developed and deployed for dynamics problems by [10, 11].

For dynamic analysis of over-constrained mechanisms, where joint reactions have to be calculated, RCH strategies 1 and 2 give incorrect results[12]. Strategy 3 is an improvement and

strategy 4 yields the most accurate results. In this work, we propose an alternative strategy in Section 1.2.4, which would be the most preferable in certain cases, where the joint reactions of interest turn out to be solvable without considering constitutive equations.

In line with the Strategy 2, Liu et. al. [13, 14] gave a method for the force analysis of over-constrained parallel mechanisms using a weighted Moore-Penrose generalized inverse. Wojtyra et. al. [15] gave a comparison of three different approaches : elimination of redundant constraints, pseudoinverse-based calculations, and the augmented Lagrangian formulation.

Strategy 3 presents better methods for handling the indeterminacy problem, by using penalty methods and weighting factors. Gonzalez [16] discussed the use of penalty formulations in dynamic simulation and analysis of redundantly constrained multibody systems. Ruzzeh [17] discussed another penalty based method for dynamics analysis of over-constrained mechanisms.

Strategy 4 involves using the material constitutive equations to get a full set of equations. In this spirit, [18] extended the Newton-Euler formulation to develop the dynamic model of an over-constrained parallel kinematic machine. Zahariev et. al. [19] discussed generalized Newton-Euler dynamic equations and applied them for the case of finite element discretization of flexible links.

The general problem statement of handling redundant constraints has been well compiled in the review papers by Liu et. al. [20] and Garcia et. al. [12].

1.2.3 Status of RCH in Multi-body Dynamics Solvers

Redundant constraints are a source of trouble for most multi-body software packages. It is recommended that they be avoided and this is evident from the manuals of well-known packages such as SolidWorks [21, 22] and SimMechanics [23]. A forum page [23] on redundant constraints reads as:

> When you have redundant constraints in your model, the computed forces in your model can not be guaranteed to be correct. While solving the system, Simulink needs to disregard one of the constrains in order to make the simulation succeed. Often redundant constraints can lead to errors while simulating so it is a good practice to avoid them.

1.2.4 An Alternative RCH Strategy

In this work, we propose an alternative strategy for redundant constraints handling (RCH). It is a combination of the four strategies discussed in 1.2.2. The strategy is based on the idea that even though **not** all the joint reactions can be solved for uniquely in an over-constrained

mechanism on the basis of dynamics equations alone, some joint reactions can be solved for uniquely, and as a consequence, some amount of extensive computations can be avoided. For the joint reactions which cannot be solved for uniquely, an FEA solver or other strategies need to be used. The focus of this work is to obtain the joint reactions which can be solved for uniquely in an over-constrained mechanism. This is named as Reaction Solvability Analysis (RSA) and formally

RCH Strategy 5 (Reaction Solvability Analysis (RSA)) *Obtain the set of joint reactions which can be solved for uniquely, without considering flexibility, in an over-constrained mechanism. If the desired reactions are in this set, then there is no need to use an FEA solver.*

As mentioned, this strategy provides opportunities for vast improvements in computational performance of many multi-body problems as these problems can now be solved simply using rigid-body assumptions.

For RSA, an algebraic approach using mobility equation was suggested by [24]. Later, Jacobian-based methods were suggested by [25] and [9]. The approach for RSA which we propose in this thesis has its roots in [9].

1.3 Coordinate Formulations for RSA

Most dynamics formulations usually hide joint reactions and only a few coordinate formulations actually allow finding joint reactions. Among them, absolute coordinates are the most natural choice when it comes to finding joint reactions. However, Blajer et. al. [26] has suggested a novel method for finding joint reactions using relative coordinates, termed as "augmented joint coordinate method".

In this work, we have suggested a modified form of natural coordinates which can be used for reaction solvability analysis (RSA), although we still have to resort to absolute coordinates for finding joint reactions. This has not been done before, to the best of our knowledge. Natural Coordinates offer the benefit that the constraint equations are maximally quadratic [27]. This advantage has proved Natural Coordinates to be a better choice in many applications[28]. Further, work related to the physical interpretation of joint reactions using Natural Coordinate (i.e. finding out the joint reactions using Natural Coordinates) has been discussed by Liu [29] and Czaplicki [30].

1.4 Scope and Contribution of the Thesis

The main goal of this work is to develop a methodology for reaction solvability analysis for the proposed RCH Strategy 5, and implement that methodology for a few over-constrained

mechanisms to demonstrate its usage. We have also compared the implementation of our methodology using various coordinate formulations, and in doing so, improved upon an existing coordinate formulation and found it to be the most suitable one for RSA. The main contributions of this thesis are:

- Formulating an RSA algorithm using Jacobian matrix investigation and implement it on a class of over-constrained mechanisms, which haven't been dealt with before in existing literature. Programs for implementing the algorithm were written in Mathematica®.
- Comparing different coordinate formulations for the kinematic modeling of mechanisms based on their effectiveness for our RSA algorithm. Several researchers have shown that absolute coordinates are the most natural choice. However, it is shown that the natural coordinates, in their slightly modified form, called as joint-augmented natural coordinates makes them not only suitable for RSA algorithm, but are better than absolute coordinates in terms of computational efficiency.
- (Auxilary Contributions) A Python program was developed for automatically generating constraint equations in natural coordinates. Another Python program was developed to automatically generate CAD models of single-loop spatial parallel mechanisms. A SymPy (a Python library for symbolic mathematics) module was developed for implementing quaternions symbolically. These contributions have been included in the Appendices.

1.5 Outline of Thesis

Rest of the thesis is organized in the following manner. In Chapter-2 Kinematics and Dynamics of Robotic Mechanisms, we first discuss the kinematic modeling of mechanisms using different types of coordinates and compare them based on their usability for RSA. We propose the "Joint Augmented Natural Coordinates" which modifies the natural coordinates to make it more suitable for RSA. Second, this chapter establishes the conventions used throughout the thesis for specifying rotations and other variables. In Chapter-3 Methodology Development, we discuss and develop methodology for RSA, based on the investigation of Jacobian matrix. Further, we present the dynamic equations of over-constrained mechanisms used in Chapter 5 to solve for the "solvable" joint reactions. In Chapter-4 Implementation of Reaction Solvability Analysis, we present the results of the implemented RSA algorithm on a select few over-constrained mechanisms. In Chapter-5 Implementation : Finding Joint Reactions, we use the dynamics equations to simulate a planar over-constrained mechanism by giving initial conditions and force excitations. The "solvable" joint reactions of that mechanism are then solved for using

these equations. In Chapter-6 Conclusions, we provide concluding remarks and discuss further research scope for this thesis.

Chapter 2

Kinematics and Dynamics of Robotic Mechanisms

In this chapter, we discuss the methods used in this thesis for modeling the kinematics of mechanisms. This includes description of the used coordinate systems and methods for representing translation and rotation (pose). The different coordinate system formulations are compared on the basis of their usability for performing reaction solvability analysis (RSA) and we propose a modification of the natural coordinates named as "joint augmented natural coordinates". Finally, the forward dynamic equations of the mechanisms are derived, which are used to simulate the forward dynamics of the mechanisms and subsequently find the joint reactions in the final chapter.

2.1 Representation of Rotations

There are multiple ways of representing rotations of a rigid body each with their merits and de-merits. Here, we discuss only 2 of them, namely rotation matrices and quaternions, and present the basic algebra associated with them.

To specify a general rotation of a rigid body in 3D space, we need a minimum of 3 parameters. Rotation matrices have 9 parameters, hence they have 6 redundant parameters or in other words there are six constraints among the 9 parameters. Quaternions have 4 parameters, hence only 1 redundant parameter. We consider quaternions to be a superior option over most other representations as they provide the minimum number of parameters without the problem of singularity normally associated with representations having minimal parameters such as the three Euler angles.

2.1.1 Rotation Matrices

Given a vector $\vec{v}$ in 3D Euclidean space, we can rotate the vector by applying a matrix transformation. For example, consider the vector $\vec{v}$ (yellow) in Figure 2.1. We can rotate it around X-axis by multiplying it with a rotation matrix and transform it into vector $\vec{u}$. We can write

$$\underline{R_X}\vec{v} = \vec{u}$$

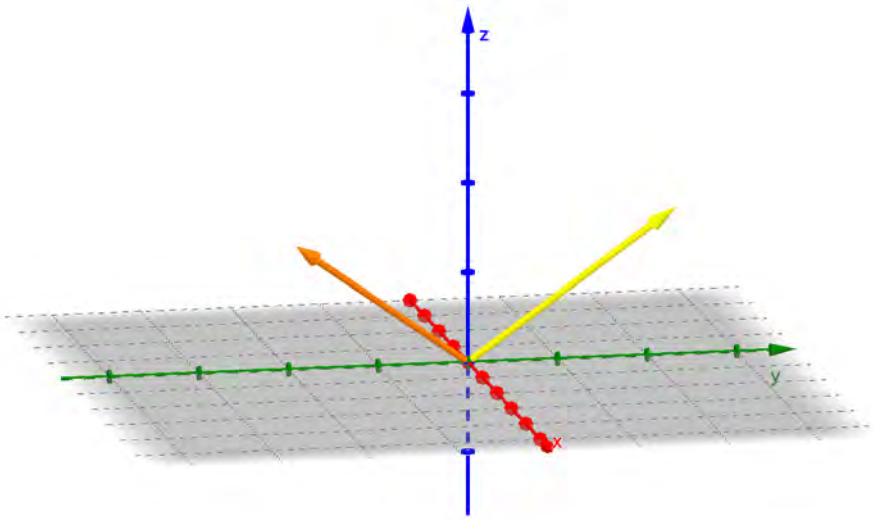


Figure 2.1: Rotation around X-Axis

The rotation matrix R_X used above can be written as

$$R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{2.1}$$

where α is the angle of rotation about the X-axis.

In the general case, we use the notation ${}^{A}R_{B}$ to denote a rotation matrix which rotates a vector in a coordinate system B to a vector in a coordinate system A. For a general translation

and rotation of a vector in 3D space, we can write

$$\vec{r}_p^{(A)} = \vec{r}_{B_o}^{(A)} + \left[{}^{A}R_B\right] \vec{r}_p^{(B)} \tag{2.2}$$

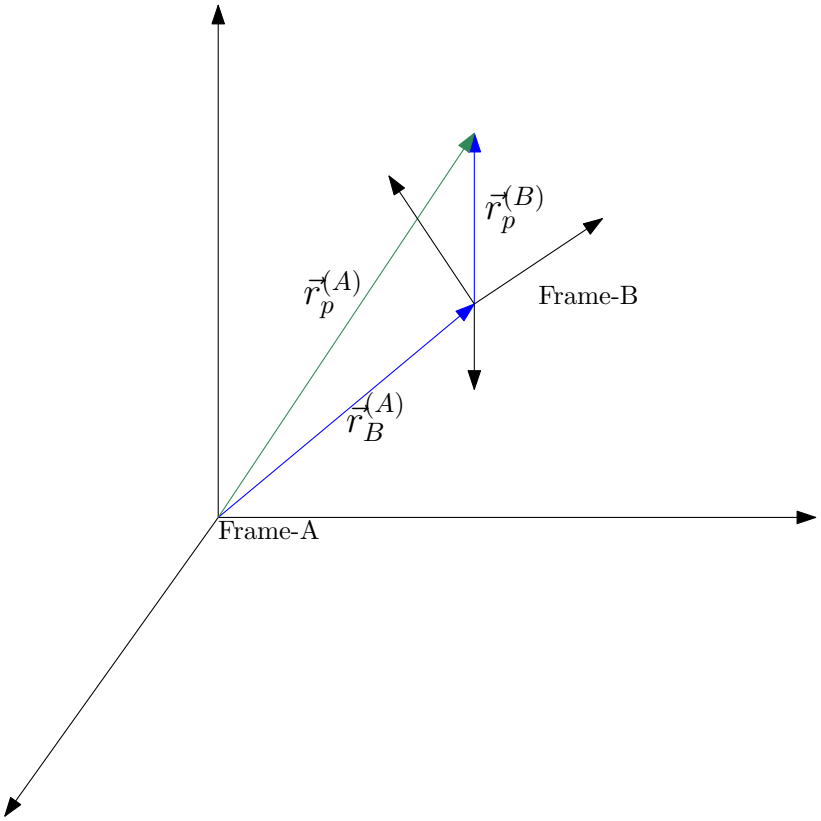


Figure 2.2: General transformation of a vector in 3D

2.1.2 Quaternions

Quaternions are a 3D counter-part of unit complex numbers in 2D. In 2D complex-plane, we can represent rotations of vectors by multiplying them by unit complex numbers. This idea was extended by Hamilton in 19th century to represent rotations in 3D space using unit quaternions. Quaternions are generally represented in the form [31] :

$$a + bi + cj + dk \tag{2.3}$$

where a, b, c, and d are real numbers and i, j and k are the fundamental quaternion units.

The following relation holds between the quaternion units:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{2.4}$$

Details about quaternions are easily available – see, for example, reference [31].

2.1.3 Composition of Rotations

Often two successive rotations of a vector is performed. In the case of matrices and quaternions, the algebra for composition is quite straightforward and uniform in both cases. If we rotate a body by multiple rotations – first by α about X-axis, then β about Y-axis, and then γ about Z-axis, then in **body-fixed axes**, the composite rotation is given by

$$\text{(Matrix Composition)} R = R_x(\alpha)R_y(\beta)R_z(\gamma) \tag{2.5}$$

$$\text{(Quaternion Composition)} \underline{Q} = \underline{Q_x}(\alpha)\underline{Q_y}(\beta)\underline{Q_z}(\gamma) \tag{2.6}$$

As can be seen in Equations (2.5) and (2.6), the composition of rotations is same for both quaternions and rotation matrices.

2.2 Equations of Motion of Multi-body Systems

In the thesis, we will be using the standard representation used in [32] and [33] to represent the dynamics and constraint equations of multi-body systems.

The equations of motion of a multi-body system can be written as

$$\mathbf{M\ddot{q} + f = Q} \tag{2.7}$$

where $\mathbf{M_{nxn}}$ is the mass/inertia matrix, $\mathbf{q_{nx1}} = (q_1, q_2, ..., q_n)^T$ is the vector of generalized coordinates, and $\mathbf{Q_{nx1}}$ is the vector of external forces and other inertia terms such as the Coriolis and centripetal terms.

A multi-body system is a constrained system. The mathematical form of its constraint equations is different for different sets of coordinates used to describe bodies in the multi-body system. In general, regardless of the coordinates used, we will represent the constraint equations

of a multi-body system as a vector containing all constraint equations. This is of the form

$$\mathbf{\Phi}(\mathbf{q}) = \begin{bmatrix} \Phi_1(\mathbf{q}) \\ \Phi_2(\mathbf{q}) \\ \vdots \\ \Phi_m(\mathbf{q}) \end{bmatrix} = \mathbf{0} \tag{2.8}$$

where, $\mathbf{\Phi}(\mathbf{q}) : R^n \rightarrow R^m$ is the vector containing all scalar constraint equations.

If some of the equations in (2.8) are dependent, it gives rise to a redundantly constrained system. It is convenient to check for this redundancy by checking the rank of the Jacobian matrix of this system. The mxn Jacobian matrix of this system can be written as

$$\mathbf{\Phi_q}(\mathbf{q}) = \begin{pmatrix} \frac{\partial \Phi_1}{\partial q_1} & \frac{\partial \Phi_1}{\partial q_2} & \cdots & \cdots \\ \frac{\partial \Phi_2}{\partial q_1} & \frac{\partial \Phi_2}{\partial q_2} & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \Phi_m}{\partial q_1} & \frac{\partial \Phi_m}{\partial q_2} & \cdots & \frac{\partial \Phi_m}{\partial q_n} \end{pmatrix} = \mathbf{0} \tag{2.9}$$

The generalised force vector $\boldsymbol{f_{n\times 1}}$ can be written as:

$$\boxed{\mathbf{f} = \mathbf{\Phi_q}^T \boldsymbol{\lambda}} \tag{2.10}$$

where $\boldsymbol{\lambda_{m\times 1}}$ is the vector of Lagrange multipliers.

Equation (2.10) is the key equation of importance in this work. This equation tells us how we can get constraint reaction forces from the given constraint equations. The reaction solvability analysis (RSA) methodology developed in Section 3.4 is based on this equation.

From equation (2.7) and (2.10), we can write

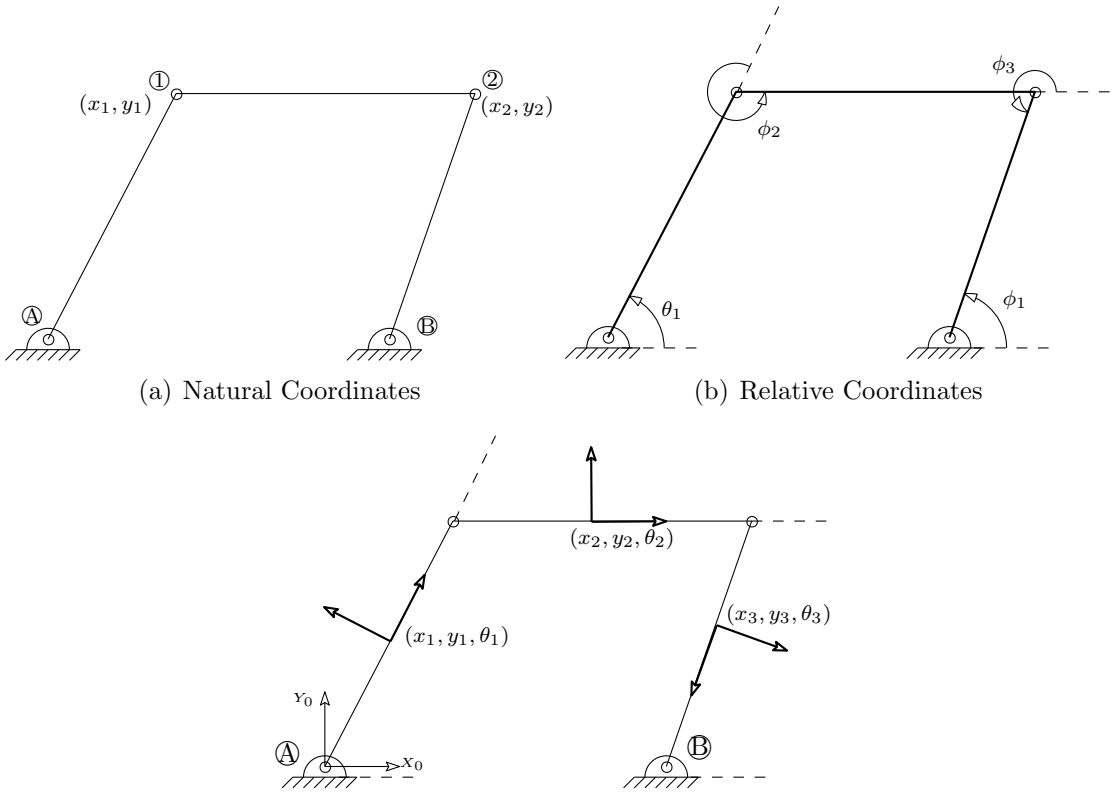
$$\mathbf{\Phi_q}^T \boldsymbol{\lambda} = \mathbf{Q} - \mathbf{M}\ddot{\mathbf{q}} = \mathbf{f} \tag{2.11}$$

The above equation (2.11) is a convenient representation in the familiar linear-algebraic form of $\boldsymbol{Ax = b}$. This equation can be investigated using tools from linear algebra in the next section we present two important propositions.

2.3 Coordinates for Kinematic Modeling

We discuss 3 types of coordinates which can be used to model the geometry of a mechanism. These are known as absolute coordinates (or reference point coordinates), natural coordinates

(or fully Cartesian coordinates), and relative coordinates (or joint coordinates). Absolute coordinates are used in multi-body packages such as Adams and DADS [33], natural coordinates were proposed by [27] and relative coordinates are popular in the field of robotics [34, 35]. We illustrate these using a planar 4-bar mechanism, as shown in Figure 2.3.



(a) Natural Coordinates

(b) Relative Coordinates

(c) Absolute Coordinates

Figure 2.3: Planar 4-Bar Modelled with different Coordinates

2.3.1 Absolute Coordinates (Reference Point Coordinates)

For modeling the 4-bar mechanism using absolute coordinates, we attach a reference frame and an origin to every link. The origin is usually attached to the center of mass of the link, although this choice is arbitrary. For planar mechanisms, this amounts to 3 coordinates per link – 2 Cartesian coordinates specifying the position of the origin and 1 angle specifying the orientation of reference frame with respect to the base frame. Figure 2.3(c) shows the planar 4-bar mechanism modeled with absolute coordinates. The absolute coordinates for the planar 4-bar are $\mathbf{q} = (x_1, y_1, \theta_1, x_2, y_2, \theta_2, x_3, y_3, \theta_3)$, i.e., we have 9 coordinates. The 4-bar mechanism is known to have one DoF, hence 8 constraint equations are required to represent the 4-bar

mechanism using absolute coordinates. The can be written as

$$(x_a - x_1) + (l_1/2)\cos(\theta_1) = 0 \tag{2.12}$$
$$(y_a - y_1) + (l_1/2)\sin(\theta_1) = 0 \tag{2.13}$$
$$(x_1 - x_2) + \frac{l_1}{2}\cos(\theta_1) + \frac{l_2}{2}\cos(\theta_2) = 0 \tag{2.14}$$
$$(y_1 - y_2) + \frac{l_1}{2}\sin(\theta_1) + \frac{l_2}{2}\sin(\theta_2) = 0 \tag{2.15}$$
$$(x_2 - x_3) + \frac{l_2}{2}\cos(\theta_2) + \frac{l_3}{2}\cos(\theta_3) = 0 \tag{2.16}$$
$$(y_2 - y_3) + \frac{l_2}{2}\sin(\theta_2) + \frac{l_3}{2}\sin(\theta_3) = 0 \tag{2.17}$$
$$(x_3 - x_b) + \frac{l_3}{2}\cos(\theta_3) = 0 \tag{2.18}$$
$$(y_3 - y_b) + \frac{l_3}{2}\sin(\theta_3) = 0 \tag{2.19}$$

For spatial mechanisms, 6 coordinates – 3 for specifying position of origin and 3 for specifying rotation (using Euler Angles) – are required. If quaternions are used to specify rotation, we have 4 coordinates for rotation, instead of 3 Euler angles. Often the use of quaternions results in significant computational efficiency. Absolute coordinates are the most natural choice when it comes to modeling multi-body systems and they have been used in many early textbooks in multi-body dynamics[32].

2.3.2 Natural Coordinates (Fully Cartesian Coordinates)

As the name suggests, fully Cartesian coordinates have only Cartesian components and no rotation components. The rotation is specified by the relative position of more than two points in 3D space. In using natural coordinates, we define several "basic" points in space for each link (rigid body) of the multi-body system. Then, constraints are defined for the rigidity of the links and for joints between links. In Figure 2.3(a), the planar 4-bar mechanism is modeled using natural coordinates. The coordinates in this case are $\mathbf{q} = (x_1, y_1, x_2, y_2)$, i.e. 4 coordinates. The mechanism has 1 DoF and hence 3 constraint equations are required. These are

$$(x_1 - x_a)^2 + (y_1 - y_a)^2 = l_1^2 \tag{2.20}$$
$$(x_1 - x_2)^2 + (y_1 - y_2)^2 = l_2^2 \tag{2.21}$$
$$(x_2 - x_3)^2 + (y_2 - y_3)^2 = l_3^2 \tag{2.22}$$

The biggest benefit that the natural coordinates offer is that all the constraint equations

are polynomials of degree 2 at the worst case. This is very helpful in developing symbolic algorithms for RSA since finding linear independence in polynomials constraint equations is easier than transcendental equations obtained for absolute or relative coordinates.

2.3.3 Relative Coordinates (Joint Coordinates)

They are the most minimal form of coordinates and very popular in modeling robots. Their use not only result in minimum number of variables and constraint equations but they are also the most intuitive. In Figure 2.3(b), the planar 4-bar mechanism is modeled using relative coordinates. The coordinates used are $\mathbf{q} = (\theta_1, \phi_1, \phi_2, \phi_3)$, i.e., with 4 coordinates. The mechanism has 1 DoF, hence 3 constraint equations are required. These constraint equations are of the form of loop-closure equations[34] and can be written as

$$l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \phi_2) + l_3 \cos(\theta_1 + \phi_2 + \phi_3) = l_0 \tag{2.23}$$

$$l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \phi_2) + l_3 \sin(\theta_1 + \phi_2 + \phi_3) = l_0 \tag{2.24}$$

$$\theta_1 + \phi_2 + \phi_2 + (\pi - \phi_1) = 4\pi \tag{2.25}$$

In this work, the use of relative coordinates is not desirable – since by their very nature, they hide the constraint reactions forces. For serial (open-loop) mechanisms, no constraint equation are required in relative coordinates and hence no constraint reactions can be found out. For parallel (close-loop) mechanisms, they offer loop-closure equations, using which the reactions of the closing constraints (in the cut joints) can be determined [26]. However, if one wishes to find constraint reactions when using joint coordinates, methods have been proposed to do the same and these are called augmented joint coordinate method [36].

