GUIA DE APLICACIÓN DE UNA METODOLOGIA PARA ASIGNAR TOLERANCIAS A UN CONJUNTO DE VARIABLES CORRELADAS

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE INGENIERÍA MECÁNICA

Ingeniería Técnica Industrial: Mecánica

Julio 2010

Autor:
Elena Mendoza Melcón.

Tutor:
Isabel Marina Gonzalez Farias.

Capítulo 1

INTRODUCCIÓN

CAPITULO 1. INTRODUCCIÓN

1.1.INTRODUCCIÓN AL PROYECTO

El concepto de tolerancia, propio de la metrología industrial, se aplica a la fabricación de piezas en serie. Dada una magnitud significativa y cuantificable propia de un producto industrial (sea alguna de sus dimensiones, su resistencia, su peso, etc.), el margen de tolerancia es el intervalo de valores en el que debe encontrarse dicha magnitud para que se acepte como válida, lo que determina la aceptación o el rechazo de los componentes fabricados, según sus valores queden dentro o fuera de ese intervalo.

El propósito de los intervalos de tolerancia es el de admitir un margen para las imperfecciones en la manufactura del componente, ya que se considera imposible la precisión absoluta desde el punto de vista técnico, o bien no se recomienda por motivos de eficiencia o costes de fabricación.

La tolerancia puede ser especificada por un rango explícito de valores permitidos, una máxima desviación de un valor nominal, o por un factor o porcentaje de un valor nominal. A lo largo de este trabajo expresaremos las tolerancias como máximas desviaciones de ciertos valores nominales. Por ejemplo, si la longitud aceptable de una barra de acero está en el intervalo 1m±0.01m, la tolerancia es de 0.01m.

El aumento de la competencia en el área de fabricación ha propiciado a lo largo de los últimos años un mayor interés por encontrar nuevos métodos que permitan realizar un diseño óptimo de las tolerancias de las piezas con el objetivo de conseguir una reducción de los costes, una mejora de la calidad del producto y la reducción del tiempo entre el diseño y el producto final.

En la actualidad, estos factores han contribuido a aumentar el interés por el diseño de tolerancias, considerando este procedimiento como una parte importante en el ciclo de vida del producto y como una forma de mejorar su calidad.

Los procesos clásicos del diseño de tolerancias están basados en la suposición de variables independientes, esta suposición puede no ser realista ya que, a menudo las diferentes variables de las que consta el proceso de fabricación están relacionadas entre sí.

En este proyecto se presenta una metodología general para la asignación de tolerancias estadísticas óptimas a variables dependientes donde la estructura de la dependencia se estima desde el proceso de producción.

El método está basado en la suposición de que la variabilidad del proceso multivariante es consecuencia de una serie de factores independientes.

De ahí, la asignación de tolerancias puede estar basada en las propiedades estadísticas de esos factores.

El diseño de tolerancias coge la independencia de los factores como una restricción y las tolerancias son asignadas de manera óptima de acuerdo a su variabilidad.

Para la aplicación del método se requiere la utilización de un programa matemático que facilite la ejecución de los cálculos estadísticos. Este programa puede ser escrito por el propio usuario dentro de un programa comercial, como por ejemplo, Matlab donde se podrán obtener paso a paso cada una de las iteraciones necesarias.

1.2. OBJETIVOS.

El objetivo de este proyecto es servir como guía en la aplicación de un método innovador para la asignación de tolerancias estadísticas óptimas a un proceso de fabricación para cualquier persona, incluso aquellas que no hayan tenido contacto con los métodos estadísticos para el diseño de tolerancias.

Este documento pretende hacer llegar de forma simplificada los pasos necesarios para llevar a cabo el diseño de tolerancias considerando correlación entre las variables y las ventajas que tiene este método con relación a los métodos tradicionales de asignación de tolerancias, mediante la comparación de casos prácticos que sirvan de ejemplo y ayuda para la comprensión de la metodología.

Por otro lado se presentara de manera resumida el desarrollo secuencial del proceso dentro de una aplicación matemática mediante el programa Matlab, donde se podrán identificar los pasos generales necesarios para la obtención de las tolerancias y su optimización.

El proyecto finalizará con tres aplicaciones prácticas de la metodología descrita, uno para el caso en el que la correlación entre las variables es lineal y dos en las que la correlación es no lineal, donde se pretende aplicar el método de una forma más real desarrollando cada caso forma reducida, desde los datos iniciales del cada problema hasta la optimización de las tolerancias.

1.3.ESTRUCTURA DEL PROYECTO.

El proyecto se divide en 5 capítulos, y un anexo en el que se incluyen dos artículos relacionados con el diseño de tolerancias considerando correlación entre las variables.

El capítulo I nos introduce al proyecto, y describe los objetivos que este persigue.

El capítulo II nos resume el recorrido histórico de la evolución de las tolerancias, nos introduce una serie de conceptos teóricos que será necesario conocer para el desarrollo de la metodología y nos describe el estado actual del diseño de tolerancias, centrándose en los métodos que consideran independencia de variables.

En el capítulo III se describirá la metodología propuesta en el proyecto para el diseño óptimo de tolerancias de manera teórica definiendo cada paso que se debe realizar para llegar a una solución óptima.

En el capítulo IV se desarrollan tres ejemplos de aplicación de la metodología propuesta en el proyecto, pasando por la definición del problema, la obtención de las tolerancias óptimas de fabricación siguiendo la secuencia teórica que se desarrollará en el capitulo III y por ultimo se resolverá el problema, y se calcularan las tolerancias óptimas de cada ejemplo utilizando la herramienta de cálculo Matlab

En el capítulo V se desarrollarán las conclusiones obtenidas así como las nuevas líneas de investigación a seguir.

Para finalizar se dará la información correspondiente a la bibliografía consultada y al conjunto de anexos incluidos para una mejor comprensión.

Capítulo 2

ESTADO ACTUAL DEL DISEÑO DE TOLERANCIAS

CAPITULO 2: CONCEPTOS TEÓRICOS IMPORTANTES Y ESTADO DEL ARTE DEL DISEÑO DE TOLERANCIAS

En este capítulo se realizara a una breve introducción al concepto de tolerancias, analizando sus diferentes tipos y su uso en el proceso de diseño y fabricación de piezas, además se explicaran brevemente algunos conceptos estadísticos relacionados con la metodología que se va a definir a lo largo del proyecto, por ultimo se explicaran brevemente los métodos que se usan actualmente para el diseño optimo de tolerancias utilizando variables independientes.

2.1. DEFINICIÓN DE TOLERANCIAS. PAPEL DE LAS TOLERANCIAS EN LA FABRICACIÓN

La aparición del concepto de tolerancia en la historia viene ligada a la época de la revolución industrial, esta época se vio marcada por la aparición de la producción en masa y la invención de la intercambiabilidad de las partes, lo que a su vez supuso la introducción del concepto de fabricación con tolerancias.

La intercambiabilidad permite la fabricación de partes para su posterior ensamblaje, sin necesidad de ajuste a la forma funcional global del producto. Este concepto se centró en ser fiable y adecuado para el montaje de las piezas, y agilizar la fabricación y reparación.

Para que un elemento mecánico funcione correctamente, es necesario que las distintas piezas que lo forman estén acopladas entre sí, en condiciones bien determinadas.

El concepto de tolerancia colaboró con la función de "dividir y unir" en producción, en la que cada parte podría ser concebida, diseñada y fabricada individualmente manteniendo los niveles de tolerancias para encajar dentro del ensamblaje sin comprometer la funcionalidad del sistema. En vista de este concepto, piezas intercambiables dieron lugar a la proliferación de la división del trabajo y la producción en masa. Esto provocó la separación entre el diseño y la fabricación. (Por lo tanto, las tolerancias se convirtieron en el principal puente de comunicación entre diseño, fabricación, e inspección).

En los procesos de fabricación, a la hora de determinar el tamaño y la forma de una pieza, observamos que existen diferencias entre las dimensiones reales de la pieza y los valores iniciales que habíamos indicado inicialmente durante el proceso de diseño, la pieza tendrá tanto mayor calidad cuanto mas se aproxime a esas indicaciones en las cotas de diseño, es fácil entender que será mas difícil y costoso aproximarse a una menor desviación, es decir, a una mayor calidad.

Conseguir que una pieza tenga una dimensión exactamente igual, con infinitas cifras decimales que su cota de diseño es imposible, entre otras causas porque ningún aparato de medida puede tener esa precisión. La experiencia demuestra que no es posible construir una pieza cuyas cotas sean exactamente iguales a las cotas que señala el plano, en la Figura 2.1 se muestra un ejemplo de las diferencias entre las cotas de diseño y las cotas obtenidas después de la fabricación.

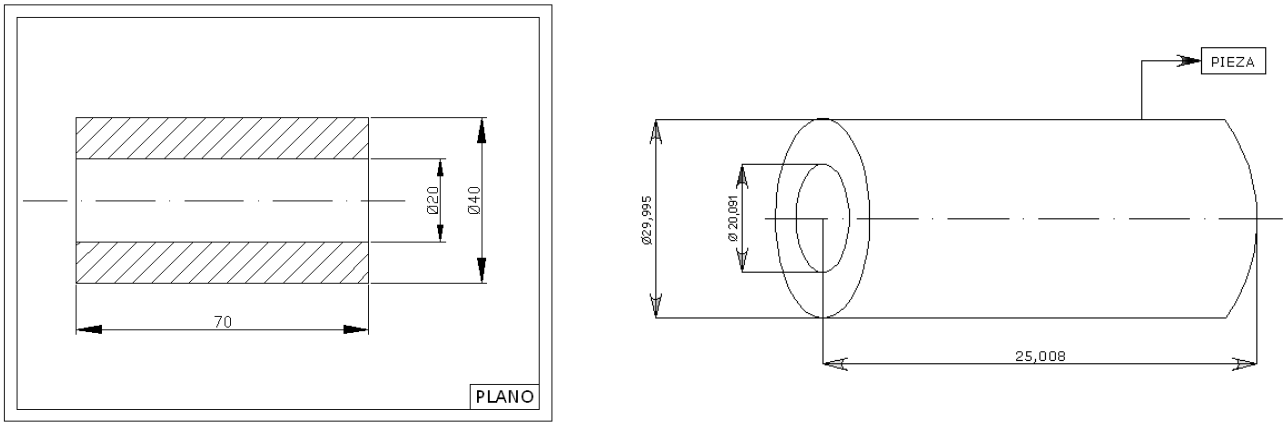


Figura 2.1. Diferencia entre cotas de diseño y cotas obtenidas en la pieza.

Esta imposibilidad de poder obtener una cota exacta, es debida a las causas siguientes:

- Errores cometidos por el aparato de medida;
- Errores e incertidumbres debidas al operario;
- Errores debidos a deformaciones mecánicas
- Errores debidos a dilataciones térmicas
- Errores debidos a la falta de precisión de la máquina.

Cuanto más esmerada sea la fabricación, empleando aparatos de medida y máquinas de más precisión, temperatura ideal de 20ºC, etc., menor será la diferencia entre las cotas reales de la pieza mecanizada y las del plano, pero de todas formas siempre se cometerá un pequeño error en la obtención de una cota determinada. Por otro lado, mientras más pequeñas sea la diferencia mayor será el coste de fabricación.

Este problema, para el que a simple vista podríamos no encontrar una solución fácil, se ve solucionado si controlamos las diferencias entre las cotas nominales o de diseño y las medidas reales que obtenemos en el proceso de fabricación.

Para ello debemos acotar de nuevo las dimensiones de la pieza y establecer unos mínimos y máximos permitidos. Podemos decir, entonces que la tolerancia es el error admitido en una cota, distinguiendo tolerancias dimensionales y tolerancias de forma que afectan a las cotas de su mismo nombre.

De esta manera se garantiza la intercambiabilidad aceptable de las piezas y el correcto montaje de elementos fabricados en serie a los cuales se les ha controlado la tolerancia en sus cotas. Será necesario un nivel mínimo de calidad para los dos elementos en contacto o simplemente relacionados.

2.2. TIPOS DE TOLERANCIAS

2.2.1. TOLERANCIAS DIMENSIONALES

La forma mas sencilla de expresar la tolerancia admitida es escribir a continuación de la cota los límites máximo y mínimo que se admiten respecto de ese valor, por ejemplo en el cilindro de la Figura 2.2, la generatriz podrá tener una longitud entre 29,07 y 30,05 mientras que el diámetro de la base podrá admitirse entre 10,05 y 10,09. Nótese que este último caso tanto el valor máximo como el mínimo quedan por encima del valor nominal de la cota. Llamaremos dimensión nominal a aquella que indica la cota. En este ejemplo seria dimensiones nominales 30 y 10 mm para la altura y el diámetro. La dimensión efectiva de una pieza ya construida será la que midamos. Las dimensiones máxima y mínima serán los valores máximo y mínimo admitidos de la dimensión efectiva. El intervalo de tolerancia IT será la diferencia entre las dimensiones máxima y mínima.

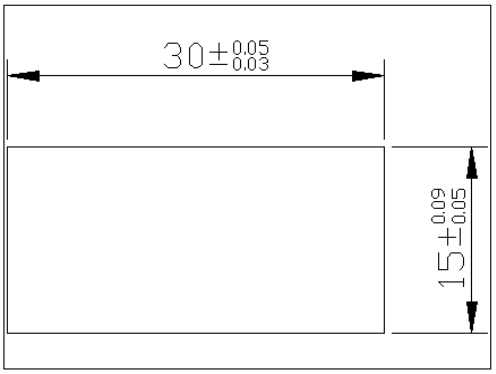


Figura 2.2.Ejemplo de tolerancia dimensional

Cuando coinciden los márgenes por encima y por debajo, es decir la cota esta centrada en su margen de tolerancia, se puede expresar con el signo ± seguido del margen, que en este caso coincide con IT/2. Se muestra un ejemplo en la Figura 2.3.

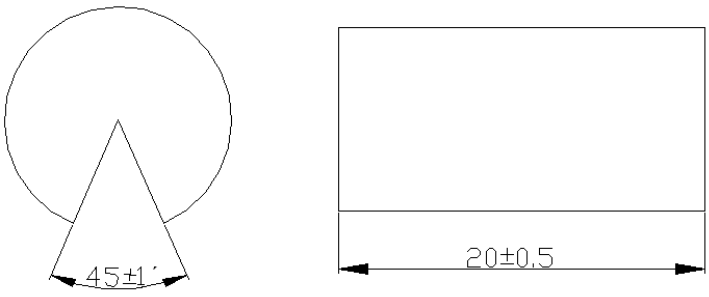


Figura 2.3. Ejemplos de tolerancias dimensionales con cota centrada en el margen de tolerancia

En resumen, las tolerancias dimensionales controlan medidas o dimensiones de una pieza, únicamente las medidas locales reales (distancias entre puntos). No controlan ni la forma, ni la posición, ni la orientación que tengan los elementos a los que se aplica la tolerancia dimensional.

2.2.2. TOLERANCIAS GEOMÉTRICAS

De igual forma que las dimensiones de una pieza presentan errores que limitamos estableciendo tolerancias dimensionales, también la propia geometría de la pieza presentará desviaciones respecto de la que idealmente de representa. Estas desviaciones se controlan mediante tolerancias geométricas, donde podemos distinguir diferentes tipos como son las tolerancias de forma, de orientación, de posición y de oscilación y de forma.

En determinadas ocasiones, como por ejemplo: mecanismos muy precisos, piezas de grandes dimensiones, etc., la especificación de tolerancias dimensionales puede no ser suficiente para asegurar un correcto montaje y funcionamiento de los mecanismos.

La Figura 2.4. muestra tres casos donde una de las piezas puede ser correcta desde el punto de vista dimensional (diámetros de las secciones dentro de tolerancia) y no ser apta para el montaje: en el primer caso tendríamos un defecto de rectitud, en el segundo caso tendríamos un defecto de coaxialidad, y en el tercer caso tendríamos un defecto de perpendicularidad. Vemos, pues, que en la fabricación se producen irregularidades geométricas que pueden afectar a la forma, posición y orientación de los diferentes elementos constructivos de las piezas.

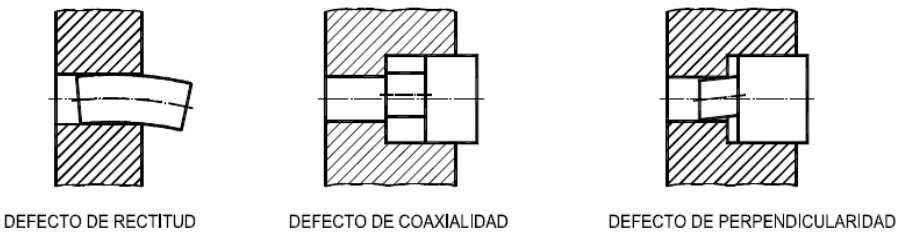


Figura 2.4. Ejemplos de piezas con tolerancias dimensionales correctas no aptas debido a tolerancias geométricas

Una tolerancia dimensional aplicada a una medida ejerce algún grado de control sobre desviaciones geométricas, por ejemplo: la tolerancia dimensional tiene efecto sobre el paralelismo y la planicidad. Sin embargo, en algunas ocasiones la tolerancia de medida no limita suficientemente las desviaciones geométricas; por tanto, en estos casos se deberá especificar expresamente una tolerancia geométrica, teniendo prioridad sobre el control geométrico que ya lleva implícita la tolerancia dimensional.

