CONSTRAINT ANALYSIS OF ASSEMBLIES USING SCREW
THEORY AND TOLERANCE SENSITIVITIES

by

Daniel K. Smith

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering
Brigham Young University
December 2001
BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Daniel K. Smith

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date ___________________________ Kenneth W. Chase, Chair

Date ___________________________ Larry L. Howell

Date ___________________________ Alan R. Parkinson
BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Daniel K. Smith in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative material including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Kenneth W. Chase
Chair, Graduate Committee

Brent L. Adams
Graduate Coordinator

Douglas M. Chabries
Dean, College of Engineering and Technology
ABSTRACT

CONSTRAINT ANALYSIS OF ASSEMBLIES USING SCREW THEORY AND TOLERANCE SENSITIVITIES

Daniel K. Smith
Department of Mechanical Engineering
Master of Science

Constraint problems in assemblies due to parts with underconstrained or overconstrained degrees of freedom need to be identified so they can be controlled by designers. The Variation-based Constraint Analysis of Assemblies (VCAA) combines tolerance analysis and constraint analysis, making it possible to perform both analyses on an assembly simultaneously. The variation sensitivity matrices used in the Direct Linearization Method (DLM) of tolerance analysis contain constraint information that can be interpreted using screw theory methods.

Previous work on screw theory constraint analysis is connected to tolerance analysis through the VCAA. The variation sensitivity matrices contain screw matrices that can be analyzed for under- and overconstraints. The sensitivities, calculated through
the Global Coordinate Method, can show if parts have underconstrained, mobile degrees of freedom, or if the parts have redundant, overconstrained degrees of freedom. Using the steps outlined in the VCAA method, it is possible to extract screw matrices directly from the sensitivity matrices in order to analyze assemblies.

The development of the VCAA method is outlined as well as the analysis of several case study assemblies. The VCAA method correctly identifies underconstrained degrees of freedom using the dependent variation sensitivity matrix and overconstraints using the geometric feature variation sensitivity matrix. The case studies show that after a DLM tolerance analysis is performed on an assembly, it can successfully be analyzed for constraint problems using the information gained from the tolerance analysis. The advantage of the VCAA method is this ability to perform a constraint analysis coincident with a variation analysis.
ACKNOWLEDGMENTS

I would like to express my utmost appreciation to Dr. Kenneth Chase who guided and helped me throughout my university experience. I would like to thank him for the many hours he spent on my behalf and the effort he expended in helping me with this research. He has been a wonderful professor, friend, and example to me in my life. Among those who also deserve my thanks are my graduate committee members, Dr. Alan Parkinson and Dr. Larry Howell. I would also like to acknowledge the ADCATS research group and the National Science Foundation, Grant DMI 0084880, for funding this research.

Additionally, it is essential that I express my love and appreciation to my wonderful wife, Mary Sue, my beautiful daughter, Elayna, my parents, and the rest of my terrific family. They never ceased to be supportive and helpful through this thesis and the rest of the joys and trials that come with student life. Their optimism, happiness, and affection have made the difference in my life. Also, I would especially like to express my appreciation to my Heavenly Father for His love, caring, and guidance through this, and all the stages of my life. Thank you!
# Table of Contents

Chapter 1. Introduction ................................................. 1
  1.1 Kinematics Introduction ........................................ 1
  1.2 Constraint Analysis Introduction .............................. 3
  1.3 Tolerance Analysis Introduction .............................. 5
  1.4 Thesis Objectives .............................................. 6
  1.5 Chapter Overviews .............................................. 7

Chapter 2. Constraint Analysis Background and Current Methods ........... 9
  2.1 Kinematics Review ............................................. 9
     2.1.1 Degrees of Freedom ..................................... 9
     2.1.2 Homogeneous Transformations ............................ 11
  2.2 Review of Current Constraint Methods ......................... 13
     2.2.1 Kinematic Constraint Pattern Analysis .................. 13
     2.2.2 Geometric Constraint Solving ............................ 17
     2.2.3 Screw Theory-Based Constraint Analysis ................. 21
  2.3 Tolerance Analysis Background ................................ 36

Chapter 3. Constraint Analysis using Screw Theory and Tolerance Sensitivities .... 49
  3.1 Current Tolerance DOFs Analysis .............................. 49
  3.2 Variation-based Constraint Analysis of Assemblies (VCAA) .......... 54
     3.2.1 Underconstrained Motion Analysis ....................... 54
        3.2.1.1 Kinematic Velocity and Tolerance Variation Analogy . 54
3.2.1.2 Variation Matrix Twist Formation .................. 59
3.2.1.3 Variation Twist Analysis of Assemblies .............. 63
3.2.1.4 Application to One-way Clutch Example .............. 71
3.2.1.5 Conclusions for Underconstrained Motion Analysis using VCAA ........................................ 73
3.2.2 Overconstrained DOFs Analysis ........................... 73
3.2.2.1 Geometric Variation and Force-Moment Relationship ......................................................... 74
3.2.2.2 Variation Matrix Wrench Formation .................. 76
3.2.2.3 Variation Wrench Formation for Assemblies ...... 80
3.2.2.4 Application to the One-way Clutch Example ......... 87
3.2.2.5 Conclusions for Overconstrained DOF Analysis using VCAA .............................................. 89
3.3 Conclusion of Constraint Analysis using Tolerance Methods .................................................. 90

Chapter 4. VCAA Examples for 2-D and 3-D Assemblies ...................... 93
4.1 Variation-based Constraint Analysis of Assemblies Methodology .... 93
4.1.1 VCAA Modeling Requirements ............................. 93
4.1.2 VCAA Methodology and Analysis Steps .................. 95
4.1.2.1 Underconstraint Analysis Steps ....................... 96
4.1.2.2 Overconstraint Analysis Steps ......................... 97
4.1.3 VCAA Interpretation of Results ........................... 100
4.2 2-D Examples of Constraint Analysis using the VCAA Method .......... 103
4.2.1 Two Cylinder Slider Plate Assembly ..................... 103
4.2.2 Three Cylinder Slider Plate Assembly ................... 109
4.2.3 Stacked Blocks Assembly ................................... 115
4.3 3-D Examples of Constraint Analysis using the VCAA Method .......... 122
4.3.1 Crank Slider Assembly ........................................ 122
4.3.2 Swash Plate Assembly ....................................... 132
4.4 Conclusions of the VCAA Example Problems ...................... 139
Chapter 5. Comparisons and Results of VCAA Method .......................... 141
  5.1 Comparisons of VCAA to Previous Constraint Analysis Methods .... 141
  5.2 Results of the VCAA Method .............................................. 148

Chapter 6. Recommendations and Conclusions ............................... 151
  6.1 Contributions of the VCAA Method ..................................... 151
  6.2 Conclusions of the VCAA Method ....................................... 152
  6.3 Recommendations for Future Work ..................................... 153

Appendix A. Derivation of Global Coordinate Method .......................... 157
  A.1 Derivation with Respect to a Length Variable ......................... 158
  A.2 Derivative with Respect to a Rotation Variable ..................... 159
  A.3 Geometric Interpretation of the Derivatives ......................... 160

Appendix B. Explanation of 3-D Point Algorithm .............................. 165

Appendix C. MATLAB® Files Used in VCAA .................................. 171

Bibliography ................................................................. 195
LIST OF TABLES

Table 2.1 - 3-D Kinematic Joint Types ................................................. 11
Table 2.2 - Global Coordinate Method Derivative Equations ...................... 43
Table 3.1 - 2-D and 3-D Kinematic Joint Types ..................................... 50
Table 3.2 - Constraint Results from [B] and Equation Comparison ............... 52
Table 3.3 - Comparisons of Tolerance and Kinematic Analyses .................. 55
Table 3.4 - Nominal Dimensions of One-way Clutch .................................. 61
Table 4.1 - Nominal Dimensions of Two Cylinder Slider Plate Assembly ........ 104
Table 4.2 - Nominal Dimensions of Three Cylinder Slider Plate Assembly ...... 110
Table 4.3 - Nominal Dimensions of Stacked Blocks Assembly ...................... 116
Table 4.4 - Nominal Dimensions of Crank Slider Assembly (θ = 45°) ............. 123
Table 4.5 - Nominal Dimensions for Swash Plate Assembly (θ = 180°) ........... 133
Table 5.1 - Nominal Dimensions of Stacked Blocks Assembly .................... 143
Table A.1 - Derivatives with Respect to a Length Variable ....................... 158
Table A.2 - Derivatives with Respect to a Rotation Variable .................... 160
Table A.3 - Derivatives with Respect to a Length Variable ....................... 161
Table A.4 - Derivatives with Respect to a Rotation Variable .................... 163
LIST OF FIGURES

Figure 1.1 - Example of limitations to Grüber's Equation, Five-bar Mechanism ........ 2
Figure 1.2 - Overconstraint Example, Door with Three Hinges ......................... 4
Figure 1.3 - Underconstraint Example, Crank Slider Assembly .......................... 5
Figure 2.1 - Body with six DOFs, in 3-D .................................................. 10
Figure 2.2 - 2D body constrained by two C's resulting in one R. Modified from
[Blanding 1999, p. 11]. ............................................................................. 14
Figure 2.3 - Body showing desired constraints and disk of radial lines. Modified from
[Blanding 1999, p. 43]. ............................................................................. 15
Figure 2.4 - Two final design concepts created using Constraint Pattern Analysis.
Modified from [Blanding 1999, p. 102]. ................................................. 16
Figure 2.5 - Evaluation of four situations of given geometric elements (lines) and the
given geometric constraints (angles and g) ............................................. 17
Figure 2.6 - Graph-based representation of elements $a$, $A$, and $B$ with constraint length $d$.
Modified from [Hoffmann and Vermeer 1995, p. 272]. .............................. 19
Figure 2.7 - A well-constrained assembly graph and its constraint graph with three
clusters. Modified from [Hoffmann and Vermeer 1995, pp. 274-275]. ........... 20
Figure 2.8 - Rotation of XYZ around Screw Axis to $X'Y'Z'$ by angle $\theta$ ............ 22
Figure 2.9 - Screw Rotation and Translation applied to a 3D Rigid Body ............... 23
Figure 2.10 - Twist screw element of angular and linear velocities ..................... 25
Figure 2.11 - Wrench screw element of resultant force and moment .................. 26
Figure 2.12 - Pin Joint and Pin-Slot Assembly. Taken from [Adams 1998, p. 92]. .... 33
Figure 2.13 - Kinematic adjustment due to component dimensional variation. Taken
from [Chase, Gao, Magleby, and Sorensen 1996, p. 797]. ...................... 37
Figure 2.14 - Adjustment due to geometric shape variations. Taken from [Chase, Gao, Magleby, and Sorensen 1996, p. 798] ......................................................... 38

Figure 3.1 - Comparison Results: Under-, Over-, and Exactly Constrained Assemblies .......................................................................................... 53

Figure 3.2 - Comparison Results: Indeterminate Assembly .......... 53

Figure 3.3 - One-way Clutch Assembly. Modified from [Faerber 1999]. ................. 56

Figure 3.4 - Vector loop used to solve the One-way Clutch Assembly. Taken from [Faerber 1999]. ................................................................. 56

Figure 3.5 - Vector Loop Model of One-way Clutch Assembly. Taken from [Gao 1993] .................................................................................. 61

Figure 3.6 - Joint Twist with 0 DOFs in 2-D ................................. 64

Figure 3.7 - Joint Twists with 1 DOF in 2-D ................................. 65

Figure 3.8 - Joint Twists with 2 DOFs in 2-D ............................... 66

Figure 3.9 - Joint Twist with 0 DOFs in 3-D ................................. 66

Figure 3.10 - Joint Twists with 1 DOF in 3-D ............................... 67

Figure 3.11 - Joint Twists with 2 DOFs in 3-D ............................... 67

Figure 3.12 - Joint Twists with 3 DOFs in 3-D ............................... 68

Figure 3.13 - Joint Twists with 4 DOFs in 3-D ............................... 69

Figure 3.14 - Joint Twists with 5 DOFs in 3-D ............................... 70

Figure 3.15 - DOFs for Kinematic Motions (k) and Geometric Feature Variations (f).

Modificed from [Chase, Gao, Magleby, and Sorensen 1996] .................. 75

Figure 3.16 - Vector loop model of One-way Clutch with geometric tolerance callouts included. ................................................................. 78

Figure 3.17 - Joint Wrench with 3 Constrained DOFs in 2-D ............ 81

Figure 3.18 - Joint Wrench with 2 Constrained DOFs in 2-D ............ 81

Figure 3.19 - Joint Wrench with 1 Constrained DOF in 2-D ............. 82

Figure 3.20 - Joint Wrench with 6 Constrained DOFs in 3-D ............. 82

Figure 3.21 - Joint Wrenches with 5 Constrained DOFs in 3-D ........... 83

Figure 3.22 - Joint Wrenches with 4 Constrained DOFs in 3-D .......... 84

Figure 3.23 - Joint Wrenches with 3 Constrained DOFs in 3-D ........... 84
CHAPTER 1. INTRODUCTION

Constraint analysis of assemblies has recently developed into an important branch of the design and manufacturing process. As mechanical assemblies have developed in complexity, the need for a thorough, defined constraint design method has evolved. Newer CAD tools require comprehensive techniques for analyzing variation. Different principles of constraint analysis are developing into a tool to aid in the research and design process. Some of these principles include kinematics, mobility analysis, screw theory, graph theory, and methods gained from manufacturing experience.

1.1 Kinematics Introduction

Kinematic analysis, of the relative joint motion of an assembly, plays an important part in determining its constraint status. For this thesis a few kinematic definitions will need to be established. A degree of freedom (DOF) is defined as the possible motions of a body with respect to a given coordinate system. For example, in two-dimensions (2-D) a body has three degrees of freedom (DOFs), two translations ($t_x$ and $t_y$) and one rotation ($\theta_z$). In three-dimensions (3-D) it has six DOFs, three translations ($t_x$, $t_y$, and $t_z$) and three rotations ($\theta_x$, $\theta_y$, and $\theta_z$). As rigid bodies are connected by joints, or kinematic pairs, DOFs are removed. Conventional kinematics texts therefore define the number of remaining DOFs as the needed inputs for a mechanism to uniquely determine the position of its links [Sandor and Erdman 1984].

At present there are a few constraint analyses utilized in applied kinematics. Grübler's equation, (1.1), is among the most used of these methods. For 2-D kinematic chains, consisting of links and joints, it predicts the mobility and number of needed inputs
to uniquely determine the mechanism’s position [Sandor and Erdman 1984], [Tsai 1999].

\[ F = 3(n - 1) - 2f_1 - 1f_2 \]  \hspace{1cm} (1.1)

In this equation, \( F \) is the number of calculated inputs for the mechanism, \( n \) is the number of links, \( f_1 \) is the number of lower pairs (pin joints), and \( f_2 \) is the number of higher pairs (cam joints or roll-sliding joints). This equation is based on general principles where lower pairs remove two DOFs and higher pairs remove one DOF.

There are a few limitations of this method that restrict its use in constraint design. When special geometry, such as parallelograms, are introduced as links in a mechanism, Grübler’s equation is inaccurate in determining its mobility. The example in Figure 1.1 shows a five-bar mechanism with special geometry. According to the Grübler analysis \( n = 5 \) and \( f_1 = 6 \), which yields \( F = 0 \), or no DOFs remaining. In actuality, this mechanism has one input to determine the final position of the rest of the links. This is because one of the links is redundant and removes no extra DOFs.

![Figure 1.1 - Example of limitations to Grübler's Equation, Five-bar Mechanism](image)

As this example shows, there are limitations to the methods of kinematic analysis. Another large drawback to these methods is that they cannot readily be taken from 2-D into 3-D. For realistic constraint analysis of complex, realistic assemblies, a 3-D system or method is a requirement. The drawbacks of the current, typical constraint methods demonstrate the need for a more encompassing constraint analysis method.
1.2 Constraint Analysis Introduction

A few crucial constraint analysis terms will be employed throughout this thesis and will be defined below. From a constraint point of view, assemblies can be represented by the constraint conditions of the DOFs. Three terms describe the mobility of assemblies: exact-constraint, overconstraint, and underconstraint.

Exact-constraint describes an assembly whose parts’ DOFs are all known and determined [Blanding 1999]. As parts and joints in assemblies mate, different DOFs are constrained to be mobile or immobile. In a 2-D assembly, exact-constraint exists when all three DOFs ($t_x$, $t_y$, and $\theta_z$) for each part are uniquely determined by mating contact between the parts of the assembly, with no redundancies. Similarly in 3-D, when all six DOFs ($t_x$, $t_y$, $t_z$, $\theta_x$, $\theta_y$, and $\theta_z$) of each part are uniquely determined by mating contact, the assembly is exactly constrained. An assembly is exactly constrained when there are no redundant constraints on immobile DOFs. It also means that the only mobile DOFs are the ones intentionally left mobile. Exact-constraint is usually the optimal design case and preferable in all situations. This is because underconstrained and overconstrained assemblies lead to various problems, as described below.

The overconstraint condition deals with the over-determination of parts’ DOFs within assemblies [Kriegel 1994], [Waldron 1966], [Whitney 1999]. When a part DOF is already constrained by mating joint and an additional joint determines the mobility or immobility of that same DOF extraneously, then the part is overconstrained. In these types of assemblies, the part DOFs are constrained to the same motion by conflicting constraints. This can result in interference, assembly problems and a need for tight tolerances. A good example of this is a standard door, see Figure 1.2, which is constrained to one DOF rotation around the hinge axis.
The three hinges constrain five DOFs leaving only one rotational axis. Each of the three hinges redundantly constrain the door to the same DOF, thus the door is overconstrained. To make the door system exactly constrained would require using only one hinge to locate and determine the hinge axis. In practice, all three hinge axes cannot be aligned exactly. Because of this fact, conflicts result which can interfere with the operation of the door or overstress the hinges, causing wear or failure.

An underconstraint emerges when parts of an assembly have DOFs that are not constrained [Adams 1998], [Whitney 1999]. In this condition, the parts’ motions are not uniquely determined. This can lead to assemblies that perform poorly or become unstable. These underconstrained parts contain a DOF that is able to move independently from the rest of the assembly. An example of a simple, underconstrained part is found in the assembly in Figure 1.3. The crank slider assembly has a revolute joint crank which controls the linear position of the slider. Between the crank and the slider is a link that is connected by two spherical joints. An idle DOF, however, exists between the two spherical joints. The connecting link can still rotate around its longitudinal axis. This assembly was not designed to control or uniquely determine the underconstrained, rotational DOF. If the connecting rod was constrained from rotating, the assembly would qualify as exactly constrained.
The terms “idle DOF” and “redundant constraint” were mentioned above, but deserve a more detailed description. An idle DOF is one that is not constrained by the mating conditions of the assembly. Its motion, rotation or translation, is unaccounted for by the assembly constraints and can move independent of the assembly, making parts of the assembly underconstrained. A redundant constraint occurs when a DOF is overly determined. When more than one mating joint acts to remove the same DOF, there is a redundant constraint; this system is now overconstrained. These two conditions are obstacles to exact-constraint design and need to be eliminated by design methods.

1.3 Tolerance Analysis Introduction

A unique goal of this research is to find a connection between constraint design methods and tolerance analysis techniques. Tolerance analysis predicts dimensional and geometric tolerance stack up in assemblies [Chase 1996]. Tolerances and variation affect all the aspects of the assembly process, from final product assembly, production cost, to process selection, tooling, and rework. Tolerances form the link between engineering design and manufacturing.

Each manufactured part contains dimensional variation which can be analyzed to
determine its impact on the entire assembly. The dimensional variation of each part can accumulate statistically and propagate kinematically, often causing unwanted variational effects in the assembly. To eliminate or control these effects, the assembly is analyzed to find which parts are the largest contributors to the tolerance stack up. Tolerances are assigned to each dimension to keep assembly variations within acceptable design limits.

One of the many methods used for variation analysis, the Direct Linearization Method (DLM), incorporates vector loops, which are chains of vectors connected from head to tail, tracing paths through the assembly [Gao 1993]. Each vector either depicts an unknown, adjustable assembly dimension (dependent variable) or a manufactured part dimension (independent variable). The vector chain forms a closed loop with the last vector tip ending on the starting point of the first vector. An assembly function can be derived from a mathematical representation of the vector loop. This assembly function relates the manufactured part dimension variations, or inputs, to the adjustable assembly dimensions, or outputs. The closed loops must uniquely locate the position of each part in the assembly. If the assembly is exactly constrained, the system of assembly equations may be solved uniquely. If the assembly is over- or underconstrained, no unique solution is possible. The DLM will be explained in depth in following sections along with its correlation to constraint analysis.

1.4 Thesis Objectives

The purpose of this thesis is to find a connection between constraint analysis methods, by which designers can assure exactly constrained 3-D assemblies, using tolerance analysis methods. The DLM vector loop paths assure that all the contributing dimensions and all kinematic variables are included in the resulting matrix equations. The joints that the loops pass through model the DOFs and constraints of the assembly. Currently there is no method to test for over- or underconstraints in these tolerance vector loops. This thesis will compare different methods with the DLM vector loop method and find out if the same constraint information can be extracted from the vector loop
equations as is found through other methods.

This thesis will first review current constraint analysis work. A summary of basic tolerance analysis principles will follow. An analysis of certain tolerance information will yield a connection to constraint analysis design method. This method will then be introduced and applied to a variety of case studies with the purpose of identifying under- and overconstraints. The objectives of this thesis are outlined below:

- Review the current constraint methods and kinematic principles with an introduction to screw theory.
- Produce an original and innovative interpretation of tolerance sensitivity information given by the Direct Linearization Method and the Global Coordinate Method.
- Develop the connection between constraint analysis methods and variation analysis information using screw theory.
- Apply the variation-based constraint analysis method to a variety of case studies (parts with different joint types).
- Compare method to current constraint methods and discuss results of research.

The relationship between variation and constraints contains a method for identifying the constraints of assemblies. This method could then be employed to analyze an assembly for variation problems and then simultaneously perform a DOF check for constraint problems. The objective of this thesis is to develop the relationship between tolerance analysis and constraint analysis, thereby allowing the designer access to variation and DOF information.

1.5 Chapter Overviews

The chapters of this thesis will be presented as follows. Chapter 2 gives the background information and literature review on constraint methods, screw theory,
kinematics, and tolerance analysis. Chapter 3 is the step-by-step, theoretical development of the variation-based constraint analysis method employing screw theory. Chapter 4 shows the application of this method in identifying under- and overconstraints to several case studies, both 2-D and 3-D. Chapter 5 contains a comparison of the new method to current methods, which discusses the advantages and disadvantages of each method. Chapter 6 includes conclusions and recommendations for future work.
CHAPTER 2. CONSTRAINT ANALYSIS
BACKGROUND AND CURRENT METHODS

A literature review of current constraint analysis methods, as well as needed constraint analysis background are presented in this chapter. Also included is a review of the key concepts in kinematics, screw theory, and tolerance analysis. These concepts will be applicable to the development of a variation-based constraint analysis method.

2.1 Kinematics Review

A few important kinematic principles will be reviewed in this section, to be used in subsequent chapters. Kinematic analysis, in two- and three-dimensions (2-D and 3-D, respectively), requires definitions for degrees of freedom, 3-D joint types, homogeneous transformation matrices, as well as their relationships to each other.

2.1.1 Degrees of Freedom

Degrees of freedom (DOFs) in this thesis are defined to be the possible motions of a rigid body with respect to a given coordinate system [Blanding 1999]. A body in 3-D space has a set of six DOFs that define its position and orientation with respect to a coordinate system. These motions are represented by three translations ($t_x$, $t_y$, and $t_z$) and three rotations ($\theta_x$, $\theta_y$, and $\theta_z$) with respect to the X-axis, Y-axis, and Z-axis. Figure 2.1 shows a body located with respect to a 3-D, Cartesian coordinate frame. The rigid body’s position and orientation can be described, with respect to the origin O, using the six DOFs. Each translation and rotation can be represented with respect to individual axes. DOF analysis yields information regarding the kinematic motion of each part of an assembly. As different parts are assembled, their DOFs can be constrained to limit their
rotational and translational motion. Determining the remaining DOFs is fundamental in analyzing an assembly for constraint information. The 3-D case above can also be limited to 2-D, with three DOFs ($t_x$, $t_y$, and $\theta_z$).

A set of parts can be joined in a variety of ways to constrain different DOFs. Kinematic joint types, and the accompanying constraints on DOFs, depend on the mating features of the individual parts [Waldron and Kinzel 1999]. Each joint type is distinguished by the mating surfaces and how the remaining DOFs, either translating or rotational, can move relative to each other. In 3-D analysis there are twelve common kinematic joint types which represent most mating conditions. These joint types, taken from [Gao, Chase, and Magleby 1998], are summarized in Table 2.1. Each kinematic joint type constrains the relative motion and orientation between the mating surfaces in specific directions. Table 2.1 lists the remaining number of DOFs, or unconstrained directions, in parenthesis and the arrows show them as rotational or translational motions about the joint axes.

Assemblies of parts may incorporate many of the joint types. The different joint types allow analysis for combinations of part DOFs to determine the overall DOFs of the assembly. Kinematic analysis can also search for the existence of idle DOFs or redundant constraints. The joint definitions form a foundation for both kinematic constraint analysis and tolerance analysis.
2.1.2 Homogeneous Transformations

Another important kinematic principle in constraint analysis is the homogeneous transformation matrix. Used in computer-aided design (CAD) and robotics analysis, this transformation has made 3-D kinematics more practical. The following is a summary of the transformation principles outlined in [Rogers and Adams 1990]. For kinematic analysis, a 4 x 4 transformation matrix is used to switch from a specific, 3-D pose (orientation and position) to a different one. This 4 x 4 matrix is generically represented by the matrix \( H \) below:

\[
H = \begin{bmatrix}
    a_x & b_x & c_x & p_x \\
    a_y & b_y & c_y & p_y \\
    a_z & b_z & c_z & p_z \\
    0 & 0 & 0 & 1
\end{bmatrix}
\]

(2.1)

Table 2.1 - 3-D Kinematic Joint Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rigid (no motion)</td>
<td>![Rigid Diagram]</td>
</tr>
<tr>
<td>Prismatic (1)</td>
<td>![Prismatic Diagram]</td>
</tr>
<tr>
<td>Revolute (1)</td>
<td>![Revolute Diagram]</td>
</tr>
<tr>
<td>Parallel Cylinders (2)</td>
<td>![Parallel Cylinders Diagram]</td>
</tr>
<tr>
<td>Cylindrical (2)</td>
<td>![Cylindrical Diagram]</td>
</tr>
<tr>
<td>Spherical (3)</td>
<td>![Spherical Diagram]</td>
</tr>
<tr>
<td>Planar (3)</td>
<td>![Planar Diagram]</td>
</tr>
<tr>
<td>Edge Slider (4)</td>
<td>![Edge Slider Diagram]</td>
</tr>
<tr>
<td>Cylindrical Slider (4)</td>
<td>![Cylindrical Slider Diagram]</td>
</tr>
<tr>
<td>Point Slider (5)</td>
<td>![Point Slider Diagram]</td>
</tr>
<tr>
<td>Spherical Slider (5)</td>
<td>![Spherical Slider Diagram]</td>
</tr>
<tr>
<td>Crossed Cylinders (5)</td>
<td>![Crossed Cylinders Diagram]</td>
</tr>
</tbody>
</table>
The $\mathbf{H}$ matrix contains the transformations necessary to translate any given 3-D coordinate frame to a new location and rotate it to a new orientation. The $\mathbf{H}$ matrix can be separated into submatrices which govern either the rotation or translation. These submatrices are shown below:

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (2.2)$$

The rotational submatrix $\mathbf{R}$ is a $3 \times 3$ matrix containing the following elements:

$$\mathbf{R} = \begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{bmatrix} \quad (2.3)$$

Each column of $\mathbf{R}$ represents the direction cosines which define the axes of rotation from one coordinate frame pose to another. The $3 \times 1$ vector $\mathbf{p}$ contains the translation transformation.

$$\mathbf{p}^T = \begin{bmatrix} P_x & P_y & P_z \end{bmatrix} \quad (2.4)$$

This vector translates a coordinate frame along the X-axis a distance $P_x$, along the Y-axis a distance $P_y$, and also along the Z-axis a distance of $P_z$. The combination of $\mathbf{R}$ and $\mathbf{p}$ in $\mathbf{H}$ gives the $4 \times 4$ homogeneous transformation that is another foundation of 3-D kinematics.

Each time a coordinate frame is taken from one pose to another, a new $\mathbf{H}$ is needed. As the pose, or kinematic position and orientation, is changed a multitude of times many transformations are needed. If only the ending pose is required, then the
necessary, intermediate $H$ transformation matrices may be concatenated to produce a transformation matrix that transforms the initial pose to the final one. The properties of transformation matrices make them very useful in kinematic analysis [Tsai 1999], [Waldron and Kinzel 1999] and tolerance analysis [Chase 1996].

2.2 Review of Current Constraint Methods

This section outlines current constraint methods utilized in both research and production fields. The different methods described below apply different aspects and principles of engineering. As this is only a review of the techniques, a study of the primary sources information is suggested for those who wish further understanding.

2.2.1 Kinematic Constraint Pattern Analysis

The first method of constraint design involves design and analysis of parts of assembly according to kinematic knowledge. This method uses understanding of kinematic joints and mating parts to devise a set of rules that a designer can follow to avoid over- and underconstraints.

In [Kriegal 1994], Jon Kriegal shows, through a series of test cases, some of the design problems involved in ignoring constraints. Overconstraints were found in many production examples. The solutions to these constraint problems incorporated the use of a knowledge of DOFs and how a part reacts when their DOFs are constrained. The main purpose of this work was a plea to educational systems to teach constraint study in university, senior-level undergraduate classes. This work cites the work of John (Jack) E. Morse, an electrical engineer whose later work dealt with this topic almost exclusively. Although Mr. Morse did not summarize this work into a written form, one of his supporters did.

In [Blanding 1999], Douglass Blanding sets forth a methodical design outline for eliminating constraint problems. A precursor to this work was cited in [Kriegal 1994]
and credit is also given, by Blanding, to John Morse. This method is also based upon a knowledge and comprehension of part reaction to a constraint of the DOFs. This technique is called Constraint Pattern Analysis.

The first step of this design method is defining degrees of freedom. DOFs are set forth in [Blanding 1999] the same way they are in this thesis. Following DOFs is the definition of constraints (C’s) and rotational freedoms (R’s). C’s are represented by lines in space with R’s representing the remaining rotational motions (translational freedoms, T’s, are represented as rotations that occur at infinity).

Two basic rules of this analysis are that along the lines of C no motion is allowed and that the intersection of two of these lines produces a R perpendicular to the intersection; see Figure 2.2. Also, this intersection of C’s form a plane upon which a disk of equivalent lines can be formed. This disk implies that any two lines in this disk can replace the original two C’s.

![Figure 2.2 - 2D body constrained by two C’s resulting in one R. Modified from [Blanding 1999, p. 11].](image)

The above figure shows the constraint of a 2-D body by two C’s forming one R. The two C’s could be replaced with any two C’s in the same 2-D plane and result in the same R.
After these necessary definitions are in place, The Rule of Complementary Patterns is defined. This rule states that a symmetry exists between the patterns of C’s and the pattern of R’s. It asserts that “Given a pattern of \( n \) lines without redundancy, the complementary pattern will contain \( 6-n \) lines and every line of one pattern will intersect every line of the complementary pattern.” This relationship between the C’s and R’s (their values sum to six) is then used to determine DOFs and the relative motion between different parts of an assembly. If one pattern is known (of C’s or R’s), then the other can be determined by symmetry. This knowledge of C’s and R’s allows analysis of mechanical connections.

Constraint Pattern Analysis is based on transformation of the different patterns of one type (C’s or R’s) into an equivalent pattern of the other type. An example of a design test case using this method is given in [Blanding 1999]. The design requirement states that a body must be mounted so that it can rotate around two different axes (called a gimbal and caster in this example). The solution involves showing the disk of radial lines defined by the intersection of the two rotational axes, as shown in Figure 2.3. The disk represents the locus of equivalent rotation axes. Any two (non-collinear) will meet the requirement.

![Figure 2.3 - Body showing desired constraints and disk of radial lines. Modified from Blanding 1999, p. 43.](image-url)
The disk of lines is in the same vertical plane as the caster and gimbal axes with the center at their intersection. The final design can be chosen from these lines of intersection. Using lines of intersections along with the knowledge of C’s and corresponding R’s allow the designer to follow the Constraint Pattern Analysis design outline and avoid over- and underconstraints. The mechanisms in Figure 2.4 show two final design possibilities, where the two R axes are lines located in the radial disk.

![Image](image.png)

**Figure 2.4 - Two final design concepts created using Constraint Pattern Analysis. Modified from [Blanding 1999, p. 102].

This method of Constraint Pattern Analysis also develops basic guidelines for flexures, couplings of four to zero C’s, structures, and web handling. This method’s basic constraint analysis is summarized by a set of design rules. These rules are listed in outline form in [Blanding 1999] and will not be repeated here. The outline sets forth a guide for designers implementing different paths to follow. One path to design with respect to C’s and one for R’s, each identifying and eliminating over- and underconstraints.

This method is a qualitative one, which states a clear methodology without stating much of its mathematical basis. It presents a logical set of steps for a designer to follow, but may be difficult to incorporate into a computer-driven constraint solver.
2.2.2 Geometric Constraint Solving

A different approach to constraint solving involves solving for the orientation and part position when given a set of geometric elements and their corresponding geometric constraints. This method, called Geometric Constraint Solving, is mathematically driven, employs graph based, constraint theory, and can be computer implemented. In [Hoffmann and Vermeer 1995], a detailed description of the theory and implementation of this method is given. Much of this section on Geometric Constraint Solving is a summary of [Hoffmann and Vermeer 1995] along with parts from [Hoffmann 1995] and [Hoffmann 1997].

The goal that this method addresses is solving for all the placements of the given geometric entities (i.e. lines, arcs, circles, angles, etc.) while satisfying the given geometric constraints between the entities. An example given in [Hoffmann and Vermeer 1995] is a set of elements that consist of three lines, shown in Figure 2.5. The constraints are that the first two lines must be perpendicular and the third line (dashed in Figure 2.5) must have a certain intersection angle, $\theta$, with the first line. This underconstrained problem has infinitely many solutions as the third line can intersect the second line at any position.

![Figure 2.5 - Evaluation of four situations of given geometric elements (lines) and the given geometric constraints (angles and g)](image)

17
The well-constrained solution to this problem involves adding another constraint to the given set, such as adding the length, \( g \), of one segment between the intersections of two lines. If a second angle, \( \phi \), were called out (as seen in Figure 2.5) then the problem would be overconstrained because that certain angle was already constrained by the right angle and the first angle \( \theta \). It is also possible to have both under- and overconstrained conditions in these problems.

