COMPUTER-AIDED TOLERANCE ANALYSIS OF BOUNDARY GEOMETRY REPRESENTATIONS OF ASSEMBLIES

by

Nathan Graham

Submitted to the Department of
Mechanical Engineering
In Partial Fulfillment of the
Requirement of the
Degree of

BACHELOR OF SCIENCE IN MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
(June, 1982)

Signature of Author: ______________________________
Department of Mechanical Engineering
June, 1982

Certified by: ______________________________
Thesis Supervisor

Accepted by: [illegible]
Department Thesis Committee

COMPUTER-AIDED TOLERANCE ANALYSIS OF BOUNDARY GEOMETRY REPRESENTATIONS OF ASSEMBLIES

by

Nathan Graham

Submitted to the Department of
Mechanical Engineering
In Partial Fulfillment of the
Requirement of the
Degree of

ABSTRACT

A method of tolerance analysis was developed. Its purpose is to determine the percentage of individually produced components which can be assembled given a distribution of tolerances. The method was implemented on a Digital Equipment Corporation VAX 11/750 digital computer using the VMS operating system in the form of an interactive FORTRAN program. The program extracts information on dimensions from a boundary geometry data representation and its output is conceptual model known as a tolerance chain.

Thesis Supervisor: David C. Gossard

Title: Associate Professor of Mechanical Engineering

ACKNOWLEDGEMENTS

I would like thank Professor David Gossard for all the help and guidance he gave me while I worked on my thesis. Also for their assistance they gave whenever I needed it, I would like to thank everyone in the MIT CADLAB: Darwin Chang, Scott Ewing, Kunwoo Lee, Phil Meyfarth, Nakajima-san, and Sakurai-san. Finally I thank the Control Data Corporation for supplying the computer and the computer time for this project.

TABLE OF CONTENTS

LIST OF FIGURES

1.Ø INTRODUCTION

A designer of an assembly chooses particular dimensions to satisfy certain engineering constraints. It is impossible for the components of the assembly to be manufactured to the exact dimensions specified, so the designer specifies an allowable range, or tolerance for each dimension. Tolerances are chosen in such a way that components fit together tightly, but may be assembled and disassembled easily. Tolerances are chosen so that the dimensions are accurate enough for the part to function but not so accurate as to entail high manufacturing costs unnecessarily. Due to the statistical nature of manufacturing processes, most but not all of the assemblies will have dimensions that lie within the range of the tolerances. As a result, some of the assemblies may not function, or it may not be possible to assemble the components.

The term "confidence level" refers to the percentage of assemblies that have a specified critical dimension which falls within a specified range. The goal of this study is to estimate the confidence level, based on statistical models of the machining processes required to produce each dimension. This method will be implemented as a computer program and tested on a case study using a three dimensional model of the assembly.

This procedure for tolerance analysis that is based on a method developed by Bjorke (1). of the University of Trondheim, the Norwegian Institute of Technology and the Production Engineering Laboratory at the Foundation of Scientific and Industrial Research, Trondheim, Norway.

Bjorke defines tolerance control as: "determining the tolerance of a functional dimension as a function of the confidence level based on given data concerning individual dimensions in the assembly." The method developed in this report is concerned with finding the confidence level from a complete given set of dimensions and tolerances.

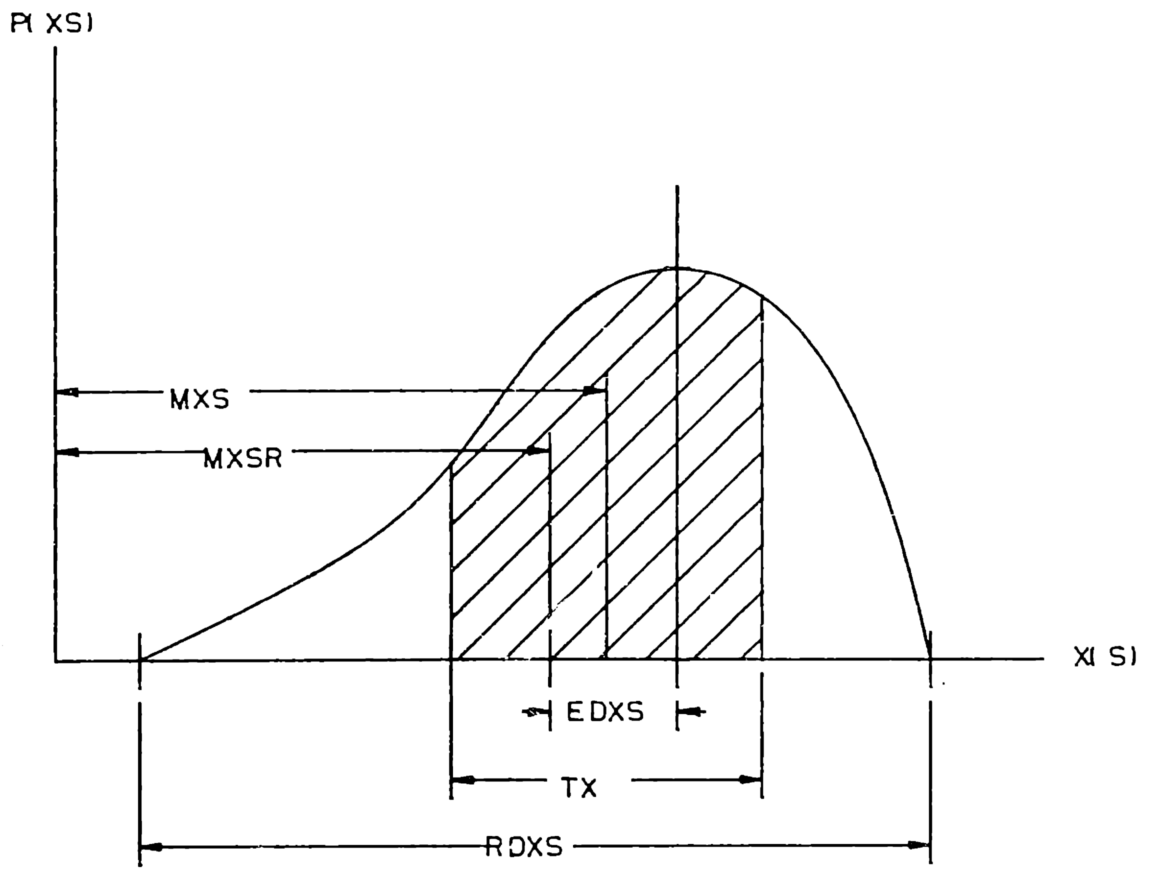
2.Ø TOLERANCE CHAINS

2.1 FUNCTIONAL DIMENSIONS

A functional or sum dimension, denoted by Xs, is a critical dimension. It affects the function of the assembly more than other dimensions. For each analysis there is only one sum dimension. For example the clearance between a shaft and a hole operation of rotating machinery would be a candidate sum dimension.

The statistical behavior of the sum dimension may be modelled by a beta probability distribution (fig.2.1). There are a number of statistical parameters that define the shape and size of the curve. MXSR is the distance to the middle of the range of the dimension. RDXS is the range of variation of the dimension. The expectation of the statistical part of the sum dimension is EDXS. TX is the tolerance of the sum dimension and MXS is the distance to the middle of the tolerance zone. The width of the hump of the curve is governed by variance of the sum dimension, VARDX.

Unlike a normal distribution, a beta distribution may be assymetrical. This increases the accuracy of the results, because many machining processes can not be expressed in terms of a normal probability distribution. The normal distribution is a special case of the more general beta distribution in which EDXS is equal to zero. The expressions governing the beta distribution are given in



PROBABILITY DISTRIBUTION
OF SUM DIMENSION

2.1 Beta probability distribution

appendix A.

Statistical parameters of the sum dimension are found by summing up the influence of individual dimensions, measured at a common location and in a common direction. This direction is known as the sum direction. The confidence level is given by the area under probability distribution curve bounded by the tolerance of the sum dimension. This area is shown hatched in fig.2.1.

2.2 FUNDAMENTAL EQUATIONS

To describe the influence of other dimensions on the sum dimension, Bjorke introduces the concept of the fundamental equation.

$$Xs = f(X(1), X(2), ... X(i), .. X(N)) \qquad (2.1)$$

where X(i) is an actual dimension in an assembly. In most cases, tolerances are small deviations, dX, in the nominal size of a dimension. This implies that The fundamental equation may be linearized.

$$X = MX + dX \qquad (2.2)$$

where MX is the distance to the middle of the tolerance zone. Equation (1) now becomes.

$$Xs = \sum_{i=1}^{N} A(i) * X(i) \qquad (2.3)$$

where A(i) is a signed constant. A linearized fundamental equation corresponds to the mathematical concept of a chain. The links or elements of the chain are dimensions. In the following sections, the fundemental equation will often be referred to as a tolerance chain, and the terms in the right hand side of the fundamental equation will be referred to as links.

3.Ø CHAIN LINKS

3.1 CLASSIFICATION OF LINKS

To determine the influence of a chain link (dimension) on the sum dimension, it is necessary to determine the relation between the link and the sum dimension. The link must be classified in terms of geometry, direction, magnitude and vector type. Also the statistical nature of that dimension must be determined.

Dimensions are classified under two types of geometries. They are spans and gaps. A span is the distance between surfaces on a part. A gap is the distance between surfaces on mating parts. There are also two types of directions. A lumped direction has a given but uncertain size. A distributed direction is uncertain as a whole. For instance the gap direction of an unrestrained shaft in a bore is totally uncertain with respect to the sum direction.

Gap magnitudes may also be lumped or distributed. Clearances are gaps that are always positive and interferences are always negative. Gaps that are indeterminately positive or negative are called transitions.

There are three vector types for dimensions. They are line, plane, and space. A dimension is a line vector when the dimension and its variation lay completely within or parallel to the axis of the sum dimension. A plane vector denotes a dimension lying completely within a plane containing the sum dimension, or a dimension lying completely within a plane parallel to the plane containing the sum dimension. When a dimension is not lying within the same plane as or a plane parallel to a plane containing the sum dimensin it is a space vector. A diagram of link classifications is shown in fig.3.1.

3.2 STATISTICAL PARAMETERS

The statistical parameters of a link are determined from the statistical parameters of the machining process required to produce the dimension associated with that link. The shapes of the distribution curves for commonly used machining processes have been expressed in terms of two unit distribution parameters and listed in appendix B. They are the unit expectation (EZ) and the unit variance (VarZ).