Podríamos definir la tolerancia geométrica de un elemento de una pieza (superficie, eje, plano de simetría, etc.) como la zona de tolerancia dentro de la cual debe estar contenido dicho elemento. Dentro de la zona de tolerancia el elemento puede tener cualquier forma u orientación, salvo si se da alguna indicación más restrictiva.

La metodología que se va a desarrollar en este trabajo se basa en tolerancias dimensionales. Sin embargo, para tener una idea mas concreta de cómo pueden afectar las tolerancias geométricas en el diseño y la fabricación de las piezas se presentan a continuación unos ejemplos.

Las tolerancias de forma establecen para un determinado elemento geométrico dos elementos similares desplazados una cierta distancia, el elemento geométrico real será admitido siempre y cuando se encuentre entre estos dos modelos.

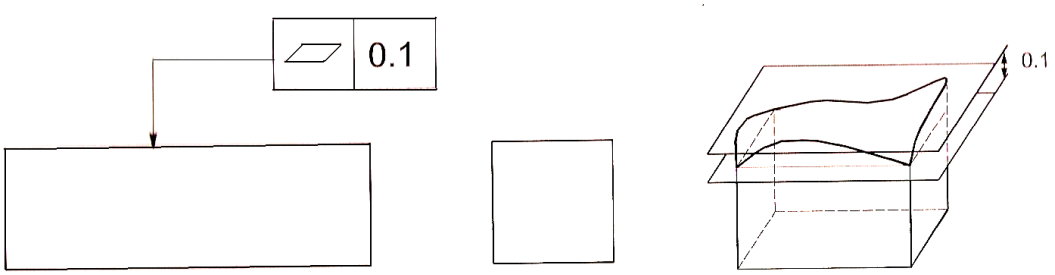


Figura 2.5. Ejemplo de tolerancia geométrica de forma

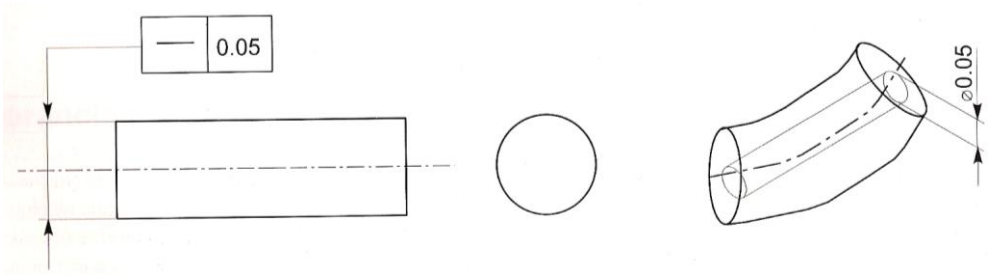


Figura 2.6. Ejemplo de tolerancia geométrica de forma en eje de simetría

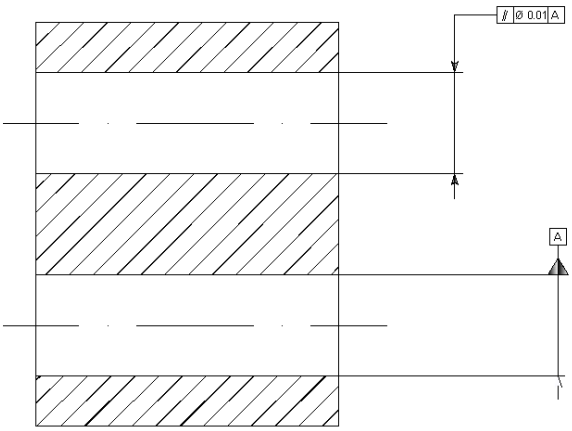


Figura 2.7. Ejemplo de tolerancia geométrica de orientación

Las tolerancias de orientación son más exigentes que las tolerancias de forma, así, en la Figura 2.7 se establece el paralelismo entre los ejes de los dos taladros, siendo el de referencia el de abajo y controlándose el de arriba. Ø

Las tolerancias de posición, como su propio nombre indica, afectan a la situación de los elementos, así por ejemplo en la Figura 2.8. se indica la coaxialidad de ejes mediante un símbolo compuesto de dos circunferencias concéntricas.

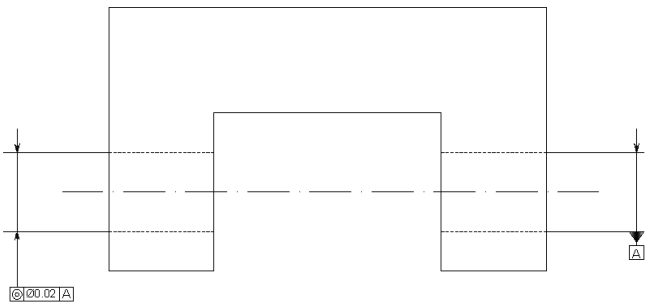


Figura 2.8. Ejemplo de tolerancia geométrica de posición

Las tolerancias de oscilación están relacionadas con el error admitido en el valor del radio de `piezas cilíndricas cuando hacemos girar estas. La tolerancia circular radial establece que el radio en una sección debe permanecer siempre entre dos valores que no difieran más del valor indicado cuando giramos en torno al eje de referencia. En la Figura 2.9 y 2.10 se representan este tipo de tolerancia para la superficie cilíndrica central.

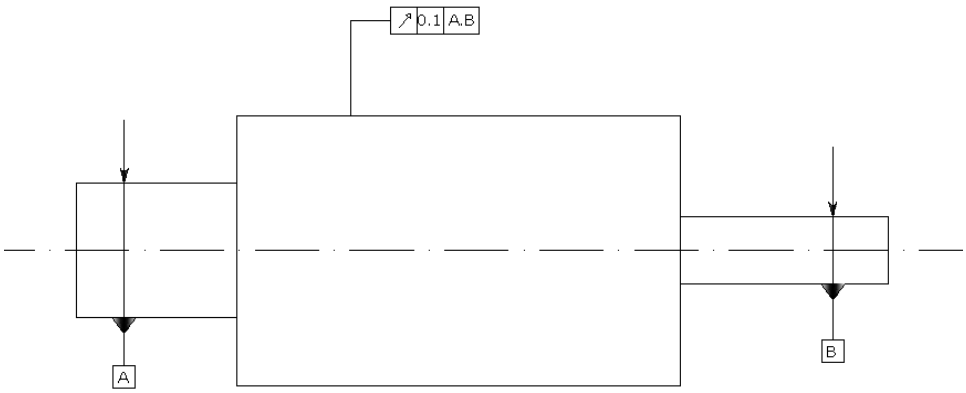


Figura 2.9. Ejemplo de tolerancia geométrica de oscilación circular radial

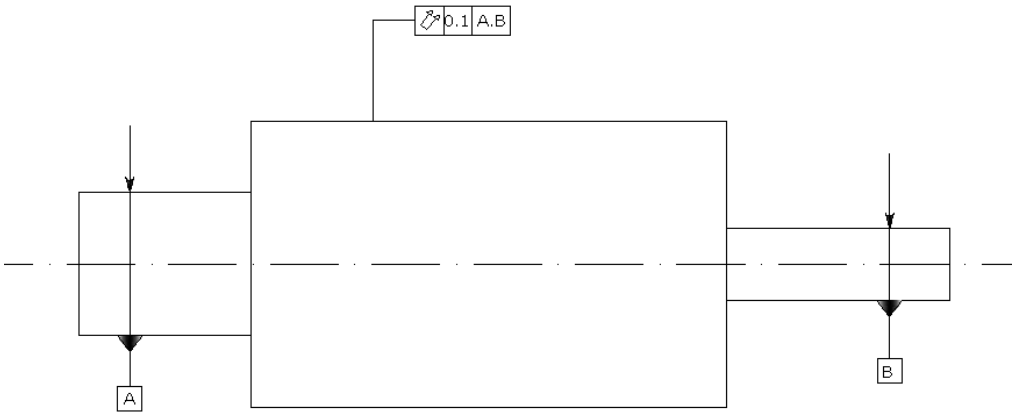


Figura 2.10. Ejemplo de tolerancia geométrica de oscilación total radial.±

Como se mencionó antes, en este proyecto se propondrá una metodología para la asignación óptima de tolerancias dimensionales. El problema de la asignación de tolerancias geométricas no es tratado aquí.

2.3. CONCEPTOS ESTADÍSTICOS RELACIONADOS CON LA METODOLOGIA

A continuación se describen algunos conceptos estadísticos importantes, que serán utilizados en el desarrollo de la metodología para la asignación de tolerancias, que se propone en este proyecto.

2.3.1. LA MATRIZ DE DATOS

Supondremos, en adelante, que hemos observado *p* variables numéricas en un conjunto de *n* elementos, cada una de estas variables se denomina variable escalar o univariante y el conjunto de las *p* variables forman una variable vectorial o multivariante. Los valores de las *p* variables univariantes en cada uno de los *n* elementos pueden representarse en una matriz, **X,** de dimensiones *(nxp*) que llamaremos matriz de datos. Denotando por x_{ij} al elemento genérico de esa matriz, que representa el valor de la variable *j* sobre el individuo *i*. Es decir:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \dots & \dots & & \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}$$

Donde i=1,…, n representa el individuo u observación
j=1,…, p representa la variable

2.3.2. EL VECTOR DE MEDIAS

La medida de centralización más utilizada para describir datos multivariantes es el vector de medias, de dimensión *p* cuyas componentes son las medias de cada una de las *p* variables.
Puede calcularse como en el caso univariante, promediando las medias de cada elemento, que ahora son vectores:

$$\overline{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^{n} x_i = \begin{bmatrix} \overline{x}_1 \\ \overline{x}_2 \\ \dots \\ \overline{x}_p \end{bmatrix}$$

2.3.3. LA MATRIZ DE VARIANZA Y COVARIANZAS

Normalmente, para una variable univariante, la variabilidad respecto a la media se mide habitualmente por la varianza, o su raíz cuadrada, la desviación típica, además, la relación lineal entre dos variables se mide por la covarianza.
La matriz de varianzas y covarianzas es una matriz cuadrada y simétrica que contiene en la diagonal las varianzas y fuera de la diagonal las covarianzas entre las variables. Para simplificar llamaremos a esta matriz matriz de covarianzas y la designaremos con Σ

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \dots & \sigma_{1p}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \dots & \sigma_{2p}^2 \\ \dots & & & \\ \sigma_{p1}^2 & \sigma_{p2}^2 & \dots & \sigma_{pp}^2 \end{pmatrix}$$

2.3.4. EL ÍNDICE DE CAPACIDAD

El índice de capacidad de un proceso C_{pk}, también denominado ratio de capacidad del proceso, es un cálculo estadístico sobre la capacidad de un proceso, se define capacidad del proceso al estudio realizado sobre la variabilidad del mismo y que nos permite determinar en que medida el proceso cumple con las expectativas para las que fue diseñado, en el caso de un proceso de fabricación se espera que el resultado del proceso cumpla con los requerimiento o las tolerancias con las que fue diseñado, calcular la capacidad de un proceso es una manera de producir un resultado medible de la variabilidad de un proceso, la capacidad del proceso se mide a través del índice de capacidad.
En resumen, la capacidad de un proceso es la capacidad que tiene el mismo para producir un resultado dentro de unos límites predefinidos (LTS, límite de tolerancias superior y LTI límite de tolerancia inferior).

El índice de capacidad juega un papel fundamental en los procesos de producción a la hora de demostrar que un proceso es fiable y esta bajo control. Los índices de capacidad que aceptados de forma general para el control de los procesos de fabricación son los siguientes, donde (LTS, LTI) son los límites de tolerancia superior e inferior de l especificación, T es la media objetivo que se busca en el proceso, $\hat{\mu}$ es la media estimada del proceso y $\hat{\sigma}$ es la variabilidad estimada del proceso:

INDICE	DESCRIPCÓN
$\hat{C}_p = \frac{LTS - LTI}{6 \times \hat{\sigma}}$	Calcula lo que el proceso seria capaz de producir si el proceso estuviera centrado. Presupone que el resultado del proceso sigue una distribución normal
$\hat{C}_{p,\inf} = \frac{\hat{\mu} - LTI}{3 \times \hat{\sigma}}$	Calcula la capacidad del proceso con especificaciones únicamente con un límite inferior. Presupone que el resultado del proceso está aproximadamente distribuido de forma normal
$\hat{C}_{p,\sup} = \frac{LTS - \hat{\mu}}{3 \times \hat{\sigma}}$	Calcula la capacidad del proceso con especificaciones únicamente con un límite superior Presupone que el resultado del proceso está aproximadamente distribuido de forma normal.
$\hat{C}_{pk} = \min\left[\frac{LTS - \hat{\mu}}{3 \times \hat{\sigma}}, \frac{\hat{\mu} - LTI}{3 \times \hat{\sigma}}\right]$	Calcula lo que el proceso es capaz de producir si el objetivo del proceso esta centrado entre los límites de la especificación. En caso de que la media del proceso no este centrada, $\hat{C}_p$ sobreestima la capacidad del proceso $\hat{C}_{pk} < 0$ si la media del proceso se sitúa fuera de los límites de especificación. Presupone que el resultado del proceso está aproximadamente distribuido de forma normal
$\hat{C}_{pm} = \frac{\hat{C}_p}{\sqrt{1 + \left(\frac{\hat{\mu} - T}{\hat{\sigma}}\right)^2}}$	Calcula la capacidad del proceso respecto a un objetivo, T. $\hat{C}_{pm}$ es siempre mayor que cero. Presupone que el resultado del proceso está aproximadamente distribuido de forma normal
$\hat{C}_{pkm} = \frac{\hat{C}_{pk}}{\sqrt{1 + \left(\frac{\hat{\mu} - T}{\hat{\sigma}}\right)^2}}$	Calcula la capacidad del proceso respecto a un objetivo, T válido para un proceso con una media descentrada. Presupone que el resultado del proceso esta aproximadamente distribuido de forma normal.

Tabla 2.1. Descripción de los diferentes tipos de índices de capacidad

A lo largo de este proyecto se utilizara el índice de capacidad más general, $\hat{C}_p$ ya que el método a desarrollar se presupondrá centrado en la media, para este caso el índice de capacidad podría asumir varios valores que los analistas clasifican entre valor 1 y valor 4 según sea la habilidad del proceso para cumplir con las especificaciones:

$\hat{C}_p$	CLASE DE PROCESO	DESCISÓN
ICP>1.33	1	Más que adecuado, incluso puede exigirse más en términos de su capacidad
1<ICP<1.33	2	Adecuado para lo que fue diseñado. Requiere control estrecho si se acerca al valor de 1
0.67<ICP<1	3	No es adecuado para cumplir con el diseño inicial
ICP<0.67	4	No es adecuado para cumplir con el diseño inicial

Tabla 2.2. Decisiones para tipos de procesos en función del valor de su índice de capacidad

2.3.5. DISTRIBUCIÓN NORMAL

Al iniciar el análisis estadístico de una serie de datos, y después de la etapa de detección y corrección de errores, un primer paso consiste en describir la distribución de las variables estudiadas y, en particular, de los datos numéricos. Además de las medidas descriptivas correspondientes, el comportamiento de estas variables puede explorarse gráficamente de un modo muy simple.

Una de las distribuciones teóricas mejor estudiadas en los textos de estadística y más utilizada en la práctica es la distribución normal, también llamada distribución gaussiana. Su importancia se debe fundamentalmente a la frecuencia con la que distintas variables asociadas a fenómenos reales y cotidianos siguen, aproximadamente, esta distribución.

El uso extendido de la distribución normal en las aplicaciones estadísticas puede explicarse, además, por otras razones. Muchos de los procedimientos estadísticos habitualmente utilizados asumen la normalidad de los datos observados. Aunque muchas de estas técnicas no son demasiado sensibles a desviaciones de la normal y, en general, esta hipótesis puede obviarse cuando se dispone de un número suficiente de datos, resulta recomendable contrastar siempre si se puede asumir o no una distribución normal. La simple exploración visual de los datos puede sugerir la forma de su distribución. No obstante, existen otras medidas, gráficos de normalidad y contrastes de hipótesis que pueden ayudarnos a decidir, de un modo más riguroso, si la muestra de la que se dispone procede o no de una distribución normal. Cuando los datos no sean normales, podremos o bien transformarlos o emplear otros métodos estadísticos que no exijan este tipo de restricciones.

A continuación se describirá la distribución normal, su ecuación matemática y sus propiedades más relevantes.

