2003-05-07

Designing Active Smart Features to Provide Nesting Forces in Exactly Constrained Assemblies

Eric Pearce
Brigham Young University - Provo

DESIGNING ACTIVE SMART FEATURES TO PROVIDE NESTING FORCES IN EXACTLY CONSTRAINED ASSEMBLIES

by

Eric L. Pearce

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Brigham Young University

August 2003

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Eric L. Pearce

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

Date	Alan R. Parkinson, Chair
Date	Kenneth W. Chase
Date	Spencer P. Magleby

BRIGHAM YOUNG UNIVERSITY

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

Date	Alan R. Parkinson Chair, Graduate Committee

Accepted for the Department

Brent L. Adams
Graduate Coordinator

Accepted for the College

Douglas M. Chabries
Dean, College of Engineering and Technology

ABSTRACT

DESIGNING ACTIVE SMART FEATURES TO PROVIDE NESTING FORCES IN EXACTLY CONSTRAINED ASSEMBLIES

Eric L. Pearce

Department of Mechanical Engineering

Master of Science

Ever since the design and manufacture of products moved from the craftsman era where individual craftsman designed and manufactured the entire product, to the mass production era, where skilled laborers were crafting interchangeable parts or in some cases single features on interchangeable parts, variation in assemblies has been a major concern to designers, manufacturers, and in a more subtle way, customers. Variation, in the end, affects quality, performance and the cost of products. One particular type of design that is particularly robust to variation is an exactly constrained design.

Several researchers have recently explored the topic of exact constraint design. An exactly constrained design is one in which each degree of freedom is constrained by a single constraint until the desired degrees of freedom for the design is attained. One attractive advantage of exactly constrained designs is that they are robust to variation.

However, exactly constrained designs often require nesting forces to maintain the configuration of the design. This research develops a method for designing features that will supply robust nesting forces such that the advantages of the exactly constrained design are preserved.

The method developed in this work takes advantage of a proven method for tolerance analysis and enhances this method to include the analysis of these features that supply nesting forces. Along with the enhancement, principles are developed that aid this analysis. All the examples provided in this work are verified using comparisons to Monte Carlo simulations. The comparisons show good results, typically less than 2% difference from the Monte Carlo simulations, verifying that this method accurately predicts variation and allows for the robust design of features that supply the nesting forces in exactly constrained assemblies.

ACKNOWLEDGMENTS

I would first like to thank my lovely wife, Harmony. Without her faith in me and her unwavering support in returning to school full-time, I would have never made it. Our children, Rachel, Rebecca, and Nathan also deserve my thanks, as they have always believed their Dad could do it. I also owe very much to my advisor, Dr. Alan Parkinson. He has supported me with intellect, guidance, and the occasional shove and obviously without him, this work would have never been accomplished. Other professors that deserve my thanks for the advice, mentorship and counsel they gave me have been Dr. Ken W. Chase, Dr. Paul F. Eastman, and Dr. Robert H. Todd. I would like to express thanks to my parents and my in-laws who have also supported and believed in my dreams and me. I would like to thank the National Science Foundation who has supported this research with grant DMI 0084880. And finally, I must express my greatest thanks to my Heavenly Father, for the inspiration, skills and talents that have brought me to this point.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Motivation for Research

Ever since the design and manufacture of products moved from the craftsman era where individual craftsman designed and manufactured the entire product, to the mass production era, where skilled laborers were crafting interchangeable parts or in some cases single features on interchangeable parts, variation in assemblies has been a major concern to designers, manufacturers, and in a more subtle way, customers. Variation, in the end, affects quality, performance and the cost of products.

All assemblies have key features or *key characteristics* (KCs) that must be satisfied for the assembly to meet its design intent. Due to variation in the parts that make up an assembly, individual part variation typically propagates to these key characteristics. A number of methodologies and tools have been developed that allow the designer to predict how part variation propagates to the key characteristics. Long viewed as a tolerancing problem, out-of-control variation in key characteristics is traditionally solved by controlling part tolerances. Even today it is common practice to control variation in the overall assembly by simply controlling the part tolerances. This often leads to tighter tolerances and higher part costs due to the limits of the chosen or required manufacturing processes.

Taguchi proposed another means of satisfying KCs[Taguchi et. al., 1999]. He indicates that instead of changing the manufacturing processes, or tightening part tolerances, assemblies can be designed such that they are insensitive to the variation in the individual parts.

Taguchi initially conceptualized the idea of a robust design. A robust design is one that meets key characteristics regardless of the variation in the parts. By design, the assemblies are able to absorb the part variation. Though the concept seems straightforward at the outset, the design of robust assemblies has only recently been studied to the extent where methodologies are now being developed. One such methodology upon which this work intends to build is the concept of a 'smart assembly.'

1.1.1 Smart Assemblies

A *smart assembly*, by definition, is an assembly that has a feature or features, not otherwise required, that allow the assembly to absorb the variation in the individual parts that make up that assembly. Thus, in theory, a designer could design an assembly that will be a perfect assembly built out of imperfect parts [Downey, et. al. 2001]. Work on smart assemblies has thus far been limited. Parkinson and Chase, 2002 present the overall general concept, and [Downey, 2001] and [Downey et. al., 2002] developed a general design methodology. The smart assembly design methodology establishes two types of features that can be designed into an assembly to make it 'smart.' The two types of smart features are "passive" and "active".

A *passive smart feature* will allow for the absorption of assembly variation once, at assembly time, but is then fixed and can no longer absorb variation without external adjustment. A Passive Smart Feature will add an additional rigid constraint to the assembly. Examples of passive smart features are slotted holes, adjustment screws, and shims.

An *active smart feature* will allow for the absorption of manufacturing, operational, and environmental variation throughout the life of the assembly. Examples of active smart features are springs and other parts that have the capacity to adjust and absorb variation continuously in order to accommodate wear or other changes over time. Active smart features essentially provide a Degree of Freedom (DoF) in the direction of variation absorption. This DoF will typically only be for small kinematic adjustments and often a force will be transmitted in the direction of the DoF as well. For example, springs are often implemented as active smart features. Springs will typically allow a DoF, but provide a force in that DoF as well. One type of design that can require forces to keep the assembly properly assembled is an exactly constrained design.

1.1.2 Exact Constraint and Nesting Forces

Blanding [Blanding, 1999] states that an exactly constrained design is one where "each and every one of the body's degrees of freedom has been individually accounted for and constrained, one constraint at a time." Exactly constrained assemblies have each DoF constrained by exactly one constraint.

In two-dimensional space, a rigid body has three degrees of freedom. For an exactly constrained, static assembly of two parts in 2D space, one part in the assembly must properly apply three constraints to the second part in the assembly. Figure 1.1 shows a block assembly as an example of an exactly constrained design. The block is restricted in the x, y and θ degrees of freedom, where C3 and C2 constrain the x and the y DoFs respectively and C1 constrains the rotation, θ.

Because of the one-to-one relationship of constraints to DoFs in exactly constrained designs, *exact constraint* (EC) design relies heavily on *nesting forces*. In fact, [Hale, 1999] states that preload, or the nesting force, "is a central concept to the design of kinematic couplings." Essentially, nesting forces help exactly constrained designs by keeping parts seated in their exactly constrained positions. Clearly in Figure 1.1 a nesting force will be required to keep the block seated against its constraints. In Figure 1.2, a nesting force is applied to the exactly constrained block assembly.

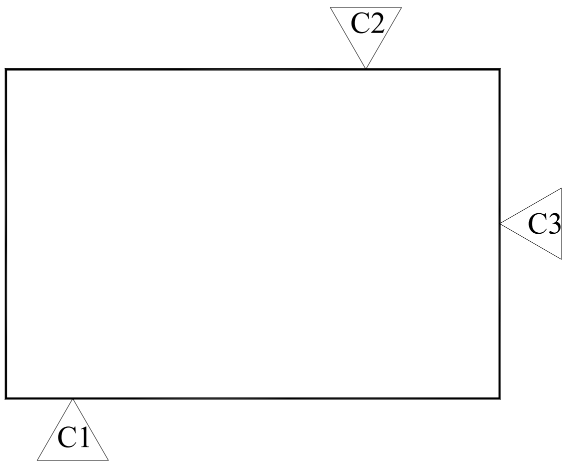


Figure 1.1: Exactly Constrained block assembly.

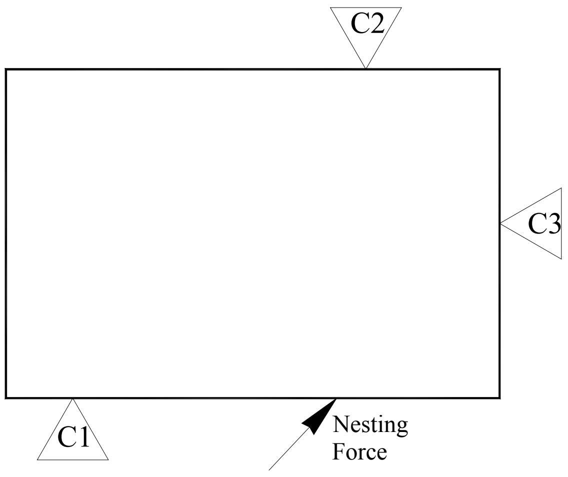


Figure 1.2: Exactly Constrained block assembly with an applied Nesting Force.

Nesting forces can themselves be seen as constraints. As seen in Figure 1.2, if a rigid device supplies the nesting force, the nesting force becomes an additional hard constraint and the design becomes an over-constrained design. If, however, the nesting force is supplied by a flexible part such as a cantilevered beam or a compression spring, the constraint is considered a *soft constraint* and can actively adjust to the variation in the parts and operating conditions.

For a nesting force to be effective, it must provide the required force over the range of variation an assembly may experience. This is true whether the variation is due to manufacturing variation at assembly time or because of operational and/or environmental conditions throughout the life of the assembly. Active Smart Features are particularly well suited to provide these nesting forces because active smart features are able to absorb variation while continuously providing the required nesting force. In fact, [Downey, 2001]states that smart features *should* be used to provide the nesting forces in an assembly. However, to this point, no method exists that specifies how to design the features that supply these nesting forces in exactly constrained assemblies.

1.2 Thesis Objectives

The purpose of this research is to develop a method for the analysis and design of the active smart features that will be used to supply nesting forces in exactly constrained assemblies. Because of the use of active smart features, these nesting forces will be robust to the variation inherent in the assemblies. Finally, this work will also illustrate the need for multiple active smart features used as nesting forces in a single assembly.

1.3 Delimitations

No mechanism synthesis will be performed in this work. It will be assumed that all the designs presented in this work have been previously synthesized to meet design requirements.

All parts will be assumed to be rigid, with the exception of the active smart feature(s).

In addition, this work will be restricted to 2D space.

Because part variations and the subsequent variations in the assembly are typically small, the deflections of the active smart features will be considered small and linear. Therefore only small deflection analysis will be used.

1.4 Significance of Research

Exact constraint is a valuable machine design tool, so much so that [Kriegel, 1994] argues that exact constraint design should be taught as part of the standard engineering curriculum. The advantages of exact constraint can be summed up in the following statements:

- Exactly Constrained designs can assemble over a wide variety of conditions.
- Exactly Constrained designs do not have any play.
- Exactly Constrained designs will not bind.
- Exactly Constrained designs do not have any internal stresses or strains caused by over-constraint.
- Exactly Constrained designs can tolerate wear of parts, minor damage and deformations caused by creep or overload.
- Exactly Constrained designs are easier to assemble and maintain than over-constrained designs.

Many exactly constrained designs rely on nesting forces. Because of this dependence on nesting forces, this work proposes to develop a method for the analysis and design of the active smart features that will supply the nesting forces in exactly constrained assemblies. It will provide yet another robust design tool to be placed in the "designer's toolbox." It will also enhance the current literature by further extending the understanding of how to design these features that will supply nesting forces as part of Exact Constraint Design.

Chapter 2 Background and Literature Review

2.1 Introduction

This chapter will explain the necessary background information and related literature that will be used throughout the remainder of this work. It begins with the topics of variation in assemblies and robust assemblies. The chapter will then provide more information about one means of achieving a robust assembly called *smart assembly design*. Then, exact constraint and nesting forces and their associated benefits as related to robust design will be discussed. The chapter will end with a presentation of a tolerance analysis tool called the Direct Linearization Method or DLM. The DLM will allow the variation in the assembly to be characterized in terms of the part variation.

2.2 Variation in Assemblies

Variation arises in assemblies as a result of the variation in the parts that make up the assemblies. Part variation usually occurs due to the manufacturing processes that were used to make the parts. For instance, when a lathe is used to form a part, the machine may be set up to cut the dimension perfectly, but due to deflection in the cutter, wear in the cutting tool or other causes, the formed parts are no longer true to the exact dimension. As a result, designers assign tolerances to dimensioned parts to indicate limits on how far features can deviate from the nominal dimension.

Designers have been applying several different methods to analyze how assigned part tolerances stack-up in a design. These various methods of analysis allow designers to adjust dimensions such that the parts, when assembled, form an assembly that meets all customer requirements. A survey of these methods is given in [Parkinson and Chase, 2002].

2.3 Robust Design

In light of the variation that exists in assemblies, a designer should be concerned with how robust his design will be with respect to this variation. Taguchi defines robustness as "the state where the technology, product, or process performance is minimally sensitive to factors causing variability (either in the manufacturing or user's environment) and aging at the lowest unit manufacturing cost" [Taguchi et. al, 1999].

According to Taguchi's definition of robustness, a robust design or assembly is one that will meet all assembly requirements regardless of the variation in the parts that make up the assembly. This type of a robust assembly results in a high level of customer satisfaction due to a higher quality product and lower consumer cost because fewer parts are scrapped. It also results in increased profit due to increase market share, less scrap, and greater efficiency in the manufacturing process.

Though the idea for robust designs initially came from Taguchi [Taguchi et all., 1989], [Peace, 1993], others have looked at applying nonlinear programming methods to include design constraints and the effects of correlation among the independent variables [Parkinson, 1995], [Yu and Ishii, 1994], [Otto and Antonsson, 1993], and [Chen et al., 1996]. These design tools are focused on either allowing the design to function properly regardless of variation, or reducing the sensitivity of the design to variation. However, in some cases, there is still sufficient variation in the design to cause problems. Two design tools that can be used to achieve robust assemblies are smart assembly features and exactly constrained designs. They are discussed in the following sections.

2.4 Smart Assemblies and Features

Smart assemblies have features, not required by the function of the design, which allow the design to absorb, or cancel out the effects of part variation so that the assembly always meets the design requirements [Parkinson and Chase, 2002], [Downey, 2001], and [Downey et. al, 2002].

The smart features that are used in the design of smart assemblies come in two variants, *passive* and *active* smart features. A passive smart feature is one that absorbs variation once, such that one or more design requirements are met, and statically maintains that requirement for the life of the assembly. Several examples of passive smart features include, but are not limited to, slotted holes, shims, adjustment screws, and welded slip joints. An active smart feature is one that actively absorbs variation for the life of the assembly. Some examples of active smart features are linear bearings, springs, and part flexure or compliance.

Smart features can be implemented in assemblies as described in an example from [Downey, 2001]. In this example a passive smart feature is used to cancel out the effects of variation in a simple assembly. The initial assembly is shown in Figure 2.1. The design requirement for this assembly is dimension '**d**'; however, due to variations in parts **A**, **B**, and **C**, the design requirement will not be met for all assemblies. Two implementations of passive smart features can be seen in Figure 2.2. In Figure 2.2*a*, shims are added to the assembly as a passive smart feature. At assembly time an indicator gage would be used to stack shims to meet dimension '**d**'. In Figure 2.2*b*, an adjustable screw would be used in conjunction with a gage to ensure dimension '**d**' is met.

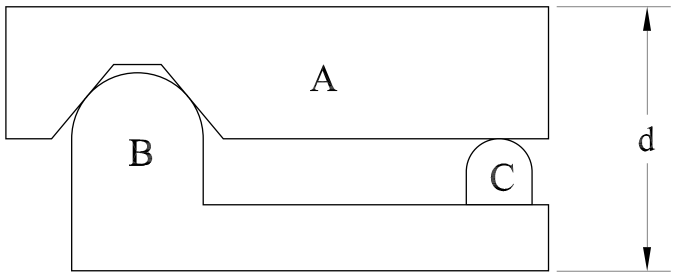


Figure 2.1: Initial design of example assembly.

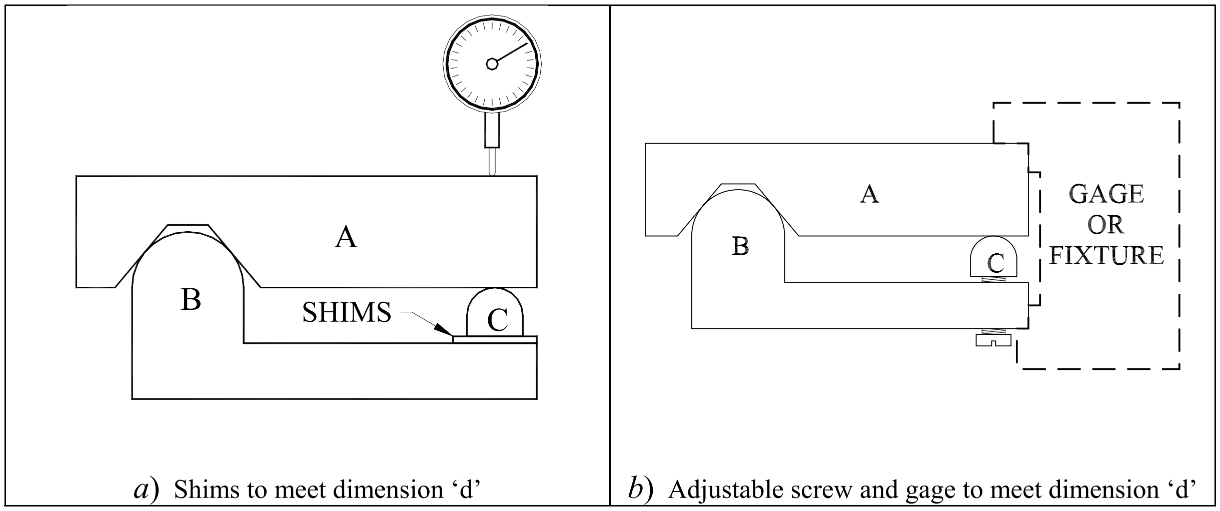

A
B
SHIMS
C
a) Shims to meet dimension 'd'
A
B
C
GAGE
OR
FIXTURE
b) Adjustable screw and gage to meet dimension 'd'

Figure 2.2: Two smart feature implementations to meet dimension 'd'.

Smart assemblies provide a powerful tool to the designer who is trying to mitigate the effects of variation in a design. One type of design in which smart features have a great advantage is an exactly constrained design. The following section will explain the idea of an exactly constrained design.

2.5 Exact Constraint and Nesting Forces

One type of design that is particularly robust to variation is an exactly constrained design. Exactly constrained designs are discussed by [Blanding, 1999], [Skakoon, 2000], [Kriegel, 1994], and [Kamm, 1993]. Additional discussion of exact constraint design can also be found in [Hale, 1999]. Hale indicates exact constraint design involves applying constraints to a body to eliminate DoFs in a one-to-one fashion. Hale goes on to say "It is the objective of exact-constraint design to achieve some desired freedom of motion or perhaps no motion by applying the minimum number of constraints required." Blanding states an exactly constrained design is one where "each and every one of the body's degrees of freedom has been individually accounted for and constrained, one constraint at a time." An example of the type of two dimensional, exactly constrained assemblies used in this work can be seen in the pinned block assembly shown in Figure 2.3. Notice the one-to-one nature of the constraints to DoFs in that the pin constrains the block in both the x and the y translations and the second constraint only constrains the rotation θ.

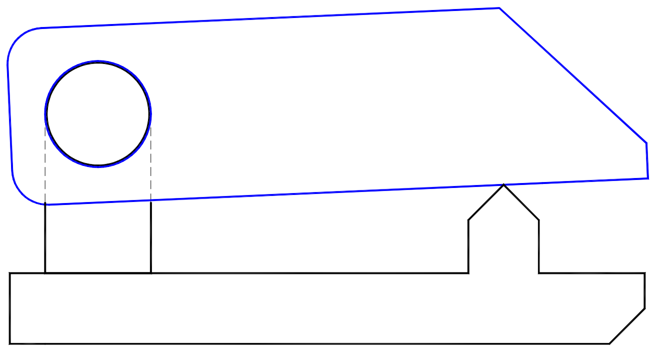


Figure 2.3 - Exactly constrained, pinned block assembly.

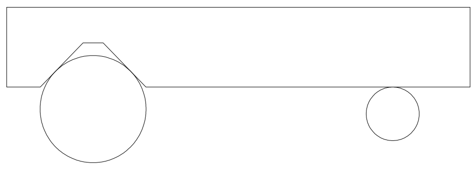
When analyzing an exactly constrained assembly, such as the Pinned Block in Figure 2.3, the advantages of exact constraint become apparent. In fact, [Kamm, 1993] writes that by employing EC design,

"you will achieve zero looseness and zero binding of moving parts; you will achieve assembly of fixed parts without strains or rework; and you will do so despite loose manufacturing tolerances and semiskilled assembly labor. You will minimize the manufacturing cost of your mechanism, you will make it more reliable, you will make it easier to disassemble and reassemble, and you will make it easier to maintain."

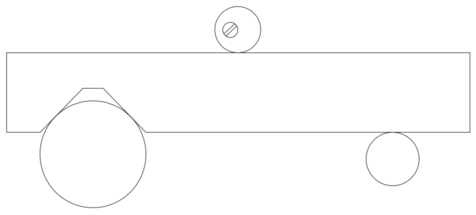
Hale indicates that in an exact constraint design "parts will fit together precisely and without backlash." Blanding states that exactly constrained designs achieve an extraordinary level of precision automatically using ordinary, low-cost, and inaccurate parts. Skakoon states that when over-constrained designs do not work as intended, the problems are hard to identify, whereas, in exactly constrained designs, the problems are easily identified and resolved.

Due to the one-to-one nature of DoFs to constraints in exactly constrained designs, there often arises a need for nesting forces in these designs. [Hale, 1999], [Blanding, 1999], and [Skakoon, 2000], all mention the need for nesting forces in exactly constrained designs. Consider the block being constrained by the two pins found in Figure 2.4*a*. It can be observed that the block could likely be unseated from the constraints if the assembly were subjected to external loads or accelerations. Therefore, in Figure 2.4*b*, *c*, and *d* three different types of nesting forces are shown being applied to the block to keep it seated against the pins. The first two, *b* and *c*, show passive smart features applying nesting forces, in which the force is applied by rigid parts and then fixed in that position. The third example in *d* illustrates an active smart feature, in the form of a compression spring, applying the nesting force. This active nesting force can provide a force throughout a range of variation.

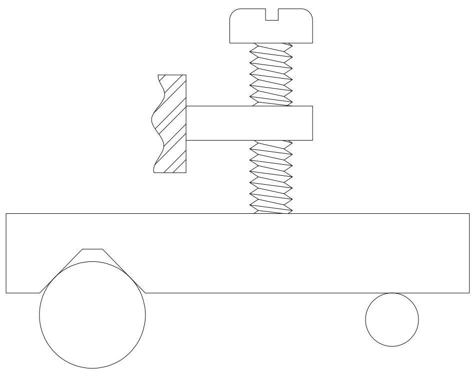
Several issues exist with nesting forces in exact constraint design. First, placement of the nesting force must be done such that the nesting force maintains the stability of the design. Second, nesting forces must be applied as 'soft constraints'. Applying nesting forces using rigid features imposes over-constraints on exactly constrained designs. Finally, part variation in rigid features used as nesting forces can result in the failure of the nesting force. Each of these issues will now be discussed in detail.



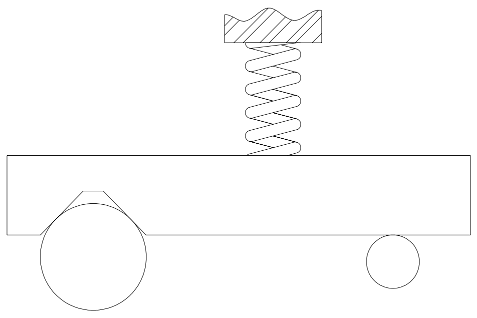
a) Exactly Constrained block on two pins.



b) Cam being used as a nesting force.



c) Set screw being used as a nesting force



d) Compression Spring being used as a nesting force.

Figure 2.4: Examples of and exactly constrained block and different types of nesting forces.

First, placement of nesting forces must be done with respect to the stability of the design. This is clearly illustrated by revisiting the exactly constrained block on two pins. A nesting force is applied to the block in Figure 2.5 at three different positions. Because of moments generated about the constraints, forces F_A and F_C would be unstable and likely unseat the block from one or more constraint. By observation, F_B would keep the block properly seated against the constraints while maintaining the stability of the design.

Second, as with the examples in Figure 2.4*b* and *c*, when hard constraints are used to provide the nesting forces, even if using passive smart features, the design becomes over-constrained. In fact, it is reasoned that by adding nesting forces, one sacrifices the benefits of exactly constrained designs [Hale, 1999]. However, if the designer only desires to absorb manufacturing variation, *and* the design *will not* experience any significant environmental or operational variation, then passive smart features can effectively be used to supply nesting forces.

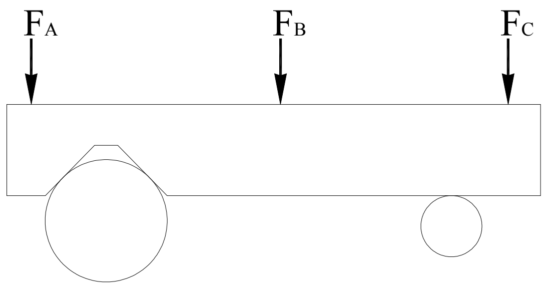


Figure 2.5: Stability of nesting force. F_A and F_C are unstable. F_B is stable.

Finally, variation in rigid features being used as nesting forces can result in the failure of the nesting force. An example of this is illustrated in the nesting forces being supplied by rigid features in Figure 2.4*b* and *c*. If variation arises in the life of the assembly due to wear or thermal effects it can clearly be seen that either these rigid features will either impose unwanted stresses or strains, or gaps will form leaving the assembly without the required nesting force.

Because of variation in assemblies, Downey states that smart features should always supply the nesting forces in exactly constrained designs [Downey, 2001]. However, Downey does not indicate the type of smart feature to be used as a nesting force. As stated previously, failure can result when passive smart features are used to provide nesting forces. This can be overcome by always using active smart features to supply the nesting forces in exactly constrained designs. Some active smart features have the unique quality of providing a force, while maintaining an active DoF in the direction of the force. This implies a need to know how the variation in the parts of an assembly will affect the design of the active smart features being used as nesting forces. This knowledge can be obtained by performing a tolerance analysis on the design. This work will rely on the Direct Linearization Method for tolerance analysis.

2.6 DLM -- Direct Linearization Method

The Direct Linearization Method (DLM) can be used to analyze tolerance stackups in both 2D and 3D assemblies. The DLM is preferred because it does not require explicit assembly functions [Chase, 1999], [Chase et. al, 1995] as do other tolerance analysis methods. It linearizes non-linear equation sets so they can be used in algebraic matrix manipulations to calculate the dependent variable sensitivities. These sensitivities can then be used to predict the variation in the dependent variables.

As this work will rely heavily on the DLM and its results, Section 2.6.1 will explain the process of applying the DLM.

2.6.1 DLM Example – Exactly Constrained Pinned Block

The Pinned Block Assembly can be seen in Figure 2.6. In this exactly constrained assembly, it is assumed that the pin will always exactly fit in the hole. The Base consists of a rigidly fixed pin that locates the Block in the *x* and *y* DoFs and a locator pin that eliminates the Block's θ DoF. The design objective for this assembly is that the points A and B in Figure 2.7 are spaced 0.67" in the *GAPy* dimension and always less than .03" in the *GAPx* dimension. Nominal values for the independent and dependent variables are shown with their respective labels in Figure 2.7. The nominal values, tolerances, standard deviations, and percent variation from nominal of the respective independent variables can be seen in Table 2.1.

The process for applying the DLM to this assembly model proceeds as follows.

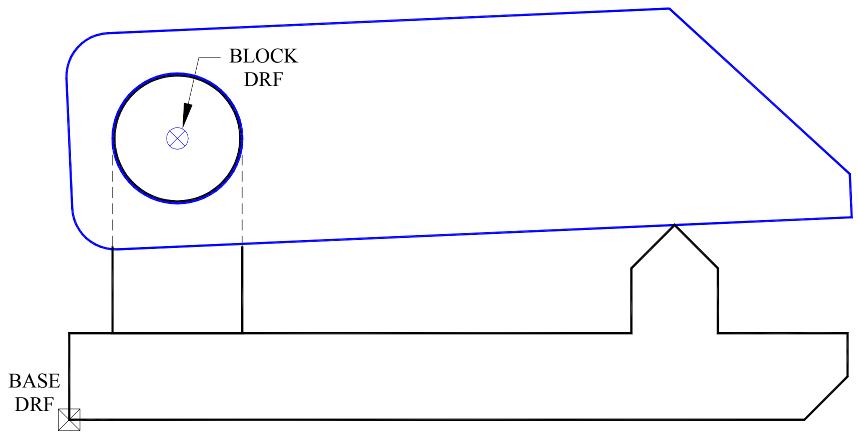


Figure 2.6: Exactly Constrained Pinned Block with DLM DRFs shown.

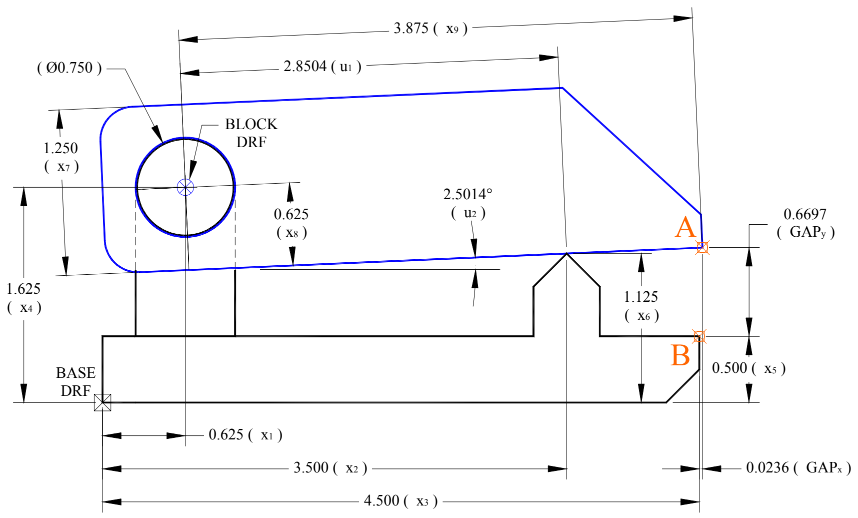


Figure 2.7 – Dimensioned Pinned Block Assembly.

Table 2.1 - Nominal values, tolerances, standard deviations, and tolerance percent of nominal values for Pinned Block Assembly.

Inputs	Nominal		WC	Std. Dev.	% of Nom.
x_1	0.625	δx_1	0.019	0.006	3.0%
x_2	3.500	δx_2	0.105	0.035	3.0%
x_3	4.500	δx_3	0.135	0.045	3.0%
x_4	1.625	δx_4	0.049	0.016	3.0%
x_5	0.500	δx_5	0.015	0.005	3.0%
x_6	1.125	δx_6	0.034	0.011	3.0%
x_7	1.250	δx_7	0.038	0.013	3.0%
x_8	0.625	δx_8	0.019	0.006	3.0%
x_9	3.875	δx_9	0.117	0.039	3.0%

STEP 1 – Assign a Datum Reference Frame to each part

Each part in the assembly must have its own local coordinate system. Individual local coordinate systems are called Datum Reference Frames (DRF) and are specific to each part in the assembly. For 2D analysis, the only two applicable DRFs can be seen in Table 2.2. The DRFs for the Pinned Block Assembly example can be seen in Figure 2.7.

Table 2.2: 2D Datum Reference Frames.

⊕	□
Center Datum	Rectangular Datum

STEP 2 – Specify Kinematic Joints between Parts

All parts in the assembly must be linked together by kinematic joints. Kinematic joints allow for the kinematic adjustments that are necessary when variation is present. Kinematic joints indicate how the parts can move relative to each other. For 2D assemblies there are only 6 different joints that are of concern. These six joints can be seen in Table 2.3. For the purposes of this thesis, additional information on how each joint transfers forces or moments has also been included.

It can be see in Table 2.3 that the set of DoFs and the set of transmitted force directions are mutually exclusive. The unconstrained directions allow kinematic motion. The constrained directions transmit forces. When active smart features are applied as nesting forces, the joint's force transmittal direction is the direction in which the nesting force will be applied.

For the Pinned Block Assembly, there are two joints. There is one Revolute joint between the pin on the Base and the hole in the Block, and there is one Edge Slider joint between the Block and the Base. The joints for the Pinned Block Assembly can be seen in Figure 2.8.

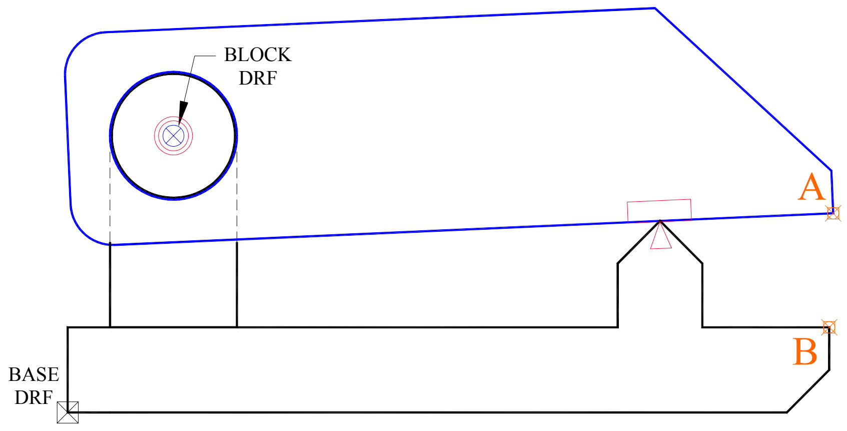
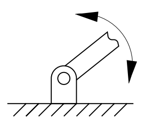
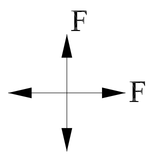
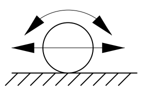
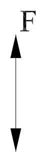
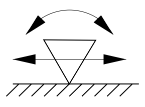
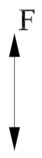
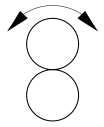
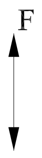
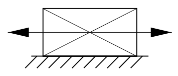
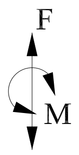
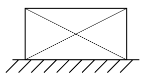
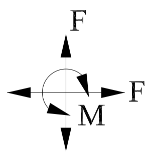


Figure 2.8: Joints applied to the Pinned Block Assembly.

Table 2.3: 2D Kinematic joints and associated degrees of freedom and capacity to transmit forces.

Joint DoF	***Forces Transmitted***	***Joint DoF***	***Forces Transmitted***
			
Revolute		**Cylinder Slider**	
			
Edge Slider		**Parallel Cylinders**	
			
Planar		**Rigid**	

STEP 3 – Form Vector Loops for the Assembly

With the joints attached, the final step in developing the tolerance analysis model requires the formation of a set of vector loops that completely describes the assembly. There are several rules that aid and govern the creation of these vector loops. The rules [Chase, 1999] for forming vectors loops are:

- Enter through a joint
- Follow the independent or dependent dimensions to a DRF
- Follow the independent or dependent dimensions leading to another joint
- Exit to the next adjacent part in the assembly
- No loop can pass through the same part or same joint twice.
- Loops must pass through each part and each joint in the assembly, but a single loop does not necessarily pass through every part or every joint.