The Transformation between the unit statistical parameters and the parameters of the dimension in question are given by

$$dX = RdX * (Z - 0.5) \qquad (3.1)$$

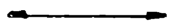
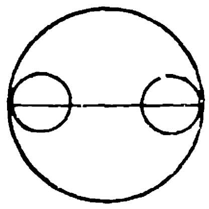
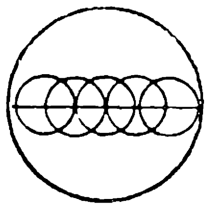
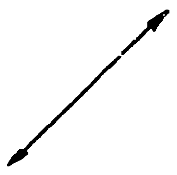
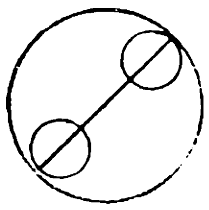
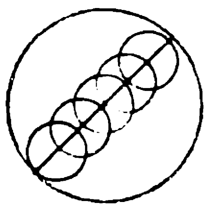
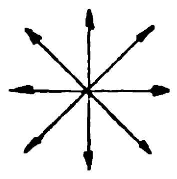
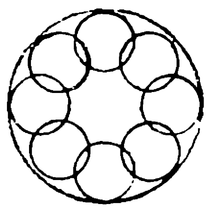
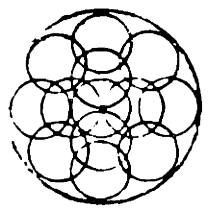
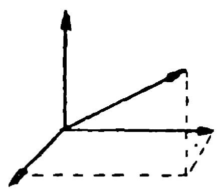
where dX is there statistical part of dimension, RdX is the range of the dimension, and Z is one of the unit parameters. In many manufacturing procedures components having dimensions outside the tolerance zone do not pass through inspection. Therefore we may say that, for components in an assembly, the range is equal to the tolerance

$$RdX = TX \qquad (3.2)$$

The sum of a variate and a constant

$$Y = X + a \qquad (3.3)$$

is given by

		Span	Gap	
			Lumped Magnitude	Distributed Magnitude
Line Vector				
Plane Vector	Lumped Direction			
	Distributed Direction			
Space Vector				

3.1 Link Classification

$$EY = EX + a \qquad (3.4)$$

$$VarY = Varx \qquad (3.5)$$

Using equations 3.1 through 3.5, it is now possible to express the expectation and the variance of a dimension by

$$EdX = TX * (EZ - 0.5) \qquad (3.6)$$

$$VardX = TX**2 * VarZ \qquad (3.7)$$

To obtain unit parameters from the tables in appendix B, a dimension must be classified in terms of form element. An example of a form element of a dimension is the distance between a reference plane and a center line. For each form element there is a set of technological processes, such as turning, grinding, or milling. For each technological process a range of tolerances is given that correspond to a range of unit parameters . Tolerances are expressed in international tolerance grades (2). Unit parameters are found by linear interpolation between the grade range and the parameter range. So given the international tolerance grade and knowledge of the form element, statistical parameters that define the probability distribution for a given dimension may be found in a table.

4.Ø ANALYSIS PROCEDURE

To analyze a tolerance chain, it is necessary to carry out two major tasks. The first task is to specify the characteristics of each link in the tolerance chain. The second task is to analyze the tolerance chain and estimate a confidence level for the assembly.

Once a tolerance chain has been chosen, there are four steps in the specification of a link. The first is chosing the sign of the link. The sign of a link depends on its relation to the sum dimension in the fundamental equation. The sign is represented by the signed constant, A, in equation 3.2. If an increase in the magnitude of a dimension increases the magnitude of the sum dimension the sign of that dimension is positive. If an increase in the magnitude of a dimension decreases the magnitude of the sum dimension the sign of that dimension is negative.

The second step is to classify the link as discussed in section 3.1. The classification of a link may require as many as six pieces of information or as few as two pieces of information, depending on the class.

The last two steps consist of indicating the dimension(s) of the link and choosing the unit parameters for each dimension. For a gap dimension there will be two dimensions for each link. The dimensions and tolerances are indicated by pointing to a particular component in the assembly data structure and then pointing to a dimension on that component. The unit parameters are chosen by first defining the form element of the dimension using appendix B as a guide. Finally a technological process must be selected from a list of possibilities for each form element.

The analysis of a tolerance chain is carried out in three steps. In the first step, a transformation is carried out on each link, to convert unit statistical parameters into dimension statistical parameters. Also the amount of influence each link has on the sum dimension is determined. As part of this process it is necessary to determine the international tolerance grade of each set of tolerances and dimensions from a table of grades. The tolerance grade is used to interpolate between minimum and maximum values in the table of unit parameters.

The statistical parameters of each dimension are found by applying equation 2.3 to the middle of the range, the range, the expectation, and the variance of each link

$$MXrs = \sum_{i=1}^{N} A(i) * MX(i) \qquad (4.1)$$

$$RXs = \sum_{i=1}^{N} |A(i)| * TX(i) \qquad (4.2)$$

$$EdXs = \sum_{i=1}^{N} A(i) * TX(i) * (EZ(i) - 0.5) \qquad (4.3)$$

$$VardXs = \sum_{i=1}^{N} A(i)**2 * TX**2 * VarZ(i) \qquad (4.4)$$

where 's' indicates the sum dimension. These equations vary depending on the classifcations, in terms of the geometry, vector type, direction, and magnited type, of each link. The equations above apply to a line vector span. A full set of the link equations listed in the analysis subprograms in appendix D.

In the second step, the statistical parameters for all the links are summed to determine the statistical parameters of the sum dimension. The parameters of the sum dimension are used to generate the beta probability distribution curve of the sum dimension. The necessary calculations are given in appendix A. The beta function requiures the generation of the gamma function. Since the confidence level is defined by the area under the distribution curve, numeric integration is carried out on the beta function. The bounds of integration are the minimum and maximum allowable dimensions. An example of this procedure will be given in section 6.4.

5.Ø STRUCTURE of COMPUTER PROGRAM

This section discusses the structure of the computer program that was developed to carry out the procedures mentioned in the previous sections. The program, called TOLANAL, which stands for tolerance analysis, is partially listed in appendix D. The source code is written in VAX FORTRAN IV-PLUS and it was developed on a Digital Equipment Corpation VAX 11/75Ø minicomputer in the Computer-Aided Design Laboratory in the Department of Mechanical Engineering.

TOLANAL was written using structured programming techniques. In other words, the program is divided up into many modules or subprograms. Some of the subprograms may themselves be broken up into smaller pieces. Each module is designed to carry out a specific task. This technique increases the amount of coding that is required, but it decreases the effort involved in debugging and maintainence.

The main module in TOLANAL is a program called CONTROL. CONTROL calls all of the other subprograms directly or indirectly. It is at the top of a hierarchy of program modules. One level below CONTROL, there are ten modules that are called directly by CONTROL. They are START, PROMPT, HELP, QUIT, GET, ADD, EDIT, ANALYZE, SHOW, and SAVE. Each of these modules may contain its own set of subprograms. One module may use subprograms that are part of another module. There are forty-two subroutines which, when added together, represent over one thousand lines of FORTRAN. The subroutines are listed in appendix E.

5.1 PROMPT

The module that is central to the interacive operation of the program is PROMPT. As the name implies, PROMPT prompts the user whenever any information is required. The PROMPT library consists of three subprograms, USER_PROMPT, LOOKUP, and MAKE_UPPER_CASE. PROMPT prompts the user by displaying text on the screen followed by a greater than sign, '>'. When the user is working in the highest level of TOLANAL, the prompt is 'TOL>'. If the user is in the EDIT module, the prompt is 'EDIT>', as in "edit what?". This system of prompts lets the user to know which task is active.

The first section of PROMPT checks for consistency of data input type. Data being requisted may be of three different types, real numbers, integer numbers, or character strings. If the data requested is a real number and a real number is not entered there are six possibilities. First PROMPT checks to see if any thing was entered at all. The user may have just hit the return key without entering anything. If that was not the case, the input is then checked to see if it was an integer. If the number is an integer, the number is converted to a real number. If the number is not an integer, then PROMPT checks whether or not the input was a command or 'valid' character string. If a real number is requested as input, PROMPT goes through a simular process.

PROMPT processes both commands and 'valid' character strings in the same manner. An example of an 'valid' character string is 'LINK', as in response to 'show what?'. The process begins by converting the input string to upper case. To do this PROMPT calls subroutine MAKE_UPPER_CASE. Upper and lower case letters have different internal computer representations and in a comparison between 'show' and 'SHOW' no match would be found.

After the string has been converted to upper case subroutine LOOKUP is called. Subroutine LOOKUP compares the input string to a list of allowable responses. Each string in the list corresponds to an OpCode Number (Operation Code). When a match is found, LOOKUP returns the OpCode associated with that match. If no match is made the OpCode equals 2. A match can be made on any group of unique first letters in the command. That is, 'ed' would match on 'EDIT'. If more than one match is made, for example 'e' matching on 'EDIT' and 'EXIT', LOOKUP returns and ambiguous command code of OpCode equal to three, and the ambiguities are displayed on the screen.

5.2 GET and SAVE

To retrieve any data stored in disk files there is the GET module. Within the GET task the user may retrieve an assembly file by invoking the GET_ASSEMBLY suprogram, or a previously specified tolerance chain may be retrieve by invoking GET_CHAIN. These subrograms don't access disk files directly. Instead they call three basic subrograms, GET_INTEGER_ARRAY, GET_REAL_ARRAY and GET_CHAR_ARRAY. These three subprograms open files and read the contents into arrays. GET_CHAIN and GET_ASSEMBLY both store the name of the most recently used file, so that if ther user has been working on the same files over a period of days, he/she can just hit a carriage return to have the same file retrieved at the beginning of a session.

Corresponding to GET there is a module called SAVE which does the reverse file operations. SAVE writes data files onto disk storage. The corresponding subprograms in the SAVE library are PUT_CHAIN, PUT_INTEGER_ARRAY, PUT_REAL_ARRAY, and PUT_CHARACTER_ARRAY.

5.3 START

Module START is the first subprogram called by CONTROL. It is only called once during a session. This module retrieves tables used by TOLANAL from disk storage using the GET subprograms. All of the tables are in a directory named 'Dmal:[Nathan.Data]'.

The table of tolerance grades is stored in a file named 'Grade.Dat'. The file is read into an array of twenty-one rows by twenty columns called Tol_Grade_Table. The rows in the array represent a range of dimensions, and the columns of the array represent international tolerance grades. Each entry in the table is a tolerance. The last two columns contain the minimum and maximum dimension of the range of dimensions in a particular row. All values in disk file are in milimeters. START converts these values to inches.

The tables containing the unit statistical parameters for specific machining processes are stored in five files. They are divided into two groups. One group contains information on the form elements, Form.Dat, and Form.Nam. The other group of files contains information on the technological processes, Tech.Dat, Tech.Nam, and Tech.Val. The data was seperated into five files to facilitate editing of the process tables and to separate integer, real, and character data so that they could be handled by the GET subprograms.

An additional table was necessary to define three graphical functions. These functions were required for the analysis of transition line gaps and lumped direction plane gaps. There were no analytical expressions available for these functions, so in order to evaluate them a least squares fit was made using a Chebyshev polynomial of the form

$$P(x) = b(1)*T(\emptyset,t) + b(2)*T(1,t) \ldots + b(m)*T((m-1),t) \quad (5.2)$$

The coefficients, b(i), for the three functions, are stored in three files named Zeta(n).Fit. START reads from these files into three arrays named Zeta(n)_Fit.