La distribución de una variable normal está completamente determinada por dos parámetros, su media μ y su desviación típica σ, Con esta notación, la densidad de la normal viene dada por la ecuación:

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left\{\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\};-\infty<x<\infty \qquad (2.1)$$

Esta ecuación determina la curva en forma de campana que define una distribución normal y que podemos ver el la Figura 2.11. Así, se dice que una característica X sigue una distribución normal de media μ y varianza σ^2, y se denota como $X\approx N(\mu,\sigma)$, si su función de densidad viene dada por (2.1)

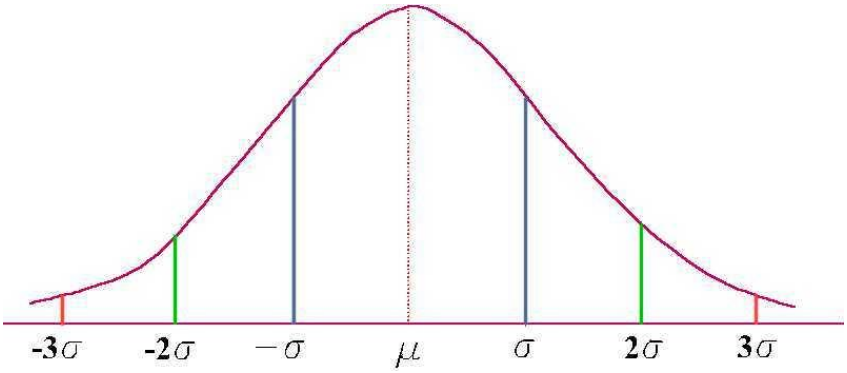


Figura 2.11.. Representación gráfica de una distribución normal

La distribución normal posee ciertas propiedades importantes que conviene destacar:

1. Tiene una única moda, que coincide con su media y su mediana.

2. La curva normal es asintótica al eje de abscisas. Por ello, cualquier valor entre $-\infty$ y ∞ es teóricamente posible. El área total bajo la curva es, por tanto, igual a 1.

3. Es simétrica con respecto a su media μ. Según esto, para este tipo de variables existe una probabilidad de un 50% de observar un dato mayor que la media, y un 50% de observar un dato menor.

4. La distancia entre la línea trazada en la media y el punto de inflexión de la curva es igual a una desviación típica σ. Cuanto mayor sea la desviación típica , más aplanada será la curva de la densidad.

5. El área bajo la curva comprendida entre los valores situados aproximadamente a dos desviaciones típicas de la media es igual a 0.95. En concreto, existe un 95% de posibilidades de observar un valor comprendido en el intervalo $(\mu - 1.96\sigma, \mu + 1.96\sigma)$

6. La forma de la campana de Gauss depende de los parámetros μ y σ. La media indica la posición de la campana, de modo que para diferentes valores de μ la gráfica es desplazada a lo largo del eje horizontal. Por otra parte, la desviación estándar determina el grado de apuntamiento de la curva. Cuanto mayor sea el valor de σ , más se dispersarán los datos en torno a la media y la curva será más plana.

Un valor pequeño de este parámetro indica, por tanto, una gran probabilidad de obtener datos cercanos al valor medio de la distribución.

7. Como se deduce del apartado anterior, no existe una única distribución normal, sino una familia de distribuciones con una forma común, diferenciadas por los valores de su media y su varianza. De entre todas ellas, la más utilizada es la distribución normal estándar, que corresponde a una distribución de media 0 y varianza 1.

A lo largo del proyecto, tal y como veremos en apartados posteriores se va a considerar que las variables que se utilizan en el desarrollo del método siguen una distribución normal, pero estas variables no son variables marginales, si no que están relacionadas unas con otras, por lo que a continuación se describe brevemente la distribución normal multivariante, que es una generalización de la distribución normal unidimensional descrita anteriormente a dimensiones superiores.

2.3.5.1. Distribución normal multivariante.

En probabilidad y estadística, una distribución normal multivariante, también llamada distribución gaussiana multivariante, es una generalización de la distribución normal unidimensional a dimensiones superiores.

Un vector aleatorio $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$ sigue una distribución normal multivariante de vector de medias: $\overline{\mathbf{X}} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \dots \\ \mu_p \end{bmatrix}$ y de matriz de covarianzas $\mathbf{\Sigma} = \begin{pmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \dots & \sigma_{1p}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \dots & \sigma_{2p}^2 \\ \dots & & & \\ \sigma_{p1}^2 & \sigma_{p2}^2 & \dots & \sigma_{pp}^2 \end{pmatrix}$

Lo que se expresa como $X \approx N_K(\overline{\mathbf{X}}, \mathbf{\Sigma})$, su función de densidad conjunta se describe mediante la siguiente ecuación:

$$f(\mathbf{X}) = f(x_1, x_2, \dots x_n) = \frac{1}{(2\pi)^{\frac{n}{2}} |\mathbf{\Sigma}|^{\frac{1}{2}}} \exp\left(\frac{-1}{2} (x-\mu)' \mathbf{\Sigma}^{-1} (x-\mu) \right) \qquad (2.2)$$

Donde $|\mathbf{\Sigma}|$ es el determinante de la matriz de covarianzas $\mathbf{\Sigma}$, nótese como la ecuación se reduce a la función de densidad de la distribución normal univariante descrita en (2.1) si $\mathbf{\Sigma}$ fuera un escalar.

Para el caso de las distribuciones normales multivariantes las curvas de equidensidad son elipsoides centrados en la media donde las direcciones de los ejes principales de los elipsoides vienen dados por los vectores propios de la matriz de covarianza $\mathbf{\Sigma}$. Las longitudes relativas de los cuadrados de los ejes principales vienen dados por los correspondientes vectores propios.
En la Figura 2.12 podemos apreciar como la representación de una distribución de datos multivariantes mediante una nube de puntos se asemeja a un elipsoide.
En la Figura 2.13 se puede observar como los ejes principales del elipsoide se corresponden con los componentes principales (vectores propios) de la matriz de covarianzas Σ.

La representación de la distribución normal multivariante para la obtención de intervalos o regiones de confianza multivariantes se describirá en el siguiente apartado.

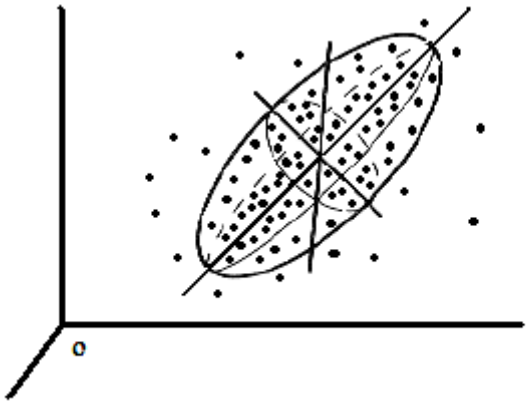


Figura 2.12. Representación de una distribución normal de datos multivariantes

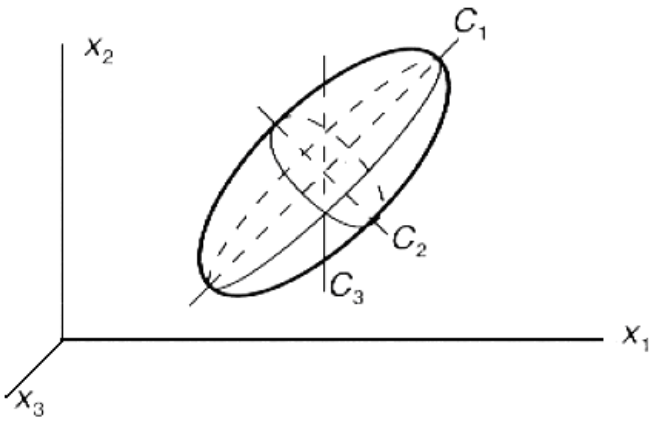


Figura 2.13. Correspondencia entre los ejes principales del elipsoide y los componentes principales de la matriz de covarianzas

Para terminar de describir las distribuciones normales multivariantes se definen a continuación una serie de propiedades que se pueden aplicar a este tipo de distribuciones:

1. Si un vector aleatorio sigue una distribución normal multivariante, puede demostrarse que todas las distribuciones marginales son normales, de forma que cada $x_i \approx N(\mu_i, \sigma_i)$. De la misma manera el resultado recíproco se cumple también, dados n variables aleatorias normales su distribución conjunta es una normal n-dimensional.

2. Puede demostrarse que las distribuciones condicionadas de cualquier dimensión son también normales, y que en particular las de dimensión uno son normales unidimensionales que tienen por media el valor esperado por la regresión lineal y por varianza, la varianza residual de esa regresión
3. Un caso importante de distribución normal multivariante es aquel en el que todas las variables son independientes, en este caso todas las covarianzas serán nulas y la matriz de covarianzas será diagonal:

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \dots & & & \\ 0 & 0 & \dots & \sigma_n^2 \end{pmatrix}$$

2.3.6. INTERVALOS DE CONFIANZA

A continuación se explica brevemente el concepto de intervalo de confianza, tanto para el caso univariante como el multivariante, siempre bajo la hipótesis de distribución normal.

2.3.6.1. Intervalo de confianza univariante de una distribución normal

Se llama intervalo de confianza en estadística a un par de números entre los cuales se estima que estará cierto valor desconocido con una determinada probabilidad de acierto. Formalmente, estos números determinan un intervalo, que se calcula a partir de datos de una muestra, y el valor desconocido es un parámetro poblacional.

La probabilidad de éxito en la estimación se representa por 1 - α y se denomina nivel de confianza. En estas circunstancias, α es el llamado error aleatorio o nivel de significación, esto es, una medida de las posibilidades de fallar en la estimación mediante tal intervalo.

El nivel de confianza y la amplitud del intervalo varían conjuntamente, de forma que un intervalo más amplio tendrá más posibilidades de acierto (mayor nivel de confianza), mientras que para un intervalo más pequeño, que ofrece una estimación más precisa, aumentan sus posibilidades de error.

Para la construcción de un determinado intervalo de confianza es necesario conocer la distribución teórica que sigue el parámetro a estimar θ. Es habitual que el parámetro se distribuya normalmente.

En definitiva, un intervalo de confianza al $1-\alpha\%$ para la estimación de un parámetro poblacional θ que sigue una determinada distribución de probabilidad, es una expresión del tipo $[\theta_1, \theta_2]$ tal que $P(\theta_1 \le \theta \le \theta_2) = 1-\alpha$, donde P es la función de distribución de probabilidad de θ.

La estimación confidencial consiste en determinar un posible rango de valores o intervalo, en los que pueda se, con una determinada probabilidad, que el valor de un parámetro se encuentra dentro de esos límites. Este parámetro será habitualmente la media o la varianza para distribuciones gaussianas.

La técnica de la estimación confidencial consiste en asociar a cada muestra un intervalo que se sospecha que debe contener al parámetro. A éste se le denomina intervalo de confianza

Evidentemente esta técnica no tiene porqué dar siempre un resultado correcto. A la probabilidad de que hayamos acertado al decir que el parámetro estaba contenido en dicho intervalo es a lo que se la denomina nivel de confianza o nivel de significación.

En el caso de las distribuciones normales generalmente se construyen intervalos de confianza de $1-\alpha = 95\%$, o lo que es lo mismo con un nivel de significación $\alpha = 5\%$, se puede comprobar, tal y como se ha definido anteriormente que una distribución normal cumple: $P(-1.96 < z < 1.96) = 0.95$ (esto se puede comprobar mediante una tabla de probabilidades normales).

Luego si una variable X tiene una distribución $X \approx N(\mu, \sigma^2)$, entonces el 95% de las veces cumple $-1.96 \le \left(\frac{X-\mu}{\sigma} \right) \sqrt{n} \le 1.96$.

Si despejamos μ en la ecuación anterior podemos obtener un intervalo de confianza para la media al 95% cuando la variable X es normal y σ^2 es conocido.

$$X - 1.96\frac{\sigma}{\sqrt{n}} \le \mu \le X + 1.96\frac{\sigma}{\sqrt{n}}$$

Siguiendo un procedimiento parecido, y utilizando las características conocidas que cumple una distribución normal podemos obtener intervalos de confianza para una proporción o para una varianza. Además este método se puede utilizar para verificar hipótesis planteadas respecto a diferentes parámetros poblacionales

2.3.6.2. Región de confianza de una distribución normal multivariante

Para entender mejor la diferencia, describimos en primer lugar la región de confianza para dos variables analizadas de forma independiente. La región de confianza para esas dos variables se representa en la figura 2.14. Como puede observarse, se trata de una región rectangular, que en el caso de dimensión mayor que 2 será un hiperrectángulo.

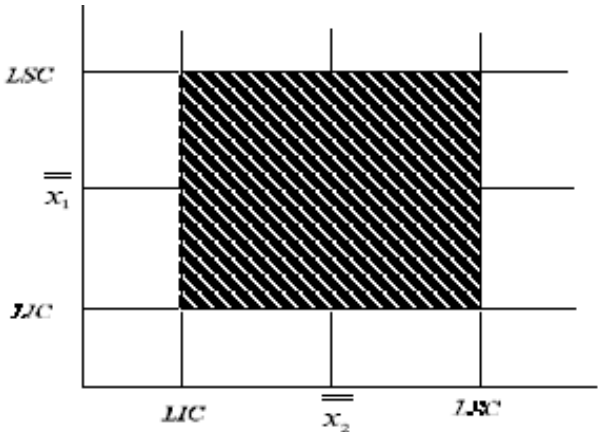


Figura 2.14. Región de confianza para dos variables analizadas individualmente.

Sin embargo, la representación de la región de confianza para dos variables correladas y analizadas de forma conjunta, tiene forma de elipse, cuya inclinación depende de la correlación existente entre las variables. La forma de la elipse dependerá de la cuantía de la correlación y de su signo. Por ejemplo, para dos variables positivamente correlacionadas, la región de confianza se representa en la figura 2.15. En el caso de dimensión mayor que 2, la región será un hiperelipsoide.

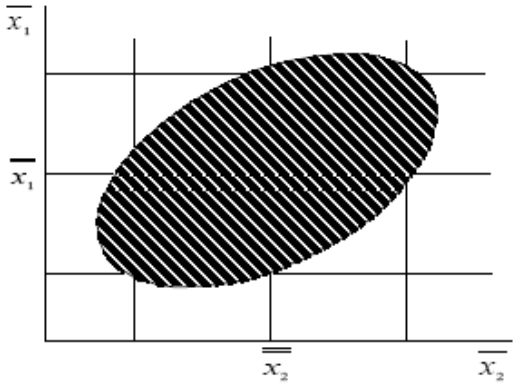


Figura 2.15 Región de confianza para dos variables correladas

2.3.6.3. Proyección de elipses para el cálculo de intervalos de confianza individuales a partir de una región de confianza multivariante

Lo que se pretende es poder expresar unos intervalos de confianza para cada variable, obtenidos a partir de su región de confianza multivariante. Es decir, pasar de una región que es un hiperelipsoide a una región que es un hiperrectángulo.

Bajo la suposición de normalidad multivariante de X, la región que concentra el $100(1-\alpha)\%$ de la distribución es un hiperelipsoide de dimensión K (dependiendo del número de variables que estemos analizando) y de volumen (ver Cramer, 1974) igual a:

$$Vol = |\Sigma_X|^{1/2} \left(\pi \chi^2_{K,1-\alpha}\right)^{k/2} \left[\Gamma\left(K/2+1\right)\right]^{-1} \qquad (2.3)$$

Donde: $|\Sigma_X|$ es el determinante de la matriz de covarianzas de X.

Γ es la distribución Gamma.

$\chi^2_{K,1-\alpha}$ es el $100(1-\alpha)\%$ de la distribución chi-cuadrado con K grados de libertad.

Notar que este volumen no solo depende de σ_i sino también de las covarianzas de $\mathbf{X}_i$. Como en el caso de los intervalos de confianza univariantes, la región de confianza univariante necesita ser expresada como cierta distancia desde la media, donde esta distancia esta relacionada con la matriz de covarianzas de **X**.

La región de confianza definida por los intervalos de confianza de cada variable $[LCI, LCS]$, donde llamaremos a LCI y LCS límites de confianza superior e inferior respectivamente, es un hiperrectangulo de dimensión K, centrado, en μ_x y con lados paralelos a $X_1, X_2, ..., X_K$. Consecuentemente este hiperrectangulo debe estar relacionado con el hiperelipsoide definido por $\boldsymbol{\Sigma}$.

Nickerson (1994) define la proyección del hiperelipsoide de volumen (2.3) en los ejes $X_1, X_2, ..., X_K$. Esta proyección define un hiperrectangulo centrado en μ y tangente al hiperelipsoide. Este hiperrectangulo puede ser usado para definir nuestro hiperrectangulo de intervalos de confianza, cogiendo las semilongitudes de los lados de este hiperrectangulo definido por Nickerson como las longitudes de la región de confianza, es decir, como los límites inferior y superior de confianza que hemos definido anteriormente. Entones las longitudes serian:

$$l_i = \sqrt{\left(\Sigma_x^1\right)_{ii} \chi^2_{K,1-\alpha}} \tag{2.4}$$

Donde el subíndice ii denota el i-esimo elemento de la diagonal inversa de la matriz de covarianzas, Σ_x. La ventaja de esta región rectangular es que es capaz de expresar la variabilidad de **X** en términos de cada variable individual. Las longitudes de los lados del hiperrectangulo conjuntamente representan aproximadamente el $100(1-\alpha)\%$ de la variabilidad de **X** en términos de cada variable. Usando l_i los límites de la región de confianza de cada variable son:

$$X_i \in [LCI, LCS] = [\mu_i - l_i, \mu_i + l_i] \tag{2.5}$$

En la Figura 2.16 podemos ver una representación gráfica del procedimiento para un caso con dos dimensiones. La región A representa la elipse con el $100(1-\alpha)\%$ de la distribución normal de Σ_x y la región B representa el rectángulo obtenido de la proyección de la elipse utilizando Nickerson (1994). Donde $[LCI_i, LCS_i]$ son los límites del intervalo de confianza inferior y superior de cada variable.

