A Generalized Approach to Kinematic Modeling for Tolerance Analysis of Mechanical Assemblies

ADCATS Report No. 91-3

Glen C. Larsen, graduate student
Department of Mechanical Engineering
Brigham Young University

MS Thesis sponsored by ADCATS

July 1991

ABSTRACT

Tolerance selection has a significant influence on the manufacturing cost of a mechanical assembly. Tolerance analysis tools based on a CAD assembly model can help the design engineer evaluate the consequences of tolerance specification early in the design process, when design changes are not prohibitively critical or expensive. Research presented in this thesis focuses on developing a generalized method of obtaining the additional kinematic information required for tolerance analysis. The method is used to create an assembly tolerance model based on elements the designer should be familiar with, including parts, datums, joints, and vectors which represent critical part and assembly dimensions. Dimension vectors are connected in series to form vector loops, which follow only controlled and kinematic dimensions in the model. The governing relationships for tolerance analysis are determined from scalar equations of the vector loops.

Thesis contributions include:

1. Generalized modeling approach for graphically creating a kinematic tolerance model for tolerance analysis of a mechanical assembly
2. Introduction of the concept of datum paths to locate kinematic joints relative to the datum reference frame of each part in the model
3. Complete automatic generation of the vector loops used to predict the effects of manufacturing variations on assembly performance
4. Automatic identification of dependent variables in order to minimize user interaction and ensure proper setup for analysis
5. Degree-of-freedom analysis and consistency checks to ensure solution is valid
6. Implementation of automatic loop generation using C and of the generalized tolerance modeling method on the AutoCAD system using the AutoLISP application language

A final product of this research is a complete assembly tolerance modeling scheme suitable for implementation on commercial CAD systems.
# TABLE OF CONTENTS

List of Figures .................................................................................................................. vi
List of Tables .................................................................................................................... viii
Acknowledgements .......................................................................................................... ix

1. **Introduction** ............................................................................................................... 1
   1.1 Motivation ................................................................................................................ 1
   1.2 Computer-Aided Tolerancing ................................................................................. 2
   1.3 Research Objective ................................................................................................. 3
   1.4 Research Approach ............................................................................................... 3
   1.5 Thesis Overview .................................................................................................... 3

2. **Background** ............................................................................................................... 4
   2.1 Assembly, Dimension, and Tolerance Modeling .................................................. 4
      2.1.1 Assembly Representation .............................................................................. 4
      2.1.2 Datums and Feature Variations .................................................................... 5
      2.1.3 Kinematic Chains and Vector Loops ............................................................ 6
      2.1.4 Tolerance Modelers ..................................................................................... 6
      2.1.5 Analysis Software ....................................................................................... 7
   2.2 Kinematic Modeling ............................................................................................... 8
   2.3 Computer-Aided Tolerance Selection (CATS) ................................................... 9
   2.4 Summary .............................................................................................................. 10

3. **Generalized Modeling Approach** .......................................................................... 11
   3.1 Modeling Procedure ............................................................................................ 11
      3.1.1 Parts ............................................................................................................ 12
      3.1.2 Joints ......................................................................................................... 12
      3.1.3 Vector Loops .............................................................................................. 17
      3.1.4 Feature Controls ....................................................................................... 19
   3.2 Assembly Model File ........................................................................................... 20
   3.3 Tolerance Analysis and Results .......................................................................... 20
   3.4 Summary .............................................................................................................. 22
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td>MODELING CONSIDERATIONS</td>
<td>23</td>
</tr>
<tr>
<td>4.1</td>
<td>Degree of Freedom Check</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>4.2.1 Procedure for DOF Analysis</td>
<td>25</td>
</tr>
<tr>
<td>4.2</td>
<td>Automatic Loop Generation</td>
<td>25</td>
</tr>
<tr>
<td>4.3</td>
<td>Dependent Variable Identification</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>4.3.1 Dependent Vector Lengths</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>4.3.2 Dependent Rotation Angles</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>4.3.3 Identification of 3-D Dependent Variables</td>
<td>32</td>
</tr>
<tr>
<td>4.4</td>
<td>Rotation of Local Coordinate Frame</td>
<td>32</td>
</tr>
<tr>
<td>4.5</td>
<td>Summary</td>
<td>35</td>
</tr>
<tr>
<td>5.</td>
<td>AUTOMATIC LOOP GENERATION</td>
<td>38</td>
</tr>
<tr>
<td>5.1</td>
<td>AVL Prototype and Limitations</td>
<td>38</td>
</tr>
<tr>
<td>5.2</td>
<td>Virtual Tree Representation</td>
<td>39</td>
</tr>
<tr>
<td>5.3</td>
<td>Determining Candidate Contact Joint Loops</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>5.3.1 Tree Search Example</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>5.3.2 Down-to-Leaf Procedure</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>5.3.3 Up-to-Branch Procedure</td>
<td>46</td>
</tr>
<tr>
<td>5.4</td>
<td>Forming Complete Loop Using Datum Paths</td>
<td>48</td>
</tr>
<tr>
<td>5.5</td>
<td>Candidate Loop Comparison and New Starting Joint</td>
<td>50</td>
</tr>
<tr>
<td>5.6</td>
<td>Modified AVL Algorithm</td>
<td>51</td>
</tr>
<tr>
<td>5.7</td>
<td>Summary</td>
<td>51</td>
</tr>
<tr>
<td>6.</td>
<td>AUTOCATS IMPLEMENTATION</td>
<td>52</td>
</tr>
<tr>
<td>6.1</td>
<td>Menus</td>
<td>53</td>
</tr>
<tr>
<td>6.2</td>
<td>AutoLISP</td>
<td>54</td>
</tr>
<tr>
<td>6.3</td>
<td>Data Structure</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>6.3.1 Datum Object</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>6.3.2 Joint Object</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>6.3.3 Dimension Vector Object</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>6.3.4 Loop Object</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>6.3.5 Feature Control Object</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>6.3.6 Data Structure</td>
<td>57</td>
</tr>
<tr>
<td>6.4</td>
<td>Comparison to Previous Prototype</td>
<td>58</td>
</tr>
<tr>
<td>6.5</td>
<td>Geometric Associativity</td>
<td>59</td>
</tr>
<tr>
<td>6.6</td>
<td>Summary</td>
<td>60</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Relationship between cost and tolerances</td>
<td>1</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Degrees of freedom of an edge joint</td>
<td>8</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Single degree of freedom in a pair of edge joints</td>
<td>8</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Example assembly of stacked parts</td>
<td>11</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Part names and datum reference frames</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Common 3-D joint types and their associated degrees of freedom</td>
<td>13</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Common 2-D joint types and their associated degrees of freedom</td>
<td>14</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>Sliding plane associated with assembly joint</td>
<td>15</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Joint defined in terms of a path to the DRF of each part</td>
<td>15</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>Datum path may include feature datums</td>
<td>16</td>
</tr>
<tr>
<td>Figure 3.8</td>
<td>Example assembly with all joints defined</td>
<td>17</td>
</tr>
<tr>
<td>Figure 3.9</td>
<td>Vector loop between contact joints</td>
<td>18</td>
</tr>
<tr>
<td>Figure 3.10</td>
<td>Loop expanded to include paths back to datum reference frames</td>
<td>18</td>
</tr>
<tr>
<td>Figure 3.11</td>
<td>Complete vector loop after removal of redundant paths</td>
<td>19</td>
</tr>
<tr>
<td>Figure 3.12</td>
<td>ANSI Y14.5 feature control symbols</td>
<td>20</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Network graph for the geometric stack assembly</td>
<td>26</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Connectivity matrix for the geometric stack assembly</td>
<td>27</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Dependent variables in vector loop</td>
<td>28</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>Dependent and independent rotations at a revolute joint</td>
<td>28</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>Determining vector length dependency</td>
<td>29</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>Problem cases for automatic identification of dependent lengths</td>
<td>30</td>
</tr>
<tr>
<td>Figure 4.7</td>
<td>Problem case for automatic identification of dependent angles</td>
<td>32</td>
</tr>
<tr>
<td>Figure 4.8</td>
<td>Rotation of local coordinate frame about Z-axis</td>
<td>33</td>
</tr>
<tr>
<td>Figure 4.9</td>
<td>Rotation of local coordinate frame about Y-axis</td>
<td>34</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Equivalent representation for transferring between parts at a joint</td>
<td>40</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Equivalent representation for transferring between joints on a part</td>
<td>40</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>Watt mechanism</td>
<td>41</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>Traversal showing first candidate loop</td>
<td>41</td>
</tr>
<tr>
<td>Figure 5.5</td>
<td>Traversal showing first rejected path</td>
<td>42</td>
</tr>
<tr>
<td>Figure 5.6</td>
<td>Traversal showing second candidate loop</td>
<td>42</td>
</tr>
<tr>
<td>Figure 5.7</td>
<td>Traversal showing second rejected path</td>
<td>43</td>
</tr>
<tr>
<td>Figure 5.8</td>
<td>Method of traversing tree in search of a leaf node</td>
<td>45</td>
</tr>
<tr>
<td>Figure 5.9</td>
<td>Method of traversing tree in search of a branch node</td>
<td>47</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 4.1  Representations of degree of freedom directions for kinematic joints........24
Table 4.2  2-D and 3-D dependent variables for each joint.................................32
Table 6.1  Capability of prototype versus new AutoCATS implementation...............59
1.1 Motivation

All manufactured parts are subject to dimensional variations caused by manufacturing process fluctuations. These variations often have adverse consequences. Variations in a single part can cause poor part performance. Variations in the parts of an assembly can stack together in such a way that the parts may not assemble or function properly. Thus it is important that a design engineer account for variations during the design process. Typically this is done by assigning tolerances to dimensions on each part, which represent acceptable upper and lower limits for manufacturing variations.

The proper specification of tolerances on part dimensions requires careful analysis. A designer cannot simply make all tolerances very small in order to ensure proper assembly performance because this drives up manufacturing costs. Conversely, if the designer assigns tolerances which are too large, costs also rise due to increased labor required to rework the unsatisfactory assemblies. Figure 1.1 illustrates the relationship between cost and tolerances.

![Figure 1.1 Relationship between cost and tolerances](image)

Since the cost curves represent entire assembly costs, values on the horizontal axis represent a set of tolerances for the assembly, not just a single tolerance. The curves show
that as tolerances are loosened, production cost decreases while the cost of rejects and rework increases. The total cost represents the sum of these two costs, with the minimum occurring between the two extremes. Thus the goal of the designer should be to assign an optimal set of tolerances which minimizes the total cost of the assembly, while still ensuring satisfactory performance.

1.2 Computer-Aided Tolerancing

Since tolerance selection has such a significant influence on cost, it would be helpful if the designer had analysis tools to determine which tolerances influence cost the most and to investigate the effects of tolerance changes. Obviously, if assembly performance is not sensitive to variations in a certain part dimension, it would be wise to specify the tolerance on that dimension as loose as possible, in order to minimize assembly cost. Tight tolerances should be reserved for other part dimensions which need to be small to ensure proper assembly performance. Thus there are relationships between tolerances, sensitivity of dimension variations, performance, and cost. Unfortunately, it is difficult and time-consuming to determine these relationships by hand. Therefore, computer techniques have been employed to model manufacturing variations and perform tolerance analysis.

In order to create a tolerance model for analysis, geometric information about the assembly is required. Since an assembly model has usually been created previously in a Computer-Aided-Design (CAD) database, this information is readily available. However, each part's actual position in the assembly will vary slightly from its ideal position, as manufacturing variations on individual part dimensions cause the assembly to adjust. In order to account for these adjustments, the tolerance model must also include kinematic relationships, in addition to geometric data. Kinematic information is added to the model by identifying joints at the part interfaces, the type of joint being based on the type of mating contact between parts. The joints are then connected in series to form kinematic chains or loops. The loops form the desired set of relationships which reflect how the assembly parts will adjust to small variations. Other constraints, such as form and feature controls, must also be taken into account. The procedure for obtaining this additional kinematic and feature information is called the tolerance modeling process.

A considerable portion of the designer's time is spent defining the kinematic loops required for tolerance analysis. In addition, determining the paths of the loops requires a level of skill based on experience. Thus significant efforts are being focused on generating
vector loops automatically. Complete automatic vector loop generation would not only reduce the amount of time required to model an assembly, but improve the model's accuracy as well.

1.3 Research Objective

The purpose of the research presented in this thesis is first, to introduce a general and systematic approach to the kinematic modeling of manufacturing variations in mechanical assemblies, and second, to completely automate the loop defining process, which is one of the greatest potential sources of error in creating a model for tolerance analysis.

1.4 Research Approach

The research approach is outlined as follows: 1) examine previous modelers and prototype systems to determine common and general requirements for tolerance modeling, 2) apply network graph theory to obtain a set of paths between contact joints, 3) use feature datum references to expand the contact joint paths and form complete loops which follow only controlled dimensions on the model, 4) automatically identify dependent variables in the set of vector loops, and 5) implement the general modeler using the AutoLISP application language of the AutoCAD system.

1.5 Thesis Overview

The next chapter discusses important background information regarding research related to computer-aided tolerance modeling and analysis. Chapter 3 presents a detailed approach to the tolerance modeling process. Chapter 4 explains some additional modeling procedures which are performed once the tolerance model has been created. Complete automation of kinematic chain or vector loop generation is described in Chapter 5. The implementation of the generalized modeling method with the AutoCAD system is discussed in Chapter 6. Chapter 7 introduces several example assemblies to be modeled and analyzed. Conclusions are presented in Chapter 8 along with recommendations for future research.
Chapter 2

BACKGROUND

This chapter focuses on current research in assembly, dimension, and tolerance modeling. A literature review is presented on assembly and tolerance modeling and other related issues. Methods are described which enforce a kinematic model to ensure realistic propagation of assembly variations.

2.1 Assembly, Dimension, and Tolerance Modeling

Most mechanical parts are designed and modeled individually using a CAD system. Models of several parts are then combined to form an assembly model. Although the CAD assembly model provides a graphic representation of how individual parts interface, it is inadequate for further applications [Srikanth et al., 1990; Shah et al., 1990]. Many researchers have recognized the need for a system which can model assembly information in addition to feature, dimension, and tolerance information, and several have attempted to classify the essential requirements of such a modeler.

2.1.1 Assembly Representation

Srikanth and Turner discuss an assembly hierarchy which includes assembly components or parts and the mating features between individual parts [Srikanth et al., 1990]. In addition, kinematic relationships are required which combine all mating conditions into a collective set representing the entire assembly. This is usually accomplished using a network graph where the nodes are assembly components and the edges are the mating conditions.

Srikanth further describes a unified foundation for assembly modeling which is based on a boundary representation (B-rep) model and a relative position hierarchy. Although this approach is more complex than is required for tolerance analysis alone, the method is general and should support other applications such as assembly sequence generation and dynamic and kinematic analyses.

Gossard, Zuffante and Sakurai describe a method of assembly modeling which combines the merits of Constructive Solid Geometry (CSG) and B-rep using a structure called an object graph [Gossard et al., 1988]. The object graph consists of a hierarchy of nodes and edges, forming a binary tree. Each edge points to an evaluated B-rep object, and each node is an operator which combines two edges (objects) and attaches a scalar...
parameter describing a single dimension relating the two objects. With this hierarchy in place, the powerful concept of dimension driven or variational geometry is introduced, in which component geometry may be rescaled by modifying a set of unique annotated dimensions. This modeling approach is advantageous because it allows engineers and designers to sketch parts quickly, focusing on kinematic and topological relationships, not on defining precise geometry. Exact dimension values are then refined later which automatically update the geometry. However, practical use of this method is greatly limited by the fact that very little provision is made for tolerance consideration.

Kim and Lee have developed a system which requires only the mating conditions between parts to derive an assembly model [Kim et al., 1989]. The initial conditions are specified in terms of the centerlines and planar faces of the two basic part interface states—the "against" and "fit" mating conditions. The "against" condition requires that the two planar contact faces be coplanar. The "fits" condition requires that the centerlines of the two parts be colinear. A network graph (with parts as nodes and virtual links or joints as edges) is used to separate the virtual links into groups and set up constraint equations. The groups and the original mating conditions are then used to determine individual joint type and degree of freedom information. The constraint equations are solved simultaneously and used to determine each part's position in space. Finally, individual joint coordinate systems are generated from derived joint information, thus completing the modeling process for the assembly.

In summary, assembly representation methods all have several elements in common; the critical elements being assembly parts, mating conditions between individual parts, and a set of relationships which considers the parts and joints collectively.

2.1.2 Datums and Feature Variations

Spotts stresses the importance of datums in engineering [Spotts, 1985]. He first identifies the need for a datum system consisting of three mutually orthogonal planes for rectangular parts. This coordinate system forms a datum reference frame, or a location from which all dimensions are referenced. Dimensioning procedures are slightly different for cylindrical parts. The datum system consists of a perfect cylinder butted against a perpendicular plane. Thus datum representations must provide for rectangular and cylindrical datum reference frames.
Spotts also mentions that the use of datums alone will not guarantee perfectly accurate manufactured parts. Because of manufacturing process fluctuations, geometric tolerancing must also include form and feature controls such as flatness, parallelism, angularity and perpendicularity. Shah and Miller confirm that form, orientation, position and runout variations must be controlled in addition to size variations [Shah et al., 1990].