- If a vector loop includes the same dimension twice, with vectors passing that dimension in opposite directions, then that dimension will be omitted due to redundancy.
- There must be enough closed loops to solve for all kinematic variables. The number of closed loops L required in an assembly is given by $L = J - P + 1$ where J is the number of joints and P is the number of parts.
- There can be as many open loops as there are design requirements on the assembly.

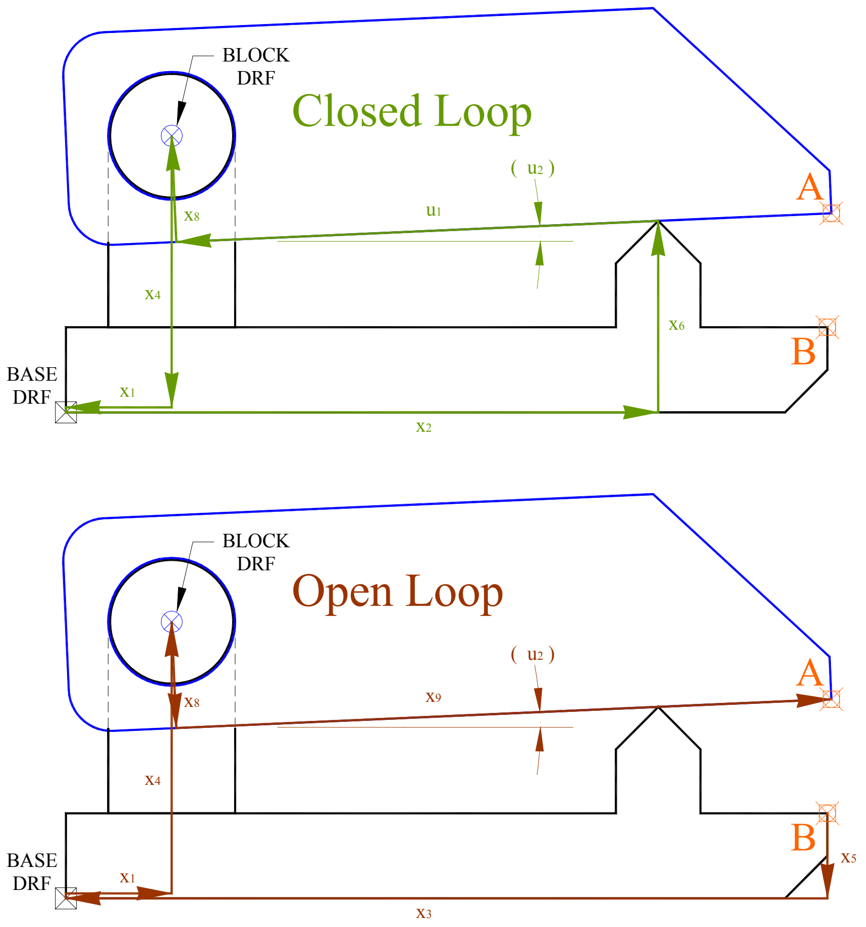


Figure 2.9 –Vector Loops for Pinned Block DLM model.

When the loops are established, the assembly variation model is defined. For the Pinned Block Assembly, there are two loops: one closed loop to identify the relative location of parts and one open loop to identify the relative location of the points A and B. These loops can be seen in Figure 2.9.

STEP 4 – Write Kinematic Equations from Vector Loops

Step 4 begins by forming the kinematic equations that will allow for the variation analysis of the Vector Assembly Model. Using the vector loop(s) created in Step 3, kinematic equations are written in the form of $h_{x,y,\theta}(x,u)=0$ for each closed vector loop and $p_{x,y,\theta}(x,u)=GAP$ for each open vector loop. Each vector is broken into *x*, *y* and θ components forming three equations for each loop. For the Pinned Block Assembly, the loop starts at the DRF of the Base and proceeds around the vector loop from vector x_2 to vector x_1. The h_x and h_y equations, given in equation 2.1 are simply the sum of the *x* and *y* components, respectively.

For the formulation of h_θ, the process involves starting at 0° and summing vector rotations around the loop. For instance, for the Pinned Block Assembly, start at 0° for x_2, then rotating 90° for x_6, rotate another $+90^\circ + u_2$ for u_1, -90° for x_8, then $+180^\circ - u_2$ for x_4, then -90° for x_6, then add -180° to return alignment to the positive x axis and to sum the equation to zero. The resulting set of h_i equations can be seen in equation 2.1. Evaluating these equations in terms of known angles results in equation set 2.2.

$$\begin{aligned}
h_x &= x_2 \cdot \cos(0^\circ) + x_6 \cdot \cos(90^\circ) + u_1 \cdot \cos(180^\circ + u_2) \\
&\quad + x_8 \cdot \cos(90^\circ + u_2) + x_4 \cdot \cos(\text{-}90^\circ) + x_1 \cdot \cos(180^\circ) = 0 \\
h_y &= x_2 \cdot \sin(0^\circ) + \mathrm{x}_6 \cdot \sin(90^\circ) + u_1 \cdot \sin(180^\circ + u_2) \\
&\quad + x_8 \cdot \sin(90^\circ + u_2) + x_4 \cdot \sin(\text{-}90^\circ) + x_1 \cdot \sin(180^\circ) = 0 \\
h_\theta &= 0^\circ + 90^\circ + 90^\circ + u_2 - 90^\circ + 180^\circ - u_2 - 90^\circ - 180^\circ = 0
\end{aligned} \tag{2.1}$$

$$\begin{aligned}
h_x &= x_2\text{-}u_1 \cdot \cos(u_2)\text{-}x_8 \cdot \sin(u_2)\text{-}x_1 = 0 \\
h_y &= x_6\text{-}u_1 \cdot \sin(u_2)+x_8 * \cos(u_2)\text{-}x_4 = 0 \\
h_\theta &= 0 = 0
\end{aligned} \tag{2.2}$$

Three items of note can be seen in equation set 2.2. First, there are two non-linear implicit equations in the two desired assembly dimensions. Second, for assemblies like the Pinned Block where there is only one dependent angular dimension, the h_θ equation will equal zero and drop out of the set. Also, obtaining explicit functions for the dependent variables can be very difficult. Even for this simple example the explicit solution for the dependent angle, u_2 is extremely complicated, and without either a numerical non-linear equation solver, or a symbolic math package such as Maple©, finding the explicit equations for these variables can be very difficult.

STEP 5 – Linearize the Vector Loop Equations for the Closed Loops

The variables in the assembly equations are only perturbed by small amounts, i.e. the tolerances, therefore it is reasonable to employ a first-order Taylor Expansion to linearize the equations. This linearization process converts a set of complex, non-linear equations to a set of linear equations, which can be manipulated with linear algebra. A summary of the process is to convert equation set $[H]=[F(X)]=[0]$ to $[dH]=[dF(X)]=[0]$. This is illustrated in detail for the Pinned Block in equation 2.3, where δx_i represents the tolerances on the respective variables. The set of $h(x_i)$ equations are differentiated against all of the independent and dependent variables for the Pinned Block.

$$\begin{aligned}
\delta h_x &= \frac{\partial h_x}{\partial x_1}\delta x_1 + \frac{\partial h_x}{\partial x_2}\delta x_2 + \frac{\partial h_x}{\partial x_4}\delta x_4 + \frac{\partial h_x}{\partial x_6}\delta x_6 + \frac{\partial h_x}{\partial x_8}\delta x_8 + \frac{\partial h_x}{\partial u_1}\delta u_1 + \frac{\partial h_x}{\partial u_2}\delta u_2 = 0 \\
\delta h_y &= \frac{\partial h_y}{\partial x_1}\delta x_1 + \frac{\partial h_y}{\partial x_2}\delta x_2 + \frac{\partial h_y}{\partial x_4}\delta x_4 + \frac{\partial h_y}{\partial x_6}\delta x_6 + \frac{\partial h_y}{\partial x_8}\delta x_8 + \frac{\partial h_y}{\partial u_1}\delta u_1 + \frac{\partial h_y}{\partial u_2}\delta u_2 = 0
\end{aligned} \tag{2.3}$$

It can be seen in equation 2.3 that the set of linearized assembly equations can be written in matrix form as follows:

$$\left[\frac{\partial h}{\partial x}\right]\{\delta x\}+\left[\frac{\partial h}{\partial u}\right]\{\delta u\}=0$$

Where:

$\left[\frac{\partial h}{\partial x}\right]$ is the matrix of partial derivatives with respect to independent variables,

$\left[\frac{\partial h}{\partial u}\right]$ is the matrix of partial derivatives with respect to dependent variables,

$\{\delta x\}$ is a vector of tolerances on the independent variables, and

$\{\delta u\}$ is a vector of unknown tolerances on dependent variables.

The values of the $\left[\frac{\partial h}{\partial x}\right]$ and $\left[\frac{\partial h}{\partial u}\right]$ matrices can be obtained by evaluating all partial derivatives at the nominal values of the independent and dependent dimensions. The nominal values for the dependent dimensions can be obtained from the CAD package that is being used to model the assembly. It is desired to solve for the unknown variations of the dependent variables. Algebraic manipulation of the matrix equation results in equation 2.4.

$$\{\delta u\}=-\left[\frac{\partial h}{\partial u}\right]^{-1}\left[\frac{\partial h}{\partial x}\right]\{\delta x\} \qquad (2.4)$$

Equation 2.4 is the essence of the DLM. The matrix $-\left[\frac{\partial h}{\partial u}\right]^{-1}\left[\frac{\partial h}{\partial x}\right]$ contains the sensitivities indicating how variation of the independent variables propagates to produce variation in the dependent variables. For the Pinned Block the $\left[\frac{\partial h}{\partial x}\right]$, $\left[\frac{\partial h}{\partial u}\right]$, $\left[\frac{\partial h}{\partial u}\right]^{-1}$ and $-\left[\frac{\partial h}{\partial u}\right]^{-1}\left[\frac{\partial h}{\partial x}\right]$ matrices are given in Table 2.4.

Table 2.4: DLM matrices for the One-way Clutch.

$\left[\frac{\partial h}{\partial x}\right]$ Matrix:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
h_x	-1.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-0.0436	0.0000
h_y	0.0000	0.0000	0.0000	-1.0000	0.0000	1.0000	0.0000	0.9990	0.0000

$\left[\frac{\partial h}{\partial u}\right]$ Matrix:

	u_1	u_2
h_x	-0.9990	-0.5000
h_y	-0.0436	-2.8750

$\left[\frac{\partial h}{\partial u}\right]^{-1}$ Matrix:

u_1	1.0086	-0.1754
u_2	-0.0153	0.3505

$-\left[\frac{\partial h}{\partial u}\right]^{-1}\left[\frac{\partial h}{\partial x}\right]$ Matrix:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
u_1	-1.0086	1.0086	0.0000	0.1754	0.0000	-0.1754	0.0000	-0.2193	0.0000
u_2	0.0153	-0.0153	0.0000	-0.3505	0.0000	0.3505	0.0000	0.3508	0.0000

Open Loops

The procedure is similar for the open loop pictured in Figure 2.9. The equations evaluated in terms of known angles for the open loop are found in equation set 2.5.

$$\begin{aligned} p_x &= x_2 - u_1 \cdot \cos(u_2) - x_8 \cdot \sin(u_2) - x_1 = GAP_x \\ p_y &= x_6 - u_1 \cdot \sin(u_2) + x_8 \cdot \cos(u_2) - x_4 = GAP_y \end{aligned} \qquad (2.5)$$

When linearized, this set of open loop equations results in an equation in the form of equation 2.6. To obtain the sensitivities for the variation in the *GAP* variables a substitution is required. By substituting δu from equation 2.4, the true sensitivities of the *GAP* variables to the independent variables can be found. This substitution results in equation 2.7. The sensitivity matrix for the *GAP* variables for the Pinned Block is found in Table 2.5.

$$dGAP = \left[\frac{\partial p}{\partial x}\right]\delta x + \left[\frac{\partial p}{\partial u}\right]\delta u \qquad (2.6)$$

$$dGAP = \left\{\left[\frac{\partial p}{\partial x}\right] - \left[\frac{\partial p}{\partial u}\right]\cdot\left[\frac{\partial h}{\partial u}\right]^{-1}\cdot\left[\frac{\partial h}{\partial x}\right]\right\}\delta x \qquad (2.7)$$

Table 2.5 - *GAP* sensitivity matrix for the Pinned Block Assembly.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
GAPx	1.0070	-0.0070	-1.0000	-0.1596	0.0000	0.1596	0.0000	0.2034	0.9990
GAPy	0.0597	-0.0597	0.0000	-0.3664	-1.0000	1.3664	0.0000	0.3687	0.0436

STEP 6 – Estimate Variation in Dependent Variables

The 6th and final step in the DLM is to estimate the variation of the dependent variables. Two common estimates of the variation are Worst Case (WC) and Root Sum Squares (RSS). Given that s_{ij} represents the sensitivity elements of the $-\left[\frac{\partial h}{\partial u}\right]^{-1}\left[\frac{\partial h}{\partial x}\right]$ and the $\left\{\left[\frac{\partial p}{\partial x}\right] + \left[\frac{\partial p}{\partial u}\right]\cdot\left[\frac{\partial h}{\partial u}\right]^{-1}\cdot\left[\frac{\partial h}{\partial x}\right]\right\}$ matrices, WC can be calculated by equation 2.8. Using a RSS estimation, the standard deviation can be obtained by equation 2.9 for both the *U* and *GAP* sets of dependent variables, respectively.

$$\textbf{Worst Case:} \quad \delta_{U_i} = \sum_{j=1}^{N} |\, s_{ij} \,| \delta x_j \qquad (2.8)$$

$$\textbf{RSS:} \quad \sigma_{U_i} = \sqrt{\sum_{j=1}^{N} \left(s_{ij} \sigma_{x_j} \right)^2} \qquad (2.9)$$

The WC and RSS calculations for the Pinned Block are illustrated below for both *GAP_x* and *GAP_y* as they were specified as the requirements for this design.

$$\textbf{WC:} \quad \delta_{GAP_x} = |\, s_{11} \,| \cdot \delta x_1 + |\, s_{12} \,| \cdot \delta x_2 + |\, s_{13} \,| \cdot \delta x_3 + |\, s_{14} \,| \cdot \delta x_4 + |\, s_{16} \,| \cdot \delta x_6 + |\, s_{18} \,| \cdot \delta x_8 + |\, s_{19} \,| \cdot \delta x_9$$

$$\delta_{GAP_y} = |\, s_{21} \,| \cdot \delta x_1 + |\, s_{22} \,| \cdot \delta x_2 + |\, s_{24} \,| \cdot \delta x_4 + |\, s_{25} \,| \cdot \delta x_5 + |\, s_{26} \,| \cdot \delta x_6 + |\, s_{28} \,| \cdot \delta x_8 + |\, s_{29} \,| \cdot \delta x_9$$

$$\textbf{RSS:} \quad \sigma_{GAP_x} = \sqrt{(s_{11} \cdot \sigma x_1)^2 + (s_{12} \cdot \sigma x_2)^2 + (s_{13} \cdot \sigma x_3)^2 + (s_{14} \cdot \sigma x_4)^2 + (s_{16} \cdot \sigma x_6)^2 + (s_{18} \cdot \sigma x_8)^2 + (s_{19} \cdot \sigma x_9)^2}$$

$$\sigma_{GAP_y} = \sqrt{(s_{21} \cdot \sigma x_1)^2 + (s_{22} \cdot \sigma x_2)^2 + (s_{24} \cdot \sigma x_4)^2 + (s_{25} \cdot \sigma x_5)^2 + (s_{26} \cdot \sigma x_6)^2 + (s_{28} \cdot \sigma x_8)^2 + (s_{29} \cdot \sigma x_9)^2}$$

$$\textbf{WC:}\quad \delta_{GAP_x} = |1.0070|\cdot 0.019 + |-0.0070|\cdot 0.105 + |-1.00|\cdot 0.135 + |-0.1596|\cdot 0.049 + |0.1596|\cdot 0.034 + |0.2034|\cdot 0.019 + |0.9990|\cdot 0.117$$

$$\delta_{GAP_x} = 0.2889$$

$$\delta GAP_y = |0.0597|\cdot 0.019 + |-0.0597|\cdot 0.105 + |-0.3664|\cdot 0.049 + |-1.00|\cdot 0.015 + |1.3664|\cdot 0.034 + |0.3687|\cdot 0.019 + |0.0436|\cdot 0.117$$

$$\delta GAP_y = 0.0989$$

$$\textbf{RSS:}\quad \sigma GAP_x = \sqrt{(1.0070\cdot 0.006)^2 + (-0.0070\cdot 0.035)^2 + (-1.00\cdot 0.045)^2 + (-0.1596\cdot 0.016)^2 + (0.1596\cdot 0.011)^2 + (0.2034\cdot 0.006)^2 + (0.9990\cdot 0.039)^2}$$

$$\sigma GAP_x = 0.0599$$

$$\sigma GAP_y = \sqrt{(0.0597\cdot 0.006)^2 + (-0.0597\cdot 0.035)^2 + (-0.3664\cdot 0.016)^2 + (-1.00\cdot 0.005)^2 + (1.3664\cdot 0.011)^2 + (0.3687\cdot 0.006)^2 + (0.0436\cdot 0.039)^2}$$

$$\sigma GAP_y = 0.0173$$

Performing two separate Monte Carlo simulations will validate the worst-case variations and the standard deviations predicted by equations 2.8 and 2.9. Thus, the standard deviations that were previously calculated were compared to a 100,000 run Monte Carlo simulation, where the independent variables were allowed to vary according to normal distributions with their respective standard deviations. This comparison is given in Table 2.6. The number 100,000 runs was chosen based on running the same simulation for 30k runs, 50k runs, 100k runs, 500k runs and finally, 1,000k runs. The difference between the standard deviations produced by the 30k and 50k was significant, but from 50k to 1,000k runs, there was virtually no change in the standard deviation; therefore, for this work *all standard deviations will be compared to Monte Carlo simulations of 100k runs*.

The worst-case values calculated previously were compared against a 500,000 run Monte Carlo simulation assuming a uniform distribution for the independent variables. The Monte Carlo simulation using a uniform distribution allows the independent variables to be at their maximums and minimums more often; this coupled with the increased number of simulation runs allowed the dependent variables to be pushed to their respective limits. More details on using a Monte Carlo simulation for worst-case comparisons can be seen in Appendix D. The worst-case comparison is given in Table 2.7. Because a worst-case analysis is focused on the limits of the problem, the comparison in Table 2.7 is given as "Good" if the limits returned by the Monte Carlo simulation fall within the limits predicted by the DLM and as "Low/High" if the Monte Carlo limits fall outside the respective "High/Low" limits returned by the DLM. For both the statistical and worst case calculations, the predictions from the DLM analysis are shown as generally conservative and good estimators of how the design will perform under typical manufacturing conditions.

As can be seen in these comparisons, the DLM is an effective means of predicting how variation in the features and parts will propagate to cause variation in the assemblies. The DLM can also be easily automated and used in CAD packages to analyze assemblies directly within the CAD model [Chase, 1999].

Table 2.6 - Percent Difference comparison between standard deviation returned by the DLM and a 100,000 run Monte Carlo simulation.

	DLM σ	**MC** σ	**Percent Error**
σ_{GAPx}	0.059923	0.060106	0.304%
σ_{GAPy}	0.017250	0.017257	0.039%

Table 2.7 - Difference value comparison between the worst-case values returned by DLM and Monte Carlo simulation.

	DLM High	**MC High**	**Compare High**	**DLM Low**	**MC Low**	**Compare Low**
GAP_x	0.4874	0.4711	Good	-0.4403	-0.4325	Good
GAP_y	0.7639	0.7670	Low	0.5756	0.5949	Good

2.7 Conclusion

This chapter has introduced several topics as background to the remainder of this work. Variation in parts will propagate to assemblies and can lead to the failure of a design in satisfying the design's key requirements. Robust design methodologies are an attempt to manipulate the design such that it will be less sensitive to the variation in the parts.

Exact constraint design is one approach for designing mechanical assemblies as it provides robustness of design and many other advantages not found in over or under constrained designs. Exactly constrained designs will often rely on nesting forces to keep the assembly properly seated against its constraints. The best way to supply nesting forces is with flexible features that actively adjust to the variations that result from manufacturing processes and environmental and operating conditions throughout the life of the assembly.

The DLM is an effective tool for analyzing tolerance models. It can provide both worst case and statistical tolerance predictions of how the assembly will propagate part and feature variation to the assembly's key requirements. The DLM will be the means of analysis for the design of the active smart features that will be used to supply the nesting forces in exactly constrained designs.

Chapter 3 General Method for Analysis and Design of Features that Supply Nesting Forces

3.1 Introduction

This chapter will explain the method developed in this research for the analysis and design of active smart features that will supply nesting forces for exactly constrained designs. For this method it is assumed that all parts are rigid except the active smart feature. The following sections will explain the steps for the assembly model setup and how subsequent analysis of the assembly model is to be performed. An example of the process will be provided which will be a continuation of the Pinned Block example in Section 2.6.1. A summary of the method will then be provided.

3.2 Assembly Model Setup

The method that has been developed in this work for the assembly model setup closely parallels the first steps in the DLM but with new principles associated with nesting forces. This method consists of 5 steps: 1) perform the design synthesis, 2) determine the placement and configuration of nesting forces, 3) apply part DRFs and kinematic joints, 4) form kinematic vector loops, and finally, 5) write the kinematic equations from the respective vector loops. The sections that follow will explain, in detail, each step of the assembly model setup.

3.2.1 STEP 1 – Perform Assembly Design Synthesis

The first step in setting up the model is generating the design for the assembly. However, because mechanism synthesis will not be covered in this work, the assembly designs will be assumed as given.

3.2.2 STEP 2 – Determine Placement and Configuration of Nesting Force

This step requires the designer to determine the placement of nesting force(s) in the assembly. This can be done by observation using a logical placement for the nesting force(s) or, in more complicated assemblies, the position of the nesting force can be determined by performing an appropriate force analysis. There are several ways to perform this force analysis; however, these methods are beyond the scope of this work. The most important issue for this placement of the nesting force(s) is it needs to result in a stable, robust design that will meet the design requirements. In this work the placement of all nesting forces will be done by inspection.

In some cases, the nesting force required will be in a resultant direction similar to the nesting force shown in Figure 1.2. It is preferable to use one feature to supply the nesting force to ease analysis. However, in some cases, the geometry may not easily allow the desired nesting force configuration. In these cases, multiple nesting forces can be combined to supply a resultant force that acts in the desired direction and magnitude. This is illustrated in Figure 3.1 where two nesting forces, F_x and F_y, are being used to supply a single, resultant nesting force, F_R.

In this work, cantilevered beams will be used as the springs that supply the nesting forces. This choice of this type of active smart feature is reasonable due to the fact that cantilevered beams can easily be molded into injection-molded parts and not increase part count or complexity in assemblies. An example of this can be seen in Figure 3.2. Here a small, poly-acrylic view window is being held in a larger plastic housing via a cantilevered spring. The spring keeps the part firmly seated in the assembly while also acting as a retainer. The Slot in the bottom keeps the part from moving side-to-side.

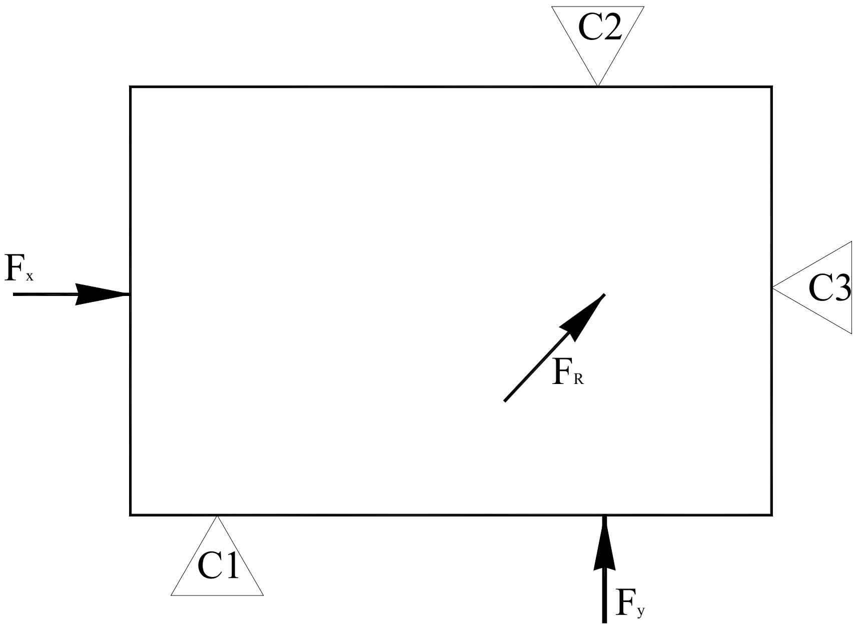


Figure 3.1 - Exactly constrained design using two forces to provide a single, resultant nesting force.

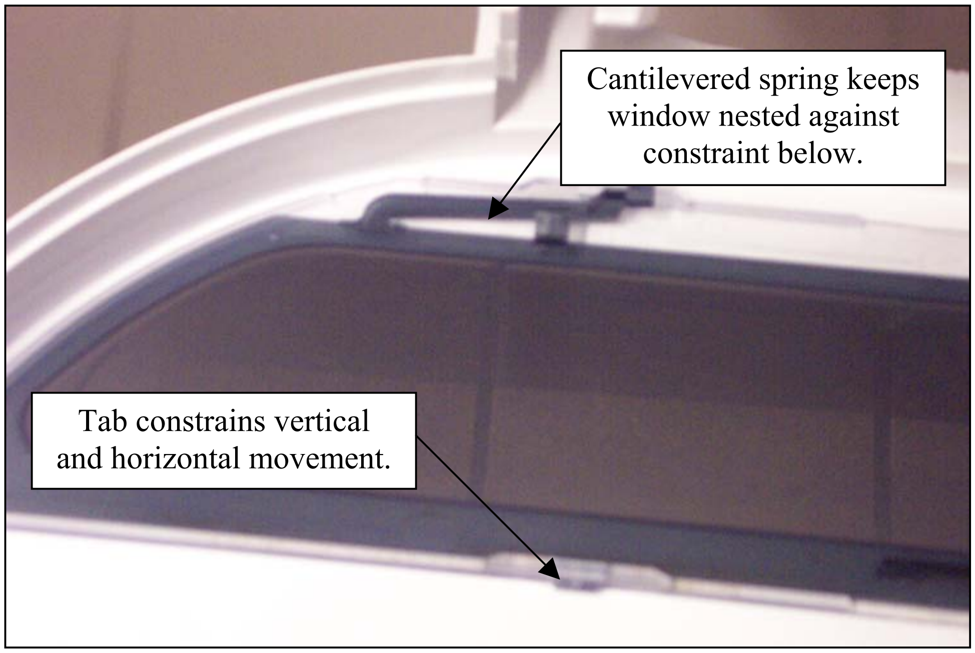

Cantilevered spring keeps window nested against constraint below.
Tab constrains vertical and horizontal movement.

Figure 3.2 - Actual assembly showing the use of a cantilevered spring molded into a part. The spring supplies a vertical nesting force.

3.2.3 STEP 3 – Apply Part DRFs and Kinematic Joints

Now that the type of feature(s) that supply the nesting force(s) have been chosen and located, the DLM will be set up. The first step in this process is applying the appropriate DRFs to each part. Then, kinematic joints are applied to the appropriate contacts in the assembly.

3.2.4 STEP 4 – Form Kinematic Assembly Vector Loops

With part DRFs located and the kinematic joints identified, vector loops are now applied to the model. All features, including the nesting forces, are to be included in their appropriate loops. The closed and open loops are formed as explained in Section 2.6.1. As the closed and open loops are identified, it will be seen that the features that supply the nesting forces and their associated independent and dependent variables are not found in the traditional closed or open loops. Therefore, nesting force loops must be created.

These nesting force loops do not contain any forces. They simply characterize the dimensional variation in the deflections that will produce the nesting forces. Most of the principles that govern the creation of closed and open loops apply to nesting force loops. However, there are two principles that require some revision.

3.2.4.1 Principles for Nesting Force Loops

First, each nesting force will have one complete, closed loop associated with it.

Second, the nesting force loop must include all independent locator dimensions for the feature that applies the respective nesting force. In 2D, two independent dimensions will position the active smart feature that applies the nesting force.

These two principles, used in conjunction with the principles established in the DLM, will allow for the creation of nesting force loops.

3.2.5 STEP 5 – Write Kinematic Equations

The kinematic equations must now be written from the vector loops. Extract the *x*, *y*, and θ equations in the form as shown in Equation 3.10, where *h*(*x,u*) are the standard DLM closed loop equations and *g*(*x,u,v*) are the equations for the new nesting force loops. Notice the addition of the *v* variables in *g*. These are the assembly or dependent variables associated with the nesting forces and their relative location with respect to other parts in the assembly. One of the dependent v_i variables will represent the deflection of the active smart feature. The other v_i will typically represent the location of the nesting force with respect to the DRF of the part to which the nesting force is applied.

$$\begin{aligned} h_{X,Y,\theta}(x,u) &= 0 \\ g_{X,Y,\theta}(x,u,v) &= 0 \end{aligned} \qquad (3.10)$$

With the creation of the equations for the nesting force loops, the assembly model setup is complete and ready for analysis.

3.3 Assembly Model Analysis

It is important to note here that forming nesting force loops is a slight deviation from the DLM as contained in [Chase, 1999]. In [Chase, 1999], the *open loops* identify the 'gap' dimensions. All locating dimensions for the gap(s) are given by independent variables and the open loop equations can be written explicitly for the gaps in terms of known independent variables.

The nature of nesting forces requires a different treatment. Typically, two dependent variables will help to locate each of the nesting force features. These dependent variables do not appear in the closed loops or the open loops of the DLM. Therefore, these loops are an addition to the DLM and must be analyzed differently.

This analysis of the nesting forces loops requires two additional steps in the analysis of the assembly model. After the traditional DLM closed and open loop analysis, these two additional steps consist of the analysis of the nesting force loops, and finally analysis of the force equation using the results of the previous steps. These steps will be explained in the following sections.

3.3.1 STEP 6 – Perform Analysis of the Closed and Open Loop Equations

The analysis of the closed and open loops is performed as explained in Section 2.6.1. Information from this analysis is also required in the nesting force loop analysis.

3.3.2 STEP 7 – Perform Analysis of the Nesting Force Loop Equations

Nesting force loop analysis builds on the closed loop sensitivities as shown in equation 2.4 which is repeated here as Equation 3.11.

$$[\delta u] = -\left[\frac{\partial h}{\partial u}\right]^{-1} \cdot \left[\frac{\partial h}{\partial x}\right] \cdot [\delta x] \qquad (3.11)$$

The nesting force loop equations contain the new set of dependent variables, *v*. The variation in these variables can be obtained by linearizing the nesting force loop equations just as was done for the closed loop equations using a first order Taylor series as seen in Equation 3.12.

$$[\partial g] = \left[\frac{\partial g}{\partial x}\right] \cdot [\delta x] + \left[\frac{\partial g}{\partial u}\right] \cdot [\delta u] + \left[\frac{\partial g}{\partial v}\right] \cdot [\delta v] = 0 \qquad (3.12)$$

This equation can be rearranged in preparation to solve for [δv] as in Equation 3.13.

$$-\left[\frac{\partial g}{\partial v}\right] \cdot [\delta v] = \left[\frac{\partial g}{\partial x}\right] \cdot [\delta x] + \left[\frac{\partial g}{\partial u}\right] \cdot [\delta u] \qquad (3.13)$$

Inverting and left multiplying by $-\left[\frac{\partial g}{\partial v}\right]^{-1}$ will provide the variation in the nesting force loop dependent variables. Also, by substituting [δu] from Equation 3.11 into Equation 3.13 [δv] can now be expressed explicitly in terms of the variation in the independent variables. This final expression for the variations in [δv] can be seen in Equation 3.14.

$$[\delta v] = \left[\frac{\partial g}{\partial v}\right]^{-1} \cdot \left\{ \left[\frac{\partial g}{\partial u}\right] \cdot \left[\frac{\partial h}{\partial u}\right]^{-1} \cdot \left[\frac{\partial h}{\partial x}\right] - \left[\frac{\partial g}{\partial x}\right] \right\} \cdot [\delta x] \qquad (3.14)$$

The sensitivities for how the variations in v_i change with respect to variations in x_i are contained in the sensitivity matrix, S. The sensitivity matrices for the closed, open and nesting force loops are given in Equation 3.15.

$$\begin{aligned}
&\text{Closed Loops}\\
&[S_{cl}] = -\left[\frac{\partial h}{\partial u}\right]^{-1} \cdot \left[\frac{\partial h}{\partial x}\right]\\
&\text{Open Loops}\\
&[S_{ol}] = \left[\frac{\partial p}{\partial x}\right]\delta x + \left[\frac{\partial p}{\partial u}\right] \cdot \left[\frac{\partial h}{\partial u}\right]^{-1} \cdot \left[\frac{\partial h}{\partial x}\right]\\
&\text{Nesting Force Loops}\\
&[S_{nfl}] = \left[\frac{\partial g}{\partial v}\right]^{-1} \cdot \left\{\left[\frac{\partial g}{\partial u}\right] \cdot \left[\frac{\partial h}{\partial u}\right]^{-1} \cdot \left[\frac{\partial h}{\partial x}\right] - \left[\frac{\partial g}{\partial x}\right]\right\}
\end{aligned} \tag{3.15}$$

Worst case and statistical variations in [δv] can be calculated using the same equations as used for the DLM, namely equations 2.8 and 2.9.

3.3.3 STEP 8 – Perform Force Analysis

The deflections and their associated variation have now been identified. The nesting forces can be analyzed using the data from the previous section. The general form of the force equation when dealing with springs is found in equation 3.16 where K is the spring constant in *force/unit length* and δ is the deflection in *length*.

$$F_i = K_i \cdot \delta_i \quad \text{where } i = max,\ min \tag{3.16}$$

This work will be using cantilevered beams to provide the nesting forces; therefore, the force analysis requires the equation of K for a cantilevered beam. K for a cantilevered beam can be obtained by recalling that the deflection for a cantilevered beam is given by $\delta = \frac{1}{3} \cdot \frac{FL^3}{EI}$, where E is the Modulus of Elasticity and L is the length of the beam from ground to the point of application of the deflecting force F. If the cantilevered beam is a simple rectangular cross section, then I is given as $I = \frac{wt^3}{12}$, where w is the width of the beam into the page and t is the thickness of the beam. Substituting these equations into equation 3.16 and then rearranging and simplifying results in equation 3.17. However, because the examples in this work all assume that the dimensional variables in K also have manufacturing variation, the worst-case equation for K is given in equation 3.18 where all subscripted variables have variation in them.

$$K = \frac{w \cdot t^3 \cdot E}{4 \cdot L^3} \qquad (3.17)$$

$$K_i = \frac{w_i \cdot t_i^{\,3} \cdot E}{4 \cdot L_j^{\,3}} \quad \text{where } i = \textit{max, min} \text{ and } j = \textit{min, max} \qquad (3.18)$$

The process for analyzing the nesting force now follows two different paths. For worst-case analysis, the results of the variation in the deflection can be applied directly to equations 3.16 and 3.18. For a statistical analysis, a new equation is required. For statistical calculations a first order uncertainty analysis as described in [Figliola and Beasley, 1995] and [Drake, 1999] will provide the desired data. The standard deviation of the force can be calculated, where all the nominal values of the independent variables and their associated standard deviations are known via equation 3.19.

$$\sigma_y = \left(\left(\frac{\partial f}{\partial x_1} \sigma_1 \right)^2 + \left(\frac{\partial f}{\partial x_2} \sigma_2 \right)^2 + \Lambda + \left(\frac{\partial f}{\partial x_n} \sigma_n \right)^2 \right)^{1/2} = \left(\sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} \sigma_i \right)^2 \right)^{1/2} \qquad (3.19)$$

This analysis will provide the information needed to refine the design of the assembly as well as the features that provide the nesting force by adjusting the nominal values of the dimensions as well as the associated tolerances to achieve a robust design. In the following section, this method is applied to the Pinned Block example from Section 2.6.1

3.4 Example: Pinned Block Assembly with Applied Nesting Force

To illustrate the method presented previously in sections 3.2 and 3.3, the Pinned Block Assembly example will be continued from Section 2.6.1.