Ya que el hiperrectangulo contiene el hiperelipsoide de cobertura $100(1-\alpha)\%$, esta región de confianza es conservadora. Mas adelante, durante la descripción de la metodología que se propone en este proyecto, este aspecto será corregido

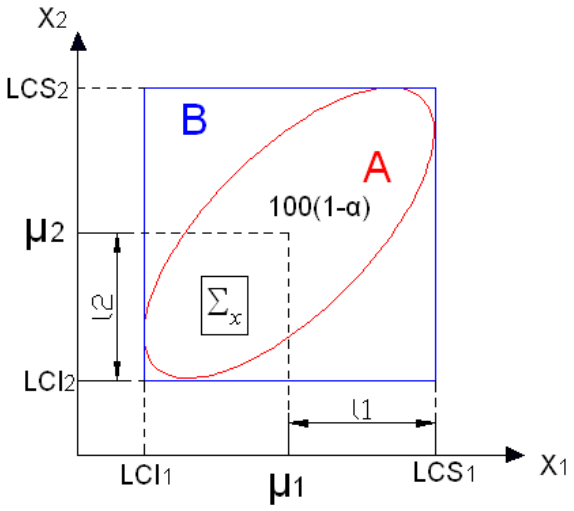


Figura 2.16. Región de confianza rectangular (A) obtenida a partir de la región de confianza multivariante (elíptica) definida por la distribución normal de Σ_x. (B)

2.3.7. ANÁLISIS DE COMPONENTES PRINCIPALES

El análisis de componentes principales (ACP) es debida a Hotelling (1933), aunque sus orígenes se encuentran en los ajustes ortogonales por mínimos cuadrados introducidos por K.Pearson (1901).

La utilidad del análisis de componentes principales es la siguiente:

1. Permite representar óptimamente en un espacio de dimensión más pequeña observaciones de un espacio general p-dimensional.

2. Permite transformar un conjunto de variables correladas en un conjunto de nuevas variables independientes, facilitando el manejo de los datos. Esto, bajo la suposición de normalidad.

Las nuevas variables son combinaciones lineales de las anteriores y se van construyendo según el orden de importancia en cuanto a la variabilidad total que recogen de la muestra.

A continuación se describe brevemente el procedimiento necesario para realizar un análisis de componentes principales:

1. Partimos de la siguiente matriz de datos datos:

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}$$

Donde *n*= individuos
p=variables

2 .Se calcula la matriz de covarianzas (puede usarse también la de correlaciones).

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \dots & \sigma_{1p}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \dots & \sigma_{2p}^2 \\ \dots & & & \\ \sigma_{p1}^2 & \sigma_{p2}^2 & \dots & \sigma_{pp}^2 \end{pmatrix}$$

3.Se diagonaliza la matriz de covarianzas, es decir, se descompone esta matriz de manera que obtengamos una matriz diagonal D cuya diagonal estará formada por los autovalores de la matriz inicial de covarianzas $(\lambda_1, \lambda_2, ...\lambda_p)$ y una matriz de autovectores **C**.

$$\mathbf{\Sigma} = \mathbf{CDC}' \tag{2.6}$$

donde **C** es la matriz de autovectores (pxp) y **D** es una matriz diagonal (pxp) de autovalores:

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ 0 & 0 & \dots & \lambda_p \end{bmatrix} \tag{2.7}$$

Las variables independientes que forman la matriz $\mathbf{Z}$ se pueden obtener como:

$$\mathbf{Z} = \tilde{\mathbf{X}}\mathbf{C} \tag{2.8}$$

donde Z son las nuevas variables independientes (nxp)

$\tilde{\mathbf{X}} = (\mathbf{X} - 1\boldsymbol{\mu}')$ Es el vector de datos al que se le resta el vector de medias (nxp)

Para el método de diseño de tolerancias considerando correlación entre las variables que vamos a describir en el proyecto, el análisis de componentes principales nos resultara útil, ya que mediante su uso podremos transformar las variables dependientes en variables independientes, simplificando el cálculo, tal como se explicara mas adelante.

2.4. PLANTEAMIENTO DEL PROBLEMA DE DISEÑO DE TOLERANCIAS

En la práctica, la calidad y funcionalidad de las piezas están relacionadas normalmente con una o más variables como pueden ser la longitud, el ancho, el peso, el voltaje etc. A las que en adelante designaremos como variables:

$$\mathbf{Y} = (Y_1, Y_2, ..., Y_J)' \qquad (2.10).$$

Para que las piezas se consideren no defectuosas deben cumplir unas especificaciones en estas variables $\mathbf{Y}$, es decir, deben cumplir unos límites de especificación que se designarán de la siguiente manera:

$$\left[LES_j, LEI_j \right]$$

Donde: LES_j es el límite de especificación superior de la variable Y_j

LEI_j es el límite de especificación inferior de la variable Y_j

Estas variables $\mathbf{Y}$ son función de una o varias variables $\mathbf{X} = (X_1, X_2, ..., X_K)'$ que son las variables que se miden en la pieza. Es decir:

$$\mathbf{Y} = f(\mathbf{X}) \qquad (2.11)$$

Por ejemplo la calidad de una pieza puede estar definida por la variable $Y_1 =$ máximo peso permitido. Esta variable, depende de otras variables como son, $X_1 =$ longitud, $X_2 =$ ancho y $X_3 =$ grosor de la pieza. Como se puede observar en la Figura 2.17. En este caso, el peso de la pieza vendrá dado de la siguiente manera, con lo que queda demostrada la relación entre las variables longitud, ancho y grosor:

$$Y_1 = P = m \cdot g$$
$$\rho = \frac{m}{V} \rightarrow m = \rho \cdot V = \rho \cdot X_1 \cdot X_2 \cdot X_3$$
$$Y_1 = g \cdot \rho \cdot \left(X_1 \cdot X_2 \cdot X_3 \right)$$

Además, en este caso, parece muy probable que las magnitudes X_1, X_2 y X_3 estén correladas debido al proceso de fabricación.

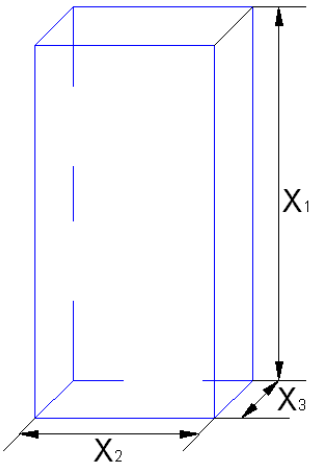


Figura 2.17. Magnitudes que se miden en la pieza.

De la misma forma, en un conjunto mecánico normalmente hay ciertas características de calidad $\mathbf{Y}$, que son función de las dimensiones de los componentes individuales que forman el ensamblaje mecánico $\mathbf{X}$.

En la Figura 2.18. se observa un ejemplo de un conjunto mecánico que consta de un bloque que debe encajar dentro de una base, para ello definiremos una serie de variables $\mathbf{X}=\left[X_1, X_2, ...X_7\right]$, que nos dan información acerca de las dimensiones del bloque y la base, además podemos definir, por ejemplo tres variables Y que serán las especificaciones que deberá cumplir el conjunto para cumplir con las funciones para las que fue diseñado, así llamaremos:

Y_1 a la anchura total de la base: $Y_1 = X_2 + X_3$

Y_2 al hueco que queda entre la base y la caja interior: $Y_2 = X_2 - X_1 - X_6$

Y_3 a la diferencia de alturas entre la caja interior y la base: $Y_3 = X_5 - X_4 - X_7$

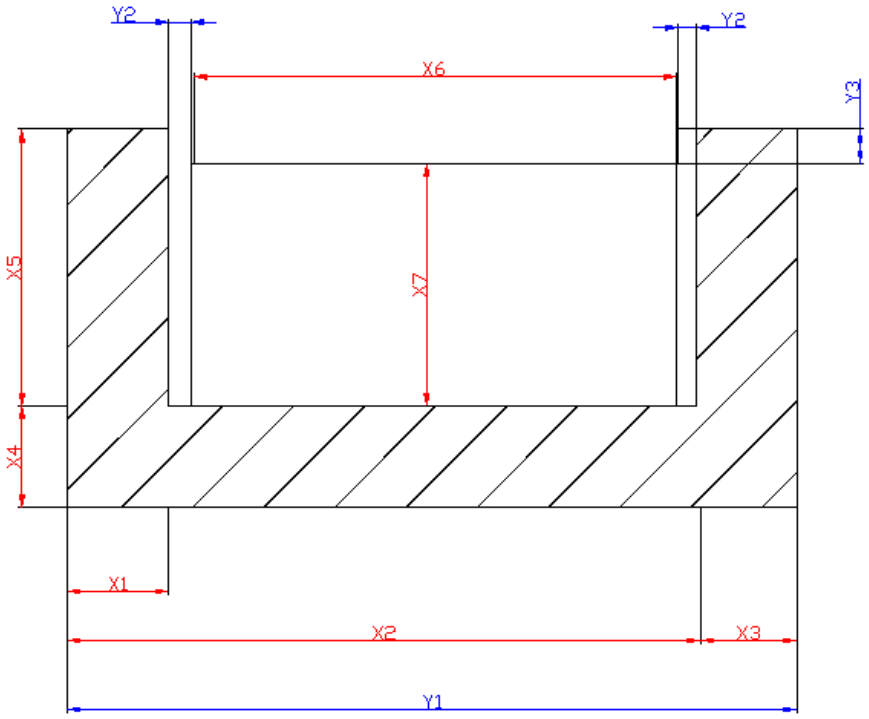


Figura 2.18. Ejemplo de conjunto mecánico con dependencia de las variables **X** e **Y**

En estos casos, las especificaciones de $\mathbf{Y}$ imponen restricciones en los valores que pueden tomar las variables $\mathbf{X}$. Es decir, las variables $\mathbf{X}$ deberán tener valores dentro de unos límites, los denominados límites de tolerancia:

$$\left[LTS_i, LTI_i\right]$$

Donde LTS_i es el límite de tolerancia superior de la variable X_i

LTI_i es el límite de tolerancia inferior de la variable X_i.

En resumen, para que una pieza o un conjunto mecánico tengan la calidad para la que fue diseñado, un conjunto de variables $\mathbf{X}$ que son las que se controlan, deberán cumplir unos límites de tolerancia que dependerán de las especificaciones establecidas para la pieza o conjunto. Mientras que las especificaciones de $\mathbf{Y}$ están impuestas por los requerimientos de funcionalidad y se asignan en el proceso de diseño, las tolerancias de $\mathbf{X}$ deben ser asignadas de forma óptima. A esta asignación óptima se le denomina diseño de tolerancias

El diseño de tolerancias tiene un impacto significativo en la calidad y en los costes de fabricación. Unas tolerancias muy pequeñas podrán asegurar el cumplimiento de las especificaciones de $\mathbf{Y}$ con una alta probabilidad, pero implican un incremento en los costes de fabricación, es decir, unas tolerancias demasiado ajustadas imponen un proceso productivo muy caro. Por el contrario, unas tolerancias amplias significan costes de fabricación bajos pero pueden conducir a problemas de calidad de las piezas, es decir conducen a un aumento en el coste de calidad. Por esta razón el problema del diseño de tolerancias se suele formular como un problema de optimización en el que se busca asignar las tolerancias que minimicen los costes totales: costes de fabricación más costes de calidad.

En general, los métodos más usados para la diseño de tolerancias están basados en aproximaciones estadísticas. A este tipo de diseño se le denomina diseño estadístico de tolerancias.

En el diseño estadístico de tolerancias, la distribución de las variables $X = (X_1, X_2, ..., X_K)'$ se asume como el resultado de la variabilidad del proceso de producción y generalmente, se asume distribución normal de las variables. La distribución de $\mathbf{Y}$ se obtiene a partir de la distribución de $\mathbf{X}$. Conocida la distribución de $\mathbf{Y}$ se calcula el porcentaje de piezas que se encuentran fuera de los limites de especificación, o lo q es lo mismo, el porcentaje de piezas defectuosas, denotado como p.

La asignación óptima de las tolerancias de las $\mathbf{X}$ es aquélla que minimiza los costes y permite cumplir las especificaciones de $\mathbf{Y}$, con una proporción de defectuosos menor o igual que α, que generalmente se toma igual a 0.0027.

Usar un criterio estadístico es más eficiente que asignar las tolerancias de manera que el 100% de las piezas cumpla todas las especificaciones. Esto en la práctica se traduce en tolerancias muy ajustadas que incrementan los costes de fabricación.

2.5. ESTADO ACTUAL DEL PROBLEMA DE DISEÑO DE TOLERANCIAS

2.5.1. DISEÑO DE TOLERANCIAS CONSIDERANDO INDEPENDENCIA ENTRE LAS VARIABLES

En la bibliografía actual sobre diseño óptimo de tolerancias la mayoría de los trabajos se basan en la suposición de independencia de las variables $\mathbf{X}$. Muchos de estos trabajos tienen algunos aspectos en común. Este es el caso, por ejemplo de los métodos propuestos en Chen (1996), Wu et al (1998), Feng and Balasu (1999), Lee and Tang (2000), Ye and Salustri (2003) y Huang et al (2005). Estos aspectos comunes será descritos a continuación y nos servirán como punto de referencia para la metodología propuesta en este trabajo.

La suposición de independencia de las variables puede no ser demasiado realista ya que en la mayoría de los casos las variables $\mathbf{X}$ están relacionadas debido al proceso de fabricación. Por ejemplo, en el caso de la Figura 2.17.

Nos referimos a independencia de las variables $\mathbf{X}$ cuando estas no están relacionadas entre si, es decir, cuando no existe entre ellas ningún tipo de correlación. La variación en alguna de las variables no afecta al resto.

A continuación se describirá brevemente la metodología utilizada en el caso de suposición de variables independientes.

2.5.1.1. Definición del problema

El problema se plantea tal y como se ha descrito en el apartado 2.4. Por simplicidad, suponemos que se tiene una variable Y que debe estar dentro de unos límites de especificación para que la pieza se considere aceptable, es decir:

$$Y \in [LEI, LES] \tag{2.12}$$

Esta variable Y es función de un conjunto de variables $\mathbf{X}$, que representan una serie de magnitudes de la pieza.

El problema consiste en asignar los límites de tolerancias de las variables $\mathbf{X}$, $[LTI_i, LTS_i], i = 1, 2, ..., K$, que minimicen los costes totales (costes de fabricación más costes de calidad) y permitan obtener una proporción de piezas defectuosas, p, menor o igual que cierto pequeño valor, α, es decir:

$$p \equiv P\left(Y \notin [LEI, LES]\right) \leq \alpha \tag{2.13}$$

Las tolerancias de $\mathbf{X}$ se definen como simétricas con respecto a su valor nominal.

2.5.1.2. Metodología

Las metodologías propuestas actualmente tienen muchos aspectos comunes. En este apartado se resume brevemente las metodologías propuestas en Chen (1996), Wu et al (1998), Feng and Balasu (1999), Lee and Tang (2000), Ye and Salustri (2003) y Huang et al (2005). En estos trabajos se utilizan distintas funciones de coste, sin embargo, aquí se describirá el procedimiento, considerando que el coste es igual para todas las variables, por lo que el problema consistirá en encontrar las tolerancias más amplias posibles, cumpliendo las restricción de que la proporción de defectuosos sea menor que α.

Se supone que las variables $\mathbf{X}$ son independientes y cada una de ellas sigue una distribución normal $X_i \square N(\mu_i, \sigma_i)$. También se asume que las medias de las variables coinciden con los valores nominales respectivos y se consideran constantes y que las tolerancias dependen de las desviaciones típicas σ_i.

Se hace uso del índice de capacidad C_p que relaciona las tolerancias de una variable con la desviación típica de la misma. Así suponiendo un índice igual a 1, $C_p = 1$, las tolerancias serán iguales $t_i = 3\sigma_i$. Y por tanto, los límites de tolerancias serán iguales a $\mu_i \pm 3\sigma_i$. Es decir,

$$X_i \in \left[LTI_i, LTS_i\right] = \left[\mu_i - 3\sigma_i, \mu_i + 3\sigma_i\right] = \left[\mu_i - t_i, \mu + t_i\right] \qquad (2.14)$$

La asignación óptima de tolerancias consiste entonces en optimizar t_i de manera que la proporción de piezas defectuosas p no sea más grande que un valor α.

Dado que $t_i = 3\sigma_i$, optimizar las tolerancias puede ser planteado como optimizar las desviaciones típicas σ_i.

Para poder verificar que la proporción de piezas defectuosas p sea la adecuada, es decir, se cumpla 2.7., necesitamos estimar la distribución de la variable. Para ello la relación $Y = f(X_1, X_2, ..., X_K)$ debe ser conocida.

En general, se supone que esta función tiene una forma analítica conocida, y es una relación lineal de la forma: $Y = a_0 + a_1 X_1 + ... + a_K X_K$ o bien $Y = a_0 + \mathbf{a}'\mathbf{X}$ Donde: $\mathbf{a} = (a_1, a_2, ..., a_K)'$.