At this point START calls GET_ASSEMBLY, which prompts the user for the name of the file that contains the dimensions of the assembly. To run an analysis TOLANAL has to have access to an assembly. Another important piece of information is whether the user wishes to work on an old tolerance chain or start a new tolerance chain. If the user wishes to work on an old chain, START calls GET_CHAIN, which prompts the user for the name of the old chain. If the user wishes to work on a new chain, then START calls the input routine ADD_LINK, and the user begins entering a new chain.

5.4 ADD

The ADD module is very crucial the tolerance chain analysis. This routine leads the user through the specification of the tolerance chain. The specification of a chain requires a great amount of user-computer interaction. For some types of links there may be up to fifteen pieces of information that are required. The data structure of the tolerance chain data is shown in appencix D. The arrays that contain the information on the individual links are set up so that the index of the first row is zero. This position is reserved for the sum dimension, ie. link number zero is the sum dimension. ADD consists mainly of two subprograms, ADD_LINK and ADD_LINK_DIM. In ADD_LINK each link is classified by one of the catagories presented in section 3.1. ADD_LINK_DIM stands for add link dimension. In this routine, the user specifies any dimensions associated with that link. There is an additional subprogram named GET_TOL_GRADE, which uses the dimension information, supplied in the previous routine, to find the international tolerance grade for a dimenison and its tolerance. To find the grade, a search is conducted through an array, TOL_GRADE_TABLE. GET_TOL_GRADE is within ADD because the user requires the tolerance grade in order to choose a technological process.

5.5 ANALYZE

The main purpose of TOLANAL is analyzing the tolerance chain. All of the analysis is carried out in the ANALYZE module. ANALYZE calls LINK_ANALYSIS to obtain the statical parameters of each link. At the same time these values are being summed up to obtain the statistical parameters of the sum dimension.

To obtain the unit statistical parameters, EZ and VarZ, for a particlular dimension, LINK_ANALYSIS calls the subprogram GET_PARAM. GET_PARAM retrieves the required values from the technology process table. Often it is necessary to perform linear interpolation between entries in the table. As you might expect, INTERPOLATE exists for that reason.

Once the unit parameters have been chosen, LINK_ANALYSIS performs the the transformations on the unit parameters. There are a number of subprograms that carry out this operation for specific classes of links. For example, there is a subprogram called DD_PLANE_GAP_ANALYSIS.

DD stands for distributed direction.

The line gap analysis and the lumped direction plane gap analysis require three graphic functions that are expressed using a Chebyshev polynomial of the form

$$P(x) = b(1)*T(0,t) + b(2)*T(1,t) \ldots + b(m)*T((m-1),t) \quad (5.2)$$

The independent variable, x, is transformed into the variable, t, which has a range of -1 to +1.

$$t = (x-xo)/xd \quad (5.3)$$

$$xo = -\ (xr+xl)/(xr-xl) \quad (5.4)$$

$$xd = 2/(xr-xl) \quad (5.5)$$

where xl and xr bound the range of the fitted data points. T(k,t), obeys the recursive formula

$$T(k,t) = 2t * T((k-1),t) - T((k-2),t) \ ; \quad k>=2 \quad (5.6)$$

with starting values

$$T(0,t) = 1 \quad (5.7)$$

$$T(1,t) = t \quad (5.8)$$

The coefficients, b(i), are in three arrays named Zeta(n)_Fit. The first two values in the array are xo and xd.

With the statistical parameters of the sum dimension in hand, we are ready to generate the probablity distribution curve. The area of integration beneath the curve and bounded by the tolerance zone will determine the confidence level. The generation of the distribution curve was straightforward except for one point that should be mentioned. The expression for a beta probability distribution curve requires the generation of the gamma function. To handle this subprogram DLN_GAMMA_FUNCTION was written.

The name DLN_GAMMA_FUNCTION stands for the double precision computer representation of the natural log of the gamma function. The computer can handle numbers up to 1Ø raised to the 38th power. Numbers greater than this are common in this particular application. The natural log of the function was used to prevent the generation of numbers in excess of the computers capacity. To preserve as much accuracy as possible the numbers were represented in double precision. This allows for numbers of greater magnitude a greater number of significant digits.

Subprogram CONF sets up the parameters for the generation of the beta distribution curve. The curve is generated by subprogram BETA. CONF then calls INTEGRATE, a Simpson's rule integration routine. INTEGRATE calls BETA to obtain values of the beta function at each integration step. The integration is performed once on the whole curve and a second time on the portion of the curve that is bounded by the tolerance of the sum dimension. The area in the tolerance zone is divided by the total area and the result is the confidence level.

6.Ø OPERATION of PROGRAM

6.1 INPUT REQUIREMENTS

This section explains the interactive operation of TOLANAL and gives brief examples of input and output. The operation of the program requires access to thirteen data files:

ASSEMNAME.DAT	FORM.DAT	ZETA1.FIT
CHAINNAME.DAT	FORM.NAM	ZETA2.FIT
HELP.HLP	TECH.DAT	ZETA3.FIT
OPDATA.DAT	TECH.NAM	
	TECH.VAL	
	GRADE.DAT	

The program is designed to be run interactively. It is operated using a set of general user commands, which activate various tasks. In a task, the user is prompted for specific answers. The keyboard input to the program is in the form of charcter string commands, character string keywords, integer numbers and real numbers. To speed data input in the ADD module, allowable responses are assigned integer values, for example 'span=1'. If the question was what geometry type, you could enter '1'.

There are eight generic commands. Each command may have pseudonyms. For example you may use 'EXIT' instead of 'QUIT'. The full set of commands are:

Command	Pseudonym(s)
ADD	INPUT, SPECIFY
ANALYZE	RUN
CHANGE	EDIT, MODIFY
GET	READ
HELP	WHAT, ?
QUIT	EXIT, STOP
SHOW	LOOK, DISPLAY, TYPE, PRINT, PLOT
SAVE	WRITE, PUT

There is also a set of keywords that act as qualifiers for the commands. This allows interaction such as 'SHOW> PROCESS'.

OLD	NEW
TABLES	ASSEMBLY
CHAIN	LINK
DIMENSION	FORM
TECHNOLOGY	PROCESS
PROBABILITY	DISTRIBUTION

Any command or keyword may be abbreviated by n first letters as long as the abbreviation is unique within the total set of commands and keywords. 'CHA' is not a legal abbreviation, since it could mean either 'CHANGE' or 'CHAIN'.

6.2 START-UP PROCEDURE

The program is activated by running the executable image CONTROL.EXE. After the program has been started, the first thing that happens, is the program prompts the user for the name of the assembly file that is going to be analyzed. This step has to be carried out before any chains can be built or analyzed. If the user wishes to retrieve the most recently used file she/he can press the return key, and the name is called up from disk storage.

Next, the user is asked whether an old or new chain will be analyzed. If the response is 'OLD' the user is prompted for the chain name. If the response is 'NEW' the program goes into the input mode. The first link is link number zero and it represents the sum dimension.

6.3 DATA INPUT

To define a tolerance chain the user puts the program into the 'ADD' mode by entering 'ADD' or one of it pseudonyms. In the 'ADD' mode the program guides the user through each of the possible sixteen steps required to define each chain link. If the input of any previous steps causes a step to be unecessary, the step is skipped. For instance a line vector does not require any specifications on its direction. If user exits from the 'ADD' mode at any step, chain specification resumes at the same step the next time the 'ADD' command is issued. From this point on anytime the 'ADD' command is used, input will begin with the last step of the last link. All links must be completed before they can be edited. Every link in the chain must be complete, before the chain can be analyzed.

If a session must be terminated before the specification of the chain is complete the chain may be saved by issuing the 'SAVE' command. The user will be prompted for a file name. If the response is a carriage return TOLANAL will use the most recent name of a tolerance chain file and it will write over the old file. If the 'QUIT' command is issued to end a tolerance analysis sesison, the user will be asked if the current chain should be saved. This is meant to prevent the loss of tolerance chain data that was for time consuming to input.

A formated output of the tolerance chain can be displayed on the screen using the 'SHOW' command. If a hardcopy of this information is required the 'PRINT' command should be used. This command sends the formatted data to a file instead of the screen. The name of the file is 'TolChain.Lis'.

6.4 EXAMPLE RUN

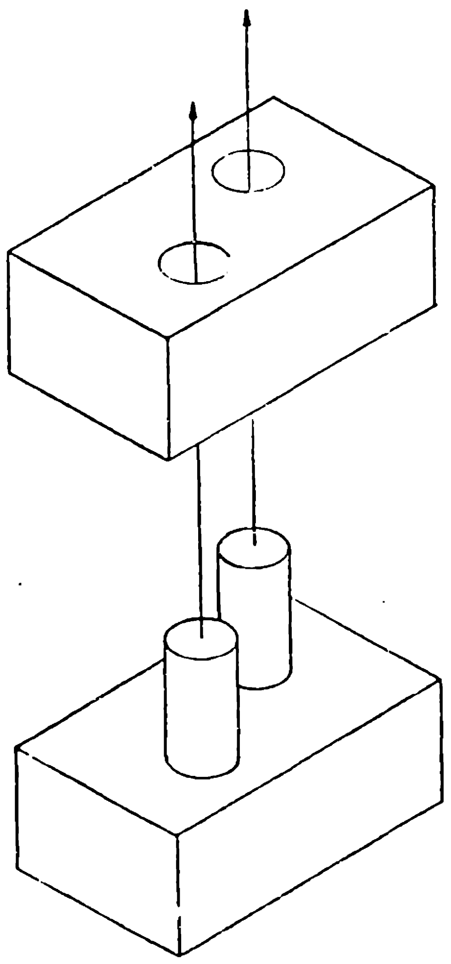
The assembly case study consists of two mating parts. One component is a rectangular prism with two parallel cylinders (or pins) projecting from one face. The other component is prisim of the same dimensions as the first except that it has two parallel holes in one face (fig.6.1-3). The assembly is stored using the data structure described in the MIT CADLAB Technical Document no. 4Ø.

The critical or sum dimension for this assembly is the clearance between the pin and its mating hole. Either pin hole pair may be chosen. The dimensions that affect the sum dimension are the distances between centers of the holes and the pins and the clearance between the second pin and its hole. These dimensions play major roles in determining whether or not the parts may be assembled (fig.6.4).