STEP 1 – Perform Assembly Design Synthesis

The assembly is shown again in Figure 3.3.

STEP 2 – Determine Placement and Configuration of Nesting Force

Due to a design requirement that the Block always remain exactly constrained relative to the Base, a nesting force will be applied to the Pinned Block Assembly to keep it appropriately seated against the constraints. Therefore, a single active smart feature will be used to apply a vertical nesting force in the position and orientation shown in Figure 3.3.

The nesting force for the Pinned Block can be seen along with the new independent variables that locate the feature that applies the nesting force and the dependent variables, v_i the nesting force loop introduces in Figure 3.4. Table 3.1 shows the values for the two new independent variables for the nesting force and their associated tolerances, standard deviations, and percentage variation for the tolerance from the nominal value for the dimension. The active smart feature that supplies the nesting force is connected to the Base. This feature is located by the dimensions x_{10} horizontally and x_{11} vertically. To clarify the relationship of the dependent variables to the parts in the assembly, as well as to clarify how and where the nesting force is applied, Figure 3.5 shows the Pinned Block perturbed by two instances of manufacturing variation. The new independent variables associated with the nesting force remain at the values listed in Table 3.1. However, the dependent nesting force variables, v_1 and v_2, change to accommodate the two changes shown in x_6, where x_6 is set to 0.9375" and 1.250" respectively.

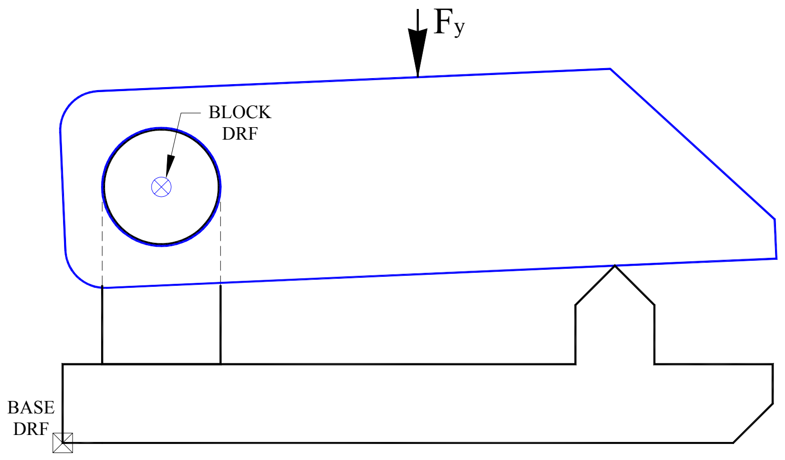


Figure 3.3 - The Pinned Block Assembly with a stable nesting force applied.

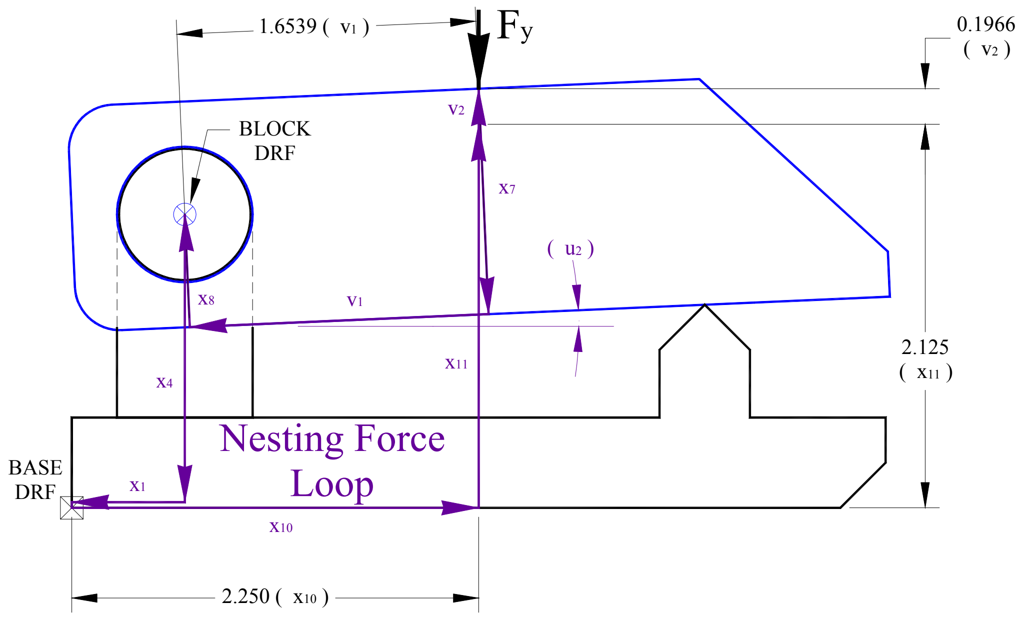


Figure 3.4 - Nesting force loop and associated variables for the Pinned Block Assembly.

Table 3.1 - Nesting force independent variables and their nominal values, tolerances, standard deviations, and tolerance percent of nominal.

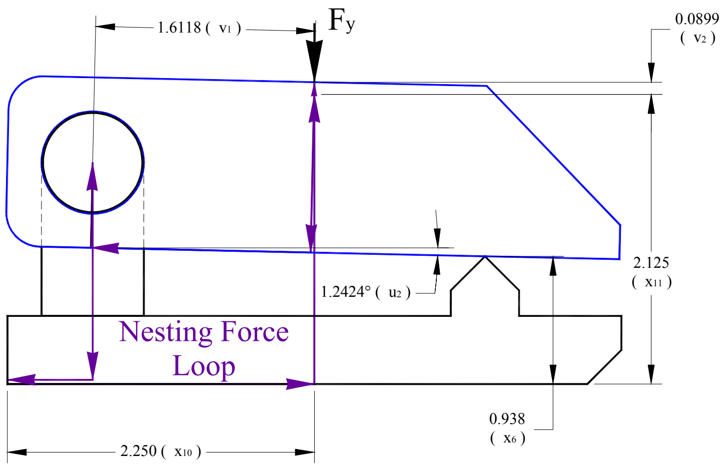
Inputs	Nominal		WC	Std. Dev.	Percent of Nominal
x_{10}	2.250	δx_{10}	0.068	0.023	3.0%
x_{11}	2.125	δx_{11}	0.064	0.021	3.0%

STEP 3 – Apply Part DRFs and Kinematic Joints

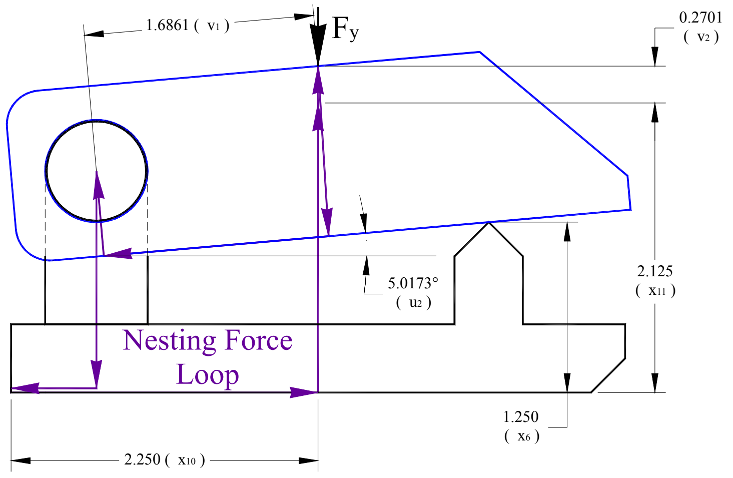
This step was discussed in Section 2.6.1 and is shown in Figure 2.8.

STEP 4 – Form Kinematic Assembly Vector Loops

The nesting force loop for the Pinned Block can be seen in Figure 3.4. Notice that the loop includes the two locators for the nesting force feature. The loop also includes the two dependent location dimensions, v_1 and v_2, with v_2 representing the deflection of the active smart feature and v_1 representing the location of the nesting force with respect to the Block's DRF.



a) x_6 perturbed by -0.1875".



b) x_6 perturbed by +0.125".

Figure 3.5 – Relationship of dependent nesting force loop dependent variables to assembly.

STEP 5 – Write Kinematic Equations

Kinematic equations are now derived for the nesting force loop. These equations for the Pinned Block in terms of known angles are given in equation 3.20. Notice that there are two equations and two unknowns.

$$\begin{aligned} g_x &= x_{10} + x_7 \cdot \sin(u_2) - v_1 \cdot \cos(u_2) - xx8 \cdot \sin(u_2) - x_1 = 0 \\ g_y &= x_{11} + v_2 - x_7 \cdot \cos(u_2) - v_1 \cdot \sin(u_2) + x_8 \cdot \cos(u_2) - x_4 = 0 \end{aligned} \tag{3.20}$$

STEP 6 – Perform Analysis of Closed and Open Loops

The analysis of the closed and open loops can be seen in Section 2.6.1.

STEP 7 – Perform Analysis of Nesting Force Loop

The matrices $\left[\frac{\partial g}{\partial x}\right]$, $\left[\frac{\partial g}{\partial u}\right]$, and $\left[\frac{\partial g}{\partial v}\right]$ are now formed by differentiating the *g*(*x*,*u*,*v*) equations with respect to the *x*, *u*, and *v* variables. These matrices as well as the $\left[\frac{\partial g}{\partial v}\right]^{-1}$ and the $\left[\frac{\partial g}{\partial v}\right]^{-1} \cdot \left\{\left[\frac{\partial g}{\partial u}\right] \cdot \left[\frac{\partial h}{\partial u}\right]^{-1} \cdot \left[\frac{\partial h}{\partial x}\right] - \left[\frac{\partial g}{\partial x}\right]\right\}$ matrices are evaluated in terms of known values and are given in Table 3.2. A complete symbolic and numerical analysis off the Pinned Block assembly can be found in Appendix A.

Using equations 2.8 and 2.9, the worst case and the statistical variations in the *v* set of dependent variables are calculated. These calculations are performed using the appropriate values from Table 2.1 and Table 3.1. These calculations are performed in a manner similar to the *GAP* equations as shown at the end of Section 2.6.1. The results of these calculations as well as the results from the Section 2.6.1 calculations are given in Table 3.3.

Table 3.2 - Matrices for nesting force loop tolerances analysis for the Pinned Block Assembly.

$\left[\frac{\partial g}{\partial x}\right]$ Matrix:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
g_x	-1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0436	-0.0436	0.0000	1.0000	0.0000
g_y	0.0000	0.0000	0.0000	-1.0000	0.0000	0.0000	-0.9990	0.9990	0.0000	0.0000	1.0000

$\left[\frac{\partial g}{\partial u}\right]$ Matrix:

	u_1	u_2
g_x	0.0000	0.6966
g_y	0.0000	-1.6250

$\left[\frac{\partial g}{\partial v}\right]$ Matrix:

	v_1	v_2
	-0.9990	0.0000
	-0.0436	1.0000

$\left[\frac{\partial g}{\partial v}\right]^{-1}$ Matrix:

v_1	-1.0010	0.0000
v_2	-0.0437	1.0000

$\left[\frac{\partial g}{\partial v}\right]^{-1} \cdot \left\{\left[\frac{\partial g}{\partial u}\right] \cdot \left[\frac{\partial h}{\partial u}\right]^{-1} \cdot \left[\frac{\partial h}{\partial x}\right] - \left[\frac{\partial g}{\partial x}\right]\right\}$ Matrix:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
v_1	-0.9903	-0.0107	0.0000	-0.2444	0.0000	0.2444	0.0437	0.2009	0.0000	1.0010	0.0000
v_2	-0.0183	-0.0253	0.0000	0.4198	0.0000	0.5802	1.0010	-0.4202	0.0000	0.0437	-1.0000

Table 3.3 - Results from both traditional DLM analysis and the new nesting force loop analysis.

	Worst Case	**Standard Deviation**
Closed Loop Analysis		
$\boldsymbol{\delta u_1}$	0.1438	0.0360
$\boldsymbol{\delta u_2}$	0.0377	0.0071
Open Loop Analysis		
$\boldsymbol{\delta GAP_x}$	0.2889	0.0599
$\boldsymbol{\delta GAP_y}$	0.0989	0.0173
Nesting Force Loop Analysis		
$\boldsymbol{\delta v_1}$	0.1138	0.0243
$\boldsymbol{\delta v_2}$	0.1563	0.0265

The design of a robust nesting force requires that there always be a deflection, v_2 in the active smart feature. In the worst-case analysis it can be seen that the predicted variation in v_2 is less than the nominal value, i.e. $v_2 - \delta v_2 > 0$. This will result in all of the assemblies having a positive nesting force.

To validate the results from the nesting force loop analysis, a 100,000 run Monte Carlo simulation was performed under the assumption that all independent variables follow a normal distribution with the respective standard deviation given in Table 2.1 and Table 3.1. The data were then analyzed statistically and these results were compared to the predicated results from the nesting force loop analysis. The results for this comparison can be seen in Table 3.4. These results show less than 1% error between the Monte Carlo and the DLM with the nesting force loop analysis.

Table 3.4 - DLM and nesting force loop analysis standard deviations compared with a 100,000 run Monte Carlo simulation.

	DLM	**Monte Carlo**	**Percent Difference**
Closed Loop:			
δu_1	0.036002	0.036062	0.1659%
δu_2	0.007144	0.007152	0.1135%
Open Loop:			
δGAP_x	0.059923	0.060106	0.3040%
δGAP_y	0.017250	0.017257	0.0393%
Nesting Force Loop:			
δv_1	0.024285	0.024334	0.2040%
δv_2	0.026539	0.026540	0.0022%

STEP 8 – Force Analysis

The nominal dimensions of the cantilevered beam for the Pinned Block Assembly along with their associated tolerances and standard deviations are shown in Table 3.5.

Table 3.5 - Cantilevered Beam dimensions. Values for nominal sizes, tolerances, standard deviations, and tolerance percent variation of nominal are shown.

Variable	Nominal		Worst Case	Standard Deviation	Percent Variation
Ly	2.530	**δLy**	0.0760	0.0260	3.00%
wy	1.000	**δwy**	0.0300	0.0100	3.00%
ty	0.030	**δty**	0.0009	0.0003	3.00%
Ey	3.E+07	**δEy**	0.0000	0.0000	0.00%

Statistical analysis of the force equation requires equation 3.19. The results of the statistical analysis of the nesting force for the Pinned Block and the associated comparison to a 100,000 run Monte Carlo simulation can be seen Table 3.6. The 100,000 run Monte Carlo simulation was run with the independent variables varying according to a normal distribution and their respective standard deviations.

The worst-case analysis was performed using equations 3.16 and 3.18. The results for this analysis of the force can be found in Table 3.7. These results were compared against a 500,000 run Monte Carlo simulation assuming a uniform distribution for the independent variables. As stated previously, only the limits are considered, and if the Monte Carlo limits are within the DLM predictions, then the comparison is considered ‘Good’, if the DLM value under-predicts the value from the Monte Carlo then the comparison is stated as ‘Low/High’ respectively for the High/Low limits.

The results given in Table 3.6 and Table 3.7 show the nesting force loop analysis provides good predictions of both the standard deviation and the worst-case limits of the nesting force.

Table 3.6 - Nesting force statistical analysis with respect to the standard deviation compared with a 100,000 run Monte Carlo simulation.

	Predicted σ	**Monte Carlo σ**	**Percent Difference**
σ_{Fy}	0.35563	0.3596	1.103%

Table 3.7 - Comparison of Worse Case analysis for the nesting force using a 500,000 run Monte Carlo Simulation

	DLM High	**MC High**	**Compare High**	**DLM Low**	**MC Low**	**Compare Low**
F_y	4.4860	4.8373	Low	0.4303	0.6150	Good
Good = DLM is provides a conservative estimate of the worst-case condition. **Low/High** = DLM under estimates the extent of the worst-case condition.						

A summary of the method now follows.

3.5 Summary of Method for Designing Nesting Forces

The method for the design of nesting forces using active smart features can be summarized as follows:

1. Perform assembly design synthesis for the exactly constrained design.
2. Determine the placement and configuration of active smart features that will provide the nesting force based on the mechanical stability of the design.
3. Apply part DRFs and kinematic joints in the model representing the contacts between parts including the nesting force feature(s).
4. Form kinematic loops according to the principles in the DLM. Also:
    i. There must be one nesting force loop for each feature that applies a nesting force.
    ii. Nesting force loops must contain all of the independent variables that locate the nesting force features.
5. Write the kinematic equations from the respective loops. These equations will exist in the forms $h_{x,y,\theta}(x,u)=0$ for closed loops, $p_{x,y,\theta}(x,u)=GAP$ for open loops, and for the nesting forces loops, $g_{x,y,\theta}(x,u,v)=0$.
6. Analyze the closed and open loops by performing the DLM on the $h_{x,y,\theta}(x,u)=0$ and $p_{x,y,\theta}(x,u)=GAP$ equations according to the process outlined in the DLM to find the respective sensitivities.
7. Analyze the nesting force loops by performing the DLM nesting force loop analysis according to equation 3.14 on the $g_{x,y,\theta}(x,u,v)$ equations.
8. Analyze the forces according to statistical and/or worst case analysis to determine the best tolerances for the independent variables that will achieve design requirements and supply a robust design.

3.6 Summary

This chapter has explained an effective method for the analysis and design of nesting forces in exactly constrained designs using active smart features. The method has been illustrated using the Pinned Block Assembly. The method has also been compared to a Monte Carlo simulation with good results. The following chapter will present case studies that will further illustrate the method and verify its validity.

Chapter 4 Case Studies

4.1 Example 1 -- 1D Latch

The 1D Latch will be the first example used to illustrate the design of nesting forces using active smart features. The graphical representation of this 1D Latch is illustrated in Figure 4.1.

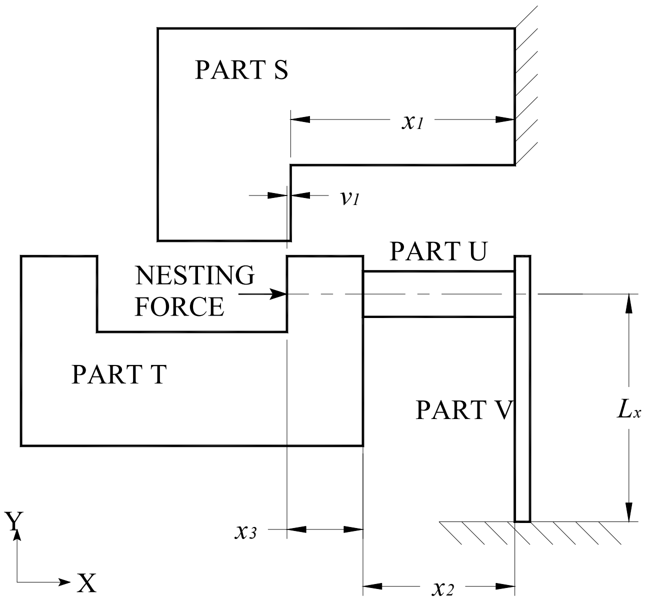


Figure 4.1 - 1D Latch Assembly.

STEPS 1 & 2 – Perform Assembly Design Synthesis & nesting Force Placement

This example consists of a simple latch mechanism consisting of four parts. Part S is the upper latch, Part T is the lower latch, Part U is a rigid connector between Parts T and V, and finally Part V is a cantilevered beam acting as the active smart feature that will provide a robust nesting force while absorbing the unwanted variation in the only assembly dimension v_1. The assembly dimension v_1 represents the deflection the active smart feature will experience to produce a nesting force to maintain the assembly as a solid stack-up. This problem requires that Parts T & U are always in compression between Parts S and V.

STEPS 3 & 4 – Apply Part DRFs and Kinematic Joints & Form Kinematic Assembly Vector Loops

With the definition of the assembly complete and the location of the nesting force determined, a vector loop is now required to indicate the geometric relationships between parts and how forces are transmitted through the assembly. Since this is a 1D problem, application of part DRFs and kinematic joints to the assembly will be skipped because it strictly involves the one-dimensional lengths of the indicated features.

STEP 5 – Write Kinematic Equations

The vector loop for the Latch is simple and results in a simple linear equation for v_1. This simple linear equation for the latch assembly is shown in equation 4.21. Equation 4.22 represents the nesting force supplied by the cantilevered beam, Part V.

$$v_1 = x_1 - x_2 - x_3 \tag{4.21}$$

$$NF = v_1 \cdot \frac{w_y \cdot t_y^{\,3} \cdot E}{4 \cdot L_y^{\,3}} \tag{4.22}$$

This Latch is a 1D problem; therefore the only dimensions and variations that are of interest are along the x-axis. A one-dimensional problem also means that all matrix equations are reduced to simple scalar equations. The nominal dimension values and their associated tolerances for this example are given in Table 4.1.

Table 4.1 - Nominal part dimensions and associated tolerances for the 1D Latch example.

Inputs		Nominal		WC	σ	Percent of Nominal
x₁	(mm)	29.50	δ***x₁***	0.885	0.295	3.000%
x₂	(mm)	10.00	δ***x₂***	0.300	0.100	3.000%
x₃	(mm)	20.00	δ***x₃***	0.600	0.200	3.000%
t_y	(mm)	1.00	δt_y	0.030	0.010	3.000%
w_y	(mm)	10.00	δw_y	0.300	0.100	3.000%
L_y	(mm)	50.00	δL_y	1.500	0.500	3.000%

STEP 6 & 7 – Perform Analysis of the Loop Equation

Before the DLM is applied to equation 4.21, equation 4.21 must be rewritten in the form Σ = 0. This is done and the DLM is applied to the result, as given in equation 4.23. Because this is a linear equation, the predicted results from the DLM will be exact.

$$\delta h_x = \frac{\partial h_x}{\partial x_1}\delta x_1 - \frac{\partial h_x}{\partial x_2}\delta x_2 - \frac{\partial h_x}{\partial x_3}\delta x_3 - \frac{\partial h_x}{\partial v_1}\delta v_1 \qquad (4.23)$$

The result of this process can be rearranged for dv_1 so there will now be an explicit expression for dv_1. This is found in equation 4.24.

$$\delta \text{v}_1 = \delta x_1 - \delta x_2 - \delta x_3 \qquad (4.24)$$

For worst-case analysis, the absolute values of the sensitivities are taken, as shown in equation 4.25.

$$\begin{aligned} \delta v_1 &= |1| \cdot \delta x_1 + |-1| \cdot \delta x_2 + |-1| \cdot \delta x_3 \\ \delta v_1 &= 0.885 + 0.300 + 0.600 = 1.785 \end{aligned} \qquad (4.25)$$

The standard deviation can be predicted in a similar manner using equation 2.9. This computation is given in equation 4.26.

$$\begin{aligned}\sigma_{v_1} &= \sqrt{\left(\frac{\partial v}{\partial x_1}\sigma_1\right)^2 + \left(\frac{\partial v}{\partial x_2}\sigma_2\right)^2 + \left(\frac{\partial v}{\partial x_3}\sigma_3\right)^2} \\ \sigma_{v_1} &= \sqrt{(1\cdot 0.079)^2 + (1\cdot .017)^2 + (1\cdot 0.034)^2} = 0.0877\end{aligned} \tag{4.26}$$

The result of equation 4.26 was compared to a 100,000 run Monte Carlo simulation where the independent variables were allowed to follow normal distributions with their respective standard deviations. The result of equation 4.25 was compared to a 500,000 run Monte Carlo simulation where the independent variables were allow to follow uniform distributions. The results of these comparisons can be seen in Table 4.2 and Table 4.3.

Table 4.2 - Comparison of the predicted standard deviation to a 100,000 run Monte Carlo Simulation.

	Nominal	**DLM σ**	**MC σ**	**Percent Difference**
***v*₁**	-0.5000	0.3702	0.3725	0.6236%

Table 4.3 - Comparison of the predicted worst-case to a 500,000 run Monte Carlo Simulation.

	Nominal	**DLM High**	**MC High**	**Compare**	**DLM Low**	**MC Low**	**Compare**
***v*₁**	0.5000	2.2850	2.2624	Good	-1.2850	-1.2522	Good
Good = DLM is provides a conservative estimate of the worst-case condition. **Low/High** = DLM under estimates the extent of the worst-case condition.							

The comparisons found in Table 4.2 and Table 4.3 show good agreement between the predicted results and those returned from the Monte Carlo simulations. However, the result of equation 4.25 shows that there will be a problem with the nesting force. Nominally, v_1 equals 0.500 mm. The predicted worst-case Low for v_1 equals $6.5-1.785=-1.285$ mm. This result indicates a potential for some assemblies to have a nesting force of zero. A design requirement states there must always be a nesting force. Therefore the tolerances of x_1, x_2, and x_3 are reduced; the revised tolerances are shown in Table 4.4. These new values result in a predicted Low of 0.1140 mm. This new Low indicates there will always be a nesting force for the Latch.

Table 4.4 - Revised tolerances for the independent variables in the Latch.

Inputs	**Nominal**		**WC**	**σ**	**Percent of Nominal**
x_1 (mm)	29.50	δx_1	0.236	0.079	0.800%
x_2 (mm)	10.00	δx_2	0.050	0.017	0.500%
x_3 (mm)	20.00	δx_3	0.100	0.034	0.500%

STEP 8 – Perform Force Analysis

The results from the DLM analysis can now be used to calculate both the worst-case values and standard deviation for the nesting force from the values calculated for the deflection, v_1. This is done by applying the nominal values and the associated variations in independent variables L_y, t_y and w_y to equations 3.16 and 3.18. The equations and resulting values are given in Table 4.5. Also, in order to predict the standard deviation of the force, the first order uncertainty analysis was performed by applying equation 3.19 to equation 4.22. The computed values are summarized in Table 4.6.

Table 4.5: Limits of the force as returned by the worst-case analysis.

Application of Equations 3.16 and 3.17	Equation Results
$F_{\max} = \left(v + \delta v\right) \cdot \frac{\left(h + \delta h\right) \cdot (t + \delta t)^3 \cdot E}{4 \cdot (L - \delta L)^3}$	4.4251 N
$F_{\min} = \left(v - \delta v\right) \cdot \frac{\left(h - \delta h\right) \cdot (t - \delta t)^3 \cdot E}{4 \cdot (L + \delta L)^3}$	0.3741 N

Comparisons were done against a Monte Carlo simulation for both the worst case and the statistical analysis. For the comparison of the standard deviation, a 100,000 run Monte Carlo simulation was performed. This simulation applied normal distributions to the independent variables. The results are shown in Table 4.6.

Table 4.6 - Statistical comparison of DLM results with 100,000 run Monte Carlo Simulation.

	Nominal Value	**DLM σ**	**Monte Carlo σ**	**Percent Difference**
v_1	-0.5000	0.087670	0.087433	0.2712%
F_x	2.0250	0.365870	0.365878	0.0023%

The worst case analysis was compared to a 500,000 run Monte Carlo simulation where the independent variables were assumed to have a uniform distribution. The results of this comparison are given in Table 4.7. These results for a one-dimensional tolerance analysis show that the method accurately predicts both the worst case and statistical variation in the deflection and the force.

Table 4.7 - Worst case analysis comparison with 500,000 run Monte Carlo simulation.

	Nominal	**DLM High**	**MC High**	**Compare High**	**DLM Low**	**MC Low**	**Compare Low**
v_1	0.5000	0.8860	0.8823	Good	0.1140	0.1178	Good
F_x	2.0250	4.42509	4.2452	Good	0.37405	0.4408	Good
Good = DLM is provides a conservative estimate of the worst-case condition. **Low/High** = DLM under estimates the extent of the worst-case condition.							

This example has illustrated the method for a 1D problem that requires a nesting force. The design used a single cantilevered beam as an active smart feature. The following example, an Exactly Constrained Block, will illustrate the method in two dimensions, as well as explain the application of multiple nesting forces.

4.2 Exactly Constrained Block

STEP 1 – Perform Assembly Design Synthesis

The exactly constrained block assembly consists of a base that constrains a block with three simple constraints. The assembly is illustrated in Figure 4.2. For this design, each constraint can be considered part of the rigid base.

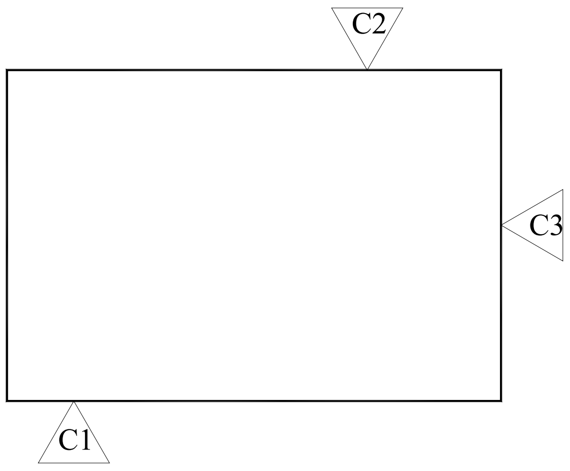


Figure 4.2 - Basic assembly for the exactly constrained block.

STEP 2 – Determine Placement and Configuration of Nesting Force

It is clear for the exactly constrained block that a nesting force will be required to keep the block seated against the constraints of the base. After a careful observation, the location of the nesting force is determined to be in the location shown in Figure 4.3. However, cantilevered beams have been chosen as the active smart features to provide the nesting force, and they can only provide a force that is normal to the surface. Therefore, two active smart features will be used to obtain the required nesting force resultant. The placement of the cantilevered beams is illustrated in Figure 4.4.

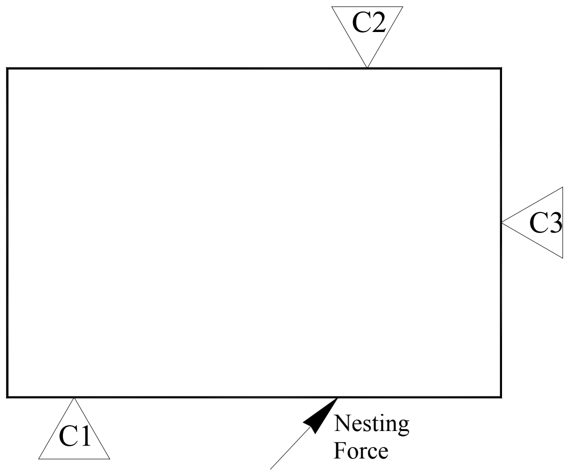


Figure 4.3 - Exactly constrained block with applied nesting force.

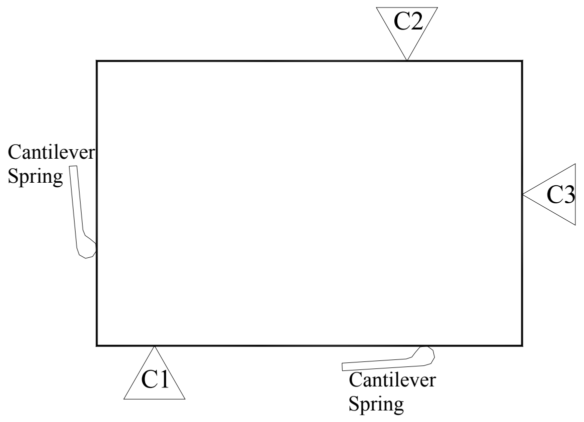


Figure 4.4 - Placement of cantilevered beams used to provide the required nesting force resultant.

With the general design done and the placement of the active smart features determined, dimensions are applied to the exactly constrained block. The dimensions are displayed in Figure 4.5. The dependent dimension u_4 represents the rotation of the block caused when the independent dimensions vary in size. The block is shown at its nominal angle of 0°. The values of all independent variables and their associated worst-case tolerances and standard deviations are given in Table 4.8.

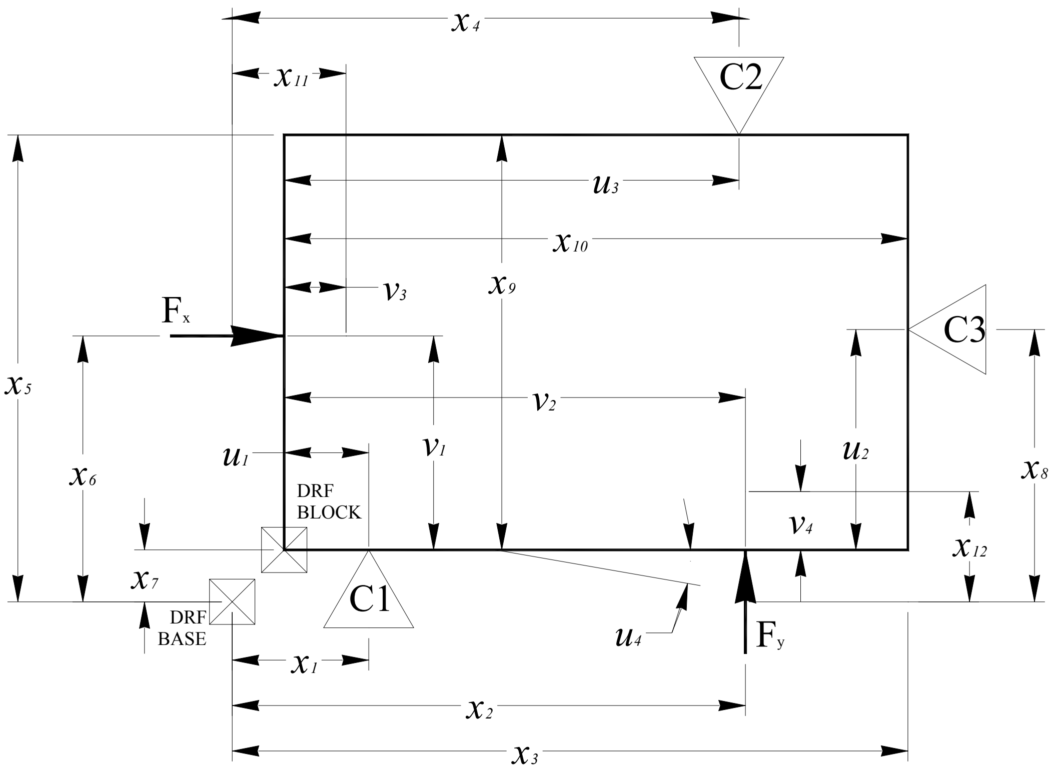


Figure 4.5 - Exactly constrained block with dimension labels applied.

STEP 3 – Apply Part DRFs and Kinematic Joints

Now that the assembly is designed and dimensioned, the DLM model can be developed. Model setup starts with establishment of Datum Reference Frames for each part and the application of kinematic joints for the contacts between parts. For the Exactly Constrained Block, the joints will be represented by the constraints. Both the part DRFs and the kinematic joints are shown in Figure 4.5.

Table 4.8 - Nominal values for independent dimensions and their respective tolerances and standard deviations.

	Nominal Value			± δX	σX	Percent Variation
x_1	2.625		δx_1	0.079	0.027	3.00%
x_2	10.000		δx_2	0.3	0.1	3.00%
x_3	13.000		δx_3	0.39	0.13	3.00%
x_4	9.750		δx_4	0.293	0.098	3.00%
x_5	9.000		δx_5	0.27	0.09	3.00%
x_6	3.000		δx_6	0.09	0.03	3.00%
x_7	1.000		δx_7	0.03	0.01	3.00%
x_8	5.250		δx_8	0.158	0.053	3.00%
x_9	8.000		δx_9	0.24	0.08	3.00%
x_{10}	12.000		δx_{10}	0.36	0.12	3.00%
x_{11}	1.290		δx_{11}	0.039	0.013	3.00%
x_{12}	1.200		δx_{12}	0.036	0.012	3.00%

STEP 4 – Form Kinematic Assembly Vector Loops

Kinematic loops are established using the principles outlined in Sections 2.6 and 3.2.4.1. For the Block there are 3 joints and 2 parts; thus $L = J - P + 1 = 3 - 2 + 1 = 2$ loops; therefore, two closed kinematic vector loops are needed to establish the relative position of the parts in the assembly. Since there are two nesting forces; according to the principles outlined in Section 3.2.4.1, two nesting force loops are required. There are no open loops for the Exactly Constrained Block assembly. The two closed vector loops are illustrated in Figure 4.6 and the two nesting force loops can be seen in Figure 4.7.