Bajo esta suposición, $\mathbf{Y}$ sigue una distribución normal $\mathbf{Y} \approx N(\eta_Y, \sigma_Y^2)$ donde:

$$\eta_y = a_0 + a_1\mu_1 + ... + a_K\mu_{K,} \tag{2.15}$$

Y la desviación típica de σ_Y puede ser calculada, dado que las variables $\mathbf{X}$ son independientes, como:

$$\sigma_Y = \sqrt{a_1^2\sigma_1^2 + a_2^2\sigma_2^2 + ...a_K^2\sigma_K^2} \tag{2.16}$$

Si la relación lineal no es válida, es posible aproximar la función $Y = f(X_1, X_2, ..., X_K)$ usando una aproximación de Taylor de primer orden. En este caso los coeficientes $(a_1, a_2, ..., a_K)$ se obtienen calculando las primeras derivadas de Y con respecto a cada variable X_i y evaluando las derivadas en $X_i = \mu_i$. Es decir:

$$a_i = \frac{\partial Y}{\partial X_i} \tag{2.17}$$

Una vez que la distribución de Y se ha calculado, el porcentaje de piezas defectuosas p se puede obtener de acuerdo a la teoría de probabilidad de una normal: (2.4).

$$p = P\left(Y \notin \left[LEI, LES\right]\right)$$

En resumen los pasos para la asignación óptima de las tolerancias son los siguientes. Para simplificar el proceso supondremos que tanto la media de las variables $\mathbf{X}$ como la de las variables Y es igual a cero: $\mu_i = 0, \eta_Y = 0$

1. Empezamos la optimización desde algunos valores iniciales $\sigma_i^{(r)}, i = 1, 2...K$, para la primera iteración (r=0), estos valores son aleatorios.

2. Calcular $\sigma_Y^{(r)}$ introduciendo los valores de $\sigma_i^{(r)}$ obtenidos en el paso anterior en la ecuación (2.7)

$$\sigma_Y = \sqrt{a_1^2\sigma_1^2 + a_2^2\sigma_2^2 + ...a_K^2\sigma_K^2}$$

3. Calcular la proporción de piezas defectuosas mediante la teoría de la probabilidad normal

$$p = P\left(Y \notin \left[LEI, LES\right]\right)$$

4. Si $\alpha - p > \varepsilon$, donde ε es un valor pequeño, se realizara la siguiente iteración, $r = r + 1$. Asignar un nuevo valor para $\sigma_i^{(r)}$ y volver al paso 2. De lo contrario continuar con el paso 5:

5. Calcular $t_i = 3\sigma_i^{(r)}$, con lo que los límites de tolerancia serán:

$$LTI_i = \mu_i - t_i$$

$$LTS_i = \mu_i + t_i$$

Este tipo de problemas también podría ser utilizado para el caso de que el proceso sea multivariante $\mathbf{Y} = (Y_1, Y_2, ..., Y_J)'$ con lo que obtendríamos J límites de especificación. $Y_J \in \left[LEI_J, LES_J\right]$. En este tipo de casos, en trabajos anteriores, las variables son tratadas como independientes y la proporción de piezas defectuosas, p, se calcula independientemente para cada variable Y_J, lo que no asegura que la proporción total de piezas defectuosas cumpla con $p \leq \alpha$.

La suposición de independencia de $\mathbf{X}$ puede ser válida en algunos casos, pero hay muchos otros casos donde estas variables están relacionadas. Esta dependencia podría ser debida a que las variables $\mathbf{X}$ dependen del propio proceso de fabricación. Por ejemplo en el proceso de moldeo o de estampado con prensa, la longitud, anchura y espesor de una pieza están relacionados.

En este proyecto se va a desarrollar un método para el diseño de tolerancias óptimas considerando correlación entre las variables $\mathbf{X}$. Esta correlación será consecuencia del proceso de fabricación.

La metodología descrita en este apartado servirá de base para la metodología que se propondrá en el proyecto.

Capítulo 3

PROPUESTA DE UNA METODOLOGIA PARA EL DISEÑO DE TOLERANCIAS CONSIDERANDO CORRELECIÓN ENTRE LAS VARIABLES

CAPÍTULO 3. PROPUESTA DE UNA METODOLOGIA PARA EL DISEÑO ÓPTIMO DE TOLERANCIAS CONSIDERANDO CORRELACIÓN ENTRE LAS VARIABLES.

En este capítulo se propondrá una metodología para el diseño óptimo de tolerancias considerando que existe una correlación o dependencia entre las variables, para ello se realizará una introducción en la que se explicarán brevemente los datos de partida del problema y algunos pasos que será necesario realizar a lo largo de la metodología. Por ultimo se desarrollara la metodología de manera secuencial.

3.1. DEFINICIÓN DEL PROBLEMA DE DISEÑO DE TOLERANCIAS CONSIDERANDO CORRELACIÓN ENTRE LAS VARIABLES.

A continuación se describirán las bases de la metodología propuesta para asignar tolerancias óptimas a una serie de características de una pieza o conjunto mecánico, variables **X,** que están correladas entre si, es decir, son variables dependientes. Este método también considera una serie de variables **Y** que definen las restricciones o especificaciones de la pieza y que dependen de las variables **X** mediante la relación $\mathbf{Y} = f(\mathbf{X})$. Luego, al ser función de las mismas variables **X** es probable que las variables **Y** también estén correladas entre sí.

Por ejemplo, en la Figura 3.1 se representa un conjunto mecánico con una serie de características $\mathbf{X} = (X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)'$ que definen un conjunto de longitudes de las piezas que forman el conjunto mecánico. En este caso, es muy probable que las variables X_1, X_2, X_3, X_4, X_5 estén correladas entre sí, debido al proceso de fabricación de la pieza. Y lo mismo ocurre en el caso de las variables X_6, X_7, X_8.Por otro lado, se definen un conjunto de variables $\mathbf{Y} = (Y_1, Y_2, Y_3, Y_4)'$ que dependen de las variables X y que definen ciertas holguras necesarias para el correcto ensamblaje de las piezas. Estas variables deben verificar una serie de restricciones para que el conjunto mecánico no sea defectuoso. Es decir,

$$Y_1 = X_4 + X_5$$
$$Y_2 = X_2 - X_1 - X_8 + X_7$$
$$Y_3 = X_7 - X_6 - X_3 + X_2$$
$$Y_4 = X_4 - X_3 - X_6$$

$$Y_1 \leq 127.13mm$$
$$Y_2 \geq 0.0076mm$$
$$Y_3 \geq 0.0254mm$$
$$Y4 \geq 0.0076mm$$

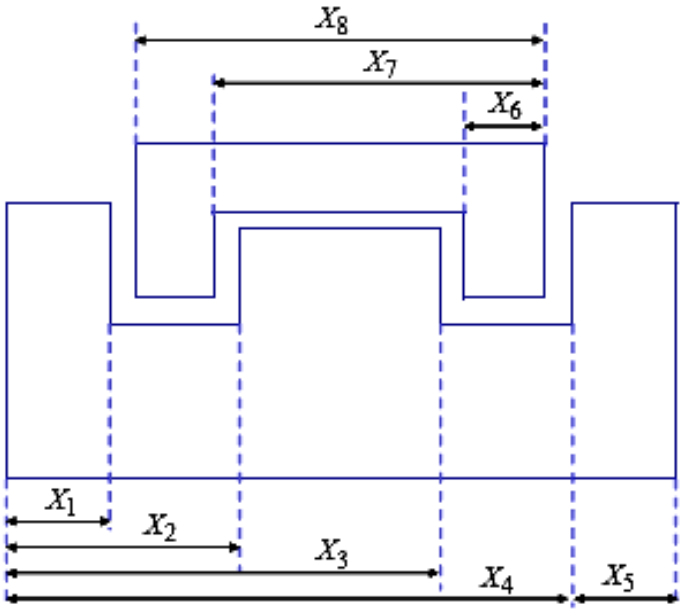


Figura 3.1. Conjunto mecánico

El objetivo es encontrar los límites de tolerancia óptimos $[LTI_i, LTS_i]$, $i = 1,2,...,K$ para el conjunto de las K variables dependientes $\mathbf{X} = (X_1, X_2, ..., X_K)'$, de manera que se cumplan las especificaciones de diseño, con una proporción de defectuosos, p, menor o igual a cierto valor α. Los límites de tolerancia inferior y superior se consideran simétricos con respecto al valor nominal de cada variable X_i ,VN_i ,. Es decir, $LTI_i = VN_i - t_i$ y $LTS_i = VN_i + t_i$, siendo t_i las tolerancias de X_i.

Luego, las tolerancias deben asignarse de manera que el conjunto de las J variables $\mathbf{Y} = (Y_1, Y_2, ..., Y_J)'$ verifiquen:

$$p = P(Y \notin S) \leq \alpha \qquad (3.1)$$

Donde S es la región rectangular de especificaciones o restricciones de la pieza, definida por $S = \left\{ \mathbf{Y} \in R^J : \left(LEI_j \le Y_j \le LES_j \right), j = 1,..., j \right\}$. Aquí LEI_j y LES_j son límites de especificación simétricos alrededor del valor nominal de Y_j. Estas especificaciones se pueden generalizar a especificaciones unilaterales donde LEI_j o LES_j son iguales a infinito.

Luego, el problema será:

$$FO : \max\ t_i;\ i = 1,2,...,K = \max \sum_{i=1}^{K} t_i$$

sujeto a: $p \le \alpha$

3.2. CONSIDERACIONES PARA LA RESOLUCIÓN DEL PROBLEMA

En este problema de asignación óptima de tolerancias se considerará que los costes de fabricación y calidad son iguales para todas las variables **X**, por lo que la asignación de tolerancias es un problema de maximización de las tolerancias t_i, sujeto a cumplir una proporción de defectuosos determinada (ecuación 3.1).
Para la resolución de este problema se asumirán los siguientes aspectos:

1. Se asume normalidad multivariante para **X**, esto es, $\mathbf{X} \sim N_K(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$ donde $\boldsymbol{\mu}_x$ es el vector de medias de **X** y $\boldsymbol{\Sigma}_x$ es la matriz de covarianzas de **X**. Ambos $\boldsymbol{\mu}_x$ y $\boldsymbol{\Sigma}_x$ dependen del proceso de fabricación de la pieza o conjunto mecánico.

2. Se supondrá que el proceso está centrado en los valores nominales, VN_i, por tanto $\mu_i = VN_i;\ \ i = 1,2,...,K$.

3. Se supone una relación lineal conocida entre **X** e **Y:**

$$\mathbf{Y} = f(X_1, X_2, ..., X_K) \tag{3.2}$$

donde cada variable puede ser expresada como $Y_j = a_{0j} + a_{1j}X_1 + ... + a_{Kj}X_K$.
Cuando la relación lineal es solo aproximadamente cierta, se puede aplicar una expansión de Taylor. En este caso los coeficientes a_{ij} serán:

$$a_{ij} = \frac{\partial Y_j}{\partial X_{ij}} \tag{3.3}$$

y el coeficiente a_{0j} se obtendrá del siguiente desarrollo de Taylor.

$$Y_j = VN_{Yj} + \sum_{i=1}^{m} \frac{\partial Y_j}{\partial X_i}\bigg|_{VN_i} (X_i - VN_i)$$

$$Y_j = VN_{Yj} + \frac{\partial Y_j}{\partial X_1}\bigg|_{VN_1} (X_1 - VN_1) + \frac{\partial Y_j}{\partial X_2}\bigg|_{VN_2} (X_2 - VN_2) + ... + \frac{\partial Y_j}{\partial X_m}\bigg|_{VN_m} (X_m - VN_m)$$

$$Y_j = a_0 + a_1X_1 + a_2X_2 + ... + a_mX_m$$

Bajo la suposición de linealidad, es correcto que $\mathbf{Y} \sim N_m(\boldsymbol{\eta}_y, \boldsymbol{\Sigma}_y)$. El vector de medias $\boldsymbol{\eta}_y$ verifica:

$$\boldsymbol{\eta}_y = \mathbf{a} + \mathbf{A}\boldsymbol{\mu}_x' \tag{3.4}$$

donde $\mathbf{a} = (a_{01}, a_{02}, ..., a_{0j})'$ y **A** es la matriz:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{21} & ... & a_{k1} \\ a_{12} & a_{11} & ... & a_{k2} \\ ... & ... & ... & ... \\ a_{1m} & a_{2m} & ... & a_{km} \end{bmatrix}$$

Y $\boldsymbol{\Sigma}_y$ verifica:

$$\boldsymbol{\Sigma}_y = \mathbf{A}\boldsymbol{\Sigma}_x\mathbf{A}' \tag{3.5}$$

Tal y como se puede apreciar, las bases de la metodología propuesta son similares a las usadas en el caso de independencia descrito en el capítulo II, apartado 2.5.1.2. En este caso, la maximización de t_i fue hecha mediante la optimización de σ_i. Ahora, en el caso de dependencia entre variables, las tolerancias t_i dependerán de toda la matriz de covarianzas de **X**. Por tanto, en la metodología que se propondrá, la maximización de las tolerancias t_i será equivalente a la optimización de la matriz de covarianzas $\boldsymbol{\Sigma}_x$

Consecuentemente para desarrollar la metodología que se propondrá en este proyecto es necesario resolver tres aspectos básicos:

1. Definir una relación entre las tolerancias de **X,** t_i y su matriz de covarianzas, $\boldsymbol{\Sigma}_x$.
2. Definir un procedimiento para optimizar $\boldsymbol{\Sigma}_x$ pero teniendo en cuenta la dependencia entre los términos de esta matriz, debida a la correlación entre las variables **X.**
3. Definir un procedimiento para poder evaluar la ecuación 3.1., es decir, calcular la proporción de piezas defectuosas, p.

En los siguientes apartados se resolverán estos tres aspectos básicos. Una vez explicado esto se describirá con detalle la metodología propuesta.

3.3. RELACIÓN ENTRE TOLERANCIAS Y MATRIZ DE COVARIANZAS

Tal y como se ha explicado en el capítulo anterior, si consideramos que **X** sigue una distribución normal multivariante, la región que concentra el $100(1-\alpha)\%$ de la distribución es un hiperelipsoide de dimensión K y cuyo volumen viene expresado en la ecuación 2.3

En el caso concreto de la metodología que estamos desarrollando, lo que necesitamos es expresar los límites de tolerancia $[LTI_i, LTS_i]$ como cierta distancia desde la media o valor nominal, estando esta distancia relacionada con $\boldsymbol{\Sigma}_x$.

La región de tolerancias definida por los intervalos de tolerancia $[LTI_i, LTS_i]$ es un hiperrectangulo de dimensión K, centrado en $\boldsymbol{\mu}_x$ y con lados paralelos a $X_1, X_2, ..., X_K$. Consecuentemente este hiperrectangulo debe estar relacionado con el hiperelipsoide definido por $\boldsymbol{\Sigma}_x$.

Como se ha explicado en el Capítulo II, aplicando la metodología descrita en el apartado 2.3.6.3, es posible proyectar un hiperelipsoide que abarque el $(1-\alpha)$ de las observaciones, como un hiperrectángulo centrado en $\boldsymbol{\mu}_x$ y tangente al hiperelipsoide, es decir proyectar un hiperrectángulo cuyos lados son paralelos a los ejes $X_1, X_2, ..., X_K$. Luego, para obtener la región de tolerancias es posible utilizar este hiperrectángulo, de manera que las tolerancias t_i serán iguales a las semilongitudes de los lados de este hiperrectángulo: es decir,

$$t_i = \sqrt{\left(\Sigma_x^{-1}\right)_{ii} \chi^2_{K,1-\alpha}} \tag{3.6}$$

donde el subíndice ii denota el i-esimo elemento de la diagonal inversa de $\boldsymbol{\Sigma}_x$. La ventaja de esta región rectangular es que es capaz de expresar la variabilidad de X en términos de cada variable individual X_i. Este hiperrectangulo representa aproximadamente el $100(1-\alpha)\%$ de la variabilidad de X en términos de cada variable X_i. Usando t_i, los límites de tolerancia de cada variable X_i se pueden encontrar como:

$$[LTI_i, LTS_i] = [VN_i - t_i, VN_i + t_i] = [\mu_i - t_i, \mu_i + t_i] \tag{3.7}$$

La figura 3.2 ilustra este procedimiento en dos dimensiones. La región A representa la elipse definida por $\boldsymbol{\Sigma}_x$ con el $100(1-\alpha)\%$ de la cobertura y la región B representa el rectángulo obtenido de la proyección de la elipse.