2.1.3 Kinematic Chains and Vector Loops

All assembly modelers must have the capability of determining a set of relationships which represent how the assembly parts interact. These relationships are generally in the form of kinematic chains or loops between the joints where parts make contact and are used to obtain the governing equations for an assembly. Wang introduces a scheme of representing assemblies and discusses the application of graph theory to generate kinematic chains automatically [Wang, 1990]. He states that "automatic tolerance chain generation is the key step in any tolerance analysis at the assembly level."

Other methods of obtaining more general kinematic equations of motion for an assembly involve Lagrange multipliers and calculating the Jacobian matrix which relates the output motions to the input parameters [Fenton et al., 1989; Nikravesh et al., 1989; Wehage, 1989]. However, since velocities and accelerations are not needed for tolerance analysis, simpler methods based on graph theory are sufficient to generate the required kinematic chains.

2.1.4 Tolerance Modelers

Shah and Miller discuss some of the requirements of a tolerance modeler. As mentioned earlier, it must support all classes of geometric tolerances (a comprehensive description of form and feature controls is defined in the ANSI Y14.5 specification [ANSI, 1982]). The modeler must also support datum reference frames and datum precedence. The tolerance modeling data structure should be geometrically associated with entities in the CAD database. The modeler should allow default tolerances. Finally, the modeler should provide graphical feedback and be able to display datum and feature information.

Faux also did an intensive evaluation of the requirements for including tolerance information in a geometric modeler [Faux, 1981]. His work supports the need for a datum reference structure, a feature structure, and in addition, an assembly data structure which includes mating features and the nature of fit between assembly components.
2.1.5 Analysis Software

Turner and Gangoiti review existing commercial tolerance analysis software [Turner et al., 1991] and discuss seven software packages including Pro/Engineer, from Parametric Technology Corp; VSA, from Applied Computer Solutions; Mechanical Advantage, from Cognition Inc; Analytix, from Saltire Software; Design View, from Premise Inc; Mechanical Engineering Workbench, from Iconnex; and VALISYS, from FMC. Turner and Gangoiti state that in general, the packages are difficult to use and some are limited in applicability. The review identifies the ideal requirements for commercial assembly tolerancing software, such as working directly with a CAD system model and requiring minimal additional input. It also stresses the importance of conveying to the user the underlying assumptions and interpretations made by the analysis package. The software review is informative and presents many factors which tolerance software developers must consider.

Computer tools for tolerance analysis have been under development at Brigham Young University (BYU) since 1984. The research has focused on developing an analysis package, called Computer-Aided Tolerance Selection (CATS), which will help designers specify tolerances based on performance requirements and manufacturing considerations [Chase et al., 1987].

An assembly modeler for tolerance analysis must be able to accurately account for the following effects:

1. Rigid body translation and rotation
2. Propagation of form and feature variations
3. Propagation of kinematic adjustments

Variations in single parts may affect the position and orientation of other assembly parts in terms of translations and rotations. The effect of these transformations on consecutive stacked parts is termed rigid body motion. Due to manufacturing variations, all part surfaces and features vary from their ideal design. Thus form and feature variations must be correctly represented. The modeling of kinematic adjustments is especially critical and demands special consideration.
2.2 Kinematic Modeling

Tolerance analysis depends entirely on a correct model. Thus, the tolerance model must realistically represent how parts adjust to manufacturing variations in a mechanical assembly. Mating conditions between parts restrict each part's movement to certain directions. These directions are called kinematic degrees of freedom (DOF). For example, a 2-D edge slider joint allows a single rotation and sliding in only one direction, as shown in Figure 2.1. The tolerance model must contain all the kinematic constraints which are present in the actual assembly.

![Figure 2.1 Degrees of freedom of an edge joint](image)

Assembly adjustment directions are limited even further when several joints are considered collectively. For example, when two edge joints are considered as a pair, the common rotation is eliminated entirely, as Figure 2.2 illustrates.

![Figure 2.2 Single degree of freedom in a pair of edge joints](image)

Thus it is essential to obtain a set of relationships between joints which reflects the adjustments parts make relative to each other. These relationships are called kinematic chains or vector loops from which the governing equations for the assembly may be determined. The number of fundamental or independent vector loops (L) required to represent an assembly network [Paul, 1979] is a function of the number of joints (J) and the number of parts (P), as Equation 2.1 shows.

\[ L = J - P + 1 \]  \( (2.1) \)
It is important to mention that this equation was derived from network topology theory, not
degrees of freedom, and may impose analysis limitations.

Tolerance analysis in not the only application which requires an accurate kinematic
model for analysis. Dynamic simulation and mechanism analysis packages such as IMP
[Sheth, 1971], DRAM [Chace, 1979] and ADAMS [Kim et al., 1989] each depend on a
kinematically accurate model. Some kinematic applications use network graph theory to
automatically obtain the required set of vector loops. The difference between CATS and
other applications which require a kinematic model is the magnitude of the input motions.
The inputs to most kinematic programs are large displacements of rigid parts, while the
CATS model inputs are small manufacturing variations of part dimensions. In addition,
CATS allows numerous sources of variation to occur simultaneously and their
accumulative effects are determined statistically.

2.3 Computer-Aided Tolerance Selection (CATS)

The initial version of CATS performed analysis on a one-dimensional tolerance
stack and used text screen data entry to create the assembly model. The direct linearization
method [Marler, 1988] provided a means of analyzing two-dimensional assemblies and
much progress has been made in 3-D tolerance analysis [Robison, 1989]. The CATS
analysis software still allows model definition by text screen entry, but higher dimension
problems require much more information, thus data entry becomes tedious, error prone and
more difficult to visualize.

Tolerance modelers which are linked to a CAD system can also be used to obtain
the additional information required to define a tolerance model. The tolerance model,
created by interactive graphics and consisting of parts and datums, joints, and loops, is
overlaid on an existing CAD geometric assembly model. This approach takes advantage of
existing geometric information and allows geometric selections rather than text responses.
Prototype modelers have been developed on several platforms including Alpha1, AutoCAD
[Simmons, 1990], CADAM [Rime, 1988], CALMA [Robison, 1988] and HP ME10
[Chun, 1987b]. However, the capability of the CATS analysis software changed
significantly during the development of these modelers, so the modelers are not consistent
and each has different modeling procedures. Thus developing a standardized tolerance
modeling method was part of the motivation for this thesis.
2.4 Summary

This chapter reviewed and discussed current research in assembly modeling and tolerance analysis and explained the importance of kinematic modeling. Progress made at BYU in developing tolerance analysis tools was presented and the need for a generalized modeling approach was introduced.
Chapter 3

GENERALIZED MODELING APPROACH

This chapter focuses on a generalized method of modeling an assembly for tolerance analysis. The basis of a complete tolerance model is a geometric assembly model. A tolerance modeling application is used with a CAD system to obtain additional kinematic and tolerance information and store it with the geometric model. Several tolerance modelers have been developed but they lack a general and consistent modeling procedure. The following sections present a more generalized approach to tolerance modeling. In addition, the role of the modeler is discussed in terms of the complete modeling-analysis process.

3.1 Modeling Procedure

One of the goals of this thesis was to develop and introduce a tolerance modeling scheme suitable for implementation on most 2-D and 3-D CAD modelers. The modeling approach must be systematic, intuitive, and general enough to represent a wide variety of mechanical assemblies. Several references discussed in Chapter 2 identified the need for a representation of assembly components and mating conditions in an assembly model. Once these elements have been established, the model must also include kinematic chains which relate assembly parts and their mating conditions collectively. In addition, Shah and Miller state that an adequate tolerance modeler must allow specification of not only size tolerances, but form and feature tolerances as well [Shah et al., 1990]. Thus the rest of this section is devoted to a model structure which includes parts and datums, joints, vector loops, and feature controls. The simple assembly shown in Figure 3.1 [Pryor, 1986] will serve as an example to demonstrate the modeling procedure.

Figure 3.1 Example assembly of stacked parts
3.1.1 Parts

The tolerance modeler must be able to distinguish between individual components or parts in the assembly model. Thus each part is assigned a unique part name. In addition, a datum reference frame (DRF) is defined identifying the location on the part from which all other datums, features and joints on the part are located. It will be shown that datum reference frames and feature datums play a significant role in the modeling process. Figure 3.2 illustrates each part in the example assembly with its associated part name and datum reference frame.

![Diagram of parts and DRFs](image)

Figure 3.2 Part names and datum reference frames

3.1.2 Joints

Locations of contact between assembly parts are called joints. A joint may involve point, line, or surface contact between two parts. The information required to define a joint includes a kinematic joint type, a global location, an orientation, the two parts in contact, and the joint's location relative to the datum reference frame on each part.

Since there are a variety of mating conditions, several common joint types have been identified in order to allow realistic modeling. The type of joint depends on the kind of contact between parts and the degrees of freedom which it allows. Commonly occurring joint types are shown with their associated degrees of freedom in Figure 3.3.
Figure 3.3 Common 3-D joint types and their associated degrees of freedom

All joints are actually three dimensional in nature, but in certain assemblies the majority of the adjustments due to dimension variations may occur in a single plane. In these cases, the assembly can be reduced to 2-D, and a 2-D subset of joint types may be used to model the assembly in a plane. The joint types used in 2-D modeling are illustrated in Figure 3.4. Each joint is identified by type along with the degrees of freedom permitted by each joint.
An important change in terminology for joint types is appropriate at this point. In past research, "rigid and center joints" were used to identify important feature locations on a single part and as endpoints of loop vectors. However, to be more consistent with the definitions of joints and datums, these "rigid and center continuation joints" will henceforth be referred to as "rectangular and centerline datums". In addition, the "constrained planar" contact joint will now be called the "rigid" joint.

Joints must be defined at all part interfaces in the assembly. Each joint must also be correctly oriented in order to accurately represent the interaction between parts and their relative degrees of freedom. For example, merely defining a revolute joint between two parts is not sufficient; the axis of allowed rotation must also be specified. Similarly, all joints which allow relative part motion about an axis or plane require information specifying the direction of the axis or plane. Figure 3.5 identifies the sliding plane; the direction in which the cylinder and the ground may adjust relative to each other during assembly. The joint origin and a second point identified by an X are used to define a vector describing this direction.
In addition to global coordinates, each joint must be located relative to the datum reference frame of each part that the joint connects. Datum paths are established which include only controlled dimensions to which the designer can specify a tolerance, and kinematic assembly dimensions which adjust at assembly time. Figure 3.6 illustrates the paths back to the DRFs of the cylinder and the ground. For this joint, each datum path consists of a single dimension from the joint to the datum reference frame on each part. Note that the path on the cylinder is a designer specified dimension, that is, a dimension which is independent of the assembly orientation. In contrast, the datum path on the ground part is a kinematic assembly dimension and depends upon the position of other parts at assembly time.
In most cases, the datum paths will not go back to the datum reference frame directly. Except in very simple assemblies, datum paths will follow intermediate part locations, called feature datums, back to the datum reference frames, thus locating the joint only in terms of controlled dimensions or kinematic variables on each part. Figure 3.7 illustrates the datum paths of another joint, including a feature datum between the joint and the DRF in the datum path on part Ground.

![Diagram of joint with DRF, Block, Ground, and Feature Datum]

Figure 3.7 Datum path may include feature datums

Using datums paths allows the designer to be flexible and explore the effects of several possible referencing layouts. Strict adherence to datums during manufacturing preserves design intent when parts are produced.

The assembly with all joints defined is shown in Figure 3.8. A unique symbol is used to identify the type of joint at each location. Note that joints 1 and 3 are cylindrical slider joints while joints 2, 4 and 5 are edge slider joints.
3.1.3 Vector Loops

The goal of the tolerance modeling process is to obtain a set of relationships which represent the assembly in terms of its geometry, topology, and kinematics. These relationships are produced by developing a set of vector loops which connect contact joints in the assembly. These loops may be either open or closed. Closed loops start and end at the same location and represent kinematic constraints on the assembly. For example, one kinematic constraint is that all parts in the assembly must maintain contact in order for the tolerance model to be valid. Open loops are used to determine assembly resultants of interest such as a clearance, orientation or position.

The assembly stack requires three vector loops, based on the number of parts and the number of joints (Equation 2.1).

\[
L = J - P + 1 = 5 - 3 + 1 = 3
\]

Since each loop may follow a variety of possible paths, a valid set of loops is not unique. The primary restriction is that the complete set of loops must include every part and every joint at least once. Figure 3.9 indicates a closed loop path between contact joints in the assembly.
Once the path between contact joints is established, the loop is expanded to include paths back to the datum reference frames of each part. Vectors in the contact joint loop are usually not controlled dimensions on the model. However, since each joint was located in terms of feature datums on each part, datum path information is added to the contact joint loop, providing a loop which does follow controlled dimensions and kinematic variables. The expanded loop for the example assembly is shown in Figure 3.10.

Certain adjacent vectors in the complete loop may be geometrically identical and do not contribute to a variational analysis on the assembly. Such vectors are removed from the loop before the analysis is performed. Figure 3.11 shows the complete loop after removal of two redundant vectors on the block.
Loops may be defined manually or automatically. Manual definition involves specifying a sequential list of joints for the loop. Once the complete loop is formed from the datum paths, tolerances are specified on the vector lengths which are independent of assembly adjustments. Tolerances are also optionally applied to the assembly-independent angles between the vectors. In contrast with manual loop creation, defining loops automatically requires only the specification of tolerances since the loop paths are generated automatically. Automatic loop generation will be discussed in greater detail in Chapters 4 and 5.

3.1.4 Feature Controls

In addition to size variations, parts exhibit feature variations due to production. For example, a machined surface is not truly flat, though it may be considered to be flat enough for its purpose. Feature controls are the mechanism used to constrain feature variations to fall within acceptable limits.

The ANSI Y14.5 specification was established to standardize control of feature variations. The standard includes five classes of feature controls: size, form, orientation, position, and runout. Feature controls in nearly all classes are based on an imaginary set of two planes or surfaces which define the region of acceptable feature variation, such as parallel planes or concentric cylinders. Individual form and feature control types are shown with their associated symbols in Figure 3.12.
The steps for modeling feature variations include choosing the type of feature variation, identifying the part and joints to which the feature control applies, specifying the width of the tolerance zone, and finally, specifying a characteristic length (if required for that specific feature control). Since feature variations are modeled on the joint level, the feature control must include all joints on a common feature. Feature variations to the tolerance model are then introduced at each joint associated with a feature control object.

Incorporating form and feature variations into existing tolerance analysis software is presently under study [Goodrich, 1991]. Mathematical models are currently being developed for each type of feature control.

### 3.2 Assembly Model File

When the modeling process has been completed, a tolerance model file is generated which contains a representation of the essential geometry, kinematics, and topology of the entire assembly. Currently, the model file is divided into separate sections for the assembly, parts, datums, joints, vector loops, and feature controls. Model files for the assemblies discussed in this thesis can be found in the Appendix A. The creation of the model file completes the assembly modeling process.

### 3.3 Tolerance Analysis and Results

The model file is used by the CATS analysis software to determine the values of the dependent variables, which are the variations of the kinematic variables in the assembly model. Initially, assembly relationships are obtained from the set of closed loops. Since
the set is composed of closed loops, the clearance vector is zero. The vector loops are linearized using a first order Taylor series and taking small perturbations about the nominal, yielding scalar equations in matrix form, shown below in Equation 3.1.

\[
{dC} = \{0\} = [A] \{dI\} + [B] \{dD\} \tag{3.1}
\]

where:

\[
\begin{align*}
{dC} & = \text{The vector of Closed loop assembly resultants} = \{0\} \\
{dI} & = \text{The tolerance vector of Independent variables} \\
[A] & = \text{The sensitivity matrix of the independent variables} \\
{dD} & = \text{The variation vector of Dependent variables} \\
[B] & = \text{The sensitivity matrix of the dependent variables}
\end{align*}
\]

The matrix equation is rearranged to solve for the unknown tolerances: the dependent variables, as shown by Equation 3.2.

\[
{dD} = -[B]^{-1} [A] \{dI\} = [k] \{dI\} \tag{3.2}
\]

where:

\[
[k] = \text{The closed loop sensitivity matrix of the dependent variables} \\
\text{with respect to the independent variables}
\]

Once the dependent variations are known, the variation on any dimension in the assembly may be obtained. If a variation of another dimension in the assembly is desired, an open loop is used to obtain the unknown assembly resultant.

\[
{dR} = [C] \{dI\} + [D] \{dD\} \tag{3.3}
\]

where:

\[
\begin{align*}
{dR} & = \text{The variation vector of the open loop clearance Resultant} \\
[C] & = \text{The sensitivity matrix of the independent variables} \\
[D] & = \text{The sensitivity matrix of the dependent variables}
\end{align*}
\]

Substituting Equation 3.2 into Equation 3.3 gives the relationship shown in Equation 3.4.

\[
{dR} = \left[ [C] - [D][B]^{-1} [A] \right] \{dI\} = [S] \{dI\} \tag{3.4}
\]

where:

\[
[S] = \text{The sensitivity matrix for the assembly resultant}
\]

With the assembly resultant now in terms of its sensitivity to changes in the independent variables, the variation in the dependent resultants in Equation 3.4 may be
expressed in several ways. The two mathematical models most commonly used for estimating tolerance accumulation are the worst case and statistical methods, shown by Equations 3.5 and 3.6.