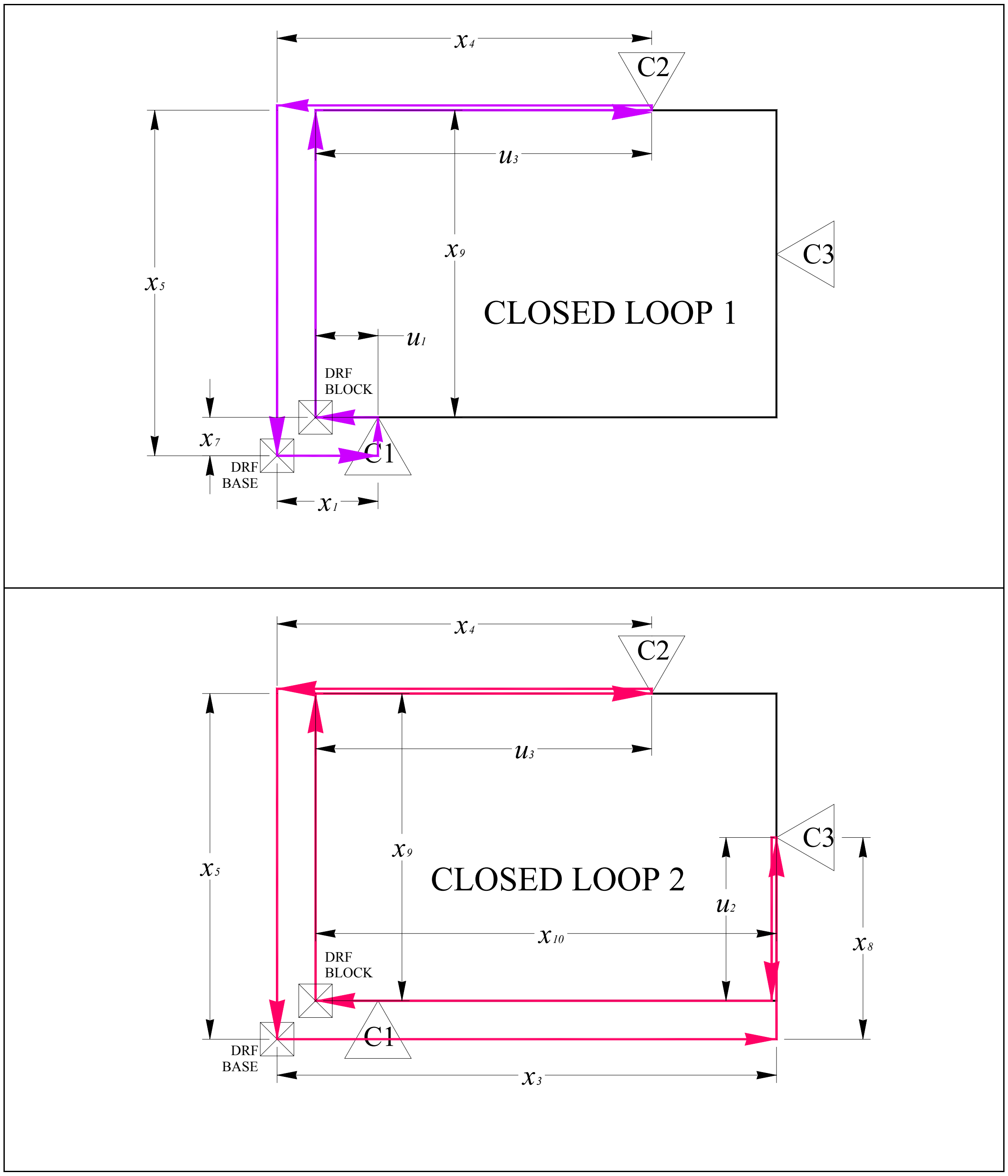


Figure 4.6 - Closed loops and their associated dimensions for the exactly constrained block.

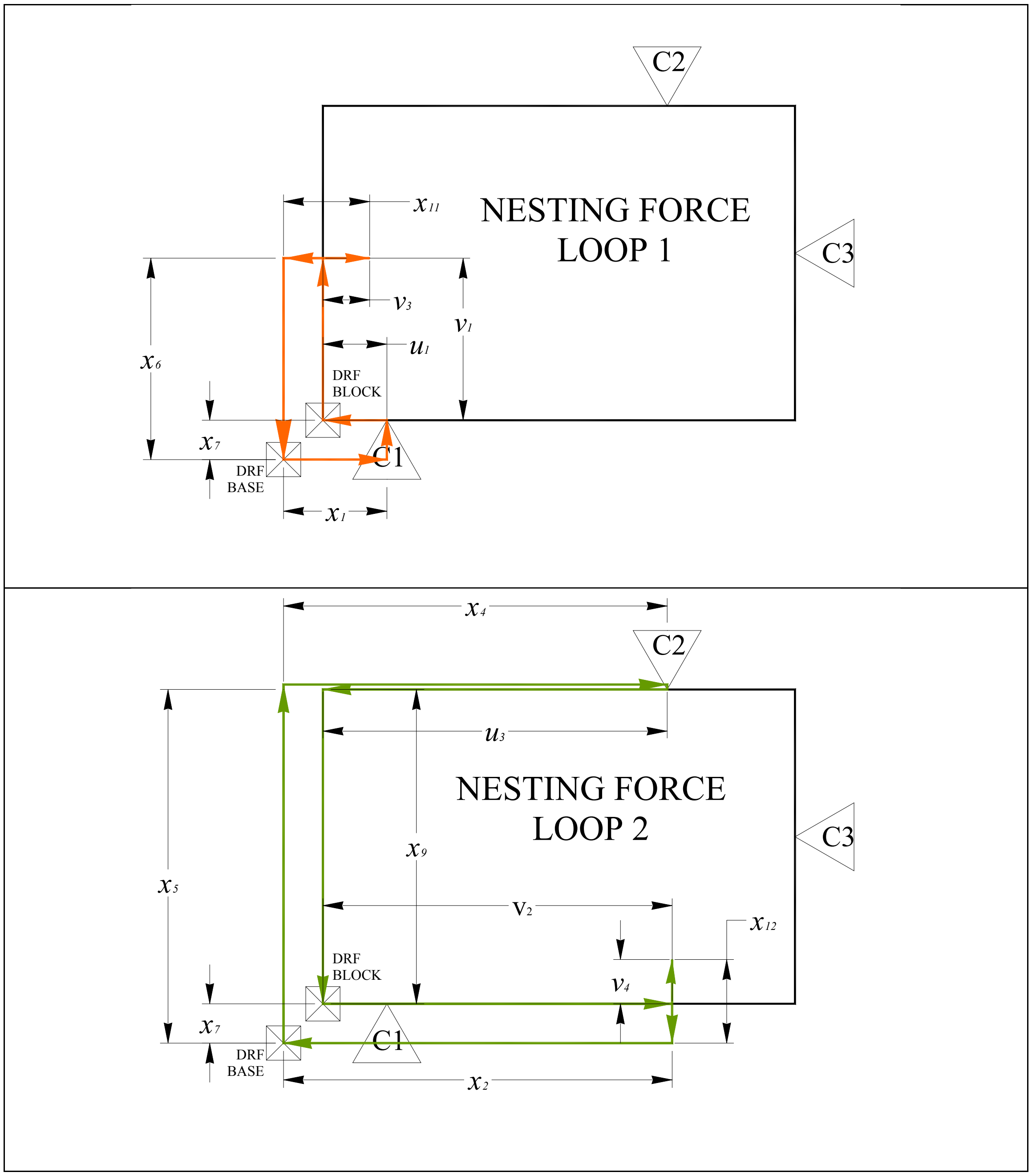


Figure 4.7 - Nesting force loops for the exactly constrained block.

Note that there are four v_i dependent variables for the two nesting force loops. For each loop, one v_i represents the deflection of the cantilevered beam. The vertical nesting force, F_y, is the result of the deflection, v_4. The horizontal nesting force, or F_x, is the result of the deflection, v_3.

STEP 5 – Write Kinematic Equations

Now that the kinematic vector loops have been formed, the equations for the respective loops can be written. The equations for the Closed Loop 1 and the Nesting Force Loop 1, evaluated in terms of known angles, can be seen in equations 4.27 and 4.28. The development of all vector loops and subsequent symbolic and numerical analysis were performed using Maple© and can be found in Appendix B.

$$\begin{aligned} h_{1x} &= x_1 - u_1 \cdot \cos(u_4) - x_9 \cdot \sin(u_4) + u_3 \cdot \cos(u_4) - x_4 = 0 \\ h_{1y} &= x_7 - u_1 \cdot \sin(u_4) + x_9 \cdot \cos(u_4) + u_3 \cdot \sin(u_4) - x_5 = 0 \end{aligned} \tag{4.27}$$

$$\begin{aligned} g_{1x} &= x_1 - x_{11} - v_3 - u_1 \cdot \cos(u_4) - v_1 \cdot \sin(u_4) = 0 \\ g_{1y} &= x_7 - x_6 - u_1 \cdot \sin(u_4) + v_1 \cdot \cos(u_4) = 0 \end{aligned} \tag{4.28}$$

STEP 6 – Perform Analysis of the Closed and Open Loop Equations

The setup of the DLM model is now finished and the model can now be analyzed. The $\left[\frac{\partial h}{\partial x}\right]$, $\left[\frac{\partial h}{\partial u}\right]$ and the $\left[\frac{\partial h}{\partial u}\right]^{-1}$ matrices are created and each of these matrices is given in both its symbolic and numerical representations in Appendix B. The values for the closed loop sensitivity matrix, equal to $-\left[\frac{\partial h}{\partial u}\right]^{-1} \cdot \left[\frac{\partial h}{\partial x}\right]$, are given in Table 4.9.

STEP 7 – Perform Analysis of the Nesting Force Loop Equations

With the $-\left[\frac{\partial h}{\partial u}\right]^{-1} \cdot \left[\frac{\partial h}{\partial x}\right]$ matrix formed, the DLM is applied to the nesting force loop equations. This requires the formation of the $\left[\frac{\partial g}{\partial x}\right]$, $\left[\frac{\partial g}{\partial u}\right]$, $\left[\frac{\partial g}{\partial v}\right]$ and the $\left[\frac{\partial g}{\partial v}\right]^{-1}$ matrices. As with the closed loop matrices, these can be found in Appendix B. The sensitivities for the dependent variables in the nesting force loop analysis are shown in Table 4.10.

Table 4.9 - Sensitivities for the closed loop dependent variables.

$$-\left[\frac{\partial h}{\partial u}\right]^{-1}\cdot\left[\frac{\partial h}{\partial x}\right]$$ Matrix:

	x_1	x_2	x_3	x_4	x_5	x_6
u_1	-1.0000	0.0000	1.0000	0.0000	0.5965	0.0000
u_2	0.0000	0.0000	0.0000	0.0000	1.4561	0.0000
u_3	0.0000	0.0000	1.0000	-1.0000	-0.5263	0.0000
u_4	0.0000	0.0000	0.0000	0.0000	-0.1404	0.0000

x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
-0.5965	0.0000	-0.5965	-1.0000	0.0000	0.0000
-0.4561	-1.0000	-1.4561	0.0000	0.0000	0.0000
0.5263	0.0000	0.5263	-1.0000	0.0000	0.0000
0.1404	0.0000	0.1404	0.0000	0.0000	0.0000

Table 4.10 - Sensitivity matrix for the nesting force loop dependent variables.

$$\left[\frac{\partial g}{\partial v}\right]^{-1}\cdot\left\{\left[\frac{\partial g}{\partial u}\right]\cdot\left[\frac{\partial h}{\partial u}\right]^{-1}\cdot\left[\frac{\partial h}{\partial x}\right]-\left[\frac{\partial g}{\partial x}\right]\right\}$$ Matrix:

	x_1	x_2	x_3	x_4	x_5	x_6
v_1	0.000	0.000	0.000	0.000	0.228	1.000
v_2	0.000	1.000	-1.000	0.000	-0.596	0.000
v_3	0.000	0.000	1.000	0.000	0.316	0.000
v_4	0.000	0.000	0.000	0.000	1.035	0.000

x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
-1.228	0.000	-0.228	0.000	0.000	0.000
0.596	0.000	0.596	1.000	0.000	0.000
-0.316	0.000	-0.316	-1.000	-1.000	0.000
-0.035	0.000	-1.035	0.000	0.000	-1.000

The results of the both the worst-case and statistical analysis for dependent variables u_i and v_i using equations 2.8 and 2.9 are given in Table 4.11.

Table 4.11 - Results of the DLM closed loop and nesting force loop analysis for the exactly constrained block assembly.

	Nominal Value		DLM WC	DLM σ
u_1	1.625	δu_1	1.151	0.1929
u_2	4.250	δu_2	0.914	0.1832
u_3	8.750	δu_3	1.327	0.2120
u_4	0.000	δu_4	0.076	0.0170
v_1	2.000	δv_1	0.243	0.0425
v_2	9.000	δv_2	1.372	0.2156
v_3	0.290	δv_3	0.960	0.1815
v_4	0.200	δv_4	0.565	0.1252

The results of both the closed and nesting force loop analysis can now be compared against Monte Carlo simulations. The standard deviations are compared against a 100,000 run simulation where the independent variables were allowed to randomly follow their respective normal distributions. The worst-case limits were compared to a 500,000 run simulation where the independent variables followed uniform distributions. The results of these comparisons are seen in Table 4.12 and Table 4.13, respectively.

Table 4.12 - Comparison of the predicted standard deviation against a 100,000 run Monte Carlo simulation for the exactly constrained block.

	DLM σ	**MC σ**	**Percent Difference**
u_1	0.1929	0.1927	0.1120%
u_2	0.1832	0.1825	0.3999%
u_3	0.2120	0.2118	0.1109%
u_4	0.0170	0.0169	0.2579%
v_1	0.0425	0.0426	0.3721%
v_2	0.2156	0.2158	0.0661%
v_3	0.1815	0.1812	0.1307%
v_4	0.1252	0.1249	0.2621%

Table 4.13 - Comparison of the predicted worst-case limits to a 500,000 run Monte Carlo Simulation for the exactly constrained block.

	DLM High	**MC High**	**Compare**	**DLM Low**	**MC Low**	**Compare**
u_1	2.7761	1.6308	Good	0.4739	0.6111	Good
u_2	5.1643	4.2537	Good	3.3357	3.3482	Good
u_3	10.0772	8.7518	Good	7.4228	7.5190	Good
u_4	0.0758	0.0744	Good	-0.0758	-0.0817	High
v_1	2.2432	1.9964	Good	1.7568	1.7021	High
v_2	10.3721	9.0089	Good	7.6279	7.7570	Good
v_3	1.2495	0.2954	Good	-1.2495	-1.2552	High
v_4	0.7649	0.2039	Good	-0.7649	-0.8040	High
Good = DLM is provides a conservative estimate of the worst-case condition. **Low/High** = DLM under estimates the extent of the worst-case condition.						

The results in Table 4.12 and Table 4.13 show overall good results with the percent difference in the predicted standard deviation an average of 0.214%. The worst case conservatively estimates the High values well, but falls slightly high on several of the Low values. Even with the slight under-estimation of the low values for worst-case, the results are still adequate for design purposes. However, there is a noticeable problem with the predicted worst-case conditions where the possibility exists that some assemblies will not have a nesting force because of a gap instead of a deflection. Therefore, tolerances of some independent variables were adjusted. The revised tolerances on the independent variables are given in Table 4.14. The new variations in the dependent variables are given in Table 4.15.

Table 4.14 - The set of all independent variables and associated tolerances. The revised tolerances are highlighted.

	Nominal Value			**± δX**	**σX**	**Percent Variation**
x_1	2.625		δx_1	0.079	0.027	3.00%
x_2	10.000		δx_2	0.300	0.100	3.00%
x_3	13.000		δx_3	**0.065**	**0.022**	**0.50%**
x_4	9.750		δx_4	0.293	0.098	3.00%
x_5	9.000		δx_5	**0.045**	**0.015**	**0.50%**
x_6	3.000		δx_6	0.090	0.030	3.00%
x_7	1.000		δx_7	**0.010**	**0.004**	**1.00%**
x_8	5.250		δx_8	0.158	0.053	3.00%
x_9	8.000		δx_9	**0.080**	**0.027**	**1.00%**
x_{10}	12.000		δx_{10}	**0.060**	**0.020**	**0.50%**
x_{11}	1.290		δx_{11}	**0.007**	**0.003**	**0.50%**
x_{12}	1.200		δx_{12}	**0.012**	**0.004**	**1.00%**

Table 4.15 - The revised variations for the dependent variables for the exactly constrained block.

	Nominal Value		**DLM WC**	**DLM σ**
u_1	1.625	δu_1	0.285	0.0443
u_2	4.250	δu_2	0.345	0.0695
u_3	8.750	δu_3	0.489	0.1037
u_4	0.000	δu_4	0.019	0.0044
v_1	2.000	δv_1	0.131	0.0312
v_2	9.000	δv_2	0.506	0.1060
v_3	0.290	δv_3	0.175	0.0315
v_4	0.200	δv_4	0.142	0.0322

STEP 8 – Perform Force Analysis

The results from the DLM show that the revised variations in the two deflections, v_3 and v_4 are now less than the nominal value; therefore there will always be a force at the desired locations. The force analysis can now be performed using these results. Cantilevered beams are being used as active smart features to supply the required nesting forces and their respective force output is given in equations 3.16 and 3.17. The independent variables that describe these beams along with their worst-case tolerances and standard deviations are given in Table 4.16.

Table 4.16 - Values for the independent variables and their respective tolerances and standard deviations for the cantilevered beams in the exactly constrained block assembly.

***X*-axis**	**Nominal Value**			**Worst Case Value**	**Standard Deviation Values**	**Percent Variation**
L_x	1.980		δL_x	0.0600	0.0200	3.00%
w_x	1.000		δw_x	0.0300	0.0100	3.00%
t_x	0.025		δt_x	0.0008	0.0003	3.00%
E_x	3.0E+07		δE_x	0.0000	0.0000	0.00%
***Y*-axis**						
L_y	2.530		δL_y	0.0760	0.0260	3.00%
w_y	1.000		δw_y	0.0300	0.0100	3.00%
t_y	0.030		δt_y	0.0009	0.0003	3.00%
E_y	3.0E+07		δE_y	0.0000	0.0000	0.00%

Calculation of the worst-case and statistical variations in the nesting forces can be performed by applying the values from Table 4.11 for v_3 and v_4, and the respective x and y values from Table 4.16, to equations 3.16, 3.18, and 3.19. The results of these calculations are shown in Table 4.17. Also included in Table 4.17 are estimates for the worst-case variation in the resultant force, F_r and the angle at which the resultant acts, θ_F. These values are provided for design purposes, to verify the resultant force is as desired. No statistical analysis was done for the resultant force.

Table 4.17 - Results of the DLM force analysis, including estimates for the resultant force, F_r and the angle of the resultant force, θ_F.

	Nominal	**Maximum**	**Minimum**	σ
F_x	4.3781	8.7089	1.4011	0.5196
F_y	2.5009	5.2704	0.5902	0.4178
F_r	5.0420	10.1795	1.5203	N.A.
θ_F (r)	0.5190	0.5442	0.3987	N.A.
θ_F (°)	29.74	31.18	22.84	N.A.

Again, two Monte Carlo simulations were performed to validate the results. For the statistical analysis, a 100,000 run Monte Carlo simulation was performed; the results are given in Table 4.18. For the worst-case analysis, a 500,000 run simulation was performed. The worst-case values are compared in Table 4.19. The results show that the DLM accurately estimates the standard deviations, with an average percent difference of only 0.255% for all the dependent variables. The worst-case comparison shows some error but for the DLM part of the analysis the under-estimation is small and could reliably be used for the purposes of design.

Table 4.18 - Comparison of DLM to 100,000 run Monte Carlo simulation for the standard deviation (σ) for all the dependent variables for the Exactly Constrained Block.

	DLM σ	**Monte Carlo σ**	**Percent Difference**
u_1	0.0443	0.0443	0.1819%
u_2	0.0695	0.0696	0.1600%
u_3	0.1037	0.1035	0.1688%
u_4	0.0044	0.0044	0.1485%
v_1	0.0312	0.0311	0.2853%
v_2	0.1060	0.1064	0.3814%
v_3	0.0315	0.0316	0.3533%
v_4	0.0322	0.0322	0.0616%
F_x	0.5196	0.5224	0.5494%
F_y	0.4178	0.4188	0.2564%

Table 4.19 - Comparison of DLM to 500,000 run Monte Carlo simulation for the worst-case conditions for all the dependent variables for the exactly constrained block.

	DLM High	**MC High**	**Compare**	**DLM Low**	**MC Low**	**Compare**
u_1	1.9095	1.6254	Good	1.3405	1.3717	Good
u_2	4.5946	4.2503	Good	3.9054	3.9072	Good
u_3	9.2391	8.7500	Good	8.2609	8.2843	Good
u_4	0.0189	0.0192	Low	-0.0189	-0.0196	High
v_1	2.1308	1.9997	Good	1.8692	1.8699	Good
v_2	9.5055	9.0006	Good	8.4945	8.5225	Good
v_3	0.4646	0.4570	Good	0.1154	0.1287	Good
v_4	0.3417	0.3477	Low	0.0583	0.0556	High
F_x	8.7089	4.3958	Good	1.4011	1.7623	Good
F_y	5.2704	2.5118	Good	0.5902	0.6398	Good
Good = DLM is provides a conservative estimate of the worst-case condition. **Low/High** = DLM under estimates the extent of the worst-case condition.						

4.3 Wedge and Cylinder Example

STEP 1 – Perform Assembly Design Synthesis

The Wedge and Cylinder assembly consists of a Base, Cylinder and a Wedge as shown below. The Wedge is intended to keep the cylinder firmly nested in the corner of the base as illustrated in Figure 4.8.

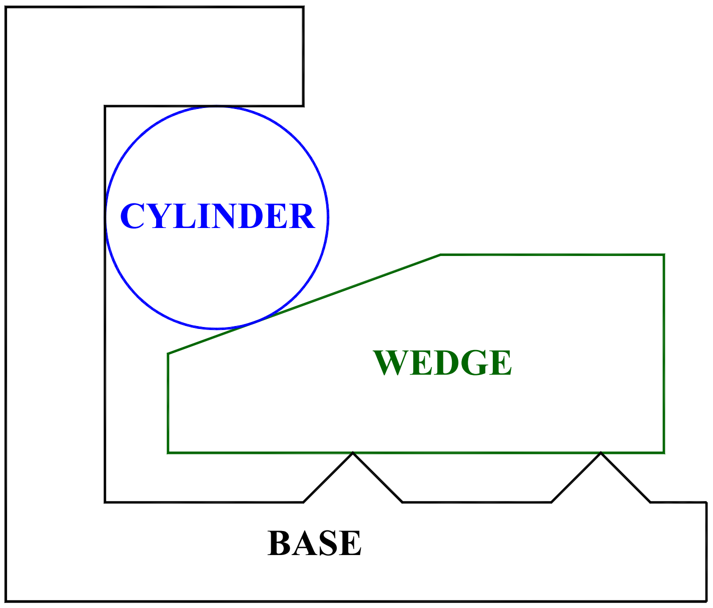


Figure 4.8 - Wedge and Cylinder assembly.

STEP 2 – Determine Placement and Configuration of Nesting Force

The placement of the nesting force in the assembly is given in Figure 4.9. It should be noted that if just a horizontal force is applied, the moment created by the Cylinder on the Wedge could un-seat the Wedge from the rear constraint. Therefore, this assembly will also require a downward nesting force that will resist the moment generated by the cylinder. The desired resultant nesting force as well as the two nesting forces that will be used to generate the resultant are displayed in Figure 4.9.

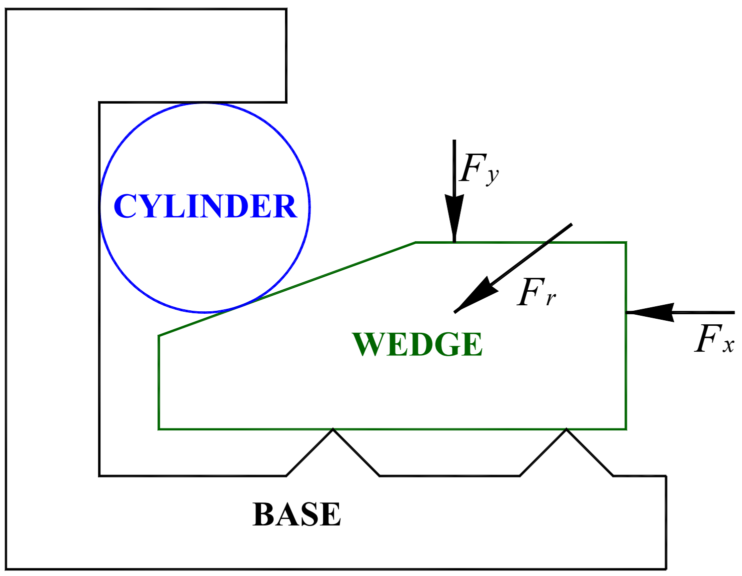


Figure 4.9 – Nesting forces applied to the Wedge assembly to obtain the desired resultant nesting force.

One possible configuration of the base, with the cantilevered beams molded into it, is shown in Figure 4.10. With the active features positioned, the assembly is now dimensioned and labeled for the DLM setup. This can be seen in Figure 4.11. The dimension, u_6, represents the rotation the block may experience due to variations in x_2 and x_4. Nominally, u_6 is 0 radians. The dependent dimensions, v_1 and v_4, represent the deflections of the active smart features in the x and y directions, respectively. The values for all the independent dimensions are shown in Table 4.20.

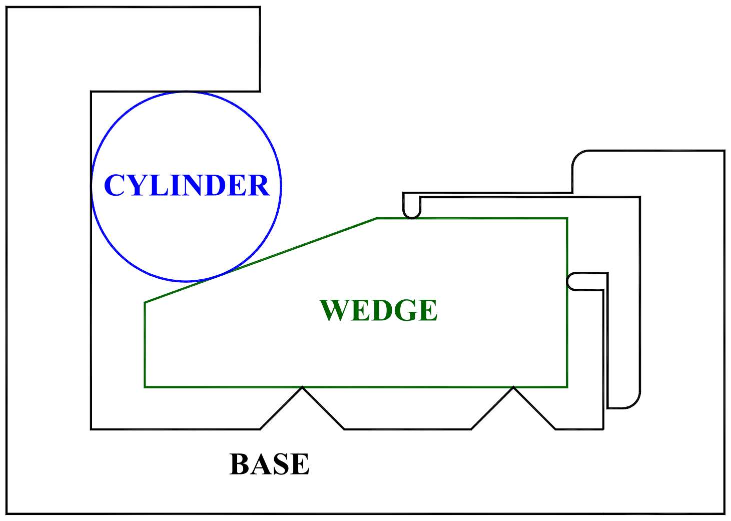


Figure 4.10 - Wedge assembly with cantilevered beams providing the required nesting forces.

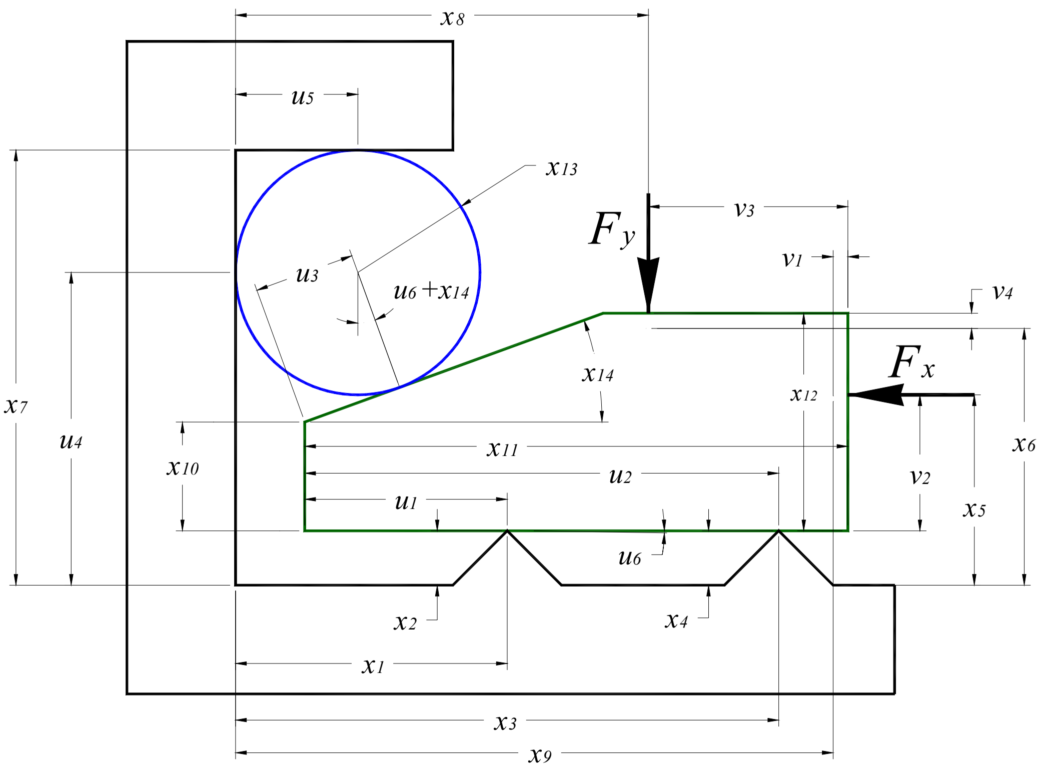


Figure 4.11 - Wedge assembly with dimensions and their respective labels.

Table 4.20 - Nominal values and the associated worst-case tolerances and standard deviations for the Wedge Assembly.

	Nominal Value			**$\pm \delta X$**	**σX**	**Percent Variation**
x_1	2.500		δx_1	0.075	0.025	3.00%
x_2	0.500		δx_2	0.015	0.005	3.00%
x_3	5.000		δx_3	0.150	0.050	3.00%
x_4	0.500		δx_4	0.015	0.005	3.00%
x_5	1.750		δx_5	0.053	0.018	3.00%
x_6	2.400		δx_6	0.072	0.024	3.00%
x_7	4.000		δx_7	0.120	0.040	3.00%
x_8	3.800		δx_8	0.114	0.038	3.00%
x_9	5.620		δx_9	0.169	0.057	3.00%
x_{10}	1.000		δx_{10}	0.030	0.010	3.00%
x_{11}	5.100		δx_{11}	0.153	0.051	3.00%
x_{12}	2.000		δx_{12}	0.060	0.020	3.00%
x_{13}	1.125		δx_{13}	0.034	0.012	3.00%
x_{14}	0.349		δx_{14}	0.011	0.004	3.00%

STEP 3 – Apply Part DRFs and Kinematic Joints

Part DRFs and kinematic joints are now applied to the model and can be seen in Figure 4.12.

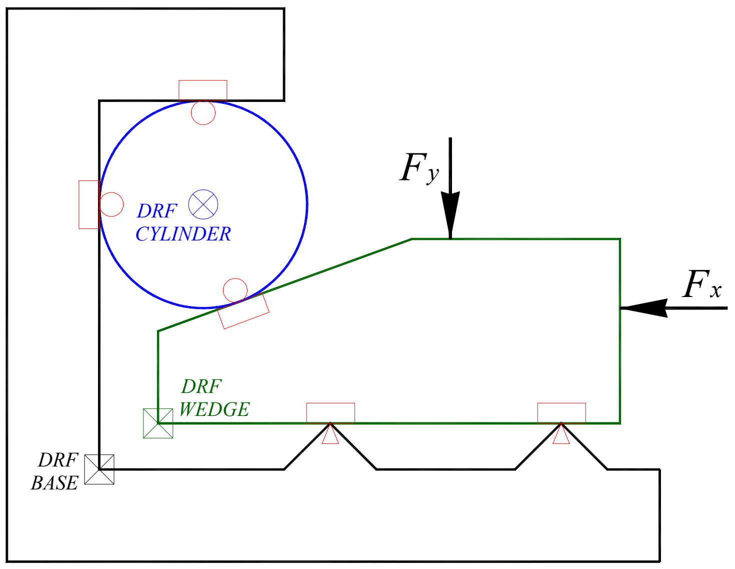


Figure 4.12 - Part DRFs and kinematic joints applied to the Wedge Assembly.

STEP 4 – Form Kinematic Assembly Vector Loops

Kinematic loops are now applied to the model. This assembly has two parts, five joints, and two nesting forces; therefore there will be 3 closed loops and two nesting force loops. The first two closed loops are illustrated in Figure 4.13. The third closed loop can be seen in Figure 4.14. The two nesting force loops are displayed in Figure 4.15. The nesting force F_x is the result of the deflection v_1, and the nesting force F_y is the result of the deflection v_4.

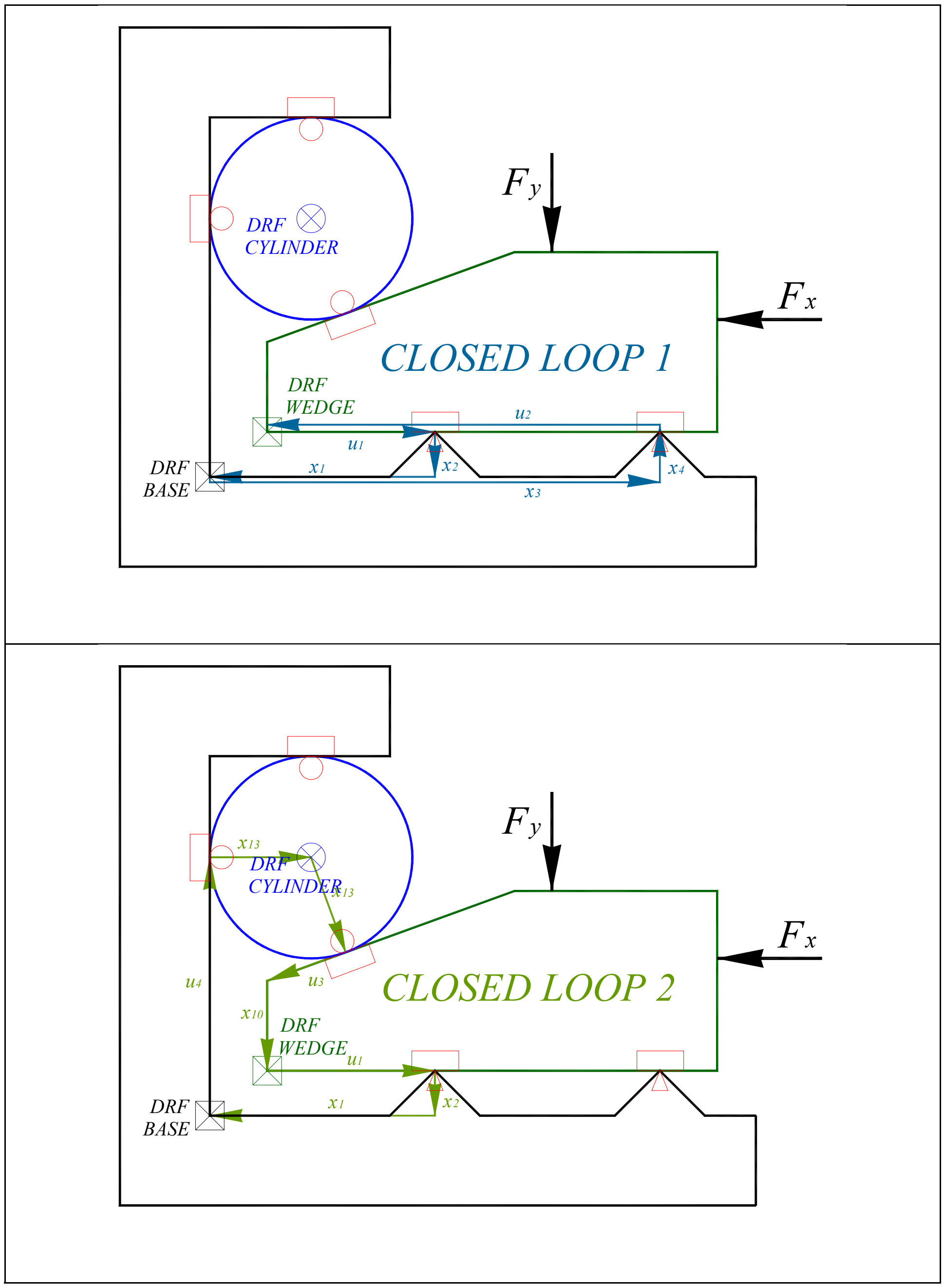


Figure 4.13 - Closed loops 1 and 2 for the Wedge Assembly.