In this method, a solution can show underconstraint, overconstraint, or well-constrained solutions. The definitions of these terms differ slightly from the ones given initially in this thesis. For this method, a well-constrained problem is one where there are a finite number of solutions. An underconstrained problem has an infinite number of solutions and an overconstrained problem has a finite number of solutions even when one geometric constraint is removed.

As stated above, the first constraint example in Figure 2.5 is initially underconstrained, with an infinite number of solutions. If the length of the segment of intersection between a pair of lines, \( g \), is given as a constraint, the problem reduces to a well-constrained problem. If the angle, \( \phi \), between the third and second line is given as a constraint the problem becomes overconstrained, as that angle is already determined by the first angle constraint. The goal of this method is to assure well-constrained problems to give more precise design models.

The basic procedure for solving constraints with the Geometric Constraint Solving method in [Hoffmann 1997] and [Hoffmann and Vermeer 1995] uses a generic solver to determine if the given geometric elements can be uniquely positioned using the given constraints, with numerical values assigned to a minimum number of the elements. The position of an element can be represented by a nonlinear, algebraic equation and the constraints are expressed as different parameters in the equations.
There are four different solver approaches that are being used by researchers to solve variational geometry problems: numerical algebraic, symbolic algebraic, logical interface and term rewriting, and graph-based constraint. The method employed by Hoffmann et al. in [Hoffmann and Vermeer 1995] is the graph-based constraint solver.

The algorithms used in this technique have a construction phase and an analysis phase. A graph-representation is constructed by the algorithm to represent the problem. Nodes in the graph represent a single element and an edge between two nodes represents the constraint that exists between the elements. An example of a graph-based representation is shown in Figure 2.6.

![Graph-based representation of elements](image)

Figure 2.6 - Graph-based representation of elements $a$, $A$, and $B$ with constraint length $d$. Modified from [Hoffmann and Vermeer 1995, p. 272].

In the figure above, line segment $a$ is defined by points $A$ and $B$ with distance of $d$. Its corresponding graph representation is also shown in Figure 2.6. This graph can be analyzed to find over-, under-, or well-constrained solutions. Rules found in [Hoffmann and Vermeer 1995] describe the process of making a well-constrained graph. When the graph has been made well-constrained, the algorithm determines the steps to solve the problem to indicate in what order and how to place the elements. In this way, the generic problem can be condensed to a specific, well-constrained assembly with full dimensions.

The geometric elements of lines, circles, etc. can be represented by coordinates and equations. Conversely, the constraints can be solely depicted by distances and angles between these elements. The geometric constraint problem then reduces to one of only
placing points and lines. Clusters, or collections, of geometric elements which need to be placed relative to each other can be gathered together. These clusters can then be merged into larger collections from implementation of rigid-body transformations.

A methodology for finding clusters is outlined in [Hoffmann and Vermeer 1995] which takes a body and creates its constraint graph. Using this it is possible to separate different clusters within the constraint graph. This method is based on adding nodes and constraint edges to a cluster until it meets the requirements of being a “filled cluster” (See [Hoffmann and Vermeer 1995] for definition). Figure 2.7 shows the formation of three clusters from a well-constrained sketch and its constraint graph.

![Figure 2.7 - A well-constrained assembly graph and its constraint graph with three clusters. Modified from [Hoffmann and Vermeer 1995, pp. 274-275].](image)

The three clusters share certain nodes which allow the graph analyzer to proceed. Clusters can be merged to form a single structure. The distances and other geometric elements are analyzed to merge the clusters, or in other words, using the rigid body transformations to bring them together in the correct relationships. The method then follows basic construction steps to solve for the different possible transformation arrangements. The solution of the analysis and construction steps, which is not summarized here, is expressed in the correct placement of the elements in Figure 2.7.
This method can be extended for 3-D assemblies. In three space, the same steps are followed: geometric entities considered, cluster formation, basic construction steps, and cluster merging. The unchanged, ultimate goal being to solve for all the element placements while satisfying the geometric constraints. The mathematical aspects of this method have not been reviewed, but can be seen in the primary sources. As of the writing of this thesis, Geometric Constraint Solving had been implemented by Hoffmann for use in computer-aided design. This method portrays how network graphs and graph theory can be employed to solve constraint problems.

2.2.3 Screw Theory-Based Constraint Analysis

Before this constraint analysis method can be properly introduced, the basic rules of screw theory will be briefly reviewed. The material in this section is taken from [Tsai 1999], [Ball 1900], [Roth 1984], [Phillips 1984], and [Adams 1998], which describe it in much greater detail.

Screw transformations are based on Chasle’s theorem, which states that a motion of a rigid body can be represented by a rotation of the body around a screw axis and a translation following the axis [Adams 1998]. The screw is basically another transformation that takes a 3-D body from an initial pose (a specific position and orientation, with respect to a given coordinate frame) to a final one. The screw transformation uses a combination of one rotation and an accompanying displacement to accomplish the change in pose. This single transformation can replace the many needed homogeneous transformation matrices it might take to produce the identical motion.

The single rotation takes place around an arbitrary, unique screw axis that passes through the origin of the reference frame of the first pose. This rotation is a special form of \( R \) (2.3) from the homogeneous transformation, where the entries for its column vectors \( \mathbf{a}, \mathbf{b}, \) and \( \mathbf{c} \) contain the rotation information. The rotation axis passes through the origin of the reference coordinate frame and has a direction defined by \( \mathbf{k} \), a unit vector that has the X, Y, and Z direction cosines needed to give the correct pose to the screw axis. Once
the screw axis is defined by \( k \), a rotation of angle \( \theta \) aligns the orientation of the initial pose coordinate frame with the final pose coordinate frame.

A screw axis and accompanying rotation are pictured in Figure 2.8. This figure shows the rotation of an initial coordinate axis \((X,Y,Z)\) to a final pose at a coordinate system of \((X',Y',Z')\). The rotation is of magnitude \( \theta \) around the screw axis, with unit vector \( k \). Although this figure shows the screw axis to be located at the center of the coordinate system, it can be an arbitrary axis located somewhere else in space. The transformation rotates the initial coordinate axis to align with the final axis. This one rotation along the screw axis accomplishes the rotation-orientation transformation.

![Figure 2.8 - Rotation of XYZ around Screw Axis to X'Y'Z' by angle \( \theta \)]

The position transformation is simply a translation along the screw axis. The body that was rotated around the screw axis only needs a translation to complete the change in pose. With the addition of this translation to the rotation, the initial pose has completed the transformation to the final one. This is shown in Figure 2.9 below.
The above figure shows a change of pose, from position 1 to position 2, around a screw axis. The rigid body located from O is rotated around the screw axis to an intermediate position (the dot-filled body) and then it is translated parallel to the screw axis, the distance \( d \), giving the final pose.

In summary, screw rotation theory says that a motion of a rigid body can be represented by a rotation of the body around a screw axis and a translation following the axis. Screw theory can also be used in kinematic methods to describe motion of rigid bodies. The constraint method described below uses screw theory as a way for collecting and analyzing 3-D kinematic motion and constraint information.

Beginning as early as 1900, screw theory has been used to represent the relative motions, as well as the forces and moments acting on a body. This screw representation has been implemented by many individuals into a constraint method capable of identifying over-, under-, and exact-constraint. This is accomplished by analyzing the relative motions and forces of the connecting, 3-D joint types within a rigid body. In [Ball 1900] statics, kinematics, and dynamics of different rigid bodies were organized.
using the screw as the base element of the geometry. The following works by [Roth 1984], [Adams 1998], and [Waldron 1966] summarize key parts of this geometry representation as well as present definitions used in screw theory constraint analysis.

The screws used in describing rigid-body motion are slightly different than the description given earlier in this section. The screw transformation (Chasle’s theorem) described above is used to transform a rigid body from one orientation and position to another one using a rotation around a screw axis and a displacement along that axis. Application of screw theory to constraint analysis requires the introduction of two types of kinematic screw representations, the twist and the wrench.

According to [Roth 1984, Adams 1998], and [Adams and Whitney 2001] if the displacement of a rigid body around a screw axis is taken to be infinitesimal, then the directions of its angular velocity and linear velocity can also be represented using screw notation. A twist is a screw representation of the first order in a two triplet, six element row vector. The first triplet represents the angular velocity sensitivities and the second represents the linear velocity sensitivities, both with respect to a unique, global coordinate frame.

\[
T = \begin{bmatrix}
\omega_x & \omega_y & \omega_z & v_x & v_y & v_z
\end{bmatrix}
\]  \hspace{1cm} (2.5)

The twist motion is represented by the rigid body in Figure 2.10, along with its corresponding angular and linear velocities (\(\omega\) and \(v\), respectively).
Similarly, forces and moments acting on a rigid body can be represented using screw notation. The works by [Waldron 1966], [Roth 1984], [Adams 1998], and [Adams and Whitney 2001] describe the formation of a wrench by combining forces and moments conforming to Poinsot’s theorem. This theorem states that any system of forces and moments acting on a rigid body can be uniquely replaced by one force and one moment. This force will be located along a coaxial, unique screw axis, around which the moment acts. A wrench describes this force and moment representation.

Like the twist, the wrench is also a six element row vector with two triplets. The first wrench triplet depicts the DOF directions of the force components and the second triplet showing the directions of the moment components, also with respect to a unique, global coordinate frame.

\[
\mathbf{W} = \begin{bmatrix} f_x & f_y & f_z & m_x & m_y & m_z \end{bmatrix} \tag{2.6}
\]

In essence, the wrench combines all the forces acting upon a rigid body into a resultant force and also combines all the moments into a resultant moment. This force and moment combination (\( \mathbf{f} \) and \( \mathbf{m} \), respectively) is directed along a unique screw axis, as shown in Figure 2.11.
With these definitions in place, it is possible to trace the development of constraint analysis using screw theory from the beginning to its current use. The earliest work by [Ball 1900] sets forth kinematics and dynamics using the screw as the basic unit of geometry. It is the foundational work and key primary source to later screw-based constraint theory. Although it does address the use of screws in kinematic analysis, it does not fully address the DOFs, constraint problem.

The challenge of applying screw theory to constraint analysis is addressed early on by [Waldron 1966]. His method of constraint solving involves defining the mobility of each of the joints of a mechanism. This is accomplished by setting up an instantaneous screw axis representation of motion at each joint. Relative motion between two joints is defined by a screw system, which is a spatial distribution of the axes based upon their screw axis order. The order of a screw axis is the connectivity of the joint, which can be analyzed for mechanism mobility.

The screw system of a mechanism can be inspected for a comparison of order. This method of comparison is further compared to Grübler’s equation and other basic mobility criteria equations, showing the advantages and drawbacks of using screw axes.
It is shown that in some cases, where standard equations cannot yield the correct mobility, this method proves to be better. The entire description within the paper is much more illuminating and involves greater examination.

The important screw theory principle found in this source involves the reciprocity of wrenches and twists. The constraints of a joint can be identified using the reciprocal of the screw systems described above. [Waldron 1966] states that the motion screw systems are in fact a reciprocal system of a corresponding screw system involving forces and moments. It is stated that if a rigid body is free to move about one screw, it will be in equilibrium with the action (force and moment) of another screw. The screw systems describing the order of the joints are twists and the reciprocal screw needed for equilibrium is the wrench. This important principle is used in successive sources to advance the screw-based constraint method.

Other later works, [Phillips 1984] and [Roth 1984], build on the basics of screw theory for kinematics, but do not apply it directly to constraint or DOFs analysis. Although [Phillips 1984] states the requirements for overconstraints, it basically discusses a qualitative approach to solving them. This work does employ screw theory as a means to establishing mobility, an improvement over Grübler’s equation. It develops the principles originally outlined in [Waldron 1966], and applies them to a variety of test cases and other robotics applications. A definitive constraint method is not presented, but the background on screw theory lends a great deal to the current constraint research.

As stated before, [Roth 1984] presents a summary of the ideas put forth in [Ball 1900]. A few kinematic ideas are presented, but the significant contribution of this work is the further development of the screw systems. According to [Roth 1984], screw systems can be defined using a sextuple of homogeneous coordinates that are called screw coordinates. These screw coordinates are the basis for the six element, two triplet representation of the twist and wrench. The development of screw coordinates in this paper also leads to greater mathematical analysis of screws and their intersections.
The advancements in screw theory application outlined above led to the graduate thesis [Konkar 1993] and the subsequent summary technical article [Konkar and Cutkosky 1995]. These two works lay the mathematical foundation for a definitive treatment of constraints using screws. First, they summarize the key principles of the previous publications and develop an algorithm that can be used in the solution of DOFs as well as constraint information. The following screw theory definitions and mathematical procedure are adapted from [Ball 1900], [Roth 1984], and [Phillips 1984], but [Konkar and Cutkosky 1995] presents the unique algorithms that are derived from these previous works.

In the following, a screw is still a six element vector that can either represent motion by a twist, or forces by a wrench. Screws can be combined in a union to form a screw set matrix. This union of screws is the combination of the screw vectors as rows of a matrix. So if $s_1, s_2, \ldots, s_n$, are the individual screws, the union would be the following.

$$S_{\text{union}} = \text{Union}(s_1, s_2, \ldots, s_n) = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix}$$ (2.7)

Also, according to [Konkar and Cutkosky 1995], the intersections of a screwset is the set of screws common to each of the screw sets. It can be computed using a double reciprocal of the different screw sets $S_1, S_2, \ldots, S_n$.

$$S_{\text{intersection}} = \text{Reciprocal} \left( \bigcup_{i=1}^{n} \text{Reciprocal}(S_i) \right)$$ (2.8)

or
\[ S_{\text{intersection}} = \text{Reciprocal}\begin{bmatrix} \text{Reciprocal}(S_1) \\ \text{Reciprocal}(S_2) \\ \vdots \\ \text{Reciprocal}(S_n) \end{bmatrix} \] (2.9)

The mathematical meaning of the reciprocal screw is not reviewed here, but it can be found in detail in [Konkar and Cutkosky 1995]. Accompanying these mathematical definitions is the concept of a virtual coefficient. This coefficient equals the time rate of work done by a wrench in moving along a corresponding twist. This virtual coefficient relationship combined with the twist-wrench reciprocal relationship found in [Waldron 1966] leads to a very important result.

If the virtual coefficient of a twist and wrench system is zero then they are mutually reciprocal. This is physically saying that the wrench does no work in moving along the twist. This discovery allows [Konkar and Cutkosky 1995] to compute the possible twists a rigid body can perform while in contact with other bodies. It is also possible to compute the wrenches exerted from other bodies on a rigid body whose motion, or twists, is known. This reciprocity relationship is expressed mathematically by the following equation:

\[ [T][W] = \{0\} \] (2.10)

If elements in twistspace are known, then the compliment elements in the wrenchspace can be solved for using the linear algebra null-space operation. Likewise, elements in the wrenchspace can be used to solve for the compliment elements in the twistspace.
It is this mathematical discovery which leads to the development of the analysis algorithm. The problem stated in [Konkar 1993] is to find the motion capacity (the full space of twists) of $A$ with respect to $G$, where $A, B, \ldots, G$ are links in a mechanism. The solution involves solving for what are termed above as the twistspace and the wrenchspace. The twistspace of $A$ is the space of twists reciprocal to the wrenches of contact acting upon $A$ from the other bodies $B, \ldots, G$. The union of all the individual wrenches acting upon $A$, again from the other bodies, is the wrenchspace. The developed algorithm can recursively solve for the wrenchspace and twistspace of each component. In essence, the relative motions and forces can be determined using the geometry and the different screw axes that represent the contact points of bodies $A$ through $G$. The complete algorithm for the solution of the twistspace and the wrenchspace is contained in [Konkar 1993] and summarized in [Konkar and Cutkosky 1995].

These algorithms made it possible to solve for the twistspaces and wrenchspaces of a mechanism, and therefore garner motion and force information. This information could then be applied to DOF constraints. The next step merged all of the previous work into a useful, efficient constraint method.

The works [Adams 1998] and [Adams and Whitney 2001] adapted the previous research into such a constraint method. The kinematic application of screw theory may again be explained using the twist and wrench. Any independent motion of a part can be represented by a twist. If there are two independent motions allowed by a kinematic joint, then two twists are all that are needed to form the twistmatrix and fully describe the set. Also, any forces and moments that can be transmitted by a joint can be described, in full, by the concatenation of wrenches into the wrenchspace.

According to [Adams 1998] the intersection of several twistmatrices (describing the relative motion of the connecting joints of parts within an assembly) can be calculated and stored in a resultant twistmatrix having the form of matrix $T$ in (2.11).
If a twist appears in the resultant twistmatrix (or intersection of all the joints' twistmatrices), it indicates that the motion described by the appearing twists is possible within the assembly. The part comprising the twists will allow the motions that appear in the resultant twistmatrix. Each row of the resultant twistmatrix represents one relative motion that is allowed by different parts within an assembly. The algorithm derived by [Konkar 1993] allows one to solve for the resultant twistmatrix and therefore solve the relative motions of an assembly.

The next step by [Adams 1998] involves finding resultant wrenchspace of parts in the assembly. With the knowledge of the relative motions of the parts in an assembly, it is possible to again employ the reciprocal nature of twists and wrenches. Consider the twists used to find the resultant twistmatrix $T$. If the reciprocal of the twists were calculated, it would yield a set of wrenchspaces describing all of the forces and moments that could exist within the assembly. The wrenchspaces represent all of the DOFs that were constrained by the joints of the parts. It is then possible to find the intersection, as was performed above, of all of the wrenchmatrices and form a resultant wrenchmatrix having the form of $W$. This is done for all sets of mating joint features for the different parts. When all the intersection of all the subsets are checked, the parts can be identified as overconstrained [Adams and Whitney 2001].
Similarly to the resultant twistmatrix, if a wrench appears in the resultant wrenchmatrix, it signifies a constrained DOF that is shared by all the joints in the assembly. Each row in the resultant wrenchmatrix represents a DOF that each joint is trying to constrain, or in other words each row represents an overconstraint. Again, the algorithm found in [Konkar 1993] allows for the solution of the resultant wrenchmatrix.

By representing the motions allowed by a 3-D joint type as twists, it is possible to analyze the assembly for over-, under-, and exact-constraints of different parts in an assembly. Solving for the resultant twistspace yields the relative motions allowed by parts in the assembly. If the assembly is underconstrained, the matrix will show the idle DOF or the allowed, uncontrolled motion. The resultant wrenchspace shows overconstraint information. If parts in the assembly are overconstrained, the wrenches in the matrix will show which DOFs are being redundantly constrained by features of the different parts.

The task of applying this constraint method to a working example is presented in detail by [Adams 1998] and again summarized by [Whitney, Mantripragada, and Adams 1999] and [Adams and Whitney 2001]. A set of 17 joint types represent the set of possible combinations of DOF constraint of rigid body objects. This set is outlined in full in the primary source and will not be reproduced here. Each joint type, or pair of mating features, has an accompanying twist representation. These representations are used to find the resulting twistmatrix and wrenchmatrix.

The example used in [Adams 1998] employs the use of the revolute joint and pin-and-slider joints. The part is pictured in Figure 2.12 below.
The part shows a revolute joint feature, called f1, and a pin-slot feature, labeled f2. This assembly can be analyzed for constraint information using the twist representation. The features (f1 and f2) have local coordinate frames, denoted in Figure 2.12 by xyz, located with respect to a global coordinate frame, denoted with XYZ. The local frames are found using the homogeneous transformation matrices discussed in 2.1.2. The homogeneous transforms and their submatrices are evaluated to find the twist matrix representation of each feature or joint.

The motion analysis involves finding the resultant twist matrix of this assembly. The first step is locating the features with respect to the global coordinate system using their homogeneous transforms, \( \mathbf{F}_1 \) and \( \mathbf{F}_2 \).

\[
\mathbf{F}_1 = \begin{bmatrix}
1 & 0 & 0 & 2 \\
0 & 1 & 0 & 2 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]  

(2.13)
\[ F_2 = \begin{bmatrix} 0 & 1 & 0 & 2 \\ -1 & 0 & 0 & 6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]  \hspace{1cm} (2.14)

These transformations are used in the twist definitions for each feature in [Adams 1998]. The definitions are applied for the hole and slot features to yield twist matrices.

\[ T_1 = \begin{bmatrix} 0 & 0 & 1 & 2 & -2 & 0 \end{bmatrix} \]  \hspace{1cm} (2.15)

\[ T_2 = \begin{bmatrix} 0 & 0 & 1 & 6 & -2 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} \]  \hspace{1cm} (2.16)

The next step in the motion analysis involves the algorithm for solving the union of twists. The resulting, intermediate matrices are concatenated into a single matrix. This matrix is transformed to reduced row-echelon form and the null-space is found. The resulting matrix is the resultant twist matrix. For the example in Figure 2.12 the resultant twist matrix is shown in (2.17).

\[ T_{\text{Resultant}} = \begin{bmatrix} 0 & 0 & 1 & 2 & -2 & 0 \end{bmatrix} \]  \hspace{1cm} (2.17)

The resultant twist matrix can be interpreted by analyzing the triplets within the vector. The first triplet, [0 0 1], shows the vector of the possible angular velocity, or the unit vector of a rotation allowed by the two parts, with respect to the global coordinate frame. This example yields a rotation around the global Z-axis. The second triplet in the twist, [2 -2 0], gives location information of the rotation. Using a point algorithm, also developed by [Konkar 1993], a type of "inverse cross-product" can be found. The point of rotation in this example, \( p = (2, 2, 0) \), is the center of the f1 feature.
The motion analysis shows that the top part of the assembly is underconstrained for rotation around the Z-axis. This plate can rotate around the pin joint and is not fully constrained by the mating joints. This is a remarkable solution in that this screw-based method is now able to identify underconstraints.

To solve for overconstraints, the resulting wrench matrix must be calculated. This is accomplished by creating a union of $\mathbf{T}_1$ and $\mathbf{T}_2$ and finding reduced row-echelon form of the result. The intersection algorithm is applied to determine resulting wrench matrix. The resultant wrench matrix shows the overconstraint information by rows. Each wrench matrix row shows overconstraint problems with displacements when appearing in the first triplet and problems with rotation when in the second triplet.

$$W_{\text{resultant}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.18)$$

Thus, the first row of (2.18) shows an overconstraint in direction of the Y-axis and a rotation about the Z-axis. This means that a change in the dimensions along the Y-axis could cause problems during assembly. The second row of (2.18) shows an overconstraint in the direction of the Z-axis, while the third and fourth rows denote overconstraints in the rotation of the X-axis and Y-axis, respectively. The last three rows show the constraints that occur because of the plates mating on a plane, with each plate not moving away from the planar mate.

The work presented in [Adams 1998] is thus far the culmination of work in this area. The screw theory-based constraint analysis method has developed from the first screw theory application to a systematic, relatively easy-to-use, mathematical process for identifying under- and overconstraints in assemblies. The mathematics has been proven and the method is applicable to a great variety of parts within assemblies.
2.3 Tolerance Analysis Background

The goal of this thesis is to develop a connection between constraint analysis and current tolerance analysis modeling methods. Of the different tolerance analysis methods, the Direct Linearization Method (DLM) presented in [Gao, Chase, and Magleby 1998] and [Chase, Gao, Magleby, and Sorensen 1996], as well as the Global Coordinate Method (GCM) described in [Gao 1993], will be reviewed here. The DLM and GCM form the basis for the remainder of the research in this thesis. The background needed in tolerance analysis modeling and vector loops will be presented in this section.

The assigning of tolerances to assemblies cannot be arbitrary or the assembly will suffer numerous effects. Tolerances and variation affect every aspect of the assembly process, from final product assembly, production cost, all the way to process selection, tooling, inspection and gaging, and scrap and rework. Tolerances form the link between engineering design and manufacturing. Statistical tolerance analysis can identify the primary sources of problems and help to eliminate them.

Variation control is a key principle of tolerance analysis. There are three major sources of variation according to [Gao, Chase, and Magleby 1998]. They include dimensional variation, geometric feature variation, and variation due to the small kinematic adjustments which come about due to assembly. A simple example involves the two-part assembly of inserting a cylinder into a groove until it makes contact on the two mating surfaces. The assembly has three component dimensions that vary, one on the cylinder and two on the tapered groove. The distance U is the assembly dimension of interest and it is affected by changes in the three component dimensions (A, R, and θ), as shown in Figure 2.13.
For different sets of parts, the distance $U$ will change to conform to the values of $A$, $R$, and $\theta$. In Figure 2.13 the gap $U_1$ represents the cylinder’s nominal position and $U_2$ represents the new position when variation is introduced. This adjustability of the cylinder describes a closure constraint on the assembly, or a kinematic constraint. Dimensional variations can accumulate statistically and propagate kinematically through the assembly, causing assembly and performance problems.

Geometric variation also plays a part in tolerance analysis. If the same parts were subjected to a change in their geometric features, then the gap dimension $U$ would also be affected. This is shown in an exaggerated form in Figure 2.14.
Local surface variation may cause the cylinder to sit on a peak or nestle in a valley in the sides of the groove. These variations can propagate through an assembly and accumulate in the same way the dimensional variations can. This simple example shows how all three sources of variation must be accounted for in order to insure that the tolerance analysis is realistic and accurate. The DLM meets these requirements.

The DLM uses 3-D, vector loops to create variational model of an assembly. In vector loop tolerance models, each vector represents a part, or component, dimension. The vectors are arranged into loops that represent those dimensions which stack together to determine the resultant assembly dimensions. The advantages of using vector loops include reducing the geometry to only the necessary parameters and the ability to use algebraic derivatives of the assembly function.

Each of the three sources of variations have a representation in the vector loop. The dimensional variations, which are due to manufacturing processes, are produced prior to assembly and are therefore considered to be independent, random variables. The geometric feature variation, which provide tolerance constraints on shape, orientation,
and location of part features, are also considered to be independent variables. They propagate similarly to the dimensional variation, but to a smaller degree. Despite this fact, they can sometimes be a significant source of tolerance stack-up. The kinematic variations, or small adjustments between mating parts which occur at assembly time, are a response to the two other sources of variation. Because kinematic variations occur only at assembly, they are considered to be dependent variables in the analysis. They are dependent upon both dimensional and geometric variation.

The DLM in [Chase, Gao, and Magleby 1998] is based on a first-order Taylor’s series expansion of the assembly kinematic constraint equation, with respect to both manufactured variables and assembly variables. The truncated Taylor’s series is solved, using linear algebra, in terms of the variations of the manufactured components. The kinematic constraint for a 3-D assembly can be represented by a closed loop. As the loop traces the dimensions from the beginning to the end, the dimensions of each part, as well as the translations and rotations of each joint, must sum to zero.

The 3-D loop equation from joint-to-joint can be represented mathematically by the homogeneous transformation matrices described in 2.1.2. The closed, assembly kinematic constraint equation is shown by equating the product of all the transformation matrices to the identity matrix, as shown in equation (2.19).

\[ [\mathbf{R}_1][\mathbf{T}_1][\mathbf{R}_2][\mathbf{T}_2]\ldots[\mathbf{R}_n][\mathbf{T}_n][\mathbf{R}_f] = [\mathbf{I}] \] (2.19)

Here \([\mathbf{R}_i]\) is the rotational transformation and \([\mathbf{T}_i]\) is the translational matrix for node \(i\) leading up to the final node \(f\). The matrix \([\mathbf{R}_f]\) is the final rotation needed to bring the entire loop back to closure. Each of these matrices are special cases of the 4 x 4 homogeneous matrix. Their representations are described below.
\[
[R_x] = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \phi_x & -\sin \phi_x & 0 \\
0 & \sin \phi_x & \cos \phi_x & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (2.20)

\[
[R_y] = \begin{bmatrix}
\cos \phi_y & 0 & \sin \phi_y & 0 \\
0 & 1 & 0 & 0 \\
-\sin \phi_y & 0 & \cos \phi_y & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (2.21)

\[
[R_z] = \begin{bmatrix}
\cos \phi_z & -\sin \phi_z & 0 & 0 \\
\sin \phi_z & \cos \phi_z & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (2.22)

\[
[T] = \begin{bmatrix}
1 & 0 & 0 & T_x \\
0 & 1 & 0 & T_y \\
0 & 0 & 1 & T_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (2.23)

According to [Gao, Chase, and Magleby 1998], the variables $\phi_x$, $\phi_y$, and $\phi_z$ represent relative rotations about their corresponding axes. The variables $T_x$, $T_y$, and $T_z$ are the components of the translation vector form one node of the loop to the next. The closed loop assembly constraint equation (2.19) is nonlinear, but for small variations about the nominal, the solutions can be approximated by using the derivatives of (2.19).
Previous to the GCM, a perturbation method for evaluating the derivatives of equation (2.19) was used. If the desired derivative is a translation or rotation at joint $i$, then a small, linear perturbation $\delta L$, or angle perturbation $\delta \phi$, is added to the original value and the matrix multiplication of (2.19) is performed. Equation (2.24) shows the result for a translation and (2.25) shows the rotation result.

$$
[R_1][T_i]...[R_n][T_i(L + \delta L)]...[R_n][T_n][0 \quad 0 \quad 0 \quad 1]^T = \{\Delta X \quad \Delta Y \quad \Delta Z \quad 1\}^T
$$ (2.24)

$$
[R_1][T_i]...[R_i(\phi + \delta \phi)][T_i]...[R_n][T_n][0 \quad 0 \quad 0 \quad 1]^T = \{\Delta X \quad \Delta Y \quad \Delta Z \quad 1\}^T
$$ (2.25)

From these equations, the derivatives can be approximated numerically. These approximations require intense computation time and the accuracy is dependent upon the size of the perturbation. For these reasons, the GCM is used to find the derivatives of the constraint assembly equation.

The linearization of the implicit assembly constraints shows where the derivatives of the assembly equation are needed. The first order Taylor’s series expansion of (2.19), the closed loop, kinematic constraints equation, can be written in matrix form.

$$
\{\Delta h\} = [A]\{\Delta x\} + [B]\{\Delta u\} = \{0\}
$$ (2.26)

The variables within (2.26) have the following definitions:

$\{\Delta h\}$ - The variations of the clearance

$\{\Delta x\}$ - The variations of the manufactured variables

$\{\Delta u\}$ - The variations of the assembly variables

$[A]$ - The first order, partial derivatives with respect to the manufactured variables

$[B]$ - The first order, partial derivatives with respect to the assembly variables

Each column of $[A]$ and $[B]$ have the following configurations.
\[ \{ A_i \} = \left\{ \frac{\partial H_x}{\partial x_i}, \frac{\partial H_y}{\partial x_i}, \frac{\partial H_z}{\partial x_i}, \frac{\partial H_{\omega_x}}{\partial x_i}, \frac{\partial H_{\omega_y}}{\partial x_i}, \frac{\partial H_{\omega_z}}{\partial x_i} \right\}^T \] (2.27)

\[ \{ B_i \} = \left\{ \frac{\partial H_x}{\partial u_i}, \frac{\partial H_y}{\partial u_i}, \frac{\partial H_z}{\partial u_i}, \frac{\partial H_{\omega_x}}{\partial u_i}, \frac{\partial H_{\omega_y}}{\partial u_i}, \frac{\partial H_{\omega_z}}{\partial u_i} \right\}^T \] (2.28)

In (2.27), \( x \) represents the \( i \)th manufactured dimension and \( u \) represents the \( i \)th assembly dimension. The derivative values contained in the \( [\mathbf{A}] \) and \( [\mathbf{B}] \) matrices, as well as the other subsequent variation sensitivity matrices, can be calculated using the GCM.

The Global Coordinate Method (GCM), presented in [Gao 1993], is a simplified way of calculating the partial derivatives and extracting sensitivities of the assembly loop equation. This is done through approximating the effects of small perturbations on the loop equation. The GCM replaces the need for using concatenated homogeneous transformation equations by representing 3-D assemblies by vector loops and relative rotations between vectors. The 3-D vector expression, shown in (2.29), is the foundation for the assembly vector loops. Note that \( \hat{i}, \hat{j}, \hat{k} \) are the unit direction vectors.

\[ V = X\hat{i} + Y\hat{j} + Z\hat{k} \] (2.29)

The GCM is derived through the differentiation of (2.29) where \( X, Y, Z \), and \( \hat{i}, \hat{j}, \hat{k} \) are functions of the arbitrary variable \( u \). Differentiating (2.29) gives equation (2.30), found in [Gao 1993].

\[ \frac{dV}{du} = \frac{dX}{du} \hat{i} + \frac{dY}{du} \hat{j} + \frac{dZ}{du} \hat{k} + \{ \hat{o} \times V \} \] (2.30)

Where

\[ \{ \hat{o} \times V \} = (\omega_y Z - \omega_z Y)\hat{i} + (\omega_z X - \omega_x Z)\hat{j} + (\omega_x Y - \omega_y Z)\hat{k} \] (2.31)
and \( \hat{\omega} = (\omega_x, \omega_y, \omega_z) \) are the direction cosine angles of the local axis of rotation to the origin. The first three terms in (2.30) represent the change of length of the vector \( V \), and the last cross-product term represents the rotation of vector \( V \) about its tail.

To find the partial derivatives with respect to translational and rotational variables, simple substitutions can be used. For the translational case, the substitution of \( u = L_i \) can be made in (2.30), which results in the loss of the cross-product term; it being a non-translational component. To evaluate the derivatives with respect to a rotational variable entails substituting \( u = \phi_i \) into (2.30). The rotational derivatives will not include any non-rotational components, which means the first three terms of (2.30) will drop out. The partial derivatives results, with translational and rotational substitutes, for the closed-loop assembly equation, \( H \), are shown in Table 2.2.

**Table 2.2 - Global Coordinate Method Derivative Equations**

<table>
<thead>
<tr>
<th>Translational Variables</th>
<th>Rotational Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{\partial H_x}{\partial L_i} = \cos \alpha )</td>
<td>( \frac{\partial H_x}{\partial \phi_i} = \omega_x Y - \omega_y Z )</td>
</tr>
<tr>
<td>( \frac{\partial H_y}{\partial L_i} = \cos \beta )</td>
<td>( \frac{\partial H_y}{\partial \phi_i} = \omega_x Z - \omega_z X )</td>
</tr>
<tr>
<td>( \frac{\partial H_z}{\partial L_i} = \cos \gamma )</td>
<td>( \frac{\partial H_z}{\partial \phi_i} = \omega_x X - \omega_z Y )</td>
</tr>
<tr>
<td>( \frac{\partial H_{x\theta}}{\partial L_i} = 0 )</td>
<td>( \frac{\partial H_{x\theta}}{\partial \phi_i} = \omega_x )</td>
</tr>
<tr>
<td>( \frac{\partial H_{y\theta}}{\partial L_i} = 0 )</td>
<td>( \frac{\partial H_{y\theta}}{\partial \phi_i} = \omega_y )</td>
</tr>
<tr>
<td>( \frac{\partial H_{z\theta}}{\partial L_i} = 0 )</td>
<td>( \frac{\partial H_{z\theta}}{\partial \phi_i} = \omega_z )</td>
</tr>
</tbody>
</table>
In Table 2.2, \( \cos \alpha \), \( \cos \beta \), and \( \cos \gamma \) are the direction cosines of the translation \( L_i \) to the global origin. The variables \( \omega_x \), \( \omega_y \), and \( \omega_z \) are the direction cosines of the axis of the rotation \( \phi_i \), while \( X, Y, \) and \( Z \) are the global coordinates of the joints. The terms \( H_x \), \( H_y \), and \( H_z \) represent the scalar sum of the translations in the global \( x, y, \) and \( z \) directions, with \( H_{\phi_x}, H_{\phi_y}, \) and \( H_{\phi_z} \) representing the sum of the global \( x, y, \) and \( z \) rotations. If the rotation is taken about the tip of \( V \) and the variations are referenced from the global coordinates, the signs of the three terms in equation (2.21) are reversed, as is shown in Table 2.2. A complete derivation of the GCM and its use in the DLM is found in [Gao 1993] and a summary is found in Appendix A.