Worst case model: \[ dR_i = \sum_{i=1}^{n} |S_{ij}| dI_j \leq T_{ASM} \]  \hspace{1cm} (3.5)

Statistical model: \[ dR_i = \sqrt{\sum_{i=1}^{n} \left[ S_{ij} dI_j \right]^2} \leq T_{ASM} \] \hspace{1cm} (3.6)

Where \( i \) = the index of the vector of assembly resultants and \( j \) = the index of the vector of independent variables

The worst case model assumes all dimensions are at their extreme limits and in their worst possible configuration. Designing using worst case analysis gives the assurance that every manufactured assembly will meet predetermined specifications \( T_{ASM} \). This results in low production waste but requires tight tolerances and thus high manufacturing costs. The statistical model takes into account the statistical probability of variations occurring simultaneously and adds them as independent random variables. The choice of the appropriate mathematical model depends on the designer's need and analysis requirements.

When the analysis is complete, the results are recorded in the model file, which is passed back to the modeler. The dependent tolerances, which were unknown when the assembly model was created, are displayed by the modeler for the designer to review and evaluate.

3.4 Summary

This chapter presented and discussed the generalized modeling method and outlined the process of representing parts, datums, joints, loops, and feature controls in the tolerance model. The chapter gives a description of the model file followed by an overview of how the analysis is performed, including the two most common analysis techniques: worst case and statistical models.
Chapter 4

MODELING CONSIDERATIONS

Chapter 3 presented a generalized and systematic method of obtaining the additional information required for assembly tolerance modeling. This chapter discusses new procedures and checks which are performed once the initial model information has been acquired and introduces additional considerations which are required for the development of an intelligent modeler. Sections include discussion of a degree of freedom analysis, automatic loop generation, automatic identification of dependent variables, and the rotation of the local coordinate frame.

4.1 Degree of Freedom Check

Chapter 2 mentioned that the kinematic joints in an assembly are used to identify how the parts adjust to dimensional variations. When the joints on a part are considered collectively, they restrict the part's motion to certain directions. These are the degrees of freedom of the part. When defining joints it is important to locate a sufficient number of joints on each part so that the tolerance model accurately represents the degrees of freedom of each part in the actual assembly. Therefore, a degree of freedom analysis is performed on the tolerance model to verify that an adequate number of joints have been defined.

An assembly model which lacks the necessary joints is underconstrained and does not take into account all the degrees of freedom of the actual assembly. Tolerance analysis performed on unconstrained assemblies is modestly illuminating at best and invalid and misleading at worst. Similarly, redundant joints unnecessarily increase the number of closed kinematic loops required (Equation 2.1), since all joints must be included in the set of loops. Traditional DOF methods [Chun, 1988a; Robison, 1989; Simmons, 1990] are not sufficiently robust to recognize parallel translational degrees of freedom and colinear rotational degrees of freedom. The procedure outlined in this section is preferred since it considers not only the number of degrees of freedom, but their directions as well. Since this method utilizes vector operations, 3-D joint representations will be used.

Joints may be classified by the degrees of freedom they allow. For example, the revolute joint allows one rotational degree of freedom, while it constrains translation normal to the rotational axis. It is impossible to define two constrained degrees of freedom uniquely, since they could be any two orthogonal directions which lie in the plane normal to the direction of allowed rotation. Some joints are more easily described by the degrees
of freedom they constrain. A point slider joint has one translational degree of freedom which is constrained (the direction normal to the surface), while two translational DOF in the sliding plane and all three rotations remain unconstrained. Since there remain an infinite number of allowed translational DOF orientations, the point slider joint is best described in terms of the degree of freedom which has been constrained.

In this thesis, the author developed a procedure for performing a DOF check which applies to joints of both allowed and constrained representations. The procedure considers translation and rotation separately and is outlined first in terms of translational DOF. When a joint is unconstrained, it allows three translational DOF, and no unique directions are considered. Joints having one translational DOF are described with a vector defining the direction of constrained translation. Joints having two constrained DOF have one allowed DOF and are described by a vector in the direction of allowed translation. Finally, totally constrained joints have no DOF and no vector directions are identified. Representation for rotational DOF is similarly developed. As Table 4.1 shows, joints are either constrained, unconstrained, or described in terms of one constrained DOF direction and the last allowed DOF direction. When two constrained directions are known, the last allowed direction is obtained using the cross product. For example, the planar joint’s DOF are described by a vector identifying the direction of constrained translation (Constrained Translation 1) and by a second vector defining the axis of allowed rotation (Last Allowed Rotation).

Table 4.1 Representations of degree of freedom directions for kinematic joints

<table>
<thead>
<tr>
<th>Joint Type</th>
<th>Number of Constrained Translations</th>
<th>Constrained Translation 1</th>
<th>Last Allowed Translation 1</th>
<th>Number of Constrained Rotations</th>
<th>Constrained Rotation 1</th>
<th>Last Allowed Rotation 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rigid</td>
<td>3</td>
<td>vec</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prismatic</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revolute</td>
<td>3</td>
<td>vec</td>
<td>2</td>
<td>vec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cylindrical</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ujoint</td>
<td>3</td>
<td>vec</td>
<td>1</td>
<td>vec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Planar</td>
<td>1</td>
<td>vec</td>
<td>2</td>
<td>vec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ball and Socket</td>
<td>3</td>
<td>vec</td>
<td>0</td>
<td>vec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge Slider</td>
<td>1</td>
<td>vec</td>
<td>1</td>
<td>vec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cylindrical Slider</td>
<td>1</td>
<td>vec</td>
<td>1</td>
<td>vec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point Slider</td>
<td>1</td>
<td>vec</td>
<td>0</td>
<td>vec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sphere Slider</td>
<td>1</td>
<td>vec</td>
<td>0</td>
<td>vec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crossed Cylinder</td>
<td>1</td>
<td>vec</td>
<td>0</td>
<td>vec</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.2.1 Procedure for DOF Analysis

In the degree of freedom analysis each component of an assembly is examined individually, considering only the joints on that part. The part is given the same DOF representation as previously described for the joints, but the part is initialized with zero translational and zero rotational DOF. As each joint on the part is considered, the degrees of freedom which a joint does not allow are removed from the part. Since actual DOF directions are identified, a part's DOF is not affected by additional joints having a constrained translational DOF parallel to a translational direction already constrained on the part. Similarly, additional joints with a constrained rotational DOF colinear to a part's already constrained rotational direction do not incorrectly remove extra degrees of freedom from the part. This flexibility is the advantage of the new degree of freedom analysis.

The steps in the DOF analysis are outlined as follows:

1. Select a part in the assembly
2. Make a list of all the joints on current part
3. Identify current joint type and joint orientation
4. Determine the number of constrained translations and define the appropriate vector directions for constrained translations 1 and 2 and last allowed translation
5. Determine the number of constrained rotations and define the appropriate vector directions for constrained rotations 1 and 2 and last allowed rotation
6. Remove the joint's constrained DOF from the part
7. Repeat steps 2 to 6 for each joint on current part
8. Display the part's DOF directions
9. Repeat steps 1 to 8 for each part in assembly

4.2 Automatic Loop Generation

When all the joints in an assembly model have been defined and a DOF analysis has confirmed that the assembly is adequately constrained, vector loops may be generated either manually or automatically. Since one of the goals of this research is to reduce the amount of effort required by the designer, significant efforts have been directed toward complete loop automation. The foundation of generating loops automatically is based on the concepts of network graph theory.
Network graph theory has been used in such disciplines as electrical circuits and kinematics to determine representative sets of equations. The topology of a network includes nodes or components, and edges or arcs which represent the connections between the nodes. In electrical theory, the edges are the conductors and the nodes are the junctions between conductors. Network graphs are used to develop the Kirchhoff current loops. For kinematics, the nodes are the mechanical components and the edges are the joints between components and network graph theory is used to obtain the equations of motion. This research will closely parallel the kinematic approach.

Simmons discusses the application of network graph theory to automatic loop generation for tolerance analysis [Simmons, 1990]. Parts in the assembly are identified as nodes in the graph while kinematic joints are shown as edges which connect the nodes. Figure 4.1 illustrates the network graph for the geometric stack assembly discussed with the generalized modeling method in Chapter 3. Note that the network graph contains no geometric information, only topological information.

![Network Graph for Geometric Stack Assembly](image.png)

**Figure 4.1** Network graph for the geometric stack assembly

A matrix called the *incidence or connectivity matrix* is used to map the relationships between parts and joints. By convention, rows in the matrix represent different assembly parts, and columns represent individual joints. Since each joint connects two parts, two 1's are placed in each joint column on the row elements corresponding to the two mating parts, and the rest of the matrix is filled with zeros. The correlation between the network graph and the connectivity matrix should be clearly apparent. The connectivity matrix for the sample assembly is shown in Figure 4.2. Note that the number of 1's in any horizontal row (part) indicates the number of joints on that part.
### Modeling Considerations

<table>
<thead>
<tr>
<th>J1</th>
<th>J2</th>
<th>J3</th>
<th>J4</th>
<th>J5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cylinder</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Block</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ground</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.2 Connectivity matrix for the geometric stack assembly

The number of required loops is determined by Equation 2.1,

\[ L = J - P + 1 = 5 - 3 + 1 = 3 \]

and the set of loops is obtained by traversing different paths between the 1's in the matrix. A horizontal path between two 1's represents crossing a part; entering at one joint and leaving at another. A vertical path between 1's represents transferring between parts at a joint. These concepts will be discussed in greater detail in Chapter 5.

Once the loop paths have been determined, the only additional information required is the tolerances on the dimensions over which the designer has direct control. When the tolerances have been obtained, the dependent variables are identified, the equations of motion are determined and finally, the variations of the dependent assembly variables are solved for.

### 4.3 Dependent Variable Identification

Two types of variations exist in the assembly model. The first kind of variations, called design or independent variables, are variations in dimensions on a single part and are a function of the manufacturing process used to machine the dimensioned features. The designer controls these variations by applying tolerances to each part dimension. The second kind of variations, termed assembly, kinematic or dependent variables, result from interactions between parts and may adjust during assembly. Each vector loop contains both independent and dependent variables, which may be in the form of vector lengths or angles between vectors. Dependent variables associated with a vector loop in the sample assembly are indicated in Figure 4.3. Dependent angles are \( \theta_1 \) and \( \theta_2 \) and the dependent lengths \( U_1 \), \( U_2 \) and \( U_3 \).
Dependent variables result from kinematic degrees of freedom at a joint. Degrees of freedom between mating parts allow the parts to adjust relative to each other at assembly time. Each kinematic degree of freedom introduces a dependent variable; a translational DOF produces a dependent vector length, and a rotational DOF contributes a dependent angle between vectors. A joint in different loops produces the same dependent variable in each loop. Since a design engineer can only directly control tolerances on the independent variables, it is necessary to distinguish between independent and dependent variables.

Chun and Robison each did some preliminary work in identifying dependent variables [Chun, 1988a; Robison, 1989]. The method used, called the Joint Sequence Method (JSM), attempts to identify dependent lengths and angles associated with a joint by considering the types of the previous joint and the next joint in the loop. Although it can handle all 2-D joints, there are several 3-D joints for which it fails. The major weakness of the joint sequence method is that it doesn't consider joint orientation when determining variable dependency. For example, consider the hinge shown in the following figure.
The rotation (θ) shown in Figure 4.4a is dependent since it depends on the position of the hinge. However, Figure 4.4b illustrates the rotation (ϕ) which is not dependent because it is about an axis other than the revolute joint’s axis of allowed rotation. Thus in order to automatically identify dependent vector lengths and angles, the joint sequence method must be expanded to include joint orientation.

4.3.1 Dependent Vector Lengths

The first type of dependent variables are vector lengths which result when there is contact between parts which can slide relative to one another. Some joints which allow relative sliding are the planar, edge slider and cylindrical slider joints. Since these joints permit adjustments between parts at assembly, any vector in the allowed sliding direction has a dependent length. Thus the variations on these sliding vectors are dependent variables.

The joint’s orientation axes are used to determine a vector’s dependency. In 2-D, the planar, edge slider and cylindrical slider joints are the only joints which have the possibility of a dependent length. These joints all have a NORMAL orientation axis which defines a direction perpendicular to the direction of allowed sliding and the slider joints have a PLANAR axis which defines the direction of line contact, as shown in Figure 4.5.

![Diagram](image)

Figure 4.5 Determining vector length dependency

If the vector in question is perpendicular to the joint’s NORMAL axis, and not parallel to the PLANAR axis, then the vector must lie off of the line contact axis and in the sliding plane. Therefore the vector’s length is dependent. A vector which is not perpendicular to
the joint's **NORMAL** axis has an independent length, since it is not in the plane of allowed sliding. Parallel and perpendicular comparisons are determined using the vector dot product, as shown below:

\[
\text{if } ((\text{Vector} \cdot \text{NORMAL} = 0) \text{ and } (\text{Vector} \cdot \text{PLANAR} = 1)) \text{ then}\n\]
\[
\text{Vector is Dependent}
\]

Although this expression is correct and very helpful, its converse is not necessarily true. The converse case occurs when both incoming and outgoing vectors are in the sliding plane for a planar joint or along the axis of line contact for edge and cylindrical slider joints. Some examples of this case are shown in Figure 4.6.

![Figure 4.6 Problem cases for automatic identification of dependent lengths](image)

In 2-D these instances never occur since the vector on the flat part is always orthogonal to the joint's **PLANAR** axis.

The dependency of a vector's length is caused by a degree of freedom at one of the joints which define the vector. For this reason, a vector's length cannot be identified as dependent or independent until both joints of each dimension vector are considered. Thus a independent vector is not dependent at either end while a dependent vector is dependent at one end or the other. Vectors which are dependent at both ends typically do not occur since at least one end must have a fixed reference.

### 4.3.2 Dependent Rotation Angles

Dependent rotation angles are the second type of dependent variables. 2-D joints which include a dependent rotation are the revolute, cylindrical, cylindrical slider and edge slider joints. For the edge slider joint the axis of the dependent rotation is the direction of line contact or in other words, the joint's **PLANAR** axis. Similarly, the dependent rotation axis for the revolute joint is about the joint's **AXIAL** axis. However, since the cylindrical and cylindrical slider joints both have a finite radius of curvature, the dependent...
axis of rotation occurs about the centerline datum AXIAL direction, which is offset from the joint by the radius distance.

All cylindrical and cylindrical slider joints occur in combination with a centerline datum so that the vector on the circular part may be orthogonal to other part surface at the point of contact. It is important to realize however, that although the dependent angle is applied at the centerline datum, the dependency was produced by a contact joint since the joint has the rotational degree of freedom, not the centerline datum. For this reason, when a part has three or more joints and two of the joints share the same centerline datum, each joint associated with the centerline datum contributes a dependent angle and two vector loops must pass through the centerline datum to account for both dependent angles. However, for a part having exactly two joints, both of which share a centerline datum, only one loop is required since there is only one path through the part.

Since revolute joints and centerline datums both have a dependent rotation in their AXIAL direction, the condition for dependency at a revolute joint or centerline datum is expressed the same way, as follows:

if ((VectorIn \cdot AXIAL = 0) \text{ and } (VectorOut \cdot AXIAL = 0)) \text{ then }

\text{Vector is Dependent}

The dependency condition for the edge slider joint is shown below:

if ((VectorIn \cdot PLANAR = 0) \text{ and } (VectorOut \cdot PLANAR = 0)) \text{ then }

\text{Vector is Dependent}

As with dependent vector lengths, there are complications in 3-D which do not occur in 2-D instances. Consider the part shown in Figure 4.7a. The angled face may be machined at an angle within a specified tolerance, thus the angle of the face is an independent variable (Figure 4.7b). However, there is also a dependent angle produced by sliding contact at two locations on the outer surface (Figure 4.7c). The solution is to separate combinations like these into completely independent and completely dependent variables which can be considered individually.
4.3.3 Identification of 3-D Dependent Variables

The scope of this thesis was limited to automating the identification of 2-D dependent variables. This was accomplished using methods which also apply to 3-D. Thus, all the dependency conditions discussed can be implemented in 3-D without modification. These rules, though true in 3-D, by no means represent the complete set of dependency conditions required for 3-D modeling. Although progress made in this thesis is a significant portion of dependent variable identification in 3-D, this is a continuing topic and is suggested for future research. A comparison between 2-D and 3-D dependent vector lengths and angles associated with each joint are shown in Table 4.2.

<table>
<thead>
<tr>
<th></th>
<th>2-D Dep Lengths</th>
<th>2-D Dep Angles</th>
<th>3-D Dep Lengths</th>
<th>3-D Dep Angles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rigid</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Prismatic</td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Revolute</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Cylindrical</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ujoint</td>
<td></td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Planar</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Ball and Socket</td>
<td></td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Edge Slider</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Cylindrical Slider</td>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Point Slider</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Sphere Slider</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Crossed Cylinder</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

4.4 Rotation of Local Coordinate Frame

The CATS analysis software requires a set of loops with relative rotations between vectors. Any vector can be transformed into any other vector direction with a series of two rotations. By convention, the first rotation is performed about the local Z-axis, after which
a new coordinate frame is established. The second rotation is about the new local Y-axis and aligns the new local X-axis with the second vector. The following discussion shows how $\theta_z$ and $\theta_y$ are determined in the general case.