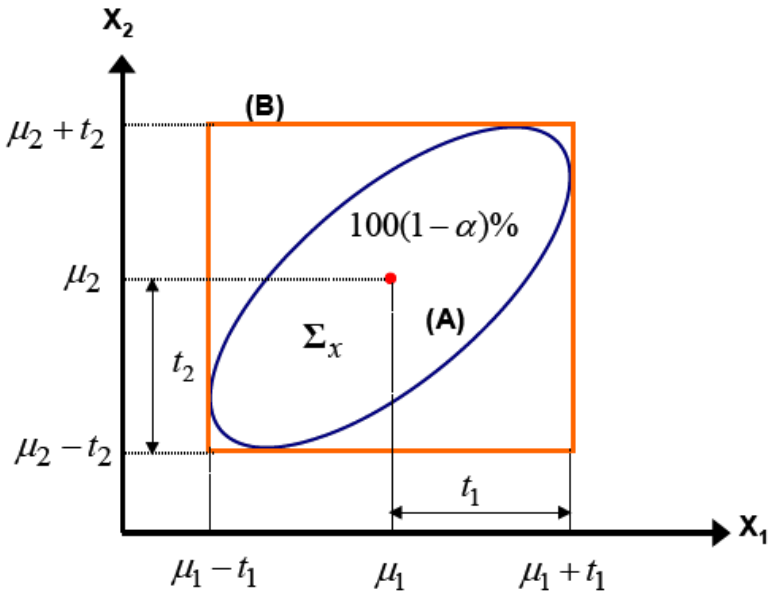


Figura 3.2. Región (A): Región elíptica definida por $\boldsymbol{\Sigma}_x$ .Región (B): Región de tolerancia

3.4. PROCEDIMIENTO PARA OPTIMIZAR LA MATRIZ DE COVARIANZAS

En el caso de la optimización de tolerancias bajo independencia de **X**, las variables a optimizar eran las desviaciones típicas σ_i que eran tratadas como independientes unas de otras. Ahora, los límites de tolerancia que queremos calcular $[LTI_i, LTS_i]$, dependen de la matriz de covarianzas $\boldsymbol{\Sigma}_x$ y no solo de las desviaciones típicas σ_i. Por lo tanto, en este caso, la variable de optimización será la matriz de covarianzas $\boldsymbol{\Sigma}_x$. En cada iteración del proceso de optimización será necesario cambiar $\boldsymbol{\Sigma}_x$. Este cambio en la matriz de covarianzas $\boldsymbol{\Sigma}_x$ implicará una nueva serie de límites de tolerancia y una nueva $\boldsymbol{\Sigma}_y$, ya que, como se ha explicado en el apartado anterior, estos conceptos están relacionados.

En la figura 3.3, a la izquierda se puede apreciar la representación de una elipse definida por $\boldsymbol{\Sigma}_x$ y la región de tolerancia obtenida utilizando las ecuaciones 3.6 y 3.7. A la derecha podemos observar la nueva elipse obtenida mediante la transformación de $\boldsymbol{\Sigma}_x$ en $\boldsymbol{\Sigma}_x^*$ y el resultado de la mayor región de tolerancia.

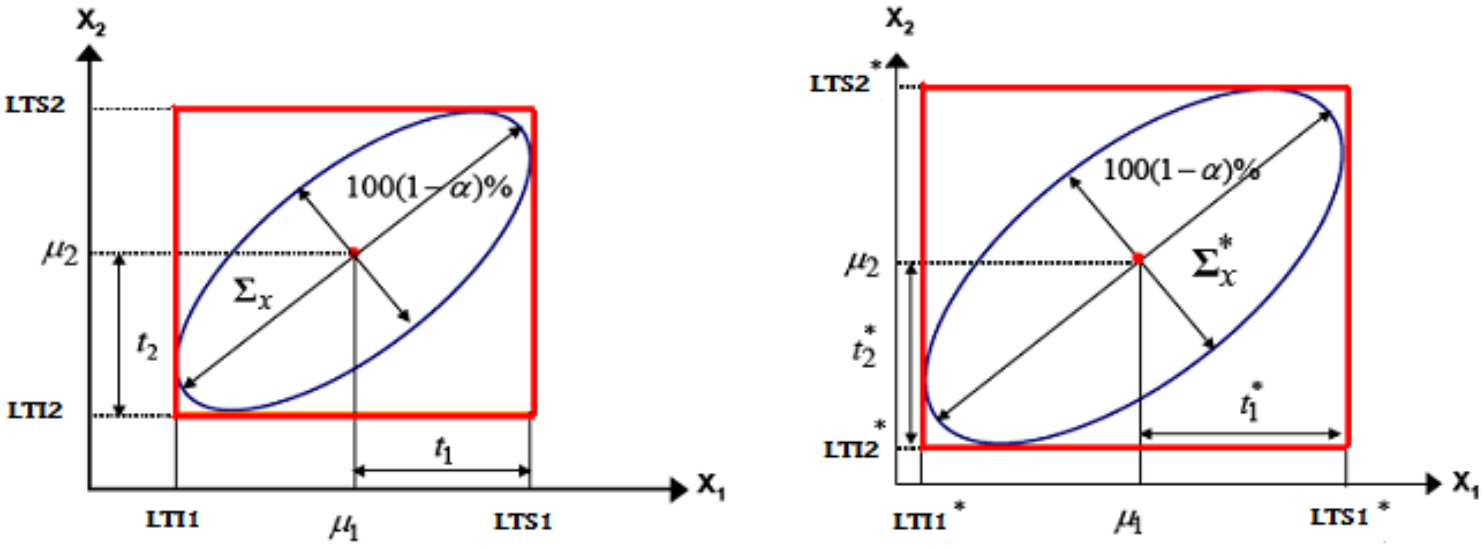


Figura 3.3. Elipse y región de tolerancia definida por: (a) $\boldsymbol{\Sigma}_X$ (b) $\boldsymbol{\Sigma}_X^*$

El problema que se nos plantea es definir el modo de modificar esta matriz $\mathbf{\Sigma}_X$, ya que esta variación podría ser obtenida mediante muchas maneras alternativas. González y Sánchez (2008) proponen una metodología, basada en el Análisis de Componentes Principales que será la que se usará durante este trabajo. En esta metodología se considera que las variables **X** no son independientes pero pueden ser consideradas como una combinación lineal de factores independientes $\mathbf{Z} = (Z_1, Z_2, ..., Z_K)'$.

En el análisis multivariante, estos factores son conocidos normalmente como factores latentes. Estos factores independientes están relacionados con el proceso de fabricación, y pueden ser interpretados como la primera fuente información sobre la estructura de dependencia de **X**. Por lo tanto, cualquier cambio en $\mathbf{\Sigma}_X$ será provocado necesariamente por variaciones en las varianzas de los factores independientes **Z**. Mientras estas modificaciones tengan en cuenta la estructura de la matriz de covarianzas $\mathbf{\Sigma}_X$, serán compatibles con el proceso de fabricación.

Para obtener estos factores independientes se usa el Análisis de componentes principales, explicado en el apartado 2.3.7. Como se describió en este apartado, los factores independientes **Z** se pueden obtener como:

$$\mathbf{Z} = \tilde{\mathbf{X}}\mathbf{C} \tag{3.8}$$

donde $\tilde{\mathbf{X}} = (\mathbf{X} - 1\boldsymbol{\mu}')$ es el vector de datos al que se le resta el vector de medias y **C** es la matriz de autovectores obtenida mediante el análisis de componentes principales:

$$\mathbf{\Sigma} = \mathbf{CDC}' \tag{3.9}$$

siendo **D la** matriz diagonal de autovalores $(\lambda_1, \lambda_2, ...\lambda_K)$:

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda_2 & ... & 0 \\ 0 & 0 & ... & \lambda_K \end{bmatrix}$$

Los autovalores $(\lambda_1, \lambda_2, ...\lambda_K)$ son las varianzas de los factores independientes Z.

Por otro lado, los autovectores representan las direcciones de los ejes del hiperelipsoide definido por Σ_X, y los autovalores representan el tamaño de estos ejes. Así, un cambio en los autovalores significa un cambio en el tamaño de los ejes y un cambio en los autovectores significa un cambio en la dirección de los ejes.

Por lo tanto, si buscamos cambios que conserven la relación de dependencia entre las variables, debemos modificar únicamente los autovalores y no las direcciones de los ejes (autovectores). Los cambios en los autovectores solo tendrán lugar si todo el proceso de fabricación es rediseñado y por tanto, cambian las correlaciones entre variables.

Para modificar la matriz de covarianzas Σ_X será entonces necesario modificar las varianzas de los nuevos factores independientes **Z**, es decir, modificar el valor de los autovalores $(\lambda_1, \lambda_2, ...\lambda_K)$. Una nueva serie de autovalores puede obtenerse mediante la multiplicación de los autovalores $(\lambda_1, \lambda_2, ...\lambda_K)$ por valores b_i, $i = 1, 2..., K$, es decir, la modificación de las varianzas de los nuevos factores independientes **Z** se podría expresar mediante:

$$\begin{aligned} \lambda_1^* &= \lambda_1 b_1 \\ \lambda_2^* &= \lambda_2 b_2 \\ &.... \\ \lambda_K^* &= \lambda_K b_K \end{aligned} \qquad (3.10)$$

Mediante la modificación de las varianzas de las nuevas variables **Z**, independientes realizadas en el paso anterior obtendremos una nueva matriz de autovalores $\mathbf{D}^*$ a través de la cual podremos obtener una nueva matriz de covarianzas $\mathbf{\Sigma}^*$ como**:**

$$\mathbf{\Sigma}^* = \mathbf{C}\mathbf{D}^*\mathbf{C}' \qquad (3.11)$$

donde:

$$\mathbf{D}^* = \begin{bmatrix} \lambda_1^* & 0 & ... & 0 \\ 0 & \lambda_2^* & ... & 0 \\ 0 & 0 & ... & \lambda_K^* \end{bmatrix}$$

y la matriz **C** sigue siendo la inicial.

Mediante este procedimiento, los cambios en $\mathbf{\Sigma}_X$ dependen de los coeficientes b_i. Por lo tanto la optimización de $\mathbf{\Sigma}_X$ puede ser tratada como la optimización de estos K coeficientes independientes b_i.En cada iteración del proceso de optimización, los coeficientes b_i serán cambiados independientemente. Este procedimiento de optimización se puede relacionar con el caso de independencia de **X**, donde las variables de optimización eran las K desviaciones típicas independientes σ_x

3. 5. CÁLCULO DE LA PROPORCIÓN DE DEFECTUOSOS

Dadas unas tolerancias de X es necesario calcular la proporción de defectuosos que producen. Las tolerancias óptimas serán aquellas que produzcan una proporción de defectuosos p, menor o igual que α, que en general se toma igual a 0.0027.

El cálculo de p, puede suponer complicados métodos matemáticos como la integración numérica o simulaciones de Montecarlo. Además este cálculo de p debe ser realizado en cada iteración del proceso de optimización. Por esta razón, en este trabajo se proponen dos procedimientos para la evaluación de la restricción $p \leq \alpha$. El primero es un procedimiento simplificado que permite una rápida evaluación de esta restricción que no necesita calcular p. Este procedimiento se basa en la proyección de Nickerson descrita en el apartado 2.3.6.3. La segunda es una forma más precisa basada en simulaciones de Montecarlo, en la cual si se calcula la proporción de defectuosos p.
Debido a estos dos procedimientos para el cálculo de p, la metodología que se propone tendrá 2 etapas, como se verá más adelante.

3.5.1. PROCEDIMIENTO PARA EVALUAR DE FORMA APROXIMADA LA PROPORCIÓN DE DEFECTUOSOS

Como se ha descrito en el apartado 3.3, dada una matriz de covarianzas $\boldsymbol{\Sigma}_X$ y un vector de medias $\boldsymbol{\mu}_x$, se obtendrá un conjunto de tolerancias t_i. Para este conjunto de tolerancias es necesario evaluar la proporción de defectuosos. El vector de medias y la matriz de covarianzas de Y se puede calcular siguiendo las ecuaciones 3.4. y 3.5, es decir como: $\boldsymbol{\eta}_y = \mathbf{a} + \mathbf{A}\boldsymbol{\mu}_x^{'}$ y $\boldsymbol{\Sigma}_Y = \mathbf{A}\boldsymbol{\Sigma}_X\mathbf{A}'$.

Para estos valores es necesario poder evaluar que el porcentaje de piezas defectuosas p, sea $p \leq \alpha$. Se usará un procedimiento basado en la teoría de los intervalos de confianza simultáneos de Nickerson (1994) que fue descrita en el capitulo II. En este caso consideramos que el porcentaje de piezas defectuosas es:

$$p = P\left(\mathbf{Y} / \in S \left| \mathbf{Y} \square\ N_m\left(\boldsymbol{\eta}_y, \boldsymbol{\Sigma}_y\right)\right.\right) \qquad (3.12)$$

De modo similar al caso de las variables **X**, podemos obtener un hiperrectangulo tangente. Siguiendo el mismo procedimiento que en el caso de las variables **X**, las semilongitudes de este hiperrectangulo correspondiente con $\boldsymbol{\Sigma}_y$ y denotadas mediante l_j pueden ser obtenidas mediante:

$$l_j = \sqrt{\left(\boldsymbol{\Sigma}_y^{-1}\right)_{jj} \chi^2_{m,1-\alpha}} \tag{3.13}$$

Donde el subíndice jj denota el elemento de la diagonal j-esimo de la inversa de $\boldsymbol{\Sigma}_y$.

Los lados de este hiperrectangulo, denotados por $\left[LI_j, LS_j\right]$ pueden ser obtenidos mediante:

$$\begin{aligned} LI_j &= \eta_j - l_j \\ LS_j &= \eta_j + l_j \end{aligned} \tag{3.14}$$

Cuando este hiperrectángulo es igual el hiperrectángulo de especificaciones, lo que significa:

$$\left[LI_j, LS_j\right] = \left[LEI_j, LES_j\right] \; j = 1,2,...,J \tag{3.15}$$

Se puede considerar que el porcentaje de piezas defectuosas p será aproximadamente igual a α.

El porcentaje de defectuosos será en realidad menos que α. Como se puede ver en la figura 3.2, al proyectar el hiperrectángulo, el área (o volumen) que abarca dicho hiperrectángulo es mayor que el de la elipse, es decir es mayor que $100(1-\alpha)\%$. Por tanto, la región fuera del hiperrectángulo es menor que α.

3.5.2. PROCEDIMIENTO PARA EVALUAR LA PROPORCIÓN DE DEFECTUOSOS BASADO EN MONTECARLO

El método de Montecarlo es una técnica que incluye el uso de números aleatorios y el cálculo de probabilidades para solucionar problemas matemáticos.
El método Montecarlo fue bautizado así por su clara analogía con los juegos de ruleta de los casinos, el más célebre de los cuales es el de Montecarlo.

La importancia actual del método Montecarlo se basa en la existencia de problemas que tienen difícil solución por métodos exclusivamente analíticos o numéricos, pero que dependen de factores aleatorios o se pueden asociar a un modelo probabilística artificial (resolución de integrales de muchas variables, minimización de funciones, etc.). Gracias al avance en diseño de los ordenadores, cálculos Montecarlo que en otro tiempo hubieran sido inconcebibles, hoy en día se presentan como asequibles para la resolución de ciertos problemas.

En este procedimiento para el cálculo de p, consideramos que el porcentaje de piezas defectuosas es producto de las piezas que están dentro del intervalo de tolerancias de **X,** esto es:

$$p = P\left(Y \notin S \middle| Y = f(X); X \in T_x\right) \tag{3.19}$$

Donde T_x es la región de tolerancias de **X** y $f(.)$ es la función que relaciona las variables **X** e **Y**.

El procedimiento puede ser resumido de la siguiente manera:

1. Para una matriz de covarianzas $\boldsymbol{\Sigma}_x$ y un vector de medias $\boldsymbol{\mu}_x$, aplicando las ecuaciones 3.6 y 3.7 calculamos la región de tolerancias T_x.

2. Generamos un conjunto de B observaciones de una distribución $\mathbf{X} \square N_K(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$ y seleccionamos únicamente las observaciones que se encuentran dentro de T_x. Denotaremos como $\mathbf{X}_T$ a esas observaciones.
3. Dadas las observaciones $\mathbf{X}_T$, obtendremos las correspondientes variables Y utilizando la relación $\mathbf{Y}_T = f(\mathbf{X}_T)$.
4. Dadas las observaciones $\mathbf{Y}_T$ y las especificaciones del problema, $\left[LEI_J, LES_J\right], j = 1,2,...,J$ se obtendrá la proporción p como el porcentaje de observaciones fuera de dichas especificaciones.

3.6. METODOLOGÍA PARA LA ASIGNACCIÓN DE TOLERANCIAS CONSIDERANDO CORRELACIÓN ENTRE LAS VARIABLES

Una vez descritos los diferentes aspectos que serán necesarios aplicar a lo largo de la metodología de diseño de tolerancias considerando correlación entre las variables procedemos a describir los pasos de la metodología de manera secuencial tal y como se ha hecho para el caso de independencia.

La metodología que se propone para el diseño óptimo de tolerancias tiene dos etapas. En la primera etapa se obtiene una asignación de las tolerancias que produce una proporción de defectuosos aproximadamente igual a α. En la segunda etapa ajustamos el diseño de tolerancias para obtener $p = \alpha$.