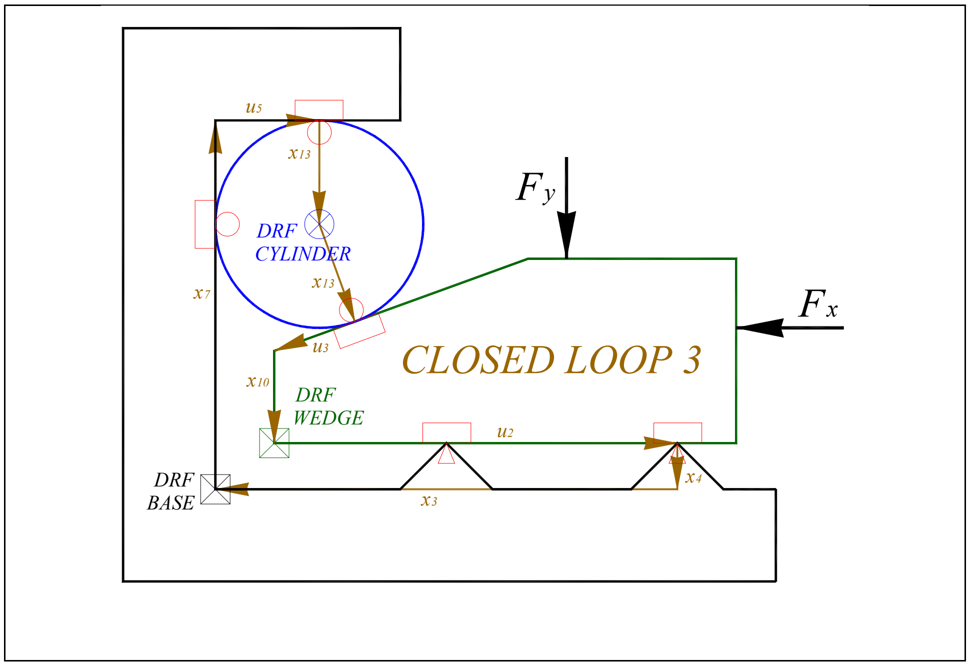


Figure 4.14 - Closed loop 3 for the Wedge Assembly.

STEP 5 – Write Kinematic Equations

Now that the kinematic vector loops have been formed, the equations for the respective loops can be written. The equations for the Closed Loop 1 and the Nesting Force Loop 1, evaluated in terms of known angles, are given in equations 4.29 and 4.30. The development of all vector loops and subsequent symbolic and numerical analysis can be found in Appendix C.

$$\begin{aligned} h_{1x} &= x_3 - u_2 \cdot \cos(u_6) + u_1 \cdot \cos(u_6) - x_1 = 0 \\ h_{1y} &= x_4 - u_2 \cdot \sin(u_6) + u_1 \cdot \cos(u_6) - x_2 = 0 \end{aligned} \tag{4.29}$$

$$\begin{aligned} g_{1x} &= u_1 \cdot \cos(u_6) + v_2 \cdot \sin(u_6) - x_{11} \cdot \cos(u_6) - x_1 + x_9 + v_1 = 0 \\ g_{1x} &= u_1 \cdot \sin(u_6) - v_2 \cdot \cos(u_6) - x_{11} \cdot \sin(u_6) - x_2 + x_5 = 0 \end{aligned} \tag{4.30}$$

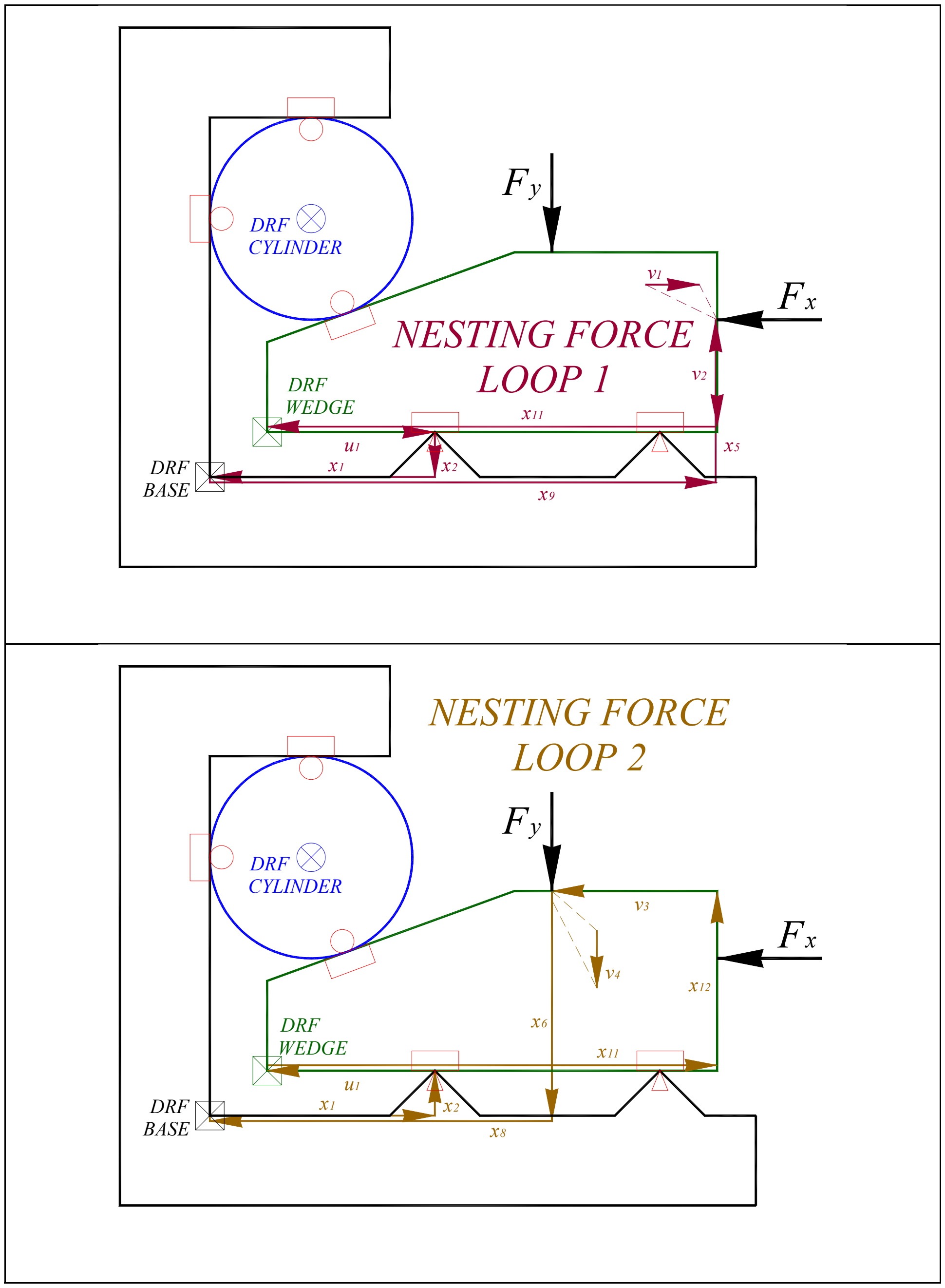


Figure 4.15 - Nesting force loops 1 and 2 for the Wedge Assembly.

STEP 6 – Perform Analysis of the Closed and Open Loop Equations

The setup of the DLM model is now finished, and the DLM is now performed on the closed loop equations that have been formed. The $\left[\frac{\partial h}{\partial x}\right]$, $\left[\frac{\partial h}{\partial u}\right]$ and the $\left[\frac{\partial h}{\partial u}\right]^{-1}$ are created and each of these matrices can be seen in both their symbolic and numerical representations in Appendix C. The values for the closed loop sensitivity matrix or $-\left[\frac{\partial h}{\partial u}\right]^{-1} \cdot \left[\frac{\partial h}{\partial x}\right]$ matrix are given in Table 4.21.

Table 4.21 - Closed loop sensitivity matrix for the Wedge Assembly.

$-\left[\frac{\partial h}{\partial u}\right]^{-1} \cdot \left[\frac{\partial h}{\partial x}\right]$ Matrix:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
u_1	1.0000	-3.3086	0.0000	0.5611	0.0000	0.0000	2.7475
u_2	0.0000	-3.3086	1.0000	0.5611	0.0000	0.0000	2.7475
u_3	0.0000	-4.5319	0.0000	1.6081	0.0000	0.0000	2.9238
u_4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000
u_5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
u_6	0.0000	-0.4000	0.0000	0.4000	0.0000	0.0000	0.0000

x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
0.0000	0.0000	-2.7475	0.0000	0.0000	-6.6713	-2.7171
0.0000	0.0000	-2.7475	0.0000	0.0000	-6.6713	-2.7171
0.0000	0.0000	-2.9238	0.0000	0.0000	-5.6713	-1.4283
0.0000	0.0000	0.0000	0.0000	0.0000	-1.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

STEP 7 – Perform Analysis of the Nesting Force Loop Equations

With the $-\left[\frac{\partial h}{\partial u}\right]^{-1}\cdot\left[\frac{\partial h}{\partial x}\right]$ matrix formed, the DLM is applied to the nesting force loop equations. This requires the formation of the $\left[\frac{\partial g}{\partial x}\right]$, $\left[\frac{\partial g}{\partial u}\right]$, $\left[\frac{\partial g}{\partial v}\right]$ and the $\left[\frac{\partial g}{\partial v}\right]^{-1}$ matrices. As with the closed loop matrices, these are shown in Appendix C. The sensitivities for the dependent variables in the nesting force loop analysis are given in Table 4.22.

Table 4.22 - Nesting force loop sensitivities for the Wedge Assembly.

$\left[\frac{\partial g}{\partial v}\right]^{-1}\cdot\left\{\left[\frac{\partial g}{\partial u}\right]\cdot\left[\frac{\partial h}{\partial u}\right]^{-1}\cdot\left[\frac{\partial h}{\partial x}\right]-\left[\frac{\partial g}{\partial x}\right]\right\}$ Matrix:

	x1	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*
v1	0.0000	3.8086	0.0000	-1.0611	0.0000	0.0000	-2.7475
v2	0.0000	0.2946	0.0000	-1.2946	1.0000	0.0000	0.0000
v3	0.0000	4.1086	0.0000	-1.3611	0.0000	0.0000	-2.7475
v4	0.0000	-0.4800	0.0000	-0.5200	0.0000	1.0000	0.0000

x8	*x9*	*x10*	*x11*	*x12*	*x13*	*x14*
0.0000	-1.0000	2.7475	1.0000	0.0000	6.6713	2.7171
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-1.0000	0.0000	2.7475	1.0000	0.0000	6.6713	2.7171
0.0000	0.0000	0.0000	0.0000	-1.0000	0.0000	0.0000

The results of the both the worst-case and statistical analysis using equations 2.8 and 2.9 are given in Table 4.23.

Table 4.23 - Results of the DLM closed loop and nesting force loop analysis for the Wedge Assembly.

	Nominal Value		**DLM WC**	**DLM σ**
u_1	1.8635	δu_1	0.8019	0.1424
u_2	4.3635	δu_2	0.8769	0.1488
u_3	0.9293	δu_3	0.7392	0.1406
u_4	2.8750	δu_4	0.1540	0.0418
u_5	1.1250	δu_5	0.0340	0.0120
u_6	0.0000	δu_6	0.0120	0.0028
v_1	0.1165	δv_1	1.0639	0.1600
v_2	1.2500	δv_2	0.0768	0.0192
v_3	1.9365	δv_3	1.0179	0.1545
v_4	0.1000	δv_4	0.1470	0.0314

The results of both the closed and nesting force loop analysis for the Wedge assembly can now be validated against Monte Carlo simulations. As with the previous examples in this work, the standard deviations are compared against a 100,000 run simulation where the independent variables were allowed to randomly follow their respective normal distributions. The worst-case limits were compared to a 500,000 run simulation where the independent variables followed uniform distributions. The results of these comparisons are seen in Table 4.24 and Table 4.25, respectively.

Table 4.24 - Comparison of the predicted standard deviation to a 100,000 run Monte Carlo simulation for the Wedge assembly.

	DLM σ	**MC σ**	**Percent Difference**
u_1	0.1424	0.1419	0.3082%
u_2	0.1488	0.1484	0.2374%
u_3	0.1406	0.1400	0.4748%
u_4	0.0418	0.0415	0.5962%
u_5	0.0120	0.0120	0.0933%
u_6	0.0028	0.0028	0.3387%
v_1	0.1600	0.1601	0.0834%
v_2	0.0192	0.0192	0.0365%
v_3	0.1545	0.1543	0.1164%
v_4	0.0314	0.0315	0.1407%

The comparisons contained in Table 4.24 and Table 4.25 both show good agreement between the predicted values and the values returned by the Monte Carlo simulations. There are a couple of dependent variables that are slightly under-predicted but the results are close and should be acceptable for design purposes. Again, because of the relatively large 3.0% variation in the worst-case tolerances, the resulting variation in both the deflections can result in gaps in at the nesting force features. Therefore, the variations in the independent variables are again adjusted with respect to the sensitivity matrix for the nesting force loop variables. The resulting changes to the set can be seen in Table 4.27.

Table 4.25 – Comparison of predicted worst-case values with 500,000 run Monte Carlo simulation for the Wedge assembly.

	DLM High	**MC High**	**Compare**	**DLM Low**	**MC Low**	**Compare**
u_1	2.6654	2.6099	Good	1.0616	1.1375	Good
u_2	5.2404	5.1715	Good	3.4866	3.5624	Good
u_3	1.6685	1.6460	Good	0.1901	0.2336	Good
u_4	3.0290	3.0285	Good	2.7210	2.7212	Good
u_5	1.1590	1.1590	Good	1.0910	1.0910	Good
u_6	0.0120	0.0128	Low	-0.0120	-0.0127	High
v_1	1.1804	1.0766	Good	-0.9474	-0.8664	Good
v_2	1.3268	1.3332	Low	1.1732	1.1687	High
v_3	2.9544	2.8333	Good	0.9186	1.0338	Good
v_4	0.2470	0.2420	Good	-0.0470	-0.0419	Good

The revised tolerances result in changes to the variations in the dependent variables for the Wedge assembly. For brevity, only the changes in the nesting force loop variables are given in Table 4.26. These results can now be used for the force analysis.

Table 4.26 – Revised results of the DLM nesting force loop analysis for the Wedge Assembly.

	Nominal Value		**DLM WC**	**DLM σ**
v_1	0.1165	δv_1	0.0923	0.0127
v_2	1.2500	δv_2	0.0578	0.0180
v_3	1.9365	δv_3	0.1991	0.0400
v_4	0.1000	δv_4	0.0670	0.0161

Table 4.27 – Revised tolerances and standard deviations for the Wedge Assembly.

	Nominal Value			**± δX**	**σX**	**Percent Variation**
x_1	2.500		δx_1	0.075	0.025	3.00%
x_2	0.500		δx_2	0.003	0.001	0.50%
x_3	5.000		δx_3	0.150	0.050	3.00%
x_4	0.500		δx_4	0.003	0.001	0.50%
x_5	1.750		δx_5	0.053	0.018	3.00%
x_6	2.400		δx_6	0.024	0.008	1.00%
x_7	4.000		δx_7	0.006	0.002	0.15%
x_8	3.800		δx_8	0.114	0.038	3.00%
x_9	5.620		δx_9	0.009	0.003	0.15%
x_{10}	1.000		δx_{10}	0.005	0.002	0.50%
x_{11}	5.100		δx_{11}	0.013	0.005	0.25%
x_{12}	2.000		δx_{12}	0.040	0.014	2.00%
x_{13}	1.125		δx_{13}	0.003	0.001	0.25%
x_{14}	0.349		δx_{14}	0.002	0.001	0.50%

STEP 8 – Force Analysis

With the revised results from the DLM, it can be seen that the variations in the two deflections, v_1 and v_4 are less than the nominal value; therefore there will always be a force at the desired locations. Now the force analysis can be performed using these results. The independent variables for each cantilevered beam along with their worst-case tolerances and respective standard deviations are presented in Table 4.16.

Table 4.28 - Values for the independent variables and their respective tolerances and standard deviations for the cantilevered beams in the Wedge assembly.

***X*-axis**	**Nominal Value**			**Worst Case Value**	**Standard Deviation Values**	**Percent Variation**
L_x	2.000		δL_x	0.0600	0.0200	3.0%
w_x	1.000		δw_x	0.0300	0.0100	3.0%
t_x	0.025		δt_x	0.0008	0.0003	3.0%
E_x	3E+07		δE_x	0.0000	0.0000	0.0%
***Y*-axis**						
L_y	2.500		δL_y	0.0750	0.0250	3.0%
w_y	1.000		δw_y	0.0300	0.0100	3.0%
t_y	0.030		δt_y	0.0009	0.0003	3.0%
E_y	3E+07		δE_y	0.0000	0.0000	0.0%

Calculating both the worst-case and statistical variations in the nesting forces requires the use of equations 3.16, 3.18, and 3.19. Applying the values from Table 4.26 for v_1 and v_4, and the respective x and y values from Table 4.28 to these equations results in the values given in Table 4.29. Also included in these results are estimates for the worst-case variation in the resultant force, F_r and the angle at which the resultant acts, θ_F. These values are provided to verify the resultant force is as desired. No statistical analysis was done for the resultant force or resultant angle.

Table 4.29 - Results of the DLM force analysis for the Wedge Assembly, including estimates for the resultant force, F_r and the angle of the resultant force, F_θ.

	Nominal	**Maximum**	**Minimum**	**σ**
F_x	1.7065	5.7340	0.2856	0.2035
F_y	1.2960	2.6690	0.3465	0.2167
F_r	2.1429	6.3248	0.4491	0.2084
θ_F (r)	3.791	4.606	3.202	N.A.
θ_F (°)	217.2	263.9	183.5	N.A.

Two Monte Carlo simulations were performed to validate the results. For the statistical analysis, a 100,000 run Monte Carlo simulation was performed and the results were compared against those returned by the DLM. This comparison is given in Table 4.30. For the worst-case analysis, a 500,000 run Monte Carlo was performed and the worst-case values are compared in Table 4.31. The results found in Table 4.30 and in Table 4.31 show that the DLM accurately estimates both the worst-case conditions as well as the standard deviations for all the dependent variables.

Table 4.30 - Comparison of DLM to 100,000 run Monte Carlo simulation for the standard deviation for all the dependent variables for the Wedge Assembly.

	DLM σ	**Monte Carlo** σ	**Percent Difference**
u_1	0.0274	0.0275	0.4093%
u_2	0.0512	0.0513	0.1033%
u_3	0.0112	0.0112	0.0700%
u_4	0.0022	0.0022	0.2221%
u_5	0.0010	0.0010	0.2647%
u_6	0.0006	0.0006	0.2276%
v_1	0.0127	0.0127	0.0230%
v_2	0.0180	0.0181	0.0215%
v_3	0.0400	0.0401	0.1842%
v_4	0.0161	0.0162	0.1261%
F_x	0.2035	0.2034	0.0673%
F_y	0.2167	0.2173	0.2930%

Table 4.31 - Comparison of DLM to 500,000 run Monte Carlo simulation for the worst-case conditions for all the dependent variables for the Wedge Assembly.

	DLM High	**Monte Carlo High**	**Compare**	**DLM Low**	**Monte Carlo Low**	**Compare**
u_1	2.0058	1.9957	Good	1.7212	1.7296	Good
u_2	4.5808	4.5633	Good	4.1462	4.1599	Good
u_3	0.9998	0.9929	Good	0.8589	0.8642	Good
u_4	2.8840	2.8840	Good	2.8660	2.8660	Good
u_5	1.1280	1.1280	Good	1.1220	1.1220	Good
u_6	0.0024	0.0026	Low	-0.0024	-0.0026	High
v_1	0.2088	0.1939	Good	0.0242	0.0385	Good
v_2	1.30777	1.30785	Low	1.19223	1.19216	High
v_3	2.1356	2.1144	Good	1.7374	1.7571	Good
v_4	0.1670	0.1662	Good	0.0330	0.0340	Good
F_x	5.7340	3.2791	Good	0.2856	0.5159	Good
F_y	2.6690	2.5766	Good	0.3465	0.3886	Good
Good = DLM provides a conservative estimate of the worst-case condition. **Low/High** = DLM under estimates the extent of the worst-case condition.						

Chapter 5 Conclusions and Recommendations

5.1 Contributions and Conclusions

The research presented in this work provides a method that will aid the design of exactly constrained assemblies. Specifically, this work has presented a method for the analysis and design of nesting forces. The method has specifically addressed the advantages of using active smart features to provide nesting forces. Due to the nature of the analysis, the resulting nesting forces will be robust to variations in the assembly arising from manufacturing, operational, and environmental conditions.

The method presented has illustrated the use of nesting force loops in the DLM analysis. Previous to this work, the only loops required in DLM analysis were the traditional closed and open loops. The closed loops allowed for the solution of the variation in the dependent dimensions that position the parts of the assembly with respect to each other. The open loops were used for characterizing the variation in other features of the assembly not contained in the closed loops. This method has shown that there are times when an assembly may need additional closed loops that can characterize the variation in features not otherwise located by either the traditional closed or open loops. These new closed loops have been labeled *nesting force loops*.

Four examples have been presented as a means of illustrating the process for designing robust nesting forces and for validating the method. Each example was presented in a step-by-step manner so the reader can understand the process of applying the method to assemblies. The results obtained for each of these examples have been verified by comparing both statistical and worst-case results to their respective Monte Carlo simulations. These comparisons have shown that though a linearized estimation is used, the method accurately predicts how the variation of independent variables propagates to the variation in the dependent variables. Using this information designers can now refine their designs to provide robust nesting forces where needed.

This research has also further validated the effectiveness of smart assemblies and their correlation to exact constraint design. Specifically, active smart features have been shown to be very effective in providing nesting forces in exactly constrained assemblies. A key element comes in the form of providing a nesting force while maintaining an active DoF in the direction of the force so that the advantages of the exactly constrained design are preserved.

5.2 Recommendations

This work has identified a method that allows for the design of robust nesting forces for exactly constrained assemblies. There are several areas that could be explored to further the work in exactly constrained assemblies and nesting forces. The recommendations that have arisen from this research are as follows:

- The assumption that all parts in the assembly are rigid simplified the analysis and made the development of this method possible. However, this assumption limits the applicability of the method to all assemblies. It is therefore recommended that further work explore assemblies made up of combinations of rigid and flexible parts.
- This work, as well as the preceding work in smart assemblies, is limited to two-dimensional analysis. Further work is needed to extend both methods into three dimensions for greater application in real assemblies.
- This work relied on observation and intuition for the placement of nesting forces. It is recognized that placement of nesting forces by observation is an over-simplification of a potentially complicated process. It is recommended that a mathematical method be developed for identifying the location of nesting forces with respect to the robust stability of the design.

Bibliography

Blanding, D. L., Exact Constraint: Machine Design Using Kinematic Principles, ASME Press, 1999.

Chase, K. W. "Chapter 13--Multi-Dimensional Tolerance Analysis" In *Dimensioning & Tolerancing Handbook,* McGraw-Hill, 1999.

Chase, K. W. and Parkinson, A. R., "A survey of research in the application of tolerance analysis to the design of mechanical assemblies," *Research in Engineering Design* 3(1), 23-27, 1991.

Chase, K. W.; Gao, J.; Magleby, S. P. "General 2-D tolerance analysis of mechanical assemblies with small kinematic adjustments," *Journal of Design and Manufacturing*, Volume 5, 1995, pp. 263-274.

Chase, K. W. and Law, M. J. "Chapter 1 – One-Way Clutch" In *AUTOCATS – Computer-Aided Tolerancing System Verification Manual PC Ver 1.1*, Brigham Young University, 1994.

Chen, W., J. K. Allen, K. L. Tsui, and F. Mistree, "A Procedure for Robust Design: Minimizing Variations Caused by Noise Factors and Control Factors," *J. of Mechanical Design*, 118:478-485, Dec. 1996.

Craig, M., "Using Dimensional Management," *Mechanical Engineering,* September 1995

Cvetko, R. "Characterization of Assembly Variation Analysis Methods," M.S. Thesis, Brigham Young University, Provo, Utah, 1997.

Drake, P. "Chapter 9--Traditional Approaches to Analyzing Mechanical Tolerance Stacks" In *Dimensioning & Tolerancing Handbook,* McGraw-Hill, 1999.

Downey, K. D. "A Formal Methodology for Smart Assembly Design," M.S. Thesis, Brigham Young University, Provo, Utah, 2001.

Downey, K. D.; Parkinson, A. R.; Chase, K. W. "Smart Assemblies for Robust Design: A Progress Report" *Proceeding of the ASME Design Engineering Technical Conferences*, DET2002/DAC-34135, September 29-October 2, Montreal Canada, 2002

Fog, A. "*Pseudo random number generators - uniform and non-uniform distributions*," World Wide Web Address http://www.agner.org/

Figliola, R. S.; Beasley, D. E. "*Theory and Design for Mechanical Measurements,*" John Wiley & Sons, Inc, 1995, 7th ed.

Gao, J.; Chase, K. W.; Magleby, S. P. "Comparison of Assembly Tolerance Analysis by the Direct Linearization and Modified Monte Carlo Simulation Methods," *Proceeding of the ASME Design Engineering Technical Conferences*, September 17-20, Boston MA, pp. 353-360, 1995

Gentle, J. E. "*Random Number Generation and Monte Carlo Methods,"* Springer-Verlag, 1998

Hale, L. C. "*Principles and Techniques for Designing Precision Machines,*" Ph.D. Thesis, Massachusetts Institute of Technology, 1999

Kamm, L. J., Designing Cost-Efficient Mechanisms: Minimum Constraint Design, Design with Commercial Components, and Topics in Design Engineering, McGraw-Hill, 1990, reprinted by Society of Automotive Engineers, 1993.

Kriegel, J. M., "Exact Constraint Design," In ASME International ME Congress and Exhibition (Winter Annual Meeting) Paper 94-WA/DE-18, November 1994.

Lapin, L. L., "*Modern Engineering Statistics*," Wadsworth Publishing Company, 1997

Lee, D. J., and A. C. Thornton. "Key Characteristics for Agile Product development and Manufacturing." *Agility Forum 4th Annual Conference Proceedings*, 1995

Lee, D. J., and A. C. Thornton. "The Identification and Use of Key Characteristics in the Product Development Process," *Proceedings of the 1996 ASME Design Theory and Methodology Conf.,* August 18-22, Irivine CA, 1996

Otto, K. N. and E. K. Antonsson, "Extensions to the Taguchi Method of Product Design," *Journal of Mechanical Design*, Vol. 115: 5, Mar. 1993

Parkinson, A. R. and Chase, K. W. "An Introduction to Adaptive Robust Design for Mechanical Assemblies," *Proceeding of the 2002 ASME Design Engineering Technical Conf.*, September 10-13, 2000

Parkinson, A. R., "Robust Mechanical Design Using Engineering Models," *Journal of Mechanical Design*, 117: 48-54, June 1995

Peace, G., *Taguchi Methods: A Hands On Approach*, Addison Wesley, 1993.

Randall, R. "Chapter 10—Statistical Background and Concepts" In *Dimensioning & Tolerancing Handbook,* McGraw-Hill, 1999.

Skakoon, J. G., "*Detailed Mechanical Design: A Practical Guide,"* ASME Press, New York, 2000

Taguchi, G; Chowdhury, S.; Taguchi, S. *"Robust Engineering",* Mcgraw-Hill, 1999

Taguchi, G., E. Elsaed, and T. Hsiang, *Quality Engineering in Production Systems*, McGraw-Hill, 1989.

Thornton, A. C. "Using Key Characteristics to Balance Cost and Quality During Product Development." *Proceedings of the 1997 ASME Design Theory and Methodology Conference*, Sacramento, CA, Sep 14-17, 1997

Yu, J. C. and K. Ishii, "Robust Design by Matching the Design with Manufacturing Variation Patterns," *Proceedings 20th ASME Design Automation Conf.,* DE-Vol. 69-2, p.7, Minneapolis, MN, 1994

Appendix A

Maple© Worksheets for Pinned Block Assembly

Maple© was used to form both the explicit equations and to perform the DLM on the Pinned Block Assembly. The following two sections contain the applicable Maple© worksheets that were used for this task.

A.1 Explicit Equation Development

Formation of the explicit functions for the dependent variables for the Pinned Block Assembly:

```
> restart;
```

Use the codegen package for exporting C code for the Monte Carlo simulation program.

```
> with(codegen):
Warning, the protected name MathML has been redefined and unprotected
```

Kinematic equations for Closed Loop 1

```
>
CL1[x]:=xx2*cos(0)+xx6*cos(pi/2)+uu1*cos(pi+uu2)+xx8*cos(pi/2+uu2)+xx4*
cos(3*pi/2)+xx1*cos(pi);
CL1[y]:=xx2*sin(0)+xx6*sin(pi/2)+uu1*sin(pi+uu2)+xx8*sin(pi/2+uu2)+xx4*
sin(3*pi/2)+xx1*sin(pi);
```

$$CL1_x := xx2 + xx6\ \cos\left(\frac{\pi}{2}\right) + uu1\ \cos(\pi + uu2) + xx8\ \cos\left(\frac{\pi}{2} + uu2\right) + xx4\ \cos\left(\frac{3\ \pi}{2}\right) + xx1\ \cos(\pi)$$

$$CL1_y := xx6\ \sin\left(\frac{\pi}{2}\right) + uu1\ \sin(\pi + uu2) + xx8\ \sin\left(\frac{\pi}{2} + uu2\right) + xx4\ \sin\left(\frac{3\ \pi}{2}\right) + xx1\ \sin(\pi)$$

Kinematic equations for Open Loop 1

```
> OL1[x]:=xx5*cos(-
pi/2)+xx3*cos(pi)+xx1*cos(0)+xx4*cos(pi/2)+xx8*cos(3*pi/2+uu2)+xx9*cos(
uu2);
OL1[y]:=xx5*sin(-
pi/2)+xx3*sin(pi)+xx1*sin(0)+xx4*sin(pi/2)+xx8*sin(3*pi/2+uu2)+xx9*sin(
uu2);
```

$$OL1_x := xx5\ \cos\left(\frac{\pi}{2}\right) + xx3\ \cos(\pi) + xx1 + xx4\ \cos\left(\frac{\pi}{2}\right) + xx8\ \cos\left(\frac{3\ \pi}{2} + uu2\right) + xx9\ \cos(uu2)$$

$$OL1_y := -xx5\ \sin\left(\frac{\pi}{2}\right) + xx3\ \sin(\pi) + xx4\ \sin\left(\frac{\pi}{2}\right) + xx8\ \sin\left(\frac{3\ \pi}{2} + uu2\right) + xx9\ \sin(uu2)$$

Kinematic equations for Nesting Force Loop 1:

```
>
NFL1[x]:=xx10*cos(0)+xx11*cos(pi/2)+vv2*cos(pi/2)+xx7*cos(3*pi/2+uu2)+v
v1*cos(pi+uu2)+xx8*cos(pi/2+uu2)+xx4*cos(3*pi/2)+xx1*cos(pi);
NFL1[y]:=xx10*sin(0)+xx11*sin(pi/2)+vv2*sin(pi/2)+xx7*sin(3*pi/2+uu2)+v
v1*sin(pi+uu2)+xx8*sin(pi/2+uu2)+xx4*sin(3*pi/2)+xx1*sin(pi);
```

$$NFL1_x := xx10 + xx11\ \cos\left(\frac{\pi}{2}\right) + vv2\ \cos\left(\frac{\pi}{2}\right) + xx7\ \cos\left(\frac{3\pi}{2} + uu2\right) + vv1\ \cos(\pi + uu2)$$
$$+ xx8\ \cos\left(\frac{\pi}{2} + uu2\right) + xx4\ \cos\left(\frac{3\pi}{2}\right) + xx1\ \cos(\pi)$$

$$NFL1_y := xx11\ \sin\left(\frac{\pi}{2}\right) + vv2\ \sin\left(\frac{\pi}{2}\right) + xx7\ \sin\left(\frac{3\pi}{2} + uu2\right) + vv1\ \sin(\pi + uu2) + xx8\ \sin\left(\frac{\pi}{2} + uu2\right)$$
$$+ xx4\ \sin\left(\frac{3\pi}{2}\right) + xx1\ \sin(\pi)$$

Evaluate loop equations in terms of known angles:

Also export the GAP equations for use in Monte Carlo simulation.

```
> CL1[x]:=eval(subs(pi=Pi,CL1[x]));
CL1[y]:=eval(subs(pi=Pi,CL1[y]));
GAPx:=eval(subs(pi=Pi,OL1[x]));
C(GAPx);
GAPy:=eval(subs(pi=Pi,OL1[y]));
C(GAPy);
NFL1[x]:=eval(subs(pi=Pi,NFL1[x]));
NFL1[y]:=eval(subs(pi=Pi,NFL1[y]));
```

$$CL1_x := xx2 - uu1\ \cos(uu2) - xx8\ \sin(uu2) - xx1$$

$$CL1_y := xx6 - uu1\ \sin(uu2) + xx8\ \cos(uu2) - xx4$$

$$GAPx := -xx3 + xx1 + xx8\ \sin(uu2) + xx9\ \cos(uu2)$$

```
      t0 = -xx3+xx1+xx8*sin(uu2)+xx9*cos(uu2);
```

$$GAPy := -xx5 + xx4 - xx8\ \cos(uu2) + xx9\ \sin(uu2)$$

```
      t0 = -xx5+xx4-xx8*cos(uu2)+xx9*sin(uu2);
```

$$NFL1_x := xx10 + xx7\ \sin(uu2) - vv1\ \cos(uu2) - xx8\ \sin(uu2) - xx1$$

$$NFL1_y := xx11 + vv2 - xx7\ \cos(uu2) - vv1\ \sin(uu2) + xx8\ \cos(uu2) - xx4$$