The following two figures demonstrate the rotations required to transform the first vector, labeled Previous Vec, into the second vector, labeled Vec. A bold symbol identifies vector quantities, and the bar symbol above certain quantities denotes unit vectors. The large dot indicates the dot product of two vectors. Figure 4.8 illustrates the current local coordinate system (consisting of unit vectors LocalX, LocalY and LocalZ) aligned with Previous Vec, which is colinear with the local X-axis.

![Figure 4.8 Rotation of local coordinate frame about Z-axis](image)

The rotation about the local Z-axis ($\theta_z$) is the angle required to rotate the local X-axis into the plane containing the local Z-axis and the second vector, Vec. The method of obtaining $\theta_z$ is outlined by the following equations which result in Equation 4.3.

\[
\beta \cos \theta_z = \frac{\text{LocalX} \cdot (X_{\text{comp}} + Y_{\text{comp}})}{}
\]

\[
\beta \sin \theta_z = \frac{\text{LocalY} \cdot (X_{\text{comp}} + Y_{\text{comp}})}{}
\]

\[
\theta_z = \text{atan2}(\beta \sin \theta_z, \beta \cos \theta_z)
\]

where $\beta$ is the magnitude of $X_{\text{comp}} + Y_{\text{comp}}$. Since $\text{atan2}$ computes the ratio of $\beta \sin \theta_z$ and $\beta \cos \theta_z$, $\beta$ cancels out and need not be calculated.
Figure 4.9 shows the X and Y axes rotated about the Z axis to their new orientation.

![Diagram showing rotation of coordinate frame](image)

**Figure 4.9 Rotation of local coordinate frame about Y-axis**

The rotation about the local Y-axis ($\theta_y$) is the angle required to rotate the local X-axis into the second vector, Vec. The method of obtaining $\theta_y$ is outlined below by the following equations resulting in Equation 4.6. Note that $\theta_y$ is negative, by definition of the right hand rule.

\[
\beta \cos \Theta_{\text{Y}} = \frac{\text{LocalX}'}{\text{Vec}} \tag{4.4}
\]

\[
\beta \sin \Theta_{\text{Y}} = \frac{\text{LocalZ}}{\text{Vec}} \tag{4.5}
\]

\[
\theta_y = - \text{atan2}(\beta \sin \Theta_{\text{Y}}, \beta \cos \Theta_{\text{Y}}) \tag{4.6}
\]

As before, $\beta$ cancels out and need not be calculated.

This discussion assumes that if there are any dependent angles, they will either be $\theta_z$ or $\theta_y$. In 2-D, if all rotations between vectors occur in the XY plane, the $\theta_y$ rotation is always zero. However, some 3-D joints have three dependent angles and require an additional rotation. This concept will be further illustrated with a 3-D example in Chapter 7.
4.5 Summary

This chapter focused on the theoretical principles upon which implementations of the generalized modeling method are based. A description of a kinematic degree of freedom analysis was discussed which accounts for parallel and colinear DOF which have already been constrained. Basic concepts of using network graph theory for automatic loop generation were discussed. A practical method was presented for automatically identifying dependent vector lengths and angles. Finally, a procedure for Z and Y rotation between vectors was developed.
Chapter 5

AUTOMATIC LOOP GENERATION

The purpose of kinematic chains or vector loops is to represent collectively the relationships between parts and joints in the assembly. Thus the set of loops must include every part and joint in the assembly. In addition, there must be enough loops to solve for all the dependent variables.

A major portion of the assembly modeling procedure is spent setting up a sufficient set of vector loops. In addition, since the choice of loop paths is largely up to the designer, it is critical that they be defined correctly. Thus automating as much of the loop defining process as possible reduces modeling time as well as reducing the possibility for error. In fact, Wang states that the automatic generation of vector loops is the "foundation of any tolerance analysis at the assembly level" [Wang, 1990]. This chapter discusses the concepts and issues of automatic vector loop (AVL) generation in a tolerance model.

5.1 AVL Prototype and Limitations

The first step in automating the generation of vector loops is obtaining a set of paths between contact joints in an assembly. Simmons outlines a method of determining these contact joint loops using network graph theory [Simmons, 1990]. His prototype demonstrates feasibility of obtaining a set of loops automatically. However, once the contact joint loop has been obtained, each path between joints must then be expanded to include appropriate individual dimensions which utilize datums on the part. Thus although determining contact joint loops greatly aids the designer in choosing an appropriate set of complete vector loops, it still leaves a significant part of the loop defining process to be performed manually.

Several concepts limit further development of the AVL prototype beyond showing feasibility. In the prototype AVL, a sort is used to arrange the matrix such that the first row contains the part having the most joints. It also attempts to diagonalize the matrix. Although these steps make it slightly easier to visually follow the loops, they are computationally unnecessary. In addition, several counter-examples were found where all joints were not included in the set of loops, thus the algorithm needs to be more robust. Finally, since the number of nodes and length of the contact joint loop often has no correlation with those of the expanded loop, the expanded loop should be the basis for the loop comparison criteria rather than the contact joint loop.
The ease of implementing and further developing an algorithm is closely tied to choice of programming language in which it is to be coded. The AVL algorithm involves tree searches and recursive procedures (functions which can call themselves), thus a language allowing function recursion would simplify implementation and maintenance. In addition, the number of parts and joints are specific to each problem so the dimensions of the connectivity matrix and other arrays are not known until run time. Dynamic memory allocation would also be an advantage. For these reasons, the new implementation was written in C rather than FORTRAN (see Appendix B). The following sections present the modified approach to automatic vector loop generation.

5.2 Virtual Tree Representation

Chapter 4 introduced the network graph representation and the connectivity matrix for an assembly. The connectivity matrix can also be formed directly from the assembly. This discussion is based more on the connectivity matrix than the network graph since the matrix plays a greater role in loop generation, however, they are both equivalent.

Once the connectivity matrix has been constructed with the rows representing the assembly parts and the columns representing the joints between parts, different paths through the matrix are determined and compared. It is convenient to illustrate traversing the matrix as a tree structure. It is important to realize that this hierarchy is not added overhead or an extra data structure. Since it does not really exist, it is referred to as the virtual tree and is simply a means of showing the path through the matrix when searching for the set of loops.

Figure 5.1 illustrates the correlation between the connectivity matrix and the virtual tree in terms of transferring between parts at a joint. The vertical arrow in Figure 5.1a indicates changing from part Cylinder to part Ground at joint J1. Figure 5.1b shows the same transfer between parts using the virtual tree. The 1's within the circle at J1 represent the two 1's identifying the mating parts in the J1 column (joint) of the matrix.
Figure 5.1 Equivalent representation for transferring between parts at a joint

Figure 5.2 shows a similar correspondence for transferring between joints on a part. The horizontal arrow in Figure 5.2a indicates the transition between joints J1 and J2 on part Ground. Figure 5.1b shows the same transfer between joints using the virtual tree representation. The 1's within the rectangle indicate 1's on the same row (part) of the matrix.

Figure 5.2 Equivalent representation for transferring between joints on a part

### 5.3 Determining Candidate Contact Joint Loops

The first step in automating vector loop generation is to obtain a candidate closed loop which passes through the contact joints in an assembly. A starting joint and part are chosen from the matrix. A candidate loop is obtained when the tree search completes a loop by returning to its starting joint. The new AVL algorithm uses a modified depth first search [Algorithms, 1983] to traverse the virtual tree and obtain candidate loops. An example is presented next which demonstrates the tree search and is followed by a detailed description of the new algorithm.
5.3.1 Tree Search Example

A simple mechanism will be used to introduce the tree search. The Watt linkage (a mechanism commonly used in kinematics) is illustrated in Figure 5.3a along with its associated network graph, in Figure 5.3b. The parts or mechanism components are numbered and labeled with a "P". The joints are also numbered and labeled with a "J". All joints are revolute type.

![Watt mechanism diagram](image)

Figure 5.3 Watt mechanism

The following figures will outline the concept of obtaining candidate loops by searching the virtual tree corresponding to an assembly. Figure 5.4 shows the virtual tree representation and the connectivity matrix for the Watt mechanism. The 1's which are normally shown in the virtual tree have been removed for clarity. The search starts at joint J1 and continues down the tree until it reaches joint J1 again, thus identifying a candidate loop consisting of joints J1-J3-J6-J4-J1.

![Traversal diagram](image)

Figure 5.4 Traversal showing first candidate loop
After the loop has been evaluated, we move back up the tree until reaching a place where an alternate path can be explored: part P1. The downward search is continued from part P1 but this time we arrive back on part P2, which has already been used. Since this loop doubles back on itself, searching along this path is terminated prematurely, as illustrated by Figure 5.5.

Figure 5.5 Traversal showing first rejected path

Again, we must move back up the tree until we reach a part where an alternate path branches from the first. This occurs on part P1, and the downward search begins again. Another candidate loop is found consisting of the following joints: J1-J3-J6-J7-J5-J2-J1, as shown in Figure 5.6.

Figure 5.6 Traversal showing second candidate loop
When the candidate loop has been evaluated, we move back up the tree until reaching another branching point, on part P1. The downward search is initiated again along an alternate path but we reach a part that has already been used before arriving at the starting joint. Again further searching along this path is halted, as Figure 5.7 indicates.

![Diagram of a tree with branching points and a table of transitions](image)

**Figure 5.7** Traversal showing second rejected path

At this point we move up the tree in search of another alternate path but in doing so we retrace the path back to the original starting location. Thus the entire tree has been traversed and there are no unexplored paths.

The power of the tree search lies in two recursive functions; one which marches down the tree in search of a leaf and the other which climbs up the tree in search of a branch. A leaf is the same joint as the joint that the search was initiated from. Since the goal is to determine a set of closed loops and since a closed loop must start and end at the same joint, the search is complete and successful when it reaches a leaf. A branch is an assembly part which has more than two joints and thus has more than one path through the part. When a branch is encountered, a choice must be made to determine which of the remaining joints on the part will be the next joint in the path.

At this point additional notation used in the virtual tree representation will be explained. The path down the tree from the starting joint is identified by -1's. Each time the search moves deeper in the tree, the 1 elements along the path are changed to -1's so
that the path is identified. The corresponding -1 paths are visible in the matrix thus maintaining a notation which is consistent with the matrix path representation used by Simmons. The encircled 1's identify the matrix element from which a function was called. The large arrow in each diagram indicates the direction of travel on the virtual tree during the function call. The small arrow illustrates the subsequent direction of travel when the next function is executed.

5.3.2 Down-to-Leaf Procedure

The down-to-leaf procedure is initiated from the matrix element of the current joint and the current part. It first finds another joint on the same part, then transfers to the other part associated with this new joint. The action which follows depends on whether or not the new part has already been used and if the new joint is a leaf. Figure 5.8 illustrates the action which is performed in each situation. The down-to-leaf function may encounter three possible cases.

Case 1: The first condition results when the new joint is not a leaf, in which case the down-to-leaf procedure is performed again, this time starting from the new part and the new joint.

Case 2: The second case occurs if the new part has already been used. If this happens, the portion of the loop between the two occurrences of this new part is redundant, and the existence of a better candidate loop is guaranteed; the one which does not include the redundant section. Thus further search along this path for a candidate loop is aborted and the up-to-branch procedure is called.

Case 3: The third case takes place if the new joint is a leaf. At this point, a candidate loop has been found consisting of the sequence of joints from the initial joint to the leaf joint. After evaluating the candidate loop, the up-to-branch procedure is called and the search continues for more candidate loops.
Figure 5.8 Method of traversing tree in search of a leaf node
5.3.3 Up-to-Branch Procedure

The up-to-branch function is used when there is no reason to continue searching down a certain path of the tree. When this has been determined, up-to-branch is called to locate the next place that a downward tree search can be initiated. When the search yields a part which serves as a branch, the next available joint on the part is determined and down-to-leaf is called again.

The up-to-branch procedure is started from the matrix element of the current joint and the current part. It finds the next joint on the part, then passes through the joint to the other part associated with this new joint. The action which follows depends on whether or not the new part is a branch, and if whether or not all the joints on the new part have already been used. A summary of the four possibilities encountered by the up-to-branch function is shown in Figure 5.9.

Case 1: The first case results if a branch is encountered and there are still unused joints on the branch. An unused joint on the branch part is selected and the down-to-leaf procedure is called from there.

Case 2: In the second case, the new part is a branch but downward paths stemming from each of the joints on the branch part have already been checked. In this case up-to-branch is called again and the upward search for a branch continues since the branch which was found could not be further explored.

Case 3: The third possibility takes place when the new part is not a branch but the search has progressed back to the top of the tree. In addition, this situation may also occur if the top of the tree is reached and the part is a branch but paths from all the joints have been explored. In either case, the entire tree has been searched and all possible candidate loops have been found. In both instances, the search is terminated.

Case 4: The final case occurs if the new part is not a branch and the current position is not at the top of the tree. Therefore, up-to-branch is executed again in search of a branch.
**Up To Branch Procedure**

At a branch?  
\[\text{yes}\]  
\[\text{no}\]

Unused joints?  
\[\text{yes}\]  
\[\text{no}\]

At top of tree?  
\[\text{yes}\]  
\[\text{no}\]

Unused Joints Left on Branch  
All Joints on Branch Used  
At Top of Tree  
Not A Branch

---

**Figure 5.9** Method of traversing tree in search of a branch node
The following is a summary of the steps in the recursive procedures just discussed.

**Down-To-Leaf procedure:**
1. Determine the next joint on the current part
2. Obtain the new joint's other mating part
3. Repeat steps 1-2 until
   a) the new joint is the starting joint (candidate loop found)
   b) the part has already been used (move up tree until finding a branch)

**Up-To-Branch procedure:**
1. Determine the previous joint on the current part
2. Obtain the new joint's other mating part
3. Repeat steps 1-2 until
   a) a branch is encountered
      i) if all joints on branch have been used (continue moving up tree)
      ii) if all joints on branch have been used and top of tree is encountered (end)
      iii) if a joint has not been used (search downward for a leaf)
   b) top of tree is reached where part is not a branch (end)

### 5.4 Forming Complete Loop Using Datum Paths

Once a candidate loop between contact joints has been found, the loop is expanded to include datum paths between joints, forming paths across each part. Thus the complete loop consists only of controlled dimensions and kinematic dimensions. Expanding the paths between contact joints involves several steps. The joints in the loop are considered in adjacent pairs (first and second joints, second and third joints, etc.), until all joints in the loop have been examined.

The first step is to expand the path between each pair of joints. Since the two joints are adjacent in the loop, they share a common part and each joint has a datum path back to the DRF on this part. The expanded path between joints includes sequentially, the first joint, the first joint's datum path back to the common DRF, the datum path of the second joint in reverse order, and finally, the second joint. This process is shown in Figure 5.10.
When the path between two joints has been expanded in terms of datum paths, the complete path is checked for redundant dimensions. In some cases, the two datum paths overlap, that is, they share common dimensions on the paths back to the DRF. Thus two adjacent loop vectors may be geometrically equivalent. These redundant dimensions are not needed for the tolerance analysis and may introduce unrealistic variation into the tolerance model. Equivalent dimensions in the paths are first checked by name. The following figure illustrates the completed path between two contact joints.

Figure 5.11 Removal of adjacent geometrically equivalent loop dimensions

Figure 5.11a shows a path segment starting at joint J10, following the datum path to DRF2, and ending at joint J11. The nodes in the loop segment are searched for "A-B-A" combinations, that is, appearances of a different node between adjacent occurrences of the
same node. One such occurrence is found: 3-2-3. The latter two nodes of this combination are removed and the resulting path is shown in Figure 5.11b. Another A-B-A combination is identified and removed, yielding the final path in Figure 5.11c. The path segment is continually checked until no further changes can be made to the path.

Adjacent equivalent dimensions may also occur by identical location. In this case, "A-B-a" combinations are found where the global location of node "A" is the same as the global location of node "a". These occurrences are removed in the same way as A-B-A combinations, with one added rule: datums are always removed from a loop segment instead of joints. For example, if a J-B-D combination occurs where J is a joint, and B and D are datums, and J and D have identical locations, the B-D pair is removed. If a D-B-J combination occurs, the D-B pair is removed leaving the joint in the loop. This rule is essential to ensure that every joint's kinematic degrees of freedom (and therefore dependent variables) are preserved in the loop. Datums are removed instead since they don't have degrees of freedom.

Similarly, when a datum and a joint occur at the same location the datum is removed and the joint remains to identify the common node in the loop. Thus D-J and J-D combinations both result in a single node J when joint J and datum D have the same location.

The process of expanding loop segments between joints and eliminating equivalent adjacent geometric dimensions is performed for the entire loop. The resulting path is called the complete candidate loop.

5.5 Candidate Loop Comparison and New Starting Joint

Each time a complete loop is determined, it is compared with the loop which is currently the best loop. The first comparison criteria is the number of nodes in the loops. If a candidate loop has fewer loop nodes than the current best, the candidate loop becomes the current best loop. If a candidate loop has the same number of nodes as the current best loop, the two loops are compared in terms of the physical length of the expanded loop. The shortest loop becomes the current best loop.