3.6.1. ETAPA 1

1. Poner el contador r=0. Estimaremos la matriz de covarianzas inicial $\boldsymbol{\Sigma}_x$ (esta estimación se debe realizar con datos obtenidos del proceso de fabricación funcionando en condiciones normales). El vector de medias $\boldsymbol{\mu}_x$ será igual a los valores nominales de diseño.
2. Usando esta matriz de covarianzas $\boldsymbol{\Sigma}_x$, mediante el Análisis de Componentes Principales (ecuación 3.8), $\boldsymbol{\Sigma}_x = \mathbf{CDC}'$, obtendremos los autovectores de la matriz **C** y la matriz diagonal inicial **D** con sus autovalores $\lambda_1, \lambda_2, ..., \lambda_K$.
3. Obtendremos el vector de medias de **Y**, $\boldsymbol{\eta}_y$ reemplazando $\boldsymbol{\mu}_x$ en $\boldsymbol{\eta}_y = a + \mathbf{A}\boldsymbol{\mu}_x$ (ecuación 3.4)
4. Obtendremos la matriz de covarianzas de **Y** , $\boldsymbol{\Sigma}_y$ reemplazando $\boldsymbol{\Sigma}_x$ en $\boldsymbol{\Sigma}_y = \mathbf{A}\boldsymbol{\Sigma}_x\mathbf{A}'$ (ecuación 3.5)
5. Obtendremos las semilongitudes $l_j, j = 1, 2, ..., m$, usando la inversa de la matriz $\boldsymbol{\Sigma}_y$ y el valor $\chi^2_{m,1-\alpha}$, en $l_j = \sqrt{\left(\boldsymbol{\Sigma}_y^{-1}\right)_{jj} \chi^2_{m,1-\alpha}}$ (ecuación 3.16). Calcularemos los límites

$$LI_j = \eta_j - l_j$$

$$LS_j = \eta_j + l_j$$

6. De acuerdo con la evaluación simplificada que se ha descrito para la verificación del criterio $p \le \alpha$, se verificará que $O = \Sigma_{j=1}^{m} \left(\left| LI_j - LEI_j \right| + \left| LS_j - LES_j \right| \right) \le \varepsilon$, siendo ε un valor pequeño . Si

$O < \varepsilon$ entonces pasamos a la Etapa 2. de otro modo seguiremos con el paso 7

7. Pondremos el contador a r=r+1. Obtendremos una nueva matriz de covarianzas $\boldsymbol{\Sigma}_x^r$ siguiendo el procedimiento de la modificación de los autovalores que se ha descrito con anterioridad, esto es, encontrar una nueva serie de autovalores λ_i^r mediante la aplicación de una serie de K coeficientes b_i^r de acuerdo con $\lambda_i^* = b_i \lambda_i$ (ecuación 3.10). Es decir, mediante la multiplicación de cada λ_x^r por un factor b_i^r. Estos nuevos autovalores serán los elementos de la diagonal de una nueva matriz $\mathbf{D}^r$. Reemplazando la matriz **C** y la nueva matriz $\mathbf{D}^r$ en $\boldsymbol{\Sigma}_x = \mathbf{CDC}'$ (ecuación) obtendremos la nueva matriz de covarianzas $\boldsymbol{\Sigma}_x^r$ y seguiremos con el paso 4.

Esta etapa puede verse como un problema de optimización donde las variables a optimizar son los K coeficientes b_i .Se pueden utilizar métodos tradicionales de optimización. Una interpretación geométrica de esta etapa es que estamos buscando un hiperelipsoide definido por $\boldsymbol{\Sigma}_y$ dentro de S con un volumen máximo, es decir, modificaremos la matriz de covarianzas $\boldsymbol{\Sigma}_y$ hasta que su volumen se ajuste lo máximo posible a la región definida por los límites de especificación $\left[LEI_j, LES_j \right]$. Estos cambios en el volumen de $\boldsymbol{\Sigma}_y$ se realizan cambiando su forma (modificando sus autovalores) pero manteniendo la dirección de sus ejes (autovectores).

La figura 3.5. representa un ejemplo de dos iteraciones consecutivas en el caso de dos dimensiones. Aquí, el rectángulo más grande representa la región especificada S. En la izquierda se representa la iteración inicial con la matriz de covarianzas $\boldsymbol{\Sigma}_y$, en la derecha se muestra la iteración consecutiva con una nueva matriz de covarianzas $\boldsymbol{\Sigma}_y^*$, y las semilongitudes l_j^* correspondientes. En este segundo caso se puede observar que en esta iteración estamos cerca de alcanzar el objetivo.

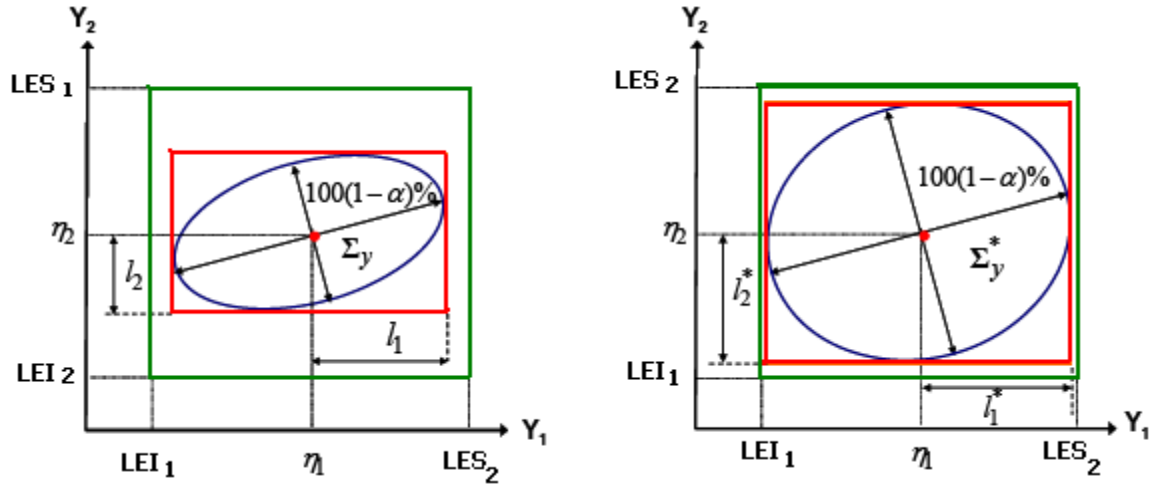


Figura 3.5. (a) Elipse inicial dentro de S. (b) Elipse dentro de S con un volumen mayor obtenida en la siguiente iteración

3.6.2. ETAPA 2

En la etapa 1 hemos obtenido el mejor tamaño para el hiperelipsoide de **X**. En la Etapa 2 mantendremos la forma de este hiperelipsoide pero modificaremos su tamaño para obtener $p = \alpha$.

Un procedimiento simple para obtener esta modificación es aplicando: $\lambda_i^* = b_i \lambda_i$ y $\boldsymbol{\Sigma}_x^* = \mathbf{C}\mathbf{D}^*\mathbf{C}'$ (ecuaciones 3.10 y 3.11) pero usando el mismo valor de *b* para todos los autovalores. Este procedimiento es equivalente a multiplicar la matriz de covarianzas de **X** por b.

Denotaremos como $\boldsymbol{\Sigma}_x^R$ a la matriz de covarianzas obtenida al final de la Etapa 1. Los siguientes pasos nos ayudaran a lograr las tolerancias óptimas:

En esta etapa se calculara el porcentaje de piezas defectuosas contando el número de ellas que quedan fuera de las especificaciones de **Y**, por lo tanto, si alguna de estas especificaciones fuera únicamente unilateral, es decir, si una variable Y estuviera definida únicamente superior o inferiormente, a la hora de calcular el porcentaje de piezas que se que queda fuera de las especificaciones se consideraría solo de forma unilateral. Durante la Etapa 1 los límites de especificación de **Y** siempre eran considerados bilaterales ya que se consideraba un rectángulo de especificaciones que debía ser igual al rectángulo proyectado de la elipse. También por este motivo el cálculo del porcentaje de piezas defectuosas es más preciso en esta segunda etapa.

1. Pondremos el contador r=0.
2. Multiplicaremos la matriz de covarianzas obtenida al final de la Etapa 1 por un valor inicial b^r: $\boldsymbol{\Sigma}_x^r = b^r \boldsymbol{\Sigma}_x^R$
3. Calcularemos las tolerancias de **X**, $\left[LTI_i^r, LTS_i^r\right], i = 1,...,K$ usando la proyección del hiperelipsoide generado por la matriz de covarianzas obtenida en el paso anterior, $t_i = \sqrt{\left(\boldsymbol{\Sigma}_x^{-1}\right)_{ii} \chi^2_{K,1-\alpha}}$ y $X_i \in \left[LTI_i, LTS_i\right] = \left[\mu_i - t_i, \mu_i + t_i\right]$ (ecuaciones 3.6 y 3.7)
4. Generaremos B réplicas de una normal $N_K \left(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x^r\right)$ y seleccionaremos los valores de $\mathbf{X}_T^r$ dentro de las tolerancias $\left[LTI_i^r, LTS_i^r\right]$
5. Obtendremos $\mathbf{Y}_T^r = f\left(\mathbf{X}_T^r\right)$ y calcularemos la proporción p^r de valores fuera de los límites de especificación.
6. Si $\left(\alpha - p^r\right) > \varepsilon$, con un valor pequeño de ε, pondremos el contador a r=r+1, pondremos también un nuevo valor de b^r y volver al paso 2 de esta etapa. De otra manera la búsqueda estaría finalizada y las tolerancias serian $\left[LTI_i^r, LTS_i^r\right]$ correspondientes a la iteración en la que nos encontremos.

Capítulo 4

EJEMPLOS DE APLICACIÓN DE LA METODOLOGÍA UTILIZANDO EL PROGRAMA DE CÁLCULO MATLAB

CAPÍTULO 4. APLICACIÓN DE LA METODOLOGÍA A TRES EJEMPLOS

En este capítulo se presentarán 3 ejemplos para ilustrar la metodología propuesta. En cada ejemplo se plantearán, en primer lugar, los datos de partida y luego se aplicará la metodología propuesta, resolviendo cada ejemplo con la toolbox de optimización de Matlab.

4.1. EJEMPLO 1 ASIGNACIÓN DE TOLERANCIAS A UN CONJUNTO MECÁNICO

4.1.1. PLANTEAMIENTO DEL PROBLEMA

Este ejemplo ilustra el diseño de tolerancias para un conjunto mecánico. Este ejemplo fue propuesto por Lee y Woo (1990) y Lee y Johnson (1993). El conjunto mecánico está compuesto por 2 piezas como se muestra en la figura 4.1. cuya relación entre las diferentes variables es lineal.

Las variables **X** del problema son un conjunto de longitudes de la pieza como se muestra en la figura 3.6. Estas variables son 8 y se denotan como $\mathbf{X} = (X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)'$.

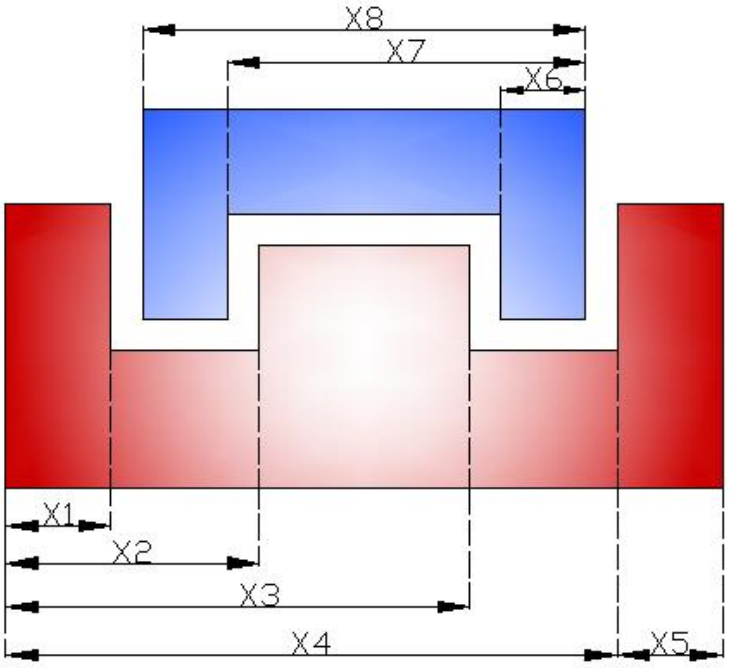


Figura 4.1. Conjunto mecánico de 2 piezas

Las especificaciones de diseño para este conjunto mecánico se definen en término de una serie de variables **Y**, denotadas como $\mathbf{Y} = (Y_1, Y_2, Y_3, Y_4)'$. La variable Y_1 define la condición que debe cumplir el tamaño de la base del conjunto mecánico. El resto de variables Y_2, Y_3, Y_4 definen el espacio libre u holgura que debe quedar entre las dos piezas. La relación $Y = f(X)$ está definida como:

$$
\begin{aligned}
Y_1 &= X_4 + X_5 \\
Y_2 &= X_2 - X_1 - X_8 + X_7 \\
Y_3 &= X_7 - X_6 - X_3 + X_2 \\
Y_4 &= X_4 - X_3 - X_6
\end{aligned}
$$

De donde se deduce que la matriz **A** puede ser escrita como:

$$
\mathbf{A} = \begin{bmatrix}
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 \\
0 & 0 & -1 & 1 & 0 & -1 & 0 & 0
\end{bmatrix}
$$

Los límites de especificación requeridos para las variables **Y** son:

$$
\begin{aligned}
Y_1 &\leq 127.13mm \\
Y_2 &\geq 0.0076mm \\
Y_3 &\geq 0.0254mm \\
Y_4 &\geq 0.0076mm
\end{aligned}
$$

Los valores nominales de las variables **X** vienen especificados por condiciones de diseño y son los siguientes:

$$\mathbf{VN}_X = (25.4,\ 50.8,\ 76.2,\ 101.6,\ 25.4,\ 25.3,\ 50.8,\ 76.1)'$$

Como se explicó antes, estos valores nominales serán las medias de las variables, ya que se considera que el proceso está centrado en esos valores, Por tanto:

$$\boldsymbol{\mu}_X = (25.4,\ 50.8,\ 76.2,\ 101.6,\ 25.4,\ 25.3,\ 50.8,\ 76.1)'$$

En el caso de la matriz de covarianzas de **X**, ésta debería estimarse con datos reales del proceso bajo control. En este caso al ser un ejemplo ficticio, tenemos que simular una matriz de covarianzas, es decir, generar una matriz de covarianzas aleatoria.

En este ejemplo, al existir dos partes individuales en el conjunto mecánico, se deben considerar los dos grupos de variables correladas. El primer grupo corresponde a las variables X_1, X_2, X_3, X_4, X_5 que definen la pieza de arriba, y el segundo grupo corresponde a las variables X_6, X_7, X_8 que definen la pieza de abajo.

Así, en primer lugar se genera una matriz de covarianzas para cada grupo de variables y luego se construye la matriz total de covarianzas. El bloque de la matriz resultante Σ_x que será usada en este ejemplo es la siguiente:

$$\mathbf{\Sigma}_x = \begin{bmatrix} 0.0105 & 0.0080 & 0.0065 & 0.0040 & 0.0107 & 0 & 0 & 0 \\ 0.0080 & 0.0071 & 0.0050 & 0.0031 & 0.0083 & 0 & 0 & 0 \\ 0.0065 & 0.0050 & 0.0048 & 0.0024 & 0.0065 & 0 & 0 & 0 \\ 0.0040 & 0.0031 & 0.0024 & 0.0066 & 0.0039 & 0 & 0 & 0 \\ 0.0107 & 0.0083 & 0.0065 & 0.0039 & 0.0454 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0024 & 0.0009 & 0.0018 \\ 0 & 0 & 0 & 0 & 0 & 0.0009 & 0.0005 & 0.0007 \\ 0 & 0 & 0 & 0 & 0 & 0.0018 & 0.0007 & 0.0016 \end{bmatrix}$$

El objetivo del problema es asignar las mayores tolerancias de **X**, de manera que la proporción de defectuosas sea menor o igual que 0.0027, es decir $p \leq \alpha = 0.0027$. Luego la función objetivo es

$$FO: \max\ t_i;\ i = 1,2,...,K = \max \sum_{i=1}^{K} t_i$$

sujeto a: $p \leq \alpha$

Con el fin de poder resolver el problema de optimización, además de ésta restricción se considerará una restricción adicional, que las tolerancias de X deben ser superiores a 0.01mm, es decir, $t_i \geq 0.01;\ i = 1,2,...,8$ De esta manera evitamos que las tolerancias tomen valores iguales o muy próximos a cero.