Using the global coordinates of the joints and the direction cosines of both the vectors and the local joint axes, the derivatives of the assembly kinematic constraint equation with respect to translational and rotational variables can be calculated easily. The partial derivatives contained in Table 3.3, calculated by the GCM, can now be used to form the variation sensitivity matrices to be used by the DLM.

As stated in [Gao, Chase, and Magleby 1998], for the matrices [A] and [B] to have correctly mapped derivatives, each vector and rotation in the loop must be identified as a dependent or an independent variable. The loop requires a set of modeling rules that will assure proper relationships between the vectors that pass through each joint and joint axis. These rules, found in [Chase 1999], require certain joint angles to be held fixed and the kinematic rotations are identified as unknowns, as the manufactured angles vary within the tolerance limits. If these rules and relationships are consistent then algorithms can be determined, which will correctly perform the derivative mapping.

The variations of the assembly or kinematic variables can be solved for by algebraically manipulating (2.26). If matrix [B] is a full-ranked matrix then the following equation is the solution.
\[ \{ \Delta \mathbf{u} \} = -[\mathbf{B}]^{-1}[\mathbf{A}][\Delta \mathbf{x}] \]  \hspace{1cm} (2.32)

If the matrix \([\mathbf{B}]\) is singular, then the system is over-determined and a least squares fit must be used to solve the closed loop equation.

\[ \{ \Delta \mathbf{u} \} = -([\mathbf{B}]^T[\mathbf{B}])^{-1}[\mathbf{B}]^T[\mathbf{A}][\Delta \mathbf{x}] \]  \hspace{1cm} (2.33)

The same approach for closed loops can be applied to open loop kinematic constraint equations. The strategy involves solving the closed loop constraints first, and then substituting the solution into the open loop kinematic constraint. Then the variations of the open loop variables can be solved using the equations below.

\[ \{ \Delta \mathbf{v} \} = [\mathbf{C}][\Delta \mathbf{x}] + [\mathbf{D}][\Delta \mathbf{u}] \]  \hspace{1cm} (2.34)

The variables within (2.34) have the following definitions:

- \(\{ \Delta \mathbf{v} \}\) - The variation of the open loop assembly variables
- \([\mathbf{C}]\) - The partial derivatives with respect to the manufactured variables in the open loop
- \([\mathbf{D}]\) - The partial derivatives with respect to the assembly equations in the open loop

If \([\mathbf{B}]\) is full-ranked, then equation (2.32) can be substituted in (2.34).

\[ \{ \Delta \mathbf{v} \} = ([\mathbf{C}] - [\mathbf{D}][\mathbf{B}]^{-1}[\mathbf{A}])[\Delta \mathbf{x}] \]  \hspace{1cm} (2.35)

If \([\mathbf{B}]\) is singular, the same substitution yields the following.

\[ \{ \Delta \mathbf{v} \} = ([\mathbf{C}] - [\mathbf{D}][\mathbf{B}]^T[\mathbf{B}])^{-1}[\mathbf{B}]^T[\mathbf{A}][\Delta \mathbf{x}] \]  \hspace{1cm} (2.36)

The DLM can also incorporate geometric feature variation. The translation and rotation derivatives can be applied to (2.19) for feature variation as they were utilized above for kinematic variation. For a complete treatment of the subject see [Chase, Gao,
Magleby, and Sorensen 1996]. If geometric feature variations are included in the assembly equation, the linearized constraint equations may be modified.

\[
\{\Delta h\} = [A]\{\Delta x\} + [B]\{\Delta u\} + [F]\{\Delta \alpha\} = \{0\} \quad (2.37)
\]

\[
\{\Delta v\} = [C]\{\Delta x\} + [D]\{\Delta U\} + [G]\{\Delta \alpha\} \quad (2.38)
\]

Equation (2.37) is used in closed loops, (2.38) in open loops. The variables within (2.37) and (2.38) have the following definitions:

\{\Delta \alpha\} - The variations of the geometric feature variables

[F] - The partial derivatives with respect to the geometric feature variables for closed loops

[G] - The partial derivatives with respect to the geometric feature variables for open loops

The rest of the variables have the same definitions as those found in (2.26) and (2.34). As was done with the previous equations, the result for \{\Delta u\} can be substituted into (2.37) and (2.38) to yield the loop assembly variations. The closed loop equation is (2.39) and the open loop equation is (2.40).

\[
\{\Delta u\} = -[B]^{-1}[A]\{\Delta x\} - [B]^{-1}[F]\{\Delta \alpha\} \quad (2.39)
\]

\[
\{\Delta v\} = ([C] - [D][B]^{-1}[A])\{\Delta x\} + ([G] - [D][B]^{-1}[F])\{\Delta \alpha\} \quad (2.40)
\]

The accumulation of variation (variation stack-up) can now be estimated using either a worst case model or a statistical model. For a worst case model, equation (2.41) is applied and for a statistical model, (2.42) is used.
\[ \Delta u_i = \sum_{j=1}^{n} S^d_{ij} \text{tol}^d_{ij} + \sum_{j=1}^{m} S^a_{ij} \text{tol}^a_{ij} \leq T_{ASM} \quad (2.41) \]

\[ \Delta u_j = \sqrt{\sum_{j=1}^{n} \left( S^d_{ij} \text{tol}^d_{ij} \right)^2 + \sum_{j=1}^{m} \left( S^a_{ij} \text{tol}^a_{ij} \right)^2} \leq T_{ASM} \quad (2.42) \]

The variables in (2.41) and (2.42) have the following definitions:

- \( S^d \) - The tolerance sensitivity matrix for the dimensional variables
- \( S^a \) - The tolerance sensitivity matrix for the geometric feature variables
- \( \text{tol}^d \) - The tolerance vector for the dimensional variables
- \( \text{tol}^a \) - The tolerance vector for the geometric feature variables
- \( n \) - The number of dimensional variables
- \( m \) - The number of geometric feature variables
- \( T_{ASM} \) - The design limit for assembly variation \( \Delta u \),

The tolerance sensitivity matrices are calculated using the following equations. If the closed loop system is well-determined then (2.43) and (2.44) are used.

\[ [S^d] = -[B]^{-1} [A] \quad (2.43) \]

\[ [S^a] = -[B]^{-1} [F] \quad (2.44) \]

If the closed loop system is over-determined then (2.45) and (2.46) can be employed.

\[ [S^d] = -[(B^T [B])^{-1} B^T] [A] \quad (2.45) \]

\[ [S^a] = -[(B^T [B])^{-1} B^T] [F] \quad (2.46) \]

This method can again be applied to open loop systems. If they are well-determined then equations (2.47) and (2.48) are used.
\[ [S'] = [C] - [D][B]^{-1}[A] \] 
\[ [S^e] = [G] - [D][B]^{-1}[F] \] 

If the open loop system is over-determined then (2.49) and (2.50) can be applied.


According to [Gao, Chase, and Magleby 1998], the estimation of the assembly rejects is based on a Normal distribution assumption for the assembly variables. The estimate of a kinematic or assembly variation is treated as three standard deviations. This deviation can be taken together with the mean value of the kinematic or assembly variables to calculate, by integration or table, the assembly rejects for a given assembly batch when the limits for the assembly variables are given. In this way, a complete tolerance analysis can be performed on an assembly. For a complete tolerance analysis performed upon 2-D and 3-D assembly examples, see [Gao, Chase, and Magleby 1998] as well as [Chase, Gao, Magleby, and Sorensen 1996].

The information gained from a variation analysis using the DLM and the GCM can be employed to solve for constraints. The tolerance analysis background presented here is the basis for the remainder of the thesis. The variation sensitivities and sensitivity matrices will be employed in a constraint analysis method for analyzing the constraint condition of DOFs.
CHAPTER 3. CONSTRAINT ANALYSIS USING SCREW THEORY AND TOLERANCE SENSITIVITIES

The DLM tolerance modeling approach contains key information needed for constraint analysis within the paths and joint types used in the vector loops. An analysis of the sensitivity matrices can identify under- and overconstraints in an assembly, while simultaneously solving for the sensitivities of the dimensional, kinematic, and geometric variation. This chapter will show how these underconstrained motions and overconstrained DOFs can be found, by combining variation analysis and screw theory.

3.1 Current Tolerance DOFs Analysis

As stated in section 2.3, the DLM of tolerance analysis identifies variation as either an independent or a dependent variable in a vector loop equation. The variables associated with variation that arise from dimensional or geometric features are considered independent. Kinematic variation occurs during the assembly process as a response to the other two types of variation and are thus dependent. The partial derivatives of the vector loop with respect to the manufactured variables are stored in the [A] matrix. The derivatives with respect to the geometric feature variables are stored in the [F] matrix and the derivatives with respect to the dependent assembly in the [B] matrix.

These matrices contain constraint information due to the vector loop and its path. It is the path of the vector loops that assures all contributing dimensions and all kinematic variables are included in the resulting matrix equations. Also, the joints, either 2-D or 3-D, model the DOFs in the assembly. Through the built-in modeling rules for treating joints, the assembly constraints are defined and are already built into the loop equations.
The formation of tolerance vector loops are based upon the dimensions, contact points, and kinematic joint types of the assembly. The kinematic joint types are essential to constraint analysis, because they determine which DOFs are constrained or kinematic. Table 3.1, modified from [Chase, Gao, and Magleby 1998], contains commonly used 2-D and 3-D kinematic joint types and lists, in parenthesis, the total number of mobile DOFs. As vector loops are formed, they must also pass through part datums as well as joints.

The nature of the joint allows for some assembly DOF determination. Mobility information is found in the $[B]$ matrix, because the dependent variables will respond kinematically to dimensional and geometric variation. This means that the assembly

**Table 3.1 - 2-D and 3-D Kinematic Joint Types**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rigid/Fixed (0)</td>
<td>Planar (1)</td>
<td>Revolute (1)</td>
<td>Parallel Cylinders (1)</td>
</tr>
<tr>
<td>Cylinder Slider (2)</td>
<td>Edge Slider (2)</td>
<td>Rectangular Datum</td>
<td>Center Datum</td>
</tr>
<tr>
<td>Rigid (no motion)</td>
<td>Prismatic (1)</td>
<td>Revolute (1)</td>
<td>Parallel Cylinders (2)</td>
</tr>
<tr>
<td>Cylindrical (2)</td>
<td>Spherical (3)</td>
<td>Planar (3)</td>
<td>Edge Slider (4)</td>
</tr>
<tr>
<td>Cylindrical Slider (4)</td>
<td>Point Slider (5)</td>
<td>Spherical Slider (5)</td>
<td>Crossed Cylinders (5)</td>
</tr>
</tbody>
</table>
will kinematically adjust along the DOF directions allowed by the dependent variables contained in the \([B]\) matrix.

There are current methods for extracting basic DOF information from the \([B]\) matrix. Treating the columns of the \([B]\) matrix as indicators, it is possible to check the constraints of the system. Generalizations can currently be drawn from square or non-square \([B]\) matrices, however they are lacking in detail. The number of equations of the system is determined by the number of vector loops the assembly requires for analysis. The number of loops can be determined by subtracting the number of parts in the assembly from the number of assembly joints and adding one, as shown in [Chase 1999]. In 2-D problems there are three equations that can be formed from each loop, an X-equation, a Y-equation, and a \(\theta_z\)-equation. In 3-D there are six equations (the six DOFs in 3-D) extracted from each vector loop. A comparison of the number of equations and dependent unknown variables can yield a quick estimation of constraint status.

If the number of dependent variables in the tolerance assembly is equal to the number of loop equations, then the system may be exactly constrained. There still may be redundancies in the dependent variables. However, if the \([B]\) matrix of this system is square and non-singular, then the assembly is exactly constrained. This indicates that there is an equal number of unknown dependent variables and vector loop equations, allowing for an exact solution. When the two numbers are equal, and \([B]\) is not square, then the assembly can have multiple conflicting interpretations. The comparison is not robust enough to account for redundant constraints and make an accurate determination of over- or exactly constrained assemblies, as shown in the examples below.

This comparison of the columns of \([B]\) and the number of equations allowed by the assembly allows for more conclusions. When the number of equations is greater than the number of columns of \([B]\), the assembly is overconstrained, but no other information is found. Similarly, when the number of equations is less than the number of columns, there is an underconstraint. There is no information on the direction or location of this
underconstraint, except that one exists. If \([B]\) is square, more analysis can be performed to check for over- or underconstraints. By checking the rank of the entire square matrix and then checking each square sub-matrix, it is possible to find where the redundant or dependent columns are. But, because many variation analyses produce non-square \([B]\) matrices, this procedure is not very effective. Table 3.2 shows the conclusions from the equation-column comparison; \(E\) is the total number of equations defined by the loops and \(C\) is the number of columns of \([B]\).

Figure 3.1 shows examples of underconstrained, overconstrained, and exactly constrained assemblies along with their respective \(C\) and \(E\) numbers. Summing for the number of equations gives \(E\), counting the dependent variables yields \(C\), and comparing them result in the conclusions found in Figure 3.1. The example shown in Figure 3.2 exposes the limitations of this comparison method through an assembly whose true constraint condition is undetermined. Despite finding \(C\) and \(E\), the true nature of the assembly cannot be determined by this comparison alone because the overconstraint due to redundancy is offset by the underconstrained motion.

While certain constraint information can be found from \([B]\) matrix inspection, it is not sufficient. In non-square \([B]\) matrix comparisons, redundant DOFs are hidden from the comparisons and no information is presented on direction and position of any over- or underconstraints. These restrictions demonstrate the need for a more comprehensive variation-based constraint analysis.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Assembly Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C &gt; E)</td>
<td>Underconstrained</td>
</tr>
<tr>
<td>(C = E)</td>
<td>Exactly constrained / Overconstrained</td>
</tr>
<tr>
<td>(C &lt; E)</td>
<td>Overconstrained</td>
</tr>
</tbody>
</table>
**Figure 3.1 - Comparison Results: Under-, Over-, and Exactly Constrained Assemblies**

- **Independent**: a, b, r₁, r₂
- **Dependent (C)**: u₁, u₂, φ₁, φ₂
- **Number of Loops**: 1
- **Equations (E)**: 3
- **C = 4, E = 3, C > E**
- Underconstrained in t_c

- **Independent**: a, b, c, d, r₁, r₂, r₃
- **Dependent (C)**: u₁, u₂, u₃, φ₁, φ₂, φ₃
- **Number of Loops**: 2
- **Equations (E)**: 6
- **C = 6, E = 6, C = E**
- Exactly constrained

- **Independent**: a, b, c, d, e, f, r₁, r₂, r₃, r₄
- **Dependent (C)**: u₁, u₂, u₃, u₄, φ₁, φ₂, φ₃
- **Number of Loops**: 3
- **Equations (E)**: 9
- **C = 8, E = 9, C < E**
- Overconstrained in t_c

---

**Figure 3.2 - Comparison Results: Indeterminate Assembly**

- **Independent**: a, b, c, d, r₁, r₂
- **Dependent (C)**: u₁, u₂, u₃, u₄, φ₁, φ₂, φ₃
- **Number of Loops**: 2
- **Equations (E)**: 6
- **C = 6, E = 6, C = E**
- (underconstraint in t_c and overconstraint in t_c not detected)
3.2 Variation-based Constraint Analysis of Assemblies (VCAA)

The Variation-based Constraint Analysis of Assemblies (VCAA) method utilizes tolerance information and screw theory to examine constraints in assemblies. This method surpasses the limitations of previous variation-based techniques and constraint analyses can be performed using the information gained from a vector-loop tolerance analysis. VCAA is divided into underconstraint and overconstraint sections, because solving for each condition is accomplished using different means.

3.2.1 Underconstrained Motion Analysis

Solving for underconstraints in the VCAA method involves interpreting the information gained from an assembly tolerance model using screw theory. As stated in section 2.2.3, a twist can be formed between two kinematic joints. In this manner, twists represent the possible locations of velocities, both angular and linear, that can be transmitted through the mating joints. This transmission-of-velocity concept is a key to extracting constraint information from tolerance loops.

3.2.1.1 Kinematic Velocity and Tolerance Variation Analogy

A key in searching for underconstraints in assemblies using screw theory is to establish twists for mating joints. The twist matrix shows the DOFs through which velocities can be transmitted. Thus, to make the connection between tolerances and underconstraints requires an association between variation and velocity.

The Tolerance Analysis using Kinematic Sensitivities (TAKS) method demonstrates an analogy between tolerance variation sensitivities and kinematic sensitivities. Developed in [Faerber 1999], the TAKS technique sets forth a methodology for gathering the tolerance sensitivities from a modified kinematic analysis. The analogy developed in the TAKS method relates the tolerance variations with kinematic velocities by comparing the relationship between tolerance analysis vector loops and the velocity equations found in kinematic vector loops. Aspects of the analogy between velocity and
variation are listed in Table 3.3, which was modified from [Faerber 1999].

Equivalent variational mechanisms (EVMs) are used in kinematic vector loops to represent the dimensional and kinematic variations involved in tolerance vector loops. The EVMs are included in the kinematic vector loop and an instantaneous velocity analysis is performed. From this velocity analysis, tolerance sensitivities can be calculated. As found in [Faerber, 1999], TAKS shows that velocities can model variation and give the same results as a conventional tolerance analysis.

The principal application of the TAKS analogy to VCAA is the relationship between variation and velocity. An example in [Faerber 1999] shows how to form [A] and [B] matrices analogous to the tolerance analysis matrices. In the analysis of a one-way clutch, [Faerber 1999] showed how to solve a tolerance problem using a kinematic vector loop analysis. The one-way clutch assembly, analyzed using the DLM in [Gao 1993], is shown in Figure 3.3. Because of symmetry, only one fourth of the clutch, shown in Figure 3.4, needs to be analyzed.

<table>
<thead>
<tr>
<th>Table 3.3 - Comparisons of Tolerance and Kinematic Analyses</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tolerance Analysis</strong></td>
</tr>
<tr>
<td>-----------------------</td>
</tr>
<tr>
<td><strong>Model:</strong> Vector loop w/ kinematic joints</td>
</tr>
<tr>
<td><strong>Describes:</strong> Probable assembly variation</td>
</tr>
<tr>
<td><strong>Input:</strong> Dimensional variations in angles and link lengths</td>
</tr>
<tr>
<td><strong>Output:</strong> Dependent dimensional variations in assembly angles and lengths</td>
</tr>
</tbody>
</table>
The example solves the tolerance assembly equation, using the DLM, as well as solving the velocity loop equation. The 2-D tolerance vector loop equation is found in (3.1) while the 2-D kinematic position loop equation is (3.2).

Assembly Equation from Variation Analysis:

\[
ae^{i\theta_1} + be^{i(\theta_1 + \alpha_b)} + c_1 e^{i(\theta_1 + \alpha_b + \alpha_{c1})} + c_2 e^{i(\theta_1 + \alpha_b + \alpha_{c1} + \alpha_{c2})} + f e^{i(\theta_1 + \alpha_b + \alpha_{c1} + \alpha_{c2} + \alpha_f)} = 0
\] (3.1)

Position Equation from Kinematics:

\[
ae^{i\phi} + be^{i\theta_2} + c_1 e^{i\theta_1} + c_2 e^{i\theta_2} + f e^{i\theta_f} = 0
\] (3.2)

The DLM requires the derivatives of the tolerance loop equation with respect to the independent and dependent variables. The velocity analysis calls for the derivatives of the kinematic position equation with respect to time. The derivative equation for the tolerance loop is found in (3.3) and for the kinematic equation in (3.4). In the clutch example, the critical assembly feature is the pressure angle, $\phi$. This variable is introduced into the analysis by substituting in the relationship between the angles $\phi$ and
\( \alpha_{c_1}, (\phi = 360^\circ - \alpha_{c_2} \text{ and } d\phi = -d\alpha_{c_2}) \), which is appropriate as the limits of \( \phi \) are the assembly design requirement. By allowing the kinematic variables \( a, c \) (where \( c = c_1 = c_2 \), due to the uniform cylinder radius), and \( f \) to change with time, such that \( \dot{a}, \dot{c} \) (where \( c = \dot{c}_1 = \dot{c}_2 \), and \( \dot{f} \neq 0 \)), this allows the dimensional variation to be modeled by a kinematic analysis.

Variation Equation:

\[
\begin{align*}
\dot{a}e^{i\theta_1} + \dot{b}e^{i(\theta_1 + \phi_0)} + \dot{c}e^{i(\theta_1 + \alpha_b + \alpha_{c_1} - \phi)} + \dot{d}e^{i(\theta_1 + \alpha_b + \alpha_{c_1} - \phi + \phi_f)}
- \dot{d}\phi(cie^{i(\theta_1 + \alpha_b + \alpha_{c_1} - \phi)} + fie^{i(\theta_1 + \alpha_b + \alpha_{c_1} - \phi + \phi_f)}) + \dot{f}e^{i(\theta_1 + \alpha_b + \alpha_{c_1} - \phi + \phi_f)} = 0
\end{align*}
\]  
(3.3)

Velocity Equation:

\[
\begin{align*}
\dot{a}e^{i\theta_1} + \dot{b}e^{i\theta_0} + \dot{c}(e^{i\theta_1} + e^{i\theta_2}) + \dot{\theta}_2(cie^{i\theta_2} + fie^{i\theta_f})
+ \dot{f}e^{i\theta_f} = 0
\end{align*}
\]  
(3.4)

In (3.3), the variables \( da, db, dc, df \), and \( d\phi \), represent small changes in dimensions, while in equation (3.4) the variables \( \dot{a}, \dot{b}, \dot{c}, \dot{f} \), and \( \dot{\theta}_{c_2} \) represent the time rate of change of \( a, b, c, f, \) and \( \theta_{c_2} \). When the equations above are resolved into X and Y equations (using Euler's formulas) then two scalar equations are formed. The scalar equations can then be arranged in matrix form, which collect the independent and dependent variables. The matrix representations are found in (3.5) and (3.6) for the tolerance problem and the velocity problem, respectively.

Linearized Variation Equation:

\[
\begin{bmatrix}
\cos \theta_1 & (\cos \theta_{c_1} + \cos \theta_{c_2}) & \cos \theta_2 \\
\sin \theta_1 & (\sin \theta_{c_1} + \sin \theta_{c_2}) & \sin \theta_2
\end{bmatrix}
\begin{bmatrix}
da \\
dc \\
df
\end{bmatrix}
+ \begin{bmatrix}
\cos \theta_b & (c \sin \theta_{c_2} + f \sin \theta_f) \\
\sin \theta_b & (-c \cos \theta_{c_2} - f \cos \theta_f)
\end{bmatrix}
\begin{bmatrix}
\db \\
d\phi
\end{bmatrix}
= 0
\]  
(3.5)

where
\[
\begin{align*}
\theta_b &= \theta_1 + \alpha_b, \quad \theta_{c_1} = \theta_1 + \alpha_b + \alpha_{c_1}, \quad \theta_{c_2} = \theta_1 + \alpha_b + \alpha_{c_1} - \phi, \\
\theta_f &= \theta_1 + \alpha_b + \alpha_{c_1} - \phi + \alpha_f
\end{align*}
\]
Matrix Velocity Equation:

\[
\begin{bmatrix}
\cos \theta_1 & (\cos \theta_1 + \cos \theta_2) & \cos \theta_f \\
\sin \theta_1 & (\sin \theta_1 + \sin \theta_2) & \sin \theta_f \\
\end{bmatrix}
\begin{bmatrix}
\dot{a} \\
\dot{c}_1 \\
\end{bmatrix}
+ \begin{bmatrix}
\cos \theta_b & (-c_2 \sin \theta_1 - f \sin \theta_f) \\
\sin \theta_b & (c_2 \cos \theta_1 + f \cos \theta_f) \\
\end{bmatrix}
\begin{bmatrix}
\dot{b} \\
\dot{\theta}_{c2} \\
\end{bmatrix} = 0
\] (3.6)

The matrix representations can then take the form of the tolerance equation (3.7) and the velocity equation (3.8).

\[
[A]_t \begin{bmatrix}
da \\
dc \\
df \\
\end{bmatrix} + [B]_t \begin{bmatrix}
db \\
d\phi \\
\end{bmatrix} = 0
\] (3.7)

\[
[A]_v \begin{bmatrix}
\dot{a} \\
\dot{c}_1 \\
\dot{f} \\
\end{bmatrix} + [B]_v \begin{bmatrix}
\dot{b} \\
\dot{\theta}_{c2} \\
\end{bmatrix} = 0
\] (3.8)

In equation (3.7), the independent dimensional variation variables are represented in \([A]_t\), while the dependent assembly variables are represented in \([B]_t\). Similarly, in equation (3.8), the independent instantaneous velocity sensitivities are found in \([A]_v\), while the dependent velocity sensitivities are in \([B]_v\). Solving equations (3.7) and (3.8) yield the tolerance and kinematic sensitivities, respectively. It is shown in [Faerber 1999] that the two analyses give equivalent sensitivity matrices when the velocity model includes dimensional variation and relative angles. Thus, it is possible to perform a tolerance analysis on an assembly by modeling it using EVMs in a kinematic software package.

A unique interpretation of the TAKS analogy leads to a significant principle used in the VCAA method. The velocity analysis, leading up to equation (3.8), stores the dependent velocity sensitivities in the \([B]_v\) matrix. The values contained in \([B]_v\)
correspond to the joints in the assembly through which velocities can be transmitted. They represent the dependent velocity sensitivities and the locations in the assembly along which velocity will be imparted to adjust for velocity changes in the independent variables according to the sensitivities contained in $[A]$.

The TAKS method demonstrates that the values of the sensitivity matrices in (3.7) and (3.8) are equivalent. The velocity-variation relationship thus leads to an important constraint interpretation when the analogy is reversed. The tolerance analysis which yields equation (3.8) as well as the tolerance sensitivities will also provide the kinematic sensitivities. Hence, the tolerance matrices contain kinematic velocity sensitivities.

Using the fact that the matrices in (3.8) are the same as the ones in (3.7), it can be shown that the $[B]$, matrix obtained from a tolerance analysis also identifies the joints in the assembly through which velocities can be transmitted. Because the values in $[B]$, are the dependent variables, they represent the locations in the assembly where variation is absorbed and where velocity is allowed. Because the $[B]$, matrix contains the velocity sensitivities and information about the assembly velocities, it also contains the necessary information to produce twist matrices for use in underconstraint screw analysis. Therefore, assembly twist matrices can be formed using the tolerance analysis $[B]$, matrix.

3.2.1.2 Variation Matrix Twist Formation

From the variation-velocity analogy it is possible to extract and assemble twist matrices using the values contained in the $[B]$ matrix of a tolerance analysis. As stated in section 2.3, $[B]$ contains the first order, partial derivatives of the assembly equation with respect to the assembly variables. Equation (2.28), reprinted as (3.9), shows the configuration of each column of $[B]$ where $u_i$ represents the $i$th assembly dimension.
\[
\{ B_i \} = \begin{bmatrix}
\frac{\partial H_x}{\partial u_i} & \frac{\partial H_y}{\partial u_i} & \frac{\partial H_z}{\partial u_i} & \frac{\partial H_{\alpha x}}{\partial u_i} & \frac{\partial H_{\alpha y}}{\partial u_i} & \frac{\partial H_{\alpha z}}{\partial u_i}
\end{bmatrix}^T
\]  

(3.9)

Applying the TAKS analogy, each of these derivatives can be treated as if they are velocity sensitivities, both linear and angular. Each column of \([B]\) then has the configuration found in (3.10).

\[
\{ B_i \} = \begin{bmatrix} v_{x_i} & v_{y_i} & v_{z_i} & \omega_{x_i} & \omega_{y_i} & \omega_{z_i} \end{bmatrix}^T
\]  

(3.10)

The configuration of columns in \([B]\) now closely resembles the velocity sensitivities of a twist matrix as shown in equation (2.5). The only operation needed to make them identical is to switch the first three elements of the column with the last three. Performing the switch yields equation (3.11), which can be compared to equation (2.5), reprinted as (3.12).

\[
\{ T_{B_i} \} = \begin{bmatrix} \omega_{x_i} & \omega_{y_i} & \omega_{z_i} & v_{x_i} & v_{y_i} & v_{z_i} \end{bmatrix}^T
\]  

(3.11)

\[
T = \begin{bmatrix} \omega_x & \omega_y & \omega_z & v_x & v_y & v_z \end{bmatrix}
\]  

(3.12)

Each dependent assembly variable accounts for a column in \([B]\), and therefore each produces a twist matrix, subsequently identified as column-twists. It should be noted that both the \([B]\) matrix and the twist matrices do not contain any actual velocities, just the DOF directions along which velocities may be transmitted. Using these velocity sensitivities, the column-twists of \([B]\) can be thus analyzed for allowed motions using the algorithms for twists found in [Adams and Whitney 2001] and [Konkar 1993].
The [B] matrix of the one-way clutch example, shown in Figure 3.3, will be analyzed to show the motions allowed by each dependent variable. The tolerance sensitivities for the clutch have been analyzed in [Gao 1993], by the DLM. The tolerance vector loop for the clutch is shown in Figure 3.5 and the nominal dimensions are listed in Table 3.4. The assembly variables of the clutch are the location of the roller, b, and the pressure angles, \( \phi_1 \) and \( \phi_2 \). From the partial derivatives of the assembly vector loop equation, \( H \), the values for the [A] and [B] matrices can be calculated using the Global Coordinate Method reviewed in section 2.3 along with the information contained in Table 3.4. The [B] matrix for the clutch, converted to 3-D, is shown in equation (3.13).

![Figure 3.5 - Vector Loop Model of One-way Clutch Assembly. Taken from [Gao 1993]](image)

<table>
<thead>
<tr>
<th>Joint Name</th>
<th>Orientation</th>
<th>Joint Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a ) - Joint 1 ( \phi_2 )</td>
<td>( \theta = 90^\circ )</td>
<td>( X = 0.0 ) ( Y = 0.0 )</td>
</tr>
<tr>
<td>( b ) - Joint 2</td>
<td>( \theta = 0^\circ )</td>
<td></td>
</tr>
<tr>
<td>( c_1 ) - Joint 3</td>
<td>( \theta = 90^\circ )</td>
<td></td>
</tr>
<tr>
<td>( c_2 ) - Joint 4 ( \phi_4 )</td>
<td>( \theta = 82.982^\circ )</td>
<td>( X = 4.8105 ) ( Y = 39.075 )</td>
</tr>
<tr>
<td>( e ) - Joint 5</td>
<td>( \theta = -97.018^\circ )</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4 - Nominal Dimensions of One-way Clutch

61
\[
[B] = \begin{bmatrix}
\frac{\partial H_x}{\partial \phi_1} & \frac{\partial H_x}{\partial \phi_2} & \frac{\partial H_x}{\partial \theta_x} \\
\frac{\partial H_y}{\partial \phi_1} & \frac{\partial H_y}{\partial \phi_2} & \frac{\partial H_y}{\partial \theta_y} \\
\frac{\partial H_z}{\partial \phi_1} & \frac{\partial H_z}{\partial \phi_2} & \frac{\partial H_z}{\partial \theta_z} \\
\frac{\partial H_{\phi_1}}{\partial \theta_x} & \frac{\partial H_{\phi_2}}{\partial \theta_x} & \frac{\partial H_{\theta_x}}{\partial \theta_x} \\
\frac{\partial H_{\phi_1}}{\partial \theta_y} & \frac{\partial H_{\phi_2}}{\partial \theta_y} & \frac{\partial H_{\theta_y}}{\partial \theta_y} \\
\frac{\partial H_{\phi_1}}{\partial \theta_z} & \frac{\partial H_{\phi_2}}{\partial \theta_z} & \frac{\partial H_{\theta_z}}{\partial \theta_z}
\end{bmatrix}
= \begin{bmatrix}
1 & -39.075 & 0 \\
0 & 4.8101 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & -1 & 1
\end{bmatrix}
\] (3.13)

The columns of \([B]\) can now be treated as twists if \([B]\) is transposed and the first three elements of each column are switched with the last three elements. Performing those operations gives the column-twists found in equations (3.14), (3.15), and (3.16).

\[
T_b = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\] (3.14)

\[
T_{\phi_1} = \begin{bmatrix}
0 & 0 & -1 & -39.075 & 4.8105 & 0
\end{bmatrix}
\] (3.15)

\[
T_{\theta_x} = \begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\] (3.16)

The interpretation of the column-twists gives remarkable results as to the motion allowed within the clutch assembly. Each twist shows a different allowed motion based on the mobile DOFs. The first column-twist, \(T_b\), shows a pure translation exists in the global X direction. Looking at vector \(b\), it can be seen that the roller will translate along the hub in the X direction; a motion based on the cylindrical slider joint at that vector. The \(T_{\phi_1}\) column-twist shows a rotation around the Z-axis direction, based on the dependent angle, \(\phi_1\). Using the point algorithm found in [Adams 1998], the global location of the Z-rotation is (4.8105, 39.075, 0), or the center of the roller. This twist is saying that the roller is free to rotate in the Z direction around its center. The last
column-twist $\mathbf{T}_{\phi_2}$ shows a rotation in $Z$ located at $(0, 0, 0)$, which is the rotation of the hub, around its center, based on the dependent variable $\phi_2$.

All of this underconstraint information originated from the tolerance analysis of the assembly based on dependent assembly variation. The sensitivities calculated by using the Global Coordinate Method yield the very information that is needed for a screw-based analysis. Finding the translations and the locations of the rotations as part of the sensitivities is due to the unique way the GCM calculates them. Interpreting the columns of the $[\mathbf{B}]$ matrix as column-twists unlocks information on the mobile DOFs and leads to the VCAA method of finding underconstraints for parts within assemblies.

3.2.1.3 Variation Twist Analysis of Assemblies

In [Adams 1998], several mating conditions were presented along with their twist representations. Some of the joints that are common to DLM tolerance analysis, as seen in Table 3.1, did not have twist representations in [Adams 1998]. Because the tolerance matrices contain twist information, all of the joint types in Table 3.1 need twist representations so that the tolerance models of assemblies can be examined using VCAA. Twist representations are important in underconstraint analysis and are listed below for both 2-D and 3-D joints. The abbreviated form of the twists is used below, which identifies the translation and rotation triplets with $v$ and $\omega$, respectively.