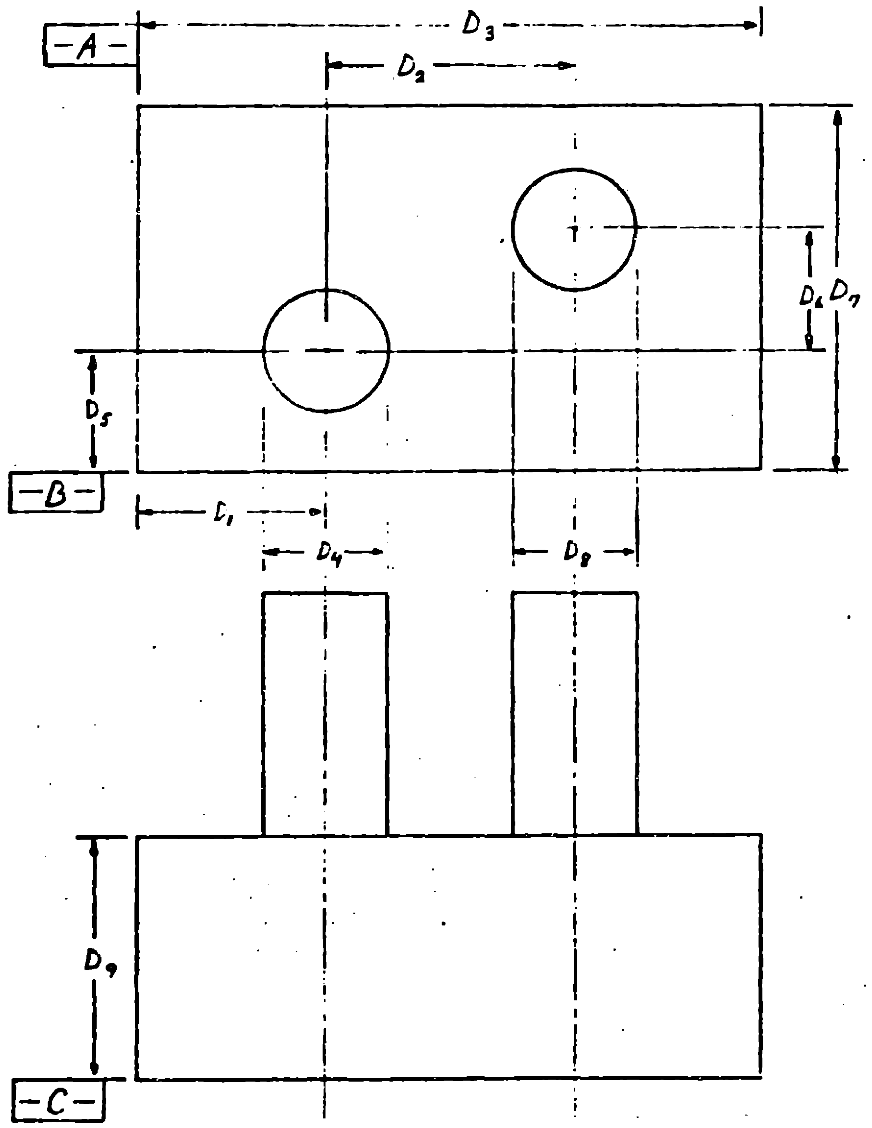
We will represent our chain as

$$Xs = X(1) - X(2) - X(3) \qquad (6.1)$$

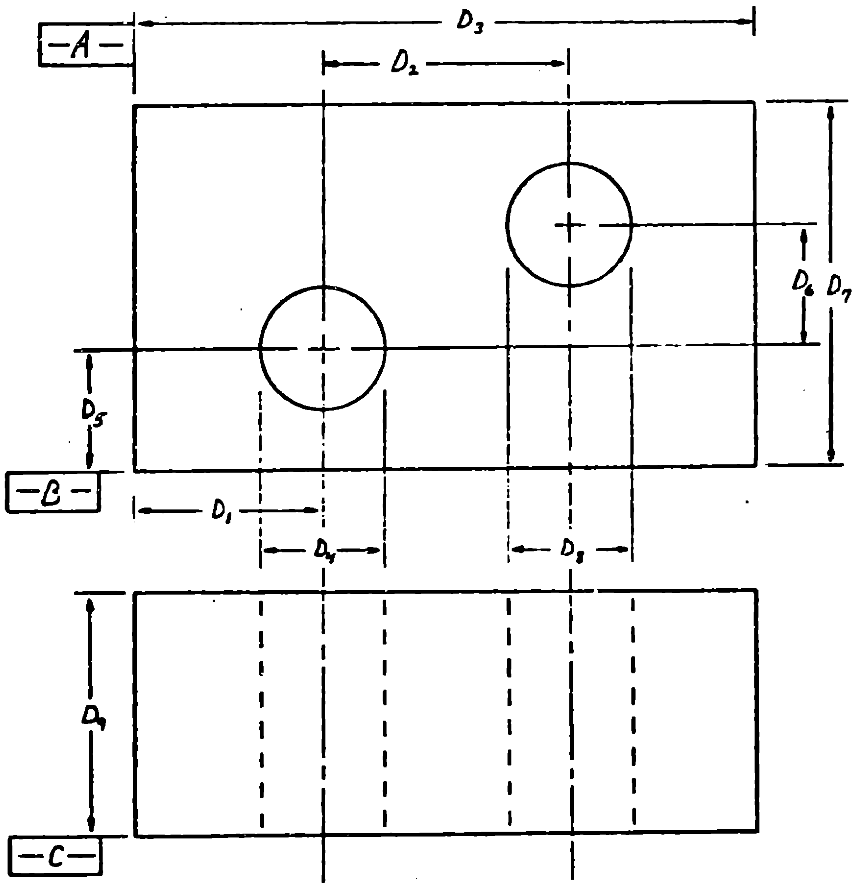
Xs is the gap between the hole dimension number four on female component and the pin dimension number four on the mail component. X(1) is the distance between the centers of the two holes. X(1) is made up of the components of Df2 and Df6 in the sum direction (fig.6.5). The convention is Df2 means dimension number 2 on the female component. Df2 makes an angle of 26.6 degrees with the sum direction and Df6 makes an angle of 63.4 degrees with the sum direction. X(2)



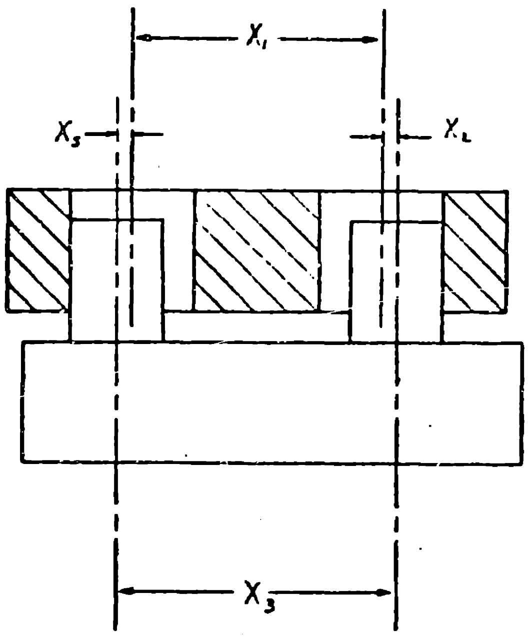
6.1 Exploded Drawing



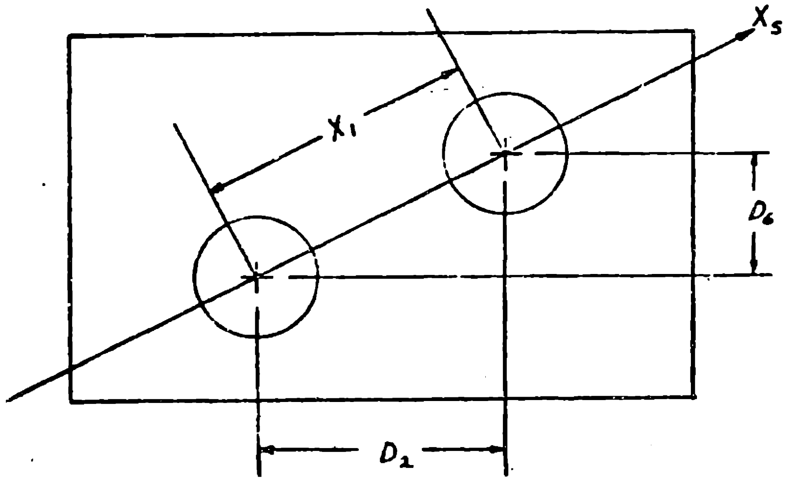
6.2 Male Component Drawing



6.3 Female Component Drawing



6.4 Schematic of Tolerance chain



6.5 Sum Direction

is the clearance gap between Df8 and Dm8. X(3) is made up of Dm2 and Dm6.

For our fundamental equation the dimension X(1) and X(3) are lumped direction plane vector spans. The dimension X(2) is a plane vector gap of distributed direction and lumped magnitude. Also note that A(1) is positive, and A(2) and A(3) are negative.

The form elements of the dimensions in question are as follows. Df2, Df6, Dm2, and Dm6 are the distance between center lines, form element number six. Df4 and Df8 are internal cylinders, form element number 2. And Dm4 and Dm8 are external cylinders, form element number 1.

The follow three pages contain a sample run. It begins by reading in a assembly file and entering the sum dimension. Then it skips to the reading in of a chain that was previously completed.

SAMPLE INPUT and OUTPUT

```
  TOL

Enter assembly name.
Press <RETURN> for default to most recent Assembly.


GET>ASSEMBLY>COMP2

RUN OLD chain or NEW chain ?


RUN>NEW

Input SUM DIMENSION


Dimension geometry type:                    1 = Span
                                            2 = Gap


ADD>LINK>GEOM>2

Enter component# of internal dimension


ADD>LINK>DIM>COMP#>2

Enter dimension#


ADD>LINK>DIM#>4

Process form element type :

1 External cylinders
2 Internal cylinders
3 Radial runout of cylindrical surfaces
4 Distance between external parallel planes
5 Distance between internal parallel planes
6 Distance between external and internal parallel planes
7 Parallelism, perpendicularity and angularity between surfaces
8 The distance from a center line to a ref plane
9 ||ism, |_arity, and <arity between a center line and a reference plane
10 The distance between center lines
11 Parallelism, perpendicularity and angularity between center lines

ADD>LINK>DIM>FORM#>1
```

SAMPLE INPUT and OUTPUT

```
Tolerance Grade = IT6


technolosy   name             max.srade       min.srade

 1           Roush turnins    IT11            IT13
 2           Grindins         IT7             IT9
 3           Face millins     IT5             IT7

ADD>LINK>TECH#>3

Enter component# of external dimension


ADD>LINK>DIM>COMP#>1

Enter dimension#


ADD>LINK>DIM#>4

Process form element type :

1 External cylinders
2 Internal cylinders
3 Radial runout of cylindrical surfaces
4 Distance between external parallel planes
5 Distance between internal parallel planes
6 Distance between external and internal parallel planes
7 Parallelism, perpendicularity and angularity between surfaces
8 The distance from a center line to a ref plane
9 ||ism, |_arity, and <arity between a center line and a reference plane
10 The distance between center lines
11 Parallelism, perpendicularity and angularity between center lines

ADD>LINK>DIM>FORM#>2

Tolerance Grade = IT7


technolosy   name             max.srade       min.srade

 1           Borins           IT11            IT13
 2           Drillins         IT10            IT14
 3           Grindins         IT8             IT10
 4           Reamins          IT7             IT9
 5           Face millins     IT5             IT7
 6           Honins           IT4             IT6

ADD>LINK>TECH#>4
```

SAMPLE INPUT and OUTPUT

```
=======================================================================================
Link#     |Sign(1)   |Geom(2)   |Vect(3)   |Dir(4)    |Theta(5)  |Mas(6)    |Distr(7)
----------+----------+----------+----------+----------+----------+----------+----------
    0     |    +     |   Gap    |  Line    |   ---    |   0.0    | Lumped   |   ---
=======================================================================================
          | Designation          |Comp(8)   |Dim(9)    |Form(10)  |Tech(11)  |Grade
          ---------------------+----------+----------+----------+----------+----------
          |   Internal           |    2     |    4     |    1     |    3     |IT6
          =============================================================================
          | Designation          |Comp(12)  |Dim(13)   |Form(14)  |Tech(15)  |Grade
          ---------------------+----------+----------+----------+----------+----------
          |   External           |    1     |    4     |    2     |    4     |IT7
          =============================================================================

ink number  1

ign of link ( + or - )

DD>LINK>SIGN>Q

OL>GET

Do you wish to retrieve a chain or assembly file?

GET>CHAIN

Enter chain name, press <RETURN> for default to most recent chain name

GET>CHAIN>

Chain name is CHAIN2

TOL>ANALYZE

 12.4% of the assemblies will have a sum
dimension within the specified tolerance

TOL>QUIT

Do you want to save the chain?

QUIT>NO
$
```

7.Ø CONCLUSIONS

In conclusion I would like to offer a few opinions and recommendations and also describe constraints put on the problem by this particular solution.