The process of obtaining candidate loops is continued until the tree has been entirely searched. At this point the best candidate loop initiated from the starting joint is known. This loop is stored, and if additional loops are required, a new starting joint is chosen for the second loop. In order to ensure that all joints in the assembly are used, it is important
that the next starting joint is one which was not used in the first loop. However, if all the joints have been used, it doesn't matter where the next loop starts. In this case the loop is started from a joint which is not the starting joint for any other loops. The searching process continues from each new starting joint until the required number of loops (as determined by Equation 2.1) has been obtained.

5.6 Modified AVL Algorithm

The steps of the modified AVL procedure are described as follows:

1. Set up connectivity matrix from network graph or actual assembly
2. Determine the number of required loops (Equation 2.1)
3. Select a starting joint and part for the first loop
4. Determine the current joint's other mating part
5. Down-To-Leaf and Up-To-Branch procedures until candidate loop is found
6. Expand contact joint loop using datum paths and form complete loop
7. Eliminate adjacent geometrically equivalent dimensions by name and location
8. Compare candidate loop with current best loop in terms of
   a) Number of loop nodes (joints and datums)
   b) Physical loop length for loops with equal number of loop nodes
9. Store current best loop from current starting joint
10. Select new starting joint which has not been used (if possible) and starting part
11. Repeat steps 4-10 until the required number of loops have been determined

5.7 Summary

This chapter discussed the AVL prototype and its limitations, and proposed a new method based on recursion. The generation of candidate loops between contact joints was explained and the expansion of these loops using datum paths was described. The procedure for removing adjacent geometrically equivalent dimensions from the loop was also explained. The methods of comparison to select the best set of loops were defined. With the discussion of automatic loop generation theory complete, we are now prepared to present the AutoCAD implementation of the generalized approach to tolerance modeling.
Chapter 6

AUTOCATS IMPLEMENTATION

The generalized modeling procedure discussed in Chapter 3 has been implemented on the AutoCAD system. The assembly tolerance modeler, termed AutoCATS, provides a powerful environment in which a complete kinematic assembly model may be created and prepared for analysis. As illustrated in Figure 6.1, AutoCATS is an extension to AutoCAD and is based on an existing geometric model for a mechanical assembly. A menu driven user interface commands the modeler and communicates with the tolerance model in the AutoCAD database just as AutoCAD itself accesses geometry from its database. The modeler also passes information back and forth with the analysis package CATS, via a model file.

![AutoCAD diagram](image)

Figure 6.1 AutoCATS modeling system

AutoCATS is used to obtain additional kinematic and topological information and associate it with the geometric model. Tolerance modeling functions as well as all the standard AutoCAD functions are available from within AutoCATS. Once the tolerance assembly model is complete, a model file is created and sent to PC-CATS for analysis. Results are passed back to AutoCATS and displayed in the modeler.
The AutoCATS tolerance modeler was implemented with the following goals in mind:

1. Graphical creation of assembly tolerance model
2. Automatic generation of vector loops
3. Development of intelligent modeler
   - automatic identification of dependent variables
   - extensive error checking to prevent erroneous models
4. Complete integration with 2D PC-CATS analysis package

6.1 Menus

The utility of any application is based on its ease of use and flexibility. Therefore, one of the goals of this thesis was developing a tolerance modeler which was powerful, yet simple and intuitive. A menu driven interface, icons, default selections and graphical feedback help provide such an environment.

The AutoCATS assembly tolerance modeler is initiated by selecting the CATS option from AutoCAD’s main menu. This brings up a set of menus unique to the CATS modeler. A separate menu is available for working with the different CATS objects: parts, joints, loops, feature controls and specifications. The menus are very parallel; all having same basic structure with NEW, EDIT and DELETE functions for each CATS object. A display menu is used to toggle the visibility of the CATS objects. All menus contain a list of other menus on the same level so that menu changes can be made without returning to the CATS main menu. A CATS model file can be generated from any menu and the complete set of AutoCAD functions are available from any menu. Figure 6.2 describes AutoCATS menu structure.
The new AutoCATS implementation is very graphically oriented. Datum, joint and feature control types may all be chosen from icon menus, which have a graphical representation of the object to be created. Although command line input is always available, the modeler has been developed so that the tolerance model may be created entirely from graphical and menu selections, without a single typed response.

The CATS menus call programming functions written in AutoLISP, AutoCAD's application programming language.

### 6.2 AutoLISP

AutoLISP is the vehicle used to develop the tolerance modeling extension. It is an application programming language resident in AutoCAD and allows development of custom applications. AutoLISP provides a programming function set which is perhaps not as extensive as some CAD systems' application languages, however, AutoLISP is very flexible and allows full access to AutoCAD's menus and database. Since the AutoLISP language is fairly simple it can be learned quickly, and a library of higher level functions can be built up relatively quickly and easily.

Part of the power of AutoLISP lies in its ability to create and manipulate lists of data. A list is a sequence of data elements, not necessarily homogeneous, in which data
can be accessed individually and collectively. AutoLISP lists are dynamically linked so the size of a list never has to be defined. This is an advantage of lists versus arrays.

Database information is stored in a list format and AutoLISP is used to retrieve and store the assembly tolerance model information just as AutoCAD does for its own entities. Under AutoLISP control AutoCATS defines a hierarchical assembly model which is based on AutoCAD's foundation for data structures: blocks and attributes.

6.3 Data Structure

Data objects are used to store the information obtained from the modeling process in the AutoCAD database. AutoCAD allows custom datatypes through the use of blocks and attributes. An attribute can store one element of information. Several attributes may be grouped together forming a block which may be accessed as a single object. Since the structure of the blocks is programmer controlled, a unique template may be defined for each CATS object.

The following sections describe the data requirements for each of the CATS objects. Many slots in the data structure are of a "handle" type, which is a pointer to an actual entity. For example, a joint points to the first datum of datum path back to the DRF on each part. Since the joint points to two parts, any part information needed can be accessed from the part, instead of storing the extra information with the joint. This also sets the foundation for geometric associativity, a topic for future research.

6.3.1 Datum Object

The structure of a datum reference frame object include its global location, two orthogonal datum axes, and a unique part name. The part name is displayed with the DRF symbol.

- next datum( datum handle, nil for DRF)
- datum location (3D point handle)
- datum axis1 (3D unit vector handle)
- datum axis2 (3D unit vector handle)
- part name (8 character string)
The attributes of a feature datum are identical to datum reference frames, however the part name is not displayed as with a DRF, and the feature datum points to another datum which eventually points to a DRF. The next datum for a DRF is always nil.

- next datum (datum handle)
- datum location (3D point handle)
- datum axis1 (3D unit vector handle)
- datum axis2 (3D unit vector handle)
- part name (8 character string)

6.3.2 Joint Object

Joint information includes the joint’s global location, vectors defining the orientation of the joint axes, the handle of the next datum on the path back to the DRF of each mating part and an integer joint name.

- joint location (3D point handle)
- orientation vector1 (3D unit vector handles)
- orientation vector2 (3D unit vector handles)
- datum path on first part (handle of next datum on path)
- datum path on second part (handle of next datum on path)
- joint name (integer)

6.3.3 Dimension Vector Object

A dimension vector is defined between two nodes which may be either joints or datums. Upper and lower tolerances associated with the vector’s length are also stored.

- joint or datum (handle)
- joint or datum (handle)
- maximum tolerance (real)
- minimum tolerance (real)
6.3.4 Loop Object

The loop object consists of several dimension vectors which form an open or closed loop. The loop has a unique name and tolerances on the angles between vectors.

- loop name (8 char string)
- dimvecs (list of dimension vector handles)
- angular tolerances (list of rotational tolerances between vectors)

6.3.5 Feature Control Object

Feature controls are applied at a joint and include a tolerance band as outlined by ANSI Y14.5. Certain feature controls are directly associated with a single geometric dimension, for example the length of a contact axis. Characteristic lengths are used in these cases.

- joint (handle)
- tolerance width(real)
- characteristic length 1 (real)
- characteristic length 2 (real)

6.3.6 Data Structure

The tolerance model objects outlined in the previous sections are combined into a data structure which represents the entire assembly. The hierarchical tolerance model begins at the CAD geometry level upon which datums are established. Joints are based on datums and dimension vectors are based on joints. Finally, loops are formed from dimension vectors yielding a complete model in the form required by the CATS analysis package. The assembly tolerance model data structure is summarized in Figure 6.3.
6.4 Comparison to Previous Prototype

The current AutoCATS implementation developed by the author is quite different from its prototype ancestor. The prototype modeler attempts to create the assembly model all at once, starting at the loop level and defining joints and feature controls in the process. The new implementation follows a more systematic and hierarchical approach and conforms to the generalized procedure outlined in Chapter 3. Parts and datums are created first, providing a foundation for datum paths and joints. Vector loops are defined next, based on the joints. Finally, feature controls are added which are also linked to individual joints. Figure 6.4 illustrates a comparison between the prototype and the generalized approach.
The new implementation also has many desirable features developed since the prototype. The following table compares the capabilities of the prototype modeler with the author's new AutoCATS implementation.

Table 6.1 Capability of prototype versus new AutoCATS implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>Prototype</th>
<th>New Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Icon Menus</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Default choice options</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>User specified part names</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>DOF analysis</td>
<td>basic, no directions</td>
<td>includes DOF directions</td>
</tr>
<tr>
<td>Automatic contact joint paths</td>
<td>tabular results</td>
<td>internal</td>
</tr>
<tr>
<td>Complete automatic loop generation</td>
<td>no</td>
<td>graphical results</td>
</tr>
<tr>
<td>Automatic identification of dependencies</td>
<td>no</td>
<td>graphical results</td>
</tr>
<tr>
<td>Elimination of equivalent dimensions</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Duplicate vector mapping</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

6.5 Geometric Associativity

The assembly tolerance model discussed previously is based on a geometric model of a mechanical assembly. Sometimes changes are made to the CAD model during the
engineering design process. The tolerance model must also include these changes in geometry in order for an accurate tolerance analysis to be performed. The tolerance model would become even more powerful if it could be linked to the geometric model in such a way that the tolerance model was automatically updated any time the geometry was modified. The concept of attaching an external application model to a geometric model in this manner is called geometric associativity.

Most CAD systems do not have the inherent capability of associating a custom application model to a geometric model. Preliminary tests were performed to evaluate the associativity possibilities for AutoCAD and a sample model written to see what would be required. Although AutoCAD was found to have the necessary foundation and a data structure established to support associativity, the actual implementation of geometric associativity was beyond the scope of this thesis and is a recommendation for future research.

6.6 Summary

This chapter outlined the implementation of the generalized modeling approach on the AutoCAD system. The tolerance model data objects and hierarchy were introduced and a comparison made with previous modelers. Finally, the role of geometric associativity in the modeling process was discussed.
Chapter 7

CASE STUDIES

One of the best ways to demonstrate the robustness of the modeling method and complete automatic loop generation is by studying several example problems. Four assemblies are presented in this chapter. The first is a clutch which permits rotation in only one direction. A ratchet and pawl assembly is considered next. The third assembly is a tightening mechanism used to hold a magnetic tape reel on a tape drive. Finally, a linear positioner is discussed. The first three assemblies are 2-D and the modeling is complete. The linear positioner is a 3-D assembly and is used to demonstrate the flexibility of the datum path concept and complete loop generation. However, it also introduces some of the complexities of 3-D modeling and in the automatic identification of 3-D dependent variables. In each case study, the AutoCATS modeler was used and the set of required loops was determined automatically.

7.1 One-way Clutch

The one-way clutch illustrated in Figure 7.1 is a common device used to start most gas lawn mowers. The hub is attached to the driveshaft of the engine and the ring contains the rollers which slide between the hub and the ring. The clutch locks when the outer ring in rotated clockwise relative to the hub so that torque may be applied to the shaft to start the engine. When the engine starts, the hub rotates faster than the ring and the clutch unlocks, since the hub rotates clockwise relative to the ring.

![Figure 7.1 One-way clutch assembly](image-url)
The assembly angle $\phi$ controls the performance of the clutch. If $\phi$ is too large, the clutch never locks since when the ring rotates, the rollers just slip and are not wedged into the hub. Conversely, if $\phi$ is too small, the clutch jams and will not unlock because the rollers wedge into the hub too tightly. Thus there is a range of $\phi$ in which the clutch will operate correctly. The range for this assembly is from $5^\circ$ to $9^\circ$, or in other words $7^\circ \pm 2^\circ$.

Several parameters influence the value of the angle $\phi$ and whether or not it lies within the acceptable range: the roller radii "a", ring radius "b", and the hub width "c". Since these dimensions may be manufactured independently, the designer can specify the parts' nominal dimensions so that the nominal assembly angle is $7^\circ$. However, cumulative variations in part dimensions affect the assembly angle and may cause it to exceed the $2^\circ$ tolerance. Thus, the only way of controlling the angle range $\phi$ is by specifying appropriate tolerances on the individual part dimensions which contribute to it.

The AutoCATS modeler is used to develop an assembly model which is used for tolerance analysis to determine if the assembly angle $\phi$ is within the specified range. Figure 7.2 illustrates the datum reference frames for each part in the assembly. Note that the DRF for the ring (the center symbol) and the DRF for the hub (the box symbol) are at the same location.

![Figure 7.2 Datum reference frames for each clutch assembly part](image)

Contact joints are created at part interfaces in the assembly. A cylindrical joint is defined between the ring and a roller. Datum paths for the cylindrical joint go directly back to the datum reference frame of each part and are shown in Figure 7.3.
Contact between the hub and roller requires a cylindrical slider joint. The datum path for the roller part goes directly back to the reference frame as illustrated in Figure 7.4a, while the datum path on the hub requires an intermediate feature datum before reaching the datum reference frame, shown in Figure 7.4b.

Since it would be meaningless to locate the slider joint by a single angled dimension, the path is split into two dimensions: one which is an assembly adjustment in the sliding plane.
and the other a specified design dimension. The last joint is a revolute joint between the hub and the center of the ring. Since the revolute joint and the datum reference frame for the hub and ring are at the same location, each datum path consists of single a dimension of zero length.

The modeler now has all the information it needs to generate a complete set of vector loops automatically. If additional cylindrical and cylindrical slider joints are defined at each roller, the assembly has 9 joints and 6 parts, thus requiring 4 loops \( L = 9 - 6 + 1 \), Equation 2.1. However, since the assembly is symmetrical with four rollers, only a quarter of the clutch needs to be analyzed. Thus with 3 joints and 3 parts, 1 loop is required \( L = 3 - 3 + 1 = 1 \). The loop between contact joints is shown in Figure 7.5a. The datum paths associated with each joint are combined with this initial loop and result in the completed loop, as illustrated in Figure 7.5b.

(a)  
(b)

Figure 7.5 Initial contact joint loop and complete vector loop

For this assembly, the following tolerances are assigned to vectors in the loop:

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Nominal</th>
<th>Tolerance</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector 1</td>
<td>50.8</td>
<td>± 0.025</td>
<td>mm</td>
</tr>
<tr>
<td>Vector 2 &amp; 3</td>
<td>11.43</td>
<td>± 0.01</td>
<td>mm</td>
</tr>
<tr>
<td>Vector 5</td>
<td>27.645</td>
<td>± 0.100</td>
<td>mm</td>
</tr>
</tbody>
</table>

The modeler is able to identify vector 4 as a dependent assembly variable and does not request a tolerance on that dimension.

The assembly was simplified using symmetry. If the symmetrical approach was not used, a similar vector loop would occur at each roller. Either method produces the same results.
Finally, the modeler writes out a model file which contains all assembly information pertinent for tolerance analysis. The model file may be used by the CATS analysis software to determine the actual variation of the assembly angle $\phi$. Additional design iterations may be performed as desired.

### 7.2 Gear and Pawl

The next assembly considered was the gear and pawl setup shown in Figure 7.6. The pawl stops the gear and holds it in a certain angular position. The issue under study was how accurately the gear could be positioned when considering manufacturing variations on component dimensions. The parameters which contribute to the assembly angle $\theta$ are $X$offset "a," $Y$offset "b," Pawl Length "c," Root Radius "d," Tip Radius "e," and Root Angle "f".

![Figure 7.6 Gear and pawl assembly](image)

Using the AutoCATS modeler, datum reference frames are established for the parts Gear, Pawl and Ground, as shown in Figure 7.7.
This assembly has 4 contact joints. A revolute joint is defined between the pawl and the ground. Figure 7.8a shows that the datum path for each part goes directly back to the datum reference frame. Note that the pawl's datum path is of zero length. Another revolute joint is located between the gear and the ground. Similar to the first revolute joint, datum paths go directly back to the datum reference frames of the gear and ground. Note that the datum path on the gear is zero length, as illustrated in Figure 7.8b.

The third and fourth joints are cylindrical sliders between the pawl and the gear. Both joints' datum paths share intermediate feature datums on the way back to the datum reference frames, as shown by Figure 7.9. Note that the datum paths include controlled...
dimensions such as Root Radius and Pawl Length and also adjustable dimensions in their appropriate directions.

Figure 7.9 Paths back to datum reference frames on the gear and pawl

The use of Equation 2.1 indicates that two vector loops are required for this assembly \((L = 4 - 3 + 1 = 2)\). The first loop between contact joints consists of the two revolute joints and one of the sliders, as illustrated by Figure 7.10a. Using datum paths, the initial loop is expanded to include controlled dimensions \((X\text{offset}, Y\text{offset}, \text{Pawl Length, Tip Radius, Root Radius and Root Angle})\) and adjustable assembly variables.