The equations contained within these twist representation figures are for use in the screw-based constraint analysis method summarized in [Adams and Whitney 2001]. They are based on the homogeneous transformation matrices mapping the location of the global axes to the local axes of each joint type. The VCAA method will only require the type of DOF included in the twist (e.g., translation or rotation) and will not require actual manipulation of the homogeneous transformations and the equations within the figures. The equations are presented here for use in later comparison. Also, in these joint twists, the coordinate axes represent the local axes of each joint. The VCAA method only needs the global information gained from a tolerance analysis and does not need the local axes.
The joint twist matrices are presented in groups based on the number of allowed DOFs. It should be noted that the twist matrix for each joint has the same number of rows as number of allowed DOFs. The 2-D rigid or fixed joint, see Figure 3.6, will have an empty twist as it allows no motion.

\[
T = \begin{bmatrix} 0 & : & 0 \end{bmatrix}
\]

**Figure 3.6 - Joint Twist with 0 DOFs in 2-D**

The 2-D one DOF joints and their associated twists are shown in Figure 3.7. When using the method described in [Adams 1998], the rotational sub-matrix \([R]\) is obtained from corresponding homogeneous transformations, which transform the joints from global coordinates to local joint coordinates. The planar joint depends on the unit vector \(\vec{k}\), the direction of the translation. The revolute joint requires the location of the pin joint as well as \(\vec{d}\), the direction of the rotation. In the parallel cylinders joint each cylinder can rotate independently, as long as the contact between the two is maintained, the twist rotation is located at the center of either cylinder. Using the VCAA method does not require any of the equations found in the figures, but are included to be used in a later comparison.
The two DOF joint twists are listed in Figure 3.8 and represent the last of the 2-D joint types. The only difference between the two joints is the location of the rotation. In the cylinder slider the rotation occurs around the center of the cylinder. In the edge slider the rotation is centered around the edge.
**Figure 3.8 - Joint Twists with 2 DOFs in 2-D**

The twist representations for 3-D joint types are similar to the 2-D versions and require the translation and rotation information from the homogeneous transformations. The locations of rotations and the directions of translations with respect to the local coordinate systems are needed. The 3-D rigid joint is shown in Figure 3.9 and it, like the 2-D version, also has an empty twist.

**Figure 3.9 - Joint Twist with 0 DOFs in 3-D**

The one DOF joint twists, shown in Figure 3.10, are the prismatic and revolute joints. The twists are identical to the 2-D joint versions and also need the same unit vectors and rotations. Also, the revolutions and translations for the rest of the following joints are calculated for use in the method found in [Adams 1998] in the same manner as done for the prismatic and revolute joints. Again, the VCAA method only needs the type of motion allowed (e.g. rotation or translation), which is also shown in the joint twist figures.
The two DOF joints, shown in Figure 3.11, include the parallel cylinders and cylindrical joints. The location of the rotation for the parallel cylinders is the center of either of the two cylinders, as they rotate independently of one another, holding contact.

**Figure 3.10 - Joint Twists with 1 DOF in 3-D**

\[
T = \begin{bmatrix}
0 & \hat{v}_x \\
\hat{v}_x & (R \hat{v}_x)^T
\end{bmatrix}
\]

\[
\hat{v}_x = (R \hat{v}_x)^T
\]

\[
R = \begin{bmatrix}
1 & 0 & 0
\end{bmatrix}
\]

---

**Figure 3.11 - Joint Twists with 2 DOFs in 3-D**

\[
T = \begin{bmatrix}
\hat{\omega}_x & \hat{\omega}_y & \hat{\omega}_z \\
0 & \hat{\omega}_x & \hat{\omega}_y \\
0 & 0 & \hat{\omega}_z
\end{bmatrix}
\]

\[
\hat{\omega}_x = r \times \hat{\omega}_z
\]

\[
r = \text{pivot location}
\]
The next set has three free DOFs, Figure 3.12, and includes the spherical joint and the planar joint. The spherical joint rotations are based on the center location and the three rotational unit vectors. Each \( \omega_i \) and \( v_i \) uses the cross product of the center location and the rotational vectors to find the twist. These unit rotations are \( \omega_1 = [1 \ 0 \ 0] \), as well as \( \omega_2 = [0 \ 1 \ 0] \) and \( \omega_3 = [0 \ 0 \ 1] \). The planar joint simply has the one rotation and two translations.

\[
T = \begin{bmatrix}
\omega_1 & : & v_1 \\
\omega_2 & : & v_2 \\
\omega_3 & : & v_3
\end{bmatrix}
\]

\[
\omega_1 = \begin{bmatrix}1 & 0 & 0\end{bmatrix}
\]

\[
v_1 = r \times \omega_1
\]

\[
r = [c_1 \  c_2 \  c_3], \text{ etc}
\]

\[
T = \begin{bmatrix}
\dot{\omega}_z & : & \ddot{v}_z \\
0 & : & \ddot{v}_x \\
0 & : & \ddot{v}_y
\end{bmatrix}
\]

Figure 3.12 - Joint Twists with 3 DOFs in 3-D

The next set of joints involve those that have four remaining DOFs, which include the edge slider and cylindrical slider. Both joints, as seen in Figure 3.13, have the same twist matrix representation. The differences are in the axis of rotation in the X-direction. The edge slider rotates about the point of contact of the edge and the cylindrical slider rotates around the center axis of the cylinder.
Figure 3.13 - Joint Twists with 4 DOFs in 3-D

The final set of joint twists, which allows for five DOFs, includes the point slider, the spherical slider, and the crossed cylinder joints. As with the four DOF joints, the difference between the point slider and spherical slider is in the location of the axes of rotation. The point slider rotates in two axes about the point contact. The spherical slider also rotates in two axes around the spherical radius. The third joint, crossed cylinders, contains rotations about the center axes of the cylinders and translations in the direction of the center axes of the cylinder. All three of these joints and accompanying twists are shown in Figure 3.14.
Figure 3.14 - Joint Twists with 5 DOFs in 3-D

In the previous screw-theory based constraint analysis, found in [Adams 1998], all of the twist representations of the common joint types are used to form twists from the locations of the joints. The features are located by means of homogenous transformations from the global coordinate system to the feature. The VCAA method looks at the twist representation of the joint type and assembles the associated twists by extracting the information from the \([B]\) matrix.

The column-twists of \([B]\) have motions that are based on the individual dependent variables of the assembly. The column-twists alone do not describe the twists of the joint types within the assembly. Using the joint twist types listed in Figures 3.6-3.14, it is
possible to form the twist matrices for each joint type in the assembly. This is accomplished by inspecting each joint and using the column-twists of $[B]$ to develop the joint twist representation. The joint twist is formed through a union of the columns of the dependent variables that are associated with it. Therefore, each joint twist is just a combination of the columns of $[B]$.

3.2.1.4 Application to One-way Clutch Example

In the one-way clutch example, the twists for each joint type can be formed using the columns of its $[B]$ matrix. There is a parallel cylinders joint that exists between the roller and the outside ring. The joint representation in Figure 3.7 shows that there is one row in the matrix, signifying the one rotational DOF. Looking at the dependent variable associated with the joint and extracting the related column, it is possible to form the joint twist. There are two dependent variables that are associated with the parallel cylinder joint. Because the two cylinders can rotate independent of one another, as long as contact is maintained, there are two column-twists associated with this joint. The first is based on the ring’s rotation angle of $\phi_1$, the second on the roller’s rotation angle of $\phi_2$. The twists for this joint are the $T_{\phi_1}$ and $T_{\phi_2}$ twist depicted in equations (3.15) and (3.16). The resulting joint twists are shown in equations (3.17) and (3.18).

$$T_{\text{parcyl}_1} = \begin{bmatrix} T_{\phi_1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & -39.075 & 4.8105 & 0 \end{bmatrix} \quad (3.17)$$

$$T_{\text{parcyl}_2} = \begin{bmatrix} T_{\phi_2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (3.18)$$

The next joint is the cylinder slider located at the contact of the roller and the hub. The joint twist, in Figure 3.8, is a two row, one rotational and one translational DOF matrix. The two dependent variables associated with this joint are the vector $b$, the translation, and the angle $\phi$, the rotation. The resultant joint twist, equation (3.19), is then the union of the $T_{\phi}$ and $T_{b}$ column-twists.
\[
T_{\text{cylslider}} = \begin{bmatrix} T_h \\ T_\phi \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & -39.075 & 4.8105 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}
\] (3.19)

The last joint is a revolute joint representing the rotation of the hub within the ring. The associated dependent variable is the angle \( \phi_2 \) and the joint twist is equation (3.20).

\[
T_{\text{revolute}} = \begin{bmatrix} T_\phi \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}
\] (3.20)

Using these twists, the parts of the assembly can now be analyzed for constraints by utilizing methods based on the screw theory analysis summarized in section 2.2.3 of this thesis. Finding the resultant twist of the parts determines their underconstraints. Assuming that the joints of the parts cannot break contact with each other, the analysis is performed on each part, one at a time, to determine the underconstraints of each part. This analysis will be described in depth in Chapter 4. The resultant twists for the three parts are given in equation (3.21).

\[
\begin{align*}
T_{\text{Resultant--Roller}} &= \begin{bmatrix} 0 & 0 & 1 & 39.075 & -4.8105 & 0 \end{bmatrix} \\
T_{\text{Resultant--Ring}} &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \\
T_{\text{Resultant--Hub}} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}
\end{align*}
\] (3.21)

Employing the point algorithm discussed in section 2.2.3, \( T_{\text{Resultant-Roller}} \) shows a rotation about the global Z-axis located at the point (4.8105, 39.0750, 0) or the center of the roller. The second triplet of the twist gives location information, but the coordinates are reversed and the sign is changed. The reason for this is explained in the derivation of the Global Coordinate Method, in Appendix A. The roller is free to rotate independent of the rest of the assembly. The resultant twist \( T_{\text{Resultant-Ring}} \) also shows that the ring has an underconstrained rotation about the Z-axis located at (0, 0, 0), which is its center point. The twist \( T_{\text{Resultant-Hub}} \) shows that the hub part has no underconstraint.
Further, detailed underconstrained analysis of different parts and assemblies as well as the methodology used to analyze and identify underconstraints will be shown step-by-step in Chapter 4.

3.2.1.5 Conclusions for Underconstrained Motion Analysis using VCAA

The Variation-based Constraint Analysis for Assemblies (VCAA) method takes the information gained from a variation analysis of an assembly and solves for the underconstrained motions within the assembly. This is accomplished by using the analogy developed in [Faerber 1999] between variation and velocity. This analogy shows that the variation analysis [B] matrix gives the directions along which velocities can be transmitted by the assembly; they can propagate through the dependent assembly variables. This analogy interpretation thus shows that twist matrices and underconstraint information exists within the columns of the [B] matrix.

To make the [B] columns correspond to the twist matrix configuration, they must first be transposed. Secondly, the first three elements in the rows (what were the [B] columns) must be switched with the last three elements. The dependent variable [B] matrix column-twists can then be used to develop the joint twist matrices for the kinematic joints modeled within the assembly. Each joint type, either 2-D or 3-D, has an associated twist representation, which shows what DOFs are allowed to move. The column-twists formed from [B] make up the joint twists. This is done by observing which dependent variables are associated with the kinematic joint and then using the corresponding column-twists to fill in the rows of the joint twists. The twist analysis methods discussed in [Adams 1998] can then be employed to evaluate the assembly for underconstraints.

3.2.2 Overconstrained DOFs Analysis

Identifying overconstraints using the VCAA method requires a similar approach to that used for underconstraints. Screw theory, as stated in section 2.2.3, says that a

73
wrench formed between two mating parts represents the possible DOF directions through which forces and moments can be transmitted. Just as the variation-velocity analogy was significant to finding underconstraints, a relationship between geometric variation and forces yields needed overconstraint information.

3.2.2.1 Geometric Variation and Force-Moment Relationship

A key in finding overconstraints within assemblies using screw theory is the requirement of building wrenches between the mating joints. The wrench matrix shows the joint DOFs through which forces and moments can be transmitted. Hence, to produce a correlation between tolerance analysis and overconstraints requires a relationship between forces and variation.

Geometric feature variation analysis of assemblies (which have geometric tolerance callouts) contains the information needed to fashion this variation-force relationship. In [Chase, Gao, Magleby, and Sorensen 1996], a method for incorporating geometric feature variations in a tolerance analysis of assemblies was set forth. As summarized in section 2.3, geometric variation can be accounted for in a variation analysis by including them in the DLM vector loops. Equation (2.34), here reprinted as (3.22), includes effects of geometric feature variation in the assembly equations for a closed loop. In (3.22), \( \{ \Delta \alpha \} \) represents the variations of the geometric feature variables and the \([F]\) matrix contains the partial derivatives of the loop equation with respect to the geometric feature variables.

\[
\{ \Delta h \} = [A]\{ \Delta x \} + [B]\{ \Delta u \} + [F]\{ \Delta \alpha \} = \{0\}
\]  

(3.22)

A clear understanding of what the geometric feature variables represent leads to a variation-force relationship. Local geometric variation, as seen in Figure 2.14, cause parts to mate on peaks or valleys of the contact surfaces. These variations will propagate through the assembly as translations or rotations (or both) at the different joints. The important aspect of these variations is the direction of propagation. Each of the 2-D and
3-D joint types in Table 3.1 has a certain number of kinematic DOFs, listed in parenthesis, which allow motion. Each joint also has a number of constrained DOFs through which motion is not allowed. The constrained DOFs are the means by which geometric feature variation is propagated.

The set of kinematic DOFs and constrained geometric feature DOFs for a joint are mutually exclusive and are related by the total number of DOFs; in 3-D, the number of kinematic DOFs and number of feature DOFs, for each joint, add to six. Geometric variation propagates through the feature DOF directions of the joints. Figure 3.15 shows two 3-D joints, the cylindrical slider and planar, with their corresponding kinematic and feature DOFs labeled.

![Figure 3.15 - DOFs for Kinematic Motions (k) and Geometric Feature Variations (f). Modified from [Chase, Gao, Magleby, and Sorensen 1996]](image)

Variation that occurs in any of the feature DOF directions will be accounted for in the $[F]$ matrix of tolerance analysis. Non-zero entries in $[F]$ signify that the geometric feature variation can propagate in the feature DOF directions of the joints. This is the variation part of the variation-force relationship.

The force part of the relationship comes from observing what happens if forces and moments are applied to the joint types. If a force or moment is applied to a joint along the kinematic DOF directions, then a linear or angular motion results. In the kinematic directions, the force would push the joint to move along the kinematic DOF
directions. If, however, a force or moment is applied along a constraint feature DOF direction, the force would be transmitted through the joint in the feature DOF directions. The joint is unable to kinematically react to the force or moment in the constrained directions and therefore must transmit the force through the joint. The constrained DOFs would respond to a force by creating a reaction force, preventing motion.

For example, if forces and moments were applied to all of the directions in the cylindrical slider, shown in Figure 3.15, translations would develop in the \( t_x \) and \( t_y \) directions, with rotations occurring in \( \theta_x \) and \( \theta_y \) directions. A force would be transmitted in the \( t_z \) direction and a moment imparted in the \( \theta_y \) direction. A similar determination can be established for all the joint types.

Combining the two parts of the relationship requires a unique interpretation of the information in the \([F]\) matrix. This matrix shows the directions through which geometric feature variation can propagate. These directions are also the same directions through which forces can be transmitted. Thus, the variation-force relationship shows that geometric feature variation as well as forces and moments can be transmitted along the same, constrained joint DOFs. Because the \([F]\) matrix contains the geometric variation DOF information, it also contains the information necessary to produce wrench matrices for use in overconstraint screw analysis. Therefore, assembly wrench matrices can be formed using the tolerance analysis \([F]\) matrix.

3.2.2.2 Variation Matrix Wrench Formation

From the variation-force relationship developed in 3.2.2.1, it is possible to assemble wrench matrices using the values contained in the \([F]\) matrix of a tolerance analysis. As shown in section 2.3, the \([F]\) matrix contains the first order, partial derivatives of the assembly equation with respect to the geometric feature variables. Equation (3.23) shows the order of each column of \([F]\) where \( \alpha_i \) represents the \( i \)th geometric feature variable.
\[
\{ F_i \} = \left\{ \frac{\partial H}{\partial \alpha_i}, \frac{\partial H}{\partial \alpha_i}, \frac{\partial H}{\partial \alpha_i}, \frac{\partial H}{\partial \alpha_i}, \frac{\partial H}{\partial \alpha_i} \right\}^T
\]  
(3.23)

Applying the variation-force relationship, it is possible to interpret each of these derivatives as if they can transmit forces and moments. There are no actual forces or moments contained in the \([F]\) matrix, merely directions along which forces and moments can be transmitted. Each column of the \([F]\) then has the configuration shown in (3.24)

\[
\{ F_i \} = \{ f_x, f_y, f_z, m_x, m_y, m_z \}^T
\]  
(3.24)

The configuration of columns in \([F]\) now resembles the form of a wrench matrix as shown in equation (2.6). The two equations can be compared with equation (3.25) representing the \([F]\) wrench and equation (2.6), reprinted as (3.26).

\[
\{ W_{Fi} \} = \{ f_x, f_y, f_z, m_x, m_y, m_z \}^T
\]  
(3.25)

\[
W = \begin{bmatrix} f_x & f_y & f_z & m_x & m_y & m_z \end{bmatrix}
\]  
(3.26)

Each geometric feature variable accounts for a column in \([F]\), and therefore each produces a wrench matrix, subsequently called column-wrenches. Column-wrenches do not contain forces or moments, but like the column-twists, contain the DOF directions along which forces and moments can propagate. The column-wrenches of \([F]\) can be analyzed for overconstrained DOFs using the algorithms for wrenches found in [Adams 1998] and [Konkar 1993].

The \([F]\) matrix of the one-way clutch was analyzed to solve for any overconstraints in the geometric feature variables. The tolerance feature sensitivities for the one-way clutch are based on the geometric tolerance callouts applied to the assembly and shown in Figure 3.16. Each geometric tolerance is also labeled as \( \alpha \), or the geometric feature variable designation.
The partial derivatives of the assembly vector loop equation, $H$, with respect to the geometric feature variables, are calculated and the values stored in the $[F]$ matrix. The $[F]$ matrix for the one-way clutch is shown in equation (3.27). The columns of $[F]$ can be treated as wrenches after they are transposed. The column-wrenches for the feature variables are found in equations (3.28), (3.29), (3.30), and (3.31). The inclusion of two geometric tolerance callouts at the same contact will often yield redundant columns in $[F]$. Repetitive columns, as shown in equations (3.28) and (3.29), will give the same column-wrench.

Thus, columns 1 and 2, and columns 3 and 4 are identical. Only one of each are needed for VCAA. However, columns 5 and 6 are not alike, because the geometric feature callout is a position tolerance, which produces a constraint in both the $X$ and $Y$ directions.
\[
[F] = \begin{bmatrix}
\Delta H_x & \Delta H_x & \Delta H_x & \Delta H_x & \Delta H_x & \Delta H_x \\
\Delta H_y & \Delta H_y & \Delta H_y & \Delta H_y & \Delta H_y & \Delta H_y \\
\Delta H_z & \Delta H_z & \Delta H_z & \Delta H_z & \Delta H_z & \Delta H_z \\
\Delta H_{\theta_1} & \Delta H_{\theta_1} & \Delta H_{\theta_1} & \Delta H_{\theta_1} & \Delta H_{\theta_1} & \Delta H_{\theta_1} \\
\Delta H_{\theta_2} & \Delta H_{\theta_2} & \Delta H_{\theta_2} & \Delta H_{\theta_2} & \Delta H_{\theta_2} & \Delta H_{\theta_2} \\
\Delta H_{\theta_3} & \Delta H_{\theta_3} & \Delta H_{\theta_3} & \Delta H_{\theta_3} & \Delta H_{\theta_3} & \Delta H_{\theta_3} \\
\Delta H_{\theta_4} & \Delta H_{\theta_4} & \Delta H_{\theta_4} & \Delta H_{\theta_4} & \Delta H_{\theta_4} & \Delta H_{\theta_4} \\
\Delta H_{\phi_1} & \Delta H_{\phi_1} & \Delta H_{\phi_1} & \Delta H_{\phi_1} & \Delta H_{\phi_1} & \Delta H_{\phi_1} \\
\Delta H_{\phi_2} & \Delta H_{\phi_2} & \Delta H_{\phi_2} & \Delta H_{\phi_2} & \Delta H_{\phi_2} & \Delta H_{\phi_2} \\
\Delta H_{\phi_3} & \Delta H_{\phi_3} & \Delta H_{\phi_3} & \Delta H_{\phi_3} & \Delta H_{\phi_3} & \Delta H_{\phi_3} \\
\Delta H_{\phi_4} & \Delta H_{\phi_4} & \Delta H_{\phi_4} & \Delta H_{\phi_4} & \Delta H_{\phi_4} & \Delta H_{\phi_4} \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
0 & 0 & 0.12219 & 0.12219 & 0 & -1 \\
1 & 1 & 0.99251 & 0.99251 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
W_{a_1} = W_{a_2} = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
W_{a_3} = W_{a_4} = \begin{bmatrix}
0.12219 & 0.99251 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
W_{a_5} = \begin{bmatrix}
0 & -1 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
W_{a_6} = \begin{bmatrix}
-1 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

The interpretation of these wrenches also provides results which correspond to the constraints of the clutch assembly. Each column-wrench shows different constraint information based on the geometric feature DOFs. The first two wrenches, \( W_{a_1} \) and \( W_{a_2} \), show that the \( t_r \) direction is a constrained DOF. The roller and hub contact will transmit a force in that direction. The second two wrenches, \( W_{a_3} \) and \( W_{a_4} \), show a constraint in the
t_x and t_y directions at the contact point of the roller and the ring. This constraint exists at
an angle defined by the two directions, each direction having a component of the
constraint. Using the inverse cosine function, an angle of 7.017° (the pressure angle φ_h)
is found between the constraints. The final two wrenches, \( W_{d2} \) and \( W_{d6} \), show the
constraints of the hub. The position tolerance callout yields two columns in the [F]
matrix. These column-wrenches say that forces can be transmitted in the t_x and t_y
directions of the hub.

All of this constraint information originated from the tolerance analysis of the
assembly based on geometric feature variation. Interpreting the columns of the [F]
matrix as wrenches unlocks information on the constrained DOFs and extends the VCAA
method into finding overconstraints for the parts within assemblies.

3.2.2.3 Variation Wrench Formation for Assemblies

Like the joint twists that were presented in 3.2.1.3, the common joint types found
in Table 3.1 need wrench representations so that the tolerance models of assemblies can
be examined for overconstraints. As with the joint twists, the local joint axes and
equations shown in the following joint wrench figures are for use in the methods
presented in [Adams 1998]. The VCAA method only requires the type of wrench (e.g.
one force and two moments, etc.) in order to solve for overconstraints. Also, the DLM
tolerance analysis yields global results, making the local joint axes unessential. These
wrench representations arelisted for both the 2-D and the 3-D joints with the abbreviated
wrench notation. The force triplet is identified with \( f \) and the moment triplet with \( m \).

In 2-D, the joint wrench matrices are presented in groups based on the number of
constrained DOFs, which is the number of mobile DOFs subtracted from three. Also
required are three additional DOF wrenches to keep the model constrained to stay in 2-D.
These wrenches, shown in the figures as the \( W_{2d} \) wrenchmatrix, are necessary to
constrain the joints in the 2-D plane. The 2-D rigid joint, in Figure 3.17, has a full
wrench because it is constrained in all three DOFs.

80
Figure 3.17 - Joint Wrench with 3 Constrained DOFs in 2-D

The 2-D joints with two constrained DOFs, shown in Figure 3.18, include the planar joint, the revolute joint, and the parallel cylinders joints. The constrained DOFs in the parallel cylinders are due to line contact between the two cylinders that is constrained to be maintained.

Figure 3.18 - Joint Wrench with 2 Constrained DOFs in 2-D
The final set of 2-D joint wrenches are those that have one constrained DOF. These are the cylinder slider and the edge slider, which are shown in Figure 3.19. The force triplets for these final 2-D joints are calculated in the same manner as done for the revolute and planar joints.

\[
W = \begin{bmatrix}
\hat{f}_y \\
W_{2D}
\end{bmatrix}
\]

Figure 3.19 - Joint Wrench with 1 Constrained DOF in 2-D

For 3-D, the joint wrench matrices are presented in groups based on the number of constrained DOFs, which is the number of mobile DOFs subtracted from six. The locations of the constrained DOFs are with respect to the local coordinate systems. The 3-D rigid joint, in Figure 3.20, has a full wrench because it is constrained in all six DOFs.

\[
W = \begin{bmatrix}
0 & \bar{m}_x \\
0 & \bar{m}_y \\
0 & \bar{m}_z \\
\bar{f}_x & 0 \\
\bar{f}_y & 0 \\
\bar{f}_z & 0
\end{bmatrix}
\]

\[
\bar{m}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

\[
\bar{f}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

Figure 3.20 - Joint Wrench with 6 Constrained DOFs in 3-D

The five constrained DOF joints, in Figure 3.21, are the prismatic and revolute joint. All of the force and moment triplets for the rest of the joints are calculated in the same manner as done for these two joints.
The four constrained DOF joints, shown in Figure 3.22, are the parallel cylinders and the cylindrical joints. The constraints on the parallel cylinder must maintain a line contact between the two cylinders. This line contact will transmit any forces or moments through the axis allowing for all four constrained DOFs. The first instance of the parallel cylinder joint in Figure 3.22, the smaller cylinder within the larger one, provides an easier model with which to observe these constraints.
Figure 3.22 - Joint Wrenches with 4 Constrained DOFs in 3-D

The three constrained DOF set, found in Figure 3.23, is comprised of the spherical joint and the planar joint.

Figure 3.23 - Joint Wrenches with 3 Constrained DOFs in 3-D

The next set of joints involve those that have two constrained DOFs, which include the edge slider, as well as the cylindrical slider. These joints, as can be seen in Figure 3.24, have the same wrench representation.
The final group of joint wrenches are those that are only constrained in one DOF. The three joints are the point slider, the spherical slider, and the crossed cylinder. These joints and their wrench representations are shown in Figure 3.25.

Each set of joint wrenches can be used to analyze constraints of parts in assemblies. In [Adams 1998], the method for analyzing overconstraints dealt with using the twist representations of the joints to find their wrenches. The VCAA method looks at the needed wrench representation of each joint type and assembles the associated wrenches by extracting the information directly from the $[F]$ matrix. As stated above, the equations associated with the joint wrench in the accompanying figures are only for use in the method found in [Adams 1998]. The VCAA method only requires the types of joint constraints (e.g. rotation or translation), which can be determined through inspection of the joint wrenches.
The columns of $[F]$ are wrenches of force and moment which are based on the geometric feature variables of the assembly. These column-wrenches alone do not describe the wrenches of the different joint types of an assembly. Using the joint wrench types listed in Figures 3.17-3.25, it is possible to form the wrench matrices for each joint in the assembly. This is done by examining each joint in the assembly and using the column-wrenches of $[F]$ to create the wrench. The joint wrench is formed as a union of the columns of the geometric feature variables that are associated with the joint. Thus, each joint wrench is just a combination of the columns of $[F]$. 

86
3.2.2.4 Application to the One-way Clutch Example

In the one-way clutch model, the wrenches for each joint type can be assembled using the columns of the tolerance analysis $[F]$ matrix. The same joints that were analyzed for underconstraints can be analyzed for overconstraints. According to Figure 3.18, the parallel cylinder joint has two rows in its wrench, along with the $W_{2D}$ wrench. Examining the geometric variables that are associated with the joint and extracting the related $[F]$ columns, it is possible to form the joint wrench. Wrench $W_{ad}$ contains the information for the constrained DOF between the roller and the ring. Because it is the geometric feature variable associated with the joint, it is included in the joint wrench. The other three rows of the joint wrench are the unit constraints in $\theta_x$, $\theta_y$, and $t_z$ contained in the 2-D constraint wrench, $W_{2D}$. The resulting wrench is found in equation (3.32).

$$W_{parcyl} = \begin{bmatrix} W_{a1} \\ W_{2D} \end{bmatrix} = \begin{bmatrix} 0.12219 & 0.99251 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$ (3.32)

The cylindrical slider joint, located at the contact of the roller and the hub, is associated with wrench $W_{ad}$ and the geometric feature between the contact. The remaining DOFs of the joint wrench will be due to the 2-D constraint wrench matrix, $W_{2D}$. The complete joint wrench is found in equation (3.33)

$$W_{cyslider} = \begin{bmatrix} W_{a1} \\ W_{2D} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$ (3.33)

The final joint, the revolute, has five constrained DOFs and the wrenches of $W_{ad}$ and $W_{e5}$ will be two of the rows, because it requires two rows to represent a position tolerance.
Again, the others will be filled by the constraints which keep the joint in the 2-D model; they are $\theta_a$, $\theta_y$ and $t_z$. The revolute joint wrench can be seen in equation (3.34).

$$
\mathbf{W}_{\text{revolute}} = \begin{bmatrix}
\mathbf{W}_{a_s} \\
\mathbf{W}_{a_b} \\
\mathbf{W}_{2D}
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0
\end{bmatrix}
$$

(3.34)

Solving for the overconstraints is based upon the wrenches derived for the assembly. As stated in [Adams and Whitney 2001], the overconstraints will be discovered as an intersection of the full or sub-sets of the wrench matrices. By solving for the intersection, or the resultant wrench matrix, it is possible to identify any overconstraints in the assembly. This is accomplished by following the steps that will be outlined in Chapter 4. Analysis of the entire set yields the wrench matrix shown in equation (3.35), which denotes that the parts in this assembly are constrained to stay in the 2-D plane, the $\theta_a$, $\theta_y$, and $t_z$ directions, in which they were modeled.

$$
\mathbf{W}_{\text{Resultant-Full-Set}} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
$$

(3.35)

Analyzing the subsets looks at the constraints of each part to see if they are overconstrained. The subset solutions for each part in the clutch example are shown in equation (3.36).
\[
W_{\text{Resultant-Roller}} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0.1228 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\]

\[
W_{\text{Resultant-Ring}} = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\]

The interpretation of (3.36) differs for each part within the assembly. As with the twist analysis, the wrench analysis looks at the DOFs of the mating joints for each part. If a row exists in the resultant wrench, it shows where there is a DOF that is redundantly constrained by the mating joints on the part. Equation (3.36) shows the redundantly constrained DOFs of each part. The wrenchmatrix \( W_{\text{Resultant-Roller}} \) shows that the roller is only constrained in \( \theta_x, \theta_y, \) and \( t_z \), or that it is constrained to stay in the 2-D plane. The resultant wrenchmatrix for the ring shows the same 2-D constraint as well as a redundancy in the \( t_x \) and \( t_y \) directions. This is saying that those two DOFs were redundantly removed by two mating joint conditions. A similar result shows up in the resultant wrenchmatrix of the hub part. There is an overconstraint in the \( t_y \) direction due to the two mating joints. These redundant constraints are conservative cautions and may not signify a danger to the assembly. In depth, step-by-step overconstraint analysis of different parts and assemblies, as well as the methodology and interpretation of the results, will be shown in Chapter 4.

3.2.2.5 Conclusions for Overconstrained DOF Analysis using VCAA

The VCAA method uses the information gathered from a variation analysis and solves for the overconstrained DOFs within an assembly. This is achieved by using the
variation-force relationship of geometric feature variation and the directions of propagation. This relationship shows that the variation analysis [F] matrix gives the directions through which forces and moments can be transmitted in the assembly; this can occur in the same directions as geometric feature variation. This relationship thus shows that wrench matrices and constraint information exists within the columns of the [F] matrix.

To make the [F] columns correspond to the wrench matrix configuration, they must simply be transposed. The [F] matrix geometric feature variable column-wrenches can be used to generate the joint wrench matrices for the assembly kinematic joints. Each joint type has an associated wrench representation, which shows what DOFs are constrained not to move. The column-wrenches from [F] and other model constraint wrenches make up the joint wrenches. This is done by noticing which geometric feature variables are associated with the kinematic joint and then using the corresponding column-wrenches to fill in the rows of the joint wrenches. Then, wrench analysis methods based on the work in [Adams 1998] can be used to solve for assembly overconstraints.

3.3 Conclusion of Constraint Analysis using Tolerance Methods

The Variation-based Constraint Analysis for Assemblies (VCAA) method links the two fields of constraint analysis and tolerance analysis. Within the variation sensitivity matrices solved for in the DLM is the information needed to analyze mobile and immobile part DOFs within assemblies. A tolerance analysis solution for the effects of variation can also be used to solve for DOF constraint problems.

The link between constraints and variation is based on a connection between variation and screw theory. The twist and wrench screw matrices, which are utilized to solve for under- and overconstraints, can be formed from the DLM variation matrices. The TAKS variation-velocity analogy, found in [Faerber 1999], shows that the [B] matrix
contains the information needed to form twists. An analogous variation-force relationship shows that the $[F]$ matrix contains the required information to form wrenches. Methods based on screw theory analysis done in [Konkar 1993] and [Adams 1998] will yield under-and overconstraint information of the parts within an assembly from the twist and wrench matrices. Therefore, it is possible to perform a tolerance variation analysis on an assembly by the DLM and at the same time, with only minimal calculation, also perform a part constraint analysis using the variation analysis data.
CHAPTER 4. VCAA EXAMPLES FOR 2-D AND 3-D ASSEMBLIES

Using the VCAA method, it is possible to analyze parts of an assembly for under- and overconstraints using a DLM tolerance analysis. Using the information gained from a tolerance analysis and following screw theory methodology, a constraint analysis of parts in an assembly is straightforward. Through a series of examples, the requirements, methodology, steps of analysis, and the interpretation of the results are presented in this chapter. These case studies will include analyses of 2-D and 3-D assemblies which contain various instances of under-, over-, and exactly constrained parts.

4.1 Variation-based Constraint Analysis of Assemblies Methodology

To employ tolerance analysis methods to solve for constraints using the VCAA, certain modeling requirements must be followed. If the modeling requirements are met and the variation analysis is performed correctly, then a series of steps may be followed to correctly analyze the parts in the assemblies for under- and overconstraints. The correct interpretation of the results is the final phase of analysis.

4.1.1 VCAA Modeling Requirements

To analyze parts of an assembly using VCAA, certain modeling requirements must be followed. Setting up the DLM vector loops calls for following exact guidelines to account for dimensional, kinematic, and geometric variation. In [Chase 1999], tolerance vector loop rules and guidelines are presented. Following these rules will ensure that the assembly will be modeled correctly both for variations and for use in VCAA.
Following the DLM modeling rules found in [Chase 1999] will yield the necessary variation information. If the loops are assembled appropriately and the GCM is employed correctly, all of the dependent kinematic variables will be represented in the [B] matrix. This matrix will contain all of the needed information to solve for underconstrained. When the DLM rules are adhered to, there are no other necessary requirements on the model to solve for underconstraints.