Solve equations for explicit functions for each dependent variable. Keep each expression in terms of the dependent angle as it will be solved for explicitly in terms of independent variables and can be evaluated first. Also export C code for use in Monte Carlo simulation:

```
> uu1:=solve(CL1[y]=0,uu1);C(uu1);
```

$$uu1 := \frac{xx6 + xx8\ \cos(uu2) - xx4}{\sin(uu2)}$$

```
      t0 = (xx6+xx8*cos(uu2)-xx4)/sin(uu2);
> vv1:=solve(NFL1[x]=0,vv1);C(vv1);
```

$$vv1 := -\frac{-xx10 - xx7\ \sin(uu2) + xx8\ \sin(uu2) + xx1}{\cos(uu2)}$$

```
      t0 = -(-xx10-xx7*sin(uu2)+xx8*sin(uu2)+xx1)/cos(uu2);
```

```
> vv2:=collect(solve(NFL1[y]=0,vv2),{cos(uu2),sin(uu2)});C(vv2);
```

$$vv2 := (xx7 - xx8)\cos(uu2) - xx11 + xx4 + \frac{(xx7 - xx8)\sin(uu2)^2 + (xx10 - xx1)\sin(uu2)}{\cos(uu2)}$$

```
      t0 = (xx7-xx8)*cos(uu2)-xx11+xx4+((xx7-
xx8)*pow(sin(uu2),2.0)+(xx10-xx1)*
sin(uu2))/cos(uu2);
```

```
> uu2:=solve(CL1[x]=0,uu2): uu2a:=simplify(uu2[1]); C(uu2a);
uu2b:=simplify(uu2[2]):
```

$$uu2a := \arctan\Big((-xx4\sqrt{(-xx1 + xx2)^2 (xx2^2 - 2\,xx1\,xx2 + xx4^2 + xx6^2 + xx1^2 - 2\,xx4\,xx6 - xx8^2)}$$
$$+ xx6\sqrt{(-xx1 + xx2)^2 (xx2^2 - 2\,xx1\,xx2 + xx4^2 + xx6^2 + xx1^2 - 2\,xx4\,xx6 - xx8^2)} + xx1^2\,xx8$$
$$- 2\,xx1\,xx2\,xx8 + xx2^2\,xx8) \Big/ ((xx6^2 - 2\,xx4\,xx6 + xx4^2 + xx1^2 - 2\,xx1\,xx2 + xx2^2)$$
$$(-xx1 + xx2)), \frac{-xx6\,xx8 + xx4\,xx8 + \sqrt{(-xx1 + xx2)^2 (xx2^2 - 2\,xx1\,xx2 + xx4^2 + xx6^2 + xx1^2 - 2\,xx4}}{xx6^2 - 2\,xx4\,xx6 + xx4^2 + xx1^2 - 2\,xx1\,xx2 + xx2^2}$$

```
      t0 = atan2((-xx4*sqrt(pow(-xx1+xx2,2.0)*(xx2*xx2-
2.0*xx1*xx2+xx4*xx4+xx6*
xx6+xx1*xx1-2.0*xx4*xx6-xx8*xx8))+xx6*sqrt(pow(-
xx1+xx2,2.0)*(xx2*xx2-2.0*xx1*
xx2+xx4*xx4+xx6*xx6+xx1*xx1-2.0*xx4*xx6-xx8*xx8))+xx1*xx1*xx8-
2.0*xx1*xx2*xx8+
xx2*xx2*xx8)/(xx6*xx6-2.0*xx4*xx6+xx4*xx4+xx1*xx1-
2.0*xx1*xx2+xx2*xx2)/(-xx1+
xx2),(-xx6*xx8+xx4*xx8+sqrt(pow(-xx1+xx2,2.0)*(xx2*xx2-
2.0*xx1*xx2+xx4*xx4+xx6*
xx6+xx1*xx1-2.0*xx4*xx6-xx8*xx8)))/(xx6*xx6-
2.0*xx4*xx6+xx4*xx4+xx1*xx1-2.0*xx1
*xx2+xx2*xx2));
```

Input known values for independent variables for verification of dependent variable explicit functions.

```
> xx1:=0.625:
xx2:=3.5:
xx3:=4.5:
xx4:=1.625:
xx5:=0.5:
xx6:=1.125:
xx7:=1.25:
xx8:=0.625:
xx9:=3.875:
xx10:=2.25:
xx11:=2.125:
```

Evaluate each dependent variable for verification:

```
> uu2:=evalf(uu2a);'uu1'=evalf(uu1);
```

$uu2 := 0.04365784167$

$uu1 = 2.850438554$

```
> 'GAPx'=evalf(GAPx);'GAPy'=evalf(GAPy);
```

$GAPx = 0.023585182$

$GAPy = 0.6697159330$

```
> 'vv1'=evalf(vv1);'vv2'=evalf(vv2);
```

$vv1 = 1.653853362$

$vv2 = 0.1965852006$

A.2 DLM Model Setup and Analysis

DLM Process applied to the Pinned Block example

Variable are declared in such a way as to facilitate insertion into a Microsoft Excel worksheet.

```
> restart;
```

Include the linalg package for matrix and vector functions.

```
> with(linalg):
Warning, the protected names norm and trace have been redefined and
unprotected
```

Kinematic equations for Closed Loop 1

```
>
CL1[x]:=xx2*cos(0)+xx6*cos(pi/2)+uu1*cos(pi+uu2)+xx8*cos(pi/2+uu2)+xx4*
cos(3*pi/2)+xx1*cos(pi);
CL1[y]:=xx2*sin(0)+xx6*sin(pi/2)+uu1*sin(pi+uu2)+xx8*sin(pi/2+uu2)+xx4*
sin(3*pi/2)+xx1*sin(pi);
```

$$CL1_x := xx2 + xx6\cos\left(\frac{\pi}{2}\right) + uu1\cos(\pi + uu2) + xx8\cos\left(\frac{\pi}{2} + uu2\right) + xx4\cos\left(\frac{3\pi}{2}\right) + xx1\cos(\pi)$$

$$CL1_y := xx6\sin\left(\frac{\pi}{2}\right) + uu1\sin(\pi + uu2) + xx8\sin\left(\frac{\pi}{2} + uu2\right) + xx4\sin\left(\frac{3\pi}{2}\right) + xx1\sin(\pi)$$

Kinematic equations for Open Loop 1

```
> OL1[x]:=xx5*cos(-
pi/2)+xx3*cos(pi)+xx1*cos(0)+xx4*cos(pi/2)+xx8*cos(3*pi/2+uu2)+xx9*cos(
uu2);
OL1[y]:=xx5*sin(-
pi/2)+xx3*sin(pi)+xx1*sin(0)+xx4*sin(pi/2)+xx8*sin(3*pi/2+uu2)+xx9*sin(
uu2);
```

$$OL1_x := xx5\cos\left(\frac{\pi}{2}\right) + xx3\cos(\pi) + xx1 + xx4\cos\left(\frac{\pi}{2}\right) + xx8\cos\left(\frac{3\pi}{2} + uu2\right) + xx9\cos(uu2)$$

$$OL1_y := -xx5\sin\left(\frac{\pi}{2}\right) + xx3\sin(\pi) + xx4\sin\left(\frac{\pi}{2}\right) + xx8\sin\left(\frac{3\pi}{2} + uu2\right) + xx9\sin(uu2)$$

Kinematic equations for Nesting Force Loop 1:

```
>
NFL1[x]:=xx10*cos(0)+xx11*cos(pi/2)+vv2*cos(pi/2)+xx7*cos(3*pi/2+uu2)+v
v1*cos(pi+uu2)+xx8*cos(pi/2+uu2)+xx4*cos(3*pi/2)+xx1*cos(pi);
NFL1[y]:=xx10*sin(0)+xx11*sin(pi/2)+vv2*sin(pi/2)+xx7*sin(3*pi/2+uu2)+v
v1*sin(pi+uu2)+xx8*sin(pi/2+uu2)+xx4*sin(3*pi/2)+xx1*sin(pi);
```

$$NFL1_x := xx10 + xx11\cos\left(\frac{\pi}{2}\right) + vv2\cos\left(\frac{\pi}{2}\right) + xx7\cos\left(\frac{3\pi}{2} + uu2\right) + vv1\cos(\pi + uu2) + xx8\cos\left(\frac{\pi}{2} + uu2\right) + xx4\cos\left(\frac{3\pi}{2}\right) + xx1\cos(\pi)$$

$$NFL1_y := xx11\sin\left(\frac{\pi}{2}\right) + vv2\sin\left(\frac{\pi}{2}\right) + xx7\sin\left(\frac{3\pi}{2} + uu2\right) + vv1\sin(\pi + uu2) + xx8\sin\left(\frac{\pi}{2} + uu2\right) + xx4\sin\left(\frac{3\pi}{2}\right) + xx1\sin(\pi)$$

Kinematic equations evaluated in terms of known angles:

```
> CL1[x]:=eval(subs(pi=Pi,CL1[x]));
CL1[y]:=eval(subs(pi=Pi,CL1[y]));
GAPx:=eval(subs(pi=Pi,OL1[x]));
GAPy:=eval(subs(pi=Pi,OL1[y]));
OL1[x]:=eval(subs(pi=Pi,OL1[x])):
OL1[y]:=eval(subs(pi=Pi,OL1[y])):
NFL1[x]:=eval(subs(pi=Pi,NFL1[x]));
NFL1[y]:=eval(subs(pi=Pi,NFL1[y]));
```

$$CL1_x := xx2 - uu1\cos(uu2) - xx8\sin(uu2) - xx1$$

$$CL1_y := xx6 - uu1\sin(uu2) + xx8\cos(uu2) - xx4$$

$$GAPx := -xx3 + xx1 + xx8\sin(uu2) + xx9\cos(uu2)$$

$$GAPy := -xx5 + xx4 - xx8\cos(uu2) + xx9\sin(uu2)$$

$$NFL1_x := xx10 + xx7\sin(uu2) - vv1\cos(uu2) - xx8\sin(uu2) - xx1$$

$$NFL1_y := xx11 + vv2 - xx7\cos(uu2) - vv1\sin(uu2) + xx8\cos(uu2) - xx4$$

Form vectors of independent and dependent variables.

Vectors formed for use in the grad() function.

```
> X:=vector([xx1,xx2,xx3,xx4,xx5,xx6,xx7,xx8,xx9,xx10,xx11]);
#X1:=vector([x1,x2,x3,x4,x5,x6,x7,x8,x9,x10]);
#X2:=vector([x11,x12,x13]);
dX:=vector([dx1,dx2,dx3,dx4,dx5,dx6,dx7,dx8,dx9,dx10,dx11]);
U:=vector([uu1,uu2]);
dU:=vector([du1,du2]);
GAP:=vector(['GAPx','GAPy']);
dGAP:=vector([dGAPx,dGAPy]);
V:=vector([vv1,vv2]);
dV:=vector([dv1,dv2]);
```

$$X := [xx1, xx2, xx3, xx4, xx5, xx6, xx7, xx8, xx9, xx10, xx11]$$

$$dX := [dx1, dx2, dx3, dx4, dx5, dx6, dx7, dx8, dx9, dx10, dx11]$$

$$U := [uu1, uu2]$$

$$dU := [du1, du2]$$

$$GAP := [GAPx, GAPy]$$

$$dGAP := [dGAPx, dGAPy]$$

$V := [\,vv1, vv2\,]$

$dV := [\,dv1, dv2\,]$

Matrices hx, hu, gx, gu and gv formed:

Matrices are formed by using the grad(), or gradient function.

```
> hx:=transpose(augment(
grad(eval(CL1[x]),X),
grad(eval(CL1[y]),X)));

hu:=transpose(augment(
grad(eval(CL1[x]),U),
grad(eval(CL1[y]),U)));

ogx:=transpose(augment(
grad(eval(OL1[x]),X),
grad(eval(OL1[y]),X)));

ogu:=transpose(augment(
grad(eval(OL1[x]),U),
grad(eval(OL1[y]),U)));

gx:=transpose(augment(
grad(eval(NFL1[x]),X),
grad(eval(NFL1[y]),X)));

gu:=transpose(augment(
grad(eval(NFL1[x]),U),
grad(eval(NFL1[y]),U)));

gv:=transpose(augment(
grad(eval(NFL1[x]),V),
grad(eval(NFL1[y]),V)));
```

$$hx := \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & -\sin(uu2) & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & \cos(uu2) & 0 & 0 & 0 \end{bmatrix}$$

$$hu := \begin{bmatrix} -\cos(uu2) & uu1\ \sin(uu2) - xx8\ \cos(uu2) \\ -\sin(uu2) & -uu1\ \cos(uu2) - xx8\ \sin(uu2) \end{bmatrix}$$

$$ogx := \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & \sin(uu2) & \cos(uu2) & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & -\cos(uu2) & \sin(uu2) & 0 & 0 \end{bmatrix}$$

$$ogu := \begin{bmatrix} 0 & xx8\ \cos(uu2) - xx9\ \sin(uu2) \\ 0 & xx8\ \sin(uu2) + xx9\ \cos(uu2) \end{bmatrix}$$

$$gx := \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & \sin(uu2) & -\sin(uu2) & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -\cos(uu2) & \cos(uu2) & 0 & 0 & 1 \end{bmatrix}$$

$$gu := \begin{bmatrix} 0 & xx7\ \cos(uu2) + vv1\ \sin(uu2) - xx8\ \cos(uu2) \\ 0 & xx7\ \sin(uu2) - vv1\ \cos(uu2) - xx8\ \sin(uu2) \end{bmatrix}$$

$$gv := \begin{bmatrix} -\cos(uu2) & 0 \\ -\sin(uu2) & 1 \end{bmatrix}$$

Input nominal values for all applicable variables:
This can be done by copying the appropriate cells from
the Excel file.

```
> xx1:=0.625:           dx1:=0.032:
xx2:=3.5:         dx2:=0.175:
xx3:=4.5:         dx3:=0.225:
xx4:=1.625:       dx4:=0.082:
xx5:=0.5:         dx5:=0.025:
xx6:=1.125:       dx6:=0.057:
xx7:=1.25:        dx7:=0.063:
xx8:=0.625:       dx8:=0.032:
xx9:=3.875:       dx9:=0.194:
xx10:=2.25:       dx10:=0.113:
xx11:=2.125:            dx11:=0.107:

uu2:=0.0436578416687264:
uu1:=2.85043856274785:

vv1:=1.65385336176647:
vv2:=0.196585200981373:
```

Double Check Loop Equations to verify they are equal to zero or appropriate gap size.

```
> 'CL1[x]'=evalf(CL1[x]);'CL1[y]'=evalf(CL1[y]);
GAPx:=evalf(GAPx);GAPy:=evalf(GAPy);
'NFL1[x]'=evalf(NFL1[x]);'NFL1[y]'=evalf(NFL1[y]);
```

$$CL1_x = 0.1000\ 10^{-9}$$

$$CL1_y = 0.0000$$

$$GAPx := 0.0236$$

$$GAPy := 0.6697$$

$$NFL1_x = 0.1000\ 10^{-9}$$

$$NFL1_y = -0.1000\ 10^{-8}$$

Matrices are now evaluated according to known inputs:

```
>
hx:=evalf[4](multiply(hx,Matrix(coldim(hx),coldim(hx),shape=identity)))
;
hu:=evalf[4](multiply(hu,Matrix(coldim(hu),coldim(hu),shape=identity)))
;
ogx:=evalf[4](multiply(ogx,Matrix(coldim(ogx),coldim(ogx),shape=identit
y)));
ogu:=evalf[4](multiply(ogu,Matrix(coldim(ogu),coldim(ogu),shape=identit
y)));
gx:=evalf[4](multiply(gx,Matrix(coldim(gx),coldim(gx),shape=identity)))
;
gu:=evalf[4](multiply(gu,Matrix(coldim(gu),coldim(gu),shape=identity)))
;
gv:=evalf[4](multiply(gv,Matrix(coldim(gv),coldim(gv),shape=identity)))
;
```

```
huinv:=eval(inverse(hu)):
huinv=evalf[4](evalm(huinv));
gvinv:=eval(inverse(gv)):
gvinv=evalf[4](evalm(gvinv));
```

$$hx := \begin{bmatrix} -1.0000 & , 1.0000 & , 0.0000 & , 0.0000 & , 0.0000 & , 0.0000 & , 0.0000 & , -0.0436 & , 0.0000 & , 0.0000 & , 0.0000 \\ 0.0000 & , 0.0000 & , 0.0000 & , -1.0000 & , 0.0000 & , 1.0000 & , 0.0000 & , 0.9990 & , 0.0000 & , 0.0000 & , 0.0000 \end{bmatrix}$$

$$hu := \begin{bmatrix} -0.9990 & -0.5000 \\ -0.0436 & -2.8740 \end{bmatrix}$$

$$ogx := \begin{bmatrix} 1.0000 & , 0.0000 & , -1.0000 & , 0.0000 & , 0.0000 & , 0.0000 & , 0.0000 & , 0.0436 & , 0.9990 & , 0.0000 & , 0.0000 \\ 0.0000 & , 0.0000 & , 0.0000 & , 1.0000 & , -1.0000 & , 0.0000 & , 0.0000 & , -0.9990 & , 0.0436 & , 0.0000 & , 0.0000 \end{bmatrix}$$

$$ogu := \begin{bmatrix} 0.0000 & 0.4553 \\ 0.0000 & 3.8980 \end{bmatrix}$$

$$gx := \begin{bmatrix} -1.0000 & , 0.0000 & , 0.0000 & , 0.0000 & , 0.0000 & , 0.0000 & , 0.0436 & , -0.0436 & , 0.0000 & , 1.0000 & , 0.0000 \\ 0.0000 & , 0.0000 & , 0.0000 & , -1.0000 & , 0.0000 & , 0.0000 & , -0.9990 & , 0.9990 & , 0.0000 & , 0.0000 & , 1.0000 \end{bmatrix}$$

$$gu := \begin{bmatrix} 0.0000 & 0.6966 \\ 0.0000 & -1.6240 \end{bmatrix}$$

$$gv := \begin{bmatrix} -0.9990 & 0.0000 \\ -0.0436 & 1.0000 \end{bmatrix}$$

$$huinv = \begin{bmatrix} -1.0090 & 0.1755 \\ 0.0153 & -0.3506 \end{bmatrix}$$

$$gvinv = \begin{bmatrix} -1.0010 & 0.0000 \\ -0.0437 & 1.0000 \end{bmatrix}$$

Form the closed loop sensitivities matrix:

```
> huinvhx:=evalf[4]((multiply(-huinv,hx))):
'huinvhx'=evalm(evalf[4](huinvhx));
```

$$huinvhx = \begin{bmatrix} -1.0090 & , 1.0090 & , 0.0000 & , 0.1755 & , 0.0000 & , -0.1755 & , 0.0000 & , -0.2193 & , 0.0000 & , 0.0000 & , 0.0000 \\ 0.0153 & , -0.0153 & , 0.0000 & , -0.3506 & , 0.0000 & , 0.3506 & , 0.0000 & , 0.3509 & , 0.0000 & , 0.0000 & , 0.0000 \end{bmatrix}$$

Calculate the sensitivities for the GAPs in the open loop:

```
> Sog:=evalf[4](evalm(ogx + ogu &* huinvhx));
## The + sign is due to huinvhx = -huinv*hx
```

$$Sog := \begin{bmatrix} 1.0070 & , -0.0070 & , -1.0000 & , -0.1596 & , 0.0000 & , 0.1596 & , 0.0000 & , 0.2034 & , 0.9990 & , 0.0000 & , 0.0000 \\ 0.0597 & , -0.0597 & , 0.0000 & , -0.3670 & , -1.0000 & , 1.3670 & , 0.0000 & , 0.3690 & , 0.0436 & , 0.0000 & , 0.0000 \end{bmatrix}$$

Intermediate step in calculating the nesting force loop sensitivities:

```
> Sg:=evalf[4](evalm(gx + gu &* huinvhx));
```

$$Sg := \begin{bmatrix} -0.9893 & , -0.0107 & , 0.0000 & , -0.2442 & , 0.0000 & , 0.2442 & , 0.0436 & , 0.2008 & , 0.0000 & , 1.0000 & , 0.0000 \\ -0.0249 & , 0.0249 & , 0.0000 & , -0.4306 & , 0.0000 & , -0.5694 & , -0.9990 & , 0.4291 & , 0.0000 & , 0.0000 & , 1.0000 \end{bmatrix}$$

Calculate the nesting force loop sensitivities:

```
> Sv:=evalf[4](evalm(-gvinv &* Sg));
```
$Sv :=$

[-0.9903 , -0.0107 , -0.0000 , -0.2444 , -0.0000 , 0.2444 , 0.0437 , 0.2010 , -0.0000 , 1.0010 , -0.0000]
[-0.0183 , -0.0254 , -0.0000 , 0.4199 , -0.0000 , 0.5801 , 1.0010 , -0.4203 , -0.0000 , 0.0437 , -1.0000]

Calculate the worst-case variations in all the dependent variables for the Pinned Block Assembly.

```
> ddU:=evalf[4](augment((multiply(abs(huinvhx),dX)))):
ddV:=evalf[4](augment((multiply(abs(Sv),dX)))):
ddG:=evalf[4](augment((multiply(abs(Sog),dX)))):
augment(evalm(dU))=evalm(ddU);
augment(evalm(dGAP))=evalm(ddG);
augment(evalm(dV))=evalm(ddV);
```

$$\begin{bmatrix} du1 \\ du2 \end{bmatrix} = \begin{bmatrix} 0.2403 \\ 0.0631 \end{bmatrix}$$

$$\begin{bmatrix} dGAPx \\ dGAPy \end{bmatrix} = \begin{bmatrix} 0.4809 \\ 0.1657 \end{bmatrix}$$

$$\begin{bmatrix} dv1 \\ dv2 \end{bmatrix} = \begin{bmatrix} 0.1898 \\ 0.2609 \end{bmatrix}$$

Calculate the standard deviation for all the dependent variables for the Pinned Block Assembly:

```
> sigma_u[1]:=sqrt(Sum('huinvhx[1,i]^2*(dX[i]/3)^2','i'=1..n));
sigma_u[2]:=sqrt(sum(huinvhx[2,i]^2*(dX[i]/3)^2,i=1..n)):
n:=11;for j from 1 to 2 do sigma_u[j]:=evalf[4](sigma_u[j]) od;
```

$$sigma\_u_1 := \sqrt{\sum_{i=1}^{n} \left(\frac{1}{9} huinvhx_{1,i}^{\ 2} dX_i^{\ 2} \right)}$$

$n := 11$

$sigma\_u_1 := 0.0602$

$sigma\_u_2 := 0.0123$

```
> sigma_gap[1]:=sqrt(Sum('Sog[1,i]^2*(dX[i]/3)^2','i'=1..n));
sigma_gap[2]:=sqrt(sum(Sog[2,i]^2*(dX[i]/3)^2,i=1..n)):
n:=11;for j from 1 to 2 do sigma_gap[j]:=evalf[4](sigma_gap[j]) od;
```

$$sigma\_gap_1 := \sqrt{\sum_{i=1}^{11} \left(\frac{1}{9} Sog_{1,i}^{\ 2} dX_i^{\ 2} \right)}$$

$n := 11$

$sigma\_gap_1 := 0.0997$

$sigma\_gap_2 := 0.0297$

```
> sigma_v[1]:=sqrt(Sum('Sv[1,i]^2*(dX[i]/3)^2','i'=1..n));
sigma_v[2]:=sqrt(sum(Sv[2,i]^2*(dX[i]/3)^2,i=1..n)):
```

```
n:=11;for j from 1 to 2 do sigma_v[j]:=evalf[4](sigma_v[j]) od;
```

$$sigma\_v_1 := \sqrt{\sum_{i=1}^{11} \left(\frac{1}{9} Sv_{1,i}^2\, dX_i^2\right)}$$

$$n := 11$$

$$sigma\_v_1 := 0.0401$$

$$sigma\_v_2 := 0.0446$$

Appendix B

Maple© Worksheets for Exactly Constrained Block Assembly

Maple© was used to form both the explicit equations and to perform the DLM on the Exactly Constrained Block Assembly. The following two sections contain the applicable Maple© worksheets that were used for this task.

B.1 Explicit Equation Development

Formation of the explicit functions for the dependent variables for the Exactly Constrained Block Assembly:

```
> restart;
```

Use the codegen package for exporting C code for the Monte Carlo simulation program.

```
> with(codegen,C):
```

Kinematic equations for Closed Loop 1

```
>
h1[x]:=x1*cos(0)+x7*cos(pi/2)+u1*cos(pi+u4)+x9*cos(pi/2+u4)+u3*cos(u4)+
x4*cos(pi)+x5*cos(-pi/2);
h1[y]:=x1*sin(0)+x7*sin(pi/2)+u1*sin(pi+u4)+x9*sin(pi/2+u4)+u3*sin(u4)+
x4*sin(pi)+x5*sin(-pi/2);
h1[u4]:=+90+90+u4-90-90-u4-180+90+90;
```

$$h1_x := x1 + x7\cos\left(\frac{\pi}{2}\right) + u1\cos(\pi + u4) + x9\cos\left(\frac{\pi}{2} + u4\right) + u3\cos(u4) + x4\cos(\pi) + x5\cos\left(\frac{\pi}{2}\right)$$

$$h1_y := x7\sin\left(\frac{\pi}{2}\right) + u1\sin(\pi + u4) + x9\sin\left(\frac{\pi}{2} + u4\right) + u3\sin(u4) + x4\sin(\pi) - x5\sin\left(\frac{\pi}{2}\right)$$

$$h1_{u4} := 0$$

Kinematic equations for Closed Loop 2

```
> h2[x]:=x3*cos(0)+x8*cos(pi/2)+u2*cos(-
pi/2+u4)+x10*cos(pi+u4)+x9*cos(pi/2+u4)+u3*cos(u4)+x4*cos(pi)+x5*cos(-
pi/2);
h2[y]:=x3*sin(0)+x8*sin(pi/2)+u2*sin(-
pi/2+u4)+x10*sin(pi+u4)+x9*sin(pi/2+u4)+u3*sin(u4)+x4*sin(pi)+x5*sin(-
pi/2);
h2[u4]:=+90+180+u4-90-90-90-u4-180+90+90;
```

$$h2_x := x3 + x8\,\cos\left(\frac{\pi}{2}\right) + u2\,\cos\left(\frac{\pi}{2} - u4\right) + x10\,\cos(\pi + u4) + x9\,\cos\left(\frac{\pi}{2} + u4\right) + u3\,\cos(u4)$$
$$+ x4\,\cos(\pi) + x5\,\cos\left(\frac{\pi}{2}\right)$$

$$h2_y := x8\,\sin\left(\frac{\pi}{2}\right) - u2\,\sin\left(\frac{\pi}{2} - u4\right) + x10\,\sin(\pi + u4) + x9\,\sin\left(\frac{\pi}{2} + u4\right) + u3\,\sin(u4) + x4\,\sin(\pi)$$
$$- x5\,\sin\left(\frac{\pi}{2}\right)$$

$$h2_{u4} := 0$$

Kinematic equations for Nesting Force Loop 1

```
> g1[x]:=x11*cos(pi)+x6*cos(-
pi/2)+x1*cos(0)+x7*cos(pi/2)+u1*cos(pi+u4)+v1*cos(pi/2+u4)-v3;## = v3
but rearranged.
g1[y]:=x11*sin(pi)+x6*sin(-
pi/2)+x1*sin(0)+x7*sin(pi/2)+u1*sin(pi+u4)+v1*sin(pi/2+u4);## = 0
g1[u4]:=0=0;
```

$$g1_x := x11\,\cos(\pi) + x6\,\cos\left(\frac{\pi}{2}\right) + x1 + x7\,\cos\left(\frac{\pi}{2}\right) + u1\,\cos(\pi + u4) + v1\,\cos\left(\frac{\pi}{2} + u4\right) - v3$$

$$g1_y := x11\,\sin(\pi) - x6\,\sin\left(\frac{\pi}{2}\right) + x7\,\sin\left(\frac{\pi}{2}\right) + u1\,\sin(\pi + u4) + v1\,\sin\left(\frac{\pi}{2} + u4\right)$$

$$g1_{u4} := 0 = 0$$

Kinematic equations for Nesting Force Loop 2

```
> g2[x]:=x12*cos(-
pi/2)+x2*cos(pi)+x5*cos(pi/2)+x4*cos(0)+u3*cos(pi+u4)+x9*cos(-
pi/2+u4)+v2*cos(u4);## = 0
g2[y]:=x12*sin(-
pi/2)+x2*sin(pi)+x5*sin(pi/2)+x4*sin(0)+u3*sin(pi+u4)+x9*sin(-
pi/2+u4)+v2*sin(u4)-v4;## = v4
g2[u4]:=0=0;
```

$$g2_x := x12\,\cos\left(\frac{\pi}{2}\right) + x2\,\cos(\pi) + x5\,\cos\left(\frac{\pi}{2}\right) + x4 + u3\,\cos(\pi + u4) + x9\,\cos\left(\frac{\pi}{2} - u4\right) + v2\,\cos(u4)$$

$$g2_y := -x12\,\sin\left(\frac{\pi}{2}\right) + x2\,\sin(\pi) + x5\,\sin\left(\frac{\pi}{2}\right) + u3\,\sin(\pi + u4) - x9\,\sin\left(\frac{\pi}{2} - u4\right) + v2\,\sin(u4) - v4$$

$$g2_{u4} := 0 = 0$$

Evaluate loop equations in terms of known angles:

```
> h1[x]:=eval(subs(pi=Pi,h1[x]));
h1[y]:=eval(subs(pi=Pi,h1[y]));
h2[x]:=eval(subs(pi=Pi,h2[x]));
h2[y]:=eval(subs(pi=Pi,h2[y]));
g1[x]:=eval(subs(pi=Pi,g1[x]));
g1[y]:=eval(subs(pi=Pi,g1[y]));
g2[x]:=eval(subs(pi=Pi,g2[x]));
g2[y]:=eval(subs(pi=Pi,g2[y]));
```

$$h1_x := x1 - u1\,\cos(u4) - x9\,\sin(u4) + u3\,\cos(u4) - x4$$

$$h1_y := x7 - u1\ \sin(u4) + x9\ \cos(u4) + u3\ \sin(u4) - x5$$

$$h2_x := x3 + u2\ \sin(u4) - x10\ \cos(u4) - x9\ \sin(u4) + u3\ \cos(u4) - x4$$

$$h2_y := x8 - u2\ \cos(u4) - x10\ \sin(u4) + x9\ \cos(u4) + u3\ \sin(u4) - x5$$

$$g1_x := -x11 + x1 - u1\ \cos(u4) - v1\ \sin(u4) - v3$$

$$g1_y := -x6 + x7 - u1\ \sin(u4) + v1\ \cos(u4)$$

$$g2_x := -x2 + x4 - u3\ \cos(u4) + x9\ \sin(u4) + v2\ \cos(u4)$$

$$g2_y := -x12 + x5 - u3\ \sin(u4) - x9\ \cos(u4) + v2\ \sin(u4) - v4$$

```
> u2:=collect(simplify(solve(h2[y]=0,u2)),{cos(u4),sin(u4)}):
u3:=collect(simplify(solve(h2[x]=0,u3)),{cos(u4),sin(u4)}):
u1:=collect(simplify(solve(h1[x]=0,u1)),{cos(u4),sin(u4)}):
v1:=collect(simplify(solve(g1[y]=0,v1)),{cos(u4),sin(u4)}):
v2:=collect(simplify(solve(g2[x]=0,v2)),{cos(u4),sin(u4)}):
v3:=collect(simplify(solve(g1[x]=0,v3)),{cos(u4),sin(u4)}):
v4:=collect(simplify(solve(g2[y]=0,v4)),{cos(u4),sin(u4)}):
```

Display explicit functions and export for C code for use in Monte Carlo simulation program:

```
> 'u1'=u1;
C(u1);
'u2'=u2;
C(u2);
'u3'=u3;
C(u3);
'v1'=v1;
C(v1);
'v2'=v2;
C(v2);
'v3'=v3;
C(v3);
'v4'=v4;
C(v4);
```

$$u1 = (-x3 + x4)\cos(u4) + (-x8 + x5)\sin(u4) + x10 + \frac{x1 - x9\ \sin(u4) - x4}{\cos(u4)}$$

```
      t0 = (-x3+x4)*cos(u4)+(-x8+x5)*sin(u4)+x10+(x1-x9*sin(u4)-
x4)/cos(u4);
```

$$u2 = x9 + \frac{((-x3 + x4)\cos(u4) + (-x8 + x5)\sin(u4))\sin(u4) + x8 - x5}{\cos(u4)}$$

```
      t0 = x9+(((-x3+x4)*cos(u4)+(-x8+x5)*sin(u4))*sin(u4)+x8-
x5)/cos(u4);
```

$$u3 = (-x3 + x4)\cos(u4) + (-x8 + x5)\sin(u4) + x10$$

```
      t0 = (-x3+x4)*cos(u4)+(-x8+x5)*sin(u4)+x10;
```

$$v1 = (x8 - x5)\cos(u4) + (-x3 + x4)\sin(u4) + x9 + \frac{-x7 + x6 - x8 + x5 + x10\ \sin(u4)}{\cos(u4)} + \frac{(x1 - x4)\sin(u4) - x9}{\cos(u4)^2}$$

```
        t0 = (x8-x5)*cos(u4)+(-x3+x4)*sin(u4)+x9+(-x7+x6-
x8+x5+x10*sin(u4))/cos(
u4)+((x1-x4)*sin(u4)-x9)/pow(cos(u4),2.0);
```

$$v2 = (-x3 + x4)\cos(u4) + (-x8 + x5)\sin(u4) + x10 + \frac{x2 - x4 - x9\sin(u4)}{\cos(u4)}$$

```
        t0 = (-x3+x4)*cos(u4)+(-x8+x5)*sin(u4)+x10+(x2-x4-
x9*sin(u4))/cos(u4);
```

$$v3 = -x4 - x11 + x1 + x3 + \frac{(x8 - x6 + x7 - x5)\sin(u4) - x10}{\cos(u4)} + \frac{-x1 + x9\sin(u4) + x4}{\cos(u4)^2}$$

```
        t0 = -x4-x11+x1+x3+((x8-x6+x7-x5)*sin(u4)-x10)/cos(u4)+(-
x1+x9*sin(u4)+x4
)/pow(cos(u4),2.0);
```

$$v4 = -x12 + x5 + \frac{(x2 - x4)\sin(u4) - x9}{\cos(u4)}$$

```
        t0 = -x12+x5+((x2-x4)*sin(u4)-x9)/cos(u4);
```

Form geometric solution for angle u4 in terms of independent variables and export C code:

```
> u4:=(arcsin((x5-x7)/sqrt(((x5-x7)^2+(x4-x1)^2)))-arcsin(x9/sqrt((x5-
x7)^2+(x4-x1)^2))):
'u4'=simplify(u4);
C(u4);
```

$$u4 = -\arcsin\left(\frac{-x5 + x7}{\sqrt{x5^2 - 2\,x5\,x7 + x7^2 + x1^2 - 2\,x1\,x4 + x4^2}}\right) - \arcsin\left(\frac{x9}{\sqrt{x5^2 - 2\,x5\,x7 + x7^2 + x1^2 - 2\,x1\,x4 + x4^2}}\right)$$

```
        t0 = asin((x5-x7)/sqrt(x5*x5-2.0*x5*x7+x7*x7+x1*x1-
2.0*x1*x4+x4*x4))-asin
(x9/sqrt(x5*x5-2.0*x5*x7+x7*x7+x1*x1-2.0*x1*x4+x4*x4));
```

Input known values for independent variables for verification of dependent variable explicit functions.

```
> x1:=2.625:    dx1:=.01:
x2:=9.875:    dx2:=.05:
x3:=13:       dx3:=.02:
x4:=9.75:     dx4:=.01:
x5:=9:        dx5:=.02:
x6:=5.125:    dx6:=.05:
x7:=1:        dx7:=.02:
x8:=5.25:     dx8:=.01:
x9:=8:        dx9:=.02:
x10:=12:       dx10:=.02:
x11:=1.29:    dx11:=.05:
x12:=1.29:    dx12:=.05:
```

Evaluate each dependent variable for verification:

```
> 'u1'=u1;
'u2'=u2;
'u3'=u3;
```

```
'u4'=u4;
'v1'=v1;
'v2'=v2;
'v3'=v3;
'v4'=v4;
```

$u1 = 1.6250$

$u2 = 4.2500$

$u3 = 8.7500$

$u4 = 0.0000$

$v1 = 4.1250$

$v2 = 8.8750$

$v3 = -0.2900$

$v4 = -0.2900$

B.2 DLM Model Setup and Analysis

<u>DLM Process applied to the Exactly Constrained Block example</u>

```
> restart;
```