2.4 Constraints and Joint Reactions

In this section, we discuss how to define constraints using different coordinate formulations and how to obtain the joint reactions based on those constraints. We show that some coordinate formulations facilitate calculating of joint reactions naturally while others do not and the choice of coordinates directly affects RSA. The use of absolute coordinates are the most facilitating when it comes to calculating joint reactions as they naturally encapsulate all the joint reactions by the very nature the constraint equations are defined using them. This is the reason why most works use absolute coordinates when it comes to calculating joint reactions. Natural coordinates facilitate joint reaction calculation for some joint types. However, we can modify the natural coordinate formulation so that they provide constraint equations for all joint reactions. Relative coordinates only allow for calculating joint reactions for closed-loop mechanisms, and do not

allow for any joint reaction calculation for open-loop mechanisms.

In the following, we will only focus on showing how different types of constraints help us get the joint reactions. For detailed information on how to model various joint constraints, one should refer [32] for constraints using absolute coordinates, reference[11] for natural coordinates and reference [34] for relative coordinates.

2.4.1 Joint Reactions from Absolute Coordinates

For planar 4-bar mechanism given in Figure 2.3(c), the constraint equations using absolute coordinates were given in Equations (2.12) – (2.19). The Jacobian matrix for these constraints can be calculated by using equation (2.9) and the constraint reactions can be obtained using the relation (2.10)

$$\mathbf{f_{8x1}} = \mathbf{\Phi_q}^T \boldsymbol{\lambda}$$

Here, all the constraint reactions encapsulated in $\mathbf{f_{8\times 1}}$ correspond to joint reactions – the first 2 elements of $\mathbf{f_{8\times 1}}$ correspond to the first joint, the next 2 elements correspond to the second joint and so on.

2.4.2 Joint Reactions from Natural Coordinates

The planar 4-bar modeled using natural coordinates, shown in Section 2.3.2, is reproduced in Figure 2.4.

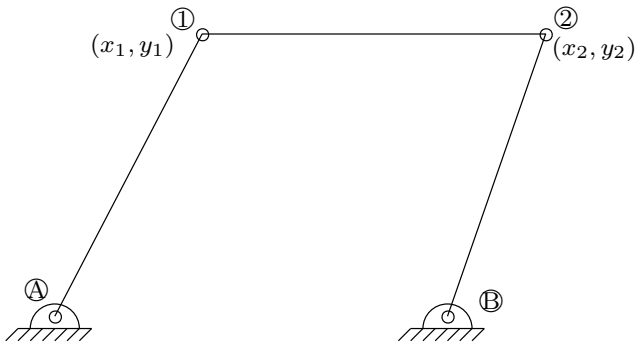


Figure 2.4: 4-Bar Modelled with Natural Coordinates

The equations for this modeling were given in equations (2.20) – (2.22) as

$$\begin{aligned}(x_1 - x_a)^2 + (y_1 - y_a)^2 &= l_1^2 \\ (x_1 - x_2)^2 + (y_1 - y_2)^2 &= l_2^2 \\ (x_2 - x_3)^2 + (y_2 - y_3)^2 &= l_3^2\end{aligned}$$

In these equations, the first corresponds to the rigid-body constraint for the first link, the second one for the rigidity of second link and so on. The revolute joints in the mechanism are captured by the sharing of the "basic" points, hence no additional constraint equation is required for them. We propose a modified formulation of natural coordinates in the next section which help us write constraint equations for other kinds of joints. This formulation requires more "basic" points.

2.4.3 Joint-Augmented Natural Coordinate Formulation

Our modified natural coordinates formulation, which we call **joint-augmented natural coordinate formulation**, is shown in Figure 2.5. Note that we have dismantled link-ends to show more clearly how the points between different links are not shared as before, rather we share them using extra constraint equations. These extra constraint equations give us the required joint reactions.

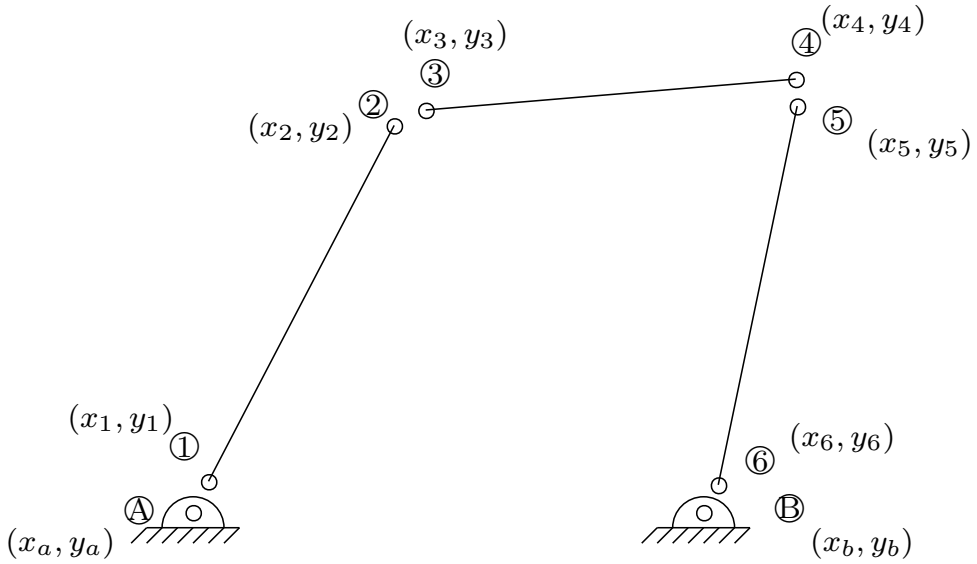


Figure 2.5: 4-Bar Modelled with Modified Natural Coordinates

Here, we have defined 2 basic points on each link, none of which is shared. The constraints equations now required lie in 2 class. These are

1. Rigid Body Constraints. These were required in the previous formulation also.
2. Joint Constraints. These were not required in the previous formulation.

The rigidity constraints are captured by the 3 equations

$$|\vec{r_i} - \vec{r_{i+1}}| = l_{i-i+1} \text{ ,where } i \in [1, 3, 5] \tag{2.26}$$

The revolute joint constraints are captured by 4 vector equations – each vector equation consists of 2 scalar equations, hence a total 8 equations – given as

$$\begin{aligned} \vec{r_A} &= \vec{r_1}, \qquad \vec{r_2} = \vec{r_3} \\ \vec{r_4} &= \vec{r_5}, \qquad \vec{r_6} = \vec{r_B} \end{aligned} \tag{2.27}$$

The coordinate vector in this formulation is $\mathbf{q} = (x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_5, y_5, x_6, y_6)$, i.e., we have 12 coordinates. We have 3 rigidity constraints (2.26) and 8 revolute joint constraints (2.27) giving rise to 1 DoF for the 4-bar mechanism as expected. We can now calculate joint reactions using constraint equations (2.27), for they correspond to the four joints in the mechanism.

2.5 Comparison of Different Coordinate Formulations

We close this chapter with a brief comparison of the different coordinate formulations that we have discussed so far.

The relative coordinates clearly offer the minimum number of coordinates and constraint equations and they are also the most intuitive. The number of coordinates in natural coordinates is more than relative coordinates, but is less than the number in absolute coordinates. When using quaternions, the number of coordinates in absolute coordinates increases further but it offers further benefits. Finally, the joint-augmented natural coordinates proposed in Section 2.4.3 have the most number of coordinates.

For our purpose of implementing reaction solvability analysis (RSA), we need formulations which provide constraint equations for joints. This is clearly not possible using relative coordinates, hence they are ruled out. Absolute coordinates are the most natural choice for RSA as all the constraints in this formulation deal with joints only. When using natural coordinates, the joint constraints are not always there. Hence, we propose a modified version of the default natural coordinate formulation – Joint-Augmented Natural Coordinate Formulation. This formulation has more number of equations than absolute coordinates, but all the equations are polynomials, and moreover, it allows the possibility of computing all joint reactions. The Joint-Augmented Natural Coordinate Formulation is worth investigating as a candidate for RSA.

Chapter 3

Methodology Development

In this chapter, we first cover some background fundamentals regarding over-constrained mechanisms. We, then, discuss how to formulate and solve the constrained dynamics equations. We also discuss how we can use them to calculate the joint reactions. Finally, we arrive at the focus of this chapter – developing methodology for Reaction Solvability Analysis (RSA) of over-constrained mechanisms.

3.1 Over-constrained Mechanisms

Over-constrained mechanisms, as mentioned in Chapter-1 Introduction, are mechanisms having finite mobility despite the DoF computed using Grübler - Kutzbach is zero or negative. This is due to the fact that some constraint reactions are dependent, i.e., some constraints are redundant. Our approach towards finding solvable joint reactions is based on finding this set of redundant constraints.

For the sake of completeness, the well-known Grübler-Kutzbach criterion is given as

$$DoF = \lambda(N - J - 1) + \Sigma_{i=1}^{J} F_i \tag{3.1}$$

where,

$$\begin{aligned} \lambda &= 3 \text{ (for planar) or } 6 \text{ (for spatial)} \\ N &= \text{total number of links} \\ J &= \text{total number of joints} \\ F_i &= \text{degree of freedom at the } i\text{th joint} \end{aligned}$$

The very first over-constrained mechanism was given by Sarrus in 19th century and probably

the most famous over-constrained mechanism was given by Bennett in 1903 in a paper titled "A New Mechanism". This mechanism is a 4-bar spatial mechanism whose DoF comes out to be -2 if calculated using Grübler-Kutzbach criterion given above. However, it can be shown that the Bennett mechanism can have finite motion and this is due to the special geometry and arrangement of the links which are not taken into account in the Grübler-Kutzbach criterion.

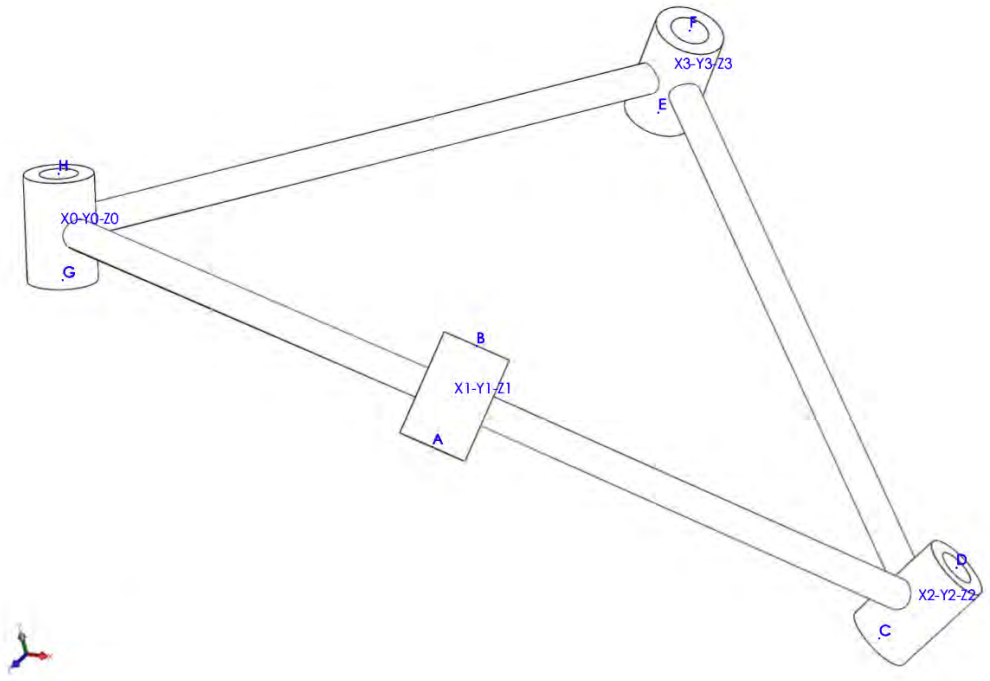


Figure 3.1: Bennett Mechanism

In the following decades, many single-loop spatial over-constrained mechanisms were developed, many of which were found to be derivatives of Bennett. A few mechanisms worth mentioning are Goldberg (5R), Myard (5R), Bricard (6R), and Schatz Linkage (6R)[37, 38, 39]. Examples of planar over-constrained mechanisms are the well-known parallelogram mechanism and the Kempe-Burmester focal mechanism[40].

3.2 Dynamics Equations Formulation

Before discussing the methodology for Reaction Solvability Analysis (RSA), we briefly discuss the nature and how to solve the equations of motion of an over-constrained mechanism. The equations of motion for an over-constrained mechanism are a set of differential-algebraic equations and we need to solve these to simulate the motion and to calculate the joint reactions.

The complete set of DAEs (Differential Algebraic Equations) can be written as

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi_q}^T\boldsymbol{\lambda} = \mathbf{Q}_e \qquad \text{: 'n' equations} \tag{3.2}$$

$$\mathbf{\Phi}(\mathbf{q}) = \mathbf{0} \qquad \text{: 'm' equations} \tag{3.3}$$

Equations (3.2) and (3.3) give the complete set of $(n+m)$ DAEs which have to be solved for the $(n+m)$ variables: $\mathbf{q}_{n\times 1}$ and $\boldsymbol{\lambda}_{m\times 1}$. It maybe mentioned (and easily seen from previous Chapter) that constraints are holonomic and hence the representation in Equation (3.3).

We can differentiate the algebraic constraint equations (3.3) to obtain acceleration constraint equations as

$$\begin{aligned}\mathbf{\Phi}\ddot{\mathbf{q}} &= -(\Phi_q\dot{\mathbf{q}})_q\dot{\mathbf{q}} - 2\Phi_{qt}\dot{\mathbf{q}} - \Phi_{tt} \\ &= \mathbf{Q}_d \qquad \Rightarrow \text{describes the terms quadratic in velocities}\end{aligned} \tag{3.4}$$

Instead of using algebraic constraint equations (3.3), we can now use acceleration constraint equations (3.4) as all our equations are now differential in nature. Combining this set of equations with (3.2), we get

$$\begin{aligned}\mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi_q}^T\boldsymbol{\lambda} &= \mathbf{Q}_e \\ \mathbf{\Phi}\ddot{\mathbf{q}} &= \mathbf{Q}_d\end{aligned} \tag{3.5}$$

where, $\mathbf{Q}_d$ describes terms like Coriolis and centripetal forces and $\mathbf{Q}_e$ describes the external generalized forces (forces and moments). Equations (3.5) can be written in a convenient matrix form as

$$\begin{bmatrix}\mathbf{M} & \mathbf{\Phi_q}^T \\ \mathbf{\Phi_q} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\ddot{\mathbf{q}} \\ \boldsymbol{\lambda}\end{bmatrix} = \begin{bmatrix}\mathbf{Q}_e \\ \mathbf{Q}_d\end{bmatrix} \tag{3.6}$$

and equation (3.6) can be manipulated to obtain explicit expressions for $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$ (see [41]) as

$$\begin{bmatrix}\ddot{\mathbf{q}} \\ \boldsymbol{\lambda}\end{bmatrix} = \begin{bmatrix}\mathbf{M} & \mathbf{\Phi_q}^T \\ \mathbf{\Phi_q} & \mathbf{0}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{Q}_e \\ \mathbf{Q}_d\end{bmatrix}$$

which gives rise to the two following equations

$$\ddot{\mathbf{q}} = \mathbf{M^{-1}Q_e} + \mathbf{M^{-1}\Phi_q^T}\left([\mathbf{\Phi_q M^{-1}\Phi_q^T}]^{-1}(\mathbf{Q_d} - \mathbf{\Phi_q M^{-1}Q_e})\right) \tag{3.7a}$$

$$\boldsymbol{\lambda} = [\mathbf{\Phi_q M^{-1}\Phi_q^T}]^{-1}\left(\mathbf{Q_d} - \mathbf{\Phi_q M^{-1}Q_e}\right) \tag{3.7b}$$

Equation (3.7) are in a convenient form to solve for $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$ and the joint reactions. Equation (3.7a) is a differential equation which can be integrated numerically, to solve for $\mathbf{q}$ and $\dot{\mathbf{q}}$. After having solved for these kinematic quantities, we can use these quantities to calculate $\boldsymbol{\lambda}$, which can then be used to calculate joint reactions given by equation (2.10).

3.2.1 Integration of ODE's

```
eqs = ∂t∂t q - (M^-1.Qe + M^-1.J^T.(Inverse[J.M^-1.J^T].(Qd - J.M^-1.Qe))) ==
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}

(*Solution is stored in 'sol' variable
sol=NDSolve[{eqs,q[0]==q0,q'[0]==q0'},
   {q,q'},{t,0,5},Method→{"EquationSimplification"→"Solve"}]

λ = Inverse[J.M^-1.J^T].(Qd - J.M^-1.Qe)

(*Solve for λ at t = 1s *)
λ /. t → 1 /. sol

(*Calculate λ for all t and Plot it *)
Table[Plot[Evaluate[λ[[j]] /. sol], {t, 0, 5}, PlotStyle → Automatic,
   AxesLabel → {Time, "Generalized Force"}, PlotLabel → fj+1], {j, 6, 11}]
```

Figure 3.2: Using NDSolve in Mathematica®

There are several methods available for solving ODEs. One fairly simple and standard method is Euler's Method. This method can only be used for first-order differential equations [42]. However, equation (3.7a) is a second-order differential equation. To use Euler's method, the second-order ODEs need to be re-formulated in standard state-space formulation with $\mathbf{q}$ and $\dot{\mathbf{q}}$ as variables. A better and more efficient way would be to use multi-step integration methods which can directly integrate our second-order equation. Runge-Kutta and Adams-Bashforth methods are a few such multi-step solvers. However, since the purpose of this work was not to compare different integration methods or improve upon them, we have directly used a robust integration method that comes in the Mathematica® package – **NDSolve**. This method deploys the best possible integration method according to the problem and hence it provides a neat abstraction. This method is depicted in Figure 3.2.

3.3 Solving Redundantly Constrained Equations

Equations (3.7b) and (2.10) can be written as

$$[\mathbf{\Phi_q M^{-1} \Phi_q^T}]\boldsymbol{\lambda} = \left(\mathbf{Q_d} - \mathbf{\Phi_q M^{-1} Q_e}\right) \tag{3.8}$$

$$\mathbf{f} = \mathbf{\Phi_q}^T \boldsymbol{\lambda} \tag{3.9}$$

As a first step, equation (3.8) is used to obtain the value of $\boldsymbol{\lambda}$, and then constraint reaction vector $\mathbf{f}$ is obtained using equation (3.9). It can be proved that for a given rectangular matrix $\mathbf{A}$ and square invertible matrix $\mathbf{B}$, $\mathbf{ABA^T}$ is invertible only if $\mathbf{A}$ has full row rank [43]. From equation (3.8), it can be seen that $[\mathbf{\Phi_q M^{-1} \Phi_q^T}]$ will be invertible (and hence $\boldsymbol{\lambda}$ will have a "unique" solution) only if $\mathbf{\Phi_q}$ (i.e., the Jacobian) has full rank. For the singular case, infinite solutions will exist for $\boldsymbol{\lambda}$ and the vector $\mathbf{f}$ obtained from equation (3.9) will also not be unique. To solve for such a singular system, two approaches are usually employed:

1. One is to eliminate all the redundant constraints (or dependent rows of the Jacobian). This is done by segregating the Jacobian matrix into a set of dependent and independent rows, and then removing the dependent rows. This segregation is not unique since if a constraint A depends on another constraint B, then B can also be seen as depending upon A and one can remove either A or B.

2. Another approach is using the pseudo-inverse of a matrix. This has not been used in this work and will not be discussed here.

When taking the elimination approach, we are essentially setting a few $\lambda's$ to be zero. Since, elimination is not unique, the set of $\lambda's$ which are 0 is also not unique. Hence, the reaction forces are not unique. However, it can be shown [25] that despite the fact that $\boldsymbol{f}$ can not be found uniquely, some components of $\boldsymbol{f}$ can be found uniquely. This happens when some of the rows of Φ_q are absolutely independent, i.e., they can not be written as the linear combination of any other row in any possible way.

We shall state this fact as a proposition:

Proposition 3.3.1 (Constraint Reactions) *If some of the constraints in rigid multi-body are absolutely independent, then the reactions corresponding to those constraints can be found out uniquely. [25]*

Now, the resultant force on a joint is because of various constraints, and not just one constraint. Resultant force is something which is of more importance, rather than the individual

contributions of constraints. Regarding the resultant force on a joint, an interesting observation can be made, which we state here as another proposition:

Proposition 3.3.2 (Joint Reactions) *The resultant reaction on a joint can be found uniquely in the following cases [25] :*

1. *if all the constraints acting on the joint are independent, or*

2. *if the constraints depend only on other constraints which are acting on the same joint*

It is important to understand the difference between the 2 propositions. In Proposition 3.3.1, we deal only with individual contributions of constraints. In Proposition 3.3.2, we are dealing with net contributions of constraints on a particular joint. Clearly, Proposition 3.3.2 is of more interest to us. These propositions can be made more clear using a simple example.

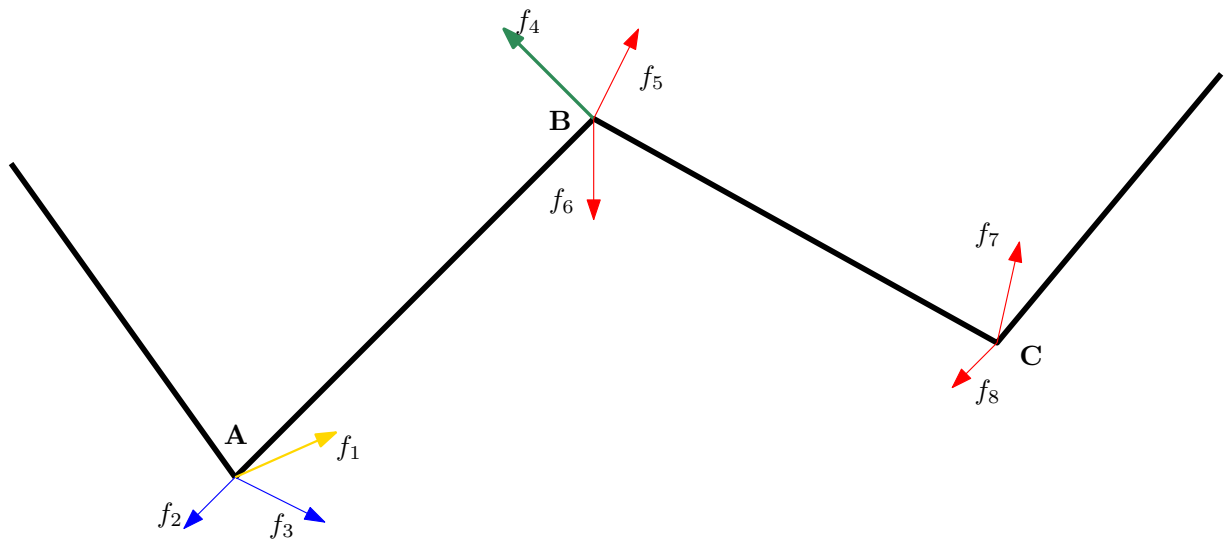


Figure 3.3: Proposition 1 and 2 Explained

In figure 3.3, constraint reactions of same color are dependent. In joint **A**, f_1 is absolutely independent and f_2 and f_3 are dependent on only each other (and they both are acting on same joint). From proposition 3.3.1, f_1 can be uniquely determined, while f_2 and f_3 can not be. However, from proposition 3.3.2, since all the constraints on joint **A** are either independent or dependent only on other constraints on joint **A**, resultant joint reaction on **A** is uniquely determined. Using similar logic, we can see that joint reactions on joints **B** and **C** are not uniquely determined, while the constraint reaction f_4 on **B** can be uniquely determined.

3.4 Development of RSA Methodology

In this section, we develop 2 methodologies. The first methodology deals with Proposition 3.3.1, i.e., finding out which constraint reactions can be found out uniquely. Second methodology deals with Proposition 3.3.2, i.e., finding out which joint reactions can be found out uniquely.