For the geometric feature variation [F] matrix, there must be geometric tolerance callouts on the parts in the assembly. For the VCAA method to correctly solve for overconstrained, the [F] matrix must contain geometric feature variables for each kinematic joint in the assembly. Therefore, geometric tolerances must be indicated for each and every kinematic joint. Assigning a geometric tolerance to every joint is not a standard modeling rule, as certain geometric variation will not add significant tolerance stack-up to the model. However, to solve for the overconstrained parts in an assembly requires that each joint has a geometric tolerance callout. It is only necessary to assign one geometric tolerance to each joint. More than one geometric tolerance at each joint will simply result in redundant columns in [F] and will not contribute further to the VCAA method of detecting overconstrained.

In summary, to employ the VCAA method simply requires following the DLM rules, found in [Chase 1999], for setting up and analyzing variation of assemblies using vector loops. Nothing additional needs to be added to the tolerance analysis in solving for underconstrained, as the [B] matrix is complete when the DLM rules are followed. However, finding overconstrained requires that geometric variation sources be applied to each joint so that the [F] matrix contains constraint information for each joint. After the geometric sources are applied to each joint, the DLM vector loops are solved normally, simply including the geometric variation.
4.1.2 VCAA Methodology and Analysis Steps

When an assembly has been correctly modeled using the DLM and geometric tolerances have been placed at each joint, the sensitivity matrices can be calculated using the GCM. The values stored in \([B]\) and in \([F]\) now contain all the information needed to solve for constraint information using the VCAA method. By following the VCAA analysis steps, it is possible to identify the constraints of parts in many different assemblies.

Before the steps of VCAA are outlined, it is important to review the necessary screw-theory based algorithms first introduced in section 2.2.3. The most essential operation necessary to solve for both twists and wrenches is the reciprocal operation. As seen in equations (2.8) and (2.9), the reciprocal operation is necessary for calculating twists from wrenches or vice versa. The mathematical meaning of the reciprocal screw is not reviewed here, but can be found in detail in [Konkar and Cutkosky 1995]. However, the mathematical operations to produce the reciprocal screw are simple and can be explained easily. The reciprocal operation, and the other needed functions, are shown in the MATLAB® files found in Appendix C.

As seen in equation (2.10), here reprinted as (4.1), the mathematical connection between the twistspace and the wrenchspace is the null space. Twists and wrenches are mutually exclusive, and due to their mathematical relationship, one can be solved for if the other is known. This can be accomplished through the linear algebra null space operation. After this is done, the transpose of the resulting matrix is taken and the first three columns are switched with the last three. These three mathematical operations produce the reciprocal screw. The point algorithm, also found in [Adams 1998], is used to interpret twists that contain rotations. It involves solving for the location of a point on the axis of rotation. This is used to help determine where the underconstrained rotations occur.
\[ [T][W] = \{0\} \] (4.1)

The steps of the VCAA method are simple and involve the screw theory operations, summarized in [Adams 1998] and [Adams and Whitney 2001], as well as simple linear algebra manipulation. The VCAA methodology steps for underconstraints and overconstraints are similar, but will be explained separately, as they use the same screw theory, but different sensitivity matrices.

4.1.2.1 Underconstraint Analysis Steps

The first step in an underconstraint analysis begins by solving for the [B] matrix through a DLM analysis of the assembly, using the GCM to find the sensitivities. All of the modeling rules found in [Chase 1999] must be followed for each vector loop. If the assembly was modeled in 2-D, [B] still must be represented in its 3-D form. If the DLM analysis yields a large, multiple loop [B] matrix, then this large matrix must be broken up into its individual sub-matrices, where each loop is represented by its own [B] matrix. When the first step is completed, there will be a [B] matrix for each loop that is \(6 \times n\) in size (where \(n\) is the number of dependent variables in the matrix).

Step two involves forming the column twist matrix for the dependent variables in [B]. This is accomplished by first transposing the [B] matrix. The resulting matrix then has its first three columns switched with its last three, so that the final matrix, \(T_{\text{column}}\), has a twist form of \(n \times 6\) in size. Each row of \(T_{\text{column}}\) is a column twist (alluding to their [B] matrix column origin), which represents the allowable motions for each dependent variable.

Step three takes the column twists of the \(T_{\text{column}}\) matrix and forms joint twists. Each kinematic joint in the assembly has mobile DOFs. Using the joint twist representations, shown in Figures 3.6-3.14, it is possible to identify which DOFs (e.g. rotation or translation) are needed to form the joint twist matrix. After the types of

96
mobile DOFs are identified, each joint twist is formed as a union of the column twists of its associated dependent variables, which allow the needed mobile DOFs. These joint twists, $T_{\text{joint}}$ for each joint $i$, describe the motions allowed by each joint.

Step four of the underconstraint VCAA method takes each $T_{\text{joint}}$ matrix found in step three and finds their respective intermediate joint wrenches. This is done by finding the reciprocal of each $T_{\text{joint}}$ and forming $W_{\text{intermediate-joint}}$ matrices for each joint.

Step five involves forming an intermediate part wrench, $W_{\text{intermediate-part}}$, for each part $j$, in the assembly. These part wrenches are formed through a union of the different $W_{\text{intermediate-joint}}$ matrices associated with the part. For example, forming $W_{\text{intermediate-part3}}$ for the theoretical part3 may involve the union of the two intermediate joint matrices, $W_{\text{intermediate-joint1}}$ and $W_{\text{intermediate-joint4}}$. Each intermediate part wrench will simply be a combination of its associated intermediate joint wrenches.

The final step for underconstraint analysis, step six, finds the resultant twist of each part by taking each intermediate part wrench and, part-by-part, performing the reciprocal operation. This will yield a $T_{\text{Resultant-part}}$ for each part. Each of these resultant twist matrices will show what motions are still allowed on the part due to the joints that constrain it. The interpretation of each of these resultant twists will be described in section 4.1.3. Figure 4.1 shows the VCAA method underconstraint analysis steps in a flowchart.

4.1.2.2 Overconstraint Analysis Steps

Overconstraint analysis methodology in the VCAA method follows very similar steps as that of solving for underconstraints. Step one is solving for the $[F]$ matrix through a DLM analysis of the assembly including geometric tolerance callouts at each joint. Again, the modeling rules found in [Chase 1999] must be followed for each vector loop. As with the $[B]$ matrix, the $[F]$ matrix must be 3-D and each loop must be represented by its own individual $[F]$ matrix. When this step is complete, there will be a
[F] matrix for each loop that is also 6 × n in size (where n is the number of geometric feature variables in the matrix).

![Diagram showing the VCAA Underconstraint Methodology](image)

**Figure 4.1 - VCAA Underconstraint Methodology**

Step two takes the [F] matrix and forms the column wrench matrix. Transposing [F] is the only necessary operation to make the \( W_{\text{column}} \) matrix, which will be \( n \times 6 \) in size. The n rows of \( W_{\text{column}} \) are the column wrenches (referring to their origin as columns in [F]), which represent the constrained DOFs for each geometric feature variable.

Step three involves forming the joint wrenches from the \( W_{\text{column}} \) matrix. Each kinematic joint has constrained DOFs. Using the joint wrench representations, found in
Figures 3.17-3.25, it is possible to associate which constrained DOFs (e.g. translation or rotation) are needed to constitute the joint wrench matrix. After the types of constrained DOFs are found, each joint wrench is formed as a union of the column wrenches of its associated geometric feature variables, which constrain the joint DOFs. These joint wrenches, \( W_{\text{joint}_i} \) for each joint \( i \), represent the constrained DOFs of each joint.

Step four of the overconstraint VCAA method consists of finding the intermediate twist matrices for each joint wrench formed in step three. This is accomplished by applying the reciprocal operation to each \( W_{\text{joint}_i} \) and creating \( T_{\text{intermediate-joint}_i} \) matrices, for each joint.

Step five requires a review of wrench theory. The work [Adams and Whitney 2001] states that the full set and all subsequent sub-sets of wrenches must be analyzed to determine if there are any overconstraints. The full set of wrenches involves including all the joint wrenches for the entire assembly. Logical subsets to be analyzed include the joint wrenches of the parts as well as the joint wrenches of the vector loops. Step five involves taking the intermediate twists and through the union operation, forming the intermediate twists for the different subsets. This can be done for each part \( j \), \( T_{\text{intermediate-part}_j} \), or for each loop \( k \), \( T_{\text{intermediate-loop}_k} \). These part or loop wrenches are the union of the associated \( T_{\text{intermediate-joint}_j} \) matrices. For example, forming \( T_{\text{intermediate-part}_2} \) for the theoretical part2 may encompass the union of the two intermediate joint twist matrices of \( T_{\text{intermediate-joint}_3} \) and \( T_{\text{intermediate-joint}_7} \). Another example would be forming the theoretical \( T_{\text{intermediate-loop}_2} \) matrix from the associated intermediate matrices.

Finally, step six takes the intermediate part or loop wrench, found in step five, and solves the resultant wrench matrix of the part or loop. This is accomplished by applying the reciprocal operation to the intermediate wrenches to yield a \( W_{\text{Resultant-part}_j} \) for each part or a \( W_{\text{Resultant-loop}_k} \) for each loop. Each of these resultant wrench matrices will show what DOFs of the subset are overconstrained due to the joints constraining it. As with the resultant twist matrices, the interpretation of each of these resultant wrenches will be
described in section 4.1.3. Figure 4.2 shows the VCAA method overconstraint analysis steps in flowchart format.

![Diagram of VCAA Overconstraint Methodology]

These systematic steps, for both underconstraints and overconstraints, make it possible to solve for constraint information directly from the tolerance analysis variation sensitivity matrices. The analysis is relatively simple and only involves linear algebra and screw theory operations.

4.1.3 VCAA Interpretation of Results

The interpretation of the resultant twist- and wrenchmatrices identifies
underconstrained and overconstrained DOFs. This is done using the methods described in [Adams 1998] as well as with additional screw theory tools. As with the methodology steps outlined in section 4.2.1, underconstraint and overconstraint interpretations merit separate treatments.

As stated in [Adams 1998], if the underconstraint resultant twistmatrix of a part is non-zero, mobile DOFs exists on that part. The interpretation of these mobile DOFs is dependent upon the information contained in the rows of the resultant twist. Each row indicates a different mobile DOF condition. By analyzing the triplets contained in each row, it is possible to identify what DOF is underconstrained and, in some cases, at what position this underconstraint occurs.

Twistmatrices contain two triplets, which describe the allowed relative motions, both rotational and translational, as seen in equation (2.11), reprinted here as (4.2). The first triplet, represented by the angular velocity vector $\omega$, shows the allowed rotational DOFs. The second triplet, depicted as the linear velocity vector $v$, shows the allowed translational DOFs. The interpretation of resultant twists involves looking at each triplet one at a time.

$$\mathbf{T}_{\text{Resultant}} = \begin{bmatrix} \omega_{1x} & \omega_{1y} & \omega_{1z} & v_{1x} & v_{1y} & v_{1z} \\ \omega_{2x} & \omega_{2y} & \omega_{2z} & v_{2x} & v_{2y} & v_{2z} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{4.2}$$

If the $\omega$ triplet is non-zero, then a rotational DOF is still allowed on the part. The location of the non-zero entries indicates the axis of rotation. For example, if there was a non-zero value in the $\omega_z$ location, that would indicated a rotation about the global Z-axis. If there were non-zero values in both the $\omega_y$ and $\omega_z$ locations, that would indicate that the part had a dual rotation around both the global Y- and Z-axes. If a rotation exists, the second $v$ triplet contains information about the location of the rotation. The point algorithm found in [Adams 1998] will establish a point along the axis of rotation from the
information contained in the second triplet.

When the \( \omega \) triplet is empty, the \( \nu \) triplet shows the mobile translational DOFs. The location of the non-zero entries also indicates the axis of translation. For example, a non-zero value in the \( \nu_x \) location would indicate a translation in the global X-axis direction. If there were non-zero values in both the \( \nu_x \) and \( \nu_y \) locations of the triplet, that would mean that there is a joint translation of the part allowed along the X- and Y-axes. Translational twists do not contain any further information on position of the mobile DOF.

The overconstraint resultant wrenchmatrix shows the constrained DOFs of parts or vector loops when the matrix is non-zero. Each row of the resultant wrench indicates a different constrained DOF condition. By analyzing the entries of the rows within the resultant wrenchmatrix, it is possible to identify what DOF is redundantly constrained.

Wrenchmatrices also contain two triplets, which describe the constrained DOFs, as seen in equation (2.12), reprinted as (4.3). The first triplet, depicted by the force vector \( f \), shows the constrained translational DOFs. The second triplet, represented by the moment vector \( m \), shows the constrained rotational DOFs. The interpretation of resultant wrenches simply requires looking at the non-zero entries.

\[
W_{\text{Resultant}} = \begin{bmatrix}
  f_{1x} & f_{1y} & f_{1z} & m_{1x} & m_{1y} & m_{1z} \\
  f_{2x} & f_{2y} & f_{2z} & m_{2x} & m_{2y} & m_{2z} \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{bmatrix}
\]  

(4.3)

The location of any non-zero entry reveals the location of the overconstrained DOF. For example, a non-zero entry in the \( f_y \) location would signify a translational overconstraint in the global Y-axis direction. A combination of non-zero values in \( f_z \) and \( m_z \) would mean that a translational overconstraint exists in the global Y-direction and a rotational overconstraint exists around the global Z-axis. There is no information on the
position of these constraints and there is no extra information contained between the two triplets, as with the twists.

The unique thing about the resultant wrench matrix is the meaning of the overconstraints. As stated in [Adams and Whitney 2001], the operations listed in Figure 4.2 will identify redundant DOFs on subsets of the assembly (e.g. parts or loops). This means that an overconstraint on the part will show up if the same DOF is constrained by two independent mating joints. Thus, the interpretation that results from wrenchmatrices is a conservative one, in that it will identify some redundant DOFs as overconstraints that may not in any way disturb the assembly. It will also identify those overconstrained redundant DOFs that can lead to problems with the assembly.

With these simple steps, the VCAA method can be used to identify underconstraints and overconstraints on parts within assemblies. Integration with tolerance analysis allows a simultaneous analysis of both variations and constraints. The following sections contain various examples of constraint analyses for a variety of cases.

4.2 2-D Examples of Constraint Analysis using the VCAA Method

The first set of example problems will be 2-D assemblies with an assortment of constraint cases. Each case assembly will present a different facet of the VCAA method and the steps involved in analyzing the constraints.

4.2.1 Two Cylinder Slider Plate Assembly

The first 2-D assembly that will be analyzed using the VCAA method will be a simple plate assembly consisting of two parts which are joined by two cylinder slider joints, whose nominal dimensions are shown in Table 4.1. This assembly, pictured in Figure 4.3, is first analyzed using the DLM, with the GCM solving for the variation sensitivity matrices. The assembly only has one vector loop and the 3-D forms of the $[B]$ and $[F]$ matrices for this loop are listed in the equations below. With step one completed,
the constraints will be solved, underconstraints first, followed by overconstraints, using the steps outlined in Figure 4.1 and Figure 4.2. All of the steps in this example will be shown to illustrate how to use the VCAA method.

![Diagram of Two Cylinder Slider Plate Assembly](image)

Figure 4.3 - Two Cylinder Slider Plate Assembly

<table>
<thead>
<tr>
<th>Table 4.1 - Nominal Dimensions of Two Cylinder Slider Plate Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vector-Part Name</strong></td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>$a$</td>
</tr>
<tr>
<td>$r_1$</td>
</tr>
<tr>
<td>$u_1 - \phi_1$</td>
</tr>
<tr>
<td>$u_2$</td>
</tr>
<tr>
<td>$r_3 - \phi_2$</td>
</tr>
<tr>
<td>$b$</td>
</tr>
</tbody>
</table>

$$[B] = \begin{bmatrix} u_1 & u_2 & \phi_1 & \phi_2 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & -1.5 & 5.5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$  

(4.4)
\[
[F] = \begin{bmatrix}
\alpha_1 & \alpha_2 \\
0 & 0 \\
1 & -1 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0
\end{bmatrix}
\]  
(4.5)

The second step (after performing the DLM analysis) in the underconstraint analysis involves forming the column twist matrix by transposing the \([B]\) matrix and then, in the resulting matrix, switching the first three columns with the second three. This operation produces the matrix shown in equation (4.6).

\[
\mathbf{T}_{\text{column}} = \begin{bmatrix}
u_1 & 0 & 0 & 0 & -1 & 0 & 0 \\
u_2 & 0 & 0 & 0 & 1 & 0 & 0 \\
\phi_1 & 0 & 0 & 1 & 0 & -1.5 & 0 \\
\phi_2 & 0 & 0 & -1 & 0 & 5.5 & 0
\end{bmatrix}
\]  
(4.6)

The third step takes the rows of the matrix in equation (4.6) and forms the joint twists using the joint types and the dependent variables associated with them. Both kinematic joints of this assembly are 2-D cylinder sliders, whose twist representation is shown in Figure 3.8. The twist representation shows that there should be one rotational DOF and one translational DOF in each joint twist. The first cylinder slider, called joint1, is associated with the two dependent variables \(u_1\) and \(\phi_1\), which are a rotation and a translation. Performing a union of the two column twists, \(u_1\) and \(\phi_1\), will yield the joint twist for the first cylinder slider. The second slider, called joint2, is associated with \(u_2\) and \(\phi_2\), which will also result in a joint twist with one rotation and one translation. The two joint twists are shown in equations (4.7) and (4.8).

\[
\mathbf{T}_{\text{joint1}} = \begin{bmatrix}
\mathbf{T}_{\text{column}, u_1} & \mathbf{T}_{\text{column}, \phi_1}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 & -1.5 & 0
\end{bmatrix}
\]  
(4.7)
\[
\mathbf{T}_{\text{joint2}} = \begin{bmatrix}
\mathbf{T}_{\text{column-u2}} \\
\mathbf{T}_{\text{column-φ2}}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & -1 & 0 & 5.5 & 0
\end{bmatrix}
\]  \hspace{1cm} (4.8)

The fourth step takes the joint twists found in step three and finds their intermediate joint wrench matrices by performing the reciprocal operation. The intermediate joint wrenches for the two joints in this assembly are shown in equations (4.9) and (4.10).

\[
\mathbf{W}_{\text{intermediate-joint1}} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1.5 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\]  \hspace{1cm} (4.9)

\[
\mathbf{W}_{\text{intermediate-joint2}} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 5.5 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\]  \hspace{1cm} (4.10)

Step five takes the intermediate wrenches from step four and unions them into the intermediate part wrenches for each part. Both parts in this assembly have the same mating joints, so the analysis for one part will be the same as the analysis of the other. Looking at the top plate, labeled part1, it is possible to find its intermediate part wrench through a union of the intermediate joint wrenches for joint1 and joint2. This matrix is shown in equation (4.11).
\[
\mathbf{W}_{\text{Intermediate-part}} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1.5 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 5.5 \\
0 & 0 & 1 & 0 & 0 & 0 \\
\end{bmatrix}
\]  

(4.11)

Step six takes the part wrenches and solves for the resultant part twist matrices for each part. The resultant twist for both parts will be the reciprocal of equation (4.11). The resultant twist, equation (4.12) will show the motions allowed on the part by the mating joints.

\[
\mathbf{T}_{\text{Resultant-part}} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

(4.12)

The resultant twist for the parts show that there is an underconstraint in the parts and motion is still allowed in the global X-direction, which can be observed by inspection of the assembly. Information gathered from the DLM variation analysis yielded the necessary underconstraint information for the assembly.

The VCAA for overconstraints begins by looking at the [\(\mathbf{F}\)] matrix from the DLM analysis. Step one was completed above and equation (4.5) contains the sensitivity matrix needed to continue. Step two forms the column wrench matrix from the [\(\mathbf{F}\)] matrix by transposing it. This wrench matrix is shown in equation (4.13).

\[
\mathbf{W}_{\text{column}} = \begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\end{bmatrix} \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

(4.13)

Step three for overconstraints forms the joint wrenches for the joints of the assembly by associating the necessary geometric feature variables and using the joint wrench representations. The 2-D cylinder slider joint wrench representation, shown in
Figure 3.19, shows that, along with the 2-D constraint wrench matrix, only one constrained translation DOF is needed. The first cylinder, joint 1, is associated with the geometric feature variable translation $a_1$ and joint 2 is associated with the translation $a_2$. Using the rows from (4.13) and the joint wrench representation, the joint wrenches for both joints, equations (4.14) and (4.15) can be formed.

\[
W_{\text{joint1}} = 
\begin{bmatrix}
W_{\text{column} - a_1} \\
W_{2D}
\end{bmatrix} = 
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\] 

(4.14)

\[
W_{\text{joint2}} = 
\begin{bmatrix}
W_{\text{column} - a_2} \\
W_{2D}
\end{bmatrix} = 
\begin{bmatrix}
0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\] 

(4.15)

Step four takes the joint wrenches and forms the intermediate joint twists by applying the reciprocal operation. This produces the matrices shown in equations (4.16) and (4.17).

\[
T_{\text{intermediate-joint1}} = 
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\] 

(4.16)

\[
T_{\text{intermediate-joint2}} = 
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\] 

(4.17)

Step five finds the intermediate subset twist matrices through a union of the twist matrices. In this example, the only subset needed will include both intermediate twists for the two parts. The intermediate part twist is shown in equation (4.18).
Step six takes the reciprocal of the intermediate twist to solve for the resultant wrench matrix of the subset, in this case either part. The resultant wrench is shown in equation (4.19).

\[ \mathbf{W}_{\text{Resultant-part}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \]  

(4.19)

This matrix shows the overconstraints of the two cylinder slider joints in the assembly. The overconstraints in the \( \theta_x \), \( \theta_y \), and \( t_z \) show that the model is constrained to stay in the 2-D plane, which correlates to the modeling rules placed on the assembly. The row that shows the overconstraint in \( t_z \) is due to an overconstraint placed between the two joints by the cylinder sliders. In 3-D it is easier to see what the analysis is saying. The extruded 3-D model of the assembly, shown in Figure 4.4, shows two lines of contact connecting the bottom plate to the top plate. Modeling rules used in [Blanding 1999] show that a line of contact is determined by two points of contact and that a plane is determined by three non-collinear points or by one line and one point. The top plate of the extruded model is overconstrained because it is being determined not by one line and one point, but by two lines.
The VCAA method combines the variation analysis performed by the DLM and screw theory based analysis to solve for the constraints of the assembly. In the plate assembly, even though the tolerance analysis was solved in 2-D, the 3-D constraints were identified. The underconstraint in \( t_x \) and the overconstraint in \( t_y \) both showed up as a result of the VCAA.

4.2.2 Three Cylinder Slider Plate Assembly

The second 2-D model, shown in Figure 4.3, is a variation of the assembly in section 4.2.1. This assembly, whose dimensions are shown in Table 4.2, has two vector loops and an additional third cylinder slider on the bottom plate. The purpose of including the analysis of this assembly is to show how to treat multiple loops and 2-D linear overconstraints. The DLM tolerance analysis, using the GCM, yields the sensitivity matrices \([B]\) and \([F]\) for each loop, shown in equations (4.20) through (4.23). This example will not show the results from every step, but it will show the most important steps.
Table 4.2 - Nominal Dimensions of Three Cylinder Slider Plate Assembly

<table>
<thead>
<tr>
<th>Vector-Part Name</th>
<th>Orientation</th>
<th>Joint Coordinates (in cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$\theta = 0^\circ$</td>
<td>$X = 0.0 \quad Y = 0.0$</td>
</tr>
<tr>
<td>$r_1$</td>
<td>$\theta = 90^\circ$</td>
<td>$X = 1.5 \quad Y = 0.0$</td>
</tr>
<tr>
<td>$u_1$ - $\phi_1$</td>
<td>$\theta = 180^\circ$</td>
<td>$X = 1.5 \quad Y = 1.0$</td>
</tr>
<tr>
<td>$u_2$</td>
<td>$\theta = 0^\circ$</td>
<td>$X = 0.0 \quad Y = 1.0$</td>
</tr>
<tr>
<td>$r_2$ - $\phi_2$</td>
<td>$\theta = -90^\circ$</td>
<td>$X = 3.5 \quad Y = 1.0$</td>
</tr>
<tr>
<td>$b$</td>
<td>$\theta = -180^\circ$</td>
<td>$X = 3.5 \quad Y = 0.0$</td>
</tr>
<tr>
<td>$u_3$</td>
<td>$\theta = 0^\circ$</td>
<td>$X = 0.0 \quad Y = 1.0$</td>
</tr>
<tr>
<td>$r_3$ - $\phi_3$</td>
<td>$\theta = -90^\circ$</td>
<td>$X = 5.5 \quad Y = 1.0$</td>
</tr>
<tr>
<td>$c$</td>
<td>$\theta = -180^\circ$</td>
<td>$X = 5.5 \quad Y = 0.0$</td>
</tr>
</tbody>
</table>

\[
[B]_1 = \begin{bmatrix}
  u_1 & u_2 & \phi_1 & \phi_2 \\
  -1 & 1 & 0 & 0 \\
  0 & 0 & -1.5 & 3.5 \\
  0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 \\
  0 & 0 & 1 & -1 \\
\end{bmatrix}
\]

(4.20)
\[
[B]_2 = \begin{bmatrix}
    u_1 & u_2 & \phi_1 & \phi_3 \\
    -1 & 1 & 0 & 0 \\
    0 & 0 & -1.5 & 5.5 \\
    0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 0 \\
    0 & 0 & 1 & -1 \\
\end{bmatrix}
\]

(4.21)

\[
[F]_1 = \begin{bmatrix}
    \alpha_1 & \alpha_2 \\
    0 & 0 \\
    1 & -1 \\
    0 & 0 \\
    0 & 0 \\
    0 & 0 \\
\end{bmatrix}
\]

(4.22)

\[
[F]_2 = \begin{bmatrix}
    \alpha_1 & \alpha_3 \\
    0 & 0 \\
    1 & -1 \\
    0 & 0 \\
    0 & 0 \\
    0 & 0 \\
\end{bmatrix}
\]

(4.23)

For underconstraint analysis, the second step takes the \([B]\) matrices from both loops and forms the column twist matrices. The third step takes those column twist matrices and forms the joint twists for each joint in the assembly. The three cylinder sliders will each have two mobile DOFs. The associated dependent variables are used to form the joint twists, which are shown for the three cylinder sliders, in equations (4.24) through (4.26). The joint twists can be formed from either of the loop matrices, as long as the associated dependent variables are used.

\[
T_{\text{joint}} = \begin{bmatrix}
    T_{\text{column1-}u1} & T_{\text{column1-}u1} \\
    0 & 0 & -1 & 0 & 0 \\
    0 & 0 & 1 & 0 & 0 & 0 \\
\end{bmatrix}
\]

(4.24)
\[ \mathbf{T}_{\text{joint2}} = \begin{bmatrix} \mathbf{T}_{\text{column1-\alpha} 2} \\ \mathbf{T}_{\text{column1-\phi} 2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 3.5 & 0 \end{bmatrix} \] (4.25)

\[ \mathbf{T}_{\text{joint3}} = \begin{bmatrix} \mathbf{T}_{\text{column2-\alpha} 3} \\ \mathbf{T}_{\text{column2-\phi} 3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 5.5 & 0 \end{bmatrix} \] (4.26)

Step four takes these joint twists and using the reciprocal operation forms the intermediate joint wrenchmatrices. For step five, these intermediate joint wrenches are then used to form the intermediate part wrenchmatrix through a union operation. Both parts in this assembly will have all three joints included in the constraint analysis. Step six takes the intermediate part wrenchmatrix and forms the resultant part twist, which will show the underconstrained DOFs. The resultant part twist for either part is shown in equation (4.27). This twists shows that motion is allowed in the \( t_x \) direction, which can be observed from the model.

\[ \mathbf{T}_{\text{Resultant- Part1}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \] (4.27)

The overconstraint information is contained within the \([\mathbf{F}]\) matrices of the assembly. The second step for identifying these overconstraints, after the variation analysis, is to form the column wrenchmatrices by transposing the \([\mathbf{F}]\) matrices. The third step takes the column wrench and the joint wrench representations to form the joint wrenches for each joint in the assembly. The 2-D cylinder sliders have one constrained translation DOF as well as the 2-D constraint wrench. The joint wrenches for each joint are found in equations (4.28) through (4.30).

\[ \mathbf{W}_{\text{joint1}} = \begin{bmatrix} \mathbf{W}_{\text{column1-\alpha} 1} \\ \mathbf{W}_{2D} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \] (4.28)
\[ \mathbf{W}_{\text{joint}2} = \begin{bmatrix} \mathbf{W}_{\text{column}1-\alpha 2} \\ \mathbf{W}_{2D} \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \] (4.29)

\[ \mathbf{W}_{\text{joint}3} = \begin{bmatrix} \mathbf{W}_{\text{column}2-\alpha 3} \\ \mathbf{W}_{2D} \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \] (4.30)

Using the reciprocal operation on the joint wrenches allows for the formation of the intermediate joint twistwrenches. The next step forms the subset intermediate twists for either the loops or parts. Each subset must be analyzed to discover the overconstraints within the assembly. The subsets that yield information on the overconstraints arise from forming intermediate loop twistmatrices for both DLM vector loops. The intermediate loop twists are shown in (4.31) and (4.32) for both loops.

\[ \mathbf{T}_{\text{intermediate-loop}1} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \] (4.31)

\[ \mathbf{T}_{\text{intermediate-loop}2} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \] (4.32)

The final step takes these twists and finds the resultant loop wrenches to solve for overconstraints. By performing the reciprocal operation on the intermediate loop twists, the resultant wrenches, found in (4.33) and (4.34), are calculated.
\[ W_{\text{Resultant-loop}1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \] (4.33)

\[ W_{\text{Resultant-loop}2} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \] (4.34)

This matrix shows the overconstraints of the three cylinder slider joints in the assembly. The overconstraints found in both resultant wrenches in the \( \theta_x \), \( \theta_y \), and \( t_z \) directions show that the model is constrained to stay in the 2-D plane, which correlates to the modeling rules placed on the assembly. In both loops there is a row that shows an overconstraint in \( t_y \), which is due to an overconstraint placed between the two joints by the cylinder sliders. Because this overconstraint shows up in both resultant loop wrenches, a 2-D linear overconstraint can be inferred. The front view of the assembly, depicted in Figure 4.5, shows the three contact points from the cylinder slider joints trying to determine a line contact. Two points determine a line and the third point is an overconstraint. By looking at the loops and finding the overconstraint in \( t_y \), it is also possible to identify 2-D collinear constraint problems.

The plates in this assembly are not only overconstrained in the same way as the two cylinder slider assembly, but also in the collinear placement of the three cylinder sliders. The extruded 3-D model of the assembly, shown in Figure 4.6, shows three lines of contact connecting the bottom plate to the top plate. The top plate of the extruded model is overconstrained because it is being determined not by one line and one point, but by three lines. Also, the figure shows the 2-D overconstraint, the line of contact formed by three collinear points.
By analyzing the subsets of the joint wrenches, it is possible to analyze constraints in both 3-D and 2-D. The analysis of the plate assembly showed the underconstrained DOF of $t_w$ as well as the multiple overconstraint problems in $t_j$ which resulted from the addition of a third cylinder slider. All of the information for this analysis arose from the DLM tolerance analysis and screw theory operations.