Include the linalg package for matrix and vector functions.

```
> with(linalg):
Warning, the protected names norm and trace have been redefined and
unprotected
```

Kinematic equations for Closed Loop 1

```
>
h1[x]:=x1*cos(0)+x7*cos(pi/2)+u1*cos(pi+u4)+x9*cos(pi/2+u4)+u3*cos(u4)+
x4*cos(pi)+x5*cos(-pi/2);
h1[y]:=x1*sin(0)+x7*sin(pi/2)+u1*sin(pi+u4)+x9*sin(pi/2+u4)+u3*sin(u4)+
x4*sin(pi)+x5*sin(-pi/2);
h1[u4]:=+90+90+u4-90-90-u4-180+90+90;
```

$$h1_x := x1 + x7\cos\left(\frac{\pi}{2}\right) + u1\cos(\pi + u4) + x9\cos\left(\frac{\pi}{2} + u4\right) + u3\cos(u4) + x4\cos(\pi) + x5\cos\left(\frac{\pi}{2}\right)$$

$$h1_y := x7\sin\left(\frac{\pi}{2}\right) + u1\sin(\pi + u4) + x9\sin\left(\frac{\pi}{2} + u4\right) + u3\sin(u4) + x4\sin(\pi) - x5\sin\left(\frac{\pi}{2}\right)$$

$$h1_{u4} := 0$$

Kinematic equations for Closed Loop 2

```
> h2[x]:=x3*cos(0)+x8*cos(pi/2)+u2*cos(-
pi/2+u4)+x10*cos(pi+u4)+x9*cos(pi/2+u4)+u3*cos(u4)+x4*cos(pi)+x5*cos(-
pi/2);
```

```
h2[y]:=x3*sin(0)+x8*sin(pi/2)+u2*sin(-
pi/2+u4)+x10*sin(pi+u4)+x9*sin(pi/2+u4)+u3*sin(u4)+x4*sin(pi)+x5*sin(-
pi/2);
h2[u4]:=+90+180+u4-90-90-90-u4-180+90+90;
```

$$h2_x := x3 + x8\cos\left(\frac{\pi}{2}\right) + u2\cos\left(\frac{\pi}{2} - u4\right) + x10\cos(\pi + u4) + x9\cos\left(\frac{\pi}{2} + u4\right) + u3\cos(u4) + x4\cos(\pi) + x5\cos\left(\frac{\pi}{2}\right)$$

$$h2_y := x8\sin\left(\frac{\pi}{2}\right) - u2\sin\left(\frac{\pi}{2} - u4\right) + x10\sin(\pi + u4) + x9\sin\left(\frac{\pi}{2} + u4\right) + u3\sin(u4) + x4\sin(\pi) - x5\sin\left(\frac{\pi}{2}\right)$$

$$h2_{u4} := 0$$

Kinematic equations for Nesting Force Loop 1

```
> g1[x]:=x11*cos(pi)+x6*cos(-
pi/2)+x1*cos(0)+x7*cos(pi/2)+u1*cos(pi+u4)+v1*cos(pi/2+u4)-v3;
g1[y]:=x11*sin(pi)+x6*sin(-
pi/2)+x1*sin(0)+x7*sin(pi/2)+u1*sin(pi+u4)+v1*sin(pi/2+u4);
g1[u4]:=0=0;
```

$$g1_x := x11\cos(\pi) + x6\cos\left(\frac{\pi}{2}\right) + x1 + x7\cos\left(\frac{\pi}{2}\right) + u1\cos(\pi + u4) + v1\cos\left(\frac{\pi}{2} + u4\right) - v3$$

$$g1_y := x11\sin(\pi) - x6\sin\left(\frac{\pi}{2}\right) + x7\sin\left(\frac{\pi}{2}\right) + u1\sin(\pi + u4) + v1\sin\left(\frac{\pi}{2} + u4\right)$$

$$g1_{u4} := 0 = 0$$

Kinematic equations for Nesting Force Loop 2

```
> g2[x]:=x12*cos(-
pi/2)+x2*cos(pi)+x5*cos(pi/2)+x4*cos(0)+u3*cos(pi+u4)+x9*cos(-
pi/2+u4)+v2*cos(u4);
g2[y]:=x12*sin(-
pi/2)+x2*sin(pi)+x5*sin(pi/2)+x4*sin(0)+u3*sin(pi+u4)+x9*sin(-
pi/2+u4)+v2*sin(u4)-v4;
g2[u4]:=0=0;
```

$$g2_x := x12\cos\left(\frac{\pi}{2}\right) + x2\cos(\pi) + x5\cos\left(\frac{\pi}{2}\right) + x4 + u3\cos(\pi + u4) + x9\cos\left(\frac{\pi}{2} - u4\right) + v2\cos(u4)$$

$$g2_y := -x12\sin\left(\frac{\pi}{2}\right) + x2\sin(\pi) + x5\sin\left(\frac{\pi}{2}\right) + u3\sin(\pi + u4) - x9\sin\left(\frac{\pi}{2} - u4\right) + v2\sin(u4) - v4$$

$$g2_{u4} := 0 = 0$$

Kinematic equations evaluated in terms of known angles:

```
> h1[x]:=eval(subs(pi=Pi,h1[x]));
h1[y]:=eval(subs(pi=Pi,h1[y]));
h2[x]:=eval(subs(pi=Pi,h2[x]));
h2[y]:=eval(subs(pi=Pi,h2[y]));
g1[x]:=eval(subs(pi=Pi,g1[x]));
g1[y]:=eval(subs(pi=Pi,g1[y]));
g2[x]:=eval(subs(pi=Pi,g2[x]));
```

```
g2[y]:=eval(subs(pi=Pi,g2[y]));
```
$h1_x := x1 - u1\cos(u4) - x9\sin(u4) + u3\cos(u4) - x4$

$h1_y := x7 - u1\sin(u4) + x9\cos(u4) + u3\sin(u4) - x5$

$h2_x := x3 + u2\sin(u4) - x10\cos(u4) - x9\sin(u4) + u3\cos(u4) - x4$

$h2_y := x8 - u2\cos(u4) - x10\sin(u4) + x9\cos(u4) + u3\sin(u4) - x5$

$g1_x := -x11 + x1 - u1\cos(u4) - v1\sin(u4) - v3$

$g1_y := -x6 + x7 - u1\sin(u4) + v1\cos(u4)$

$g2_x := -x2 + x4 - u3\cos(u4) + x9\sin(u4) + v2\cos(u4)$

$g2_y := -x12 + x5 - u3\sin(u4) - x9\cos(u4) + v2\sin(u4) - v4$

Form vectors of independent and dependent variables.

Vectors formed for use in the grad() function.

```
> X:=vector([x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12]);
dX:=vector([dx1,dx2,dx3,dx4,dx5,dx6,dx7,dx8,dx9,dx10,dx11,dx12]);
U:=vector([u1,u2,u3,u4]);
dU:=vector([du1,du2,du3,du4]);
V:=vector([v1,v2,v3,v4]);
dV:=vector([dv1,dv2,dv3,dv4]);
```
$X := [x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12]$

$dX := [dx1, dx2, dx3, dx4, dx5, dx6, dx7, dx8, dx9, dx10, dx11, dx12]$

$U := [u1, u2, u3, u4]$

$dU := [du1, du2, du3, du4]$

$V := [v1, v2, v3, v4]$

$dV := [dv1, dv2, dv3, dv4]$

Matrices hx, hu, gx, gu and gv formed:

Matrices are formed by using the grad(), or gradient function.

```
> hx:=transpose(augment(
grad(eval(h1[x]),X),
grad(eval(h1[y]),X),
grad(eval(h2[x]),X),
grad(eval(h2[y]),X)));

hu:=transpose(augment(
grad(eval(h1[x]),U),
grad(eval(h1[y]),U),
grad(eval(h2[x]),U),
grad(eval(h2[y]),U)));

gx:=transpose(augment(
grad(eval(g1[x]),X),
grad(eval(g1[y]),X),
grad(eval(g2[x]),X),
grad(eval(g2[y]),X)));

gu:=transpose(augment(
grad(eval(g1[x]),U),
grad(eval(g1[y]),U),
grad(eval(g2[x]),U),
grad(eval(g2[y]),U)));

gv:=transpose(augment(
grad(eval(g1[x]),V),
grad(eval(g1[y]),V),
grad(eval(g2[x]),V),
grad(eval(g2[y]),V)));
```

$$hx := \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -\sin(u4) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & \cos(u4) & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & -\sin(u4) & -\cos(u4) & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & \cos(u4) & -\sin(u4) & 0 & 0 \end{bmatrix}$$

$$hu := \begin{bmatrix} -\cos(u4) & 0 & \cos(u4) & u1\sin(u4) - x9\cos(u4) - u3\sin(u4) \\ -\sin(u4) & 0 & \sin(u4) & -u1\cos(u4) - x9\sin(u4) + u3\cos(u4) \\ 0 & \sin(u4) & \cos(u4) & u2\cos(u4) + x10\sin(u4) - x9\cos(u4) - u3\sin(u4) \\ 0 & -\cos(u4) & \sin(u4) & u2\sin(u4) - x10\cos(u4) - x9\sin(u4) + u3\cos(u4) \end{bmatrix}$$

$$gx := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & \sin(u4) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -\cos(u4) & 0 & 0 & -1 \end{bmatrix}$$

$$gu := \begin{bmatrix} -\cos(u4) & 0 & 0 & u1\sin(u4) - v1\cos(u4) \\ -\sin(u4) & 0 & 0 & -u1\cos(u4) - v1\sin(u4) \\ 0 & 0 & -\cos(u4) & u3\sin(u4) + x9\cos(u4) - v2\sin(u4) \\ 0 & 0 & -\sin(u4) & -u3\cos(u4) + x9\sin(u4) + v2\cos(u4) \end{bmatrix}$$

$$gv := \begin{bmatrix} -\sin(u4) & 0 & -1 & 0 \\ \cos(u4) & 0 & 0 & 0 \\ 0 & \cos(u4) & 0 & 0 \\ 0 & \sin(u4) & 0 & -1 \end{bmatrix}$$

Input nominal values for all applicable variables:
This can be done by copying the appropriate cells from the Excel file.

```
> x1:=2.625:        dx1:=0.01:
x2:=10:             dx2:=0.05:
x3:=13:             dx3:=0.02:
x4:=9.75:           dx4:=0.01:
x5:=9:              dx5:=0.02:
x6:=3:              dx6:=0.05:
```

```
x7:=1:              dx7:=0.02:
x8:=5.25:           dx8:=0.01:
x9:=8:              dx9:=0.02:
x10:=12:            dx10:=0.02:
x11:=1.29:          dx11:=0.05:
x12:=1.2:           dx12:=0.05:

u1:=1.625:
u2:=4.25:
u3:=8.75:
u4:=0:
v1:=2:
v2:=9:
v3:=-0.289999999999999:
v4:=-0.2:
```

Double Check Loop Equations to verify they are equal to zero or appropriate gap size.

```
> 'h1[x]'=evalf(h1[x]);'h1[y]'=evalf(h1[y]);
'h2[x]'=evalf(h2[x]);'h2[y]'=evalf(h2[y]);
'g1[x]'=evalf(g1[x]);'g1[y]'=evalf(g1[y]);
'g2[x]'=evalf(g2[x]);'g2[y]'=evalf(g2[y]);
```

$h1_x = 0.0000$

$h1_y = 0.0000$

$h2_x = 0.0000$

$h2_y = 0.0000$

$g1_x = 0.0000$

$g1_y = 0.0000$

$g2_x = 0.0000$

$g2_y = 0.0000$

Matrices are evaluated according to known inputs:

```
> hx:=multiply(hx,Matrix(coldim(hx),coldim(hx),shape=identity));
hu:=multiply(hu,Matrix(coldim(hu),coldim(hu),shape=identity));
gx:=multiply(gx,Matrix(coldim(gx),coldim(gx),shape=identity));
gu:=multiply(gu,Matrix(coldim(gu),coldim(gu),shape=identity));
gv:=multiply(gv,Matrix(coldim(gv),coldim(gv),shape=identity));
huinv:=eval(inverse(hu)):
huinv=evalf[4](evalm(huinv));
gvinv:=eval(inverse(gv)):
gvinv=evalf[4](evalm(gvinv));
```

$$hx := \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$hu := \begin{bmatrix} -1.0000 & 0.0000 & 1.0000 & -8.0000 \\ 0.0000 & 0.0000 & 0.0000 & 7.1250 \\ 0.0000 & 0.0000 & 1.0000 & -3.7500 \\ 0.0000 & -1.0000 & 0.0000 & -3.2500 \end{bmatrix}$$

$$gx := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}$$

$$gu := \begin{bmatrix} -1.0000 & 0.0000 & 0.0000 & -2.0000 \\ 0.0000 & 0.0000 & 0.0000 & -1.6250 \\ 0.0000 & 0.0000 & -1.0000 & 8.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.2500 \end{bmatrix}$$

$$gv := \begin{bmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

$$huinv = \begin{bmatrix} -1.0000 & -0.5965 & 1.0000 & -0.0000 \\ -0.0000 & -0.4561 & 0.0000 & -1.0000 \\ -0.0000 & 0.5263 & 1.0000 & 0.0000 \\ -0.0000 & 0.1404 & 0.0000 & -0.0000 \end{bmatrix}$$

$$gvinv = \begin{bmatrix} 0.0000 & 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 & 0.0000 \\ -1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & -1.0000 \end{bmatrix}$$

Form the closed loop sensitivities matrix.

```
huinvhx:=eval((multiply(-huinv,hx))):
'huinvhx'=evalf[4](evalm(huinvhx));
```

huinvhx =

```
[ 1.0000 , 0.0000 , -1.0000 , 0.0000 , -0.5965 , 0.0000 , 0.5965 , 0.0000 , 0.5965 , 1.0000 ,
0.0000 , 0.0000 ]
[ 0.0000 , 0.0000 , 0.0000 , 0.0000 , -1.4560 , 0.0000 , 0.4561 , 1.0000 , 1.4560 , 0.0000 , 0.0000
, 0.0000 ]
[ 0.0000 , 0.0000 , -1.0000 , 1.0000 , 0.5263 , 0.0000 , -0.5263 , 0.0000 , -0.5263 , 1.0000 ,
0.0000 , 0.0000 ]
[ 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.1404 , 0.0000 , -0.1404 , 0.0000 , -0.1404 , 0.0000 ,
0.0000 , 0.0000 ]
```

Intermediate step in calculating the nesting force loop sensitivities:

```
> Sg:=evalf[4](evalm(gx + gu &* huinvhx));
## The + sign is due to huinvhx = -huinv*hx
```

```
Sg :=
      [0.0000 , 0.0000 , 1.0000 , 0.0000 , 0.3157 , 0.0000 , -0.3157 , 0.0000 , -0.3157 , -1.0000 ,
      -1.0000 , 0.0000 ]
      [0.0000 , 0.0000 , 0.0000 , 0.0000 , -0.2282 , -1.0000 , 1.2280 , 0.0000 , 0.2282 , 0.0000 ,
      0.0000 , 0.0000 ]
      [0.0000 , -1.0000 , 1.0000 , 0.0000 , 0.5967 , 0.0000 , -0.5967 , 0.0000 , -0.5967 , -1.0000 ,
      0.0000 , 0.0000 ]
      [0.0000 , 0.0000 , 0.0000 , 0.0000 , 1.0350 , 0.0000 , -0.0351 , 0.0000 , -1.0350 , 0.0000 ,
      0.0000 , -1.0000 ]
```

Calculate the nesting force loop sensitivities:

```
> sV:=evalf[4](evalm(-gvinv &* Sg));
sV :=
      [-0.0000 , -0.0000 , -0.0000 , -0.0000 , 0.2282 , 1.0000 , -1.2280 , -0.0000 , -0.2282 , -0.0000 ,
      -0.0000 , -0.0000 ]
      [-0.0000 , 1.0000 , -1.0000 , -0.0000 , -0.5967 , -0.0000 , 0.5967 , -0.0000 , 0.5967 , 1.0000 ,
      -0.0000 , -0.0000 ]
      [-0.0000 , -0.0000 , 1.0000 , -0.0000 , 0.3157 , -0.0000 , -0.3157 , -0.0000 , -0.3157 , -1.0000 ,
      -1.0000 , -0.0000 ]
      [-0.0000 , -0.0000 , -0.0000 , -0.0000 , 1.0350 , -0.0000 , -0.0351 , -0.0000 , -1.0350 , -0.0000 ,
      -0.0000 , -1.0000 ]
```

Calculate the worst-case variations in all the dependent variables for the Exactly Constrained Block Assembly:

```
> ddU:=evalf[4](augment((multiply(abs(huinvhx),dX)))):
ddV:=evalf[4](augment((multiply(abs(sV),dX)))):
augment(evalm(dU))=evalm(ddU);
augment(evalm(dV))=evalm(ddV);
```

$$\begin{bmatrix} du1 \\ du2 \\ du3 \\ du4 \end{bmatrix} = \begin{bmatrix} 0.0858 \\ 0.0774 \\ 0.0816 \\ 0.0084 \end{bmatrix}$$

$$\begin{bmatrix} dv1 \\ dv2 \\ dv3 \\ dv4 \end{bmatrix} = \begin{bmatrix} 0.0837 \\ 0.1258 \\ 0.1089 \\ 0.0921 \end{bmatrix}$$

Calculate the standard deviation for all the dependent variables for the Exactly Constrained Block Assembly:

```
> sigma_u[1]:=sqrt(Sum(huinvhx[1,i]^2*(dX[i]/3)^2,i=1..n));
sigma_u[2]:=sqrt(sum(huinvhx[2,i]^2*(dX[i]/3)^2,i=1..n)):
sigma_u[3]:=sqrt(Sum(huinvhx[3,i]^2*(dX[i]/3)^2,i=1..n)):
sigma_u[4]:=sqrt(sum(huinvhx[4,i]^2*(dX[i]/3)^2,i=1..n)):
n:=12;for j from 1 to 4 do sigma_u[j]:=evalf[4](sigma_u[j]) od;
n:='n':
```

$$sigma\_u_1 := \sqrt{\sum_{i=1}^{n} \left(\frac{1}{9} huinvhx_{1,i}^2 \, dX_i^2\right)}$$

$n := 12$

$sigma\_u_1 := 0.0121$

$sigma\_u_2 := 0.0145$

$sigma\_u_3 := 0.0117$

$sigma\_u_4 := 0.0016$

```
> sigma_v[1]:=sqrt(Sum(sV[1,i]^2*(dX[i]/3)^2,i=1..n));
sigma_v[2]:=sqrt(sum(sV[2,i]^2*(dX[i]/3)^2,i=1..n)):
sigma_v[3]:=sqrt(Sum(sV[3,i]^2*(dX[i]/3)^2,i=1..n)):
sigma_v[4]:=sqrt(sum(sV[4,i]^2*(dX[i]/3)^2,i=1..n)):
n:=12;for j from 1 to 4 do sigma_v[j]:=evalf[4](sigma_v[j]) od;
```

$$sigma\_v_1 := \sqrt{\sum_{i=1}^{n} \left(\frac{1}{9} sV_{1,i}^2 \, dX_i^2\right)}$$

$sigma\_v_1 := 0.0187$

$sigma\_v_2 := 0.0204$

$sigma\_v_3 := 0.0195$

$sigma\_v_4 := 0.0193$

Appendix C

Maple© Worksheet for Wedge Assembly

Maple© was used to form both the explicit equations and to perform the DLM on the Wedge Assembly. The following section contains the Maple© worksheet that was used for this task.

C.1 DLM Model Setup, Analysis & Explicit Equation Development

DLM Process applied to the Wedge Assembly:

```
> restart:
```

Include the linalg package for matrix and vector functions. Include codegen package for generating dependent variable functions:

```
> with(linalg):with(codegen):
Warning, the protected names norm and trace have been redefined and
unprotected

Warning, the protected name MathML has been redefined and unprotected
```

Kinematic equations for Closed Loop 1

```
> h1[x]:=x3*cos(2*pi)+x4*cos(pi/2)+u2*cos(pi+u6)+u1*cos(u6)+x2*cos(-
pi/2)+x1*cos(pi);
h1[y]:=x3*sin(2*pi)+x4*sin(pi/2)+u2*sin(pi+u6)+u1*sin(u6)+x2*sin(-
pi/2)+x1*sin(pi);
```

$$h1_x := x3\ \cos(2\ \pi) + x4\ \cos\left(\frac{\pi}{2}\right) + u2\ \cos(\pi + u6) + u1\ \cos(u6) + x2\ \cos\left(\frac{\pi}{2}\right) + x1\ \cos(\pi)$$

$$h1_y := x3\ \sin(2\ \pi) + x4\ \sin\left(\frac{\pi}{2}\right) + u2\ \sin(\pi + u6) + u1\ \sin(u6) - x2\ \sin\left(\frac{\pi}{2}\right) + x1\ \sin(\pi)$$

Kinematic equations for Closed Loop 2

```
> h2[x]:=u4*cos(pi/2)+x13*cos(2*pi)+x13*cos(-
pi/2+x14+u6)+u3*cos(pi+x14+u6)+x10*cos(-pi/2+u6)+u1*cos(u6)+x2*cos(-
pi/2)+x1*cos(pi);
h2[y]:=u4*sin(pi/2)+x13*sin(2*pi)+x13*sin(-
pi/2+x14+u6)+u3*sin(pi+x14+u6)+x10*sin(-pi/2+u6)+u1*sin(u6)+x2*sin(-
pi/2)+x1*sin(pi);
```

$$h2_x := u4\cos\left(\frac{\pi}{2}\right) + x13\cos(2\pi) + x13\cos\left(-\frac{\pi}{2} + x14 + u6\right) + u3\cos(\pi + x14 + u6)$$
$$+ x10\cos\left(\frac{\pi}{2} - u6\right) + u1\cos(u6) + x2\cos\left(\frac{\pi}{2}\right) + x1\cos(\pi)$$

$$h2_y := u4\sin\left(\frac{\pi}{2}\right) + x13\sin(2\pi) + x13\sin\left(-\frac{\pi}{2} + x14 + u6\right) + u3\sin(\pi + x14 + u6)$$
$$- x10\sin\left(\frac{\pi}{2} - u6\right) + u1\sin(u6) - x2\sin\left(\frac{\pi}{2}\right) + x1\sin(\pi)$$

Kinematic equations for Closed Loop 3

```
> h3[x]:=x7*cos(pi/2)+u5*cos(2*pi)+x13*cos(-pi/2)+x13*cos(-
pi/2+x14+u6)+u3*cos(pi+x14+u6)+x10*cos(-pi/2+u6)+u2*cos(u6)+x4*cos(-
pi/2)+x3*cos(pi);
h3[y]:=x7*sin(pi/2)+u5*sin(2*pi)+x13*sin(-pi/2)+x13*sin(-
pi/2+x14+u6)+u3*sin(pi+x14+u6)+x10*sin(-pi/2+u6)+u2*sin(u6)+x4*sin(-
pi/2)+x3*sin(pi);
```

$$h3_x := x7\cos\left(\frac{\pi}{2}\right) + u5\cos(2\pi) + x13\cos\left(\frac{\pi}{2}\right) + x13\cos\left(-\frac{\pi}{2} + x14 + u6\right) + u3\cos(\pi + x14 + u6)$$
$$+ x10\cos\left(\frac{\pi}{2} - u6\right) + u2\cos(u6) + x4\cos\left(\frac{\pi}{2}\right) + x3\cos(\pi)$$

$$h3_y := x7\sin\left(\frac{\pi}{2}\right) + u5\sin(2\pi) - x13\sin\left(\frac{\pi}{2}\right) + x13\sin\left(-\frac{\pi}{2} + x14 + u6\right) + u3\sin(\pi + x14 + u6)$$
$$- x10\sin\left(\frac{\pi}{2} - u6\right) + u2\sin(u6) - x4\sin\left(\frac{\pi}{2}\right) + x3\sin(\pi)$$

Kinematic equations for Nesting Force Loop 1

```
> g1[x]:=v2*cos(-pi/2+u6)+x11*cos(pi+u6)+u1*cos(u6)+x2*cos(-
pi/2)+x1*cos(pi)+x9*cos(2*pi)+x5*cos(pi/2)+v1*cos(2*pi);
g1[y]:=v2*sin(-pi/2+u6)+x11*sin(pi+u6)+u1*sin(u6)+x2*sin(-
pi/2)+x1*sin(pi)+x9*sin(2*pi)+x5*sin(pi/2)+v1*sin(2*pi);
```

$$g1_x := v2\cos\left(\frac{\pi}{2} - u6\right) + x11\cos(\pi + u6) + u1\cos(u6) + x2\cos\left(\frac{\pi}{2}\right) + x1\cos(\pi) + x9\cos(2\pi)$$
$$+ x5\cos\left(\frac{\pi}{2}\right) + v1\cos(2\pi)$$

$$g1_y := -v2\sin\left(\frac{\pi}{2} - u6\right) + x11\sin(\pi + u6) + u1\sin(u6) - x2\sin\left(\frac{\pi}{2}\right) + x1\sin(\pi) + x9\sin(2\pi)$$
$$+ x5\sin\left(\frac{\pi}{2}\right) + v1\sin(2\pi)$$

Kinematic equations for Nesting Force Loop 2

```
> g2[x]:=x6*cos(-
pi/2)+x8*cos(pi)+x1*cos(2*pi)+x2*cos(pi/2)+u1*cos(pi+u6)+x11*cos(u6)+x1
2*cos(pi/2+u6)+v3*cos(pi+u6)+v4*cos(pi/2);
g2[y]:=x6*sin(-
pi/2)+x8*sin(pi)+x1*sin(2*pi)+x2*sin(pi/2)+u1*sin(pi+u6)+x11*sin(u6)+x1
2*sin(pi/2+u6)+v3*sin(pi+u6)+v4*sin(pi/2);
```

$$g2_x := x6\cos\left(\frac{\pi}{2}\right) + x8\cos(\pi) + x1\cos(2\pi) + x2\cos\left(\frac{\pi}{2}\right) + u1\cos(\pi + u6) + x11\cos(u6) + x12\cos\left(\frac{\pi}{2} + u6\right) + v3\cos(\pi + u6) + v4\cos\left(\frac{\pi}{2}\right)$$

$$g2_y := -x6\sin\left(\frac{\pi}{2}\right) + x8\sin(\pi) + x1\sin(2\pi) + x2\sin\left(\frac{\pi}{2}\right) + u1\sin(\pi + u6) + x11\sin(u6) + x12\sin\left(\frac{\pi}{2} + u6\right) + v3\sin(\pi + u6) + v4\sin\left(\frac{\pi}{2}\right)$$

Kinematic equations evaluated in terms of known angles:

```
> h1[x]:=eval(subs(pi=Pi,h1[x]));
h1[y]:=eval(subs(pi=Pi,h1[y]));
h2[x]:=eval(subs(pi=Pi,h2[x]));
h2[y]:=eval(subs(pi=Pi,h2[y]));
h3[x]:=eval(subs(pi=Pi,h3[x]));
h3[y]:=eval(subs(pi=Pi,h3[y]));
g1[x]:=eval(subs(pi=Pi,g1[x]));
g1[y]:=eval(subs(pi=Pi,g1[y]));
g2[x]:=eval(subs(pi=Pi,g2[x]));
g2[y]:=eval(subs(pi=Pi,g2[y]));
```

$$h1_x := x3 - u2\cos(u6) + u1\cos(u6) - x1$$

$$h1_y := x4 - u2\sin(u6) + u1\sin(u6) - x2$$

$$h2_x := x13 + x13\sin(x14 + u6) - u3\cos(x14 + u6) + x10\sin(u6) + u1\cos(u6) - x1$$

$$h2_y := u4 - x13\cos(x14 + u6) - u3\sin(x14 + u6) - x10\cos(u6) + u1\sin(u6) - x2$$

$$h3_x := u5 + x13\sin(x14 + u6) - u3\cos(x14 + u6) + x10\sin(u6) + u2\cos(u6) - x3$$

$$h3_y := x7 - x13 - x13\cos(x14 + u6) - u3\sin(x14 + u6) - x10\cos(u6) + u2\sin(u6) - x4$$

$$g1_x := v2\sin(u6) - x11\cos(u6) + u1\cos(u6) - x1 + x9 + v1$$

$$g1_y := -v2\cos(u6) - x11\sin(u6) + u1\sin(u6) - x2 + x5$$

$$g2_x := -x8 + x1 - u1\cos(u6) + x11\cos(u6) - x12\sin(u6) - v3\cos(u6)$$

$$g2_y := -x6 + x2 - u1\sin(u6) + x11\sin(u6) + x12\cos(u6) - v3\sin(u6) + v4$$

Form vectors of independent and dependent variables.

Vectors formed for use in the grad() function.

```
> X:=vector([x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14]);
dX:=vector([dx1,dx2,dx3,dx4,dx5,dx6,dx7,dx8,dx9,dx10,dx11,dx12,dx13,dx1
4]);
U:=vector([u1,u2,u3,u4,u5,u6]);
dUU:=vector([du1,du2,du3,du4,du5,du6]);
V:=vector([v1,v2,v3,v4]);
dVV:=vector([dv1,dv2,dv3,dv4]);
```

$$X := [x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14]$$

$$dX := [dx1, dx2, dx3, dx4, dx5, dx6, dx7, dx8, dx9, dx10, dx11, dx12, dx13, dx14]$$

$U := [\, u1, u2, u3, u4, u5, u6 \,]$

$dUU := [\, du1, du2, du3, du4, du5, du6 \,]$

$V := [\, v1, v2, v3, v4 \,]$

$dVV := [\, dv1, dv2, dv3, dv4 \,]$

Matrices hx, hu, gx, gu and gv formed:
Matrices are formed by using the grad(), or gradient function.

```
> hx:=transpose(augment(
grad(eval(h1[x]),X),
grad(eval(h1[y]),X),
grad(eval(h2[x]),X),
grad(eval(h2[y]),X),
grad(eval(h3[x]),X),
grad(eval(h3[y]),X)));

hu:=transpose(augment(
grad(eval(h1[x]),U),
grad(eval(h1[y]),U),
grad(eval(h2[x]),U),
grad(eval(h2[y]),U),
grad(eval(h3[x]),U),
grad(eval(h3[y]),U)));

gx:=transpose(augment(
grad(eval(g1[x]),X),
grad(eval(g1[y]),X),
grad(eval(g2[x]),X),
grad(eval(g2[y]),X)));

gu:=transpose(augment(
grad(eval(g1[x]),U),
grad(eval(g1[y]),U),
grad(eval(g2[x]),U),
grad(eval(g2[y]),U)));

gv:=transpose(augment(
grad(eval(g1[x]),V),
grad(eval(g1[y]),V),
grad(eval(g2[x]),V),
grad(eval(g2[y]),V)));
>
```

$hx :=$

$$[-1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$
$$[0, -1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$
$$[-1, 0, 0, 0, 0, 0, 0, 0, 0, \sin(u6), 0, 0, 1+\sin(x14+u6),$$
$$x13\ \cos(x14+u6)+u3\ \sin(x14+u6)]$$
$$[0, -1, 0, 0, 0, 0, 0, 0, 0, -\cos(u6), 0, 0, -\cos(x14+u6),$$
$$x13\ \sin(x14+u6)-u3\ \cos(x14+u6)]$$
$$[0, 0, -1, 0, 0, 0, 0, 0, 0, \sin(u6), 0, 0, \sin(x14+u6),$$
$$x13\ \cos(x14+u6)+u3\ \sin(x14+u6)]$$
$$[0, 0, 0, -1, 0, 0, 1, 0, 0, -\cos(u6), 0, 0, -1-\cos(x14+u6),$$
$$x13\ \sin(x14+u6)-u3\ \cos(x14+u6)]$$

$hu :=$

$$[\cos(u6), -\cos(u6), 0, 0, 0, u2\ \sin(u6)-u1\ \sin(u6)]$$
$$[\sin(u6), -\sin(u6), 0, 0, 0, -u2\ \cos(u6)+u1\ \cos(u6)]$$
$$[\cos(u6), 0, -\cos(x14+u6), 0, 0,$$
$$x13\ \cos(x14+u6)+u3\ \sin(x14+u6)+x10\ \cos(u6)-u1\ \sin(u6)]$$
$$[\sin(u6), 0, -\sin(x14+u6), 1, 0,$$
$$x13\ \sin(x14+u6)-u3\ \cos(x14+u6)+x10\ \sin(u6)+u1\ \cos(u6)]$$
$$[0, \cos(u6), -\cos(x14+u6), 0, 1,$$
$$x13\ \cos(x14+u6)+u3\ \sin(x14+u6)+x10\ \cos(u6)-u2\ \sin(u6)]$$
$$[0, \sin(u6), -\sin(x14+u6), 0, 0,$$
$$x13\ \sin(x14+u6)-u3\ \cos(x14+u6)+x10\ \sin(u6)+u2\ \cos(u6)]$$

$$gx := \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\cos(u6) & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -\sin(u6) & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & \cos(u6) & -\sin(u6) & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & \sin(u6) & \cos(u6) & 0 & 0 \end{bmatrix}$$

$$gu := \begin{bmatrix} \cos(u6) & 0 & 0 & 0 & 0 & v2\ \cos(u6)+x11\ \sin(u6)-u1\ \sin(u6) \\ \sin(u6) & 0 & 0 & 0 & 0 & v2\ \sin(u6)-x11\ \cos(u6)+u1\ \cos(u6) \\ -\cos(u6) & 0 & 0 & 0 & 0 & u1\ \sin(u6)-x11\ \sin(u6)-x12\ \cos(u6)+v3\ \sin(u6) \\ -\sin(u6) & 0 & 0 & 0 & 0 & -u1\ \cos(u6)+x11\ \cos(u6)-x12\ \sin(u6)-v3\ \cos(u6) \end{bmatrix}$$

$$gv := \begin{bmatrix} 1 & \sin(u6) & 0 & 0 \\ 0 & -\cos(u6) & 0 & 0 \\ 0 & 0 & -\cos(u6) & 0 \\ 0 & 0 & -\sin(u6) & 1 \end{bmatrix}$$