Figure 7.10 First contact joint loop and expanded vector loop
The following tolerances are assigned to vectors and angles in the first loop:

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Nominal</th>
<th>Tolerance</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector 1:</td>
<td>0.3000</td>
<td>± 0.0015</td>
<td>in</td>
</tr>
<tr>
<td>Vector 2:</td>
<td>0.4000</td>
<td>± 0.0015</td>
<td>in</td>
</tr>
<tr>
<td>Vector 3:</td>
<td>0.375</td>
<td>± 0.002</td>
<td>in</td>
</tr>
<tr>
<td>Angle between 3 &amp; 4:</td>
<td>0.0</td>
<td>± 0.8</td>
<td>°</td>
</tr>
<tr>
<td>Vector 5:</td>
<td>0.015</td>
<td>± 0.001</td>
<td>in</td>
</tr>
<tr>
<td>Vector 6:</td>
<td>0.300</td>
<td>± 0.001</td>
<td>in</td>
</tr>
</tbody>
</table>

The modeler identifies the length of vector 4 as a dependent variable and therefore does not request a tolerance for it.

The second loop is between the two cylindrical slider joints at the tip of the pawl. This contact joint loop is expanded to include the cylindrical centerline feature datum at the tip of the pawl and the rectangular feature datum between the two gear teeth. The initial and expanded loops are illustrated in Figure 7.11.

The following tolerances are assigned to vectors and angles in the second loop:

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Nominal</th>
<th>Tolerance</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector 5 &amp; 7:</td>
<td>0.015</td>
<td>± 0.001</td>
<td>in</td>
</tr>
<tr>
<td>Angle between 4 &amp; 8:</td>
<td>62.0</td>
<td>± 0.8</td>
<td>°</td>
</tr>
</tbody>
</table>

Again, the modeler does not require tolerances for the lengths of vectors 4 and 8 because it recognizes them as dependent variables.
A model file is generated which represents the ratchet assembly and is available for CATS analysis.

### 7.3 Tape Drive Locking Hub

In this case study, we model a locking assembly which is used to mount and hold a magnetic tape reel in place on a tape drive hub. Although the locking hub mechanism is a three-dimensional example problem, it is symmetrical and a simplified cross section will be considered for 2-D modeling. Figure 7.12 illustrates the parts in the locking hub assembly.

![Diagram of locking hub assembly](image)

**Figure 7.12 Operation of locking hub tape drive assembly**

The assembly consists of a plunger which slides vertically against a guide attached to the base. The arm slides horizontally on the base and makes contact with the bevelled surface of the plunger. The arm's position is determined by the height of the plunger and as the plunger slides down, it forces the arm and pad outward against the inner surface of the tape reel, locking it in place.

The dimensions which govern the operation of the locking hub assembly are shown in Figure 7.13.

![Dimensions of locking hub assembly](image)

**Figure 7.13 Dimensions for locking hub tape drive assembly**
Some of the dimensions are derived from others for convenience. For example, when dimensions "a," "c" and "d" are specified for the plunger, angle $\theta$ is set. Similarly, dimension "e" is the difference between "f" and "R" for the arm, and "g" is the difference between the arm height "h" and the arm radius "R". These derived dimensions will be used in datum paths.

The AutoCATS modeler is used to identify datum reference frames on each of the parts as shown in Figure 7.14. All DRFs are rectangular type.

![Datum reference frames for locking hub tape drive assembly](image)

**Figure 7.14 Datum reference frames for locking hub tape drive assembly**

Kinematic joints are located at part interfaces in the assembly. Rigid joints are established between the arm and the pad (J0) and between the plunger and the pad (J1). A planar joint is defined between the base and the pad (J2), and a cylindrical slider joint is located between the plunger and the arm (J3). Datum paths associated with the joints are shown in Figure 7.15. Both datum paths for joint J0 are of zero length and the path on the plunger is of zero length for joint J1. Note that all datum paths follow only controlled dimensions and dimensions of kinematic adjustment.

![Datum paths for locking hub assembly](image)

**Figure 7.15 Datum paths for locking hub assembly**
With 4 parts and 4 joints, Equation 2.1 indicates that only one loop is needed to represent the assembly \((L = 4 - 4 + 1 = 1)\). Whenever an assembly requires only one loop, the loop can be determined easily by simply linking the datum paths between joints. Automatic loop generation also confirms the final path as shown by Figure 7.16.

![Diagram](image)

(a)  
(b)  

Figure 7.16 Contact joint loop and expanded loop

The following tolerances are assigned to vectors and angles in the loop:

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Nominal</th>
<th>Tolerance</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector 1:</td>
<td>0.050</td>
<td>± 0.002</td>
<td>in</td>
</tr>
<tr>
<td>Vector 2:</td>
<td>0.488</td>
<td>± 0.004</td>
<td>in</td>
</tr>
<tr>
<td>Vector 4:</td>
<td>0.318</td>
<td>± 0.003</td>
<td>in</td>
</tr>
<tr>
<td>Vector 5:</td>
<td>1.3550</td>
<td>± 0.0015</td>
<td>in</td>
</tr>
<tr>
<td>Angle between 5 &amp; 6:</td>
<td>104.9</td>
<td>± 0.5</td>
<td>°</td>
</tr>
<tr>
<td>Vector 7:</td>
<td>0.060</td>
<td>± 0.002</td>
<td>in</td>
</tr>
<tr>
<td>Vector 8:</td>
<td>0.318</td>
<td>± 0.003</td>
<td>in</td>
</tr>
</tbody>
</table>

AutoCATS identifies vectors 3 and 6 as dependent lengths and does not request a tolerance on these dimensions. The specification of tolerances in the loop concludes the modeling process and a model file is generated which represents the assembly.

### 7.4 Linear Positioner

The final case study is an assembly used for precision linear positioning. The carriage has four wheels attached to one side and two wheels attached to the other side. These wheels make contact with two cylindrical rails so that the carriage may translate horizontally in one direction. Figure 7.17 illustrates the linear positioner with the datum reference frames defined for each part.
The assembly has 6 crossed cylinder joints between the wheels and the rails, 6 revolute joints between the wheels and the carriage, and a single rigid joint connecting the two rails. Figure 7.18a shows the location of the revolute joint between Wheel1 and the Carriage and the associated datum paths.

The other 5 revolute joints are similarly defined, with all datum paths on the carriage coming directly out the front face. Figure 7.18b illustrates the crossed cylinder joint between Wheel1 and Rail1 and the corresponding datum paths. Five more crossed cylinder joints are also defined for the other contact locations between the wheel and rails.
Equation 2.1 indicates that five vector loops are required to represent the assembly (\(L = 13 - 9 + 1 = 5\)). Automatic loop generation is employed to determine the set of loops. The first loop is shown in Figure 7.19.

![Figure 7.19 First loop for linear positioner](image)

The remaining loops are similarly obtained thus demonstrating that the generalized modeling approach and automatic loop generation are valid for 3-D modeling as well. However, as mentioned earlier new issues arise which require special consideration.

In 2-D, each joint can contribute at most one dependent rotation, and all rotations are about a single axis. The only modeling requirement in 2-D is to determine whether or not a rotation between vectors is dependent. In 3-D, a joint may have up to three dependent angles, thus several additional rotations must be included to account for all dependent angles. It is also essential that there are enough loops defined for CATS to solve for all the dependent variables in the assembly. Since Equation 2.1 is based on network topology theory and not kinematic degrees of freedom, it may require modification for use in 3-D to account for all the dependent variables. In other words, more loops may be required in 3-D than Equation 2.1 indicates if the assembly is unconstrained. Further study in this area is suggested for future research.

7.5 Summary of Testing

In addition to the geometric stack assembly and the Watt mechanism, the case studies presented in this chapter verify the robustness of the generalized modeling procedure. Collectively, they also provide a diverse set of assemblies to demonstrate and test automatic loop generation. In every case a correct set of loops was obtained, and
dependent variables were identified correctly, thus proving the value of the methods and algorithms described in this thesis.

However, it should be mentioned that the success of the AVL algorithm depends entirely on the connectivity matrix. Therefore, unless the correct number of joints have been defined for an assembly of parts, there is no guarantee of a correct solution. The degree of freedom check provides valuable assistance in setting up the problem correctly but the final decision remains the designer's.

It is also imperative that assembly geometry be defined exactly with all parts touching in their precise nominal orientation. Inaccurate geometry may appear acceptable to the eye but will cause inexact calculation of joint axes followed by erroneous identification of variable dependencies. Thus assembly models must be created using geometric constructs (such as intersection, tangent, perpendicular, etc.) instead of merely sketching the parts or estimating mating locations.

Despite these precautionary considerations, contributions of this thesis furnish the designer with significantly more design insight than was previously available. As with almost all engineering design tools, the tolerance modeler and automatic loop generation provide valuable assistance as long as their limitations are known and not exceeded.
Chapter 8

CONCLUSIONS AND RECOMMENDATIONS

8.1 Research Contributions

This thesis brings new insight to the current issues in tolerance modeling and analysis of mechanical assemblies. Research presented in this thesis represents a significant step toward automating the creation of assembly tolerance models and the generation of governing equations for tolerance analysis. The author's contributions are listed below and then discussed.

1. Generalized modeling process for graphically creating assembly tolerance model
2. Concept of datum paths for locating joint on both parts
3. Complete automatic generation of vector loops
4. Automatic identification of dependent variables
5. DOF analysis and consistency checks
6. AutoCAD implementation

A generalized method was presented which provides a systematic approach to assembly modeling for tolerance analysis. The method involves setting up new data elements such as datum reference frames, kinematic joints, vector loops or kinematic chains, and feature controls. Loops may be set up manually or automatically. These elements are combined in a hierarchical assembly model which represents the actual mechanical assembly.

The datum path was introduced as a new concept. A kinematic joint is located relative to the datum reference frame on each part and two paths traced through feature datums back to each datum reference frame are stored as joint information. Datum paths require the designer to think in terms of datums and ensure design intent in the assembly model.

The process of generating vector loops has been completely automated. Network graph theory is used to determine a set of paths between contact joints. Datum paths are used to extend these paths to loops that follow only controlled and kinematic dimensions on the assembly model. The removal of redundant dimensions eliminates the introduction of artificial and unnecessary variation into the assembly tolerance model. A new robust
algorithm was developed and implemented in C. Complete automatic loop generation has greatly reduced the possibility for error in the modeling procedure.

Dependent lengths and angles within the loops are now determined automatically. Dependent variable identification is a significant contribution because it prevents designers from assigning tolerances to part dimensions which they cannot directly control. Assigning tolerances to these dimensions may be meaningless since they are not design parameters but instead functions of design parameters and result from adjustments to manufacturing variations at assembly time. However, once the actual dependent variations are known, they can be compared to design specifications in order to verify proper assembly performance.

A degree-of-freedom analysis has been developed which not only a keeps a count of each part's DOF but also considers the part's DOF directions, an essential step in identifying unconstrained assembles and recognizing redundant joints. This enhances the designer's ability to set up an assembly model correctly thus ensuring an accurate connectivity matrix for automatic loop generation.

The generalized approach to tolerance modeling has been implemented on the AutoCAD system and the resulting tolerance modeler provides a simple and intuitive method of creating an accurate assembly model for tolerance analysis. The previous AutoCATS prototype has been completely restructured and rewritten to reflect current research progress. The implementation has been used to model a variety of assemblies.

The topics covered in this thesis pave the way for improved design tools in the design for manufacture process.

8.2 Recommendations for Future Work

Many of the topics in this thesis are broad enough themselves to warrant individual consideration. Future research may be directed toward further study of the following:

1. Further development of editing capabilities in the AutoCATS implementation.
2. Display in the modeler all tolerances which are returned by CATS.
3. Tighter integration of AutoCATS modeler and CATS analysis. Investigate obtaining the interactive information CATS needs from the AutoCATS modeler (process and cost information, equivalent variables, etc.).
4. A study which investigates different sets of valid loops and their effects on assembly tolerances. Different datum reference systems represent different methods of assembly and machine setup.

5. Determination of optimal datum paths. AutoCATS may be used as a design tool to reduce an assembly resultant's sensitivity to manufacturing variations.

6. More complete examination of 3-D joint types and 3-D modeling. Chapter 4 and the linear positioner case study introduced several new issues for 3-D modeling.

7. 3-D dependent variable identification. For sliding joints investigate the selection of one part having a feature to which the joint is attached, while the joint's location on the other mating part may adjust. Dependency can then be determined by checking whether a vector's endpoints are both attached to the same part or whether one end's position can adjust.

8. Further investigation and implementation of geometric associativity.

9. Addition of more thorough testing by name and eliminating testing by location.

10. Feature recognition and "smart" geometry to automatically determine joint type and locations and infer part positions.

11. Incorporate procedures that attach extra information to an existing symbol for a new object instead of creating the same symbol on top of an existing one. The data structure is already established with this in mind.

12. Further development of the DOF analysis to include determination of coincident centers of rotation for multiple slider joint.

13. Instead of checking for the number of nodes and physical length of a loop, test on the number of independent variables as best loop criteria. This minimizes user interaction for automatically generated loops.

14. Add interactive equivalence capability to the modeler allowing the user to graphically link equivalent offset vectors and angles so that the analysis treats and varies them simultaneously in different loops.

15. Further study to verify that the set of loops which result from Equation 2.1 is adequate to solve for all the dependent variables in an assembly. Explore underconstrained assemblies and the number of loops required for representation.

APPENDIX A

Assembly Model Files

Autoloop Input/Output Files

Connectivity Matrices
Stkbks: Model File

FILE>> STEBLKJSJ  
28-JUN-91 : 17:58:0

DESCRIPTION OF DATA FILE:

--------- ASSEMBLY: STKBKJSJ ( ) --------------------------------------------------

DESCRIPTION:

TEX -----------------------------------------------

ANM NAME: 
ANO ASSM NO.: 
DRA DRAW NO.: 
MOD LOOP_3 ( ) LOOP_2 ( ) LOOP_1 ( )

--------- SPECIFICATIONS: -----------------------------------------------

DESCRIPTION:

TEX TYPE REF-NOD/DIM 2ND-NOD/DIM BASIC DIM MAX TOL MIN TOL

--------- PART: GROUND ( ) ----------------------------------------------- MODIFIED: 28-JUN-91

DESCRIPTION:

PNM NAME: 
PNU PART NO.: 
DRA DRAW NO.: 
PDI DIMENSIONS:

COS

--------- PART: BLOCK ( ) ----------------------------------------------- MODIFIED: 28-JUN-91

DESCRIPTION:

TEX -----------------------------------------------

PNM NAME: 
PNU PART NO.: 
DRA DRAW NO.: 
PDI DIMENSIONS:

COS

--------- PART: CYLINDER( ) ----------------------------------------------- MODIFIED: 28-JUN-91

DESCRIPTION:

TEX -----------------------------------------------

PNM NAME: 
PNU PART NO.: 
DRA DRAW NO.: 
PDI DIMENSIONS:

COS

--------- DATUM LIST: -----------------------------------------------

DESCRIPTION:

TEX DATUM NAME TYPE X    Y    Z
DRF DATUM3 ( ) RECTAN( ) 7.4397607 8.2004540 0.000000000( )
DRF DATUM2 ( ) RECTAN( ) 9.2326826 11.683600 0.000000000( )
DRF DATUM1 ( ) CYCLEN( ) 14.059760 26.918694 0.000000000( )
AXI AXIAL( ) 0.0000000 0.0000000 1.000000000( )
DAT DATUM12 ( ) RECTAN( ) 11.344760 8.2004540 0.000000000( )
DAT DATUM11 ( ) RECTAN( ) 11.344760 12.260454 0.000000000( )
DAT DATUM10 ( ) RECTAN( ) 35.564260 12.260454 0.000000000( )
DAT DATUM8 ( ) RECTAN( ) 35.564260 8.2004540 0.000000000( )
DAT DATUM6 ( ) RECTAN( ) 7.4397607 18.248160 0.000000000( )
### JOINT LIST:

<table>
<thead>
<tr>
<th>TEX JOINT NAME</th>
<th>TYPE</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOI JOINT9</td>
<td>EDGSLI</td>
<td>11.344760</td>
<td>12.260454</td>
<td>0.0000000</td>
</tr>
<tr>
<td>AXI</td>
<td>PLANAR</td>
<td>0.2634712</td>
<td>-0.964667</td>
<td>0.0000000</td>
</tr>
<tr>
<td>AXI</td>
<td>NORMAL</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>1.0000000</td>
</tr>
<tr>
<td>JOI JOINT13</td>
<td>EDGSLI</td>
<td>35.564760</td>
<td>18.875454</td>
<td>0.0000000</td>
</tr>
<tr>
<td>AXI</td>
<td>PLANAR</td>
<td>-0.2634712</td>
<td>0.9646672</td>
<td>0.0000000</td>
</tr>
<tr>
<td>AXI</td>
<td>NORMAL</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>1.0000000</td>
</tr>
<tr>
<td>JOI JOINT5</td>
<td>EDGSLI</td>
<td>7.4397607</td>
<td>16.248160</td>
<td>0.0000000</td>
</tr>
<tr>
<td>AXI</td>
<td>PLANAR</td>
<td>1.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>AXI</td>
<td>NORMAL</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>1.0000000</td>
</tr>
<tr>
<td>JOI JOINT7</td>
<td>CYLSLI</td>
<td>15.803940</td>
<td>20.532597</td>
<td>0.0000000</td>
</tr>
<tr>
<td>AXI</td>
<td>PLANAR</td>
<td>0.2634712</td>
<td>-0.964667</td>
<td>0.0000000</td>
</tr>
<tr>
<td>AXI</td>
<td>NORMAL</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>1.0000000</td>
</tr>
<tr>
<td>JOI JOINT4</td>
<td>CYLSLI</td>
<td>7.4397607</td>
<td>26.918694</td>
<td>0.0000000</td>
</tr>
<tr>
<td>AXI</td>
<td>PLANAR</td>
<td>1.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>AXI</td>
<td>NORMAL</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>1.0000000</td>
</tr>
</tbody>
</table>