4.2.3 Stacked Blocks Assembly

The stacked blocks assembly, pictured in Figure 4.7, is a conceptual assembly used to teach tolerance analysis methods. There are three parts in the assembly, which include a base, a sliding block, and a cylinder. The block is placed on the base so that it slides until it comes in contact with the left side wall. The cylinder is then placed to contact the top of the block and the left side wall of the base. The nominal dimensions of the independent manufactured variables and the nominal dependent assembly variable dimensions are listed in Table 4.3. Geometric tolerances are assigned to each joint and shown in Figure 4.8. To solve this assembly using the DLM requires three vector loops, shown in Figure 4.9, which pass through the part datums and joints, following the rules in [Chase 1999]. The DLM tolerance analysis yields the sensitivity matrices $[B]$, and $[F]$, for each loop $i$. They are listed in the equations below for use in VCAA.
Figure 4.7 - Stacked Blocks Assembly

Table 4.3 - Nominal Dimensions of Stacked Blocks Assembly

<table>
<thead>
<tr>
<th>Independent Variables</th>
<th>Basic Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cylinder Radius - B</td>
<td>6.620 mm</td>
</tr>
<tr>
<td>Cylinder Radius - C</td>
<td>6.620 mm</td>
</tr>
<tr>
<td>Base Step Width - F</td>
<td>3.905 mm</td>
</tr>
<tr>
<td>Base Step Height - G</td>
<td>4.060 mm</td>
</tr>
<tr>
<td>Block Thickness - I</td>
<td>6.805 mm</td>
</tr>
<tr>
<td>Base Step Location - J</td>
<td>28.125 mm</td>
</tr>
<tr>
<td>Base Step Height - K</td>
<td>10.675 mm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dependent Variables</th>
<th>Basic Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cylinder/Base Contact - A</td>
<td>18.7182 mm</td>
</tr>
<tr>
<td>Angle $\phi_1$</td>
<td>74.724°</td>
</tr>
<tr>
<td>Cylinder/Block Contact - D</td>
<td>8.6705 mm</td>
</tr>
<tr>
<td>Angle $\phi_2$</td>
<td>164.724°</td>
</tr>
<tr>
<td>Block/Ground Contact - E</td>
<td>10.0477 mm</td>
</tr>
<tr>
<td>Angle $\phi_3$</td>
<td>105.276°</td>
</tr>
<tr>
<td>Block/Ground Contact - H</td>
<td>2.1894 mm</td>
</tr>
<tr>
<td>Angle $\phi_4$</td>
<td>105.276°</td>
</tr>
<tr>
<td>Block/Ground Contact - L</td>
<td>27.2965 mm</td>
</tr>
</tbody>
</table>
Figure 4.8 - Joint types and geometric tolerances for Stacked Blocks Assembly

Figure 4.9 - DLM vector loops for Stacked Blocks Assembly

\[
[B]_1 = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-0.26347 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[ A \quad D \quad E \quad H \quad L \quad \phi_1 \quad \phi_2 \quad \phi_3 \quad \phi_4 \]

\[ \begin{align*}
0 & -0.96467 & 0 & 0 & 0 & -18.718 & 10.048 & 0 & 0 \\
1 & -0.26347 & -1 & 0 & 0 & 6.620 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{align*} \]

(4.35)
\[
[B]_2 = \begin{bmatrix}
A & D & E & H & L & \phi_1 & \phi_2 & \phi_3 & \phi_4 \\
0 & 0 & 0 & -0.96467 & 0 & 0 & 10.048 & 4.060 & 0 \\
0 & 0 & -1 & -0.26347 & 0 & 0 & 0 & -3.905 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
\end{bmatrix}
\] (4.36)

\[
[B]_3 = \begin{bmatrix}
A & D & E & H & L & \phi_1 & \phi_2 & \phi_3 & \phi_4 \\
0 & 0 & 0 & 0.96467 & -0.96467 & 0 & 0 & -4.060 & 10.675 \\
0 & 0 & 0 & 0.26347 & -0.26347 & 0 & 0 & 3.905 & -28.125 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
\end{bmatrix}
\] (4.37)

\[
[F]_1 = \begin{bmatrix}
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_5 & \alpha_6 & \alpha_7 \\
1 & 1 & 0.26347 & 0.26347 & -1 & 0 & 0 & 0 \\
0 & 0 & -0.96467 & -0.96467 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\] (4.38)

\[
[F]_2 = \begin{bmatrix}
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 \\
0 & 0 & 0 & 0 & -1 & -0.26347 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.96467 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\] (4.39)
\[
[F]_3 = \begin{bmatrix}
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 \\
0 & 0 & 0 & 0 & 0.26347 & -0.26347 \\
0 & 0 & 0 & 0 & -0.96467 & 0.96467 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\] (4.40)

The sensitivity matrices can be used in the VCAA method to solve for the constraints of the parts within the assembly. The underconstraints are solved using the steps outlined in Figure 4.1 beginning with the DLM variation analysis. The joint twists are formed for all five joints, which are labeled in Figure 4.8. The joint twists are shown in equations (4.41) through (4.45).

\[
[T]_{\text{cyllslider1}} = \begin{bmatrix}
T_{\alpha1} \\
T_{\alpha2} \\
T_{\alpha3} \\
T_{\alpha4} \\
T_{\alpha5} \\
T_{\alpha6} \\
T_{\alpha7}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & -1 & -18.718 & 6.620 & 0 \\
0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\] (4.41)

\[
[T]_{\text{cyllslider2}} = \begin{bmatrix}
T_{\alpha1} \\
T_{\alpha2} \\
T_{\alpha3} \\
T_{\alpha4} \\
T_{\alpha5} \\
T_{\alpha6} \\
T_{\alpha7}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & -1 & -18.718 & 6.620 & 0 \\
0 & 0 & 0 & -0.96467 & -0.26347 & 0
\end{bmatrix}
\] (4.42)

\[
[T]_{\text{edgslider1}} = \begin{bmatrix}
T_{\alpha1} \\
T_{\alpha2} \\
T_{\alpha3} \\
T_{\alpha4} \\
T_{\alpha5} \\
T_{\alpha6} \\
T_{\alpha7}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 1 & 10.048 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0
\end{bmatrix}
\] (4.43)

\[
[T]_{\text{edgslider2}} = \begin{bmatrix}
T_{\alpha1} \\
T_{\alpha2} \\
T_{\alpha3} \\
T_{\alpha4} \\
T_{\alpha5} \\
T_{\alpha6} \\
T_{\alpha7}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 1 & 4.060 & 0 & 0 \\
0 & 0 & 0 & 0.96467 & 0.26347 & 0
\end{bmatrix}
\] (4.44)

\[
[T]_{\text{edgslider3}} = \begin{bmatrix}
T_{\alpha1} \\
T_{\alpha2} \\
T_{\alpha3} \\
T_{\alpha4} \\
T_{\alpha5} \\
T_{\alpha6} \\
T_{\alpha7}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 1 & 10.675 & 0 & 0 \\
0 & 0 & 0 & -0.96467 & 0.26347 & 0
\end{bmatrix}
\] (4.45)

From the assembly joint twists the resultant twist matrix can be found for each part by first finding the intermediate joint wrenches and the intermediate part wrenches. Performing the necessary operations, for steps four and five, yields the resultant part twists found in the equations below.
\[
\mathbf{T}_{\text{Resultant - cylinder}} = \begin{bmatrix} 0 & 0 & 1 & 18.718 & -6.620 & 0 \end{bmatrix} \quad (4.46)
\]

\[
\mathbf{T}_{\text{Resultant - block}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.47)
\]

\[
\mathbf{T}_{\text{Resultant - base}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.48)
\]

The interpretation of the part twists shows that the block and the base have no underconstrained DOFs. The cylinder part has a mobile DOF of \( \theta_z \) located, by using the point algorithm, at the global point (18.718, 6.620, 0), its center axis point. This result can be verified by inspection. The second twist triplet will show rotation location information, but the order and signs are reversed. The reason for the changes is outlined in the derivation of the GCM shown in Appendix A.

The overconstraint information is solved for using the steps outlined in Figure 4.2 and starts with the \([\mathbf{F}]\) matrices for the loops. From these matrices the joint wrenches can be formed using the associated geometric feature variables. The joint wrenches for each of the five joints are listed in the equations below.

\[
\mathbf{W}_{\text{cylslider1}} = \begin{bmatrix} \mathbf{W}_{a1} & \mathbf{W}_{2D} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (4.49)
\]

\[
\mathbf{W}_{\text{cylslider2}} = \begin{bmatrix} \mathbf{W}_{a3} & \mathbf{W}_{2D} \end{bmatrix} = \begin{bmatrix} 0.26347 & -0.96467 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (4.50)
\]

\[
\mathbf{W}_{\text{edgelslider1}} = \begin{bmatrix} \mathbf{W}_{a5} & \mathbf{W}_{2D} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (4.51)
\]
\[
W_{\text{edgeslider2}} = \begin{bmatrix}
W_{a_6} \\
W_{2D}
\end{bmatrix} = \begin{bmatrix}
-0.26347 & 0.96467 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\]  \hspace{1cm} (4.52)

\[
W_{\text{edgeslider3}} = \begin{bmatrix}
W_{a_7} \\
W_{2D}
\end{bmatrix} = \begin{bmatrix}
-0.26347 & 0.96467 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\]  \hspace{1cm} (4.53)

The joint wrenches for each joint are used to determine the resultant wrenches for the different subsets of the assembly. By analyzing the different permutations of joints within the assembly and performing steps four and five, it is possible to find all of the subset wrenches. All of the subsets, except for one, yield the same result, the assembly is constrained to stay in the 2-D plane. The exception is the Loop 3 subset wrench. This wrench includes the geometric feature variables of \(a_6\) and \(a_7\), whose results are found in equation (4.54).

\[
W_{\text{Resultant-Loop3}} = \begin{bmatrix}
-0.2731 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\]  \hspace{1cm} (4.54)

The first row of the resultant wrench matrix for Loop 3 says that there is an overconstraint located in the global \(t_x\) and \(t_y\) directions, associated with the parts edgeslider2 and edgeslider3. This is showing an overconstraint in the extruded 3-D assembly. This overconstraint comes from the base contact with the block. The base and block are constrained by two lines of contact, which over-determines the location of the block. The overconstraints in both \(t_x\) and \(t_y\) show the effect of the block angle. Any variation in \(t_x\) will also result in variation of \(t_y\), and vice versa, because the directions are coupled due to the incline of the block. The overconstraint of the extruded 3-D stacked blocks assembly is shown in Figure 4.10.
Figure 4.10 - 3-D extruded view of Stacked Blocks Assembly with lines of constraint

The VCAA of the stacked blocks model yielded the underconstraint of the roller and the overconstraint of the base-block contact. By following the VCAA steps and correctly setting up the model for a DLM tolerance analysis, the constraint analysis can be performed easily.

4.3 3-D Examples of Constraint Analysis using the VCAA Method

The Variation-based Constraint Analysis of Assemblies method can also be applied to 3-D models. The DLM tolerance analysis rules, as well as the GCM results, are extremely important to the 3-D analysis. The two 3-D models presented in this section represent actual mechanical assemblies, which have been analyzed for variation and now can be analyzed for constraints.

4.3.1 Crank Slider Assembly

The crank slider assembly, shown in Figure 4.11, consists of four parts and four kinematic joints. The parts include a base, a crank arm and a slider connected by a link. As the crank arm rotates around the revolute joint, the prismatic slider translates within the base groove due to the contact from the link. The nominal dimensions for the independent and dependent variables, taken from [Gao 1993], are listed in Table 4.4 and the geometric tolerance callouts, as well as joint names, are shown in Figure 4.12.
Figure 4.11 - Crank Slider Assembly

Table 4.4 - Nominal Dimensions of Crank Slider Assembly ($\theta = 45^\circ$)

<table>
<thead>
<tr>
<th>Vector - Joint Name</th>
<th>Orientations (direction cosines $\omega_1, \omega_2, \omega_3$)</th>
<th>Global Joint Coordinates $(X, Y, Z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A - rigid</td>
<td>(0, 0, 1)</td>
<td>(0, 0, 0)</td>
</tr>
<tr>
<td>B - rigid</td>
<td>(-1, 0, 0)</td>
<td>(0, 0, 20)</td>
</tr>
<tr>
<td>C - revolute</td>
<td>(0, 0.7071, -0.7071)</td>
<td>(-12, 0, 20)</td>
</tr>
<tr>
<td>D - spherical1</td>
<td>(-0.9239, -0.3536, -0.1465)</td>
<td>(-12, 10.6066, 9.3934)</td>
</tr>
<tr>
<td>$\phi_x$ (x) - spherical1</td>
<td>(-0.9239, -0.3536, -0.1465)</td>
<td>(-12, 10.6066, 9.3934)</td>
</tr>
<tr>
<td>$\phi_y$ (y) - spherical1</td>
<td>(0, 0.7071, 0.7071)</td>
<td>(-12, 10.6066, 9.3934)</td>
</tr>
<tr>
<td>$\phi_z$ (z) - spherical1</td>
<td>(-0.1566, 0.6984, -0.6984)</td>
<td>(-12, 10.6066, 9.3934)</td>
</tr>
<tr>
<td>E - spherical2</td>
<td>(0, 0, -1)</td>
<td>(-39.7164, 0, 5)</td>
</tr>
<tr>
<td>$\phi_x$ (x) - spherical2</td>
<td>(0, 0, -1)</td>
<td>(-39.7164, 0, 5)</td>
</tr>
<tr>
<td>$\phi_y$ (y) - spherical2</td>
<td>(-0.3492, 0.6223, 0.7006)</td>
<td>(-39.7164, 0, 5)</td>
</tr>
<tr>
<td>$\phi_z$ (z) - spherical2</td>
<td>(0.8721, 0.4894, 0)</td>
<td>(-39.7164, 0, 5)</td>
</tr>
<tr>
<td>U - prismatic</td>
<td>(1, 0, 0)</td>
<td>(-39.7164, 0, 5)</td>
</tr>
</tbody>
</table>
As seen in Figure 4.11, the assembly requires only one vector loop for a variation analysis. The variation analysis of a mechanism requires multiple analyses of the assembly in one static pose at a time. The revolute joint angle, $\theta$, can be rotated around a full $360^\circ$, but the variation analysis is performed only at discrete values of $\theta$. In Table 4.4, the values reflect the nominal assembly dimensions when $\theta = 45^\circ$. Due to the model conditions, the revolute joint is kept essentially rigid for the tolerance analysis. This implies that the revolute joint will not be included in the constraint analysis. The independent variables of the assembly include $A$, $B$, $C$, $D$, and $E$, while the dependent variables are the angles $\phi_1$, $\phi_2$, $\phi_3$, $\phi_4$, $\phi_5$, $\phi_6$, and the translation $U$. Step one of the VCAA method requires the variation analysis to be performed. The DLM analysis produces the sensitivity matrices, $[B]$ and $[F]$, listed below.

$$[B] = \begin{bmatrix}
\phi_1 & \phi_2 & \phi_3 & \phi_4 & \phi_5 & \phi_6 & U \\
1.7678 & 0.8579 & -13.9677 & 0 & -3.1115 & -2.4468 & 1 \\
-0.9239 & 0 & -0.1566 & 0 & -0.3492 & 0.8721 & 0 \\
-0.3536 & 0.7071 & 0.6984 & 0 & 0.6223 & 0.4894 & 0 \\
-0.1464 & 0.7071 & -0.6984 & -1 & 0.7006 & 0 & 0
\end{bmatrix}$$

(4.55)
\[
[F] = \begin{bmatrix}
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 \\
0 & 0 & -0.9239 & -0.3492 & -0.1566 \\
0.7071 & 0.7071 & -0.3536 & 0.6223 & 0.6984 \\
-0.7071 & 0.7071 & -0.1465 & 0.7006 & -0.6984 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\alpha_6 & \alpha_7 & \alpha_8 & \alpha_9 & \alpha_{10} & \beta_1 & \beta_2 & \beta_3 \\
0 & -0.4894 & 0.8721 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.8721 & 0.4894 & 1 & 0 & 0 & 0 & 39.7164 \\
-1 & 0 & 0 & 0 & -1 & 0 & -39.7164 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

(4.56)

The 3-D underconstraint solution will follow all the steps shown in Figure 4.1. Step two of the process requires forming the column twist matrix by transposing the \([B]\) matrix and switching the first three entries in each row with the last three. This matrix is shown in (4.57).

\[
T_{\text{column}} = \begin{bmatrix}
\phi_1 \\
\phi_2 \\
\phi_3 \\
\phi_4 \\
\phi_5 \\
\phi_6 \\
U \\
\end{bmatrix} = \begin{bmatrix}
-0.9239 & -0.3536 & -0.1464 & 1.7678 & -10.4357 & 14.0419 \\
0 & 0.7071 & 0.7071 & 0.8579 & 8.4853 & -8.4853 \\
-0.1566 & 0.6984 & -0.6984 & -13.9677 & -9.8513 & -6.7201 \\
-0.3492 & 0.6223 & 0.7006 & -3.1115 & 26.0784 & -24.7154 \\
0.8721 & 0.4894 & 0 & -2.4468 & 4.3604 & -19.4355 \\
0 & 0 & 0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

(4.57)

Step three takes the rows of \(T_{\text{column}}\) and forms the joint twists for each joint in the assembly. There are three joints that will be analyzed, the two spherical joints and the prismatic joint. The revolute joint is held rigid for the tolerance analysis and is not included in the constraint analysis. The joint twists are composed of the dependent variable rows that are associated with each joint.
\[ T_{\text{spherical1}} = \begin{bmatrix} T_{\text{column-\phi1}} \\ T_{\text{column-\phi2}} \\ T_{\text{column-\phi3}} \end{bmatrix} \]

\[
= \begin{bmatrix} -0.9239 & -0.3536 & -0.1464 & 1.7678 & -10.4357 & 14.0419 \\ 0 & 0.7071 & 0.7071 & 0.8579 & 8.4853 & -8.4853 \\ -0.1566 & 0.6984 & -0.6984 & -13.9677 & -9.8513 & -6.7201 \end{bmatrix}
\]

\[ T_{\text{spherical2}} = \begin{bmatrix} T_{\text{column-\phi4}} \\ T_{\text{column-\phi5}} \\ T_{\text{column-\phi6}} \end{bmatrix} \]

\[
= \begin{bmatrix} 0 & 0 & -1 & 0 & -39.7164 & 0 \\ -0.3493 & 0.6223 & 0.7006 & -3.1115 & 26.0784 & -24.7154 \\ 0.8721 & 0.4894 & 0 & -2.4468 & 4.3604 & -19.4355 \end{bmatrix}
\]

\[ T_{\text{prismatic}} = \begin{bmatrix} T_{\text{column-\phi'}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \]

After the joint twists have been formed, the intermediate joint wrenches are formed by performing the reciprocal operation. These intermediate joint wrenches, with values rounded to the nearest $1 \times 10^{-6}$ value, are shown in equations (4.61) through (4.63).

\[ W_{\text{intermediate-\text{spherical1}}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 9.3934 & -10.6066 \\ 0 & 1 & 0 & -9.3934 & 0 & 0 & -12 \\ 0 & 0 & 1 & 10.6066 & 12 & 0 \end{bmatrix} \]

\[ W_{\text{intermediate-\text{spherical2}}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 1 & 0 & -5 & 0 & -39.7164 \end{bmatrix} \]

\[ W_{\text{intermediate-\text{prismatic}}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \]
Step five takes the intermediate joint wrench matrices and forms the intermediate part matrices for each part in the assembly. As stated previously, the crank part of the assembly is treated as if it were rigidly attached to the base, while rotated through discrete values of \( \theta \). This leaves three parts to analyze for underconstraints: the base, the link and the slider. The base mates to the assembly through the spherical1 joint and the prismatic joint. The link is connected to the assembly by the two spherical joints and the slider is bound by the spherical2 and prismatic joints. The associated intermediate joint wrenches are shown in equations (4.64) through (4.66).

\[
W_{\text{intermediate-base}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 9.3934 & -10.6066 \\
0 & 1 & 0 & -9.3934 & 0 & -12 \\
0 & 0 & 1 & 10.6066 & 12 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\]  

(4.64)

\[
W_{\text{intermediate-link}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 9.3934 & -10.6066 \\
0 & 1 & 0 & -9.3934 & 0 & -12 \\
0 & 0 & 1 & 10.6066 & 12 & 0 \\
1 & 0 & 0 & 0 & 5 & 0 \\
0 & 1 & 0 & -5 & 0 & -39.7164 \\
0 & 0 & 1 & 0 & 39.7164 & 0
\end{bmatrix}
\]  

(4.65)

\[
W_{\text{intermediate-slider}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 5 & 0 \\
0 & 1 & 0 & -5 & 0 & -39.7164 \\
0 & 0 & 1 & 0 & 39.7164 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\]  

(4.66)
The sixth and final step for identifying underconstraints involves finding the resultant twist matrices for each part in the assembly. This is done by applying the reciprocal operation to the intermediate joint wrenches. The resultant twists for the three parts are shown in the equations below.

\[
T_{\text{Resultant - base}} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0
\end{bmatrix} \quad (4.67)
\]

\[
T_{\text{Resultant - link}} = \begin{bmatrix}
6.3086 & 2.4142 & 1 & -12.0711 & 71.2596 & -95.8838
\end{bmatrix} \quad (4.68)
\]

\[
T_{\text{Resultant - slider}} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0
\end{bmatrix} \quad (4.69)
\]

The resultant twists show that there are no underconstrained DOFs in the base or the slider parts. There is no unexpected motion in those two parts due to the mating conditions. The link part has a rotation allowed by the two mating spherical joints. This rotation is a combined rotation in the global \( \theta_x, \theta_y, \) and \( \theta_z \) directions. To locate this rotation a variation of the point algorithm is employed. This point3d algorithm, explained in the appendix, shows the location of the rotational axis to be through the global points (-12, 10.6066, 9.3934) and (-39.7164, 0, 5). This shows that a free rotation exists along the longitudinal axis of the link, between the two spherical joints. This longitudinal rotation shows up as a rotation around all three axis directions. Also, the ratios between the \( \theta_x, \theta_y, \) and \( \theta_z \) values in the resultant twist show the magnitude of the respective angular rotation components. The VCAA method analytically identifies the underconstraint using the DLM tolerance analysis information.

The overconstraint analysis of the crank slider assembly uses the information contained in the [F] matrix, shown in equation (4.56). Following the VCAA procedure, the second step requires the formation of the \( W_{\text{column}} \) matrix from the transpose of [F]. This matrix is shown in equation (4.70), where \( \alpha_i \) represents the geometric feature translations and \( \beta_i \) represents the geometric feature rotations.
\[
\begin{bmatrix}
\alpha_1 & 0 & 0.7071 & -0.7071 & 0 & 0 & 0 \\
\alpha_2 & 0 & 0.7071 & 0.7071 & 0 & 0 & 0 \\
\alpha_3 & -0.9239 & -0.3536 & -0.1465 & 0 & 0 & 0 \\
\alpha_4 & -0.3492 & 0.6223 & 0.7006 & 0 & 0 & 0 \\
\alpha_5 & -0.1566 & 0.6984 & -0.6984 & 0 & 0 & 0 \\
\alpha_6 & 0 & 0 & -1 & 0 & 0 & 0 \\
\alpha_7 & -0.4894 & 0.8721 & 0 & 0 & 0 & 0 \\
\alpha_8 & 0.8721 & 0.4894 & 0 & 0 & 0 & 0 \\
\alpha_9 & 0 & 1 & 0 & 0 & 0 & 0 \\
\alpha_{10} & 0 & 0 & -1 & 0 & 0 & 0 \\
\beta_1 & 0 & 0 & 0 & 1 & 0 & 0 \\
\beta_2 & 0 & 0 & -39.7164 & 0 & 1 & 0 \\
\beta_3 & 0 & 39.7164 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

(4.70)

This matrix is used to form the joint wrenchmatrices. As with the underconstraint solution, there are three joints to analyze in this assembly, the two spherical joints and the prismatic (the revolute joint acting as a rigid joint). Using the associated geometric feature variables, both translation and rotation, it is possible to form the joint wrenches for all three joints. The spherical1 joint is associated with the \( \alpha_3, \alpha_4, \) and \( \alpha_5 \) translation variables; the spherical2 joint is related to the \( \alpha_6, \alpha_7, \) and \( \alpha_8 \) translation variables. The prismatic joint has two translations, \( \alpha_9 \) and \( \alpha_{10} \), and three rotations, \( \beta_1, \beta_2, \)and \( \beta_3 \). These joint wrenches are shown in the equations below.

\[
W_{\text{spherical1}} = \begin{bmatrix} W_{\alpha_3} \\ W_{\alpha_4} \\ W_{\alpha_5} \end{bmatrix} = \begin{bmatrix} -0.9239 & -0.3536 & -0.1465 & 0 & 0 & 0 \\ -0.3492 & 0.6223 & 0.7006 & 0 & 0 & 0 \\ -0.1566 & 0.6984 & -0.6984 & 0 & 0 & 0 \end{bmatrix}
\]

(4.71)

\[
W_{\text{spherical2}} = \begin{bmatrix} W_{\alpha_6} \\ W_{\alpha_7} \\ W_{\alpha_8} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 & 0 \\ -0.4894 & 0.8721 & 0 & 0 & 0 & 0 \\ 0.8721 & 0.4894 & 0 & 0 & 0 & 0 \end{bmatrix}
\]

(4.72)
\[
\mathbf{W}_{\text{prismatic}} = \begin{bmatrix}
W_{a_{1}} & 0 & 1 & 0 & 0 & 0 \\
W_{a_{2}} & 0 & 0 & -1 & 0 & 0 \\
W_{\beta_{1}} & 0 & 0 & 0 & 1 & 0 \\
W_{\beta_{2}} & 0 & 0 & -39.7164 & 0 & 1 \\
0 & 39.7164 & 0 & 0 & 0 & 1
\end{bmatrix}
\] (4.73)

The next step finds the intermediate twist matrices by applying the reciprocal function on the joint wrenches. The resulting twists are shown in the equations below.

\[
\mathbf{T}_{\text{intermediate-spherical1}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\] (4.74)

\[
\mathbf{T}_{\text{intermediate-spherical2}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\] (4.75)

\[
\mathbf{T}_{\text{intermediate-prismatic}} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\] (4.76)

These intermediate twists are joined, through a union operation, into different subsets to solve for varying overconstraints. This assembly yields overconstraints in each of the parts. The three parts, the base, the link, and the slider, are treated as the subsets of overconstraint analysis. The intermediate base twist is a union of the prismatic and the spherical1 joint twists. The intermediate link twist is a union of the two spherical joint twists and the slider twist is a union of the prismatic and spherical2 joints. These unions give the three intermediate part twists, shown in equations (4.77) to (4.79).

\[
\mathbf{T}_{\text{intermediate-base}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\] (4.77)
\[ T_{\text{intermediate\text{-}link}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0
\end{bmatrix} \] (4.78)

\[ T_{\text{intermediate\text{-}slider}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix} \] (4.79)

The final step of overconstraint analysis takes the intermediate twist matrices for the parts and solves for their respective resultant wrench matrices. This is done through the reciprocal operation. Finding the reciprocal of each intermediate part twist gives the resultant part wrenches found in the equations below.

\[ W_{\text{Resultant\text{-}base}} = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0
\end{bmatrix} \] (4.80)

\[ W_{\text{Resultant\text{-}link}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0
\end{bmatrix} \] (4.81)

\[ W_{\text{Resultant\text{-}slider}} = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0
\end{bmatrix} \] (4.82)

The interpretation of these matrices shows the overconstrained DOFs that are present on each of the parts. The resultant wrench matrix for the base shows an overconstraint in the \( t_y \) and \( t_z \) directions. This result shows up because the spherical1 joint and the slider joint both constrain those two directions. The resultant wrench for the link shows an overconstraint in the \( t_x \), \( t_y \), and \( t_z \) directions, which is because spherical1 joint constrains the same DOFs that spherical2 joint does. The slider wrench shows the
same overconstraint condition as the base, an overconstraint in the $t_y$ and $t_z$ directions. This says that the prismatic contact removes those two DOFs and the spherical2 joint does so redundantly. As stated, the resultant wrench shows redundantly defined DOFs, which may or may not be assembly design problems. Because this method is conservative in its approach to overconstraints, the designer must also inspect the assembly as these warnings appear. In this example, the crank slider assembly overconstraints do not signify a problem in the assembly.

### 4.3.2 Swash Plate Assembly

The final 3-D assembly that will be analyzed is the swash plate assembly pictured in Figure 4.13. The assembly consists of an angled plate rigidly attached to a central shaft, which can rotate around the shaft axis. The cylinder part can pivot about the follower part, which can also pivot from the arm. The mechanism is designed to maintain contact between the cylinder and the swash plate. The nominal dimensions for the independent and dependent variables are listed in Table 4.5 and the geometric tolerance callouts, as well as joint names, are shown in Figure 4.14.

![Swash Plate Assembly Diagram](image)

*Figure 4.13 - Swash Plate Assembly*
Table 4.5 - Nominal Dimensions for Swash Plate Assembly (θ = 180°)

<table>
<thead>
<tr>
<th>Vector - Joint Name</th>
<th>Orientations (direction cosines ω₁, ω₂, ω₃)</th>
<th>Global Joint Coordinates (X, Y, Z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A - rigid</td>
<td>(0, -1, 0)</td>
<td>(0, 5, 0)</td>
</tr>
<tr>
<td>B - rigid</td>
<td>(-1, 0, 0)</td>
<td>(0, 3.5, 0)</td>
</tr>
<tr>
<td>C - rigid</td>
<td>(0, -1, 0)</td>
<td>(-5, 3.5, 0)</td>
</tr>
<tr>
<td>D - revolute1</td>
<td>(0, 0, 1)</td>
<td>(-5, 3, 0)</td>
</tr>
<tr>
<td>E - revolute1</td>
<td>(0, 0, 1)</td>
<td>(-5, 3, 2.5)</td>
</tr>
<tr>
<td>φ₁ - revolute1</td>
<td>(0, 0, 1)</td>
<td>(-5, 3, 2.5)</td>
</tr>
<tr>
<td>F - revolute1</td>
<td>(0.948405, -0.3170608, 0)</td>
<td>(-5, 3, 3.75)</td>
</tr>
<tr>
<td>G - revolute1</td>
<td>(0.948405, -0.3170608, 0)</td>
<td>(0.2553731, 1.2431072, 3.75)</td>
</tr>
<tr>
<td>H - revolute2</td>
<td>(-0.3170608, -0.948405, 0)</td>
<td>(0.0651366, 0.674064, 3.75)</td>
</tr>
<tr>
<td>φ₂ - revolute2</td>
<td>(0.948405, -0.3170608, 0)</td>
<td>(0.0651366, 0.674064, 3.75)</td>
</tr>
<tr>
<td>l - cylslider</td>
<td>(-0.17364818, -0.9848101, 0)</td>
<td>(0, 0.304628, 3.75)</td>
</tr>
<tr>
<td>φ₃ - cylslider</td>
<td>(0, 0, -1)</td>
<td>(0, 0.304628, 3.75)</td>
</tr>
<tr>
<td>U₁ - cylslider</td>
<td>(0.9848101, -0.17364818, 0)</td>
<td>(0, 0.304628, 3.75)</td>
</tr>
<tr>
<td>U₂ - cylslider</td>
<td>(0, 0, -1)</td>
<td>(0, 0.304628, 3.75)</td>
</tr>
<tr>
<td>L - cylslider</td>
<td>(0, 1, 0)</td>
<td>(0, 0.304628, 0)</td>
</tr>
</tbody>
</table>

Figure 4.14 - Joint types and geometric tolerances for Swash Plate Assembly
This assembly requires only one vector loop for a variation analysis and, as with the crank slider assembly, is analyzed one static pose at a time. The swash plate angle, $\theta$, can be rotated a full $360^\circ$, but the variation analysis is only done at discrete values of $\theta$. The nominal assembly dimensions shown in Table 4.5 are for a rotation of $\theta = 180^\circ$ around the $Y$-axis, where $\theta$ is measured between the center line of the arm and the $X$-axis, which marks the lowest point on the plate. Due to these conditions, the cylindrical joint between the swash plate shaft and the arm is treated as rigid. The independent variables of the assembly are $A$, $B$, $C$, $D$, $E$, $F$, $G$, $H$, $I$, and $L$, while the dependent variables are the angles $\phi_1$, $\phi_2$, and $\phi_3$, as well as the translations $U_1$, and $U_2$. A few other conditions complete the swash plate model. The cylindrical slider joint, cylder, has one of its rotations, the rotation around an axis normal to the plate, removed because it would not be allowed by the assembly. Lastly, there is no geometric tolerance callout assigned to the cylindrical joint between the shaft and the arm, because it is treated as essentially rigid. Step one of the VCAA method requires the $[B]$ and $[F]$ matrices from the DLM tolerance analysis. They are shown in the equations below.

$$
[B] = \begin{bmatrix}
\phi_1 & \phi_2 & \phi_3 & U_1 & U_2 \\
3 & 1.188978 & -0.674064 & 0.984808 & 0 \\
5 & 3.55652 & 0.065137 & -0.173648 & 0 \\
0 & -1.25994 & 0 & 0 & -1 \\
0 & 0.948405 & 0 & 0 & 0 \\
0 & -0.31706 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0
\end{bmatrix}
$$

(4.83)
The underconstraint solution involves computing the column twist matrix from the \([B]\) matrix. From the column twist matrix, the joint twists can be formed for each joint in the assembly through a union of the rows of the associated dependent variables. The joint twists for the two revolute joints and the cylinder slider joints are shown in the equations below. It should be noted that the cylinder slider only has three DOFs due to the removed rotation mentioned above.

\[
\begin{bmatrix}
\alpha_1 & \alpha_2 & \alpha_3 & \beta_1 & \beta_2 \\
0.948407 & -0.317061 & 0 & 1.188959 & 3.55652 \\
-0.317056 & -0.948405 & 0 & 3.556526 & -1.18898 \\
0 & 0 & -1 & -1.25994 & 5.69321 \\
0 & 0 & 0 & 0.948407 & -0.317061 \\
0 & 0 & 0 & -0.317056 & -0.948405 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
\alpha_4 & \alpha_5 & \alpha_6 & \beta_3 & \beta_4 & \alpha_7 & \beta_\theta_5 \\
-0.317061 & -0.948407 & 0 & 3.55652 & -1.243107 & 0.173635 & -0.65118 \\
-0.94841 & 0.317056 & 0 & -1.18898 & 0.255373 & 0.98481 & -3.69303 \\
0 & 0 & -1 & 0.151943 & 0 & 0 & 0.3 \\
0 & 0 & 0 & -0.31706 & 0 & 0 & -0.984808 \\
0 & 0 & 0 & -0.948405 & 0 & 0 & 0.1736482 \\
0 & 0 & 0 & 0 & -1 & 0 & 0
\end{bmatrix}
\]  

\[
\mathbf{T}_{\text{revolute1}} = [\mathbf{T}_{\phi_1}] = \begin{bmatrix} 0 & 0 & 1 & 3 & 5 & 0 \end{bmatrix} \tag{4.85}
\]

\[
\mathbf{T}_{\text{revolute2}} = [\mathbf{T}_{\phi_2}] = \begin{bmatrix} 0.948408 & -0.31706 & 0 & 1.18898 & 3.55652 & -1.256638 \end{bmatrix} \tag{4.86}
\]

136
\[ T_{\text{cylinder}} = \begin{bmatrix} T_{s} \\ T_{v1} \\ T_{v2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & -0.67406 & 0.065137 & 0 \\ 0 & 0 & 0 & 0.984808 & -0.173648 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \] (4.87)

With these joint twists, it is simple to follow the remainder of the steps to identify any underconstraints. The joint twists are transformed into the intermediate wrench matrices using the reciprocal operation. Next, the intermediate part wrenches are formed through a union of the joint wrenches, for the swash plate, follower, and cylinder parts. The final step is finding the resultant twistmatrices for each of the parts. This is accomplished by taking the reciprocal of the intermediate part wrenches. The resultant part twistmatrices for the swash plate assembly are shown in equations (4.88) through (4.90).

\[ T_{\text{Resultant - plate}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix} \] (4.88)

\[ T_{\text{Resultant - follower}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix} \] (4.89)

\[ T_{\text{Resultant - cylinder}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix} \] (4.90)

The results of the underconstraint analysis shows that no parts of the swash plate assembly contain underconstrained DOFs. The swash plate variation analysis shows that no parts of the assembly will have unaccounted mobile DOFs.

The overconstraint solution of the swash plate assembly involves the formation of the joint wrenches, using the associated geometric feature variables from the column wrench matrix, originally formed by the \([F]\) matrix. The joint wrenches for all three joints are shown in the equations below.
\[
W_{\text{revolute1}} = \begin{bmatrix}
W_{a_1} \\
W_{a_2} \\
W_{a_3} \\
W_{\beta_1} \\
W_{\beta_2}
\end{bmatrix}
\]
\[
= \begin{bmatrix}
0.948407 & -0.317056 & 0 & 0 & 0 & 0 \\
-0.31706 & -0.948405 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 \\
1.188959 & 3.556525 & -1.25994 & 0.948407 & -0.317056 & 0 \\
3.55652 & -1.18898 & 5.693208 & -0.317061 & -0.948405 & 0
\end{bmatrix}
\] (4.91)

\[
W_{\text{revolute2}} = \begin{bmatrix}
W_{a_4} \\
W_{a_5} \\
W_{a_6} \\
W_{\beta_1} \\
W_{\beta_2}
\end{bmatrix}
\]
\[
= \begin{bmatrix}
-0.317061 & -0.948405 & 0 & 0 & 0 & 0 \\
-0.948407 & 0.317056 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 \\
3.5565195 & -1.18898 & 0.151943 & -0.317061 & -0.948405 & 0 \\
-1.243107 & 0.255373 & 0 & 0 & 0 & -1
\end{bmatrix}
\] (4.92)

\[
W_{\text{cylslider}} = \begin{bmatrix}
W_{a_7} \\
W_{\beta_1}
\end{bmatrix}
\]
\[
= \begin{bmatrix}
0.173635 & 0.98481 & 0 & 0 & 0 & 0 \\
-0.6511807 & -3.69303 & 0.3 & -0.984808 & 0.173648 & 0
\end{bmatrix}
\] (4.93)

With the joint wrenches formed, it is possible to apply the screw theory steps to find the resultant wrenchmatrices with the overconstraint information. This is done by first finding the intermediate joint twist matrices, through the reciprocal operation. Then the intermediate twist matrices for the subsets are formed as a union of the intermediate joint twists. The subsets used in this assembly are the three parts, the plate, the follower, and the cylinder. The resultant wrenchmatrices are found by taking the reciprocal of the
intermediate part wrenches. The resultant wrenches for the swash plate assembly will show which DOFs are redundantly constrained. The resultant wrenches are shown in equations (4.94) through (4.96).

\[
W_{\text{Resultant - plate}} = \begin{bmatrix}
0.1763136 & 1 & 0 & 0 & 0 \\
-0.00165 & 0 & 1 & -3.28269 & 0.578827 & 0
\end{bmatrix}
\]  \hspace{1cm} (4.94)

\[
W_{\text{Resultant - follower}} = \begin{bmatrix}
0 & 0 & 0 & 0.334309 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0
\end{bmatrix}
\]  \hspace{1cm} (4.95)

\[
W_{\text{Resultant - cylinder}} = \begin{bmatrix}
0.1763136 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
\]  \hspace{1cm} (4.96)

The interpretation of these resultant wrenches shows the redundantly constrained DOFs. The resultant wrench for the plate shows two rows of coupled constraints. The joints of the assembly constrain the plate jointly in \( t_x \) and \( t_y \). Also, the \( t_x, t_y, \theta_x, \) and \( \theta_y \) directions are jointly constrained by the mating conditions on the plate-arm part. These constraints originate from the revolute joint between the arm and the follower as well as the cylinder slider. These overconstraints only show the redundantly defined DOFs on the plate and do not denote a problem in the design. The resultant wrench for the follower shows a coupled overconstraint in the rotations of \( \theta_x \) and \( \theta_y \) due to the two revolute joints. The next three rows show that the translational DOFs of the follower are constrained redundantly by each revolute joint. The resultant wrench for the cylinder simply shows that it is constrained in the \( t_x \) and \( t_y \) directions by the revolute joint and the cylinder slider mate. These overconstraints simply show the redundantly defined or constrained DOFs of the parts within the assembly. They are cautions and do not necessarily denote a problem in the design of the assembly.
4.4 Conclusions of the VCAA Example Problems

Using the VCAA methodology, both 2-D and 3-D problems can be analyzed for constraints using the variation information gained from a tolerance analysis. The steps outlined and the methodology shown in this chapter can be applied to a variety of assemblies. If the DLM tolerance loops are constructed correctly and the geometric tolerances are assigned correctly, screw theory can be used to manipulate the information contained in the variation sensitivity matrices.