3.4.1 Methodology 1

According to Proposition 3.3.1, finding out which constraint reactions can be found uniquely is straightforward. We simply need to find the rows of the Jacobian $\mathbf{\Phi_q}$ which are independent. A basic algorithm for this is to compare the rank of the Jacobian matrix $\mathbf{\Phi_q}$ with a reduced matrix $\mathbf{\Phi_q^{\setminus i}}$, whose ith row (corresponding to ith constraint) has been crossed out. If the rank of both matrices are same, it means removing ith constraint had no effect, hence it was redundant/dependent.

3.4.2 Methodology 2

From Proposition 3.3.2, to affirm that a joint reaction can be found out uniquely, we need to make sure that all the constraints acting on it are either independent or dependent only on the constraints of the same joint. For this, first of all, we can split the Jacobian matrix $\mathbf{\Phi_q}$ into two matrices – one, which contains constraints acting on a particular joint and other which contains constraints **not** acting on that particular joint. To analyze the joint **i**, we can get two matrices for that joint as

$$\mathbf{\Phi_q^i} : \text{Matrix with all the constraints (rows) acting on joint } \mathbf{i}$$
$$\mathbf{\Phi_q^{-i}} : \text{Matrix with all the constraints (rows) not acting on joint } \mathbf{i}$$

We can now find the ranks of the corresponding matrices as

$$r_i : \text{Rank of Matrix } \mathbf{\Phi_q^i}$$
$$r_{-i} : \text{Rank of Matrix } \mathbf{\Phi_q^{-i}}$$

If all of the constraints corresponding to ith joint are either independent or dependent only among themselves, then the following relation should hold:

$$\boxed{r = r_i + r_{-i}} \tag{3.10}$$

This can be proved rigorously using the concept of direct sum from Linear Algebra[9]. The

relation (3.10) is the basis of our methodology for our algorithm to perform RSA, which can be stated conveniently as:

$$r = r_i + r_{-i} \implies \text{Joint Reaction can be found uniquely}$$
$$r \neq r_i + r_{-i} \implies \text{Joint Reaction can NOT be found uniquely}$$

3.5 Comparison of RSA Implementation using Natural and Absolute Coordinates

Now, we will discuss how the methodology can be implemented using Natural Coordinates, and what benefits it offers over Absolute Coordinates. We have already discussed in Section 2.4.3 how Natural Coordinates in their default form are not suitable for RSA, for they do not capture the joint reactions in their constraint equations. We then modified the formulation by adding extra node points to the formulation, which captured the joint reactions. This modified formulation was termed as "Joint Augmented Natural Coordinates".

In this section, we will show how the constraint equations turn out to be maximally quadratic in the case of these modified Natural Coordinates. This in turn leads to very easy to handle Jacobian matrices, which leads to faster RSA implementation. This is the main contribution of this thesis.

Let us take the case of the 4-bar mechanism that we took in Section 2.3. The constraint equations for Absolute Coordinates came out to be transcendental, as depicted in Equations (2.12) - (2.19). The Jacobian turns out to be:

$$\mathbf{\Phi_q} = \begin{pmatrix} -1 & 0 & -\frac{1}{2}\sin(\theta_1(t)) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2}\cos(\theta_1(t)) & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -\frac{1}{2}\sin(\theta_1(t)) & -1 & 0 & -\frac{1}{2}\sin(\theta_2(t)) & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2}\cos(\theta_1(t)) & 0 & 0 & \frac{1}{2}\cos(\theta_2(t)) & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -\frac{1}{2}\sin(\theta_2(t)) & -1 & 0 & -\frac{1}{2}\sin(\theta_3(t)) \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}\cos(\theta_2(t)) & 0 & 0 & \frac{1}{2}\cos(\theta_3(t)) \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{1}{2}\sin(\theta_3(t)) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2}\cos(\theta_3(t)) \end{pmatrix} \tag{3.11}$$

The constraint equations using the Modified Natural Coordinates for the same mechanism are as given in Equations (2.20) - (2.22) and Equation (2.27). The Jacobian for these set of

constraints turn out to be:

$$\Phi_q = \begin{pmatrix} 2(x_1 - x_a) & 2(y_1 - y_a) & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \\ 2(x_1 - x_a) & 2(y_1 - y_a) & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \\ 2(x_1 - x_2) & 2(y_1 - y_2) & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdots & -2(y_2 - y_3) & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdots & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.12}$$

Our RSA methodology as proposed in Section 3.4.2 is based on calculating the ranks of the Jacobian and its components. Now, the Jacobian matrix using our proposed Natural Coordinates is a very simple one, with only linear components. Hence its rank can be found even symbolically. While the Jacobian matrix generated using Absolute Coordinates has transcendental components and hence will be difficult to evaluate symbolically.

3.6 Summary

In this chapter, we first explained over-constrained mechanisms with a few examples in Section 3.1. Dynamics Equations were then formulated in Section 3.2, and methods of integrating them discussed. With dynamics equations in hand, we then discuss the strategies to solve redundantly constrained systems in Section 3.3. Here, we give an important proposition 3.3.2 which tells us when the resultant joint reactions on any joint can be found uniquely. The RSA methodologies are then developed in Section 3.4 using this proposition. RSA Methodology-2 developed in sub-section 3.4.2 is the primary methodology used in the following chapter.

Chapter 4

Implementation of Reaction Solvability Analysis

In this chapter, we model a few over-constrained mechanisms of interest using appropriate coordinate formulation(s), and write their constraint equations. Then the reaction solvability analysis is done on them using our defined Methodology-2 in 3.4.2.

4.1 Parallelogram Mechanism

We start with the kinematic modeling of this well-known planar mechanism shown in Figure 4.1. As shown the mechanism consists of 5 links and 6 joints.

4.1.1 Kinematic Modeling (Absolute Coordinates)

The degrees of freedom using Grübler-Kutzbach criterionis

$$DoF = 3\left(N - J - 1\right) + \sum F_i = 3(5 - 6 - 1) + 6 = 0$$

and since it can move, it is an over-constrained mechanism. For its kinematic modeling, using absolute coordinates, the pose of each link is denoted by 3 numbers – $[x_i, y_i, \theta_i]$ resulting in coordinates given by

$$\vec{q} = \left[\vec{r}_1^T \theta_1 \vec{r}_2^T \theta_2 \vec{r}_3^T \theta_3 \vec{r}_4^T \theta_4\right]^T \qquad \vec{r}_i = [\vec{x}_i \vec{y}_i]^T$$
$$\vec{q} = [x_1, y_1, \theta_1, x_2, y_2, \theta_2, x_3, y_3, \theta_3, x_4, y_4, \theta_4,]$$

Let $\vec{s}_a^{(i)}$, $\vec{s}_b^{(i)}$, $\vec{s}_p^{(i)}$ and $\vec{s}_q^{(i)}$ be coordinates of a, b, p, and q respectively in local frame i. The constraint equations based on these absolute coordinates are

$$
\begin{aligned}
\Phi^1(\vec{q}) &= \begin{bmatrix} \phi_1(\vec{q}) \\ \phi_2(\vec{q}) \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \vec{0} \\
\Phi^2(\vec{q}) &= \begin{bmatrix} \phi_3(\vec{q}) \\ \phi_4(\vec{q}) \end{bmatrix} = \vec{r}_1 + \underline{R}_1 \vec{s}_p^1 - \vec{r}_4 - \underline{R}_4 \vec{s}_p^4 = \vec{0} \\
&= \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{bmatrix} \begin{bmatrix} l/2 \\ 0 \end{bmatrix} - \begin{bmatrix} x_4 \\ y_4 \end{bmatrix} - \begin{bmatrix} c_4 & -s_4 \\ s_4 & c_4 \end{bmatrix} \begin{bmatrix} -l/2 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} x_1 - x_4 \\ y_1 - y_4 \end{bmatrix} - \begin{bmatrix} c_1 + c_4 & -(s_1 + s_4) \\ s_1 + s_4 & c_1 + c_4 \end{bmatrix} \begin{bmatrix} l/2 \\ 0 \end{bmatrix} = \vec{0} \\
\Phi^3(\vec{q}) &= \begin{bmatrix} \phi_5(\vec{q}) \\ \phi_6(\vec{q}) \end{bmatrix} = \vec{r}_1 + \underline{R}_1 \vec{s}_a^1 - \vec{r}_2 - \underline{R}_2 \vec{s}_a^4 = 0 \\
&= \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{bmatrix} \begin{bmatrix} l \\ 0 \end{bmatrix} - \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} - \begin{bmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{bmatrix} \begin{bmatrix} -l/2 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} x_1 - x_4 \\ y_1 - y_4 \end{bmatrix} - \begin{bmatrix} 2c_1 + c_2 & -(2s_1 + s_2) \\ 2s_1 + s_2 & 2c_1 + c_2 \end{bmatrix} \begin{bmatrix} l/2 \\ 0 \end{bmatrix} = \vec{0} \\
\Phi^4(\vec{q}) &= \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} + \begin{bmatrix} c_3 & -s_3 \\ s_3 & c_3 \end{bmatrix} \begin{bmatrix} l \\ 0 \end{bmatrix} - \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} - \begin{bmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{bmatrix} \begin{bmatrix} l/2 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} x_3 - x_2 \\ y_3 - y_2 \end{bmatrix} - \begin{bmatrix} 2c_3 + c_2 & -(2s_3 + s_2) \\ 2s_3 + s_2 & 2c_3 + c_2 \end{bmatrix} \begin{bmatrix} l/2 \\ 0 \end{bmatrix} \\
\Phi^5(\vec{q}) &= \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} + \begin{bmatrix} c_3 & -s_3 \\ s_3 & c_3 \end{bmatrix} \begin{bmatrix} l/2 \\ 0 \end{bmatrix} - \begin{bmatrix} x_4 \\ y_4 \end{bmatrix} - \begin{bmatrix} c_4 & -s_4 \\ s_4 & c_4 \end{bmatrix} \begin{bmatrix} l/2 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} x_3 - x_4 \\ y_3 - y_4 \end{bmatrix} - \begin{bmatrix} c_3 - c_4 & -(s_3 - s_4) \\ s_3 - s_4 & c_3 - c_4 \end{bmatrix} \begin{bmatrix} l/2 \\ 0 \end{bmatrix} \\
\Phi^6(\vec{q}) &= \begin{bmatrix} \phi_{11}(\vec{q}) \\ \phi_{12}(\vec{q}) \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} l \\ 0 \end{bmatrix}
\end{aligned}
$$

Consider that the mechanism is at the position such that $\theta_1 = \theta_3 = \pi/2$ and $\theta_2 = \theta_4 = 0$.

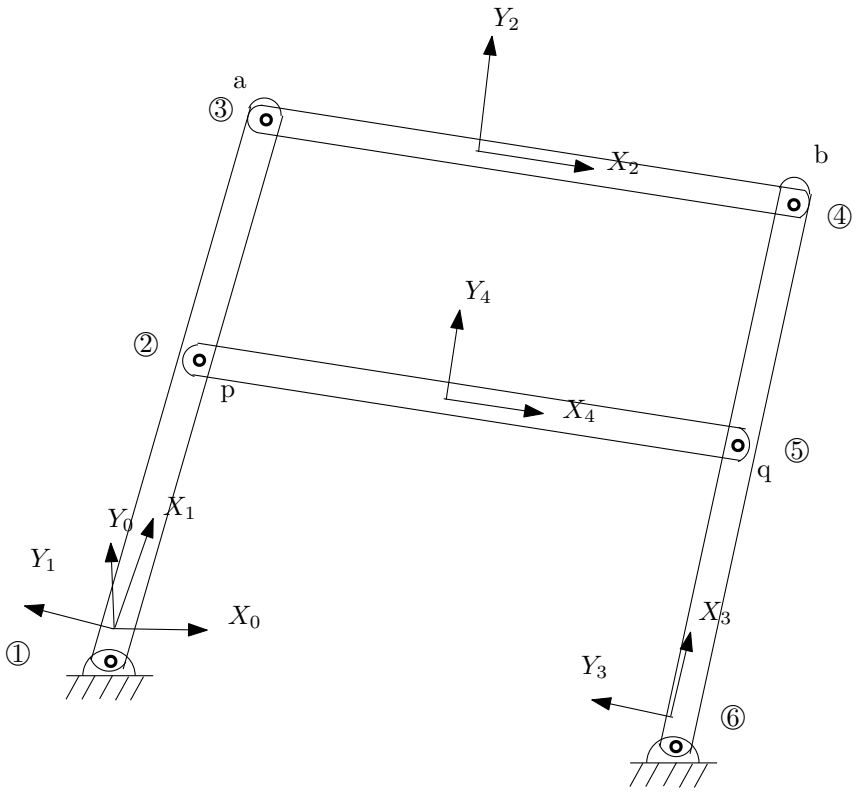


Figure 4.1: Parallelogram Mechanism

The Jacobian at this representative position can be calculated as:

$$\Phi_q = \begin{bmatrix} \Phi_q^1 \\ \Phi_q^2 \\ \vdots \\ \Phi_q^6 \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} (\phi_1)_q \\ (\phi_2)_q \\ \vdots \\ (\phi_{12})_q \end{pmatrix} \end{bmatrix}$$

$$\Phi_q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -l/2 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & l/2 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & -l & -1 & 0 & l/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -l/2 & 1 & 0 & -l & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -l/2 & -1 & 0 & -l/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4.1.2 Apply RSA Methodology-2

Now, let us apply our RSA Methodology-2 developed in 3.4.2, which says that unique reactions can be found only for joints for which $r_i + r_{-i} = r$.

The rank **r** of Φ_q is 10. The ranks r_i and r_{-i} of the matrices Φ_q^i's and Φ_q^{-i}'s are given in table 4.1. For all joints, we can see that $r_i + r_{-i} > r$, where $r = 10$. Hence, for none of the joints in this over-constrained parallelogram mechanism we can obtain unique reactions.

i	$r_i = rank(\Phi_q^i)$	$r_{-i} = rank(\Phi_q^{-i})$	$r_i + r_{-i}$
1	2	9	11
2	2	9	11
3	2	9	11
4	2	9	11
5	2	9	11
6	2	9	11

Table 4.1: Methodology-2 Results for Parallelogram Mechanism

4.2 Kempe Burmester Mechanism

The Kempe-Burmester is a well known over-constrained mechanism[40] and we use our RSA methodology to obtain which joint reactions, if any, can be computed.

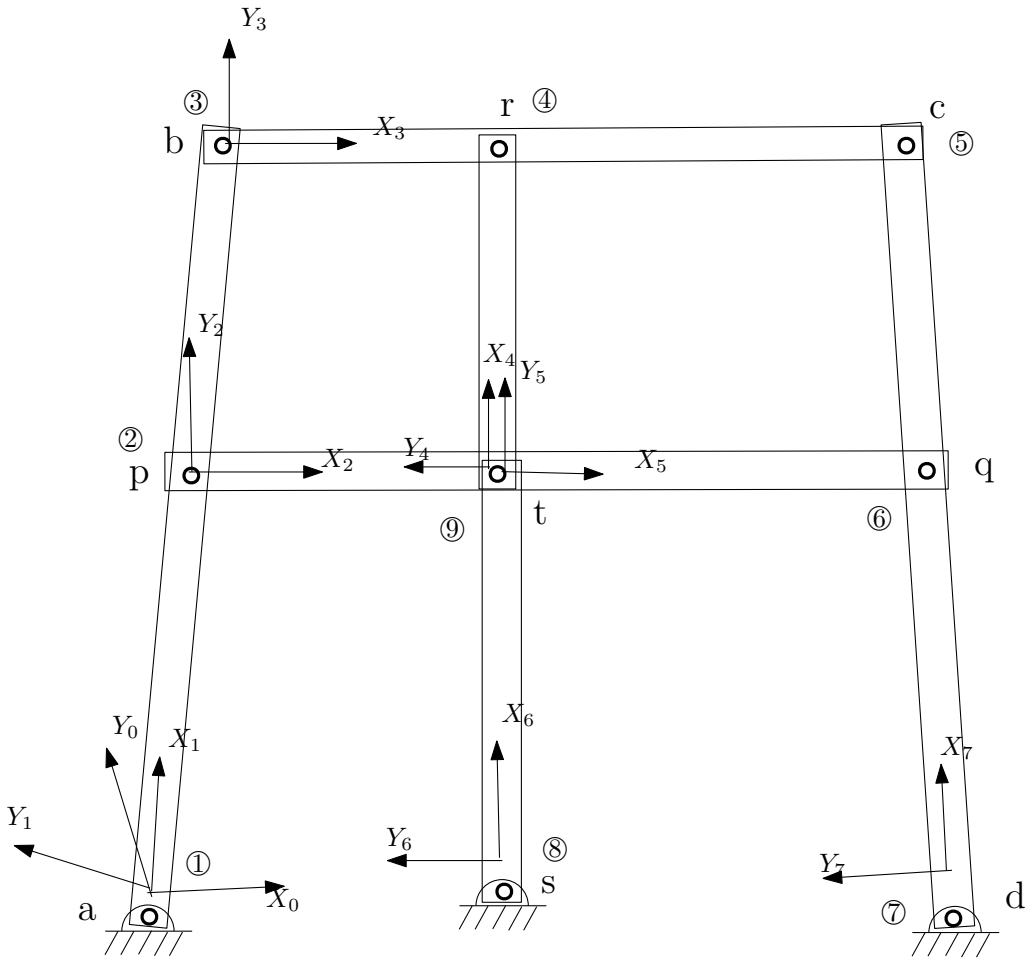


Figure 4.2: Kempe Burmester Mechanism

4.2.1 Kinematic Modeling (Absolute Coordinates)

The Kempe-Burmester mechanism is known to have the following special characteristics.

Points p, q, r, s lie in a circle.

$$\frac{l_{bp}}{l_{ap}} = \frac{l_{cq}}{l_{dq}}$$

$$\frac{l_{br}}{l_{cr}} = \frac{l_{as}}{l_{ds}}$$

Following in the same vein as the Parallelogram Mechanism, we model Kempe-Burmester using absolute coordinates as depicted in Figure 4.2. The constraint equations for this mecha-

nism are:

$$\Phi^1(\vec{q}) = \begin{bmatrix}\phi_1(\vec{q})\\ \phi_2(\vec{q})\end{bmatrix} = \begin{bmatrix}x_1\\ y_1\end{bmatrix} = \vec{0}$$

$$\Phi^2(\vec{q}) = \begin{bmatrix}\phi_3(\vec{q})\\ \phi_4(\vec{q})\end{bmatrix} = \vec{r}_2 + \underline{R}_2\vec{s}_p^{\,2} - \vec{r}_1 - \underline{R}_1\vec{s}_p^{\,1} = \vec{0}$$

$$= \begin{bmatrix}x_2 - x_1\\ y_2 - y_1\end{bmatrix} - \begin{bmatrix}c_1\\ s_1\end{bmatrix} l_{ap}$$

$$\Phi^3(\vec{q}) = \begin{bmatrix}\phi_5(\vec{q})\\ \phi_6(\vec{q})\end{bmatrix} = \vec{r}_3 - \vec{r}_1 - \underline{R}_1\vec{s}_b^{\,1} = \vec{0}$$

$$= \begin{bmatrix}x_3 - x_1\\ y_3 - y_1\end{bmatrix} - \begin{bmatrix}c_1\\ s_1\end{bmatrix} l_{ab}$$

$$\Phi^4(\vec{q}) = \begin{bmatrix}\phi_7(\vec{q})\\ \phi_8(\vec{q})\end{bmatrix} = \vec{r}_3 + \underline{R}_3\vec{s}_r^{\,3} - \vec{r}_4 - \underline{R}_4\vec{s}_r^{\,4} = \vec{0}$$

$$= \begin{bmatrix}x_3 - x_4\\ y_3 - y_4\end{bmatrix} - \begin{bmatrix}c_3\\ s_3\end{bmatrix} l_{br} - \begin{bmatrix}c_4\\ s_4\end{bmatrix} l_{tr} = \vec{0}$$

$$\Phi^5(\vec{q}) = \begin{bmatrix}\phi_9(\vec{q})\\ \phi_{10}(\vec{q})\end{bmatrix} = \vec{r}_3 + \underline{R}_3\vec{s}_c^{\,3} - \vec{r}_7 - \underline{R}_7\vec{s}_c^{\,7} = \vec{0}$$

$$= \begin{bmatrix}x_3 - x_7\\ y_3 - y_7\end{bmatrix} - \begin{bmatrix}c_3\\ s_3\end{bmatrix} l_{bc} - \begin{bmatrix}c_7\\ s_7\end{bmatrix} l_{dc} = \vec{0}$$

$$\Phi^6(\vec{q}) = \begin{bmatrix}\phi_{11}(\vec{q})\\ \phi_{12}(\vec{q})\end{bmatrix} = \vec{r}_5 + \underline{R}_5\vec{s}_q^{\,5} - \vec{r}_7 - \underline{R}_7\vec{s}_q^{\,7} = \vec{0}$$

$$= \begin{bmatrix}x_5 - x_7\\ y_5 - y_7\end{bmatrix} - \begin{bmatrix}c_5\\ s_5\end{bmatrix} l_{tq} - \begin{bmatrix}c_7\\ s_7\end{bmatrix} l_{qd} = \vec{0}$$

$$\Phi^7(\vec{q}) = \begin{bmatrix}\phi_{13}(\vec{q})\\ \phi_{14}(\vec{q})\end{bmatrix} = \begin{bmatrix}x_7\\ y_7\end{bmatrix} = \begin{bmatrix}l_{ad}\\ 0\end{bmatrix}$$

$$\Phi^8(\vec{q}) = \begin{bmatrix}\phi_{15}(\vec{q})\\ \phi_{16}(\vec{q})\end{bmatrix} = \begin{bmatrix}x_6\\ y_6\end{bmatrix} = \begin{bmatrix}l_{as}\\ 0\end{bmatrix}$$

$$\Phi^9(\vec{q}) = \begin{bmatrix}\phi_{17}(\vec{q})\\ \phi_{18}(\vec{q})\end{bmatrix} = \begin{bmatrix}x_4 - x_5\\ y_4 - y_5\end{bmatrix} = \begin{bmatrix}0\\ 0\end{bmatrix}$$

$$\Phi^{10}(\vec{q}) = \begin{bmatrix}\phi_{19}(\vec{q})\\ \phi_{20}(\vec{q})\end{bmatrix} = \vec{r}_5 - \vec{r}_6 - \underline{R}_6\vec{s}_t^{\,6} = \vec{0}$$

$$= \begin{bmatrix} x_5 - x_6 \\ y_5 - y_6 \end{bmatrix} - \begin{bmatrix} c_6 \\ s_6 \end{bmatrix} l_{ts} = \vec{0}$$

$$\Phi^{11}(\vec{q}) = \begin{bmatrix} \phi_{21}(\vec{q}) \\ \phi_{22}(\vec{q}) \end{bmatrix} = \vec{r}_4 - \vec{r}_2 - \underline{R}_2 \vec{s}_t^2 = \vec{0}$$

$$= \begin{bmatrix} x_4 - x_2 \\ y_4 - y_2 \end{bmatrix} - \begin{bmatrix} c_2 \\ s_2 \end{bmatrix} l_{pt}$$

The Jacobian of this mechanism was calculated using **Mathematica®** and the results of the same are provided in Appendix A.1

4.2.2 Applying RSA Methodology-2

The RSA Methodology-2 was applied on this mechanism as can be seen in the Mathematica® Notebook of Kempe Burmester in Appendix A.1. We will directly show and discuss those results.

First of all, rank of Jacobian Matrix is $r = rank(\Phi_q) = 21$ and the ranks r_i and r_{-i} are shown in the table.

i	$r_i = rank(\Phi_q^i)$	$r_{-i} = rank(\Phi_q^{-i})$	$r_i + r_{-i}$
1	2	20	22
2	2	20	22
3	2	20	22
4	2	20	22
5	2	20	22
6	2	20	22
7	2	19	21
8	2	20	22
9	6	16	22

Table 4.2: Methodology-2 Results for Kempe-Burmester Mechanism

From these results, we can see that only for joint ⑦, $r_i + r_{-i} = r$. Hence, for joint ⑦, we can find the reactions uniquely. For all other joints, unique joint reaction solution is not possible.

4.3 Bennett Mechanism (Absolute Coordinates)

In this section, we model the Bennett Mechanism using absolute coordinates and apply the developed RSA methodology. Using absolute coordinates, it was not possible to implement this algorithm symbolically, hence we used a particular configuration of the Bennett Mechanism to solve it numerically.