--- LOOP: LOOP_3 ---

### LOOP:

<table>
<thead>
<tr>
<th>TEX</th>
<th>VECTOR LOOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEX X0</td>
<td>Y0</td>
</tr>
<tr>
<td>INIT</td>
<td>7.4397607</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEX</th>
<th>NODE</th>
<th>PART/#1-#2</th>
<th>BASIC DIM</th>
<th>MAX TOL</th>
<th>MIN TOL</th>
<th>DEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOI JOINT5</td>
<td>15.276134</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZRO</td>
<td>8.6705335</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIM BLOCK/5-7</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JOI JOINT7</td>
<td>90.000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZRO</td>
<td>6.6200000</td>
<td>0.2000000</td>
<td>-0.2000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIM CYLINDER/1-7</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOD DATUM1</td>
<td>74.723865</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZRO</td>
<td>6.6200000</td>
<td>0.2000000</td>
<td>-0.2000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIM CYLINDER/1-4</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOD JOINT4</td>
<td>90.000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZRO</td>
<td>18.7182400</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIM GROUND/3-4</td>
<td>-180.000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOD DATUM3</td>
<td>10.047796</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZRO</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIM GROUND/3-5</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOD JOINT5</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Description:

**LOOP: LOOP_2**

<table>
<thead>
<tr>
<th>TEXT</th>
<th>VECTOR LOOP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XO</strong></td>
<td><strong>YO</strong></td>
</tr>
<tr>
<td><strong>INT</strong></td>
<td>11.344760( )</td>
</tr>
</tbody>
</table>

**NODE** | **PART/1-2** | **BASIC DIM** | **MAX TOL** | **MIN TOL** | **DEP** |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NOD JOINT2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZRO</strong></td>
<td>-164.7238( )</td>
<td>0.0000000( )</td>
<td>0.000000( )</td>
<td>0.000000( )</td>
<td><strong>DEP</strong></td>
</tr>
<tr>
<td><strong>DIM BLOCK/2-9</strong></td>
<td>( )</td>
<td>2.1894369( )</td>
<td>0.0000000( )</td>
<td>0.000000( )</td>
<td><strong>DEP</strong></td>
</tr>
<tr>
<td><strong>NOD DATUM2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZRO</strong></td>
<td>179.999999( )</td>
<td>0.0000000( )</td>
<td>0.000000( )</td>
<td>0.000000( )</td>
<td>( )</td>
</tr>
<tr>
<td><strong>DIM BLOCK/2-13</strong></td>
<td>( )</td>
<td>27.296540( )</td>
<td>0.0000000( )</td>
<td>0.000000( )</td>
<td><strong>DEP</strong></td>
</tr>
<tr>
<td><strong>NOD JOINT13</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZRO</strong></td>
<td>-105.2761( )</td>
<td>0.0000000( )</td>
<td>0.000000( )</td>
<td>0.000000( )</td>
<td><strong>DEP</strong></td>
</tr>
<tr>
<td><strong>DIM GROUND/8-13</strong></td>
<td>( )</td>
<td>10.675000( )</td>
<td>0.1250000( )</td>
<td>-0.125000( )</td>
<td>( )</td>
</tr>
<tr>
<td><strong>NOD DATUM8</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZRO</strong></td>
<td>-179.9929( )</td>
<td>0.0000000( )</td>
<td>0.000000( )</td>
<td>0.000000( )</td>
<td>( )</td>
</tr>
<tr>
<td><strong>DIM GROUND/8-10</strong></td>
<td>( )</td>
<td>4.0600000( )</td>
<td>0.1500000( )</td>
<td>-0.150000( )</td>
<td>( )</td>
</tr>
<tr>
<td><strong>NOD DATUM10</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZRO</strong></td>
<td>89.992943( )</td>
<td>0.0000000( )</td>
<td>0.000000( )</td>
<td>0.000000( )</td>
<td>( )</td>
</tr>
<tr>
<td><strong>DIM GROUND/9-10</strong></td>
<td>( )</td>
<td>24.219500( )</td>
<td>0.3500000( )</td>
<td>-0.350000( )</td>
<td>( )</td>
</tr>
<tr>
<td><strong>NOD JOINT29</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Description:

**LOOP: LOOP_1**

<table>
<thead>
<tr>
<th>TEXT</th>
<th>VECTOR LOOP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XO</strong></td>
<td><strong>YO</strong></td>
</tr>
<tr>
<td><strong>INT</strong></td>
<td>11.344760( )</td>
</tr>
</tbody>
</table>

**NODE** | **PART/1-2** | **BASIC DIM** | **MAX TOL** | **MIN TOL** | **DEP** |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NOD JOINT9</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZRO</strong></td>
<td>-90.0000000( )</td>
<td>0.0000000( )</td>
<td>0.0000000( )</td>
<td>0.0000000( )</td>
<td><strong>DEP</strong></td>
</tr>
<tr>
<td><strong>DIM GROUND/9-12</strong></td>
<td>( )</td>
<td>4.060000000( )</td>
<td>0.150000000( )</td>
<td>-0.150000000( )</td>
<td>( )</td>
</tr>
<tr>
<td><strong>NOD DATUM12</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZRO</strong></td>
<td>-90.0000000( )</td>
<td>0.0000000( )</td>
<td>0.0000000( )</td>
<td>0.0000000( )</td>
<td>( )</td>
</tr>
<tr>
<td><strong>DIM GROUND/3-12</strong></td>
<td>( )</td>
<td>3.905000000( )</td>
<td>0.125000000( )</td>
<td>-0.125000000( )</td>
<td>( )</td>
</tr>
<tr>
<td><strong>NOD DATUM3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZRO</strong></td>
<td>-90.0000000( )</td>
<td>0.0000000( )</td>
<td>0.0000000( )</td>
<td>0.0000000( )</td>
<td>( )</td>
</tr>
<tr>
<td><strong>DIM GROUND/3-5</strong></td>
<td>( )</td>
<td>10.047706000( )</td>
<td>0.000000000( )</td>
<td>0.000000000( )</td>
<td><strong>DEP</strong></td>
</tr>
<tr>
<td><strong>NOD JOINT5</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZRO</strong></td>
<td>-164.7238000( )</td>
<td>0.0000000( )</td>
<td>0.0000000( )</td>
<td>0.0000000( )</td>
<td><strong>DEP</strong></td>
</tr>
<tr>
<td><strong>DIM BLOCK/2-5</strong></td>
<td>( )</td>
<td>6.805000000( )</td>
<td>0.075000000( )</td>
<td>-0.075000000( )</td>
<td>( )</td>
</tr>
<tr>
<td><strong>NOD DATUM2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZRO</strong></td>
<td>90.0000000( )</td>
<td>0.0000000( )</td>
<td>0.0000000( )</td>
<td>0.0000000( )</td>
<td>( )</td>
</tr>
<tr>
<td><strong>DIM BLOCK/2-9</strong></td>
<td>( )</td>
<td>2.1894369000( )</td>
<td>0.000000000( )</td>
<td>0.000000000( )</td>
<td><strong>DEP</strong></td>
</tr>
<tr>
<td><strong>NOD JOINT9</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Stkblks: autoloop.in

0
DATUM PATHS
0.0000010000000000
3
5
8
0,1,2,11.344760733240447,12.260454081850771,0.000000000000000
6
8,5
1,1,2,35.564760733240455,18.875454081850769,0.000000000000000
6
11,10,9,8,5
2,1,2,7.439760733240457,18.248160995556308,0.000000000000000
6
5
3,0,1,15.803940452536956,20.532597363976883,0.000000000000000
7
12,6
4,0,2,7.439760733240459,26.918694537274110,0.000000000000000
7
5
5,2,7.439760733240458,8.200454081850772,0.000000000000000
6,1,9.232682635024846,11.683600381615586,0.000000000000000
7,0,14.059760733240459,26.918694537274110,0.000000000000000
8,2,11.344760733240458,8.200454081850772,0.000000000000000
9,2,11.344760733240447,12.260454081850771,0.000000000000000
10,2,35.564260733240452,12.260454081850771,0.000000000000000
11,2,35.564760733240455,8.200454081850772,0.000000000000000
12,1,7.439760733240458,18.248160995556315,0.000000000000000
Stkblks: Connectivity Matrix

<table>
<thead>
<tr>
<th></th>
<th>J0</th>
<th>J1</th>
<th>J2</th>
<th>J3</th>
<th>J4</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0 (Cylinder)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P1 (Block)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>P2 (Ground)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Stkblks: autoloop.out

0,2
8,2
5,2
2,1
6,1
0,1
***
0,1
6,1
1,2
11,2
10,2
0,2
***
2,1
3,0
7,0
4,2
5,2
2,2
***
EOF
Clutch: Model File

FILE>> CLUTCHJ  28-JUN-91  18:25:0
DESCRIPTION OF DATA FILE:
------------- ASSEMBLY: CLUTCHJ  ( ) ___________________________________________
DESCRIPTION:
TEX
------------- NAME: LOOP_1  ( )
MOD
==== SPECIFICATIONS: -------------------------------------------------------------
DESCRIPTION:
TEX TYPE REF-NOD/DIM 2ND-NOD/DIM BASIC DIM MAX TOL MIN TOL
------------- PART: HUB  ( ) MODIFIED: 28-JUN-91
DESCRIPTION:
TEX
------------- NAME:
PNU PART NO.: DRA DRAW NO.: PDI DIMENSIONS:
COS
------------- PART: RING  ( ) MODIFIED: 28-JUN-91
DESCRIPTION:
TEX
------------- NAME:
PNU PART NO.: DRA DRAW NO.: PDI DIMENSIONS:
COS
------------- PART: ROLLER  ( ) MODIFIED: 28-JUN-91
DESCRIPTION:
TEX
------------- NAME:
PNU PART NO.: DRA DRAW NO.: PDI DIMENSIONS:
COS
--------- DATUM LIST: -----------------------------------------------------------
DESCRIPTION:
TEX DATUM NAME TYPE X Y Z
DRF DATUM3  ( ) RECTAN( ) 50.000000 50.000000 0.000000( )
DRF DATUM2  ( ) CYLCEN( ) 50.000000 50.000000 0.000000( )
AXI        AXIAL ( ) 0.000000 0.000000 1.000000( )
DRF DATUM1  ( ) CYLCEN( ) 54.810537 89.075000 0.000000( )
AXI        AXIAL ( ) 0.000000 0.000000 1.000000( )
DRF DATUM6  ( ) RECTAN( ) 50.000000 77.645000 0.000000( )
--------- JOINT LIST: -----------------------------------------------------------
DESCRIPTION:
TEX JOINT NAME TYPE X Y Z
JOI JOINT4  ( ) REVOLU( ) 50.000000 50.000000 0.000000( )
AXI        AXIAL ( ) 0.000000 0.000000 1.000000( )
JOI JOINT5  ( ) CYLIND( ) 56.207145 100.41935 0.000000( )
AXI        AXIAL ( ) 0.000000 0.000000 1.000000( )
JOI JOINT7  ( ) CYLSLI( ) 54.810537 77.645000 0.000000( )
AXI        FLNAR( ) -1.000000 0.000000 0.000000( )
AXI        NORMAL( ) 0.000000 0.000000 1.000000( )
### LOOP: LOOP_1

<table>
<thead>
<tr>
<th>Description:</th>
<th>PROJECT_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table:</td>
<td>PROJECT_1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEXT</th>
<th>VECTOR LOOP</th>
<th>VALUE</th>
<th>VALUE</th>
<th>VALUE</th>
<th>VALUE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEX</td>
<td>X0</td>
<td>Y0</td>
<td>Z0</td>
<td>XANG0</td>
<td>YANG0</td>
<td>ZANG0</td>
</tr>
<tr>
<td>INIT</td>
<td>50.000000</td>
<td>50.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>TEX</td>
<td>NODE</td>
<td>PART/#1-#2</td>
<td>BASIC DIM</td>
<td>MAX TOL</td>
<td>MIN TOL</td>
<td>DEP</td>
</tr>
<tr>
<td>NOD JOINT4</td>
<td>ZRO</td>
<td>82.981610</td>
<td>0.000000</td>
<td>0.000000</td>
<td>DEP</td>
<td></td>
</tr>
<tr>
<td>NOD JOINT5</td>
<td>ZRO</td>
<td>-180.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>(</td>
<td></td>
</tr>
<tr>
<td>NOD DATUM1</td>
<td>ZRO</td>
<td>7.0183898</td>
<td>0.000000</td>
<td>0.000000</td>
<td>DEP</td>
<td></td>
</tr>
<tr>
<td>NOD JOINT7</td>
<td>ZRO</td>
<td>-90.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>(</td>
<td></td>
</tr>
<tr>
<td>NOD DATUM6</td>
<td>ZRO</td>
<td>90.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>DEP</td>
<td></td>
</tr>
<tr>
<td>NOD JOINT4</td>
<td>ZRO</td>
<td>27.645000</td>
<td>0.100000</td>
<td>-0.100000</td>
<td>(</td>
<td></td>
</tr>
</tbody>
</table>

---

### END:
Clutch: autoloop.in

0
DATUM_PATHS
0.000010000000000
3
3
4
0,2,1,50.000000000000000,50.000000000000000,0.000000000000000
3
4
1,0,1,56.207145692527291,100.419354838709907,0.000000000000000
0
5
4
2,0,2,54.810537911708657,77.645000000000010,0.000000000000000
5
6,3
3,2,50.000000000000000,50.000000000000000,0.000000000000000
4,1,50.000000000000000,50.000000000000000,0.000000000000000
5,0,54.810537911708650,89.075000000000015,0.000000000000000
6,2,50.000000000000000,77.645000000000010,0.000000000000000
0
### Clutch: Connectivity Matrix

<table>
<thead>
<tr>
<th></th>
<th>J0</th>
<th>J1</th>
<th>J2</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0  (Roller)</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P1  (Ring)</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>P2  (Hub)</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### Clutch: autoloop.out

0,1  
1,0  
5,0  
2,2  
6,2  
0,2  
***  
EOF
### Ratchet: Model File

**FILE**> SANDIAJ

---

**DESCRIPTION OF DATA FILE:**

--------------- Assembly: SANDIAJ ( ) Modified: 29-JUN-91

**DESCRIPTION:**

**TEX**

**ANM** NAME:

**ANO ASSM NO.**

**DRA DRAW NO.**

**MOD**

--- Specifications: ( ) Modified: 29-JUN-91

**DESCRIPTION:**

**TEX TYPE** REP-NOD/DIM 2ND-NOD/DIM BASIC DIM MAX TOL MIN TOL

--------------- Part: GROUND ( ) Modified: 29-JUN-91

**DESCRIPTION:**

**TEX**

**PNM** NAME:

**PNU PART NO.**

**DRA DRAW NO.**

**FDI DIMENSIONS**

**COS**

--------------- Part: PAWL ( ) Modified: 29-JUN-91

**DESCRIPTION:**

**TEX**

**PNM** NAME:

**PNU PART NO.**

**DRA DRAW NO.**

**FDI DIMENSIONS**

**COS**

--------------- Part: GEAR ( ) Modified: 29-JUN-91

**DESCRIPTION:**

**TEX**

**PNM** NAME:

**PNU PART NO.**

**DRA DRAW NO.**

**FDI DIMENSIONS**

**COS**

--------------- DATUM LIST: Modified: 29-JUN-91

**DESCRIPTION:**

**TEX DATUM NAME**

<table>
<thead>
<tr>
<th>TYPE</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRF DATUM2</td>
<td>( ) RECTAN</td>
<td>1.4000000</td>
<td>1.0000000</td>
</tr>
<tr>
<td>DRF DATUM3</td>
<td>( ) CYLCEN</td>
<td>1.4000000</td>
<td>1.3000000</td>
</tr>
<tr>
<td>AXI</td>
<td>AXIAL ( )</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>DRF DATUM1</td>
<td>( ) CYLCEN</td>
<td>1.0000000</td>
<td>1.0000000</td>
</tr>
<tr>
<td>AXI</td>
<td>AXIAL ( )</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>DAT DATUM7</td>
<td>( ) RECTAN</td>
<td>1.0914723</td>
<td>1.3638128</td>
</tr>
<tr>
<td>DAT DATUM6</td>
<td>( ) CYLCEN</td>
<td>1.1121058</td>
<td>1.3843621</td>
</tr>
<tr>
<td>AXI</td>
<td>AXIAL ( )</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
</tbody>
</table>