Solve equations to obtain explicit functions for all dependent variables in terms of independent variables and generate C code for Monte Carlo simulation program:

```
> u1:=solve(h1[x]=0,u1):
u2:=solve(h3[y]=0,u2):
u3:=solve(h2[x]=0,u3):
u4:=solve(h2[y],u4):
u5:=solve(h3[x],u5):
v3:=solve(g2[x],v3):
v2:=solve(g1[y],v2):
```

```
v1:=solve(g1[x],v1):
v4:=solve(g2[y],v4):
u1:=collect(simplify(u1),{cos(x14),sin(x14)});C(u1);
u2:=collect(simplify(u2),{cos(x14),sin(x14)});C(u2);
u3:=collect(simplify(u3),{cos(x14),sin(x14)});C(u3);
u4:=collect(simplify(u4),{cos(x14),sin(x14)});C(u4);
u5:=collect(simplify(u5),{cos(x14),sin(x14)});C(u5);
v3:=collect(simplify(v3),{cos(x14),sin(x14)});C(v3);
v2:=collect(simplify(v2),{cos(x14),sin(x14)});C(v2);
v1:=collect(simplify(v1),{cos(x14),sin(x14)});C(v1);
v4:=collect(simplify(v4),{cos(x14),sin(x14)});C(v4);
```

$$u1 := (x3\cos(x14+u6)\sin(u6)+\cos(u6)\,x7\cos(x14+u6)-\cos(u6)\,x13\cos(x14+u6) - x13\cos(u6)\sin(x14+u6)-\cos(u6)\,x13-\cos(u6)\sin(x14+u6)\,x10\sin(u6) - x10\cos(u6)^2\cos(x14+u6)-\cos(u6)\,x4\cos(x14+u6)-x1\cos(x14+u6)\sin(u6) + x1\cos(u6)\sin(x14+u6))/(\cos(u6)\,(-\cos(x14+u6)\sin(u6)+\cos(u6)\sin(x14+u6)))$$

```
      t0 = (x3*cos(x14+u6)*sin(u6)+cos(u6)*x7*cos(x14+u6)-
cos(u6)*x13*cos(x14+
u6)-x13*cos(u6)*sin(x14+u6)-cos(u6)*x13-
cos(u6)*sin(x14+u6)*x10*sin(u6)-x10*pow
(cos(u6),2.0)*cos(x14+u6)-cos(u6)*x4*cos(x14+u6)-
x1*cos(x14+u6)*sin(u6)+x1*cos(
u6)*sin(x14+u6))/cos(u6)/(-
cos(x14+u6)*sin(u6)+cos(u6)*sin(x14+u6));
```

$$u2 := (x7\cos(x14+u6)-x13\cos(x14+u6)-x13\sin(x14+u6)-x13-\sin(x14+u6)\,x10\sin(u6) + \sin(x14+u6)\,x3-x10\cos(u6)\cos(x14+u6)-x4\cos(x14+u6))/(-\cos(x14+u6)\sin(u6)+\cos(u6)\sin(x14+u6))$$

```
      t0 = (x7*cos(x14+u6)-x13*cos(x14+u6)-x13*sin(x14+u6)-x13-
sin(x14+u6)*x10*
sin(u6)+sin(x14+u6)*x3-x10*cos(u6)*cos(x14+u6)-x4*cos(x14+u6))/(-
cos(x14+u6)*
sin(u6)+cos(u6)*sin(x14+u6));
```

$$u3 := (-x13\sin(u6)-x13\sin(x14+u6)\sin(u6)-x10+x3\sin(u6)+\cos(u6)\,x7-\cos(u6)\,x13 - \cos(u6)\,x13\cos(x14+u6)-\cos(u6)\,x4)/(-\cos(x14+u6)\sin(u6)+\cos(u6)\sin(x14+u6))$$

```
      t0 = (-x13*sin(u6)-x13*sin(x14+u6)*sin(u6)-
x10+x3*sin(u6)+cos(u6)*x7-cos(
u6)*x13-cos(u6)*x13*cos(x14+u6)-cos(u6)*x4)/(-
cos(x14+u6)*sin(u6)+cos(u6)*sin(
x14+u6));
```

$$u4 := \frac{\cos(u6)\,x7-\cos(u6)\,x13-\cos(u6)\,x4+x2\cos(u6)+x3\sin(u6)-\sin(u6)\,x1}{\cos(u6)}$$

```
      t0 = (cos(u6)*x7-cos(u6)*x13-
cos(u6)*x4+x2*cos(u6)+x3*sin(u6)-sin(u6)*x1)
/cos(u6);
```

$u5 := x13$

```
      t0 = x13;
```

$$v3 := -(-x8\cos(x14+u6)\sin(u6)+x8\cos(u6)\sin(x14+u6)+x3\cos(x14+u6)\sin(u6)$$
$$+\cos(u6)\,x7\cos(x14+u6)-\cos(u6)\,x13\cos(x14+u6)-x13\cos(u6)\sin(x14+u6)$$
$$-\cos(u6)\,x13-\cos(u6)\sin(x14+u6)\,x10\sin(u6)-x10\cos(u6)^2\cos(x14+u6)$$
$$-\cos(u6)\,x4\cos(x14+u6)+x11\cos(u6)\cos(x14+u6)\sin(u6)$$
$$-x11\cos(u6)^2\sin(x14+u6)-x12\cos(x14+u6)+x12\cos(x14+u6)\cos(u6)^2$$
$$+x12\sin(u6)\cos(u6)\sin(x14+u6))/(\cos(u6)$$
$$(-\cos(x14+u6)\sin(u6)+\cos(u6)\sin(x14+u6)))$$

```
       t0 = -(-
x8*cos(x14+u6)*sin(u6)+x8*cos(u6)*sin(x14+u6)+x3*cos(x14+u6)*sin(
u6)+cos(u6)*x7*cos(x14+u6)-cos(u6)*x13*cos(x14+u6)-
x13*cos(u6)*sin(x14+u6)-cos(
u6)*x13-cos(u6)*sin(x14+u6)*x10*sin(u6)-
x10*pow(cos(u6),2.0)*cos(x14+u6)-cos(u6
)*x4*cos(x14+u6)+x11*cos(u6)*cos(x14+u6)*sin(u6)-
x11*pow(cos(u6),2.0)*sin(x14+
u6)-
x12*cos(x14+u6)+x12*cos(x14+u6)*pow(cos(u6),2.0)+x12*sin(u6)*cos(u6
)*sin(
x14+u6))/cos(u6)/(-cos(x14+u6)*sin(u6)+cos(u6)*sin(x14+u6));
```

$$v2 := (x11\cos(u6)\cos(x14+u6)-x11\cos(u6)^3\cos(x14+u6)$$
$$-x11\cos(u6)^2\sin(u6)\sin(x14+u6)+x3\cos(x14+u6)-x3\cos(x14+u6)\cos(u6)^2$$
$$+\sin(u6)\cos(u6)\,x7\cos(x14+u6)-\sin(u6)\cos(u6)\,x13\cos(x14+u6)$$
$$-\cos(u6)\sin(x14+u6)\,x13\sin(u6)-\cos(u6)\,x13\sin(u6)-x10\cos(u6)\sin(x14+u6)$$
$$+x10\cos(u6)^3\sin(x14+u6)-x10\cos(u6)^2\cos(x14+u6)\sin(u6)$$
$$-\sin(u6)\cos(u6)\,x4\cos(x14+u6)-x1\cos(x14+u6)+x1\cos(x14+u6)\cos(u6)^2$$
$$+\sin(u6)\,x1\cos(u6)\sin(x14+u6)+x2\cos(u6)\cos(x14+u6)\sin(u6)$$
$$-x2\cos(u6)^2\sin(x14+u6)-x5\cos(u6)\cos(x14+u6)\sin(u6)+x5\cos(u6)^2\sin(x14+u6))$$
$$\Big/(\cos(u6)^2(-\cos(x14+u6)\sin(u6)+\cos(u6)\sin(x14+u6)))$$

```
       MapleGenVar2 = x11*cos(u6)*cos(x14+u6)-
x11*pow(cos(u6),3.0)*cos(x14+u6)-
x11*pow(cos(u6),2.0)*sin(u6)*sin(x14+u6)+x3*cos(x14+u6)-
x3*cos(x14+u6)*pow(cos(
u6),2.0)+sin(u6)*cos(u6)*x7*cos(x14+u6)-
sin(u6)*cos(u6)*x13*cos(x14+u6)-cos(u6)
*sin(x14+u6)*x13*sin(u6)-cos(u6)*x13*sin(u6)-
x10*cos(u6)*sin(x14+u6);
       MapleGenVar1 = MapleGenVar2+x10*pow(cos(u6),3.0)*sin(x14+u6)-
x10*pow(cos(
u6),2.0)*cos(x14+u6)*sin(u6)-sin(u6)*cos(u6)*x4*cos(x14+u6)-
x1*cos(x14+u6)+x1*
cos(x14+u6)*pow(cos(u6),2.0)+sin(u6)*x1*cos(u6)*sin(x14+u6)+x2*cos(
u6)*cos(x14+
u6)*sin(u6)-x2*pow(cos(u6),2.0)*sin(x14+u6)-
x5*cos(u6)*cos(x14+u6)*sin(u6)+x5*
pow(cos(u6),2.0)*sin(x14+u6);
       MapleGenVar2 = 1/pow(cos(u6),2.0)/(-
cos(x14+u6)*sin(u6)+cos(u6)*sin(x14+
u6));
```

```
       t0 = MapleGenVar1*MapleGenVar2;
```

$$v1 := -\,(-x2\,\cos(u6)^3\,\cos(x14+u6) - x13\,\cos(u6)\,\sin(x14+u6) + x11\,\cos(u6)\,\cos(x14+u6)\,\sin(u6) - \cos(u6)\,\sin(x14+u6)\,x10\,\sin(u6) - x10\,\cos(u6)^2\,\cos(x14+u6) - x11\,\cos(u6)^2\,\sin(x14+u6) - \cos(u6)\,x13 - x1\,\cos(u6)^3\,\sin(x14+u6) + \sin(u6)\,x5\,\cos(u6)^2\,\sin(x14+u6) - \sin(u6)\,x2\,\cos(u6)^2\,\sin(x14+u6) - x9\,\cos(u6)^2\,\cos(x14+u6)\,\sin(u6) + x9\,\cos(u6)^3\,\sin(x14+u6) - \cos(u6)\,x13\,\cos(x14+u6) - \cos(u6)\,x4\,\cos(x14+u6) + x3\,\cos(x14+u6)\,\sin(u6) + \cos(u6)\,x7\,\cos(x14+u6) + x5\,\cos(u6)^3\,\cos(x14+u6) - x1\,\cos(x14+u6)\,\sin(u6) + x1\,\cos(x14+u6)\,\sin(u6)\,\cos(u6)^2 - x5\,\cos(u6)\,\cos(x14+u6) + x1\,\cos(u6)\,\sin(x14+u6) + x2\,\cos(u6)\,\cos(x14+u6))\,\big/\,(\cos(u6)^2\,(-\cos(x14+u6)\,\sin(u6) + \cos(u6)\,\sin(x14+u6)))$$

```
       MapleGenVar2 =
x2*pow(cos(u6),3.0)*cos(x14+u6)+x13*cos(u6)*sin(x14+u6)-
x11*cos(u6)*cos(x14+u6)*sin(u6)+cos(u6)*sin(x14+u6)*x10*sin(u6)+x10
*pow(cos(u6)
,2.0)*cos(x14+u6)+x11*pow(cos(u6),2.0)*sin(x14+u6)+cos(u6)*x13+x1*p
ow(cos(u6),
3.0)*sin(x14+u6)-
sin(u6)*x5*pow(cos(u6),2.0)*sin(x14+u6)+sin(u6)*x2*pow(cos(u6)
,2.0)*sin(x14+u6)+x9*pow(cos(u6),2.0)*cos(x14+u6)*sin(u6);
       MapleGenVar1 = MapleGenVar2-
x9*pow(cos(u6),3.0)*sin(x14+u6)+cos(u6)*x13*
cos(x14+u6)+cos(u6)*x4*cos(x14+u6)-x3*cos(x14+u6)*sin(u6)-
cos(u6)*x7*cos(x14+u6
)-x5*pow(cos(u6),3.0)*cos(x14+u6)+x1*cos(x14+u6)*sin(u6)-
x1*cos(x14+u6)*sin(u6)
*pow(cos(u6),2.0)+x5*cos(u6)*cos(x14+u6)-x1*cos(u6)*sin(x14+u6)-
x2*cos(u6)*cos(
x14+u6);
       MapleGenVar2 = 1/pow(cos(u6),2.0)/(-
cos(x14+u6)*sin(u6)+cos(u6)*sin(x14+
u6));
       t0 = MapleGenVar1*MapleGenVar2;
```

$$v4 := \frac{x6\,\cos(u6) - x2\,\cos(u6) + \sin(u6)\,x1 - x12 - \sin(u6)\,x8}{\cos(u6)}$$

```
       t0 = (x6*cos(u6)-x2*cos(u6)+sin(u6)*x1-x12-
sin(u6)*x8)/cos(u6);
```

> u6:=arctan((x4-x2),(x3-x1));C(u6);

$u6 := \arctan(x4 - x2,\, x3 - x1)$

```
       t0 = atan2(x4-x2,x3-x1);
```

Input nominal values for all applicable variables:
This can be done by copying the appropriate cells from the Excel file.

```
> x1:=2.5:          dx1:=0.01:
x2:=0.5:            dx2:=0.005:
x3:=5:              dx3:=0.02:
x4:=0.5:            dx4:=0.005:
```

```
x5:=1.75:          dx5:=0.02:
x6:=2.4:           dx6:=0.02:
x7:=4:             dx7:=0.005:
x8:=3.8:           dx8:=0.01:
x9:=5.62:          dx9:=0.01:
x10:=1:            dx10:=0.005:
x11:=5.1:          dx11:=0.01:
x12:=2:            dx12:=0.01:
x13:=1.125:        dx13:=0.001:
x14:=0.349065850398866:        dx14:=0.00872664625997165:
```

Evaluate all dependent variables for known independent variable values:

```
> 'u1'=evalf[4](u1);
'u2'=evalf[4](u2);
'u3'=evalf[4](u3);
'u4'=evalf[4](u4);
'u5'=evalf[4](u5);
'u6'=evalf[4](u6);
'v1'=evalf[4](v1);
'v2'=evalf[4](v2);
'v3'=evalf[4](v3);
'v4'=evalf[4](v4);
```

$u1 = 1.8640$

$u2 = 4.3610$

$u3 = 0.9296$

$u4 = 2.8750$

$u5 = 1.1250$

$u6 = 0.0000$

$v1 = 0.1140$

$v2 = 1.2500$

$v3 = 1.9380$

$v4 = -0.1000$

Double Check Loop Equations to verify they are equal to zero:

```
> 'h1[x]'=evalf(h1[x]);'h1[y]'=evalf(h1[y]);
'h2[x]'=evalf(h2[x]);'h2[y]'=evalf(h2[y]);
'h3[x]'=evalf(h3[x]);'h3[y]'=evalf(h3[y]);
'g1[x]'=evalf(g1[x]);'g1[y]'=evalf(g1[y]);
'g2[x]'=evalf(g2[x]);'g2[y]'=evalf(g2[y]);
```

$h1_x = 0.1000\ 10^{-8}$

$h1_y = 0.0000$

$h2_x = 0.0000$

$h2_y = 0.0000$

$h3_x = -0.1000\ 10^{-8}$

$h3_y = 0.0000$

$g1_x = -0.7000\ 10^{-9}$

$g1_y = 0.0000$

$g2_x = 0.1000\ 10^{-8}$

$g2_y = 0.0000$

Matrices are now evaluated according to know inputs:

```
> hx:=multiply(hx,Matrix(coldim(hx),coldim(hx),shape=identity));
hu:=multiply(hu,Matrix(coldim(hu),coldim(hu),shape=identity));
gx:=multiply(gx,Matrix(coldim(gx),coldim(gx),shape=identity));
gu:=multiply(gu,Matrix(coldim(gu),coldim(gu),shape=identity));
gv:=multiply(gv,Matrix(coldim(gv),coldim(gv),shape=identity));
huinv:=eval(inverse(hu)):
huinv=evalf[4](evalm(huinv));
gvinv:=eval(inverse(gv)):
gvinv=evalf[4](evalm(gvinv));
```

hx :=

```
[-1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0]
[0 , -1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0]
[-1.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000
, 0.0000 , 1.3420 , 1.3750 ]
[0.0000 , -1.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , -1.0000 ,
0.0000 , 0.0000 , -0.9397 , -0.4885 ]
[0.0000 , 0.0000 , -1.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000
, 0.0000 , 0.3420 , 1.3750 ]
[0.0000 , 0.0000 , 0.0000 , -1.0000 , 0.0000 , 0.0000 , 1.0000 , 0.0000 , 0.0000 , -1.0000 ,
0.0000 , 0.0000 , -1.9397 , -0.4885 ]
```

$$hu := \begin{bmatrix} 1.0000 & -1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & -2.5000 \\ 1.0000 & 0.0000 & -0.9397 & 0.0000 & 0.0000 & 2.3750 \\ 0.0000 & 0.0000 & -0.3420 & 1.0000 & 0.0000 & 1.3750 \\ 0.0000 & 1.0000 & -0.9397 & 0.0000 & 1.0000 & 2.3750 \\ 0.0000 & 0.0000 & -0.3420 & 0.0000 & 0.0000 & 3.8750 \end{bmatrix}$$

gx :=

[-1.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 1.0000 , 0.0000 , -1.0000 , 0.0000 , 0.0000 , 0.0000]

[0.0000 , -1.0000 , 0.0000 , 0.0000 , 1.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000]

[1.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , -1.0000 , 0.0000 , 0.0000 , 1.0000 , 0.0000 , 0.0000 , 0.0000]

[0.0000 , 1.0000 , 0.0000 , 0.0000 , 0.0000 , -1.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 0.0000 , 1.0000 , 0.0000 , 0.0000]

$$gu := \begin{bmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.2500 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & -3.2365 \\ -1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & -2.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.3000 \end{bmatrix}$$

$$gv := \begin{bmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & -1.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

$$huinv = \begin{bmatrix} 0.0000 & -3.3090 & 1.0000 & 0.0000 & 0.0000 & -2.7470 \\ -1.0000 & -3.3090 & 1.0000 & 0.0000 & 0.0000 & -2.7470 \\ 0.0000 & -4.5320 & 0.0000 & 0.0000 & 0.0000 & -2.9240 \\ 0.0000 & -1.0000 & 0.0000 & 1.0000 & 0.0000 & -1.0000 \\ 1.0000 & 0.0000 & -1.0000 & 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & -0.4000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$$

$$gvinv = \begin{bmatrix} 1.0000 & 0.0000 & 0.0000 & -0.0000 \\ 0.0000 & -1.0000 & -0.0000 & 0.0000 \\ 0.0000 & -0.0000 & -1.0000 & -0.0000 \\ 0.0000 & 0.0000 & -0.0000 & 1.0000 \end{bmatrix}$$

Form the closed loop sensitivities matrix:

```
> huinvhx:=eval((multiply(-huinv,hx))):
'huinvhx'=evalf[4](evalm(huinvhx));
```

```
huinvhx =
      [ 1.0000  , -3.3090  , 0.0000  , 0.5611  , 0.0000  , 0.0000  , 2.7470  , 0.0000  , 0.0000  , -2.7470  ,
      0.0000  , 0.0000  , -6.6710  , -2.7170  ]
      [ 0.0000  , -3.3090  , 1.0000  , 0.5611  , 0.0000  , 0.0000  , 2.7470  , 0.0000  , 0.0000  , -2.7470  ,
      0.0000  , 0.0000  , -6.6710  , -2.7170  ]
      [ 0.0000  , -4.5320  , 0.0000  , 1.6080  , 0.0000  , 0.0000  , 2.9240  , 0.0000  , 0.0000  , -2.9240  ,
      0.0000  , 0.0000  , -5.6710  , -1.4280  ]
      [ 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 1.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000
      , 0.0000  , -1.0000  , 0.0000  ]
      [ 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000
      , 0.0000  , 1.0000  , 0.0000  ]
      [ 0.0000  , -0.4000  , 0.0000  , 0.4000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000
      , 0.0000  , 0.0000  , 0.0000  ]
```

```
> eta:=evalf[4](evalm(gx + gu &* huinvhx));
## The + sign is due to huinvhx = -huinv*hx
η :=
      [ 0.0000  , -3.8090  , 0.0000  , 1.0610  , 0.0000  , 0.0000  , 2.7470  , 0.0000  , 1.0000  , -2.7470  ,
      -1.0000  , 0.0000  , -6.6710  , -2.7170  ]
      [ 0.0000  , 0.2940  , 0.0000  , -1.2940  , 1.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000
      , 0.0000  , 0.0000  , 0.0000  ]
      [ 0.0000  , 4.1090  , 0.0000  , -1.3610  , 0.0000  , 0.0000  , -2.7470  , -1.0000  , 0.0000  , 2.7470  ,
      1.0000  , 0.0000  , 6.6710  , 2.7170  ]
      [ 0.0000  , 0.4800  , 0.0000  , 0.5200  , 0.0000  , -1.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000  , 0.0000
      , 1.0000  , 0.0000  , 0.0000  ]
```

```
> sV:=evalf[4](evalm(-gvinv &* eta));
sV :=
      [ -0.0000  , 3.8090  , -0.0000  , -1.0610  , -0.0000  , -0.0000  , -2.7470  , -0.0000  , -1.0000  , 2.7470  ,
      1.0000  , -0.0000  , 6.6710  , 2.7170  ]
      [ -0.0000  , 0.2940  , -0.0000  , -1.2940  , 1.0000  , -0.0000  , -0.0000  , -0.0000  , -0.0000  , 0.0000  ,
      0.0000  , -0.0000  , 0.0000  , 0.0000  ]
      [ -0.0000  , 4.1090  , -0.0000  , -1.3610  , -0.0000  , -0.0000  , -2.7470  , -1.0000  , -0.0000  , 2.7470  ,
      1.0000  , -0.0000  , 6.6710  , 2.7170  ]
      [ -0.0000  , -0.4800  , -0.0000  , -0.5200  , -0.0000  , 1.0000  , -0.0000  , -0.0000  , -0.0000  , -0.0000  ,
      -0.0000  , -1.0000  , -0.0000  , -0.0000  ]
```

```
> dU:=evalf[4](augment((multiply(abs(huinvhx),dX)))):
augment(evalm(dUU))=evalm(dU);
dV:=evalf[4](augment((multiply(abs(sV),dX)))):
augment(evalm(dVV))=evalm(dV);
```

$$\begin{bmatrix} du1 \\ du2 \\ du3 \\ du4 \\ du5 \\ du6 \end{bmatrix} = \begin{bmatrix} 0.0872 \\ 0.0972 \\ 0.0781 \\ 0.0060 \\ 0.0010 \\ 0.0040 \end{bmatrix}$$

$$\begin{bmatrix} dv1 \\ dv2 \\ dv3 \\ dv4 \end{bmatrix} = \begin{bmatrix} 0.1022 \\ 0.0279 \\ 0.1052 \\ 0.0350 \end{bmatrix}$$

Calculate the standard deviation for all the dependent variables for the Pinned Block Assembly:

```
> sigma_u[1]:=sqrt(Sum(huinvhx[1,i]^2*(dX[i]/3)^2,i=1..n));
sigma_u[2]:=sqrt(sum(huinvhx[2,i]^2*(dX[i]/3)^2,i=1..n)):
sigma_u[3]:=sqrt(Sum(huinvhx[3,i]^2*(dX[i]/3)^2,i=1..n)):
sigma_u[4]:=sqrt(sum(huinvhx[4,i]^2*(dX[i]/3)^2,i=1..n)):
sigma_u[5]:=sqrt(Sum(huinvhx[5,i]^2*(dX[i]/3)^2,i=1..n)):
sigma_u[6]:=sqrt(sum(huinvhx[6,i]^2*(dX[i]/3)^2,i=1..n)):
n:=14;for j from 1 to 6 do sigma_u[j]:=evalf[4](sigma_u[j]) od;
n:='n':
```

$$sigma\_u_1 := \sqrt{\sum_{i=1}^{n} \left(\frac{1}{9} huinvhx_{1,i}^{\ 2} dX_i^{\ 2}\right)}$$

$n := 14$

$sigma\_u_1 := 0.0123$

$sigma\_u_2 := 0.0136$

$sigma\_u_3 := 0.0115$

$sigma\_u_4 := 0.0017$

$sigma\_u_5 := 0.0003$

$sigma\_u_6 := 0.0009$

```
> sigma_v[1]:=sqrt(Sum(sV[1,i]^2*(dX[i]/3)^2,i=1..n));
sigma_v[2]:=sqrt(sum(sV[2,i]^2*(dX[i]/3)^2,i=1..n)):
sigma_v[3]:=sqrt(Sum(sV[3,i]^2*(dX[i]/3)^2,i=1..n)):
sigma_v[4]:=sqrt(sum(sV[4,i]^2*(dX[i]/3)^2,i=1..n)):
n:=14;for j from 1 to 4 do sigma_v[j]:=evalf[4](sigma_v[j]) od;
```

$$sigma\_v_1 := \sqrt{\sum_{i=1}^{n} \left(\frac{1}{9} sV_{1,i}^{\ 2} dX_i^{\ 2}\right)}$$

$n := 14$

$sigma\_v_1 := 0.0132$

$sigma\_v_2 := 0.0070$

$sigma\_v_{3} := 0.0136$

$sigma\_v_{4} := 0.0075$

Appendix D
Using Monte Carlo for Worst-Case Comparisons

This appendix will reference the Pinned Block assembly explained in both sections 2.6.1 and 3.4.

A uniform distribution will allow independent variables to have an equal probability of being at their maximum or minimum values. Also, a uniform distribution will never exceed the set limits for a variable. A histogram for the independent variable, x_1 can be seen in Figure D.1. As can be seen in the histogram, x_1 is at its limits just as frequently as it is any other value. Also, Figure D.1 shows that x_1 never exceeds its limits.

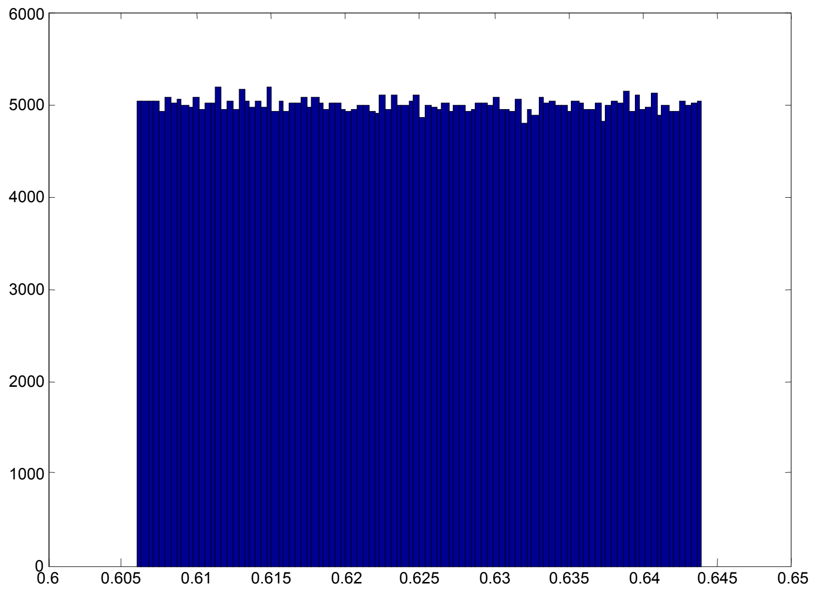


Figure D.1 – Histogram of the uniformly distributed independent variable, x_1.

In contrast, a normal distribution is based on a standard deviation and has some probability that it will exceed the dimensional limits for the variable. This indicates that, though the probability is small, there is a chance that if enough independent variables exceed their chosen tolerance limits because of their normal distributions, the dependent variables will continuously be pushed to greater and greater limits. A histogram or the independent variable, x_1 is illustrated in Figure D.2.

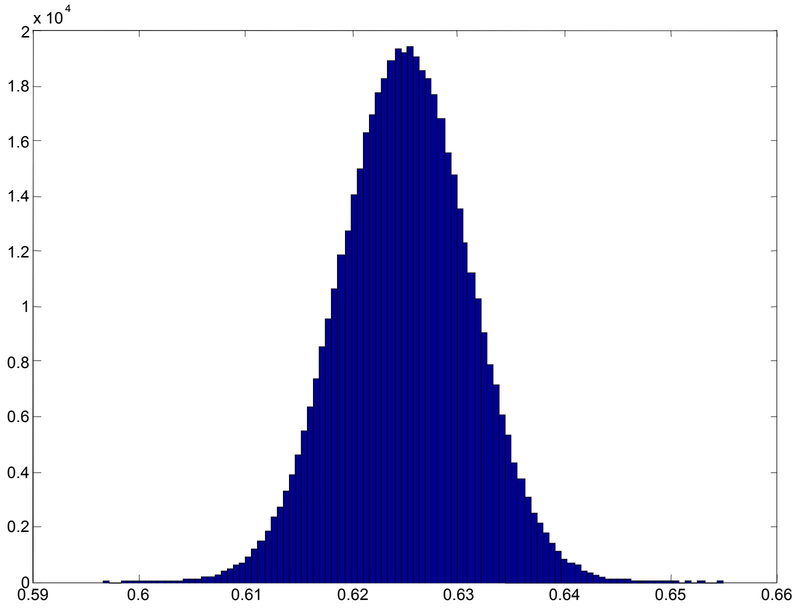


Figure D.2 – Histogram of the normally distributed independent variable, x_1.

When the dependent variables are considered, allowing all the independent variables to vary according to a normal distribution cannot be used for worst-case analysis. This is best illustrated with a comparison between the same dependent variable, in this case, u1, and two Monte Carlo distributions, one with independent variables that were allowed to follow a normal distribution, and one where the independent variables were allowed to follow uniform distributions. Both these simulations were run with 500,000 runs. Figure D.3 was generated from data acquired from the normally distributed inputs using the *histfit(variable,numbins)* function associated with the statistical package in MatLab©. It plots a normal curve over a similarly scaled histogram of the data found in *variable* and with the number of histogram bins, *numbins*. It can be seen that there are values that extend further into the tail. Again, because independent variables are described by a normal distribution with tails that extend to ±∞, the dependent variable will vary likewise. Zooming in on right-hand tail shows how these random values begin to populate the tail. This is illustrated in Figure D.4.

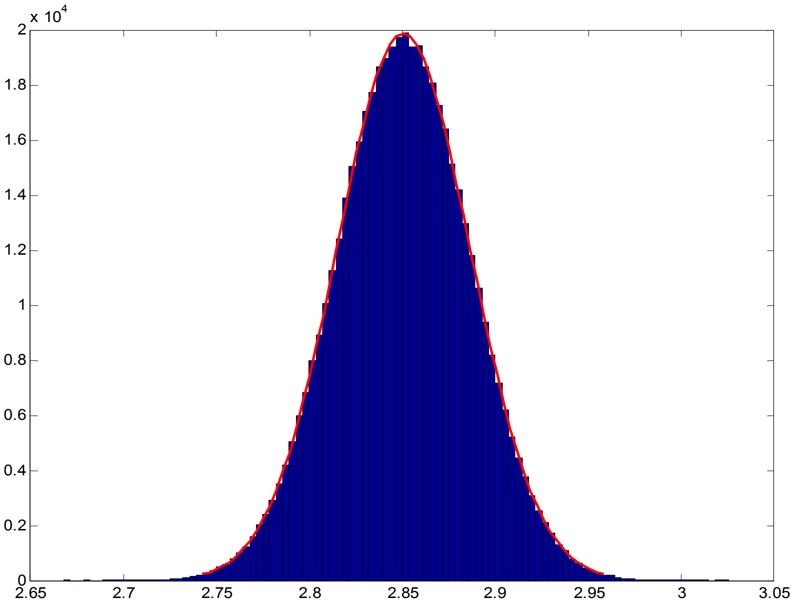


Figure D.3 – *Histfit()* plot of the data for the dependent variable, u_1.

In contrast, Figure D.5 shows a histogram of data generated for u_1 using a Monte Carlo simulation where the independent variables had uniform distributions. Notice how both end of the histogram fall off rapidly and there are little to no tails as there were with the normally distributed independent variables. Again, zooming in on the right-hand limits of the histogram, as shown in Figure D.6, further shows fact that there is a finite limit to the extents of values for u_1.

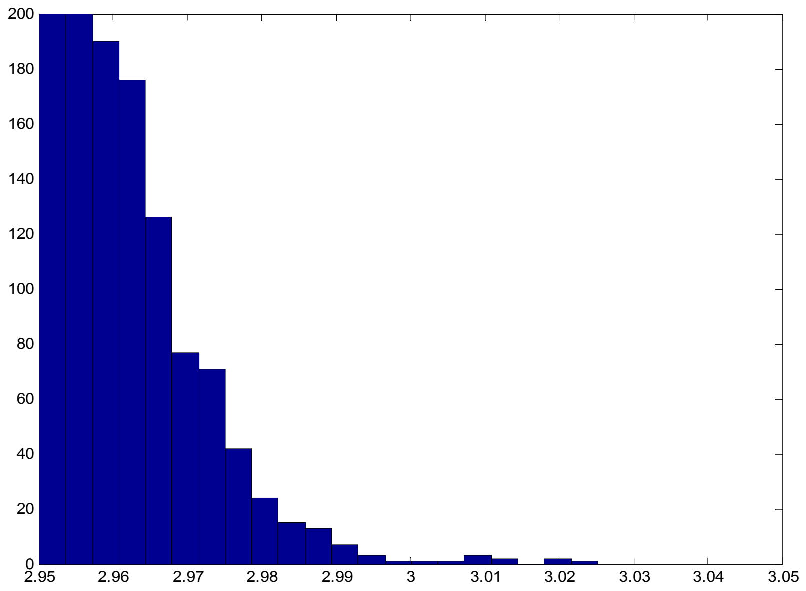


Figure D.4 – Zoomed in on right-hand tail of the *Histfit()* plot for the dependent variable, u_1.

In conclusion, it was determined from the knowledge that a uniform distribution has finite limits that are the same as the manufacturing limits, and by analyzing histograms for the dependent variables, as shown in Figure D.3 thru Figure D.6 that a 500,000 run Monte Carlo simulation where the independent variables were uniformly distributed would be sufficient for estimating the worst-case conditions on dependent variables.

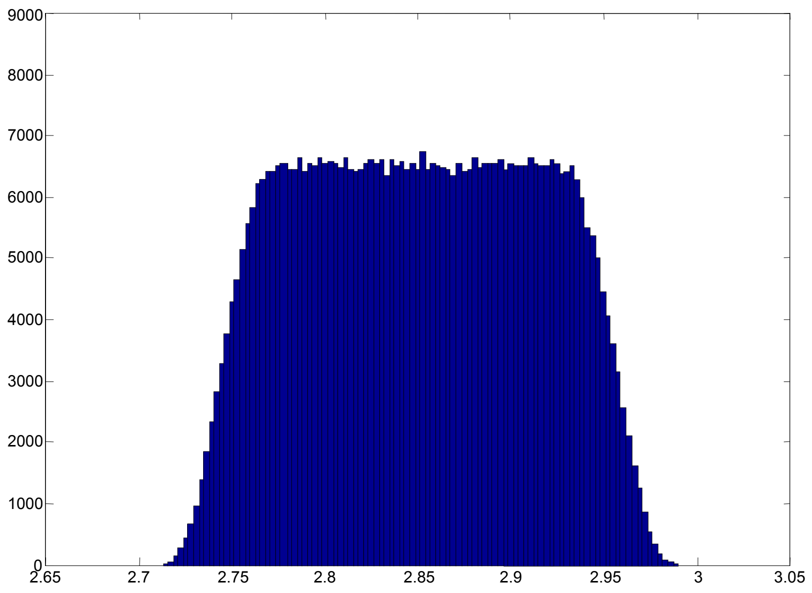


Figure D.5 – Histogram of the data for the dependent variable, u_1 when the independent variables were allowed to follow a uniform distribution.

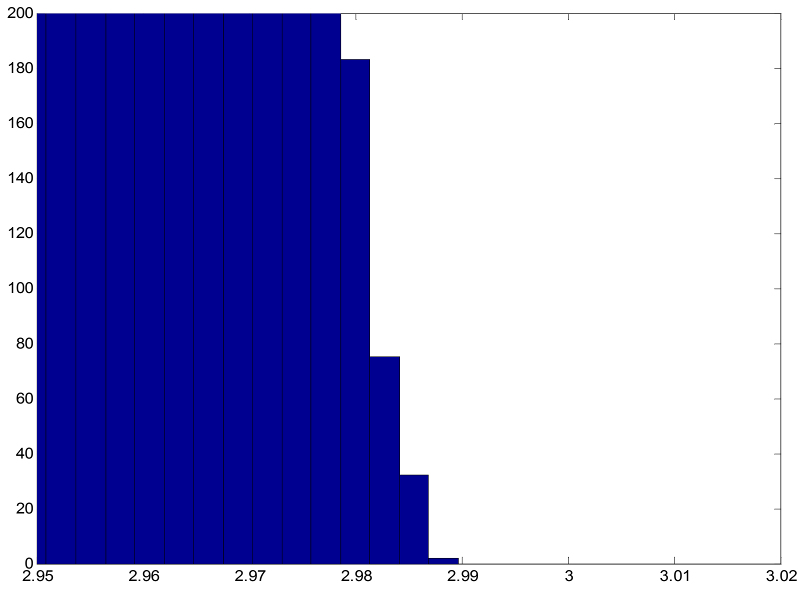


Figure D.6 – Histogram of the data for the dependent variable, u_1 when the independent variables were allowed to follow a uniform distribution.