4.3.1 Kinematic Modeling

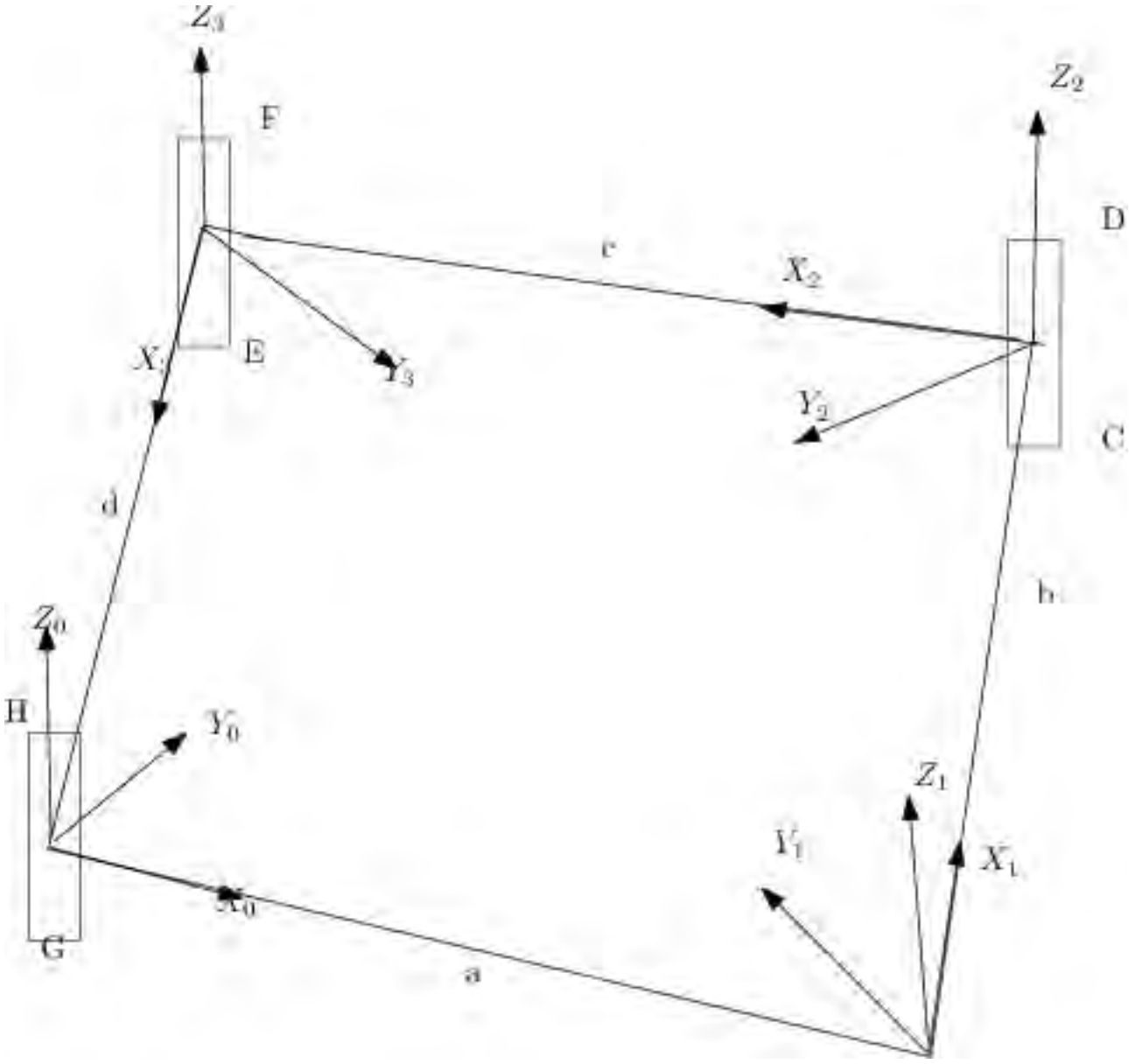


Figure 4.3: Bennett Mechanism

The position coordinates of various defined points are given as

$$\underline{S}_A^{(0)} = \begin{bmatrix} a \\ -l\cos\theta_1 \\ -l\sin\theta_1 \end{bmatrix} \qquad \underline{S}_A^{(1)} = \begin{bmatrix} 0 \\ 0 \\ -l \end{bmatrix}$$

$$\underline{S}_B^{(0)} = \begin{bmatrix} a \\ l\cos\theta_1 \\ l\sin\theta_1 \end{bmatrix} \qquad \underline{S}_B^{(1)} = \begin{bmatrix} 0 \\ 0 \\ l \end{bmatrix}$$

$$\underline{S}_C^{(1)} = \begin{bmatrix} b \\ -l\cos\theta_2 \\ -l\sin\theta_2 \end{bmatrix} \qquad \underline{S}_C^{(2)} = \begin{bmatrix} 0 \\ 0 \\ -l \end{bmatrix}$$

$$\underline{S}_D^{(1)} = \begin{bmatrix} a \\ l\cos\theta_2 \\ l\sin\theta_2 \end{bmatrix} \qquad \underline{S}_D^{(2)} = \begin{bmatrix} 0 \\ 0 \\ l \end{bmatrix}$$

$$\underline{S}_E^{(2)} = \begin{bmatrix} c & -l\cos\theta_3 & -l\sin\theta_3 \end{bmatrix}^T \qquad \underline{S}_E^{(3)} = \begin{bmatrix} 0 & 0 & -l \end{bmatrix}^T$$

$$\underline{S}_F^{(2)} = \begin{bmatrix} c & l\cos\theta_3 & l\sin\theta_3 \end{bmatrix}^T \qquad \underline{S}_F^{(3)} = \begin{bmatrix} 0 & 0 & l \end{bmatrix}^T$$

$$\underline{S}_G^{(3)} = \begin{bmatrix} d & -l\cos\theta_4 & -l\sin\theta_4 \end{bmatrix}^T \qquad \underline{S}_G^{(0)} = \begin{bmatrix} 0 & 0 & -l \end{bmatrix}^T$$

$$\underline{S}_H^{(3)} = \begin{bmatrix} d & l\cos\theta_4 & l\sin\theta_4 \end{bmatrix}^T \qquad \underline{S}_H^{(0)} = \begin{bmatrix} 0 & 0 & l \end{bmatrix}^T$$

The constraint equations are givens as:

$$\begin{aligned}
\vec{r}_1 + \underline{R}_1\vec{S}_A^{(1)} - \vec{S}_A^{(0)} &= \vec{0} \\
\vec{r}_1 + \underline{R}_1\vec{S}_B^{(1)} - \vec{S}_B^{(0)} &= \vec{0} \\
\vec{r}_2 + \underline{R}_2\vec{S}_C^{(2)} - \vec{r}_1 - \underline{R}_1\vec{S}_C^{(1)} &= \vec{0} \\
\vec{r}_2 + \underline{R}_2\vec{S}_D^{(2)} - \vec{r}_1 - \underline{R}_1\vec{S}_D^{(1)} &= \vec{0} \\
\vec{r}_3 + \underline{R}_3\vec{S}_E^{(3)} - \vec{r}_2 - \underline{R}_2\vec{S}_E^{(2)} &= \vec{0} \\
\vec{r}_3 + \underline{R}_3\vec{S}_F^{(3)} - \vec{r}_2 - \underline{R}_2\vec{S}_F^{(2)} &= \vec{0} \\
\vec{S}_G^{(0)} - \vec{r}_3 - \underline{R}_3\vec{S}_G^{(3)} &= \vec{0} \\
\vec{S}_H^{(0)} - \vec{r}_3 - \underline{R}_3\vec{S}_H^{(3)} &= \vec{0}
\end{aligned}$$

We have used the Z-X-Z Euler Angles (Body-fixed) to construct the rotation matrices and the following configuration of the Bennett Mechanism was used for computations.

$$\begin{aligned}
\{\alpha_1, \beta_1, \gamma_1\} &= \frac{1}{180}\pi\{240, 20, 0\}; \\
\{\alpha_2, \beta_2, \gamma_2\} &= \frac{1}{180}\pi\{148.4, 34.46, 31.57\};
\end{aligned}$$

$$\{\alpha_3, \beta_3, \gamma_3\} = \frac{1}{180}\pi\{0, 20, 63.13\};$$
$$\{x_1, y_1, z_1\} = \{10, 0, 0\};$$
$$\{x_2, y_2, z_2\} = \{5, 8.138, -2.962\};$$
$$\{x_3, y_3, z_3\} = \{-4.519, 8.921, 0\};$$

These values were obtained by solving the initial-position problem of Bennett Mechanism, i.e., by solving the loop-closure equations, and this was done using quaternions as explained in Appendix B.1.

4.3.2 Applying RSA Methodology-2

The RSA Methodology-2 was applied on this mechanism as can be seen in the Mathematica® Notebook of Bennett Mechanism shown in Appendix A.2. The main results are that the rank of Jacobian matrix is $r = rank(\Phi_q) = 18$ and the ranks r_i and r_{-i} are shown in the table.

i	$r_i = rank(\Phi_q^i)$	$r_{-i} = rank(\Phi_q^{-i})$	$r_i + r_{-i}$
1	5	17	22
2	6	16	22
3	6	16	22
4	5	15	20

Table 4.3: Methodology-2 Results for Bennett Mechanism

As we can see in the results, for all the joints, $r_i + r_{-i} > r = 18$ and hnce, for none of the joints the reactions can be found uniquely.

4.4 Bennett Mechanism (Natural Coordinates)

In this section, we implement RSA on the Bennett Mechanism modeled using the natural coordinates. As we will see, using the natural coordinates, the equations are simpler and amenable for symbolic computations unlike when using absolute coordinates in the previous section 4.3.

4.4.1 Kinematic Modeling

To model this spatial mechanism using natural coordinates, we use 8 basic points and 8 unit vectors. The constraint equations are as follows:

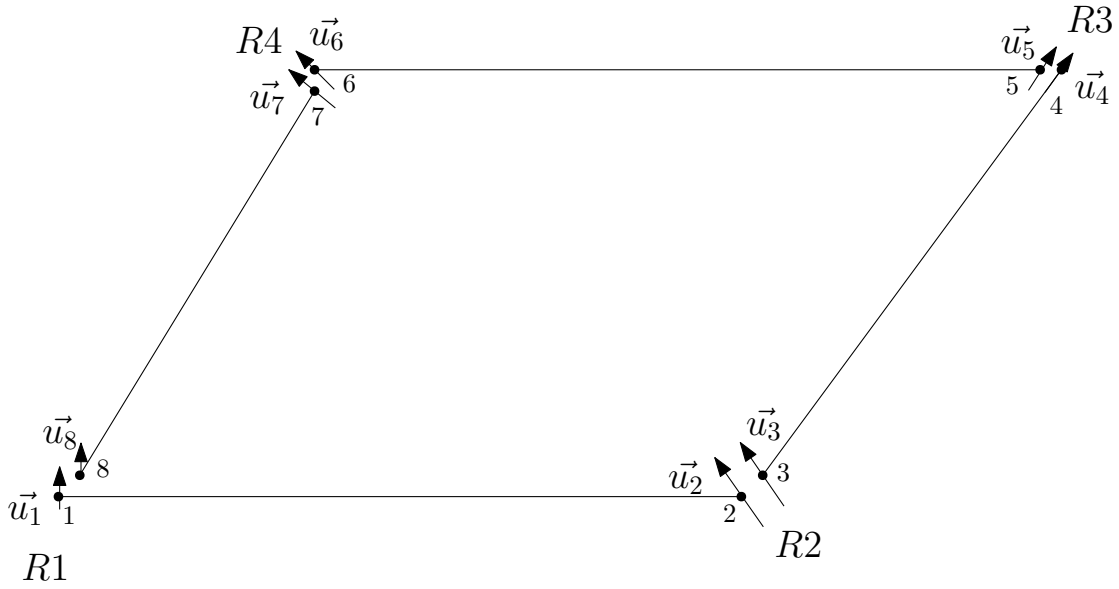


Figure 4.4: Bennett Mechanism modeled using Natural Coordinates

Revolute Joint Constraints:

$$\text{R1:} \quad \left.\begin{array}{l} \vec{r_2} = \vec{r_3} \\ \vec{u_2} = \vec{u_3} \end{array}\right\} \quad \text{Revolute Joint 1} \tag{4.1}$$

$$\text{R2:} \quad \left.\begin{array}{l} \vec{r_4} = \vec{r_5} \\ \vec{u_4} = \vec{u_5} \end{array}\right\} \quad \text{Revolute Joint 2} \tag{4.2}$$

$$\text{R3:} \quad \left.\begin{array}{l} \vec{r_6} = \vec{r_7} \\ \vec{u_6} = \vec{u_7} \end{array}\right\} \quad \text{Revolute Joint 3} \tag{4.3}$$

$$\text{R4:} \quad \left.\begin{array}{l} \vec{r_8} = \vec{r_1} \\ \vec{u_8} = \vec{u_1} \end{array}\right\} \quad \text{Revolute Joint 4} \tag{4.4}$$

Rigid-body and Fixed (grounding) Constraints:

$$\begin{array}{llll}
(\vec{r_2}-\vec{r_1}).\vec{u_1}=0 & (\vec{r_2}-\vec{r_1}).\vec{u_2}=0 & |\vec{r_2}-\vec{r_1}|^2=a^2 & \vec{u_1}.\vec{u_2}=\cos(\alpha) \\
(\vec{r_3}-\vec{r_4}).\vec{u_4}=0 & (\vec{r_3}-\vec{r_4}).\vec{u_3}=0 & |\vec{r_3}-\vec{r_4}|^2=a^2 & \vec{u_4}.\vec{u_3}=\cos(\alpha) \\
(\vec{r_6}-\vec{r_5}).\vec{u_5}=0 & (\vec{r_6}-\vec{r_5}).\vec{u_6}=0 & |\vec{r_6}-\vec{r_5}|^2=a^2 & \vec{u_5}.\vec{u_6}=\cos(\alpha) \\
(\vec{r_7}-\vec{r_8}).\vec{u_8}=0 & (\vec{r_7}-\vec{r_8}).\vec{u_7}=0 & |\vec{r_7}-\vec{r_8}|^2=a^2 & \vec{u_8}.\vec{u_7}=\cos(\alpha) \\
\vec{r_1}=\vec{0} & \vec{r_2}=\vec{p} & & \\
|\vec{u_i}|^2=1,\ \forall i\in[1,8] & & &
\end{array} \tag{4.5}$$

4.4.2 Applying RSA Methodology-2

One can note that the constraint equations here are mostly quadratic, except for the norm conditions, which can also be implemented as quadratic conditions, by squaring those conditions, as has been done in Equations (4.5). The benefit of having a set of constraints which are all quadratic is that the elements of the Jacobian becomes linear and we can hope to apply our RSA algorithm symbolically – linear independence between linear equations is easy to find symbolically and for the Bennett Mechanism this is indeed the case.

As shown in the Mathematica® Notebook in Appendix A.3, we did not have to take any specific numerical value for our RSA algorithm. The RSA algorithm gives the results symbolically and since they are same as mentioned in Section 4.3, they are not repeated here.

4.5 A Planar Over-Constrained Mechanism

In this section, we will implement our RSA methodology on a planar over-contrained mechanism using natural coordinates. The mechanism is shown in Figure 4.5.

4.5.1 Modeling using Natural Coordinates

The prismatic (P) joint constraint equations can be written as

$$\text{P1:}\quad \left.\begin{array}{l}\vec{r_1}\times\vec{r_3}=0\\(\vec{r_3}-\vec{r_{10}}).\vec{r_1}=l_1l_6\cos(\phi_1)\end{array}\right\}\quad \text{Prismatic Joint 1} \tag{4.6}$$

$$\text{P2:}\quad \left.\begin{array}{l}\vec{r_7}\times\vec{r_8}=0\\(\vec{r_9}-\vec{r_7}).\vec{r_8}=l_5l_4\cos(\phi_2)\end{array}\right\}\quad \text{Prismatic Joint 2} \tag{4.7}$$

$$\text{P3:}\quad \left.\begin{array}{l}(\vec{r_{10}}-\vec{r_3})\times(\vec{r_3}-\vec{r_9})=0\\(\vec{r_{10}}-\vec{r_3}).(\vec{r_9}-\vec{r_7})=l_5l_6\end{array}\right\}\quad \text{Prismatic Joint 3} \tag{4.8}$$

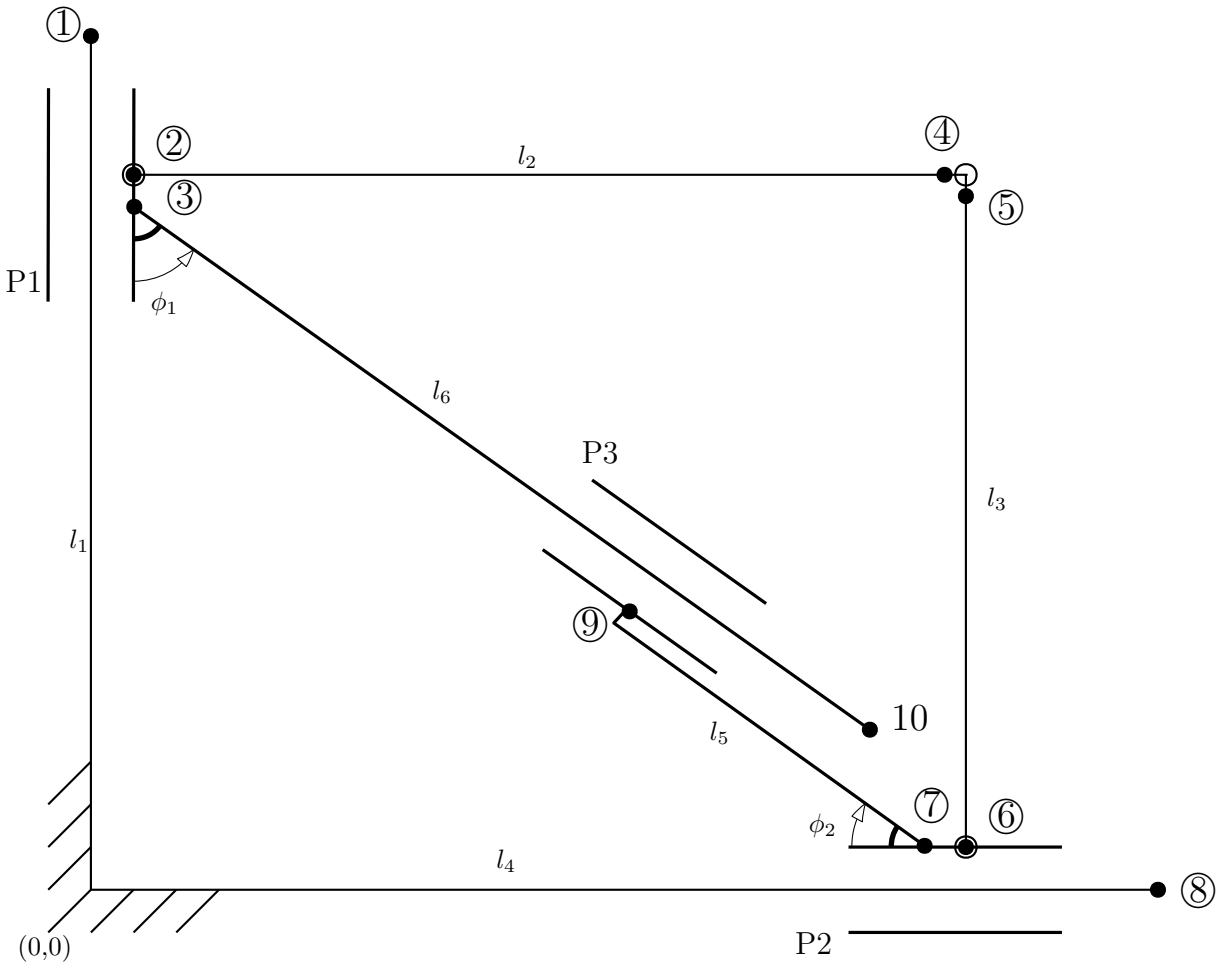


Figure 4.5: A Planar over-constrained mechanism

The revolute (R) joint constraint equations are given as

$$\text{R1:} \quad \left.\vec{r_2} = \vec{r_3}\right\} \quad \text{Revolute Joint 1} \tag{4.9}$$

$$\text{R2:} \quad \left.\vec{r_4} = \vec{r_5}\right\} \quad \text{Revolute Joint 2} \tag{4.10}$$

$$\text{R3:} \quad \left.\vec{r_7} = \vec{r_6}\right\} \quad \text{Revolute Joint 3} \tag{4.11}$$

The rigid-body and fixed (grounding) constraints are

$$\begin{aligned} \vec{r_1} &= (0, l_1)^T & \vec{r_8} &= (l_4, 0)^T \\ |\vec{r_4} - \vec{r_2}| &= l_2 & |\vec{r_7} - \vec{r_9}| &= l_5 \\ |\vec{r_5} - \vec{r_6}| &= l_3 & |\vec{r_3} - \vec{r_{10}}| &= l_6 \end{aligned} \tag{4.12}$$

4.5.2 Applying RSA Methodology-2

The RSA Methodology-2 was applied on this mechanism as can be seen in the Mathematica® Notebook in the Appendix A.4. The rank of Jacobian matrix is $r = rank(\Phi_q) = 19$ and the ranks r_i and r_{-i} are shown in the table 4.4.

i	$r_i = rank(\Phi_q^i)$	$r_{-i} = rank(\Phi_q^{-i})$	$r_i + r_{-i}$
P1	2	18	20
P2	2	18	20
P3	2	18	20
R1	2	17	19
R2	2	17	19
R3	2	17	19

Table 4.4: Methodology-2 Results for the Planar Mechanism

As we can see in the results, for all the prismatic joints, $r_i + r_{-i} > r = 19$. Hence, for none of the prismatic joints can the reactions be found uniquely. However, for all revolute joints $r_i + r_{-i} = r = 19$. Hence, all the revolute joint reactions can be found uniquely.

4.6 Results and Discussion

In this chapter, we perfomed RSA on 4 mechanisms – 3 planar and 1 spatial Bennett mechanism. The kinematics of these mechanisms were modeled either using absolute coordinate formulation or joint-augmented natural coordinate formulation. The Bennett Mechanism (spatial) was modeled using both the formulations so that a comparison could be made between the two formulations. We discovered that the natural coordinates are a much better choice for doing RSA. While using the absolute coordinates, we had to resort to specific numerical values for the configuration of mechanism, using Natural Coordinates we could solve the problem symbolically for any general configuration.

When using absolute coordinates, the numerical values for the configuration of the mechanism were obtained by solving the initial-position problem of the mechanism, i.e., solving the loop-closure equations. These loop closure equations were formulated using quaternions and

vectors, instead of homogeneous transformation matrices and the details and source code of this exercise are included in Appendices B.1 and C. We expected to gain some computational advantage with this, however no advantage was observed using quaternions. This could be attributed to the fact that we were using a very high-level library in Python which has its own overheads.

Chapter 5

Implementation : Finding Joint Reactions

In this chapter, we demonstrate how we can calculate the "solvable" joint reactions in over-constrained mechanisms. This provides a completion to our problem statement. We demonstrate this using one of the mechanisms, namely the planar mechanism in section 4.5, which we analyzed in the previous chapter. Recall that in this mechanism, we could find joint reactions in all the revolute joints but not in the prismatic joints.

5.1 Kinematic Modeling of the Planar Mechanism

For the purpose of finding joint reactions, we will model the chosen planar mechanism using absolute coordinates. This is because all the constraints in absolute coordinates formulation are directly related to joints, hence they are much more convenient to use than other formulations. This aspect was discussed in Section 2.5.

This mechanism, modelled using absolute coordinates, has been shown in Figure 5.1. There are 12 constraints in this formulation which can be given as follows.