--------------- JOINT LIST: Modified: 29-JUN-91

**DESCRIPTION:**

**TEX JOINT NAME**

<table>
<thead>
<tr>
<th>TYPE</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOI JOINT5</td>
<td>( ) REVOLU</td>
<td>1.4000000</td>
<td>1.3000000</td>
</tr>
<tr>
<td>AXI</td>
<td>AXIAL ( )</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>JOI JOINT4</td>
<td>( ) REVOLU</td>
<td>1.0000000</td>
<td>1.0000000</td>
</tr>
<tr>
<td>AXI</td>
<td>AXIAL ( )</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>JOI JOINT9</td>
<td>( ) CYLSLI</td>
<td>1.0975586</td>
<td>1.3880196</td>
</tr>
<tr>
<td>AXI</td>
<td>PLANAR ( )</td>
<td>0.9698159</td>
<td>-0.263837</td>
</tr>
<tr>
<td>AXI</td>
<td>NORMAL ( )</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>JOI JOINT8</td>
<td>( ) CYLSLI</td>
<td>1.1153890</td>
<td>1.3697258</td>
</tr>
<tr>
<td>AXI</td>
<td>PLANAR ( )</td>
<td>0.2400049</td>
<td>-0.970771</td>
</tr>
<tr>
<td>AXI</td>
<td>NORMAL ( )</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
</tbody>
</table>
--- LOOP: LOOP_2  ---

**DESCRIPTION:**

--- LOOP: LOOP_1  ---

**DESCRIPTION:**
Ratchet: autoloop.in

0
DATUM PATHS
0.00001000000000
3
4
5
0,1,2,1.40000000000000,1.30000000000000,0.00000000000000
5
4
1,0,2,1.00000000000000,1.00000000000000,0.00000000000000
6
4
2,1,0,1.097558620412412,1.388019671485225,0.00000000000000
8,5
7,6
3,1,0,1.115389086582691,1.369725831752877,0.00000000000000
8,5
7,6
4,2,1.40000000000000,1.00000000000000,0.00000000000000
5,1,1.40000000000000,1.30000000000000,0.00000000000000
6,0,1.00000000000000,1.00000000000000,0.00000000000000
7,0,1.091472379153194,1.363812878441222,0.00000000000000
8,1,1.112105860362622,1.384362102643629,0.00000000000000
0
Ratchet: Connectivity Matrix

<table>
<thead>
<tr>
<th></th>
<th>J0</th>
<th>J1</th>
<th>J2</th>
<th>J3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0 (Gear)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P1 (Pawl)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P2 (Ground)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Ratchet: Autoloop.out

0, 2
4, 2
1, 0
7, 0
2, 1
8, 1
0, 1
***
2, 0
7, 0
3, 1
8, 1
2, 1
***
EOF
Hublock: autoloop.in

0
DATUM_Paths
0.00001000000000
4
4
7
0,2,3,2.063230032118370,0.98800000000000,0.00000000000000
5
4
1,0,1,0.25000000000000,0.69500000000000,0.00000000000000
7
6
2,1,3,2.113230032118373,0.50000000000000,0.00000000000000
6
8,4
3,0,2,1.687255935742299,1.003459759033620,0.00000000000000
9,7
10,5
4,3,2.063230032118373,0.98800000000000,0.00000000000000
5,2,2.063230032118370,0.98800000000000,0.00000000000000
6,1,0.25000000000000,0.50000000000000,0.00000000000000
7,0,0.25000000000000,0.69500000000000,0.00000000000000
8,3,2.113230032118374,0.98800000000000,0.00000000000000
9,0,1.60500000000000,0.69500000000000,0.00000000000000
10,2,1.745230032118374,0.98800000000000,0.00000000000000
0
Hublock: Connectivity Matrix

<table>
<thead>
<tr>
<th></th>
<th>J0</th>
<th>J1</th>
<th>J2</th>
<th>J3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0  (Plunger)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P1  (Base)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>P2  (Arm)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P3  (Pad)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Hublock: Autoloop.out

0, 3
8, 3
2, 1
6, 1
1, 0
9, 0
3, 2
10, 2
0, 2
***
EOF
APPENDIX B

Auto-Loop Generator Function Summary

The automatic vector loop (AVL) generator described in Chapter 5 of this thesis is written in C and consists of the 22 functions described below. The format of the function summary is as follows:

* returned type    function name (argument1 type argument1, arg2 type arg2, ...)
* Description

void main ()
This function is the engine of the autoloop generator. It reads in input file autoloop.in and dynamically allocates arrays of the appropriate size for the problem. It sets up the connectivity matrix and determines the starting joint and part and the goal part and joint. Down_To_Leaf is called and the search begins for candidate loops. When all candidate loops have been explored and the best set of expanded loops has been found, an output file called autoloop.out is generated which contains the best set of loops including joints and datums in the path.

void New_GoalJoint (void)
This function determines a new goal joint to search for in the connectivity matrix. Previous starting joints are kept track of in an array called StartJnts, where the index represents a joint: StartJnts[4] represents joint 4. Joints which have been used in a loop are indentified with a 1 and joints used as starting joints for prior loops are identified with a -1.

void Down_To_Leaf (int Part1, int Joint1, int StartJnt)
This function marches down the virtual tree as it searches for a leaf node (the goal joint). It is recursive and may call itself or Up_To_Branch depending on the new part and joint it determines.

void Up_To_Branch (int Part2, int Joint2, enum boolean AtLeaf)
This function drives up the virtual tree as it searches for a branch (a part with more than two joints). It is recursive and may call itself or Down_To_Leaf depending on the new part and joint it determines.

int Next_Joint (int Part, int JntCount)
This function locates the next joint on the current part and returns an integer identifying this joint.

int Next_Part (int Joint, int CurrentPart)
This function locates the other mating part associated with the current
joint returns an integer identifying this part.

void Print_Incid (void)
This function is prints the incidence matrix to the screen so that the user may follow the path in the matrix. The matrix is filled with zeros except for two 1's in each column (joint) identifying the two rows (parts) which the joint connects. The path through the matrix is shown with -1's.

void Used (void)
This function prints to the screen the stack of parts used and the stack of joints used in the order that they were traversed from the starting node.

void Reset_Array (int *TheArray, int ArrayLength, int TheValue)
This function resets the specified array, 'TheArray', of length 'ArrayLength', filling the array with elements having value 'TheValue'.

void Check_Best (void)
This function compares one candidate loop with the current best loop first in terms of the number of loop nodes (joints and datums) in the expanded loop. The new best loop is the one with the least number of nodes. If the candidate loop and the current best loop have the same number of nodes, they are compared in terms of physical length of the expanded loop. The shortest of the two loops becomes the new current best loop.

void Sort_Loop (int *JntArray, int *PartArray, int ArrayLength)
This function sorts an array resulting in the lowest element at index 0 lowest of its two neighboring elements. The array represents a closed loop and retains the order of nodes in the loop after the sort.

float Phys.Length(int *TempJntsUsed, int *TempPartsUsed, int NumUsedJnts,
int *NumLoopItems)
This function determines the physical length of an expanded candidate loop.

float Dist_Pt_Pt(Point Pnt1, Point Pnt2)
This function calculated the distance between two 3-D points

enum boolean Branch (int Part)
This function returns true if the row 'Part' is a branch, that is, if there is more than one path into and out of the 'Part'

enum boolean Member (int Item, int *ItemsUsed, int NumItems)
This function returns true if 'Item' is an element is 'ItemsUsed'

void One_To_Neg_One (int Part, int Joint)
This function sets the element in row 'Part' and column 'Joint' of matrix 'Incid' to -1

void Neg_One_To_One (int Part, int Joint)
This function sets the element in row 'Part' and column 'Joint' of matrix 'Incid' to +1
int **allocate_2D_int_array (int row, int col)
   This function allocates a 2-D integer array and returns a pointer

void free_2D_int_array (int **pointer)
   This function frees memory a 2-D integer array

float Expand_Line(int Jnt1, int Jnt2, int PartName, int *TempLoop,
   int *NumPathItems)
   This function expands the path between two joints using datum paths.
   The expanded path consists of the first joint followed its datum path
   back to the common DRF. The second joint's datum path is then appended
   to the reversed order followed finally by the second joint.

enum boolean Equal_Points(Point Pnt1, Point Pnt2)
   This function determines if points Pnt1 and Pnt2 are equal and returns
   True if they are.

void Remove_Dup_Paths(int *Path, int *NumPathItems)
   This function removes geometrically equivalent dimensions from a path
   so that unrealistic variation is not introduced into the tolerance
   model. Vectors are removed by name and by location.
REFERENCES

Anderson, Carl (1990), General System for Least Cost Tolerance Allocation in Mechanical Assemblies, M.S. Thesis, Brigham Young University, Provo, UT, August.


Simmons, Andre P. (1990), Automated Vector Loop Generation for Kinematic Models of Mechanical Assemblies, M.S. Thesis, Brigham Young University, Provo, UT, August.


MECHANICAL ENGINEERING

THESIS PROSPECTUS

Name: Glen Crane Larsen

Soc. Sec. No.: 615-05-9890

Address: 15 East, 200 North, Apt. E

Orem, UT 84057

Date: 11-21-90

Specialty Area: Tolerance Analysis in Mechanical Assemblies

Degree: M.S.

Proposed Title:

A Generalized Approach to Kinematic Modeling for Tolerance Analysis of Mechanical Assemblies

- Including Automatic Generation of Vector Loops -

Statement of the problem

Tolerance stackup due to manufacturing variations in mechanical assemblies can be modeled and analyzed using matrix methods. A CAD system is used to extract existing geometric information from an engineering assembly model and obtain the additional data required to define a tolerance model. It is proposed that a generalized method for tolerance modeling be developed which may be implemented on almost any CAD modeler. Part of the general method includes defining a set of vector loops representing the kinematic constraints on the assembly. Network graph theory has been used to determine an initial set of paths between contact joints in the assembly model. It is proposed that this method be expanded to ensure that these paths follow controlled dimensions and datums on the parts in the assembly. The addition of this method will completely automate the process of defining the vector loops used to set up the matrices for analysis.

Analysis and review of previous work in the field

Recent progress in automating the generation of vector loops (for setting up matrix equations) included prototype software which in most cases identified paths between contact joints in an assembly (several counter-examples have been found). Although these contact joint paths greatly aid the engineer in locating vector paths, a significant part of the loop defining process was still manual.

Method to be followed

The research approach is outlined as follows: 1) examine previous modelers and prototypes to determine common and general requirements for tolerance modeling, 2) apply network graph theory to obtain a set of paths between contact joints, 3) use feature datum references to expand the contact joint paths to complete loops which follow only controlled dimensions on the model, 4) automatically identify dependent variables in the completed set of loops, and 5) implement the general modeler using the AutoLISP application language of the AutoCAD system.

Contribution to be made by this thesis

Previous theory used by prototype software in automating loop generation between contact joints will be incorporated into the AutoCAD-CATS modeler. A method will be implemented that allows the complete automation of vector loops and ensures that the loop paths follow controlled dimensions on the engineering model. This research will also include automatic identification of dependent variables in the loops which result from kinematic degrees of freedom in an assembly. A final product of this research will be a complete 2-D modeler which combines many existing modeling concepts and prototypes into practical software.

Delimitations of the problem

Although this research will be developed in a potential 3-D environment and some of the methods derived apply to 3-D, the majority of the thesis discussion is focused on 2-D datatypes, example problems, and modeling techniques.

Signed: Kenneth Wilson
Chairman, Advisory Committee

Date: 11/20/90

Signed: [Signature]

Committee Member

Date: 11/20/90

Signed: [Signature]
Chairman or Graduate Coordinator of Major Department

Date: 11/20/90
Motivation

All manufactured parts are subject to dimensional variations caused by manufacturing process fluctuations. These variations often have adverse consequences. Variations in a single part can cause poor part performance. Variations in several parts can stack together in such a way that the parts may not assemble or function properly. Thus it is important that a design engineer account for these variations during the design process. Typically this is done by assigning tolerances to dimensions on each part, which represent acceptable upper and lower limits on the manufacturing variations.

The proper specification of tolerances on part dimensions requires careful analysis. A designer cannot simply make all tolerances very small in order to insure proper assembly performance because this drives up manufacturing costs. Conversely, if the designer assigns tolerances which are too large, costs also rise due to increased labor required to rework the unsatisfactory assemblies. Figure 1 illustrates the relationship between cost and tolerances.

![Diagram showing cost and tolerances](image)

Figure 1 Relationship Between Cost and Tolerances

Since the cost curves represent entire assembly costs, values on the horizontal axis represent a set of tolerances for the assembly, not just a single tolerance. Thus the goal of the designer should be to assign an optimal set of tolerances in order to minimize the total cost of the assembly, while still insuring satisfactory performance.
Computer Aided Tolerancing

Since tolerance selection has such a significant influence on cost, it would be helpful if the designer had a tool to help make decisions based on the tolerances which influence cost the most. Obviously, if assembly cost is very sensitive to changes in a certain part dimension, it would be advantageous to specify the tolerance on that dimension as "loose" as possible, in order to minimize cost. If tolerances on other part dimensions then need to be "tightened" to ensure assembly performance, tight tolerances are specified on relatively cost-insensitive dimensions. Unfortunately, obtaining these relationships is time-consuming and they are difficult to determine by hand. Computer techniques have thus been employed to model manufacturing variations and perform tolerance analysis.

In order to create a tolerance model for analysis, geometric information about the assembly is required. Since an assembly model has usually been created previously in a Computer-Aided-Design (CAD) database, this information is readily available. However, each part's actual position in the assembly will vary slightly from its ideal position, as manufacturing variations on individual part dimensions cause the assembly to adjust. In order to account for these adjustments, the tolerance model must also include kinematic relationships, in addition to geometric data. Kinematic information is added to the model by creating unique joint objects at the part interfaces, based on the type of mating contact. The joints are then connected in series to form kinematic chains or loops. The loops form the desired set of relationships which reflect how the assembly parts will adjust to small variations. Other constraints, such as ANSI Y14.5 form and feature controls, must also be taken into account. The procedure of obtaining this additional required information is called the tolerance modeling process.

A considerable portion of the designer's time is spent in defining kinematic loops and significant efforts are being focused on generating vector loops automatically. Complete automatic vector loop generation would not only reduce the amount of time required to model an assembly, but improve the model's accuracy as well.

Research Objective

The purpose of the research presented in this thesis is first, to develop a general and systematic approach to the kinematic modeling of manufacturing variations in mechanical assemblies, and second, to completely automate the loop defining process, which is one of the greatest potential sources of error in creating a model for tolerance analysis.
Research Approach

The research approach is outlined as follows: 1) examine previous modelers and prototype systems to determine common and general requirements for tolerance modeling, 2) apply network graph theory to obtain a set of paths between contact joints, 3) use feature datum references to expand the contact joint paths to complete loops which follow only controlled dimensions on the model, 4) automatically identify the dependent variables in the completed set of vector loops, and 5) implement the general modeler using the AutoLISP application language of the AutoCAD system.

Thesis Overview

The next chapter will discuss important background information regarding research related to computer aided tolerance modeling and analysis. Chapter 3 presents a detailed approach to the tolerance modeling process. The implementation of this method with the AutoCAD system is discussed in Chapter 4. Complete automation of kinematic chain or vector loop generation is described in Chapter 5. Chapter 6 introduces several example assemblies to be modeled and analyzed. Conclusions are presented in Chapter 7 along with recommendations for future research.
References


5. Paul, Burton, Kinematics and Dynamics of Planar Machinery, Prentice Hall, 1979


7. Sedgewick, Robert, Algorithms, Addison-Wesley, 1979


FIG. 12. Undriven structural joints (a–f, left to right, top to bottom): revolute joint (1 DOF); prismatic joint (1 DOF); Hooke's joint (2 DOF); cylinder joint (2 DOF); ball and socket joint (3 DOF); planar joint (3 DOF)

FIG. 13. Foundation actuator modules (a–c, left to right): models 1–3

b. M1 In Yoke Branches—Divide M1 into separate halves and place in each branch of the yoke. The result is a compact simple assembly, providing 270° of rotation and somewhat more rugged than a above.

c. M2 Overlapping Links—Continuous rotation of two links as in elbow shown, in base of robot, in forearm, or last driver before the robot end-plate. Not as rugged as a and b above for the same weight. Relatively compact for exceptional dexterity.

d. M3 In Prism Joint—Here, module M3 drives a slider.
FIG. 12. Undriven structural joints (a–f, left to right, top to bottom): revolute joint (1 DOF); prismatic joint (1 DOF); Hooke's joint (2 DOF); cylindrical joint (2 DOF); ball and socket joint (3 DOF); planar joint (3 DOF)

FIG. 13. Foundation actuator modules (a–c, left to right): models 1–3

b. M1 In Yoke Branches—Divide M1 into separate halves and place in each branch of the yoke. The result is a compact simple assembly, providing 270° of rotation and somewhat more rugged than a above.

c. M2 Overlapping Links—Continuous rotation of two links as in elbow shown, in base of robot in forearm, or last driver before the robot end-plate. Not as rugged as a and b above for the same weight. Relatively compact for exceptional dexterity.

d. M3 In Prism Joint—Here, module M3 drives a slider

Tesar and Butler: A Generalized Modular Architecture for Robot Structures