The variation information contained in the $[\mathbf{B}]$ and $[\mathbf{F}]$ matrices is the basis for finding the resultant twists and wrenches for parts within assemblies. The resultant twists will show the underconstrained DOFs of the parts where motion may still be allowed by the assembly. The resultant wrenches will show which DOFs have been redundantly constrained and act as a warning to the designer for possible assembly problems. Both cases of under- and overconstrained parts can be identified by the VCAA method and the designer can then use the information gained to improve the assembly.
CHAPTER 5. COMPARISONS AND RESULTS OF VCAA METHOD

The VCAA method demonstrates the unique connection between tolerance analysis and screw theory constraint analysis. The relationship between the DLM and screw theory is based upon the sensitivities found by the Global Coordinate Method. The reason the GCM yields twists and wrenches is due to its derivation and application in variation analysis. The vector loops define the assembly constraints and part interactions. The results of VCAA and a comparison with other constraint methods will be presented in this chapter.

5.1 Comparisons of VCAA to Previous Constraint Analysis Methods

A comparison of the VCAA method to current methods will show its advantages and disadvantages. Because of the link with screw theory, it will be advantageous to first compare the constraint analysis of [Adams 1998] to the other two methods reviewed in Chapter 2. Following the comparison of the current methods, the VCAA method can be compared to the screw theory-based method for advantages or disadvantages.

The kinematic Constraint Pattern Analysis method discussed in [Blanding 1999] and [Kriegel 1994] sets forth a methodical design outline for eliminating constraint problems. It is based upon a knowledge and comprehension of part reaction to a constraint of the DOFs. This method is a qualitative one, which states a clear methodology without presenting the underlying mathematical foundation. A connection between VCAA and this method could be discovered in that there are a variety of similarities that can be explored. The relationship between constraints (C) and rotational
freedoms (R) is very similar to that of the relationship between constrained DOFs (f) and mobile DOFs (k). Both pairs are mutually exclusive and must sum to six, for each rigid body or joint. A connection between Constraint Pattern Analysis and VCAA may exist, but cannot be expressed mathematically. The steps involved in this method are logical to follow, but it becomes difficult to mathematically quantify the problems and the models.

The Geometric Constraint Solving method discussed in [Hoffmann and Vermeer 1995] takes computer models and applies the principles of clusters, network graphs, graph theory, and geometric entities to initial formation of geometric models. First, geometric entities are considered, then cluster formations are grouped together. Next, the basic construction steps are followed ending with cluster merging and analysis. This is a computer-driven mathematically-based method which allows for the detailed analysis of computer-aided design models. This method is programmable and can be incorporated into a CAD program well. The underlying theory is somewhat difficult to understand and the definition of nodes, geometric entities, and so forth, is non-intuitive.

The Screw Theory-Based Constraint Analysis method used in [Adams and Whitney 2001] shows under- and overconstraints based on the DOFs of the joints and the mating conditions of the parts. The mathematical nature of the method is simple to automate and simple to use. Necessary information includes the types of features on the part and their locations and orientations. The formation of twists and wrenches is simple and accurately describes the DOF conditions. The underconstraint motion analysis is accurate and useful, but the overconstraint analysis is overly cautious.

Of the current methods reviewed in this thesis, the Screw Theory-Based Constraint Analysis method is superior. Due to its ease of use, its mathematical basis, and the simplicity of the necessary information, the screw theory method is the benchmark of the current methods. Because of this, the VCAA method will be compared to this method and the advantages and disadvantages will be discussed. This comparison will be shown through the analysis of a case study.
The VCAA method showed that the information contained in the variation sensitivity loops were actually twists and wrenches. The steps of each method are therefore almost identical, with the screw theory operations used in both methods being the ones presented in [Adams 1998]. Through a case study it is possible to see the differences between the two methods. The stacked blocks model presented in Chapter 4 will be the example used to compare the screw theory method with the VCAA method. The geometry for the stacked blocks model was shown in Figure 4.7, here reprinted as Figure 5.1 and the dimensions were presented in Table 4.3, here reprinted as Table 5.1.

![Figure 5.1 - Stacked Blocks Assembly](image)

**Table 5.1 - Nominal Dimensions of Stacked Blocks Assembly**

<table>
<thead>
<tr>
<th>Independent Variables</th>
<th>Basic Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cylinder Radius - B</td>
<td>6.620 mm</td>
</tr>
<tr>
<td>Cylinder Radius - C</td>
<td>6.620 mm</td>
</tr>
<tr>
<td>Base Step Width - F</td>
<td>3.905 mm</td>
</tr>
<tr>
<td>Base Step Height - G</td>
<td>4.060 mm</td>
</tr>
<tr>
<td>Block Thickness - I</td>
<td>6.805 mm</td>
</tr>
<tr>
<td>Base Step Location - J</td>
<td>28.125 mm</td>
</tr>
<tr>
<td>Base Step Height - K</td>
<td>10.675 mm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dependent Variables</th>
<th>Basic Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cylinder/Base Contact - A</td>
<td>18.7182 mm</td>
</tr>
<tr>
<td>Angle $\phi_1$</td>
<td>74.724°</td>
</tr>
<tr>
<td>Cylinder/Block Contact - D</td>
<td>8.6705 mm</td>
</tr>
<tr>
<td>Angle $\phi_2$</td>
<td>164.724°</td>
</tr>
<tr>
<td>Block/Ground Contact - E</td>
<td>10.0477 mm</td>
</tr>
<tr>
<td>Angle $\phi_3$</td>
<td>105.276°</td>
</tr>
<tr>
<td>Block/Ground Contact - H</td>
<td>2.1894 mm</td>
</tr>
<tr>
<td>Angle $\phi_4$</td>
<td>105.276°</td>
</tr>
<tr>
<td>Block/Ground Contact - L</td>
<td>27.2965 mm</td>
</tr>
</tbody>
</table>
The comparison begins with the underconstraint analysis and the formation of the joint twist. The method in [Adams 1998] uses the geometry and the location of the joints to form joint twists from the homogeneous transformations, according to the equations shown in Figures 3.7-3.8. The VCAA method forms the joint twists from the variation sensitivity [B] matrix. The velocity-variation analogy predicts that the joint twist will be the same, because they are representing the same motions. The joint twists formed from the VCAA method are shown in equations (5.1) through (5.5).

\[
T_{\text{Cylslider1}} = \begin{bmatrix} T_h \n T_d \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & -18.718 & 6.620 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.1)
\]

\[
T_{\text{Cylslider2}} = \begin{bmatrix} T_h \n T_d \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & -18.718 & 6.620 & 0 \\ 0 & 0 & 0 & -0.96467 & -0.26347 & 0 \end{bmatrix} \quad (5.2)
\]

\[
T_{\text{Edgeslider1}} = \begin{bmatrix} T_h \n T_d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 10.048 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix} \quad (5.3)
\]

\[
T_{\text{Edgeslider2}} = \begin{bmatrix} T_h \n T_d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 4.060 & -3.905 & 0 \\ 0 & 0 & 0 & 0.96467 & 0.26347 & 0 \end{bmatrix} \quad (5.4)
\]

\[
T_{\text{Edgeslider3}} = \begin{bmatrix} T_h \n T_d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 10.675 & -28.125 & 0 \\ 0 & 0 & 0 & -0.96467 & -0.26347 & 0 \end{bmatrix} \quad (5.5)
\]

For comparison, the joint twists calculated by the screw theory in [Adams 1998] and the equations in Figures 3.7-3.8 are shown in equations (5.6) through (5.10).

\[
T_{\text{Cylslider1}} = \begin{bmatrix} 0 & 0 & 1 & 18.718 & -6.620 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.6)
\]

\[
T_{\text{Cylslider2}} = \begin{bmatrix} 0 & 0 & 1 & 18.718 & -6.620 & 0 \\ 0 & 0 & 0 & 0.96467 & 0.26347 & 0 \end{bmatrix} \quad (5.7)
\]

\[
T_{\text{Edgeslider1}} = \begin{bmatrix} 0 & 0 & 1 & 10.048 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.8)
\]
\[
\mathbf{T}_{\text{edgeslider}2} = \begin{bmatrix}
0 & 0 & 1 & 4.060 & -3.905 & 0 \\
0 & 0 & 0 & 0.96467 & 0.26347 & 0
\end{bmatrix}
\] (5.9)

\[
\mathbf{T}_{\text{edgeslider}3} = \begin{bmatrix}
0 & 0 & 1 & 10.675 & -28.125 & 0 \\
0 & 0 & 0 & 0.96467 & 0.26347 & 0
\end{bmatrix}
\] (5.10)

Automatically, it can be seen that the joint twists gained from each method are exactly the same. The only exception are rows in one method that are negative of their counterparts in the other method. These negatives are irrelevant in that linear algebra operations can quickly account for the difference. The resultant twist for both methods yield the same outcome. The only part that has any underconstrained DOFs is the cylinder, which can rotate about the \( \theta_z \) axis, located at the global point (18.718, 6.620, 0). The resultant twists for both methods are shown below. Both methods use the same steps to solve for the resultant twist, as shown in Figure 5.2, while the means of getting the joint twists remains different.

\[
\mathbf{T}_{\text{Resultant—VCAA Based}} = \begin{bmatrix}
0 & 0 & 1 & 18.718 & -6.620 & 0 \\
\end{bmatrix}
\] (5.11)

\[
\mathbf{T}_{\text{Resultant—Screw Based}} = \begin{bmatrix}
0 & 0 & 1 & 18.718 & -6.620 & 0 \\
\end{bmatrix}
\] (5.12)

**VCAA / Screw Theory Based**

<table>
<thead>
<tr>
<th>JointTwists (from different sources)</th>
<th>T</th>
<th>T</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate Joint Wrenches</td>
<td>W</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>Intermediate Part Wrenches</td>
<td></td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>Resultant Twist</td>
<td></td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 5.2 - Comparison of methods for underconstraint solution*
The overconstraint treatment in the VCAA method differs from the one presented in [Adams 1998]. The difference lies in the starting point. The VCAA method uses the joint wrenches, which are formed from the geometric feature variation sensitivity matrix. It then follows the steps outlined in Figure 4.2, by calculating the intermediate joint twists, the intermediate part/loop twists, and the resultant wrench matrices. The screw theory method does not start with the joint wrenches for each mating condition. It solves for the resultant wrench matrix by starting from the intermediate joint twists. Using the reciprocal relationship that the twist space and the wrench space have, the method in [Adams 1998] uses the joint twists, solved for in the underconstraint method, and uses them to form the intermediate part/loop twists. This difference in step sequence is outlined in Figure 5.3, shown below.

![Figure 5.3 - Comparison of methods for overconstraint solution](image)

By not beginning from the joint wrenches, the resultant wrench of the screw theory method differs from the one produced by the VCAA method. The VCAA joint wrenches are shown in equations (4.49) through (4.53) and will not be reproduced here. The resultant wrench from the VCAA method is shown below. It shows the 2-D plane constraint as well as the two line overconstraint illustrated in Figure 4.10.
\[
W_{\text{Resultant-VCAA}} = \begin{bmatrix}
-0.2731 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
\end{bmatrix}
\] (5.13)

The method in [Adams 1998] forms the resultant wrench by treating the original joint twists as the intermediate joint twists, listed in equations (5.6) through (5.10), and continuing from there. By following the outlined steps, the resultant wrench for the screw theory-based method is shown in equation (5.14).

\[
W_{\text{Resultant-Screw Based}} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
\end{bmatrix}
\] (5.14)

The difference between the two cases is a simple one, but quite important. The screw theory based method failed to detect the two line overconstraint problem demonstrated by Figure 4.10. The only overconstraints identified were the 2-D constraints that keep the assembly in plane. The coupled X- and Y-axis constraints only show up using the VCAA method. This overconstraint is crucial for the designer to correctly account for any constraint problems. Because the overconstraint method in [Adams 1998] begins from joint twists, it will not work in every situation. In order to correctly identify overconstraints, the joint wrenches must be the starting point. The VCAA method uses the geometric feature sensitivity matrix to form joint wrenches and therefore will correctly identify the overconstraints that the screw theory based method overlooks.

This difference is an advantage that the VCAA method has over the screw based method. By using the theory presented in [Adams 1998], it was possible to augment and improve the method. The VCAA method has all the advantages that the screw theory
based method has. It has a mathematical basis, and it requires simple input information. Also, due to the origin of the joint twists and wrenches, it also accounts for overconstraints that may be missed by other methods. The main advantage to the VCAA method is that all of the information needed comes as a natural byproduct of a tolerance analysis. This allows the designer to perform a variation analysis and simultaneously perform a constraint analysis using a common assembly model. Included among its disadvantages is the fact that the VCAA is overly cautious in searching for overconstraints. It will identify overconstraints that may not be assembly problems, but will provide a warning flag of places in the assembly for designers to check.

Overall, because of its advantages and ease of use, the VCAA method is superior to the other methods presented in this thesis. It contains a great amount of DOF information and shows over- and underconstraints along with providing the tolerance sensitivities.

5.2 Results of the VCAA Method

The VCAA method establishes a unique result. A key issue that was discovered by this research was the connection between the tolerance sensitivities, derived using the GCM, and screw theory. It was shown in the development of the VCAA method that the connection between screw theory and tolerance analysis arose from the variation-velocity analogy as well as the variation-force relationship. A result of this research, and the comparisons performed, is that there is a deeper correlation between the GCM and screw theory. The analogies used to identify the correlation between the tolerance sensitivities and screw theory are actually effects related to the major source of correlation. The sensitivities are determined by the Global Coordinate Method and the connection to screw theory is found within the GCM equations and derivation.

The GCM calculates the sensitivities by starting with the 3-D vector expression and its derivative. These equations, presented in Chapter 2 as (2.29) and (2.30), are the
basis for how the GCM calculates screw matrices. The GCM is derived through the differentiation of (5.15) where all the variables are functions of the arbitrary variable $u$. The result is reprinted below as equation (5.16).

\[
V = X\hat{i} + Y\hat{j} + Z\hat{k}
\]  
(5.15)

\[
\frac{dV}{du} = \frac{dX}{du} \hat{i} + \frac{dY}{du} \hat{j} + \frac{dZ}{du} \hat{k} + \{\hat{\omega} \times V\}
\]  
(5.16)

Where

\[
\{\hat{\omega} \times V\} = \left(\omega_z Z - \omega_y Y\right)\hat{i} + \left(\omega_x X - \omega_z Z\right)\hat{j} + \left(\omega_y Y - \omega_x Z\right)\hat{k}
\]  
(5.17)

and $\hat{\omega} = (\omega_x, \omega_y, \omega_z)$ are the direction cosine angles of the local axis of rotation to the origin. The derivatives are calculated using the global coordinates and the direction cosines of the local joint axes, using the equations shown in Table 2.2. The derivatives are used to form the $[B]$ and $[F]$ sensitivity matrices. The derivatives with respect to translational and rotational variables give the twist and wrench information directly.

This result stems from the treatment of variables in the GCM, the derivation of which is shown in detail in Appendix A. The twist matrix equations, shown in the figures in Ch. 3, are solved as a combination of linear and angular velocity. The GCM solves the 3-D vector expression (position) with respect to translation and rotation, which represent the linear and angular velocity sensitivities, respectively. The correlation arises from the fact that the screw matrix equations found in [Adams 1998] are solving for the same fundamental relationships as the GCM. They are solving for the change in the feature or vector with respect to translation or rotation. Thus, the GCM is in actuality yielding screw information. Equation (5.16) is the basis of the GCM and is also a screw expression with one difference being that the screw theory-based matrices model the assembles using the homogeneous transformation, while the GCM is based on a simple vector model. The screw is a rotation and a translation along the screw rotation axis. Equation (5.16) is showing a rotation and a translation along the axis of rotation. The
fact that the GCM is giving screw information leads to the results found in the VCAA method.

The translational derivatives will show the same information as the linear velocity twists. The rotational derivatives will also give the same information as the angular velocity twists. This is all due to the derivative equations arising from (5.16) and (5.17). Any change along a length variable will be accounted for in the first three terms of (5.16). Any rotational variable change will be accounted for by the cross-product term in equation (5.16). A similar process can be applied to show how the GCM derives wrench matrices for constrained DOFs.

The variation-velocity analogy and the variation-force relationship are still valid means to show how the DLM tolerance matrices yield screw matrices, but the underlying theory is based in the GCM. The analogies held the first correlation between screw theory and variation analysis. The deeper foundation is contained in the GCM, from whence the sensitivities are derived. Further research on the exact reason for this correlation is merited.
CHAPTER 6. RECOMMENDATIONS AND CONCLUSIONS

The purpose of this thesis has been to find a connection between the Direct Linearization Method for tolerance analysis and degree of freedom constraint analysis. One goal of assembly design is to achieve exactly constrained assemblies, which avoid interference fits from overconstraints and indefinite located parts from underconstraints. Through the VCAA under- and overconstrained parts can be identified and eliminated, leaving the designer with an optimal assembly. This chapter will outline the conclusions that can be drawn from the VCAA method. It will also outline recommendations for future work in this area.

6.1 Contributions of the VCAA Method

The Variation-based Constraint Analysis of Assemblies method is a detailed, screw theory-based system for correctly identifying under- and overconstraints. The major contribution of VCAA is the source of the constraint information. The VCAA uses the information contained in DLM tolerance vector loops to solve for the constraints. Thus, a designer can perform a variation analysis and also perform a constraint analysis using the same model and information. The VCAA allows a constraint analysis to be done simultaneously with a DLM tolerance analysis.

Also, the VCAA method is built upon the work done by [Adams 1998] and adds to the method by identifying overconstraints that were missed previous. By starting with the joint wrenches, it becomes possible to solve for the overconstraint information directly and not overlook any overconstraints. The VCAA uses the screw theory basics
summarized in [Adams 1998], but it augments and furthers the method by using the
sensitivity matrices to form both joint twists and joint wrenches. With this, the complete
set of over- and underconstraints can be determined.

The VCAA method also presents the correlation between the Global Coordinate
Method and screw theory. This correlation was discovered through the variation-velocity
analogy and a variation-force relationship. However upon deeper research, it was shown
that the GCM is the mathematical foundation that gives the same basic results as screw
theory. This connection is a significant contribution of this research, in that it means the
sensitivity derivatives calculated by the GCM are equivalent to screw matrix entries.
This has applications outside of constraint analysis and may be used in other fields of
research.

6.2 Conclusions of the VCAA Method

The Variation-based Constraint Analysis of Assemblies method is based upon the
DLM tolerance analysis and the variation sensitivities, calculated through the Global
Coordinate Method. Using the connection between the VCAA method and screw theory
it is possible to form twist- and wrenchmatrices from the sensitivity matrices. Then,
using the screw theory constraint analysis methods presented in [Adams 1998] and
[Adams and Whitney 2001], it is possible to analyze the underconstraints and
overconstraints of parts within assemblies.

The initial connection between VCAA and screw theory appeared in the variation-
velocity analogy, from the TAKS method, as well as the variation-force relationship,
based on mobile and constrained DOFs. This connection was then shown to be a result of
the relationship between screw theory and the GCM, with the velocity and force results
being unexpected effects. The fact that the GCM produces screw matrix information is
the key to the effectiveness of the VCAA method. Because of this connection, the
tolerance sensitivity matrices can be manipulated to form twists and wrenches for use in
constraint analysis.

Thus, the VCAA method takes the twist and wrench information contained in the sensitivity matrices, provided by the GCM, and analyzes parts of assemblies for constraint information. The VCAA method employs the operations and steps presented in [Adams 1998] for dealing with the screw theory and locating the constraint conditions. A contribution from the VCAA method that augmented the work done in [Adams 1998] showed up in the overconstraint cases. The VCAA method forms the initial joint wrenches, from the tolerance analysis, needed for an accurate representation of the overconstraints.

The VCAA method successfully analyzes constraints of assemblies, while a DLM tolerance analysis is performed simultaneously. This method accurately establishes the connection between tolerance analysis and constraint analysis. It allows designers flexibility by producing variation and constraint data, which they can use to account for the assembly variation and the joint mating conditions to improve their designs.

6.3 Recommendations for Future Work

Further work that could be done in this area involve different aspects of VCAA. On the tolerance analysis side, the manipulation of the variation sensitivity matrices has potential for further study. Underconstraint information arises from the dependent variable matrix, [B]. Likewise, overconstraint information comes from the geometric variation matrix, [F]. Currently, the VCAA method takes the columns of these matrices to form twists and wrenches. It may be possible to gather the DOF information directly from these matrices without separating them into joint twists or joint wrenches. This would involve a direct linear algebraic manipulation and analysis of the matrices. This information may be contained in the redundant or dependent rows. Also, a relationship between [B] and [F] may exist that is similar to the twistspace and the wrenchspace. As shown in equation (2.10), the twistspace and wrenchspace are related through the

153
nullspace. There may be an analogous relationship between $[B]$ and $[F]$. A successful analysis of these two matrices may eliminate the need to use screw theory methods.

A few parts of the VCAA method could be studied further. The overconstraints are identified from the subsets of the joint wrenches. As shown in Chapter 4, good subsets for analysis include parts and loops (see Figure 4.2). Future work could discover why the vector loops make good subsets for the overconstraint analysis. Also, the open loop variation matrices, $[C]$ and $[G]$ were not accounted for in the scope of this thesis. The open loop matrices may contain DOF information and could be used in a similar analysis. The exact mathematical reasons for why the GCM and screw theory are interlinked could be explored further. The relationship is clearly there, but the rudimentary treatment of the subject in this thesis does not explore the entire field. Also, the VCAA method could also be automated into a tolerance analysis software package. The necessary information is contained in the sensitivity matrices and the VCAA method could be programmed to interact with variation analysis software.

The screw theory used in the VCAA method has a few areas that could contain future work. The overconstraint solutions are overly cautious and often present overconstraints that are not problems to the design. Further research could involve algebraically finding the cases where redundant wrenches do not truly over constrain the parts within the assembly. Linear algebra manipulation of the joint wrenches and intermediate joint twists may identify the cause of the cautious results and help to more accurately pinpoint overconstraints. Redundant wrenches may have a pattern in the redundancies that may allow for identification and analysis of true overconstraints. These patterns might appear if screw theory rules were merged with the Constraint Pattern Analysis rules found in [Blanding 1999]. Merging the two methods may remove the overconstraint problems.

Finally, assembly order of assemblies was not addressed in this thesis. Analyzing the parts of an assembly as they are placed in order may yield a more accurate view of the
constraint situation for each part. This may be done by looking at the joints of assembled parts. As new parts are assembled, the subsequent joints do not serve to locate the previously assembled parts and should be omitted when analyzing the constraint situation of the part. Also, defining terms such as degree of motion (DOM) and degree of constraint (DOC) and then finding their places in the screw theory matrices may also simplify the VCAA method and constraint analysis process.
APPENDIX A. DERIVATION OF GLOBAL COORDINATE METHOD

The following derivation of the Global Coordinate Method is found in "Global Coordinate Method for Determining Sensitivity in Assembly Tolerance Analysis" in Proceedings of the ASME International Mechanical Engineering Conference and Exposition, Anaheim, California, 1998 as well as in [Gao 1993].

The Global Coordinate Method (GCM) in 3-D represents assemblies by vector chains and relative rotations between the adjacent vectors. The GCM is derived by differentiating the 3-D vector expression.

\[ V = X\hat{i} + Y\hat{j} + Z\hat{k} \]  \hspace{1cm} (A.1)

Let \( X, Y, Z \), and \( \hat{i}, \hat{j}, \hat{k} \) be functions of the arbitrary variable \( u \). Differentiating (A.1) gives the following:

\[ \frac{dV}{du} = \frac{dX}{du} \hat{i} + \frac{dY}{du} \hat{j} + \frac{dZ}{du} \hat{k} + \{\hat{\omega} \times V\} \]  \hspace{1cm} (A.2)

where:

\[ \{\hat{\omega} \times V\} = \left(\omega_x Z - \omega_z Y\right) \hat{i} + \left(\omega_z X - \omega_x Z\right) \hat{j} + \left(\omega_x Y - \omega_y Z\right) \hat{k} \]  \hspace{1cm} (A.3)

and \( \hat{\omega} = (\omega_x, \omega_y, \omega_z) \) are the direction cosine angles of the local axis of rotation to the origin. The first three terms in (A.2) represent the change of length of the vector \( V \), and the last cross-product term represents the rotation of vector \( V \) about its tail.
A.1 Derivation with Respect to a Length Variable

If vector \( V \) represents a vector loop, the derivative with respect to a length variable is obtained by letting \( u = L_i \) in equation (A.3). In equation (A.2), the cross product term is due to the rotation of \( V \), and the first three terms are due to the change in its length. When the derivative is taken with respect to a length variable, the rotational term drops out giving (A.4).

\[
\frac{\partial V}{\partial L_i} = \lim_{\delta L_i \to 0} \left( \frac{\partial X}{\partial L_i} \hat{i} + \frac{\partial Y}{\partial L_i} \hat{j} + \frac{\partial Z}{\partial L_i} \hat{k} \right) \tag{A.4}
\]

The terms \( \frac{\partial X}{\partial L_i}, \frac{\partial Y}{\partial L_i}, \frac{\partial Z}{\partial L_i} \) do not change for any variation \( \delta L \), along the vector \( L_i \).

The derivatives of the assembly function are therefore constant terms and are equal to the direction cosines of the vector, measured in the global coordinate system. These derivatives are shown in Table A.1 where \( \alpha, \beta, \gamma \) are the direction cosine angles of vector \( L_i \) in the global \( X, Y, \) and \( Z \) directions. Also, the terms \( H_x, H_y, \) and \( H_z \) represent the scalar sum of the translations in the global \( x, y, \) and \( z \) directions, with \( H_{\theta_x}, H_{\theta_y}, \) and \( H_{\theta_z} \) representing the sum of the global \( x, y, \) and \( z \) rotations.

<table>
<thead>
<tr>
<th>Table A.1 - Derivatives with Respect to a Length Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Translational Equations</strong></td>
</tr>
<tr>
<td>( \frac{\partial H_x}{\partial L_i} = \cos \alpha )</td>
</tr>
<tr>
<td>( \frac{\partial H_y}{\partial L_i} = \cos \beta )</td>
</tr>
<tr>
<td>( \frac{\partial H_z}{\partial L_i} = \cos \gamma )</td>
</tr>
</tbody>
</table>
A.2 Derivative with Respect to a Rotation Variable

Similar to the length variable, obtaining the derivative of (A.2) with respect to a rotation variable, let \( u = \phi \), where \( \phi \) is a rotation about one of the axes of the local joint coordinate axes. The only term of (A.2) that is related to rotations is the cross product term. Equation (A.3) gives the derivatives of the assembly functions with respect to the angular variable, where the rotation is applied at the joint, or at the tail of vector \( V \), and the variation is measured in the local joint axes. The result of this measurement is shown in Figure A.1. If the variations are to be referenced from the global coordinate frame and the rotation would be applied to the tip of vector \( V \), the result would be that the signs in equation (A.3) need to be reversed, yielding (A.5). The final result using the global coordinates is shown in Figure A.2.

\[
\frac{\mathcal{N}}{\partial \phi} = (\omega_y Y - \omega_z Z) \hat{i} + (\omega_z Z - \omega_x X) \hat{j} + (\omega_x X - \omega_y Y) \hat{k}
\]  

(A.5)

\[
\Delta X = -V \delta \beta \sin(\theta_v) = -Y \delta \beta
\]

\[
\frac{\partial X}{\partial \beta} = -Y
\]

\[
\Delta Y = V \delta \beta \cos(\theta_v) = X \delta \beta
\]

\[
\frac{\partial Y}{\partial \beta} = X
\]
Figure A.1 - Variation measured at the local joint

Figure A.2 - Variation measured at global origin

The derivatives for the translational constraint equations in terms of the global coordinates, $X$, $Y$, and $Z$ of joint $i$ and the direction cosines of the rotation are easily derived from (A.5). The $\omega$ terms are the direction cosines angles of the axes of rotation and the derivatives are shown in Table A.2.

<table>
<thead>
<tr>
<th>Translational Equations</th>
<th>Rotational Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{\partial H_x}{\partial \phi_i} = \omega_Y - \omega_Z$</td>
<td>$\frac{\partial H_x}{\partial \phi_i} = \omega_x$</td>
</tr>
<tr>
<td>$\frac{\partial H_y}{\partial \phi_i} = \omega_Z - \omega_X$</td>
<td>$\frac{\partial H_y}{\partial \phi_i} = \omega_y$</td>
</tr>
<tr>
<td>$\frac{\partial H_z}{\partial \phi_i} = \omega_Y - \omega_Z$</td>
<td>$\frac{\partial H_z}{\partial \phi_i} = \omega_z$</td>
</tr>
</tbody>
</table>

Table A.2 - Derivatives with Respect to a Rotation Variable

A.3 Geometric Interpretation of the Derivatives

The geometric interpretation of the derivatives illustrates the process further. This section can be seen in its entirety in [Gao 1993]. Figure A.3 shows a 3-D vector loop with a small perturbation $\delta L_j$ along the vector $L_j$. The terms $\Delta X$, $\Delta Y$, and $\Delta Z$ are the resultant variations at the end point of the last vector. When these variations are divided by $\delta L_j$, the results are the derivatives with respect to $L_j$. So the derivatives with respect to a translational variable $\delta L_i$ can be calculated simply, as shown in Table A.3. Since Figure A.1 represents a partial derivative, only one variable is perturbed, while the rest add tip to tail unchanged. The result is all the vectors after $L_j$ in the loop translate as a rigid body, so the loop closure error is equal to $\delta L_j$ in magnitude and direction.
Table A.3 - Derivatives with Respect to a Length Variable

<table>
<thead>
<tr>
<th>Translational Equations</th>
<th>Rotational Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{\partial H_x}{\partial l_i} \approx \frac{\Delta X}{\Delta l_i} = \cos \alpha$</td>
<td>$\frac{\partial H_{\theta z}}{\partial l_i} = 0$</td>
</tr>
<tr>
<td>$\frac{\partial H_y}{\partial l_i} \approx \frac{\Delta Y}{\Delta l_i} = \cos \beta$</td>
<td>$\frac{\partial H_{\theta y}}{\partial l_i} = 0$</td>
</tr>
<tr>
<td>$\frac{\partial H_z}{\partial l_i} \approx \frac{\Delta Z}{\Delta l_i} = \cos \gamma$</td>
<td>$\frac{\partial H_{\theta x}}{\partial l_i} = 0$</td>
</tr>
</tbody>
</table>

For a rotational variation, the derivatives are harder to illustrate, but the method for getting the derivatives is similar to translational. Figure A.4 shows a 3-D vector loop with an angle variation around its local z axis. This variation, at Joint 3, will produce translational and rotational variations at the global coordinate origin. These are shown as $\Delta X$, $\Delta Y$, $\Delta Z$ for the translational and $\Delta \Theta_x$, $\Delta \Theta_y$, $\Delta \Theta_z$ for the rotational variation. It is simple to get the resultant angle variations at the global coordinate system if a unit vector, $\omega$, is placed in the global X, Y, and Z directions. This vector represents the rotational variation, whose components in X, Y, and Z are $\omega_x$, $\omega_y$, and $\omega_z$, respectively. Therefore,
\[ \Delta \Theta_x = \omega_x \delta \Theta_{3z} \]
\[ \Delta \Theta_y = \omega_y \delta \Theta_{3z} \]
\[ \Delta \Theta_z = \omega_z \delta \Theta_{3z} \]  
(A.6)

**Figure A.4 - A 3-D vector loop with angle perturbation \( \delta \Theta_{3z} \)**

If the angle variable \( \delta \Theta_{3z} \) is considered as a vector, \( \omega_x, \omega_y, \) and \( \omega_z \) become vector components and have the same meanings as the direction cosines that were found in the translational variable case. So solving for the \( \Delta X, \Delta Y, \Delta Z \) variation due to \( \delta \Theta_{3z} \) is all that remains.

Figure A.5 shows how \( \Delta X \) is calculated. The figure shows that only rotations about \( Y_1 \) and \( Z_1 \) affect \( \Delta X \). Thus:

\[ \Delta X = Y \omega_z \delta \Theta_{3z} - Z \omega_y \delta \Theta_{3z} \]  
(A.7)

In a similar fashion, \( \Delta Y \) and \( \Delta Z \) can be found:

\[ \Delta Y = Z \omega_x \delta \Theta_{3z} - X \omega_z \delta \Theta_{3z} \]  
(A.8)

\[ \Delta Z = X \omega_y \delta \Theta_{3z} - Y \omega_z \delta \Theta_{3z} \]  
(A.9)
Figure A.5 - Components of $\Delta X$ caused by $\delta \theta_{32}$

Thus, the generalized derivatives can be expressed as shown in Table A.4.