Prismatic joint constraint equations:

$$\text{P1:} \quad \left.\begin{array}{l} \Phi_1 : y_1 = 0 \\ \Phi_2 : \theta_1 = 0 \end{array}\right\} \quad \text{Prismatic Joint 1} \tag{5.1}$$

$$\text{P2:} \quad \left.\begin{array}{l} \Phi_3 : x_2 = 0 \\ \Phi_4 : \theta_2 = 0 \end{array}\right\} \quad \text{Prismatic Joint 2} \tag{5.2}$$

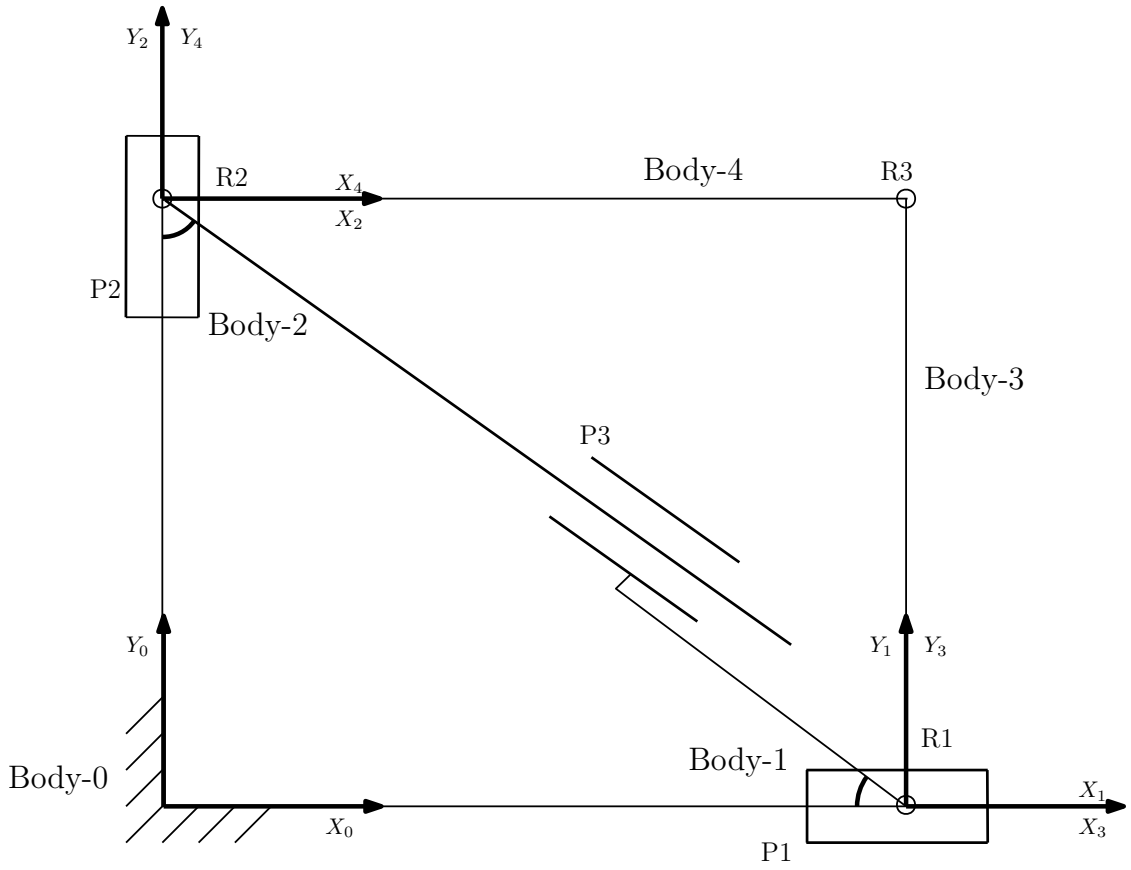


Figure 5.1: Absolute Coordinates Formulation of a planar mechanism

$$\text{P3:} \quad \left.\begin{array}{l} \Phi_5 : x_2 - x_1 + y_2 - y_1 = 0 \\ \Phi_6 : \theta_2 - \theta_1 = 0 \end{array}\right\} \quad \text{Prismatic Joint 3} \tag{5.3}$$

Revolute joint constraint equations:

$$\text{R1:} \quad \left.\begin{array}{l} \Phi_7 : x_1 - x_3 = 0 \\ \Phi_8 : y_1 - y_3 = 0 \end{array}\right\} \quad \text{Revolute Joint 1} \tag{5.4}$$

$$\text{R2:} \quad \left.\begin{array}{l} \Phi_9 : x_2 - x_4 = 0 \\ \Phi_{10} : y_2 - y_4 = 0 \end{array}\right\} \quad \text{Revolute Joint 2} \tag{5.5}$$

$$\text{R3:} \quad \left.\begin{array}{l} \Phi_{11} : x_3 - \sin(\theta_3) - x_4 - \cos(\theta_4) = 0 \\ \Phi_{12} : y_3 - \cos(\theta_3) - y_4 - \sin(\theta_4) = 0 \end{array}\right\} \quad \text{Revolute Joint 3} \tag{5.6}$$

5.2 Dynamics of the Planar Mechanism

All the analysis in this section was done in Mathematica®, and the corresponding Mathematica® notebook has been included in Appendix A.4.

5.2.1 Step-1 : Finding Jacobian

The Jacobian for this mechanism is first calculated. Being over-constrained, the Jacobian (12×12) comes out to be singular.

$$\mathbf{\Phi_q} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\cos(\theta_3(t)) & -1 & 0 & \sin(\theta_4(t)) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -\sin(\theta_3(t)) & 0 & -1 & -\cos(\theta_4(t)) \end{pmatrix} \tag{5.7}$$

Performing the analysis in Mathematica®, the Jacobian is found to be rank-deficient by 1. We then find the dependent rows, and eliminate one of the dependent rows randomly (in this case, we eliminated row-2). Now, we have a 11×12 Jacobian, which is not rank-deficient and this Jacobian can now be used to do further analysis.

5.2.2 Step-2 : Finding mass matrix

We assume that the mass of Body-1 and Body-2 are concentrated at joints R1 and R2, respectively. Body-3 and Body-4 are assumed to be rods with uniform mass distribution. The mass matrices of these 4 bodies are given as:

$$M_1 = \begin{pmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & \frac{l_1^2 m}{12} \end{pmatrix} \tag{5.8}$$

$$M_2 = \begin{pmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & \frac{l_2^2 m}{12} \end{pmatrix} \tag{5.9}$$

$$M_3 = \begin{pmatrix} m & 0 & \frac{1}{2}l_3(-m)\sin(\theta_3(t)) \\ 0 & m & \frac{1}{2}l_3 m\cos(\theta_3(t)) \\ \frac{1}{2}l_3(-m)\sin(\theta_3(t)) & \frac{1}{2}l_3 m\cos(\theta_3(t)) & \frac{l_3^2 m}{3} \end{pmatrix} \tag{5.10}$$

$$M_4 = \begin{pmatrix} m & 0 & \frac{1}{2}l_4(-m)\sin(\theta_4(t)) \\ 0 & m & \frac{1}{2}l_4 m\cos(\theta_4(t)) \\ \frac{1}{2}l_4(-m)\sin(\theta_4(t)) & \frac{1}{2}l_4 m\cos(\theta_4(t)) & \frac{l_4^2 m}{3} \end{pmatrix} \tag{5.11}$$

The final combined mass matrix is given as:

$$\mathbf{M} = \begin{pmatrix} M_1 & 0 & 0 & 0 \\ 0 & M_2 & 0 & 0 \\ 0 & 0 & M_3 & 0 \\ 0 & 0 & 0 & M_4 \end{pmatrix} \tag{5.12}$$

5.2.3 Step-3 : Integrating Dynamics Equations

Now, we use the equations (3.7) to solve the dynamics of this mechanism. We integrate the equations (3.7a) to solve for $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$. Then, we can use these values to evaluate $\boldsymbol{\lambda}$ at any given time or time interval, using equation (3.7b). For doing this, we take the time period to be from $t = 0$ to $t = 5$ seconds. We also apply an external force on the centre of mass (CM) of Body-1 in positive X-direction. The external force vector $\mathbf{Q_e}$ thus becomes:

$$Q_e = \{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\} \tag{5.13}$$

After this, we solve for $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ using NDSolve. These values are then used to calculate $\boldsymbol{\lambda}$ and subsequently constraint reactions $\mathbf{f}$, using the relation $\mathbf{f} = \boldsymbol{\Phi}_{\mathbf{q}}{}^T\boldsymbol{\lambda}$. The values of constraint forces thus calculated can now be plotted as shown in figures 5.2 to 5.7. We have only plotted the reactions corresponding to constraints on joints R_1, R_2 and R_3, because as found in section 4.5, it's only for these joints that the reactions can be found uniquely. The constraints corresponding to these joints are Φ_7 - Φ_{12}, and the corresponding calculated forces f_7 - f_{12}. As these joints are rotary joints, the generalized constraint forces are all forces and not moments. For example, f_7 and f_8 are forces on joints R_1 in X and Y directions, respectively.

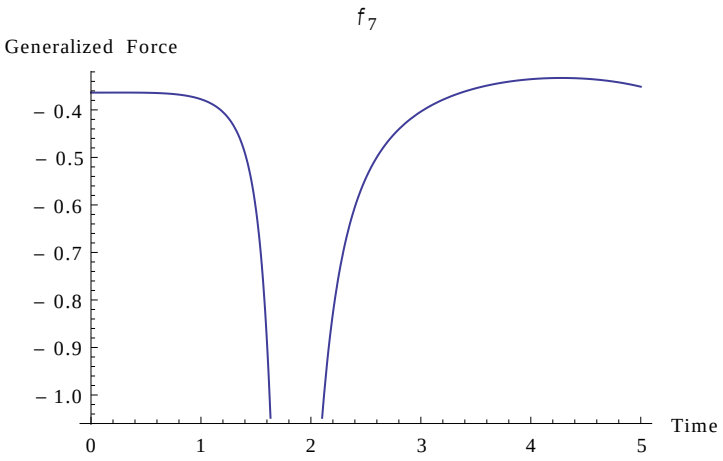


Figure 5.2: Reaction Force vs Time

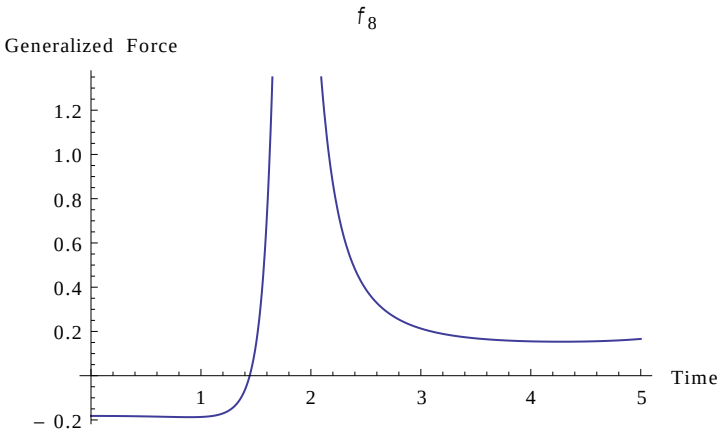


Figure 5.3: Reaction Force vs Time

5.3 Summary

In this chapter, we gave the complete step-by-step procedure for finding joint reactions in an over-constrained mechanism, and demonstrated with a planar mechanism shown in Figure 5.1. The kinematic modeling of the mechanism was done using absolute coordinates in section 5.1, and the constraint equations laid out. Then, in section 5.2, we formulate the Jacobian of the system and find its mass matrix. These are then substituted in the dynamics equations, which are integrated using NDSolve as mentioned in section 5.2.3. The results of integration are plotted in Figures 5.2 - 5.7.

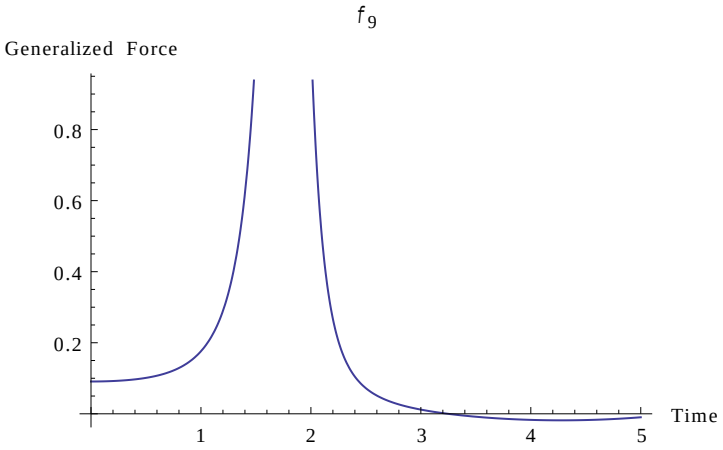


Figure 5.4: Reaction Force vs Time

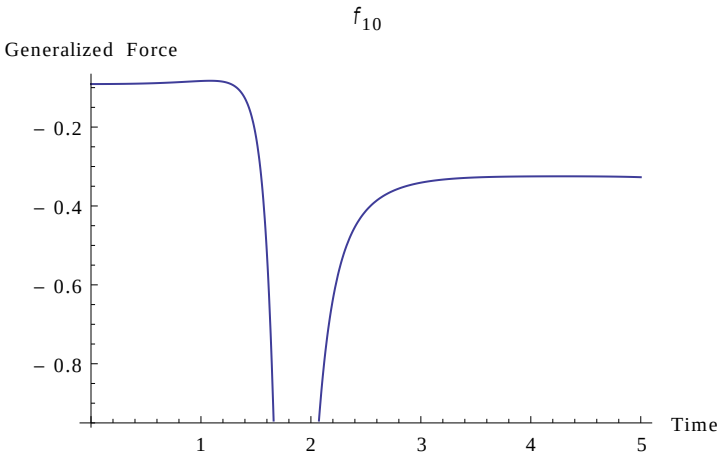


Figure 5.5: Reaction Force vs Time

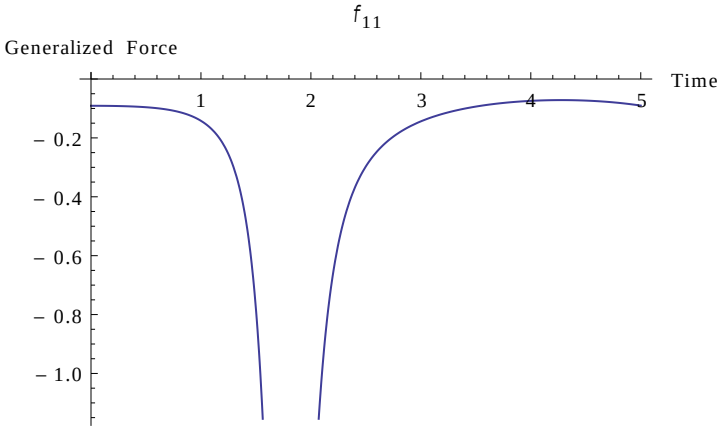


Figure 5.6: Reaction Force vs Time

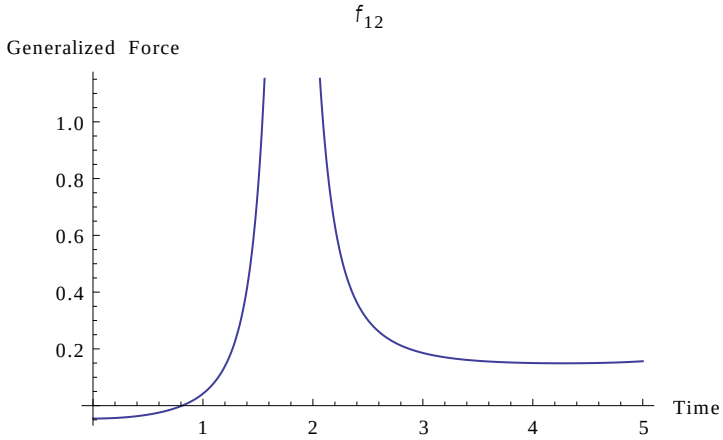


Figure 5.7: Reaction Force vs Time

Chapter 6

Conclusions

6.1 Concluding Remarks

In this thesis, we have formulated a methodology for Reaction Solvability Analysis (RSA) of over-constrained mechanisms in Chapter 3. This methodology is used to analyze several planar and spatial over-constrained mechanisms in Chapter 4. We also present a procedure to solve the dynamics of these mechanisms in Section 3.2, which was then used to find the actual joint reactions of an over-constrained mechanism in Chapter 5.

One major contribution of this thesis lies in investigating various coordinate formulations for the purpose of doing RSA and finding which are suitable for this purpose. To this end, we compared various coordinate formulations in Chapter 2, and found that the absolute coordinates are the only coordinates which capture joint reactions directly, hence the only ones usable for RSA. The relative coordinates, owing to their minimal nature, hide most of the joint reactions, and the natural coordinates capture joint reactions of only certain types of joints. However, we showed that if natural coordinates are modified, they can capture the joint reactions of any joint type. This modification is named as Joint-Augmented Natural Coordinate Formulation in Section 2.4.3 and this formulation, while non-minimal, provided several benefits over absolute coordinates. In Chapter 4, the Bennett mechanism was solved using both absolute and the joint-augmented natural coordinates. While using the former, we had to resort to numerical values of its orientation, using the latter we could solve the problem symbolically. This gain in computational efficiency can be attributed to the fact that constraint equations in natural coordinates are all quadratic equations, hence resulting in a Jacobian matrix whose elements are linear and thus easy to investigate.

Finally, in Chapter 5, we provided a completion to our problem statement by solving for the "solvable" joint reactions in one planar mechanism. This involved bringing dynamics equations

into the picture, which coupled with the constraint equations, are a set of differential-algebraic equations (DAEs). For finding joint reactions, we used absolute coordinates, as constraint reactions corresponding to natural coordinates are hard to interpret physically.

6.2 Future Directions

In our work, we focused only on holonomic constraints. One possible extension would be to extend the RSA algorithms to systems with non-holonomic constraints. Further, the polynomial nature of constraints using natural coordinates holds great potential for RSA. Since our problem statement ultimately boils down to checking the (linear or non-linear) independence of constraints, one can analyze the constraints vector $\mathbf{\Phi}$ itself for independent constraints, without going into the Jacobian. Checking for linear independence of polynomials is well established. However, linear independence would only provide us a weak idea about reaction solvability, for constraints could depend non-linearly as well. In this case, one can check for algebraic independence (a nonlinear generalization of linear independence of polynomials) of polynomials as some recent developments seem to suggest that this may be possible[44].

The Jacobian method that we have used can have problems of singularity and the uniquely solvable joint reactions can change when the mechanism is in a singular location, i.e., the joints which are solvable may become un-solvable in singular locations and vice-versa. This hasn't been investigated in this thesis, and can be a good problem to investigate in the future.

Finally, as discussed in the previous section, the thesis recommended natural coordinates for use in RSA but not in actually solving joint reactions. This was because the physical interpretation of constraint reactions is not easy when using the natural coordinates, as commented by the originator of this formulation himself [11]. A scope of future work could be to attempt to physically interpret these joint reactions so that they can also be used to "find" joint reactions and thus providing a uniform natural-coordinates based formulation.

Appendices

Appendix A

Mathematica® Notebooks

In the following pages, we have attached the Mathematica® Notebooks for RSA of Kempe Burmester and Bennett Mechanism, and for Dynamics of a planar mechanism.

A.1 Kempe Burmester

```
(***************************)
(* Kempe Burmester Problem *)
(***************************)

mat = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}

f1 = x1; f2 = y1;

f3 = x2 - x1 - Cos[p1] * Lap; f4 = y2 - y1 - Sin[p1] * Lap;

f4
-y1 + y2 - Lap Sin[p1]

jacob = D[{f1, f2, f3, f4}, {{x1, y1, p1, x2, y2, p2}}]
{{1, 0, 0, 0, 0, 0}, {0, 1, 0, 0, 0, 0},
 {-1, 0, Lap Sin[p1], 1, 0, 0}, {0, -1, -Lap Cos[p1], 0, 1, 0}}

{{1, 0, 0, 0, 0, 0}, {0, 1, 0, 0, 0, 0},
 {-1, 0, Lap Sin[p1], 1, 0, 0}, {0, -1, -Lap Cos[p1], 0, 1, 0}}
{{1, 0, 0, 0, 0, 0}, {0, 1, 0, 0, 0, 0},
 {-1, 0, Lap Sin[p1], 1, 0, 0}, {0, -1, -Lap Cos[p1], 0, 1, 0}}

f5 = x3 - x1 - Cos[p1] * Lab; f6 = y2 - y1 - Sin[p1] * Lab;

f7 = x3 - x4 + Cos[p3] * Lbr - Cos[p4] * Ltr;
f8 = y3 - y4 + Sin[p3] * Lbr - Sin[p4] * Ltr;

f9 = x3 - x7 + Cos[p3] * Lbc - Cos[p7] * Ldc;
f10 = y3 - y7 + Sinh[p3] * Lbc - Sin[p7] * Ldc;

f11 = x5 - x7 + Cos[p5] * Ltq - Cos[p7] * Lqd;
f12 = y5 - y7 + Sin[p5] * Ltq - Sin[p7] * Lqd;

f13 = x7 - Lad; f14 = y7;

f15 = x6 - Las; f16 = y6;

f17 = x4 - x5; f18 = y4 - y5;
f19 = x5 - x6 - Cos[p6] * Lts; f20 = y5 - y6 - Sin[p6] * Lts;

f21 = x4 - x2 - Cos[p2] * Lpt; f22 = y4 - y2 - Sin[p2] * Lpt;
```

```
jacob = D[{f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13,
   f14, f15, f16, f17, f18, f19, f20, f21, f22}, {{x1, y1, p1, x2, y2,
    p2, x3, y3, p3, x4, y4, p4, x5, y5, p5, x6, y6, p6, x7, y7, p7}}]
```

```
{{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {-1, 0, Lap Sin[p1], 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, -1, -Lap Cos[p1], 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {-1, 0, Lab Sin[p1], 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, -1, -Lab Cos[p1], 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 1, 0, -Lbr Sin[p3], -1, 0, Ltr Sin[p4], 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 1, Lbr Cos[p3], 0, -1, -Ltr Cos[p4], 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 1, 0, -Lbc Sin[p3], 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, Ldc Sin[p7]},
 {0, 0, 0, 0, 0, 0, 0, 1, Lbc Cosh[p3], 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -Ldc Cos[p7]},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, -Ltq Sin[p5], 0, 0, 0, -1, 0, Lqd Sin[p7]},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, Ltq Cos[p5], 0, 0, 0, 0, -1, -Lqd Cos[p7]},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, Lts Sin[p6], 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, -Lts Cos[p6], 0, 0, 0},
 {0, 0, 0, -1, 0, Lpt Sin[p2], 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, -1, -Lpt Cos[p2], 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}
```

```
jacob // MatrixForm
```

1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
-1	0	Lap Sin[p1]	1	0	0	0	0	0	0	0	0	0	0
0	-1	-Lap Cos[p1]	0	1	0	0	0	0	0	0	0	0	0
-1	0	Lab Sin[p1]	0	0	0	1	0	0	0	0	0	0	0
0	-1	-Lab Cos[p1]	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	-Lbr Sin[p3]	-1	0	Ltr Sin[p4]	0	0
0	0	0	0	0	0	0	1	Lbr Cos[p3]	0	-1	-Ltr Cos[p4]	0	0
0	0	0	0	0	0	1	0	-Lbc Sin[p3]	0	0	0	0	0
0	0	0	0	0	0	0	1	Lbc Cosh[p3]	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	-1	0
0	0	0	0	0	0	0	0	0	0	1	0	0	-
0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	-1	0	Lpt Sin[p2]	0	0	0	1	0	0	0	0
0	0	0	0	-1	-Lpt Cos[p2]	0	0	0	0	1	0	0	0

```
Dimensions[jacob]
```

```
{22, 21}
```

```
MatrixRank[jacob]
```

```
21
```

```
Lap = 1.0; Lbp = 1.0; Lbr = 0.6614; Lrc = 0.6614;
Clear[Lap, Lbp, Lbr, Lrc]
```

```
Lap = 1.0; Ltq = 1.0; Lab = 2.0; Lbr = 0.6614;
Ltr = 1.25; Lts = 0.6614; Ldc = 2.0; Lqd = 1.0; Lpt = 1.0;
```

```
Dimensions[jacob]
```

```
{22, 21}
```

```
MatrixRank[jacob]
```

```
21
```

```
(*****************************)
(* Here Starts the Method-A+ *)
(*****************************)
```

```
pq1 = jacob[[{1, 2}, All]]; pqm1 = Drop[jacob, {1, 2}, 0];
```

```
r1 = MatrixRank[pq1]
```

```
2
```

```
rm1 = MatrixRank[pqm1]
```

```
20
```

```
(*Used the following bash script to generate formulaes:*)
(*For the 9th constraint, manually wrote the formula*)
(* (j=1;for i in $(seq 1 2 22);do printf'pq'$j'=jacob[[{'$i','$((
  $i+1))'},All]];';((j++));done;)|oclip *)
```

```
pq1 = jacob[[{1, 2}, All]]; pq2 = jacob[[{3, 4}, All]]; pq3 = jacob[[{5, 6}, All]];
pq4 = jacob[[{7, 8}, All]]; pq5 = jacob[[{9, 10}, All]];
pq6 = jacob[[{11, 12}, All]]; pq7 = jacob[[{13, 14}, All]];
pq8 = jacob[[{15, 16}, All]]; pq9 = jacob[[{17, 18, 19, 20, 21, 22}, All]];
```

```
pq9 // MatrixForm
```