This method of tolerance anlysis is based on assumptions about the shapes of the probability distribution curves of dimensions. There are a list of common shapes in appendix B. The exact shape of the distribution curve depends highly upon the practices of the associated manufacturing facility. Each facility would have their own set of curves. The above conditions imply that the process table should be flexible. As it stands now TOLANAL has no tools for editing the process tables. But the tools could be easily developed and incorporated into the rest of the program. In the program the internal data structure of the technological process tables, was designed with editing in mind. All that is necessary is to write a process editing subprogram and put it in the 'EDIT' module.

Another point to be made is that a large amount of data that is input by hand for each link. It would be very beneficial to reduce this load on the user. It should not be necessary for the user to input some of this information since it is already stored in the assembly data structure.

An example of implied specifications is the specification of form element. If the dimension is a diameter it will have an associated edge. The edge will be part of a cylindrical surface representing a hole or a shaft. If it is part of a hole, there will be no surfaces at both ends of the hole. Therefore the form element is an internal cylinder. Once a sum direction has been chosen, it is possible to find the vector type of a dimension in the same manner.

TOLANAL is just a foundation on which many tools can be built. There are a lot of features that should be added to TOLANAL, and it was set up so that extra feature could be added with minimal impact on the rest of the system. A few modifications have been suggested that may make this a very powerful tool. In an ideal situation, a manufacturing facility would use this program in conjunction with designers to analyze an assembly that has been completely designed on a computer-aided design system to determine what percentage of a certain critical dimension on assemblies would meet dimensional specifications.

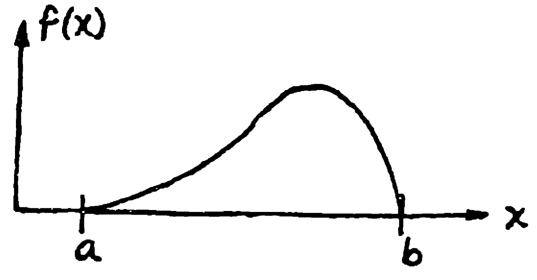
REFERENCES

1. Computer-Aided Tolerancing, Oyvind Bjorke, Tapir publishers 1978,Trondheim, Norway.

2. Engineering Drawing and Design, 2nd Ed., Cecil Jensen and Jay Helsel McGraw-Hill Book Company 1979, United States.

APPENDIX A

BETA PROBABILITY DISTRIBUTION



$$f(x) = \frac{1}{(b-a)B(\gamma,\eta)}\left(\frac{x-a}{b-a}\right)^{\gamma-1}\left(1-\left(\frac{x-a}{b-a}\right)\right) \tag{A.1}$$

$$a = MXsr - RdX/2 \tag{A.2}$$

$$b = MXsr + RdX/2 \tag{A.3}$$

$$\gamma = \frac{(EX-a)^2(b-EX) - varX(EX-a)}{varX(b-a)} \tag{A.4}$$

$$\eta = \frac{(EX-a)(b-EX)^2 - varX(b-EX)}{varX(b-a)} \tag{A.5}$$

$$EX = MXsr + EdX \tag{A.6}$$

$$VarX = VardX \tag{A.7}$$

$$B(\gamma,\eta) = \frac{\Gamma(\gamma)\Gamma(\eta)}{\Gamma(\gamma+\eta)} \tag{A.8}$$

$$\Gamma = \int_0^\infty t^{x-1}e^{-t}dt \tag{A.9}$$

APPENDIX B

TECHNOLOGICAL PROCESS TABLES

Form element	Technological process	TX		EZ		varZ	
		min	max	min	max	min	max
External cylinders	Rough turning	IT11	IT13	0.40	0.60	0.034	0.054
	Finishing turning	IT7	IT9	0.40	0.60	0.030	0.054
	Grinding	IT5	IT7	0.50	0.55	0.028	0.047
Internal cylinders	Rough turning	IT11	IT13	0.40	0.50	0.034	0.054
	Drilling	IT10	IT14	0.40	0.50	0.034	0.066
	Finishing turning	IT8	IT10	0.40	0.50	0.030	0.054
	Reaming	IT7	IT9	0.45	0.50	0.028	0.040
	Grinding	IT5	IT7	0.40	0.50	0.028	0.047
	Honing	IT4	IT6	0.45	0.50	0.028	0.034
Radial runout of cylindrical sufaces	Turning	0.02	0.10	0.30	0.40	0.035	0.040
	Grinding	0.01	0.02	0.30	0.40	0.035	0.038
The distance between external parallel planes	Cut off	IT12	IT16	0.30	0.70	0.040	0.083
	Turning	IT10	IT13	0.50	0.65	0.028	0.054
	Milling, planing	IT10	IT15	0.40	0.60	0.028	0.054
	Grinding	IT5	IT7	0.45	0.55	0.028	0.050

APPENDIX B

TECHNOLOGICAL PROCESS TABLES

Form element	Technological process	TX		EZ		varZ	
		min	max	min	max	min	max
The distance between internal parallel planes	Turning	IT10	IT13	0.40	0.60	0.034	0.054
	Milling, planing	IT10	IT15	0.38	0.62	0.034	0.054
	Grinding	IT5	IT7	0.38	0.62	0.040	0.050
The distance between external and internal planes	Turning	IT10	IT13	0.40	0.60	0.034	0.054
	Milling, planing	IT10	IT15	0.50	0.55	0.034	0.054
	Grinding	IT5	IT7	0.48	0.52	0.034	0.050
Parallelism, perpendicularity and angularity between surfaces	Planing	$\frac{0.1}{300}$	$\frac{0.2}{300}$	0.50	0.50	0.034	0.054
	Milling	$\frac{0.1}{300}$	$\frac{0.3}{300}$	0.50	0.50	0.034	0.040
	Grinding	$\frac{0.02}{300}$	$\frac{0.1}{300}$	0.50	0.50	0.040	0.054

APPENDIX B

TECHNOLOGICAL PROCESS TABLES

Form element	Technological process	TX		EZ		varZ	
		min	max	min	max	min	max
The distance from a center line to a reference plane	Boring	IT7	IT10	0.48	0.52	0.028	0.040
	Face milling	IT9	IT12	0.50	0.65	0.028	0.047
	Face grinding	IT6	IT9	0.65	0.70	0.047	0.056
Parallelism, perpendicularity and angularity between a center line and a reference plane	Boring	$\frac{0.05}{300}$	$\frac{0.1}{300}$	0.50	0.50	0.034	0.047
	Face milling	$\frac{0.1}{300}$	$\frac{0.3}{300}$	0.50	0.50	0.034	0.040
	Face grinding	$\frac{0.05}{300}$	$\frac{0.1}{300}$	0.50	0.50	0.040	0.047
The distance between center-lines	Boring	IT7	IT10	0.50	0.55	0.028	0.047

APPENDIX B

TECHNOLOGICAL PROCESS TABLES

Form element	Technological process	TX		EZ		varZ	
		min	max	min	max	min	max
Parallelism, perpen-duçularity and angularity between center lines	Boring	$\frac{0.05}{300}$	$\frac{0.2}{300}$	0.5	0.5	0.034	0.047

APPENDIX C

LIST OF SUBROUTINES

Subprogram Name	Source File	Module Library
ADD_LINK	ADDIM.FOR	ADD
ADD_LINK_DIM	ADDLINK.FOR	ADD
ANALYZE	ANALYZE.FOR	ANALYZE
BETA	CONF.FOR	ANALYZE
CONF	CONF.FOR	ANALYZE
CONTROL	CONTROL.FOR	CONTROL
DD_PLANE_GAP_ANALYSIS	DDANAL.FOR	ANALYZE
DINTEGRATE	DINTEG.FOR	ANALYZE
DLNGAMMAFUNCTION	DLNGAMMA.FOR	ANALYZE
EDIT	EDIT.FOR	EDIT
EDIT_LINK	EDITLINK.FOR	EDIT
GAP_ANALYSIS	GAPANAL.FOR	ANALYZE
GET	GET.FOR	GET
GET_ASSEMBLY	GETASSEM.FOR	GET
GET_CHAIN	GETCHAIN.FOR	GET
GET_CHAR_ARRAY	GETCHAR.FOR	GET
GET_INTEGER_ARRAY	GETINT.FOR	GET
GET_LINK_PARAM	GETPARAM.FOR	ANALYZE
GET_OPDATA	GETOPDATA.FOR	START
GET_REAL_ARRAY	GETREAL.FOR	GET
GET_TOL_GRADE	GETGRADE.FOR	ADD
GRADE_SYMBOL	GRADESYM.FOR	SHOW
HELP	HELP.FOR	CONTROL
INTERPOLATE	INTERP.FOR	ANALYZE
LD_PLANE_GAP_ANALYSIS	LDANAL.FOR	ANALYZE
LINE_GAP_ANALYSIS	LGANAL.FOR	ANALYZE
LINK_ANALYSIS	LINKANAL.FOR	ANALYZE
LOOKUP	LOOKUP.FOR	PROMPT
MAKE_UPPER_CASE	MAKEUP.FOR	PROMPT
POLYNOMIAL	POLYCURV.FOR	ANALYZE
PRINT_LINK	PRINT.FOR	SHOW
PUT_CHAIN	PUTCHAIN.FOR	SAVE
PUT_CHAR_ARRAY	PUTCHAR.FOR	SAVE
PUT_INTEGER_ARRAY	PUTINT.FOR	SAVE
PUT_REAL_ARRAY	PUTREAL.FOR	SAVE
QUIT	QUIT.FOR	CONTROL
SAVE	SAVE.FOR	SAVE
SHOW	SHOW.FOR	SHOW
SHOW_LINK	SHOWLINK.FOR	SHOW
SPAN_ANALYSIS	SPANANAL.FOR	ANALYZE
START	START.FOR	START
USER_PROMPT	PROMPT.FOR	PROMPT