Table A.4 - Derivatives with Respect to a Rotation Variable

<table>
<thead>
<tr>
<th>Translational Equations</th>
<th>Rotational Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{\partial H_x}{\partial \phi_i} \approx \frac{\Delta X}{\delta \phi_i} = \omega_y Y - \omega_z Z$</td>
<td>$\frac{\partial H_{3x}}{\partial \phi_i} \approx \frac{\Delta \Theta_x}{\delta \phi_i} = \omega_z$</td>
</tr>
<tr>
<td>$\frac{\partial H_y}{\partial \phi_i} \approx \frac{\Delta Y}{\delta \phi_i} = \omega_z Z - \omega_x X$</td>
<td>$\frac{\partial H_{3y}}{\partial \phi_i} \approx \frac{\Delta \Theta_y}{\delta \phi_i} = \omega_y$</td>
</tr>
<tr>
<td>$\frac{\partial H_z}{\partial \phi_i} \approx \frac{\Delta Z}{\delta \phi_i} = \omega_x X - \omega_y Y$</td>
<td>$\frac{\partial H_{3z}}{\partial \phi_i} \approx \frac{\Delta \Theta_z}{\delta \phi_i} = \omega_z$</td>
</tr>
</tbody>
</table>
APPENDIX B. EXPLANATION OF 3-D POINT ALGORITHM

The point algorithm described in [Adams 1998] is used to locate a screw axis. This is related to the velocity relationship found in twists. Linear velocity \( v \) is calculated as a cross-product of the angular velocity \( \omega \) and the radius \( r \), as shown in Figure B.1. The linear and angular velocities are known and the point algorithm finds the vector \( r \), which is the location of the point, \( p \), on the screw axis. The solution found in [Adams 1998] will yield a point on the axis. In 3-D, rotational resultant twistmatrices give more complex coordinates, which makes the location of the screw axis harder to calculate.

![Figure B.1 - Angular and linear velocity on a rigid body](image)

The point3d algorithm was devised to take the 3-D effects into account, as seen in the slider crank assembly analysis in section 4.3.1. It represents a similar “inverse cross-
product" solution. A 3-D rotational resultant twistmatrix gives three angular velocity values in the first triplet and three numbers in the second triplet. These three numbers in the second triplet are related by the angular velocities and the rotational cross-product equations derived in Appendix A. Equation (B.1) shows the rotational derivative equation from the GCM and Table A.2 shows the translational derivative equations with respect to a rotation variable. The last three numbers in a 3-D rotational resultant twist are showing these translational partial derivatives. The goal of the point3d algorithm is to reverse solve for the $X$, $Y$, and $Z$ global coordinates from the values given in the resultant twist. The rotational resultant twist values and relationship equations are shown in (B.2) through (B.8).

$$\frac{\partial \mathbf{N}}{\partial \phi_i} = (\omega_x Y - \omega_y Z)\hat{i} + (\omega_x Z - \omega_z X)\hat{j} + (\omega_y X - \omega_z Y)\hat{k} \quad (B.1)$$

$$\mathbf{T}_{\text{Resultant}} = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \quad (B.2)$$

where,

$$a = \omega_x \quad (B.3)$$
$$b = \omega_y \quad (B.4)$$
$$c = \omega_z \quad (B.5)$$
$$d = \omega_x Y - \omega_y Z \quad (B.6)$$
$$e = \omega_x Z - \omega_z X \quad (B.7)$$
$$f = \omega_y X - \omega_z Y \quad (B.8)$$

The equations (B.6) through (B.8) contain three equations with only three unknowns, but they are not independent equations. Because of that dependence, it is impossible to solve for the unknown $X$, $Y$, and $Z$ global coordinates without arbitrarily choosing one and solving the other two from that one. Thus the need for the point3d algorithm and its inputs is established.
Graphically, the problem can be looked at in another way. When the elements of the linear and angular velocity are known, the radius becomes the variable. If the cross product equation is expanded, it yields three equations and three unknowns (the global coordinates of the point p), but the three equations are not linearly independent. Therefore, one of the unknown variables must be arbitrarily chosen to solve for the other two. Figure B.2 shows, if the velocities are known, the radius r, can be any one of infinite i number of vector choices.

![Diagram](image)

**Figure B.2 - Infinite radius vectors as a solution of the inverse cross product problem**

Therefore, the point3d algorithm takes the locations of the joints and uses one of the coordinates to solve for the other two. Thus, it solves for one of the possible radii by arbitrarily using one of the joint coordinates to solve for the other two. The MATLAB® file used to execute this operation is shown below. It requires the twist vector as well as the global location of the joint.

```matlab
File: point3d.m
Purpose: Point Algorithm from Konkar
Author: Danny Smith
Date: 11 January 2001
Revised: 7 March 2001
Notes: I adjusted this to include the effects of the 'inverse cross product'
```
% I am passing the twist vector(MS) and the location vector(ES)
function M = point3d(MS,ES)

%MS = [1 2 0 4 5 6] %For testing purposes
%ES = [3 4 5]

[m,n] = size(MS);
[u,v] = size(ES);

if m == 1
    if n == 6
        a = MS(1,1);
b = MS(1,2);
c = MS(1,3);
d = MS(1,4);
e = MS(1,5);
f = MS(1,6);

        if u == 1
            if v == 3
                x = -ES(1,1);
y = -ES(1,2);
z = -ES(1,3);

                if a == 0 | b == 0 | c == 0

                    if a == 0
                        if b == 0 & c == 0
                            %Stuff for a and b and c = 0
                            M = [0 0 0];
                            disp('Pure Translation')
                        elseif b == 0
                            %Stuff for a and b = 0
                            M = [-e/c d/c 0];
                        elseif c == 0
                            %Stuff for a and c = 0
                            M = [-d/b f/b 0];
                        else
                            %Stuff for a = 0 only
                            M = [-e/c y (d + c*y)/b]; % for one x value
                            %M = [-f/b y d+c*y/b]; % for the second x value
                        end
                    elseif b == 0
                        if c == 0
                            %Stuff for b and c = 0
                            M = [0 -f/a e/a];
                        else

168
\%
%Stuff for b = 0 only
M = -(e + a*z)/c - d/c z; \% for one y value
%M = -(e + a*z)/c e/a z; \% for the second y value
end

else
%Stuff for c = 0 only
M = -[x (f + b*x)/a d/b]; \% for one z value
%M = -[x (f + b*x)/a -e/a]; \% for the second z value
end

else
%Stuff for a, b, and c holding some value
M = -(a*y/b) + ((d*a)/(c*b)) + (e/c) y ((c*y) + d)/b ];
end

else
disp('Wrong Location Column Dimensions')
M = [0 0 0];
end
else
disp('Wrong Location Row Dimensions')
M = [0 0 0];
end
else
disp('Wrong Twist Column Dimensions')
M = [0 0 0];
end
else
disp('Wrong Twist Row Dimensions')
M = [0 0 0];
end

%M
APPENDIX C. MATLAB® FILES USED IN VCAA

There are many MATLAB® files (MATLAB® is a trademark of The MathWorks, Inc.) that were used in the case studies contained in Chapter 4. The text of these is reprinted here for use in verifying the results. Included in this appendix are the following example files:

- oneway1.m - One-Way Clutch Assembly
- beams1a.m - Two Cylinder Slider Plate Assembly
- beams1b.m - Three Cylinder Slider Plate Assembly
- blocks.m - Stacked Blocks Assembly
- slider.m - Crank Slider Assembly
- swash.m - Swash Plate Assembly

Also included are the functions and screw theory files needed to execute the case study files. They include the following:

- point.m - Konkar’s point algorithm
- recip.m - Reciprocal operation function
- flip.m - Switches the triplets of the twists
- twist2d.m - Interprets 2-D resultant twists
- dowrench.m - Interprets resultant wrenches
- twist3d.m - Interprets 3-D resultant twists

The files can be saved and executed in MATLAB® for verification and use in other case studies.
A = [0 0.1222 -0.1222; 1 1.9925 -0.9925; 0 0 0; 0 0 0; 0 0 0; 0 0 0];
B = [1 -39.075 0; 0 4.8105 0; 0 0 0; 0 0 0; 0 0 0; 0 -1 -1];
F = [0 0.12187 0 -1; 1 0.9925 -1 0; 0 0 0; 0 0 0; 0 0 0; 0 0 0];

% Motion Analysis
AU = flip(A');
TwistA = AU;
%twist2d(TwistA)
WrenchA = recip(AU);

TU = flip(B');
TwistB = TU
%twist2d(TwistB)
T1 = [TwistB(2,:)] % Twist of parallel cylinders
T1b = [TwistB(3,:)] % Second Twist of parallel cylinders
T2 = [TwistB(1,:); TwistB(2,:)] % Twist of cylindrical slider
T3 = [TwistB(3,:)] % Twist of revolute joint

W1 = recip(T1b)
W2 = recip(T2)
W3 = recip(T3)

W = [W1,W2] % Experiment for underconstraints
%W = [W1;W3]
%W = [W2;W3]

Twist = recip(W)
twist2d(Twist)
rTwist = ref(Twist);

%Constraint Analysis
T = [T1; T2; T3];
%T = [T1; T2];
%T = [T2; T3];
%T = [T1; T3];
WrenchB = recip(T)
rWrench = ref(WrenchB);

%WrenchB = recip(TU)
WrenchB2 = recip([TU(2,:);TU(1,:)]); %Alternate the twists
%downrench(WrenchB)

%F-Matrix Constraint Analysis
WrenchF = F';
%downrench(WrenchF)
W1 = WrenchF(1,:)
W2 = WrenchF(2,:)
W3 = [WrenchF(3,:); WrenchF(4,:)]

Tw1 = recip(WrenchF(1,:))
Tw2 = recip(WrenchF(2,:))
Tw3 = recip([WrenchF(3,:); WrenchF(4,:)])

TwistF = [Tw1; Tw2; Tw3]
Wrench = recip(TwistF)
downrench(Wrench)

%rWrenchF = ref(WrenchF);
nW1 = [W1;
0 0 1 0 0 0;
0 0 0 1 0 0;
0 0 0 0 1 0] %Wrench for cylinder slider
nW2 = [W2;
0 0 1 0 0 0;
0 0 0 1 0 0;
0 0 0 0 1 0] %Wrench for parallel cylinder
nW3 = [W3;
0 0 1 0 0 0;
0 0 0 1 0 0;
0 0 0 0 1 0] %Wrench for revolute joint

Tnw1 = recip(nW1)
Tnw2 = recip(nW2)
Tnw3 = recip(nW3)

nTwistF = [Tnw2; Tnw3]
nWrench = recip(nTwistF)
downrench(nWrench)
clear all;

A = [1 0 0 -1;
    0 1 -1 0;
    0 0 0 0;
    0 0 0 0;
    0 0 0 0];

B = [-1 1 0 0;
    0 0 -1.5 5.5;
    0 0 0 0;
    0 0 0 0;
    0 0 1 -1];

F = [0 0;
     1 -1;
     0 0;
     0 0;
     0 0;
     0 0];

% Motion Analysis

TwistB = flip(B);

% 2-D joints
T1 = [TwistB(2,:);TwistB(1,:)];
T2 = [TwistB(3,:);TwistB(1,:)];

aT1 = [0 0 1 0 0 0;TwistB(1,:)]; %Joint Twists
aT2 = [0 0 1 0 0 0;TwistB(1,:)];

Wt1 = recip(T1);
Wt2 = recip(T2);
Wt = [Wt1;Wt2];
Twist = recip(Wt)
twist2d(Twist)

% Test of starting with the wrenches
%Tt1 = recip(Wt1);
% Tt2 = recip(Wt2);
% Tt = [Tt1; Tt2]
% Wrencht = recip(Tt)

% B-Matrix and Twist Constraint Analysis
T = [T1; T2];
aT = [aT1; aT2];

WrenchB = recip(T)
aWrenchB = recip(aT)
%dowrench(WrenchB)

% F-Matrix and Wrench Constraint Analysis
WrenchF = F

% 2-D joints
W1 = [WrenchF(1,:)];
W2 = [WrenchF(2,:)];
W3 = [-1 0 0 0 0];  % Include a side joint
W4 = [1 0 0 0 0];  % Include another side joint

Tw1 = recip(W1);
Tw2 = recip(W2);
Tw3 = recip(W3);
Tw4 = recip(W4);

% Twloop1 = [Tw4; Tw1];
Twloop1 = [Tw1; Tw2; Tw3; Tw4]
% Twloop2 = [Tw4; Tw2];
% Twloop3 = [Tw4; Tw3];
Wrench1 = recip(Twloop1)
% Wrench2 = recip(Twloop2)
% Wrench3 = recip(Twloop3)
% Wrench = [Wrench1; Wrench2; Wrench3]
Wrench = [Wrench1]

W = [W1; W2; W3];
TwistF = recip(W);
nWrenchF = recip(TwistF)
dowrench(nWrenchF)
clear all;
A1 = [1 0 0 -1;
     0 1 -1 0;
     0 0 0 0
     0 0 0 0
     0 0 0 0
     0 0 0 0];

B1 = [-1 1 0 0;
     0 0 -1.5 3.5;
     0 0 0 0
     0 0 0 0
     0 0 0 0
     0 0 1 -1];

F1 = [0 0;
     1 -1;
     0 0;
     0 0;
     0 0;
     0 0];

A2 = [1 0 0 -1;
     0 1 -1 0;
     0 0 0 0
     0 0 0 0
     0 0 0 0
     0 0 0 0];

B2 = [-1 1 0 0;
     0 0 -1.5 5.5;
     0 0 0 0
     0 0 0 0
     0 0 0 0
     0 0 1 -1];

F2 = [0 0;
     1 -1;
     0 0;
     0 0;
     0 0;
     0 0];

F = [0 0 0;1 -1 0;0 0 0;0 0 0;1 0 -1;0 0 0];
%Motion Analysis

TwistB1 = flip(B1');
TwistB2 = flip(B2');

T1 = [TwistB1(2,:);TwistB1(1,:)];  %Joint Twists
T2 = [TwistB1(3,:);TwistB1(1,:)];
T3 = [TwistB2(3,:);TwistB2(1,:)];
aT1 = [0 0 1 0 0 0;TwistB1(1,:)];  %Joint Twists
aT2 = [0 0 1 0 0 0;TwistB1(1,:)];
aT3 = [0 0 1 0 0 0;TwistB2(1,:)];
Wt1 = recip(T1);
Wt2 = recip(T2);
Wt3 = recip(T3);
Wt = [Wt1;Wt2;Wt3];
Twist = recip(Wt)

twist2d(Twist)
T = [aT1;aT2;aT3];
WrenchB = recip(T)
%WrenchB1 = recip(TwistB1)
%WrenchB2 = recip(TwistB2)

%Constraint Analysis

%Loop 1
WrenchF1 = F1'
w11 = recip(WrenchF1(1,:));
w21 = recip(WrenchF1(2,:));

TwistF1 = [w11;w21];

%Loop 2
WrenchF2 = F2'
w12 = recip(WrenchF2(1,:));
w22 = recip(WrenchF2(2,:));

TwistF2 = [w12;w22];

%Combination of Loops

W1 = [WrenchF1(1,:)];
W2 = [WrenchF1(2,:)];
W3 = [WrenchF2(2,:)];
W = [W1;W2;W3];
TwistF = recip(W);

nWrenchF1 = recip(TwistF1);
nWrenchF2 = recip(TwistF2);
%TwistF = [TwistF1;TwistF2];

tWrenchF = recip(TwistF)
nWrenchF = [nWrenchF1;nWrenchF2]
dowrench(tWrenchF)
clear all;

B = [0 -0.96467 0 0 0 -18.718 10.048 0 0 0;
1 -0.26347 -1 0 0 6.62 0 0 0 0;
0 0 0 0 0 -1 1 0 0 0;
0 0 0 -0.96467 0 0 10.048 4.06 0 0;
0 0 -1 -0.26347 0 0 0 -3.905 0 0;
0 0 0 0 0 1 1 0 0 0;
0 0 0 .96467 -0.96467 0 0 -4.06 10.675 0;
0 0 0 .26347 -.26347 0 0 3.905 -28.125 0;
0 0 0 0 0 0 -1 1 1 1];

F = [1 0.26347 -1 0 0 0 0 0;
0 -0.96467 0 0 0 0 0 0;
0 0 0 0 0 0 0 0;
0 0 0 -1 -0.26347 0 0 0;
0 0 0 0 0.96467 0 0 0;
0 0 0 0 0 0 0 0;
0 0 0 0 0 0.26347 -0.26347 0 0;
0 0 0 0 -0.96467 0.96467 0 0;
0 0 0 0 0 0 0 0];

B1 = [B(1,:);
B(2,:);
0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0;
B(3,:)];

TwistB1 = flip(B1');

B2 = [B(4,:);
B(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;
B(6,:)];

TwistB2 = flip(B2');

B3 = [B(7,:);
B(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];
B(9,:); 

TwistB3 = flip(B3');

%Underconstraint Analysis 
TwistB = [TwistB1; 
TwistB2; 
TwistB3];

%Joint Twists 
t1 = [TwistB(6,:);TwistB(1,:)]
t2 = [TwistB(6,:);TwistB(2,:)]
t3 = [TwistB(7,:);TwistB(3,:)]
t4 = [TwistB(26,:);TwistB(22,:)]
t5 = [TwistB(27,:);TwistB(23,:)]

%Intermediate Joint Wrenches 
w1 = recip(t1); 
w2 = recip(t2); 
w3 = recip(t3); 
w4 = recip(t4); 
w5 = recip(t5);

%Intermediate Part wrench 
wbloc = [w1;w2];

%Resultant Twist 
tresblock = recip(wbloc) 
twist2d(tresblock)

%Adams Resultant Wrench 
Tadams = [t4;t5]; 
WrenchAdams = recip(Tadams)

%Overconstraint Analysis 
F1 = [F(1,:); 
F(2,:); 
0 0 0 0 0; 
0 0 0 0 0; 
0 0 0 0 0; 
F(3,:)];

WrenchF1 = (F1');

F2 = [F(4,:); 
F(5,:); 
0 0 0 0 0; 
0 0 0 0 0; 
0 0 0 0 0; 
F(6,:)];
WrenchF2 = (F2);

F3 = [F(7,:);
     F(8,:);
     0 0 0 0 0;
     0 0 0 0 0;
     0 0 0 0 0;
     F(9,:)];

WrenchF3 = (F3');

WrenchF = [WrenchF1;
           WrenchF2;
           WrenchF3];

%2-D Wrench
w2d = [0 0 0 1 0 0; 0 0 0 1 0; 0 0 1 0 0];

%Joint Wrenches
wf1 = [WrenchF(1,:);w2d]
wf2 = [WrenchF(2,:);w2d]
wf3 = [WrenchF(3,:);w2d]
wf4 = [WrenchF(15,:);w2d]
wf5 = [WrenchF(15,:);w2d]

%Intermediate Joint Twists
tf1 = recip(wf1);
tf2 = recip(wf2);
tf3 = recip(wf3);
tf4 = recip(wf4);
tf5 = recip(wf5);

%Intermediate Part/Loop Twist
tf = [tf4;tf5];

%Resultant Wrench
wrenchf = recip(tf)
clear all;
A = [0 -1 0 -.9239 0; 
0 0 .7071 -.3535 0; 
1 0 -.7071 -.1464 -1; 
0 0 0 0; 
0 0 0 0; 
0 0 0 0];
B = [ 1.76776715 0.857864351 -13.967746 0 -3.1114915 -2.446785 1; 
0; 
-0.9238795 0 -0.156558 0 -0.3491937 0.8720836 0;
-0.3535534 0.707106781 0.6983873 0 0.6222983 0.489357 0;
-0.1464466 0.707106781 -0.6983873 -1 0.7005773 5.262698-08 0];

F = [0 0 -0.9239 -0.3492 -0.1566 0 -0.4894 0.8721 0 0 0 0 
0; 
0.7071 0.7071 -0.3536 0.6223 0.6984 0 0.8721 0.4894 1 0 0 0 
39.7164; 
-0.7071 0.7071 -0.1465 0.7006 -0.6984 -1 0 0 0 -1 0 -39.7164 4 
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 0 0 0 0 0 0 0];

F = [0 0 -0.9239 -0.3492 -0.1566 0 -0.4894 0.8721 0 0 0 0 
0; 
0.7071 0.7071 -0.3536 0.6223 0.6984 0 0.8721 0.4894 0 0 39.7164; 
-0.7071 0.7071 -0.1465 0.7006 -0.6984 -1 0 0 0 0 -39.7164 
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 1];
\[ Y = [10.6066 10.6066 0 0 0]; \]
\[ \text{LocB} = \text{LocB}; \]
\[ \text{TwistB} = \text{flip(B')}; \]
\[ \% \text{twist3d(TwistB,LocB)} \]
\[ t1 = [\text{TwistB(1,:)}; \% \text{First spherical joint} \]
\[ \text{TwistB}(2,:); \]
\[ \text{TwistB}(3,:)] \]
\[ t2 = [\text{TwistB(4,:)}; \% \text{Second spherical joint} \]
\[ \text{TwistB}(5,:); \]
\[ \text{TwistB}(6,:)] \]
\[ t3 = [\text{TwistB(7,:)}; \% \text{Prismatic joint} \]
\[ r1 = \text{rref(t1);} \]
\[ r2 = \text{rref(t2);} \]
\[ w1 = \text{recip(t1);} \]
\[ w2 = \text{recip(t2);} \]
\[ w3 = \text{recip(t3);} \]
\[ w1 = 1000000*w1; \% \text{To compensate for accuracy problems} \]
\[ w1 = \text{round(w1);} \% \text{This rounds to the nearest 0.000001} \]
\[ w1 = 0.000001*w1 \]
\[ w2 = 1000000*w2; \]
\[ w2 = \text{round(w2);} \]
\[ w2 = 0.000001*w2 \]
\[ w3 = 1000000*w3; \]
\[ w3 = \text{round(w3);} \]
\[ w3 = 0.000001*w3 \]
\[ w = [w1;w2] \]
\[ \text{result} = \text{recip(w)} \]
\[ \text{result} = \text{rref(result)} \]
\[ \text{WrenchB} = \text{recip(TwistB)}; \]
\[ \text{RW} = \text{rref(WrenchB)}; \]
\[ \text{rRW} = \text{point(RW)}; \]
\[ \text{WrenchF} = F; \]
\[ \text{rWrenchF} = \text{rref(WrenchF)}; \]
\[ \text{wr1} = [\text{WrenchF(3,:);WrenchF(4,:);WrenchF(5,:)}] \]
\[ \text{wr2} = [\text{WrenchF(6,:);WrenchF(7,:);WrenchF(8,:)}] \]
\[ \text{wr3} = [\text{WrenchF(9,:);WrenchF(10,:);WrenchF(11,:);WrenchF(12,:);WrenchF(13,:)}] \]
\[ tr1 = \text{recip(wr1)} \]
\[ tr2 = \text{recip(wr2)} \]
\[ tr3 = \text{recip(wr3)} \]
tbase = [tr1;tr3]
tlink = [tr1;tr2]
tslink = [tr2;tr3]

wbase = recip(tbase)
wlink = recip(tlink)
tslink = recip(tslider)

tf1 = recip(WrenchF(1,:));
tf2 = recip(WrenchF(2,:));
tf3 = recip(WrenchF(3,:));
tf4 = recip(WrenchF(4,:));
tf5 = recip(WrenchF(5,:));
tf6 = recip(WrenchF(6,:));
tf7 = recip(WrenchF(7,:));
tf8 = recip(WrenchF(8,:));
tf9 = recip(WrenchF(9,:));
tf10 = recip(WrenchF(10,:));

TwistF = [tf1;tf2;tf3;tf4;tf5;tf6;tf7;tf8;tf9;tf10];

nWrenchF = recip(TwistF);

WrenchF;
rWrenchF;
TwistF;
nWrenchF;
% File: swash.m
% Purpose: Example of Swash Plate --
% Using vector loops and matrices
% Author: Danny Smith
% Date: 2 March 2001
% Modif:

%Original Spreadsheet numbers
%A = [0 -0.707106381 0 -0.707108281 0.70710621 0.707109581
0.663792144 -0.087485944 0.087485712 -0.173649331 0;
-1 0 -1 0 0 -0.344614413 -0.992316693 0.992316734
-0.98480755 1;
% 0 0.707107181 0 -0.707105281 0.707107353 0.707103981 -0.663792811
0.087485944 -0.087485712 0 0;
% 0 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0];
%B = [0 2.121318 1.024763471 3.237665295 0.08613531 -0.504040464;
% 0 5.000002034 0.77565252 -0.570892435 3.721177075 0.088874796;
% -2.121321 -1.02476492 -1.905734521 -0.56999046 -0.859095153;
% 0 0.707107 0.707107 -0.17365 0.701673 0;
% 1 0 0 -0.98481 -0.12372 0;
% 0 0.707106 0.707106 0 -0.70167 0];
%F = [0 3.5 -0.163799442 5.709457781 0.1408043 -1.024769268 -0.173649331;
% 0 0 2.346866032 -0.861539962 3.520287771 -0.775653708 -0.98480755;
% -3.5 0 -1.38218778 2.39051641 -1.686770606 1.024769268 0;
% 1 0 0.663793 -0.24368 0.663793 -0.70711 0;
% 0 0 -0.34461 -0.93874 -0.34461 -9.3E-17 0;
% 0.1 0.66379 0.243679 -0.66379 -0.70711 0];

%Updated Spreadsheet numbers
A = [0 -1 0 0 0 0.948406915 0.948405211 -0.317060813
-0.173635407 0;
-1 0 -1 0 0 -0.317055709 -0.317060806 -0.948405209 -0.9848
10005 1;
0 0 0 1 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];
B = [3 1.188978024 -0.6740641 0.984807753 0;
5 3.556519543 0.065136612 -0.173648178 0;
0 -1.259938159 0 0 -1;
0 0.948405211 0 0 0;
0 -0.317060806 0 0 0;
1 0 -1 0 0];
F = [0.948406915 -0.317060813 0 1.18895891 3.556519534 -0.317060813
-0.948406915 0 3.556519524 -1.243107213 0.173635407 -0.651180666;
-0.317055709 -0.948405209 0 3.556525933 -1.18897805 -0.948405209 0.31705
5709 0 -1.18897805 0.255373096 0.984810005 -3.693029074;
=[theta1 theta2 theta3 theta4 theta5 K];
%Global coordinates of the joints, for use in point3d()

%LocB = [0 -1.76777 2.103183 1.928873 2.015697 1.928873;
  3.5 3 1.449236 -0.03548 0.45692 -0.03548;
  0 5.303301 3.200118 3.287604 3.287604 3.287604];
%LocB = LocB';

%Motion Analysis - Underconstraints

%Column Twist Matrix
TwistB = flip(B')
%twist3d(TwistB,LocB)

%Joint Twists
T1 = [TwistB(1,:)]                   %intermediate twist for revolute1
T2 = [TwistB(2,:)]                   %intermediate twist for revolute2
T3 = [TwistB(3,:);TwistB(4,:);TwistB(5,:)] %intermediate twist for cyslider

%Intermediate Joint wrenches
w1 = recip(t1)
w2 = recip(t2)
w3 = recip(t3)

%Intermediate part wrenches
wplate = [w1;w3]
wfollower = [w1;w2]   wtcylinder = [w2;w3]

%Resultant Twist Matrix
Tplate = recip(wplate)
Tfollower = recip(wfollower)
Tcylinder = recip(wtcylinder)

%Test of Adams overconstraints
WrenchB = recip(TwistB);
RW = rref(WrenchB);

%Overconstraint DOF analysis
%Column Wrench
WrenchF = F;
rWrenchF = rref(WrenchF);

%Joint wrenches
w1 = [WrenchF(1,:); WrenchF(2,:); WrenchF(3,:); WrenchF(4,:); WrenchF(5,:)]
w2 = [WrenchF(6,:); WrenchF(7,:); WrenchF(8,:); WrenchF(9,:); WrenchF(10,:)]
w3 = [WrenchF(11,:); WrenchF(12,:)]

%Intermediate joint twists
tw1 = recip(w1)
tw2 = recip(w2)
tw3 = recip(w3)

%Intermediate subset twists
twloop = [tw1; tw2; tw3]
twplate = [tw1; tw3]
twfollower = [tw1; tw2]
twcylinder = [tw2; tw3]

%Resultant Wrenchmatrices
wloop = recip(twloop)
wplate = recip(twplate)
wfollower = recip(twfollower)
wcyliner = recip(twcylinder)
function M = point(MS)
[m,n] = size(MS);
if m == 1
    if n == 6
        a = MS(1,1);
        b = MS(1,2);
        c = MS(1,3);
        d = MS(1,4);
        e = MS(1,5);
        f = MS(1,6);
        W = [0 c -b; -c 0 a; b -a 0];
        W = [1 c -b; -c 1 a; b -a 1];
        V = [d e f];
    end
    if a == 0 | b == 0 | c == 0
        if c == 0
            if b == 0
                if a == 0
                    M = [0 0 0];
                    disp('Pure Translation')
                else
                    M = [0 -b/a e/a];
                end
            else
                M = [b/a 0 -d/b];
            end
        end
        else
            M = [-e/c d/c 0];
        end
    else
        y = -10.6066;
        M = -(a*y/b)+(d*a)/(c*b)+(e/c) y ((c*y)+d)/b ];
        M = (inv(W)*V');
    end
end

disp('Wrong Column Dimensions')
M = [0 0 0];
end
else
    disp('Wrong Row Dimensions')
    M = [0 0 0];
end
function R = recip(T)
    % takes the reciprocal of a screw matrix
    p = (null(T,'r'));  % uses reduced row form of null operator
    %p = (null(T));
    R = flip(p);
function W = flip(WU)  %FLIPs columns of WU
% col 1 becomes 4, 2 to 5, 3 to 6, 4 to 1, 5 to 2, 6 to 3

[i,j] = size(WU);

if i > 0
    if j == 6
        for m = 1:i
            for k = 1:3
                W(m,k) = WU(m,k+3);
                W(m,k+3) = WU(m,k);
            end
        end
    else
        disp ('Empty Matrix!')
        W = [0 0 0 0 0 0];
    end
end
function twist2d(M)
  [n,m] = size(M);
  q = 1;
  
  while q < n+1
    T = M(q,:);
    a = T(1,1);
    b = T(1,2);
    c = T(1,3);
    d = T(1,4);
    e = T(1,5);
    f = T(1,6);
    
    if a == 0 & b == 0 & c == 0
      if d == 0
        if e == 0 & f == 0
          disp('No Motion Allowed')
        elseif e == 0
          disp('Pure Translation in Z')
        elseif f == 0
          disp('Pure Translation in Y')
        else
          disp('Translation in Y and Z')
        end
      elseif e == 0
        if f == 0
          disp('Pure Translation in X')
        else
          disp('Translation in X and Z')
        end
      elseif f == 0
        disp('Translation in X and Y')
      else
        disp('Translation in X, Y, and Z')
      end
    end
  end
end

elseif a == 0 | b == 0 | c == 0
  if a == 0
    if b == 0 & c == 0
      %Stuff for a and b and c = 0
      %Nothing will happen, the code above takes care of it
    elseif b == 0
      %Stuff for a and b = 0
      disp('Rotation about Z located at:')
      %disp('With screw axis located in 2D at:')
      p = point(T)
      %num2str(p)
    elseif c == 0
      %Stuff for a and c = 0
      disp('Rotation about Y located at:')
      %disp('With screw axis located in 2D at:')
      p = point(T)
    else
      %Stuff for a = 0 only
      disp('Rotation about Y and Z located at:')
      %disp('With screw axis located in 2D at:')
      p = point(T)
  end

elseif b == 0
  if c == 0
    %Stuff for b and c = 0
    disp('Rotation about X located at:')
    %disp('With screw axis located in 2D at:')
    p = point(T)
  else
    %Stuff for b = 0 only
    disp('Rotation about X and Z located at:')
    %disp('With screw axis located in 2D at:')
    p = point(T)
  end

else
  %Stuff for c = 0 only
  disp('Rotation about X and Y located at:')
  %disp('With screw axis located in 2D at:')
  p = point(T)
end
  else
    %Stuff for a, b, and c holding some value
    disp('Rotation about X, Y, and Z located at:')
    %disp('With screw axis located in 2D at:')
    p = point(T)
end
disp(' ')
q = q+1;
end
disp(' ')
function twist3d(M,E) %Pass in the Twist and Locations

[n,m] = size(M);
[u,v] = size(E);

q = 1;

while q < n+1
    T = M(q,:);
    L = E(q,:);

    a = T(1,1);
    b = T(1,2);
    c = T(1,3);
    d = T(1,4);
    e = T(1,5);
    f = T(1,6);

    if a == 0 & b == 0 & c == 0
        if d == 0
            if e == 0 & f == 0
                disp('No Motion Allowed')
            elseif e == 0
                disp('Pure Translation in Z')
            elseif f == 0
                disp('Pure Translation in Y')
            else
                disp('Translation in Y and Z')
            end
        elseif e == 0
            if f == 0
                disp('Pure Translation in X')
            elseif e == 0 only
                disp('Translation in X and Z')
            end
        elseif f == 0
            disp('Translation in X and Z')
        end
    end
elseif a == 0 | b == 0 | c == 0
  if a == 0
    if b == 0 & c == 0
      %Stuff for a and b and c == 0
      %Nothing will happen, the code above takes care of it
    elseif b == 0
      %Stuff for a and b = 0
      disp('Rotation about Z located at:')
      p = point3d(T,L)
    elseif c == 0
      %Stuff for a and c = 0
      disp('Rotation about Y located at:')
      p = point3d(T,L)
    else
      %Stuff for a = 0 only
      disp('Rotation about Y and Z located at:')
      p = point3d(T,L)
  end
  elseif b == 0
    if c == 0
      %Stuff for b and c = 0
      disp('Rotation about X located:')
      p = point3d(T,L)
    else
      %Stuff for b = 0 only
      disp('Rotation about X and Z located at:')
      p = point3d(T,L)
  end
  elseif c == 0
    %Stuff for c = 0 only
    disp('Rotation about X and Y located at:')
    p = point3d(T,L)
  else
    %Stuff for a, b, and c holding some value
    disp('Rotation about X, Y, and Z located at:')
    p = point3d(T,L)
  end
  disp('')
q = q+1;
end
%disp('For Rotation Locations, use point3d.m');
disp('')
function dowrench(M)

[n,m] = size(M);

q = 1;

while q < n+1
    W = M(q,:)
    a = W(1,1);
    b = W(1,2);
    c = W(1,3);
    d = W(1,4);
    e = W(1,5);
    f = W(1,6);

    if a ~= 0 | b ~= 0 | c ~= 0 | d ~= 0 | e ~= 0 | f ~= 0
        disp('This is Overconstrained in:')
        if a ~= 0
            disp(' Translation of X')
        end
        if b ~= 0
            disp(' Translation of Y')
        end
        if c ~= 0
            disp(' Translation of Z')
        end
        if d ~= 0
            disp(' Rotation of X')
        end
        if e ~= 0
            disp(' Rotation of Y')
        end
        if f ~= 0
            disp(' Rotation of Z')
        end

    else
        disp('This is not Overconstrained')
    end

    q = q+1;
    disp('')
end
disp('')
BIBLIOGRAPHY