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0.6614 & \text{Sin}[p6] & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & -0.6614 & \text{Cos}[p6] & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 1. & \text{Sin}[p2] & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & -1. & \text{Cos}[p2] & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

```
(*Again a Bash script for pqm's. And Manual for pqm9*)
(*(j=1;for i in $(seq 1 2 22);do echo'pqm'$j'=Drop[jacob,{'$i','$((($i+1))'},0];';
  ((j++));done;)|oclip*)

pqm1 = Drop[jacob, {1, 2}, 0];
pqm2 = Drop[jacob, {3, 4}, 0];
pqm3 = Drop[jacob, {5, 6}, 0];
pqm4 = Drop[jacob, {7, 8}, 0];
pqm5 = Drop[jacob, {9, 10}, 0];
pqm6 = Drop[jacob, {11, 12}, 0];
pqm7 = Drop[jacob, {13, 14}, 0];
pqm8 = Drop[jacob, {15, 16}, 0];
pqm9 = Drop[jacob, {17, 22}, 0];

(*Now bash script for printing formulaes for ri's and rmi's *)
(* for i in $(seq 9);do echo "r$i"'=MatrixRank[pq'$i'];
 rm'$i'=MatrixRank[pqm'$i'];'; done; *)

r1 = MatrixRank[pq1]; rm1 = MatrixRank[pqm1];
r2 = MatrixRank[pq2]; rm2 = MatrixRank[pqm2]; r3 = MatrixRank[pq3];
rm3 = MatrixRank[pqm3]; r4 = MatrixRank[pq4]; rm4 = MatrixRank[pqm4];
r5 = MatrixRank[pq5]; rm5 = MatrixRank[pqm5]; r6 = MatrixRank[pq6];
rm6 = MatrixRank[pqm6]; r7 = MatrixRank[pq7]; rm7 = MatrixRank[pqm7];
r8 = MatrixRank[pq8]; rm8 = MatrixRank[pqm8];
r9 = MatrixRank[pq9]; rm9 = MatrixRank[pqm9];


rm8 = MatrixRank[pqm8];

pqm8 // MatrixForm
```

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1.\,\text{Sin}[p1] & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & -1.\,\text{Cos}[p1] & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 2.\,\text{Sin}[p1] & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & -2.\,\text{Cos}[p1] & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -0.6614\,\text{Sin}[p3] & -1 & 0 & 1.25\,\text{Sin}[p4] & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.6614\,\text{Cos}[p3] & 0 & -1 & -1.25\,\text{Cos}[p4] & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\text{Lbc}\,\text{Sin}[p3] & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \text{Lbc}\,\text{Cosh}[p3] & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 1.\,\text{Sin}[p2] & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & -1.\,\text{Cos}[p2] & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}$$

```
MatrixRank[pqm8]
```

```
20
```

```
{r1, r2, r3, r4, r5, r6, r7, r8, r9}
```

```
{2, 2, 2, 2, 2, 2, 2, 2, 6}
```

```
{rm1, rm2, rm3, rm4, rm5, rm6, rm7, rm8, rm9}
```

```
{20, 20, 20, 20, 20, 20, 19, 20, 16}
```

A.2 Bennett Mechanism RSA using Absolute Coordinates

```
SetOptions[EvaluationNotebook[], CellContext → Notebook]

(*********************************************)
(*********************************************)
(******* Notebook for Bennett Mechanism *******)
(*********************************************)
(*********************************************)

(*Applying Specific Bennnett Conditions*)
a = c = b = d;
θ1 = θ3 = θ; θ2 = θ4 = -θ;

(*********************************************)
(* Absolute Coordinates for each joint axis *)
(*********************************************)

r0 = {0, 0, 0};
r1 = {x1, y1, z1};
r2 = {x2, y2, z2};
r3 = {x3, y3, z3};
ψ0 = {α0, β0, γ0};
ψ1 = {α1, β1, γ1};
ψ2 = {α2, β2, γ2};
ψ3 = {α3, β3, γ3};

Q = Join[r1, ψ1, r2, ψ2, r3, ψ3];
(*Complete Vector Containing all the system variables*)

(*****************************************************)
(* Rotational Matrices based on Z-X-Z Euler Angles *)
(*****************************************************)

R0 = {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};
R1 = {{Cos[α1] Cos[γ1] - Sin[α1] Cos[β1] Sin[γ1],
     Sin[α1] Cos[β1] (-Cos[γ1]) + Cos[α1] (-Sin[γ1]), Sin[α1] Sin[β1]},
    {Cos[α1] Cos[β1] Sin[γ1] + Sin[α1] Cos[γ1], Cos[α1] Cos[β1] Cos[γ1] - Sin[α1] Sin[γ1],
     Cos[α1] (-Sin[β1])}, {Sin[β1] Sin[γ1], Sin[β1] Cos[γ1], Cos[β1]}};
R2 = {{Cos[α2] Cos[γ2] - Sin[α2] Cos[β2] Sin[γ2],
     Sin[α2] Cos[β2] (-Cos[γ2]) + Cos[α2] (-Sin[γ2]), Sin[α2] Sin[β2]},
    {Cos[α2] Cos[β2] Sin[γ2] + Sin[α2] Cos[γ2], Cos[α2] Cos[β2] Cos[γ2] - Sin[α2] Sin[γ2],
     Cos[α2] (-Sin[β2])}, {Sin[β2] Sin[γ2], Sin[β2] Cos[γ2], Cos[β2]}};
R3 = {{Cos[α3] Cos[γ3] - Sin[α3] Cos[β3] Sin[γ3],
     Sin[α3] Cos[β3] (-Cos[γ3]) + Cos[α3] (-Sin[γ3]), Sin[α3] Sin[β3]},
    {Cos[α3] Cos[β3] Sin[γ3] + Sin[α3] Cos[γ3], Cos[α3] Cos[β3] Cos[γ3] - Sin[α3] Sin[γ3],
     Cos[α3] (-Sin[β3])}, {Sin[β3] Sin[γ3], Sin[β3] Cos[γ3], Cos[β3]}};
```

```
(*************************************************************************)
(* Positions of various points from different local coordinate systems *)
(*************************************************************************)

s_A^0 = {a, -l Cos[θ1], -l Cos[θ1]};
s_A^1 = {0, 0, -l};
s_B^0 = {a, l Cos[θ1], l Cos[θ1]};
s_B^1 = {0, 0, l};
s_C^1 = {b, -l Cos[θ2], -l Cos[θ2]};
s_C^2 = {0, 0, -l};
s_D^1 = {b, l Cos[θ2], l Cos[θ2]};
s_D^2 = {0, 0, l};
s_e^2 = {c, -l Cos[θ3], -l Cos[θ3]};
s_e^3 = {0, 0, -l};
s_F^2 = {c, l Cos[θ3], l Cos[θ3]};
s_F^3 = {0, 0, l};
s_G^3 = {d, -l Cos[θ4], -l Cos[θ4]};
s_G^0 = {0, 0, -l};
s_H^3 = {d, l Cos[θ4], l Cos[θ4]};
s_H^0 = {0, 0, l};

(************************)
(* Constraint Equations *)
(************************)

Φ^1 = Join[r1 + R1.s_A^1 - r0 - R0.s_A^0, r1 + R1.s_B^1 - r0 - R0.s_B^0];
Φ^2 = Join[r2 + R2.s_C^2 - r1 - R1.s_C^1, r2 + R2.s_D^2 - r1 - R1.s_D^1];
Φ^3 = Join[r3 + R3.s_e^3 - r2 - R2.s_e^2, r3 + R3.s_F^3 - r2 - R2.s_F^2];
Φ^4 = Join[r0 + R0.s_G^0 - r3 - R3.s_G^3, r0 + R0.s_H^0 - r3 - R3.s_H^3];

(*********************************)
(* Complete Constraint Equation *)
(*********************************)

phi = Join[Φ^1, Φ^2, Φ^3, Φ^4]; MatrixForm[Φ]

Φ

(********************)
(* Jacobian Matrix *)
(********************)
```

```
Φq = D[phi, {Q}]; MatrixForm[Φq] (* Jacobian Matrix *)
```

$$\begin{pmatrix}
1 & 0 & 0 & -l\,\text{Cos}[\alpha_1]\,\text{Sin}[\beta_1] \\
0 & 1 & 0 & -l\,\text{Sin}[\alpha_1]\,\text{Sin}[\beta_1] \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & l\,\text{Cos}[\alpha_1]\,\text{Sin}[\beta_1] \\
0 & 1 & 0 & l\,\text{Sin}[\alpha_1]\,\text{Sin}[\beta_1] \\
0 & 0 & 1 & 0 \\
-1 & 0 & 0 & l\,\text{Cos}[\theta]\,\text{Cos}[\alpha_1]\,\text{Sin}[\beta_1] - d\,(-\text{Cos}[\gamma_1]\,\text{Sin}[\alpha_1] - \text{Cos}[\alpha_1]\,\text{Cos}[\beta_1]\,\text{Sin}[\gamma_1]) + l\,\text{Cos} \\
0 & -1 & 0 & l\,\text{Cos}[\theta]\,\text{Sin}[\alpha_1]\,\text{Sin}[\beta_1] + l\,\text{Cos}[\theta]\,(-\text{Cos}[\beta_1]\,\text{Cos}[\gamma_1]\,\text{Sin}[\alpha_1] - \text{Cos}[\alpha_1]\,\text{Sin}[\gamma_1] \\
0 & 0 & -1 & 0 \\
-1 & 0 & 0 & -l\,\text{Cos}[\theta]\,\text{Cos}[\alpha_1]\,\text{Sin}[\beta_1] - d\,(-\text{Cos}[\gamma_1]\,\text{Sin}[\alpha_1] - \text{Cos}[\alpha_1]\,\text{Cos}[\beta_1]\,\text{Sin}[\gamma_1]) - l\,\text{Co} \\
0 & -1 & 0 & -l\,\text{Cos}[\theta]\,\text{Sin}[\alpha_1]\,\text{Sin}[\beta_1] - l\,\text{Cos}[\theta]\,(-\text{Cos}[\beta_1]\,\text{Cos}[\gamma_1]\,\text{Sin}[\alpha_1] - \text{Cos}[\alpha_1]\,\text{Sin}[\gamma_1 \\
0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}$$

```
Dimensions[%45]
```

```
{24, 18}
```

```
(*****************************)
(* Here Starts the Method-A+ *)
(*****************************)

Φq^1 = Φq[[{1, 2, 3, 4, 5, 6}, All]];
Φq^2 = Φq[[{7, 8, 9, 10, 11, 12}, All]];
Φq^3 = Φq[[{13, 14, 15, 16, 17, 18}, All]];
Φq^4 = Φq[[{19, 20, 21, 22, 23, 24}, All]];

Φq^-1 = Drop[Φq, {1, 6}, 0];
Φq^-2 = Drop[Φq, {7, 12}, 0];
Φq^-3 = Drop[Φq, {13, 18}, 0];
Φq^-4 = Drop[Φq, {19, 24}, 0];
```

```
MatrixRank[Φq^-1]
MatrixRank[Φq^-2]
MatrixRank[Φq^-3]
(*MatrixRank[Φq^-4]*)
```

```
17
16
16
```

```
MatrixRank[Φq^1]
MatrixRank[Φq^2]
MatrixRank[Φq^3]
MatrixRank[Φq^4]
```

```
5
6
6
5
```

```
(*Giving specific numerical values to position coordinates*)
{x1, y1, z1} = {10, 0, 0};
{x2, y2, z2} = {5, 8.138, -2.962};
{x3, y3, z3} = {-4.519, 8.921, 0};
{α1, β1, γ1} = {240, 20, 0} * Pi / 180;
{α2, β2, γ2} = {148.4, 34.46, 31.57} * Pi / 180;
{α3, β3, γ3} = {0, 20, 63.13} * Pi / 180;
```

```
MatrixRank[Φq^1]
MatrixRank[Φq^2]
MatrixRank[Φq^3]
MatrixRank[Φq^4]
```

```
5
6
6
5
```

```
MatrixRank[Φq^-1]
MatrixRank[Φq^-2]
MatrixRank[Φq^-3]
MatrixRank[Φq^-4]
```

```
17
16
16
15
```

```
MatrixRank[Φq]
```

```
18
```

A.3 Bennett Mechanism RSA using Natural Coordinates

```
(*********************************************************)
(* Bennett Mechanism Reaction Solvability Analysis (RSA) *)
(* Using Natural Coordinates *)
(*********************************************************)
SetOptions[EvaluationNotebook[], CellContext → Notebook]

MyNorm[a_] := a[[1]]^2 + a[[2]]^2 + a[[3]]^2

Do[(r_j = {x_j, y_j, z_j}), {j, 8}]
Do[(u_j = {e_j, f_j, g_j}), {j, 8}]
Q = Join[r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8];

(*Constraint Equations*)
R1 = Join[r_2 - r_3, u_2 - u_3];
R2 = Join[r_4 - r_5, u_4 - u_5];
R3 = Join[r_6 - r_7, u_6 - u_7];
R4 = Join[r_8 - r_1, u_8 - u_1];
Φ = Join[R1, R2, R3, R4];
(*Adding 8 Unit vector constraints*)
Do[(Φ = Join[Φ, {MyNorm[u_j] - 1}]), {j, 8}]

(*Adding Rigid link constraints*)
Do[(Φ = Join[Φ, {(r_j+1 - r_j).u_j}]), {j, 1, 7, 2}]
Do[(Φ = Join[Φ, {(r_j+1 - r_j).u_j+1}]), {j, 1, 7, 2}]
Do[(Φ = Join[Φ, {MyNorm[r_j+1 - r_j] - a^2}]), {j, 1, 7, 2}]
Do[(Φ = Join[Φ, {u_j.u_j+1 - Cos[α_j]}]), {j, 1, 7, 2}]

(*Adding Fixind(Ground) constraints*)
Φ = Join[Φ, r_1];
Φ = Join[Φ, r_2 - {p, q, t}];

Φ // MatrixForm
```

$$\begin{pmatrix} x_2 - x_3 \\ y_2 - y_3 \\ z_2 - z_3 \\ e_2 - e_3 \\ f_2 - f_3 \\ g_2 - g_3 \\ x_4 - x_5 \\ y_4 - y_5 \\ z_4 - z_5 \\ e_4 - e_5 \\ f_4 - f_5 \\ g_4 - g_5 \\ x_6 - x_7 \\ y_6 - y_7 \\ z_6 - z_7 \end{pmatrix}$$

$$\begin{pmatrix}
e_6 - e_7 \\
f_6 - f_7 \\
g_6 - g_7 \\
-x_1 + x_8 \\
-y_1 + y_8 \\
-z_1 + z_8 \\
-e_1 + e_8 \\
-f_1 + f_8 \\
-g_1 + g_8 \\
-1 + e_1^2 + f_1^2 + g_1^2 \\
-1 + e_2^2 + f_2^2 + g_2^2 \\
-1 + e_3^2 + f_3^2 + g_3^2 \\
-1 + e_4^2 + f_4^2 + g_4^2 \\
-1 + e_5^2 + f_5^2 + g_5^2 \\
-1 + e_6^2 + f_6^2 + g_6^2 \\
-1 + e_7^2 + f_7^2 + g_7^2 \\
-1 + e_8^2 + f_8^2 + g_8^2 \\
e_1 (-x_1 + x_2) + f_1 (-y_1 + y_2) + g_1 (-z_1 + z_2) \\
e_3 (-x_3 + x_4) + f_3 (-y_3 + y_4) + g_3 (-z_3 + z_4) \\
e_5 (-x_5 + x_6) + f_5 (-y_5 + y_6) + g_5 (-z_5 + z_6) \\
e_7 (-x_7 + x_8) + f_7 (-y_7 + y_8) + g_7 (-z_7 + z_8) \\
e_2 (-x_1 + x_2) + f_2 (-y_1 + y_2) + g_2 (-z_1 + z_2) \\
e_4 (-x_3 + x_4) + f_4 (-y_3 + y_4) + g_4 (-z_3 + z_4) \\
e_6 (-x_5 + x_6) + f_6 (-y_5 + y_6) + g_6 (-z_5 + z_6) \\
e_8 (-x_7 + x_8) + f_8 (-y_7 + y_8) + g_8 (-z_7 + z_8) \\
-a^2 + (-x_1 + x_2)^2 + (-y_1 + y_2)^2 + (-z_1 + z_2)^2 \\
-a^2 + (-x_3 + x_4)^2 + (-y_3 + y_4)^2 + (-z_3 + z_4)^2 \\
-a^2 + (-x_5 + x_6)^2 + (-y_5 + y_6)^2 + (-z_5 + z_6)^2 \\
-a^2 + (-x_7 + x_8)^2 + (-y_7 + y_8)^2 + (-z_7 + z_8)^2 \\
-\text{Cos}[\alpha_1] + e_1 e_2 + f_1 f_2 + g_1 g_2 \\
-\text{Cos}[\alpha_3] + e_3 e_4 + f_3 f_4 + g_3 g_4 \\
-\text{Cos}[\alpha_5] + e_5 e_6 + f_5 f_6 + g_5 g_6 \\
-\text{Cos}[\alpha_7] + e_7 e_8 + f_7 f_8 + g_7 g_8 \\
x_1 \\
y_1 \\
z_1 \\
-p + x_2 \\
-q + y_2 \\
-t + z_2
\end{pmatrix}$$

J = D[Φ, {Q}] (* Jacobian Matrix *)

```
{{0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
```

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0},
{-1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
-1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, $2 e_1$,
$2 f_1$, $2 g_1$, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, $2 e_2$, $2 f_2$, $2 g_2$, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, $2 e_3$, $2 f_3$, $2 g_3$, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, $2 e_4$, $2 f_4$, $2 g_4$, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, $2 e_5$, $2 f_5$, $2 g_5$, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2e_6 & 2f_6 & 2g_6 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2e_7 & 2f_7 & 2g_7 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2e_8 & 2f_8 & 2g_8 \\
-e_1 & -f_1 & -g_1 & e_1 & f_1 & g_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-x_1+x_2 & -y_1+y_2 & -z_1+z_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -e_3 & -f_3 & -g_3 & e_3 & f_3 & g_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -x_3+x_4 & -y_3+y_4 & -z_3+z_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -e_5 & -f_5 & -g_5 & e_5 & f_5 & g_5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -x_5+x_6 & -y_5+y_6 & -z_5+z_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -e_7 & -f_7 & -g_7 & e_7 & f_7 & g_7 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -x_7+x_8 & -y_7+y_8 & -z_7+z_8 & 0 & 0 & 0 & 0 \\
-e_2 & -f_2 & -g_2 & e_2 & f_2 & g_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -x_1+x_2 & -y_1+y_2 & -z_1+z_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -e_4 & -f_4 & -g_4 & e_4 & f_4 & g_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -x_3+x_4 & -y_3+y_4 & -z_3+z_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -e_6 & -f_6 & -g_6 & e_6 & f_6 & g_6 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -x_5+x_6 & -y_5+y_6 & -z_5+z_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -e_8 & -f_8 & -g_8 & e_8 & f_8 & g_8 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -x_7+x_8 & -y_7+y_8 & -z_7+z_8 & 0 & 0 & 0 & 0 \\
-2(-x_1+x_2) & -2(-y_1+y_2) & -2(-z_1+z_2) & 2(-x_1+x_2) & 2(-y_1+y_2) & 2(-z_1+z_2) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -2(-x_3+x_4) & -2(-y_3+y_4) & -2(-z_3+z_4) & 2(-x_3+x_4) & 2(-y_3+y_4) & 2(-z_3+z_4) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2(-x_5+x_6) & -2(-y_5+y_6) & -2(-z_5+z_6) & 2(-x_5+x_6) & 2(-y_5+y_6) & 2(-z_5+z_6) & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2(-x_7+x_8) & -2(-y_7+y_8) & -2(-z_7+z_8) & 2(-x_7+x_8) & 2(-y_7+y_8) & 2(-z_7+z_8) & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_2 \\
f_2 & g_2 & e_1 & f_1 & g_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & e_4 & f_4 & g_4 & e_3 & f_3 & g_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_6 & f_6 & g_6 & e_5 & f_5 & g_5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_8 & f_8 & g_8 & e_7 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix},
$$

```
{0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}
```

```
MatrixRank[J]
```

48

```
Dimensions[J]
```

{54, 48}

```
RSA[cList_] := (J^i = J[[cList, All]]; J^-i = Complement[J, J^i];
  Print["J^i = ", MatrixRank[J^i], " and J^-i = ", MatrixRank[J^-i]];)
```

```
RSA[{1, 2, 3, 4, 5, 6}]
```

J^i = 6 and J^-i = 47

A.4 Dynamics of a Planar Mechanism

```
(*********************************)
(*Dynamics of A Planar Mechanism*)
(*********************************)
SetOptions[EvaluationNotebook[], CellContext → Notebook]

(*Defining Absolute Coordinates*)
Do[(r_j = {x_j[t], y_j[t]}), {j, 4}]
q = Join[r_1, {θ_1[t]}, r_2, {θ_2[t]}, r_3, {θ_3[t]}, r_4, {θ_4[t]}]
```

$$\{x_1[t], y_1[t], \theta_1[t], x_2[t], y_2[t], \theta_2[t], x_3[t], y_3[t], \theta_3[t], x_4[t], y_4[t], \theta_4[t]\}$$

```
(*Defining Constraints*)
Φ = {y_1[t], θ_1[t], x_2[t], θ_2[t], x_2[t] - x_1[t] + y_2[t] - y_1[t],
  θ_2[t] - θ_1[t], x_1[t] - x_3[t], y_1[t] - y_3[t], x_2[t] - x_4[t], y_2[t] - y_4[t],
  x_3[t] - Sin[θ_3[t]] - x_4[t] - Cos[θ_4[t]], y_3[t] + Cos[θ_3[t]] - y_4[t] - Sin[θ_4[t]]}
```

$$\{y_1[t], \theta_1[t], x_2[t], \theta_2[t], -x_1[t] + x_2[t] - y_1[t] + y_2[t], -\theta_1[t] + \theta_2[t], x_1[t] - x_3[t], y_1[t] - y_3[t], x_2[t] - x_4[t], y_2[t] - y_4[t], -\text{Cos}[\theta_4[t]] - \text{Sin}[\theta_3[t]] + x_3[t] - x_4[t], \text{Cos}[\theta_3[t]] - \text{Sin}[\theta_4[t]] + y_3[t] - y_4[t]\}$$

```
MatrixForm[%]
```

$$\begin{pmatrix} y_1[t] \\ \theta_1[t] \\ x_2[t] \\ \theta_2[t] \\ -x_1[t] + x_2[t] - y_1[t] + y_2[t] \\ -\theta_1[t] + \theta_2[t] \\ x_1[t] - x_3[t] \\ y_1[t] - y_3[t] \\ x_2[t] - x_4[t] \\ y_2[t] - y_4[t] \\ -\text{Cos}[\theta_4[t]] - \text{Sin}[\theta_3[t]] + x_3[t] - x_4[t] \\ \text{Cos}[\theta_3[t]] - \text{Sin}[\theta_4[t]] + y_3[t] - y_4[t] \end{pmatrix}$$

```
J = D[Φ, {q}] (* Jacobian Matrix *)
```

$$\{\{0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}, \{0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0\}, \{0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0\}, \{0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0\}, \{-1, -1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0\}, \{0, 0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 0\}, \{1, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0\}, \{0, 1, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0\}, \{0, 0, 0, 1, 0, 0, 0, 0, 0, -1, 0, 0\}, \{0, 0, 0, 0, 1, 0, 0, 0, 0, 0, -1, 0\}, \{0, 0, 0, 0, 0, 0, 1, 0, -\text{Cos}[\theta_3[t]], -1, 0, \text{Sin}[\theta_4[t]]\}, \{0, 0, 0, 0, 0, 0, 0, 1, -\text{Sin}[\theta_3[t]], 0, -1, -\text{Cos}[\theta_4[t]]\}\}$$

```
q' = D[q, t]
```

$$\{x_1'[t], y_1'[t], \theta_1'[t], x_2'[t], y_2'[t], \theta_2'[t], x_3'[t], y_3'[t], \theta_3'[t], x_4'[t], y_4'[t], \theta_4'[t]\}$$

```
l1 = l2 = 1 / 2;
l3 = l4 = 1;
m = 1;
(*θ3=θ4=0;*)
```