APPENDIX D

COMPUTER CODE

```
C==========================================================================
C Data structure of tolerance chain
C--------------------------------------------------------------------------
        Parameter       MLink = 1ØØ
        Parameter       MLinkDim = 2*MLink
C--------------------------------------------------------------------------

        Integer Link( 8, Ø:MLink )
C       Filename AssemName.Lin
C       Link( 1, I )    Status          1-16
C       Link( 2, I )    Geometry        1-2 Span, Gap
C       Link( 3, I )    Vector          1-3 Line, Plane, Space
C       Link( 4, I )    Direction       1-2 Lumped, Distributed
C       Link( 5, I )    Magnitude       1-2 Lumped, Distributed
C       Link( 6, I )    Distribution    1-2 Normal, Rectangular
C       Link( 7, I )    Link_Dim1_Pntr
C       Link( 8, I )    Link_Dim2_Pntr

        Real Link_Value( 6, Ø:MLink )
C       Filename AssemName.LiV
C       Link_Value( 1, I )      Signed_Const      A
C       Link_Value( 2, I )      Angle             Theta if lumped dir
C       Link_Value( 3, I )      Middle_X_Range    Mxr
C       Link_Value( 4, I )      Range_Dx          Rdx
C       Link_Value( 5, I )      Expectation_Dx    Edx
C       Link_Value( 6, I )      Variance_Dx       Vardx

        Integer Link_Dim( 4, MLinkDim )
C       Filename AssemName.LiD
C       Link_Dim( 1, I )        Comp_Pntr
C       Link_Dim( 2, I )        Dim_Value_Pntr
C       Link_Dim( 3, I )        Form_Pntr
C       Link_Dim( 4, I )        Tech_Pntr

        Real    Link_Dim_Value( MLinkDim )
C       Filename AssemName.LDV
C       Link_Dim_Value( 1, I )  Grade
C               Depending on Form_Element
C               1. = IT1,..
C               Ø.Ø2 = Ø.Ø2,...
C               Ø.1 = Ø.1/3ØØ.,...

        Common / Chain /        Link,           NLink,
        1                       Link_Value,
        1                       Link_Dim,       NLinkDim,
        1                       Link_Dim_Value,
        1                       Link_Pntr
C==========================================================================
```

APPENDIX D

COMPUTER CODE

```
C=========================================================================
C Data Structure of internally stored tables
C-------------------------------------------------------------------------
        Parameter       MForm = 11
        Parameter       MTech = 32
        Parameter       MTechName = 14

        Integer Form( 11, MForm )
C       Filename Form.Dat
C       Form( 1, I )    No_Tech
C       Form( 2, I )    Tech1_Pntr
C       ...
C       Form( 11, I )   Tech10_Pntr

        Character *80 Form_Name( MForm )
C       Filename Form.Nam

        Integer Tech( 2, MTech )
C       Filename Tech.Dat
C       Tech( 1, I )    Form_Pntr
C       Tech( 2, I )    Tech_Name_Pntr

        Character *20 Tech_Name( MTechName )
C       Filename Tech.Nam

        Real Tech_Value( 6, MTech )
C       Filename Tech.Val
C       Tech_Value( 1, I )      Grade_Min
C       Tech_Value( 2, I )      Grade_Max
C       Tech_Value( 3, I )      Unit_Exp_Min
C       Tech_Value( 4 ,I )      Unit_Exp_Max
C       Tech_Value( 5, I )      Unit_Var_Min
C       Tech_Value( 6, I )      Unit_Var_Max

        Common / Process /      Form,           NForm,  NFormName,
        1                       Tech,           NTech,  NTechName,
        1                       Tech_Value
        Common / Process_Name / Form_Name,
        1                       Tech_Name
        Common / Grade /        Grade_Table( 20, 21 )

        Parameter               MZetaCoef = 20
        Common / Zeta_Fit /     Zeta1_Coef( MZetaCoef ), NZeta1_Coef,
        1                       Zeta2_Coef( MZetaCoef ), NZeta2_Coef,
        1                       Zeta3_Coef( MZetaCoef ), NZeta3_Coef
C=========================================================================
```

APPENDIX D

COMPUTER CODE

```
C=====================================================================
        Program TOLANAL
C=====================================================================
        Include '_Dmal:[Nathan.Common]Status.Cmn'
C---------------------------------------------------------------------
        Call Start

10      If ( OpCode .Lt. 600 .Or. OpCode .Ge. 2000 )
        1       Call User_Prompt( 'TOL>', Com, Ans, IVal, Val )
C---------------------------------------------------------------------
        If ( OpCode .Ge. 400 .And. OpCode .Lt. 500 ) Then
            Mode = 0
            Call Help
        Else If ( Opcode .Ge. 500 .And. OpCode .Lt. 600 ) Then
            Mode = Quit_Mode
            Call Quit
            If ( Opcode .Eq. 2001 .Or. Opcode .Eq. 2002 ) Go To 999
        Else If ( OpCode .Ge. 1000 .And. OpCode .Lt. 1100 ) Then
            Mode = Get_Mode
            Call Get
        Else If ( OpCode .Ge. 1100 .And. OpCode .Lt. 1200 ) Then
            Mode = Add_Mode
            Call Add_Link
        Else If ( OpCode .Ge. 1200 .And. OpCode .Lt. 1300 ) Then
            Mode = Edit_Mode
            Call Edit
        Else If ( OpCode .Ge. 1300 .And. OpCode .Lt. 1400 ) Then
            Mode = Analyze_Mode
            Call Analyze
            OpCode = 500
        Else If ( OpCode .Ge. 1400 .And. OpCode .Lt. 1500 ) Then
            Mode = Show_Mode
            Call Show
        Else If ( OpCode .Ge. 1500 .And. OpCode .Lt. 1600 ) Then
            Mode = Save_Mode
            Call Save
        Else If ( OpCode .Ge. 2000 ) Then
            Write ( 6, 801 )
801         Format(/' *** Invalid at this level ***'/)
            End If
C---------------------------------------------------------------------
        Go To 10
999     Continue
        End
C=====================================================================
```

APPENDIX D

COMPUTER CODE

```
C===========================================================================
        Subroutine User_Prompt( Prompt, Type, Ans, IVal, Val )
C===========================================================================
        Include 'Dmal:[Nathan.Common]Opdata.Cmn'
        Include 'Dmal:[Nathan.Common]Status.Cmn'
        Character *(*) Prompt
        Byte String(2Ø)
C---------------------------------------------------------------------------
        Write( 6, 8Ø1 ) Prompt
8Ø1     Format(/' ', A,$)
C---------------------------------------------------------------------------
        Read ( 5, 7Ø1 ) Length, ( String(I), I=1, Length )
7Ø1     Format ( Q, 2ØA1 )

        If ( Length .Eq. Ø ) Then
            OpCode = 1
            Return
        Else If ( Type .Eq. Integ_Data ) Then
            Decode( Length, 7Ø2, String, Err=6Ø1 ) IVal
7Ø2         Format( I1Ø )
            OpCode = Ø
            Return
        Else If ( Type .Eq. Real_Data ) Then
            Decode( Length, 7Ø3, String, Err=6Ø1 ) Val
7Ø3         Format( F1Ø.5 )
            Decode( Length, 7Ø4, String, Err=6Ø1 ) Ans
            If ( Index( Ans, '.') .Eq. Ø ) Then
                Length = Length + 1
                String(Length) = '.'
                Decode( Length, 7Ø3, String, Err=6Ø1 ) Val
                End If
            OpCode = Ø
            Return
            End If
6Ø1     Decode( Length, 7Ø4, String, Err=6Ø2 ) Ans
7Ø4     Format( A )
6Ø2     Call Make_Upper_Case( Ans )
        Call Lookup( Ans, Opcode )

        If ( Opcode .Eq. 2 ) Write( 6, 8Ø2 )
8Ø2         Format(/' **** Unkown Command **** '/
1                   ' Type  "HELP"  for  help '/
1                   ' ************************ '/)

        If ( Opcode .Eq. 3 ) Write( 6, 8Ø3 )
            Format(/' *** Ambiguous Command ** '/)
C---------------------------------------------------------------------------
        Return
        End
C===========================================================================
```

APPENDIX D

COMPUTER CODE

```
C=========================================================================
       Subroutine Make_Upper_Case (String)
C-------------------------------------------------------------------------
C  Convert all lowercase letters to uppercase,
C  and remove leading blanks.
C-------------------------------------------------------------------------
       Character *(*) String
       Character Letter *1

       L = Len (String)

C-------------------------------------------------------------------------
C  String --> STRING
C-------------------------------------------------------------------------
       Do 2ØØ I = 1, L
         Letter = String (I:I)

C        Check if Letter is lowercase.
         If ( (Letter .ge. 'a') .and. (Letter .le. 'z') ) Then
           Number = Ichar (Letter)      !ascii to integer
           Number = Number - 32
           Letter = Char (Number)       !integer to ascii
         Endif

         String = String (:I-1) // Letter // String (I+1:)

2ØØ    Continue
C-------------------------------------------------------------------------
C   ___String --> String___
C-------------------------------------------------------------------------
       Do 3ØØ I = 1, L
         If (String (:1) .eq. ' ') String = String (2:)
3ØØ    Continue
C-------------------------------------------------------------------------
       Return
       End
C=========================================================================
```

APPENDIX D

COMPUTER CODE

```
C=========================================================================
        Subroutine Lookup (Command, Opcode)
C-------------------------------------------------------------------------
C  Search through lookup table for the opcode
C  which corresponds to this command.
C-------------------------------------------------------------------------
        Include 'Dmal:[Nathan.Common]OpData.Cmn'

        Integer Opcode
        Character Command *(*)
        Character First_Ambiguous_Name *10

        Opcode = 2                          !Opcode for Opcode not found.
        LocSpace = Index (Command, ' ')

        Do 100 I = 1, Nop
          If ( Command (:LocSpace-1)
        1       .eq. Op_Name (I)(:LocSpace-1) ) Then
C               Write (6, 10) Op_Number(I), Op_Name (I)
10              Format (1H , 'Opcode: ', I6, ', ', A)
C           ---------------------------------------
C           Check for multiple command equivalence.
C           ---------------------------------------
            If ( (Opcode .eq. 2) ) Then
                First_Ambiguous_Name = Op_Name (I)
                Opcode = Op_Number (I)
            Else
                Opcode = 3                  !Redundent command.
                Write( 6, 804 ) Op_Name( I ), char(7)
804             Format(1h,5x,A)
            Endif
          Endif
100     Continue

        If ( (Opcode .eq. 3) ) Write( 6, 804 ) First_Ambiguous_Name

        Return
        End
C=========================================================================
```