$$\mathbf{M_1} = \left\{\{m, 0, 0\}, \{0, m, 0\}, \left\{0, 0, \frac{m\, l_1^2}{12}\right\}\right\}$$

$$\mathbf{M_2} = \left\{\{m, 0, 0\}, \{0, m, 0\}, \left\{0, 0, \frac{m\, l_2^2}{12}\right\}\right\}$$

$$\mathbf{M_3} = \left\{\left\{m, 0, \frac{1}{2}(-m)\, l_3 \sin[\theta_3[t]]\right\}, \left\{0, m, \frac{1}{2} m\, l_3 \cos[\theta_3[t]]\right\}, \left\{\frac{1}{2}(-m)\, l_3 \sin[\theta_3[t]], \frac{1}{2} m\, l_3 \cos[\theta_3[t]], \frac{m\, l_3^2}{3}\right\}\right\}$$

$$\mathbf{M_4} = \left\{\left\{m, 0, \frac{1}{2}(-m)\, l_4 \sin[\theta_4[t]]\right\}, \left\{0, m, \frac{1}{2} m\, l_4 \cos[\theta_4[t]]\right\}, \left\{\frac{1}{2}(-m)\, l_4 \sin[\theta_4[t]], \frac{1}{2} m\, l_4 \cos[\theta_4[t]], \frac{m\, l_4^2}{3}\right\}\right\}$$

$$\left\{\{1, 0, 0\}, \{0, 1, 0\}, \left\{0, 0, \frac{1}{48}\right\}\right\}$$

$$\left\{\{1, 0, 0\}, \{0, 1, 0\}, \left\{0, 0, \frac{1}{48}\right\}\right\}$$

$$\left\{\left\{1, 0, -\frac{1}{2}\sin[\theta_3[t]]\right\}, \left\{0, 1, \frac{1}{2}\cos[\theta_3[t]]\right\}, \left\{-\frac{1}{2}\sin[\theta_3[t]], \frac{1}{2}\cos[\theta_3[t]], \frac{1}{3}\right\}\right\}$$

$$\left\{\left\{1, 0, -\frac{1}{2}\sin[\theta_4[t]]\right\}, \left\{0, 1, \frac{1}{2}\cos[\theta_4[t]]\right\}, \left\{-\frac{1}{2}\sin[\theta_4[t]], \frac{1}{2}\cos[\theta_4[t]], \frac{1}{3}\right\}\right\}$$

```
M = ArrayFlatten[{{M1, 0, 0, 0}, {0, M2, 0, 0}, {0, 0, M3, 0}, {0, 0, 0, M4}}]
```

$$\left\{\{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}, \{0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}, \left\{0, 0, \frac{1}{48}, 0, 0, 0, 0, 0, 0, 0, 0, 0\right\}, \{0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0\}, \{0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0\}, \left\{0, 0, 0, 0, 0, \frac{1}{48}, 0, 0, 0, 0, 0, 0\right\}, \left\{0, 0, 0, 0, 0, 0, 1, 0, -\frac{1}{2}\sin[\theta_3[t]], 0, 0, 0\right\}, \left\{0, 0, 0, 0, 0, 0, 0, 1, \frac{1}{2}\cos[\theta_3[t]], 0, 0, 0\right\}, \left\{0, 0, 0, 0, 0, 0, -\frac{1}{2}\sin[\theta_3[t]], \frac{1}{2}\cos[\theta_3[t]], \frac{1}{3}, 0, 0, 0\right\}, \left\{0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, -\frac{1}{2}\sin[\theta_4[t]]\right\}, \left\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, \frac{1}{2}\cos[\theta_4[t]]\right\}, \left\{0, 0, 0, 0, 0, 0, 0, 0, 0, -\frac{1}{2}\sin[\theta_4[t]], \frac{1}{2}\cos[\theta_4[t]], \frac{1}{3}\right\}\right\}$$

```
MatrixRank[J]
```

```
11
```

```
(*RowReduce[J] //MatrixForm*)
```

```
J = Drop[J, {2}, 0] (*Eliminating a dependent constraint, arbitrarily*)
```

```
{{0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0},
 {-1, -1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0}, {0, 0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 0},
 {1, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0}, {0, 1, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0},
 {0, 0, 0, 1, 0, 0, 0, 0, 0, -1, 0, 0}, {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, -1, 0},
 {0, 0, 0, 0, 0, 0, 1, 0, -Cos[Θ3[t]], -1, 0, Sin[Θ4[t]]},
 {0, 0, 0, 0, 0, 0, 0, 1, -Sin[Θ3[t]], 0, -1, -Cos[Θ4[t]]}}
```

```
JIn = PseudoInverse[J]
```

A very large output was generated. Here is a sample of it:

```
{ <<1>> }
```

Show Less | Show More | Show Full Output | Set Size Limit...

```
Qe = {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
q′ = D[q, t]
```

```
{x1′[t], y1′[t], Θ1′[t], x2′[t], y2′[t], Θ2′[t], x3′[t], y3′[t], Θ3′[t], x4′[t], y4′[t], Θ4′[t]}
```

```
Qd = -(∂{q} (J.q′)).q′
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, -Sin[Θ3[t]] Θ3′[t]^2 - Cos[Θ4[t]] Θ4′[t]^2,
 Cos[Θ3[t]] Θ3′[t]^2 - Sin[Θ4[t]] Θ4′[t]^2}
```

MIn = Inverse[M]

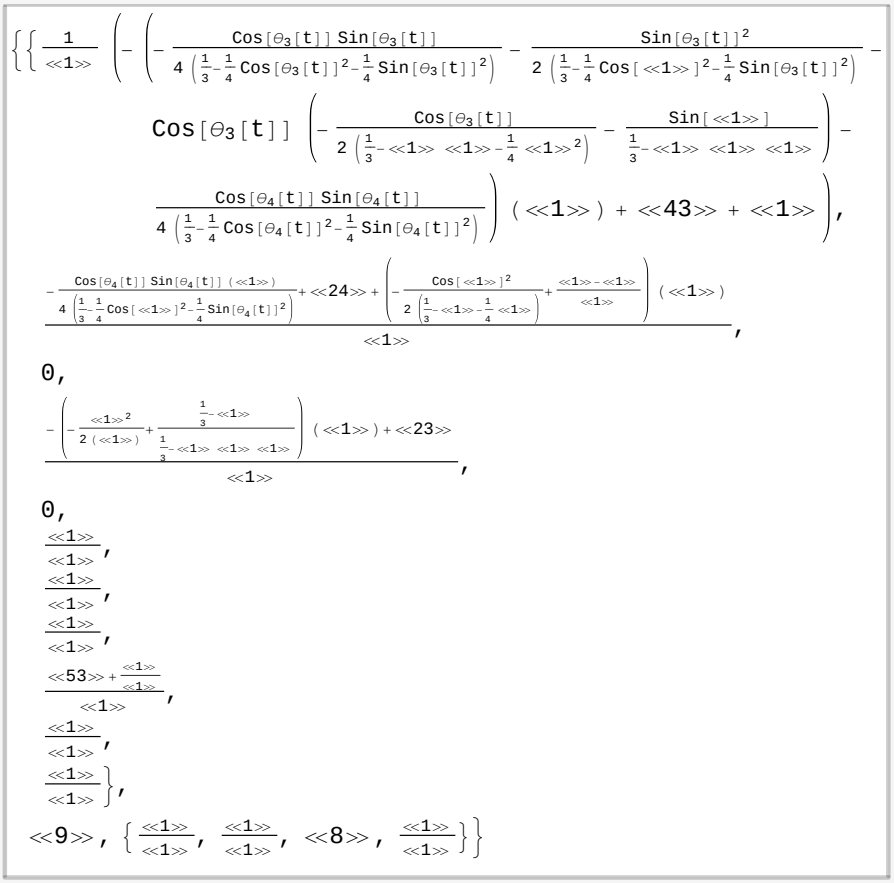
$$\left\{\{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}, \{0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\},\right.$$
$$\{0, 0, 48, 0, 0, 0, 0, 0, 0, 0, 0, 0\}, \{0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0\},$$
$$\{0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0\}, \{0, 0, 0, 0, 0, 48, 0, 0, 0, 0, 0, 0\},$$
$$\left\{0, 0, 0, 0, 0, 0, \frac{\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_3[t]]^2}{\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_3[t]]^2 - \frac{1}{4}\text{Sin}[\theta_3[t]]^2},\right.$$
$$\left. -\frac{\text{Cos}[\theta_3[t]]\,\text{Sin}[\theta_3[t]]}{4\left(\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_3[t]]^2 - \frac{1}{4}\text{Sin}[\theta_3[t]]^2\right)}, \frac{\text{Sin}[\theta_3[t]]}{2\left(\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_3[t]]^2 - \frac{1}{4}\text{Sin}[\theta_3[t]]^2\right)}, 0, 0, 0\right\},$$
$$\left\{0, 0, 0, 0, 0, 0, -\frac{\text{Cos}[\theta_3[t]]\,\text{Sin}[\theta_3[t]]}{4\left(\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_3[t]]^2 - \frac{1}{4}\text{Sin}[\theta_3[t]]^2\right)},\right.$$
$$\left.\frac{\frac{1}{3} - \frac{1}{4}\text{Sin}[\theta_3[t]]^2}{\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_3[t]]^2 - \frac{1}{4}\text{Sin}[\theta_3[t]]^2}, -\frac{\text{Cos}[\theta_3[t]]}{2\left(\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_3[t]]^2 - \frac{1}{4}\text{Sin}[\theta_3[t]]^2\right)}, 0, 0, 0\right\},$$
$$\left\{0, 0, 0, 0, 0, 0, \frac{\text{Sin}[\theta_3[t]]}{2\left(\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_3[t]]^2 - \frac{1}{4}\text{Sin}[\theta_3[t]]^2\right)},\right.$$
$$\left. -\frac{\text{Cos}[\theta_3[t]]}{2\left(\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_3[t]]^2 - \frac{1}{4}\text{Sin}[\theta_3[t]]^2\right)}, \frac{1}{\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_3[t]]^2 - \frac{1}{4}\text{Sin}[\theta_3[t]]^2}, 0, 0, 0\right\},$$
$$\left\{0, 0, 0, 0, 0, 0, 0, 0, 0, \frac{\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_4[t]]^2}{\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_4[t]]^2 - \frac{1}{4}\text{Sin}[\theta_4[t]]^2},\right.$$
$$\left. -\frac{\text{Cos}[\theta_4[t]]\,\text{Sin}[\theta_4[t]]}{4\left(\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_4[t]]^2 - \frac{1}{4}\text{Sin}[\theta_4[t]]^2\right)}, \frac{\text{Sin}[\theta_4[t]]}{2\left(\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_4[t]]^2 - \frac{1}{4}\text{Sin}[\theta_4[t]]^2\right)}\right\},$$
$$\left\{0, 0, 0, 0, 0, 0, 0, 0, 0, -\frac{\text{Cos}[\theta_4[t]]\,\text{Sin}[\theta_4[t]]}{4\left(\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_4[t]]^2 - \frac{1}{4}\text{Sin}[\theta_4[t]]^2\right)},\right.$$
$$\left.\frac{\frac{1}{3} - \frac{1}{4}\text{Sin}[\theta_4[t]]^2}{\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_4[t]]^2 - \frac{1}{4}\text{Sin}[\theta_4[t]]^2}, -\frac{\text{Cos}[\theta_4[t]]}{2\left(\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_4[t]]^2 - \frac{1}{4}\text{Sin}[\theta_4[t]]^2\right)}\right\},$$
$$\left\{0, 0, 0, 0, 0, 0, 0, 0, 0, \frac{\text{Sin}[\theta_4[t]]}{2\left(\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_4[t]]^2 - \frac{1}{4}\text{Sin}[\theta_4[t]]^2\right)},\right.$$
$$\left.\left. -\frac{\text{Cos}[\theta_4[t]]}{2\left(\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_4[t]]^2 - \frac{1}{4}\text{Sin}[\theta_4[t]]^2\right)}, \frac{1}{\frac{1}{3} - \frac{1}{4}\text{Cos}[\theta_4[t]]^2 - \frac{1}{4}\text{Sin}[\theta_4[t]]^2}\right\}\right\}$$

Jt = Transpose[J]

$$\{\{0, 0, 0, -1, 0, 1, 0, 0, 0, 0, 0, 0\}, \{1, 0, 0, -1, 0, 0, 1, 0, 0, 0, 0, 0\},$$
$$\{0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0\}, \{0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0\},$$
$$\{0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0\}, \{0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0\},$$
$$\{0, 0, 0, 0, 0, -1, 0, 0, 0, 1, 0, 0\}, \{0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 1, 0\},$$
$$\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -\text{Cos}[\theta_3[t]], -\text{Sin}[\theta_3[t]]\},$$
$$\{0, 0, 0, 0, 0, 0, 0, 0, -1, 0, -1, 0\}, \{0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, -1\},$$
$$\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \text{Sin}[\theta_4[t]], -\text{Cos}[\theta_4[t]]\}\}$$

```
JMinJt = Inverse[J.MIn.Jt]
```

A very large output was generated. Here is a sample of it:

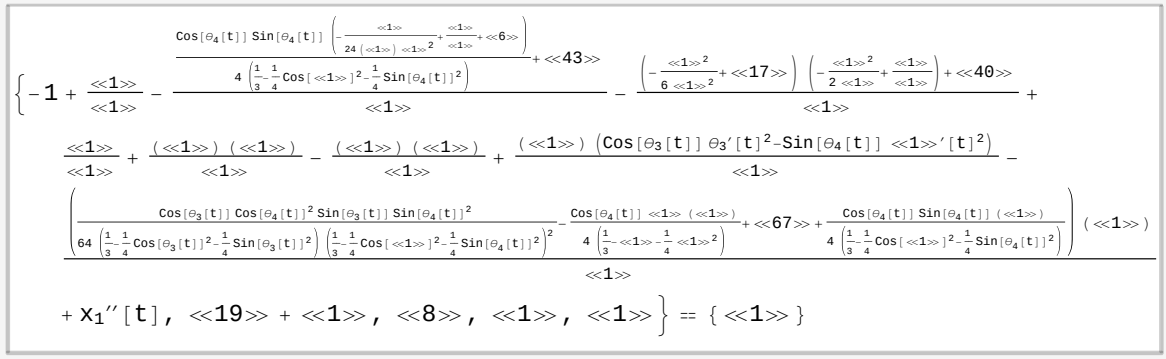


Show Less | Show More | Show Full Output | Set Size Limit...

```
(*q''=MIn.Qe+MIn.Jt.(JMinJt.(Qd-J.MIn.Qe))*)
```

```
eqs =
 ∂t ∂t q - (MIn.Qe + MIn.Jt.(JMinJt.(Qd - J.MIn.Qe))) == {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
(*eqns1 = ∂t∂tq - (MIn.Qe+MIn.Jt.(JMinJt.(Qd-J.MIn.Qe)))*)
```

A very large output was generated. Here is a sample of it:



Show Less | Show More | Show Full Output | Set Size Limit...

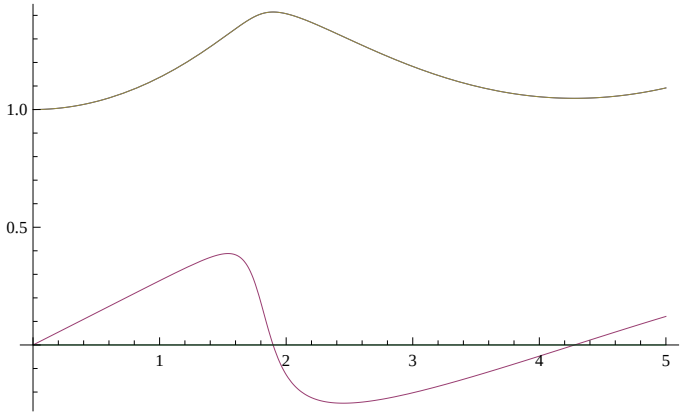
```
(*λ = JMinJt.(Qd-J.MIn.Qe)*)
(*Qc = {Q1[t],Q2[t],Q3[t],Q4[t],Q5[t],Q6[t],Q7[t],Q8[t],Q9[t],Q10[t],Q11[t],Q12[t]}
   eqns2 = Qc -Jt.JMInJt.(Qd-J.MIn.Qe)*)
(*eqns2 = Qc- (Qe- M.Inverse[J].Qd)*)

(*eqns=Join[eqns1,eqns2]=={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}*)
```

```
s = NDSolve[{eqs, x1[0] == 1, y1[0] == 0, θ1[0] == 0, x1'[0] == 0, y1'[0] == 0,
    θ1'[0] == 0, x2[0] == 0, y2[0] == 1, θ2[0] == 0, x2'[0] == 0, y2'[0] == 0, θ2'[0] == 0,
    x3[0] == 1, y3[0] == 0, θ3[0] == 0, x3'[0] == 0, y3'[0] == 0, θ3'[0] == 0,
    x4[0] == 0, y4[0] == 1, θ4[0] == 0, x4'[0] == 0, y4'[0] == 0, θ4'[0] == 0},
   {x1, y1, θ1, x2, y2, θ2, x3, y3, θ3, x4, y4, θ4, x1', y1', θ1', x2', y2', θ2', x3', y3',
    θ3', x4', y4', θ4'}, {t, 0, 5}, Method -> {"EquationSimplification" -> "Solve"}]
```

```
{{x1 → InterpolatingFunction[{{0., 5.}}, <>],
  y1 → InterpolatingFunction[{{0., 5.}}, <>],
  θ1 → InterpolatingFunction[{{0., 5.}}, <>],
  x2 → InterpolatingFunction[{{0., 5.}}, <>],
  y2 → InterpolatingFunction[{{0., 5.}}, <>],
  θ2 → InterpolatingFunction[{{0., 5.}}, <>],
  x3 → InterpolatingFunction[{{0., 5.}}, <>],
  y3 → InterpolatingFunction[{{0., 5.}}, <>],
  θ3 → InterpolatingFunction[{{0., 5.}}, <>],
  x4 → InterpolatingFunction[{{0., 5.}}, <>],
  y4 → InterpolatingFunction[{{0., 5.}}, <>],
  θ4 → InterpolatingFunction[{{0., 5.}}, <>],
  x1' → InterpolatingFunction[{{0., 5.}}, <>],
  y1' → InterpolatingFunction[{{0., 5.}}, <>],
  θ1' → InterpolatingFunction[{{0., 5.}}, <>],
  x2' → InterpolatingFunction[{{0., 5.}}, <>],
  y2' → InterpolatingFunction[{{0., 5.}}, <>],
  θ2' → InterpolatingFunction[{{0., 5.}}, <>],
  x3' → InterpolatingFunction[{{0., 5.}}, <>],
  y3' → InterpolatingFunction[{{0., 5.}}, <>],
  θ3' → InterpolatingFunction[{{0., 5.}}, <>],
  x4' → InterpolatingFunction[{{0., 5.}}, <>],
  y4' → InterpolatingFunction[{{0., 5.}}, <>],
  θ4' → InterpolatingFunction[{{0., 5.}}, <>]}}
```

```
Plot[Evaluate[{x3[t], x3'[t] , y2[t], θ1[t]} /. s], {t, 0, 5}, PlotStyle → Automatic]
```



```
θ3[2] /. s
J /. t → 2 /. s
λ = JMinJt.(Qd - J.MIn.Qe)
```

```
{0.886271}
```

```
{{{0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0},
   {-1, -1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0}, {0, 0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 0},
   {1, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0}, {0, 1, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0},
   {0, 0, 0, 1, 0, 0, 0, 0, 0, -1, 0, 0}, {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, -1, 0},
   {0, 0, 0, 0, 0, 0, 1, 0, -0.632306, -1, 0, -0.774719},
   {0, 0, 0, 0, 0, 0, 0, 1, -0.774719, 0, -1, -0.632306}}}
```

A very large output was generated. Here is a sample of it:

$$\left\{\frac{1}{\ll 1\gg}\left(-\left(-\frac{\cos[\theta_4[t]]^2}{2\left(\frac{1}{3}-\frac{1}{4}\cos[\theta_4[t]]^2-\frac{1}{4}\sin[\theta_4[t]]^2\right)}+\frac{\frac{1}{3}-\frac{1}{4}\sin[\theta_4[t]]^2}{\frac{1}{3}-\frac{1}{4}\cos[\theta_4[t]]^2-\frac{1}{4}\sin[\theta_4[t]]^2}\right)\right.\right.$$
$$\left(\frac{\cos[\theta_4[t]]\sin[\theta_4[t]]\left(\ll 32\gg+\frac{\ll 1\gg}{\ll 1\gg}\right)}{4\left(\frac{1}{3}-\frac{1}{4}\cos[\theta_4[t]]^2-\frac{1}{4}\sin[\theta_4[t]]^2\right)}-\right.$$
$$\left.\left.\left(\frac{\cos[\ll 1\gg]^2}{9(\ll 1\gg)^3}-\frac{\ll 1\gg}{6\ll 1\gg}+\ll 41\gg+\frac{\ll 1\gg^2\ll 1\gg^2}{16(\ll 1\gg)(\ll 1\gg)}\right)(\ll 1\gg)\right)+\ll 23\gg\right)-$$
$$\frac{-(\ll 1\gg)\left(-\frac{1}{9(\ll 1\gg)^2}+\frac{\ll 1\gg}{\ll 1\gg}+\frac{\ll 1\gg}{\ll 1\gg}+\frac{\cos[\theta_3[t]]\ll 2\gg\sin[\theta_4[t]]}{16(\ll 1\gg)\left(\frac{1}{3}-\frac{1}{4}\ll 1\gg^2-\frac{1}{4}\ll 1\gg^2\right)}\right)+\ll 60\gg}{\ll 1\gg}+$$
$$\frac{(\ll 1\gg)\left(-\sin[\theta_3[t]]\ll 1\gg'[t]^2-\cos[\ll 1\gg[t]]\ll 1\gg\right)}{\ll 1\gg}+$$
$$\frac{\left(\ll 43\gg+\frac{\ll 1\gg}{\ll 1\gg}\right)\left(\cos[\theta_3[t]]\theta_3'[t]^2-\sin[\theta_4[t]]\theta_4'[t]^2\right)}{\ll 1\gg},$$
$$\ll 1\gg, 0, \ll 6\gg, \ll 1\gg,$$
$$\left.\frac{\ll 1\gg}{\ll 1\gg}-\frac{\ll 1\gg}{\ll 1\gg}+\frac{\ll 1\gg}{\ll 1\gg}+\frac{(\ll 1\gg)(\ll 1\gg\ll 1\gg-\ll 1\gg)}{\ll 1\gg}\right\}$$

Show Less | Show More | Show Full Output | Set Size Limit...