APPENDIX D

COMPUTER CODE

```
C=========================================================================
C  Data structure of all analysis routines
C-------------------------------------------------------------------------
        Integer Geometry, Vector, Direction, Magnitude, Distribution
        Real    A, Mx, Ux, Lx, Tx, Ez, Varz,
        1       Mxa, Uxa, Lxa, Txa, Eza, Varza,
        2       Mxb, Uxb, Lxb, Txb, Ezb, Varzb,
        3       Varoxr, Varxr, Exr, Mxr, Rdx, Edx, Vardx
C-------------------------------------------------------------------------
        Parameter       Span = 1,
        1               Gap = 2,
        2               Line = 1,
        3               Plane = 2,
        4               Space = 3,
        5               Lumped = 1,
        6               Distributed = 2,
        7               Normal = 1,
        8               Rectangular = 2
C=========================================================================
C  This is to be included in all analysis routines
C-------------------------------------------------------------------------
        A = Link_Value( 1, Link_Pntr )
        Geometry =      Link( 2, Link_Pntr )
        Vector =        Link( 3, Link_Pntr )
        Direction =     Link( 4, Link_Pntr )
        Magnitude =     Link( 5, Link_Pntr )
        Distribution =  Link( 6, Link_Pntr )
C-------------------------------------------------------------------------
        If ( Geometry  .Eq. Span ) Then
        Call Get_Link_Param( 1, Mx, Ux, Lx, Tx, Ez, Varz, Theta )

        Else If ( Geometry  .Eq. Gap ) Then
        Call Get_Link_Param( 1, Mxa, Uxa, Lxa, Txa, Eza, Varza, Theta )
        Call Get_Link_Param( 2, Mxb, Uxb, Lxb, Txb, Ezb, Varzb, Theta )

        End If
C=========================================================================
```

APPENDIX D

COMPUTER CODE

```
C=============================================================
        Subroutine Span_Analysis
C=============================================================
C       This routine finds this statistitical contribution of
C       a span chain link to the sum dimension of the tolerance
C       chain, depending on the link vector type.
C-------------------------------------------------------------
        Include '[Nathan.Common]Assembly.Cmn'
        Include '[Nathan.Common]Chain.Cmn'
        Include '[Nathan.Common]GetParam.Cmn'
        Include '[Nathan.Common]GetParam.Inc'
C-------------------------------------------------------------
        If ( Geometry .Eq. Line ) Then
            Mxr = A * Mx
            Rdx = Abs( A ) * Tx
            Edx = A * Tx * ( Ez - Ø.5 )
            Vardx = A**2 * Tx**2 * Varz
C-------------------------------------------------------------
        Else If ( Geometry .Eq. Plane ) Then
C-------------------------------------------------------------
            If ( Direction .Eq. Lumped ) Then
                Mxr = Cos( Theta ) * Mx
                Rdx = Abs( A*Cos( Theta ) ) * Tx
                Edx = A*Cos( Theta ) * Tx * ( Ez - Ø.5 )
                Vardx = (A*Cos( Theta ))**2 * Tx**2 * Varz
C-------------------------------------------------------------
            Else If ( Direction .Eq. Distributed ) Then
                Mxr = Ø.
                Rdx = Abs( A ) * 2. * Tx
                Edx = Ø.
                Varozr = Varz + Ez**2
                Vardx = A**2 * Ø.5 * Txr**2 * Varozr
C-------------------------------------------------------------
                End If
C-------------------------------------------------------------
            End If
C-------------------------------------------------------------
        Link_Value( 3, Link_Pntr ) = Mxr
        Link_Value( 4, Link_Pntr ) = Rdx
        Link_Value( 5, Link_Pntr ) = Edx
        Link_Value( 6, Link_Pntr ) = Vardx
C-------------------------------------------------------------
        Return
        End
C=============================================================
```

APPENDIX D

COMPUTER CODE

```
C=====================================================================
      Subroutine Line_Gap_Analysis
C=====================================================================
C     This routine finds this statistitical contribution of
C     each line gap chain link to the sum dimension of the
C     tolerance chain, depending on the magnitued type.
C---------------------------------------------------------------------
      Include '[Nathan.Common]Chain.Cmn'
      Include '[Nathan.Common]Tables.Cmn'
      Include '[Nathan.Common]GetParam.Cmn'
      Include '[Nathan.Common]GetParam.Inc'
C---------------------------------------------------------------------
      If ( Magnitude .Eq. Lumped ) Then
C         -----------------------------------------------------------
          If ( Lxb .Ge. Uxa ) Then
C             -------------------------------------------------------
C             Clearance Lxb >= Uxa
              Mxr = (A/2.) * (Mxb - Mxa)
              Rdx = (Abs(A)/2.) * (Txb - Txa)
              Edx = (A/2.) * (Txb * (Ezb-Ø.5) - Txa * (Eza-Ø.5) )
              Vardx = (A**2/4.) * (Txb**2 * Varzb + Txa**2 * Varza)
C             -------------------------------------------------------
          Else If ( Lxb .Le. Uxa .And. Uxb .Ge. Lxa ) Then
C             -------------------------------------------------------
C             Transition Lxb <= Uxa and Uxb >= Lxa
              Exr = Ø.5 * (Mxb - Mxa + Txb*(Ezb-Ø.5) - Txa*(Eza-Ø.5) )
              Varxr = Ø.25 * (Txb**2 * Varzb + Txa**2 * Varza)
              sigm_xr = Sqrt( Varxr )
              zeta = Exr / sigma_xr
              zeta_Ex = Polynomial( zeta, NZetal_Coef, Zetal_Coef )
              If ( Distribution .Eq. Rectangular ) Then
              zeta_Varx = Polynomial( zeta, NZeta2_Coef, Zeta2_Coef )
              Else If ( Distribution .Eq. Normal ) Then
              zeta_Varx = Polynomial( zeta, NZeta3_Coef, Zeta3_Coef )
              End If

              Mxr = (A/4.) * (Uxb - Lxa)
              Rdx = (Abs( A )/2.) * (Uxb - Lxa)
              Edx = A * zeta_Ex * sigma_xr - Mxr
              Vardx = A**2 * zeta_Varx ∓ Varxr
C             -------------------------------------------------------
          Else If ( Uxb .Le. Lxa ) Then
C             -------------------------------------------------------
C             Interference Uxb <= Lxa
C             Neglected
C             -------------------------------------------------------
              Mxr = Ø.
              Rdx = Ø.
              Edx = Ø.
              Vardx = Ø.
C             -------------------------------------------------------
              End If
```

APPENDIX D

COMPUTER CODE

```
      Else If ( Magnitude .Eq. Distributed ) Then
C         ------------------------------------------------------------
          If ( Distribution .Eq. Normal ) Then
C             --------------------------------------------------------
C             Norm Distr Center Loc
              Exr = Ø.5 * (Mxb - Mxa + Txb*(Ezb-Ø.5) - Txa*(Eza-Ø.5) )
              Varxr = Ø.25 * (Txb**2 * Varzb + Txa**2 * Varza)
              Varxoxr = Varxr + Exr**2
              Mxr = Ø.
              Rdx = Abs( A ) * (Uxb - Lxa)
              Edx = Ø.
              Vardx = (A**2/9.) * Varxoxr
C             --------------------------------------------------------
          Else If ( Distribution .Eq. Rectangular ) Then
C             --------------------------------------------------------
C             Rect Distr Center Loc
              Exr = Ø.5 * (Mxb - Mxa + Txb*(Ezb-Ø.5) - Txa*(Eza-Ø.5) )
              Varxr = Ø.25 * (Txb**2 * Varzb + Txa**2 * Varza)
              Varxoxr = Varxr + Exr**2
              Mxr = Ø.
              Rdx = Abs( A ) * (Uxb - Lxa)
              Edx = Ø.
              Vardx = (A**2/3.) * Varxoxr
C             --------------------------------------------------------
              End if
C         ------------------------------------------------------------
          End if
C---------------------------------------------------------------------
      Link_Value( 3, Link_Pntr ) = Mxr
      Link_Value( 4, Link_Pntr ) = Rdx
      Link_Value( 5, Link_Pntr ) = Edx
      Link_Value( 6, Link_Pntr ) = Vardx
C---------------------------------------------------------------------
      Return
      End
C=====================================================================
```

APPENDIX D

COMPUTER CODE

```
C==========================================================================
        Function Polynomial( X, Poly_Deg, Coef )
C==========================================================================
C This routine evaluated the Chebychev polynomial
C--------------------------------------------------------------------------
        Real X, Z, Coef( 1 ), T( 100 )
        Integer Poly_Deg
C--------------------------------------------------------------------------
        Z = Coef(1)*(X+1.5) + Coef(2)
        T( 1 ) = 1.
        T( 2 ) = Z
        If ( Poly_Deg .Gt. 2 ) Then
C           ---------------------------------------------------------------
            Do 10 k=3, Poly_Deg
                T( k ) = 2 * Z * T( k - 1 ) - T( k - 2 )
10              Continue
C           ---------------------------------------------------------------
            End If
C--------------------------------------------------------------------------
        Polynomial = 0.
        Do 20 I=1, Poly_Deg
            Polynomial = Polynomial + Coef( I + 2) * T( I )
20          Continue
C--------------------------------------------------------------------------
        Return
        End
C==========================================================================
```

APPENDIX D

COMPUTER CODE

```
C===========================================================================
       Subroutine LD_Plane_Gap_Analysis
C===========================================================================
C      This routine finds this statistical contribution of
C      each chain link to the sum dimension of the tolerance
C      chain, for lumped direction plane gaps.
C---------------------------------------------------------------------------
       Include '[Nathan.Common]Chain.Cmn'
       Include '[Nathan.Common]Tables.Cmn'
       Include '[Nathan.Common]GetParam.Cmn'
       Include '[Nathan.Common]GetParam.Inc'
C---------------------------------------------------------------------------
       If ( Magnitude .Eq. Lumped ) Then
C          ----------------------------------------------------------------
           If ( Lxb .Ge. Uxa ) Then
C              ------------------------------------------------------------
C              Clearance Lxb >= Uxa
               Mxr = (A*Cos(Theta)/2.) * (Mxb - Mxa)
               Rdx = (Abs(A*Cos(Theta))/2.) * (Txb - Txa)
               Edx = (A*Cos(Theta)/2.)*(Txb*(Ezb-0.5)-Txa*(Eza-0.5))
               Vardx = (A*Cos(Theta)**2/4.)*(Txb**2*Varzb+Txa**2*Varza)
C              ------------------------------------------------------------
           Else If ( Lxb .Le. Uxa .And. Uxb .Ge. Lxa ) Then
C              ------------------------------------------------------------
C              Transition Lxb <= Uxa and Uxb >= Lxa
               Exr = 0.5 * (Mxb - Mxa + Txb*(Ezb-0.5) - Txa*(Eza-0.5) )
               Varxr = 0.25 * (Txb**2 * Varzb + Txa**2 * Varza)
               sigm_xr = Sqrt( Varxr )
               zeta = Exr / sigma_xr
               zeta_Ex = Polynomial( zeta, NZeta1_Coef, Zeta1_Coef )
               If ( Distribution .Eq. Rectangular ) Then
               zeta_Varx = Polynomial( zeta, NZeta2_Coef, Zeta2_Coef )
               Else If ( Distribution .Eq. Normal ) Then
               zeta_Varx = Polynomial( zeta, NZeta3_Coef, Zeta3_Coef )
               End If

               Mxr = (A*Cos(Theta)/4.) * (Uxb - Lxa)
               Rdx = (Abs( A*Cos(Theta) )/2.) * (Uxb - Lxa)
               Edx = A*Cos(Theta) * zeta_Ex * sigma_xr - Mxr
               Vardx = A*Cos(Theta)**2 * zeta_Varx * Varxr
C              ------------------------------------------------------------
           Else If ( Uxb .Le. Lxa ) Then
C              ------------------------------------------------------------
C              Interference Uxb <= Lxa
C              Neglected
C              ------------------------------------------------------------
               Mxr = 0.
               Rdx = 0.
               Edx = 0.
               Vardx = 0.
C              ------------------------------------------------------------
               End If
```