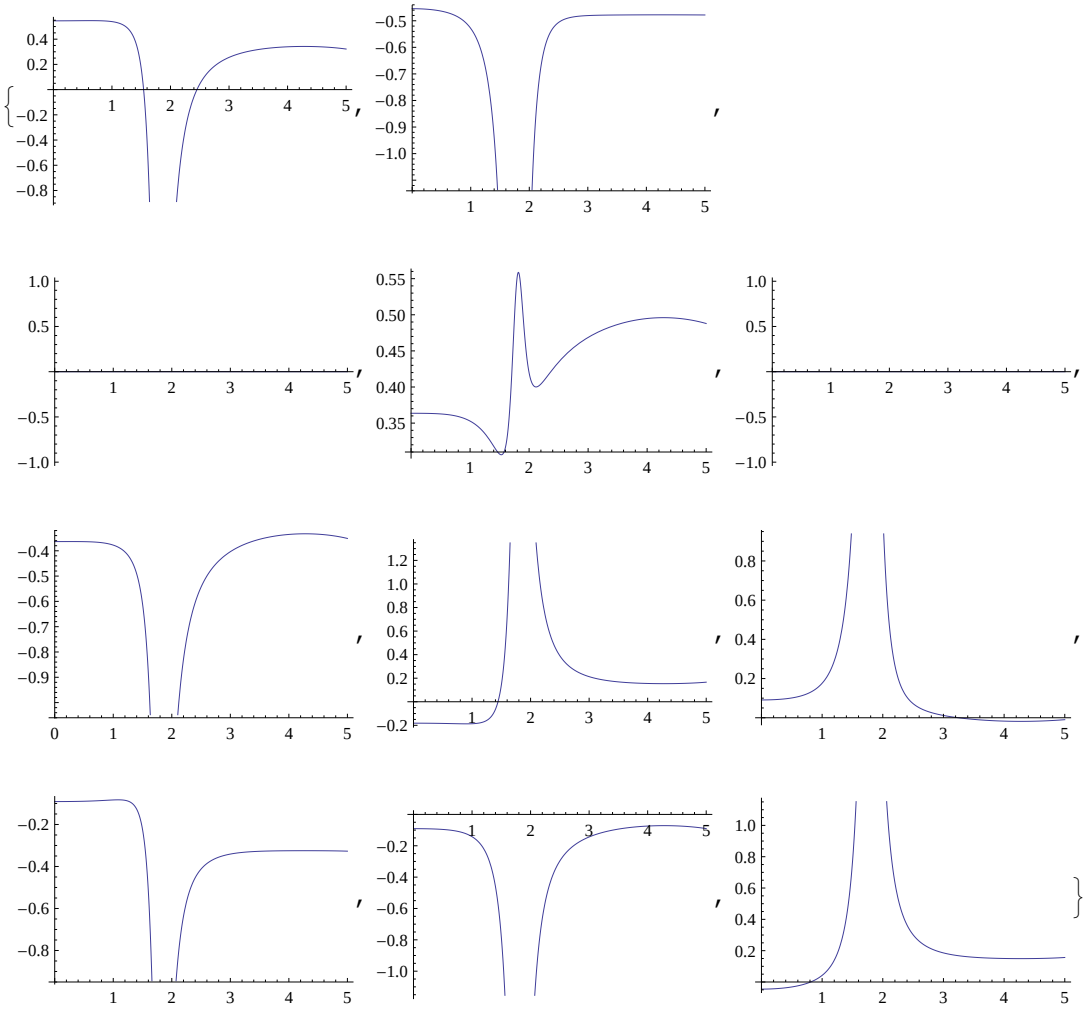
```
λ /. t → 1 /. s
```

```
{{0.539284, -0.528874, 0, 0.352806, 0, -0.377552,
  -0.186478, 0.176067, -0.0831644, -0.141989, 0.0421465}}
```

```
Dimensions[λ]
(*This should come out to be 11, and not 12*)
```

```
{11}
```

```
Table[Plot[Evaluate[λ[[j]] /. s], {t, 0, 5}, PlotStyle → Automatic], {j, 1, 11}]
```



Appendix B

Tools Developed

B.1 Quaternion Based Loop-closure

Usually loop-closure equations are written using transformation matrices. These loop closure equations are of the form:

$$ {}^{0}T_{1}^{1}T_{2}^{2}T_{3}...^{n-1}T_{n}^{n}T_{0} = [I] \tag{B.1} $$

In such equations as Equation (B.1), we have 16 parameters, while only 6 are independent. Hence, a redundancy of 10. We suggest using Quaternions instead of rotation/transformation matrices, as they have a redundancy of only 1. They can offer a much better computational performance.

The set of equations when using quaternions would be:

$$ \begin{aligned} {}^{0}Q_{1}^{1}Q_{2}^{2}Q_{3}...^{n-1}Q_{n}^{n}Q_{0} &= 1 + 0\hat{i} + 0\hat{j} + 0\hat{k} \\ \vec{r}_{0}^{0} = \vec{r}_{n}^{0} + [{}^{0}R_{n}]\vec{s}_{0}^{n} &= [ooo]^{T} \end{aligned} \tag{B.2} $$

In equation (B.2), for rotation part of loop-closure, we have used quaternions, while translation part of loop-closure is captured by vectors. The total parameters are 7 (4 for quaternions and 3 for vectors); hence a redundancy of 1.

A SymPy based Symbolic library for quaternions was developed which has been included in Appendix C. Here, we demonstrate the usage of that library to solve the forward kinematics of Bennet Mechanism:

```
from sympy.abc import alpha, beta, a, b, c, d
from sympy import *
from sympy_quaternions import *
```

```
4  from sympy.vector import *
5
6  #Bennet Setup
7  n = 4 #Number of links/joints in the mechanism
8  a = 1
9  alpha = pi/8
10 #Inputting DH-parameters for this mechanism in the format (alpha, a, d). Theta
       is the variable and hence is not provided
11 dhp = ((alpha, a, 0), (-alpha, a, 0), (alpha, a, 0), (-alpha, a, 0))
12 ALPHA = (alpha, -alpha, alpha, -alpha)
13 A = (a, a, a, a)
14 THETA = list(symbols('theta0:' + str(n)))
15 X, Y, Z = symbols('X0:' + str(n)), symbols('Y0:' + str(n)), symbols('Z0:' + str(
       n))
16
17 #Initial Condition for FWD Kinematics
18 THETA[0] = pi/2
19
20 #Quaternion Composition for Rotation Loop-closure Equations
21 angle = pi/2
22 Q = [] #list of quaternions
23 for i in range(n):
24     Q.append(quat_around_axis(ALPHA[i], (1, 0, 0)))
25     Q.append(quat_around_axis(THETA[i], (0, 0, 1)))
26
27 q = qmult_many(Q) #Body-fixed rotation composition
28 equations = qoperate(q, (1,0,0,0), "-")
29
30 #Axes Transformation for Vector Loop-closure Equations
31 O = CoordSys3D('O')
32 B = O.orient_new_axis('B', 0, O.k) #Initiate body-frame 'B' by replicating space
       -frame 'O'
33 for i in range(n):
34     body_orienter = BodyOrienter(ALPHA[i], THETA[i], 0, 'XZX')
35     B = B.orient_new('B', (body_orienter, ), location=A[i]*B.i)
36
37 #express(B.position_wrt(O), O)
38 v = B.origin.express_coordinates(O)
39 equations = equations + v #Final set of equations is Ready
40
```

```
solve(equations, THETA[1:])
```

Code Sample B.1: Bennett Forward Kinematics using Quaternions

B.2 Automatic Constraint Equations Generation

In this section, we demonstrate how constraint equations in Natural Coordinates can be generated automatically using SymPy, for Bennett Mechanism:

```
from sympy.vector import CoordSys3D
from sympy.vector import Vector
N = CoordSys3D('N') #For natural constraints, we only this one global coordinate system

n = 4
R = [] #List of position vectors of points
U = [] #List of unit direction vectors
for i in range(n):
    r = symbols('r_'+str(i)+'x:z')
    u = symbols('u_'+str(i)+'x:z')
    R.append(r[0]*N.i + r[1]*N.j + r[2]*N.k)
    U.append(u[0]*N.i + u[1]*N.j + u[2]*N.k)

A = (1, 1, 1, 1)
ALPHA = (pi/8, -pi/8, pi/8, -pi/8)


def components(v):
    '''Returns components of a given vector as list'''
    return list(v.components.values())


def equations_link(r1, r2, u1, u2, a, alpha):
    '''This generates constraint equations for a rigid link,
    with 2 basic points and 2 unit direction vectors'''
    r12 = r1 - r2
    return (r12 & u1) - 0, (r12 & u2) - 0, (r12 & r12) - a**2, (u1 & u2) - cos(alpha)


def equations_unit_normal(u):
    '''This generates constraint equations based on the condition that a direction vector is unit'''
```

```
    return (u & u) - 1

EQUATIONS = []

#Appending equations applying rigid link and unit-normal constraints
for i in range(n):
    j = i + 1
    if j == n: j = 0 #Cyclic
    eq = list(equations_link(R[i], R[j], U[i], U[j], A[i], ALPHA[i]))
    EQUATIONS += eq
    EQUATIONS.append(equations_unit_normal(U[i]))

#Appending equations applying fixed position constraint on r0 and r1
p1 = A[0]*N.i #Position of r1
EQUATIONS += list((R[0]).components.values())
EQUATIONS += list((R[1] - p1).components.values())

#len(EQUATIONS)
for e in EQUATIONS:
    e
```

Code Sample B.2: Generating Constraint Equations using SymPy for Bennett Mechanism

B.3 Generating CAD Models

FreeCAD is an open-source CAD tool, which exposes its API using Python programming language. CADQuery is a higher level tool built on top of it, which can be used to programmatically create CAD models. A sample Myard Mechanism's link can be created using CADQuery, by giving the DH Parameters of the mechanism. The sample code is shown in:

```
# This example is meant to be used from within the CadQuery module of FreeCAD.
import cadquery as cq
from Helpers import show

#DH Parameters
a = 10.0
alpha = 30 #Angle in Degrees
d = 2.0 #0 for Bennett mechanism

joint_radius = a/50.0
joint_length = a/10.0
link_radius = a/100.0

```

```
14 if d > 0:
15     result = cq.Workplane("front").workplane() \
16                 .circle(joint_radius) \
17                 .extrude(joint_length) \
18                 .workplane() \
19                 .transformed(rotate=(0, 90, 0)) \
20                 .circle(link_radius) \
21                 .extrude(a) \
22                 .faces(">X") \
23                 .workplane() \
24                 .transformed(rotate=(0, 90, 0)) \
25                 .transformed(rotate=(alpha, 0, 0)) \
26                 .circle(link_radius) \
27                 .extrude(d) \
28                 .faces(">X") \
29                 .workplane() \
30                 .transformed(offset=(0, 0, -d)) \
31                 .circle(joint_radius) \
32                 .extrude(joint_length/2, both=True)
33
34 else:
35     result = cq.Workplane("front").workplane() \
36                 .circle(joint_radius) \
37                 .extrude(joint_length) \
38                 .workplane() \
39                 .transformed(rotate=(0, 90, 0)) \
40                 .circle(link_radius) \
41                 .extrude(a) \
42                 .faces(">X") \
43                 .workplane() \
44                 .transformed(rotate=(0, 90, 0)) \
45                 .transformed(rotate=(alpha, 0, 0)) \
46                 .circle(joint_radius) \
47                 .extrude(joint_length/2, both=True)
48
49 # Render the solid
50 show(result)
```

Code Sample B.3: Automatic CAD Model Generation

The code shown in Sample Code B.3 generates only one link of the whole mechanism. It can be used to iteratively generate all the links. A complete assembly is not yet possible, and is currently a work in progress.

A sample assembly of Bennett Mechnism was made in SolidWorks, after the generation of links using the above code:

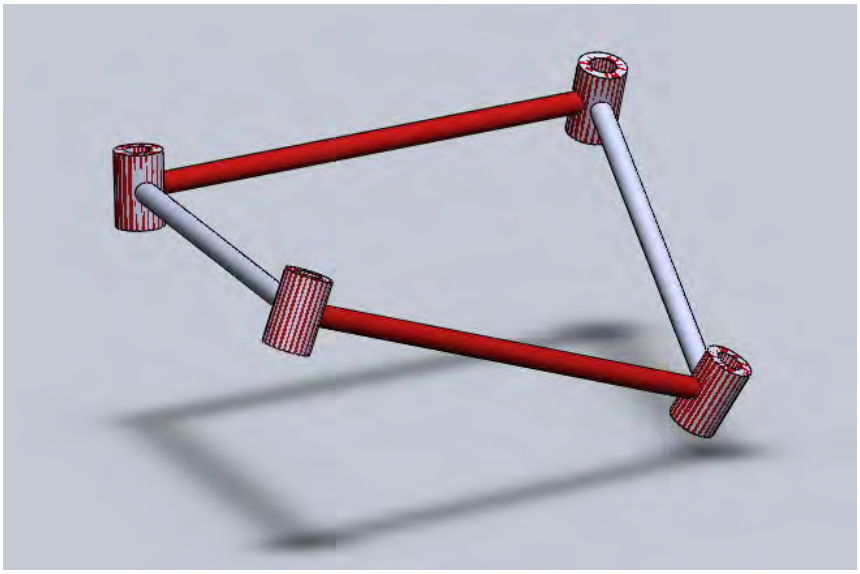


Figure B.1: Automatically generated Bennett Mechanism, rendered in SolidWorks

Appendix C

SymPy based Quaternion Library

SymPy is a Python framework for Symbolic Algebra. We developed a Quaternion Module, implementing some basic algebras of Quaternions (but not all, only what was relevant for us). The code has been derived from another Quaternion library in NumPy (a Python framework for numerical computations), and modified for use in SymPy.

```
''' Symbolic formulae for quaternions '''

from sympy import *
from sympy.matrices import Matrix


def qoperate(q1, q2, operator):
    '''Apply the given operator on the 2 quaternions
    operator : "+" or "-" are supported currently
    '''
    w1, x1, y1, z1 = q1
    w2, x2, y2, z2 = q2
    if operator == "+":
        q = w1+w2, x1+x2, y1+y2, z1+z2
    if operator == "-":
        q = w1-w2, x1-x2, y1-y2, z1-z2

    return q


#Send the quaternions in reverse order, this is what I noticed.
#I mean the first quaternion should be the second rotation applied in succession
```

```
#Okay, I got it. They are doing body-fixed rotation by default. So, it's correct.
def qmult(q1, q2):
    ''' Multiply two quaternions

    Parameters
    ----------
    q1 : 4 element sequence
    q2 : 4 element sequence

    Returns
    -------
    q12 : shape (4,) array

    Notes
    -----
    See : http://en.wikipedia.org/wiki/Quaternions#Hamilton_product
    '''
    w1, x1, y1, z1 = q1
    w2, x2, y2, z2 = q2
    w = w1*w2 - x1*x2 - y1*y2 - z1*z2
    x = w1*x2 + x1*w2 + y1*z2 - z1*y2
    y = w1*y2 + y1*w2 + z1*x2 - x1*z2
    z = w1*z2 + z1*w2 + x1*y2 - y1*x2
    return w, x, y, z


def qmult_many(qlist):
    """Multiplies multiple quaternions in one go.

    :qlist: List/tuple of quaternions, in the normal order in which you wanna multiply them
    :returns: Resultant quaternion

    """
    q = qlist[0]
    for i in range(len(qlist)-1):
        q = qmult(q, qlist[i+1])
    return q


def quat_around_axis(theta, axis):
```

```
63      ''' Quaternion for rotation of angle `theta` around axis `axis`
64
65      Parameters
66      ----------
67      theta : symbol
68         angle of rotation
69      axis : 3 element sequence
70         vector (assumed normalized) specifying axis for rotation
71
72      Returns
73      -------
74      quat : 4 element sequence of symbols
75         quaternion giving specified rotation
76
77      Notes
78      -----
79      Formula from http://mathworld.wolfram.com/EulerParameters.html
80      '''
81      # axis vector assumed normalized
82      t2 = theta / 2.0
83      st2 = sin(t2)
84      return (cos(t2),
85              st2 * axis[0],
86              st2 * axis[1],
87              st2 * axis[2])
88
89
90  #I'm not using this as I can't figure out how yaw-pitch-roll related to X-Y-Z
        axes
91  #TODO: Modify this function so that it uses X-Y-Z and such
92  def quat_from_eulers(eulers):
93      """Creates a quaternion from a set of Euler angles.
94      Eulers are an array of length 3 in the following order::
95          [yaw, pitch, roll]
96      """
97      pitch, yaw, roll = eulers
98
99      halfPitch = pitch * 0.5
100     sP = sin(halfPitch)
101     cP = cos(halfPitch)
102
103     halfRoll = roll * 0.5
```

```
104     sR = sin(halfRoll)
105     cR = cos(halfRoll)
106 
107     halfYaw = yaw * 0.5
108     sY = sin(halfYaw)
109     cY = cos(halfYaw)
110 
111     return (-cY * sP * cR) - (sY * cP * sR),\
112             (cY * sP * sR) - (sY * cP * cR),\
113             (sY * sP * cR) - (cY * cP * sR),\
114             (cY * cP * cR) + (sY * sP * sR)
115 
116 
117 def quat2axis(quat):
118     ''' Angle-axis from Quaternion
119 
120     Parameters
121     ----------
122     theta : symbol
123        angle of rotation
124     axis : 3 element sequence
125        vector (assumed normalized) specifying axis for rotation
126 
127     Returns
128     -------
129     quat : 4 element sequence of symbols
130        quaternion giving specified rotation
131 
132     Notes
133     -----
134     Formula from http://mathworld.wolfram.com/EulerParameters.html
135     '''
136     qw, qx, qy, qz = quat
137     angle = simplify(2 * acos(qw))
138     x = simplify(qx / sqrt(1-qw*qw))
139     y = simplify(qy / sqrt(1-qw*qw))
140     z = simplify(qz / sqrt(1-qw*qw))
141     return (angle, x, y, z)
142 
143 
144 def quat2mat(quat):
145     ''' Symbolic conversion from quaternion to rotation matrix
```

```
146
147     For a unit quaternion
148
149     From: http://en.wikipedia.org/wiki/Rotation_matrix#Quaternion
150     '''
151     w, x, y, z = quat
152     return Matrix([
153             [1 - 2*y*y-2*z*z, 2*x*y - 2*z*w, 2*x*z+2*y*w],
154             [2*x*y+2*z*w, 1-2*x*x-2*z*z, 2*y*z-2*x*w],
155             [2*x*z-2*y*w, 2*y*z+2*x*w, 1-2*x*x-2*y*y]]).applyfunc(lambda i:
    trigsimp(i))
```

Code Sample C.1: Quaternion Module for Sympy

References

[1] Grigore Gogu. *[Part-1] Structural synthesis of parallel robots Pt. 1 : Methodology*. Springer, Dordrecht, 2008. 1

[2] B P Nagaraj. *Kinematic And Static Analysis Of Over-Constrained Mechanisms And Deployable Pantograph Masts*. PhD thesis, IISc, Sep 2009. 1

[3] Kenneth J. Waldron. Symmetric overconstrained linkages. ASME, 1968. 2

[4] K. J. Waldron. Overconstrained linkages. *Environment and planning B*, 6(4):393–402, 1979. 2

[5] Constantinos Mavroidis and Bernard Roth. 01 Analysis of overconstrained mechanisms. *Journal of mechanical design*, 117(1):69–74, 1995. 2

[6] C. Mavroidis and B. Roth. 02 New and revised overconstrained mechanisms. *Journal of Mechanical Design*, 117(1):75–82, 1995. 2

[7] Marek Wojtyra. Joint reactions in rigid body mechanisms with dependent constraints. *Mechanism and Machine Theory*, 44(12):2265–2278, Dec 2009. 2

[8] J Clerk Maxwell. L. on the calculation of the equilibrium and stiffness of frames. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 27(182):294–299, 1864. 3

[9] Marek Wojtyra and Janusz Frczek. Solvability of reactions in rigid multibody systems with redundant nonholonomic constraints. *Multibody System Dynamics*, 30(2):153–171, Aug 2013. 3, 5, 25

[10] E Bayo and Ragnar Ledesma. Augmented Lagrangian and mass-orthogonal projection methods for constrained multibody dynamics. *Nonlinear Dynamics*, 9(1):113–130, 1996. 3

[11] J. Garca de Jaln and Eduardo Bayo. *Kinematic and dynamic simulation of multibody systems*. Mechanical Engineering Series, Springer, New York, 1994. 3, 16, 51

[12] J. GarcadeJaln and M. D. Gutirrez-Lpez. Multibody dynamics with redundant constraints and singular mass matrix: existence, uniqueness, and determination of solutions for accelerations and constraint forces. *Multibody System Dynamics*, 30(3):311–341, Oct 2013. 3, 4

[13] Yundou Xu, Wenlan Liu, Jiantao Yao, and Yongsheng Zhao. A method for force analysis of the overconstrained lower mobility parallel mechanism. *Mechanism and Machine Theory*, 88:31–48, 2015. 4

[14] Wenlan Liu, Yundou Xu, Jiantao Yao, and Yongsheng Zhao. The weighted MoorePenrose generalized inverse and the force analysis of overconstrained parallel mechanisms. *Multibody System Dynamics*, 39(4):363–383, 2017. 4

[15] Marek Wojtyra and Janusz Frczek. Comparison of Selected Methods of Handling Redundant Constraints in Multibody Systems Simulations. *Journal of Computational and Nonlinear Dynamics*, 8(2):021007, Jul 2012. 4

[16] Francisco Gonzlez and Jzsef Kvecses. Use of penalty formulations in dynamic simulation and analysis of redundantly constrained multibody systems. *Multibody System Dynamics*, 29(1):57–76, Jan 2013. 4

[17] Bilal Ruzzeh and Jzsef Kvecses. A Penalty Formulation for Dynamics Analysis of Redundant Mechanical Systems. *Journal of Computational and Nonlinear Dynamics*, 6(2):021008, 2011. 4

[18] Z. M. Bi and Bongsu Kang. An Inverse Dynamic Model of Over-Constrained Parallel Kinematic Machine Based on NewtonEuler Formulation. *Journal of Dynamic Systems, Measurement, and Control*, 136(4):041001–041001–9, Mar 2014. 4

[19] E. Zahariev and J. Cuadrado. Dynamics of over-constrained rigid and flexible multibody systems. In *12th IFToMM World Congress, Besanon, France*, 2007. 4

[20] Wen-Lan Liu, Yun-Dou Xu, Jian-Tao Yao, and Yong-Sheng Zhao. Methods for Force Analysis of Overconstrained Parallel Mechanisms: A Review. *Chinese Journal of Mechanical Engineering*, 30(6):1460–1472, 2017. 4

[21] 2012 SOLIDWORKS Help - Redundant Constraints. 4

[22] 2012 SOLIDWORKS Help - Redundant Mate Force Results. 4

[23] Why does my model not output the expected results when I have redundant constraints in SimMechanics 3.1.1 (R2009b)? - MATLAB Answers - MATLAB Central. 4

[24] Shin-Min Song and Xiaochun Gao. The mobility equation and the solvability of joint forces/torques in dynamic analysis. *Journal of Mechanical Design*, 114(2):257–262, 1992. 5

[25] Marek Wojtyra. Joint reaction forces in multibody systems with redundant constraints. *Multibody System Dynamics*, 14(1):23–46, 2005. 5, 23, 24

[26] Wojciech Blajer. On the Determination of Joint Reactions in Multibody Mechanisms. *Journal of Mechanical Design*, 126(2):341, 2004. 5, 15

[27] Javier Garca Jaln. Twenty-five years of natural coordinates. *Multibody System Dynamics*, 18(1):15–33, Jun 2007. 5, 13

[28] Thomas Uchida, Alfonso Callejo, Javier GarcadeJaln, and John McPhee. On the Grbner basis triangularization of constraint equations in natural coordinates. *Multibody System Dynamics*, 31(3):371–392, Mar 2014. 5

[29] Cheng Liu, Qiang Tian, Haiyan Hu, and Daniel Garca-Vallejo. Simple formulations of imposing moments and evaluating joint reaction forces for rigid-flexible multibody systems. *Nonlinear Dynamics*, 69(1-2):127–147, Jul 2012. 5

[30] Adam Czaplicki. Are natural coordinates a useful tool in modeling planar biomechanical linkages? *Journal of Biomechanics*, 40(10):2307–2312, Jan 2007. 5

[31] Eric W. Weisstein. Quaternion. 10, 11

[32] Parviz E. Nikravesh. *Computer-aided analysis of mechanical systems*. Prentice-Hall, Englewood Cliffs, N.J, 1988. 11, 14, 16

[33] Dan Negrut and Andrew Dyer. ADAMS/Solver Primer. *Ann Arbor*, 2004. 11, 13

[34] Ashitava Ghosal. *Robotics: Fundamental Concepts and Analysis*. Oxford University Press, Feb 2006. 13, 15, 16

[35] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. 13

[36] Wojciech Blajer and Adam Czaplicki. An alternative scheme for determination of joint reaction forces in human multibody models. *Journal of Theoretical and Applied Mechanics*, 43, 2005. 15

[37] CHEN Yan. *Design of Structural Mechanisms*. PhD thesis, 2003. 20

[38] J Eddie Baker. The Bennett, Goldberg and Myard linkagesin perspective. *Mechanism and Machine Theory*, 14(4):239–253, 1979. 20

[39] J Eddie Baker. An analysis of the Bricard linkages. *Mechanism and Machine Theory*, 15(4):267–286, 1980. 20

[40] W. Wunderlich. On Burmester focal mechanism and Hart's straight-line motion. 1967. 20, 31

[41] Paulo Flores. *Concepts and Formulations for Spatial Multibody Dynamics*. SpringerBriefs in Applied Sciences and Technology. Springer International Publishing, Cham, 2015. 21

[42] Ahmed A. Shabana. *Computational dynamics*. John Wiley & Sons, Chichester, West Sussex ; Hoboken, 3rd ed edition, 2010. 22

[43] Gilbert Strang. *Introduction to linear algebra*. Wellesley-Cambridge Press, Wellesley, Mass, 3. ed., rev. internat. ed edition, 2005. OCLC: 255007465. 23

[44] Amit Kumar Sinhababu. *Testing algebraic independence of polynomials over finite fields*. PhD thesis, Indian Institute of Technology Kanpur, 2014. 51