APPENDIX D

COMPUTER CODE

```
      Else If ( Magnitude .Eq. Distributed ) Then
C            -----------------------------------------------------------
            If ( Distribution .Eq. Normal ) Then
C                 ------------------------------------------------------
C                 Norm Distr Center Loc
                  Exr = Ø.5 * (Mxb - Mxa + Txb*(Ezb-Ø.5) - Txa*(Eza-Ø.5) )
                  Varxr = Ø.25 * (Txb**2 * Varzb + Txa**2 * Varza)
                  Varxoxr = Varxr + Exr**2
                  Mxr = Ø.
                  Rdx = Abs( A*Cos(Theta) ) * (Uxb - Lxa)
                  Edx = Ø.
                  Vardx = (A*Cos(Theta)**2/9.) * Varxoxr
C                 ------------------------------------------------------
            Else If ( Distribution .Eq. Rectangular) Then
C                 ------------------------------------------------------
C                 Rect Distr Center Loc
                  Exr = Ø.5 * (Mxb - Mxa + Txb*(Ezb-Ø.5) - Txa*(Eza-Ø.5) )
                  Varxr = Ø.25 * (Txb**2 * Varzb + Txa**2 * Varza)
                  Varxoxr = Varxr + Exr**2
                  Mxr = Ø.
                  Rdx = Abs( A*Cos(Theta) ) * (Uxb - Lxa)
                  Edx = Ø.
                  Vardx = (A*Cos(Theta)**2/3.) * Varxoxr
C                 ------------------------------------------------------
                  End if
C            -----------------------------------------------------------
             End if
C-----------------------------------------------------------------------
      Link_Value( 3, Link_Pntr ) = Mxr
      Link_Value( 4, Link_Pntr ) = Rdx
      Link_Value( 5, Link_Pntr ) = Edx
      Link_Value( 6, Link_Pntr ) = Vardx
C-----------------------------------------------------------------------
      Return
      End
C=======================================================================
```

APPENDIX D

COMPUTER CODE

```
C===================================================================
        Subroutine DD_Plane_Gap_Analysis
C===================================================================
C       This routine finds this statistitical contribution of
C       each chain link to the sum dimension of the tolerance
C       chain, for distributed direction plane gaps.
C-------------------------------------------------------------------
        Include '[Nathan.Common]Chain.Cmn'
        Include '[Nathan.Common]GetParam.Cmn'
        Include '[Nathan.Common]GetParam.Inc'
C-------------------------------------------------------------------
        If ( Magnitude .Eq. Lumped ) Then
C           --------------------------------------------------------
            Exr = Ø.5 * (Mxb - Mxa + Txb * (Ezb-Ø.5) - Txa * (Eza-Ø.5) )
            Varxr = Ø.25 * (Txb**2 * Varzb + Txa**2 * Varza)
            Varxoxr = Varxr + Exr**2
            Mxr = Ø.
            Rdx = Abs( A ) * (Uxb - Lxa)
            Edx = Ø.
            Vardx = (A**2/2.) * Varxoxr
C           --------------------------------------------------------
        Else If ( Magnitude .Eq. Distributed ) Then
C           --------------------------------------------------------
            Exr = Ø.5 * (Mxb - Mxa + Txb * (Ezb-Ø.5) - Txa * (Eza-Ø.5) )
            Varxr = Ø.25 * (Txb**2 * Varzb + Txa**2 * Varza)
            Varxoxr = Varxr + Exr**2
            Mxr = Ø.
            Rdx = Abs( A ) * (Uxb - Lxa)
            Edx = Ø.
            Vardx = (A**2/4.) * Varxoxr
C           --------------------------------------------------------
            End if
C-------------------------------------------------------------------
        Link_Value( 3, Link_Pntr ) = Mxr
        Link_Value( 4, Link_Pntr ) = Rdx
        Link_Value( 5, Link_Pntr ) = Edx
        Link_Value( 6, Link_Pntr ) = Vardx
C-------------------------------------------------------------------
        Return
        End
C===================================================================
```

APPENDIX D

COMPUTER CODE

```
C=====================================================================
        Subroutine Conf( Middle_Sum_Dim_Range,
        1                Range_Sum_Dim,
        2                Expectation_Sum_Dim,
        3                Variance_Sum_Dim,
        4                Middle_Sum_Dim_Tol,
        5                Tolerance_Sum_Dim,
        6                Confidence_Level )
C---------------------------------------------------------------------
C       This routine finds the confidence level from a given
C       set of statistcal parameters for the sum dimension.
C       Method - integrate the area under the probability
C       function using tolerance of the sum dIMension as the
C       limits of integration.
C-------Routines required - LnGammaFunction, Dintegrate
C=====================================================================
        Real    Middle_Sum_Dim_Range,
        1       Middle_Sum_Dim_Tol
        Double Precision DLnGG, DLnGE, DLnGGE
        Double Precision Beta, Const, Gamma, Eta, AA, BB,
        1                XLiml, Xlim2, Conf_Area, Whole_Area
        Common /Beta_Param/ Const, Gamma, Eta, AA, BB
        External        Beta
C---------------------------------------------------------------------
        AA = Middle_Sum_Dim_Range - Range_Sum_Dim / 2.
        If ( AA .Le. 1. ) Then
            Middle_Sum_Dim_Tol = Middle_Sum_Dim_Tol -
        1        ( Middle_Sum_Dim_Range - Range_Sum_Dim ) + 1.
            Middle_Sum_Dim_Range = Range_Sum_Dim +1.
            End If
        AA = Middle_Sum_Dim_Range - Range_Sum_Dim / 2.
        BB = Middle_Sum_Dim_Range + Range_Sum_Dim / 2.
        EX = Expectation_Sum_Dim + Middle_Sum_Dim_Tol

        Gamma = ( (EX-AA)**2 * (BB-EX) - Variance_Sum_Dim * (EX-AA) )
        1       / ( Variance_Sum_Dim * Range_Sum_Dim )
        ETA = ( (EX-AA) * (BB-EX)**2 - Variance_Sum_Dim * (BB-EX) )
        1       / ( Variance_Sum_Dim * Range_Sum_Dim )

        Call DLnGammaFunction( Gamma , DLnGG, IER )
        Call DLnGammaFunction( Eta, DLnGE, IER )
        Call DLnGammaFunction(  (Gamma +ETA) , DLnGGE, IER )

        Const = DLnGGE - DLnGG - DLnGE
```

APPENDIX D

COMPUTER CODE

```
      XLim1 = Middle_Sum_Dim_Tol - Tolerance_Sum_Dim / 2.
      IF ( XLim1 .LT. (AA*1.00001D0) ) XLim1 = AA * 1.00001D0

      XLim2 = Middle_Sum_Dim_Tol + Tolerance_Sum_Dim / 2.
      IF ( XLim2 .GT. (BB*0.99999D0) ) XLim2 = BB * 0.99999D0

      Call Dintegrate( XLim1, XLim2, Integ_Step, Beta, Conf_Area )

      XLim1 = AA * 1.00001D0
      XLim2 = BB * 0.99999D0

      Confidence_Level = Conf_Area / Whole_Area * 1.0D+2
      Return
      End
C=============================================================================
      Function Beta( X )

      Double Precision Beta, Const, Gamma, Eta, AA, BB, ZZ
      Common /Beta_Param/ Const, Gamma, Eta, AA, BB

      Beta = Exp( Const + (Gamma-1.0D0)*Log((X-AA)/(BB-AA))
     1                   + (Eta-1.0D0)*Log(1.-(X-AA)/(BB-AA)) )

      Return
      End
C=============================================================================
```

APPENDIX D

COMPUTER CODE

```
C=====================================================================
      Subroutine DLnGammaFunction( X, DLnG, Ier )
C=====================================================================
      Double Precision X, Xo, DLnG, DTabGamma
      Integer N, Ier
C-------Approximation to tabulated values of the gamma function
C-------for arguments 1 < Xo <= 2
      DTabGamma( Xo ) = 1.ØDØ - Ø.57710166DØ * (Xo-1.ØDØ)
      1                       + Ø.98585399DØ * (Xo-1.ØDØ)**2
      2                       - Ø.87642182DØ * (Xo-1.ØDØ)**3
      3                       + Ø.83282120DØ * (Xo-1.ØDØ)**4
      4                       - Ø.56847290DØ * (Xo-1.ØDØ)**5
      5                       + Ø.25482049DØ * (Xo-1.ØDØ)**6
      6                       - Ø.05149930DØ * (Xo-1.ØDØ)**7
C---------------------------------------------------------------------
      Ier = Ø
      If ( X .Gt. 1.ØD+38 ) Then
              Ier = 2
      Else If ( X .Ge. 1.ØDØ ) Then
              N = IdInt( X ) - 1
              Xo = Mod( X, 1.ØDØ ) + 1.ØDØ
              DLnG = Log( DTabGamma( Xo ) )
              Do 1Ø, k=Ø, (N-1)
1Ø                    DLnG = DLnG + Log( k + Xo )
      Else If ( X .Le. Ø.ØDØ ) Then
              Ier = 1
      Else
              DLnG = Log( DTabGamma( X + 1.ØDØ ) / X )
      End If
C---------------------------------------------------------------------
      Return
      End
C=====================================================================

C=====================================================================
      Subroutine DIntegrate( a, b, n, Function, Area )
C------- Simpson's numeric Integration
C=====================================================================
      Double Precision a, b, h, Coef, Function, Area
      Coef( k ) = 2 + 2 * Mod( k, 2 )
C---------------------------------------------------------------------
      h = ( b - a ) / n
      Area = Function( a ) + Function( b )
      Do 1Ø k=1, ( n - 1 )
              Area = Area + Coef( k ) * Function( a + k * h )
1Ø            Continue
      Area = ( h / 3.ØDØ ) * Area
C---------------------------------------------------------------------
      Return
      End
C=====================================================================
```