A continuación se describirán los pasos necesarios para obtener las tolerancias óptimas de las variables $\mathbf{X} = (X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)'$ siguiendo el procedimiento que se ha desarrollado durante este capítulo

4.1.2. APLICACIÓN DE LA METODOLOGÍA

4.1.2.1. Etapa 1

1. Se realiza el análisis de componentes principales de la matriz de covarianzas de **X** de dimensión 8x8:

$$\boldsymbol{\Sigma}_X = \mathbf{CDC}'$$

donde **C**=matriz de autovectores (8x8) y **D**=matriz diagonal de autovalores (8x8)

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0.7126 & -0.1440 & 0 & -0.2238 & 0.5710 & 0.3088 \\ 0 & 0 & -0.3941 & 0.7272 & 0 & -0.2049 & 0.4638 & 0.2426 \\ 0 & 0 & -0.5795 & -0.6711 & 0 & -0.1885 & 0.3763 & 0.1919 \\ 0 & 0 & -0.0322 & -0.0103 & 0 & 0.9331 & 0.3348 & 0.1267 \\ 0 & 0 & -0.0103 & -0.0021 & 0 & 0.0412 & -0.4530 & 0.8905 \\ 0.5678 & 0.3476 & 0 & 0 & 0.7462 & 0 & 0 & 0 \\ -0.7499 & 0.5922 & 0 & 0 & 0.2948 & 0 & 0 & 0 \\ -0.3394 & -0.7270 & 0 & 0 & 0.5969 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_8 \end{bmatrix} = \begin{bmatrix} 0.0001 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0002 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0005 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0008 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0042 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0046 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.0151 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0533 \end{bmatrix}$$

2. El siguiente paso consistiría en obtener el vector de medias y la matriz de covarianzas de **Y** usando las ecuaciones 3.4. y 3.5 :

$$\boldsymbol{\eta}_Y = a + \mathbf{A}\boldsymbol{\mu}_X = 0 + \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 25.4 \\ 50.8 \\ 76.2 \\ 101.6 \\ 25.4 \\ 25.3 \\ 50.8 \\ 76.1 \end{bmatrix} = \begin{bmatrix} 127 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$$

Luego, el vector de medias es $\boldsymbol{\eta}_Y = (127,\ 0.1,\ 0.1,\ 0.1)'$

La matriz de covarianzas inicial de **Y** será igual a:

$$\boldsymbol{\Sigma}_y = \mathbf{A}\boldsymbol{\Sigma}_x\mathbf{A}'$$

$$\boldsymbol{\Sigma}_y = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.0102 & 0.0034 & 0.0066 & 0.0078 & 0.0107 & 0 & 0 & 0 \\ 0.0034 & 0.0020 & 0.0022 & 0.0025 & 0.0036 & 0 & 0 & 0 \\ 0.0066 & 0.0022 & 0.0051 & 0.0050 & 0.0068 & 0 & 0 & 0 \\ 0.0078 & 0.0025 & 0.0050 & 0.0108 & 0.0086 & 0 & 0 & 0 \\ 0.0107 & 0.0036 & 0.0068 & 0.0086 & 0.0433 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0027 & 0.0010 & 0.0007 \\ 0 & 0 & 0 & 0 & 0 & 0.0010 & 0.0005 & 0.0002 \\ 0 & 0 & 0 & 0 & 0 & 0.0007 & 0.0002 & 0.0004 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & -1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_y = \begin{bmatrix} 0.0598 & -0.0033 & 0.0025 & 0.0016 \\ -0.0033 & 0.0023 & 0.0013 & 0.0015 \\ 0.0025 & 0.0013 & 0.003 & 0.002 \\ 0.0016 & 0.0015 & 0.002 & 0.009 \end{bmatrix}$$

3. Obtendremos las semilongitudes del hiperrectángulo definido por el hiperelipsoide definido por $\boldsymbol{\eta}_y$ y $\boldsymbol{\Sigma}_y$

$$l_j = \sqrt{\left(\boldsymbol{\Sigma}_y^{-1}\right)_{jj} \chi^2_{m,1-\alpha}}$$

Los lados de este hiperrectángulo, denotados por $\left[LI_j, LS_j\right]$ se obtienen, tal y como se ha definido anteriormente en la ecuación 3.7:

$$LI_j = \eta_j - l_j$$

$$LS_j = \eta_j + l_j$$

A continuación se representa este procedimiento simplificado en dos dimensiones, para poder ir haciéndonos una idea más gráfica de la metodología, ya que al tener más de tres variables el ejemplo que estamos desarrollando sería muy complicado de representar.

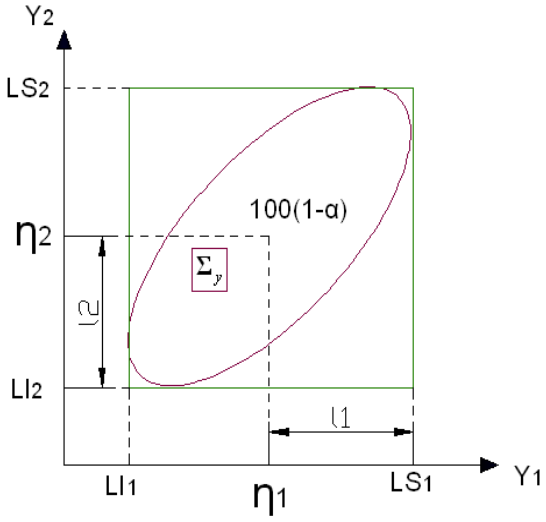


Figura 4.2. Elipse y rectángulo definidos por $\boldsymbol{\eta}_y$ y $\boldsymbol{\Sigma}_y$

4. Cuando el hiperrectángulo calculado en el paso anterior sea igual al hiperrectangulo de especificaciones (figura 4.3), es decir:

$$\left[LI_j, LS_j\right] = \left[LEI_j, LES_j\right] \; j = 1,...,4$$

el porcentaje de piezas defectuosas *p* será aproximadamente igual a α .

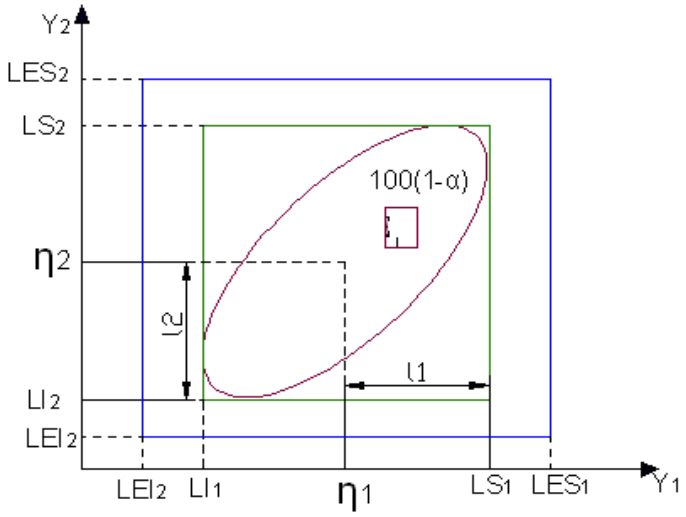


Figura 4.3. Comparación del rectángulo proyectado con el rectángulo formado por los límites de especificación,

Si ambos rectángulos son iguales pasamos a la Etapa 2, con la matriz de covarianzas de **X** obtenida en la última iteración. De otro modo seguiremos con el paso 5.

Tal y como se puede observar en las figuras 4.2 y 4.3 para comparar las longitudes del hiperrectangulo definido por $\boldsymbol{\eta}_y$ y $\boldsymbol{\Sigma}_y$ con el formado con los límites de especificación, estamos considerando que existen los límites de especificación tanto inferior como superior para cada variable para poder calcular el porcentaje de piezas que se salen de las especificaciones, en el ejemplo que estamos analizando, los datos iniciales del problema solo nos imponen restricciones unilaterales paras las especificaciones de **Y**, por lo que deberemos calcular unos límites virtuales superiores o inferiores respectivamente centrados en el valor nominal de **Y** con los que podamos acotar el rectángulo acotado por los límites de especificación y así poder compararlo con la proyección de la elipse definida por $\boldsymbol{\eta}_y$ y $\boldsymbol{\Sigma}_y$, obteniendo así el porcentaje de piezas defectuosas. Para ello se realizara lo siguiente:

Los datos iniciales del problema nos definen los límites de especificación de **Y** de la siguiente manera:

$$Y_1 \leq 127.13mm$$
$$Y_2 \geq 0.0076mm$$
$$Y_3 \geq 0.0254mm$$
$$Y_4 \geq 0.0076mm$$

Estos límites se podrían escribir como la media a la que se suma o se resta respectivamente el límite de especificación, y si quisiéramos que hubiera límite tanto inferior como superior bastaría con dividir la especificación que tenemos entre dos para que esté centrada en la media:

$$Y_1 = 127 + 0.13mm \Rightarrow \frac{0.13}{2} = 0.065 \Rightarrow Y_1 = 127 \pm 0.065$$
$$Y_2 = 0.1 - 0.0924mm \Rightarrow \frac{0.0924}{2} = 0.0462 \Rightarrow Y_2 = 0.1 \pm 0.0462$$
$$Y_3 = 0.1 - 0.0746mm \Rightarrow \frac{0.0746}{2} = 0.0373 \Rightarrow Y_3 = 0.1 \pm 0.0373$$
$$Y_4 = 0.1 - 0.0924mm \Rightarrow \frac{0.0924}{2} = 0.0462 \Rightarrow Y_4 = 0.1 \pm 0.0462$$

Por lo tanto los límites de especificación inferior y superior de las variables Y serán:

$$[LEI_1, LES_1] = [126.935, 127.065]$$
$$[LEI_2, LES_2] = [0.0538, 0.1462]$$
$$[LEI_3, LES_3] = [0.0627, 0.1373]$$
$$[LEI_4, LES_4] = [0.0538, 0.1462]$$

5. En el caso de que los dos rectángulos no sean iguales deberemos obtener una nueva matriz de covarianzas de **X**, $\mathbf{\Sigma}_x^r$, mediante la modificación de los autovalores que hemos obtenido con la aplicación del análisis de componentes principales realizado en el paso 2. Los nuevos valores serán $\lambda_i^* = b_i \lambda_i$. Con éstos valores podremos obtener una nueva matriz $\mathbf{D}^r$ y en consecuencia la nueva matriz de covarianzas de **X**.

$$\lambda_1^* = \lambda_1 b_1$$
$$\lambda_2^* = \lambda_2 b_2$$
$$\lambda_3^* = \lambda_3 b_3$$
$$\lambda_4^* = \lambda_4 b_4$$
$$\lambda_5^* = \lambda_5 b_5$$
$$\lambda_6^* = \lambda_6 b_6$$
$$\lambda_7^* = \lambda_7 b_7$$
$$\lambda_8^* = \lambda_8 b_8$$

Con los autovalores modificados obtenemos la nueva matriz $\mathbf{D}^r$:

$$\mathbf{D}^r = \begin{bmatrix} \lambda_1^* & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_2^* & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3^* & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_4^* & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda_5^* & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_6^* & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_7^* & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_8^* \end{bmatrix}$$

A través de esta nueva matriz $\mathbf{D^r}$ obtendremos la nueva matriz de covarianzas de **X**, $\mathbf{\Sigma}_x^r$, mediante, $\mathbf{\Sigma}_x^r = \mathbf{C}\mathbf{D}^r\mathbf{C}'$.

En el caso del ejemplo que se está analizando los valores de los coeficientes b_i obtenidos al final de la optimización de la etapa 1 son:

$$b_1 = 0.0010$$
$$b_2 = 0.2610$$
$$b_3 = 0.4552$$
$$b_4 = 0.1086$$
$$b_5 = 0.1609$$
$$b_6 = 0.0088$$
$$b_7 = 0.1095$$
$$b_8 = 0.0177$$

Con estos valores de los coeficientes b_i obtenemos la nueva matriz $\mathbf{D^r}$:

$$\mathbf{D}^r = \begin{bmatrix} 0.0001 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0002 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0005 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0008 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0041 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0046 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.0152 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0533 \end{bmatrix}$$

Con lo que la nueva matriz de covarianzas de **X**, que será la matriz inicial de covarianzas de la Etapa 2 sería igual a:

$$\boldsymbol{\Sigma}_{\mathbf{x}}^{'} = \begin{bmatrix} 0.0007 & 0.0004 & 0.0003 & 0.0003 & -0.0002 & 0 & 0 & 0 \\ 0.0004 & 0.0005 & 0.0003 & 0.0003 & -0.0001 & 0 & 0 & 0 \\ 0.0003 & 0.0003 & 0.0004 & 0.0002 & -0.0001 & 0 & 0 & 0 \\ 0.0003 & 0.0003 & 0.0002 & 0.0002 & -0.0001 & 0 & 0 & 0 \\ -0.0002 & -0.0001 & -0.0001 & -0.0001 & 0.0011 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0004 & 0.0002 & 0.0003 \\ 0 & 0 & 0 & 0 & 0 & 0.0002 & 0.0001 & 0.0001 \\ 0 & 0 & 0 & 0 & 0 & 0.0003 & 0.0001 & 0.0003 \end{bmatrix}$$

4.1.2.2. Etapa 2

En la Etapa 2 se modificará la matriz $\boldsymbol{\Sigma}_x$ mediante la modificación homogénea de todos los autovalores, es decir: $\lambda_i^* = b\lambda_i$ y $\boldsymbol{\Sigma}_x^* = \mathbf{C}\mathbf{D}^*\mathbf{C}'$ (se usa el mismo valor de *b* para todos los autovalores)

Este procedimiento es equivalente a multiplicar la matriz de covarianzas de **X** por b.

Denotaremos como $\boldsymbol{\Sigma}_x^R$ a la matriz de covarianzas obtenida al final de la Etapa 1.

Los siguientes pasos permiten obtener las tolerancias óptimas:

1. Iniciamos la etapa con la matriz $\Sigma_x^r = b\Sigma_x^R$, siendo en este caso *b*=1.

2. Calculamos las tolerancias de **X**, $\left[LTI_i^r, LTS_i^r\right], i = 1,...,K$ usando la proyección del hiperelipsoide generado por la matriz de covarianzas obtenida en el paso anterior:

$$t_i = \sqrt{\left(\boldsymbol{\Sigma}_x^{-1}\right)_{ii} \chi^2_{K,1-\alpha}}$$

$$X_i \in \left[LTI_i, LTS_i\right] = \left[\mu_i - t_i, \mu_i + t_i\right]$$

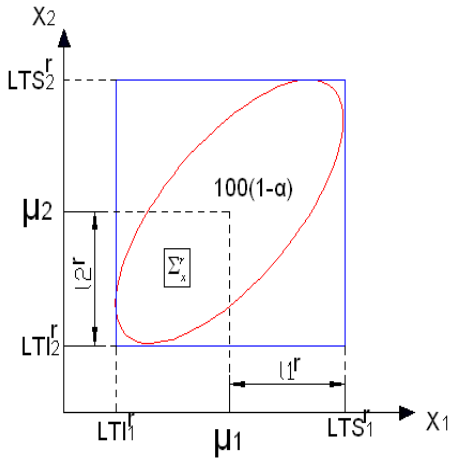


Figura 4.4.Hiperelipsoide e hiperrectángulo definidos por $\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x^r$

El siguiente paso consistirá en generar $B = 10^6$ réplicas de una normal $N_K \approx \left(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x^r\right)$ y seleccionaremos los valores de $\mathbf{X}_T^r$ dentro de las tolerancias $\left[LTI_i^r, LTS_i^r\right]$.

3. Obtendremos $\mathbf{Y}_T^r = f\left(\mathbf{X}_T^r\right)$ y calcularemos la proporción p^r de valores fuera de los límites de especificación de **Y**. En este caso los límites de especificación serán los unilaterales, es decir, los límites reales del problema.

4. Si $\left(\alpha - p^r\right) > \varepsilon$, con un valor $\varepsilon = 5 \cdot 10^{-5}$, realizaremos una nueva iteración, poniendo un nuevo valor de b^r y volveremos al paso 2 de esta etapa.
   De otra manera la búsqueda estaría finalizada y las tolerancias serian $\left[LTI_i^r, LTS_i^r\right]$ correspondientes a la iteración en la que nos encontremos.

4.1.3. SOLUCIÓN DEL PROBLEMA UTILIZANDO EL PROGRAMA DE CÁLCULO MATLAB

A continuación se presentan de forma detallada los programas de MATLAB utilizados para resolver el problema planteado arriba. En cada programa, se marcarán en rojo las líneas de comando que será necesario modificar dependiendo del problema que se esté resolviendo, y por lo tanto que se deberán modificar si queremos aplicar la metodología a cualquier otro problema que se nos plantee, además los caracteres que aparecen con letra de color verde no pertenecen al programa, si no que son comentarios añadidos para poder llegar a un mejor entendimiento del mismo.

En este ejemplo se escribirán los programas completos, pero tanto para este ejemplo como en los siguientes en los que solo se mostraran los pasos mas importantes podemos consultar los programas completos en los anexos del trabajo.

En las imágenes del programa que se mostraran en este capitulo podemos ver por un lado los programas escritos en el editor de Matlab, que podemos reconocer porque a su izquierda aparece el numero de la línea del programa en la que nos encontramos, y por otro lado algunos de los resultados que se pueden obtener si hacemos que Matlab ejecute el programa que hemos creado.

4.1.3.1.Etapa 1

Los programas que se escriben a continuación para la resolución de la etapa 1 de la metodología descrita en el proyecto son para la utilización de una función definida en el propio Matlab que se llama fmincon. Esta función se utiliza para encontrar un mínimo ajustado de una función de varias variables. La función fmincon intenta resolver problemas de forma que dada una función determinada encuentra un mínimo sujeto a una serie de restricciones que se encuentre dentro de unos límites determinados.

min F(X) subject to: A*X <= B, Aeq*X = Beq (restriciones lineales)

X C(X) <= 0, Ceq(X) = 0 (restricciones no lineales)

LB <= X <= UB (límites)

Dentro de la propia función fmincon de Matlab podemos elegir entre cuatro tipos de implementación, además, deberemos definir una serie de características de la forma:

X = FMINCON(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON,OPTIONS)

Donde FUN será la función objetivo que se quiere optimizar

X0 es el punto inicial para el comienzo de las iteraciones

A, B definen la inecuación lineal A*X <= B. que define la función objetivo

Aeq,Beq definen la ecuación lineal Aeq*X = Beq que se debe cumplir para que la solución sea aceptada

LB,UB definen una serie de limites inferiores y superiores para las variables X

NONLCON representa las ecuaciones e inecuaciones no lineales respectivamente C(X) <= 0 y Ceq(X) = 0.

OPTIONS los parámetros de optimización se sustituirán por otros creados con la función OPTIMSET

Cada vez que se requiera utilizar la función fmincon en Matlab se deberán definir cuales de estas características son necesarias en función de si la función es o no lineal, si tiene limites definidos etc., si no necesitáramos utilizar alguna de estas características escribiríamos [ ], en el lugar correspondiente.

Además se debe definir como queremos que la función muestre los resultado, en este caso se ha elegido lo siguiente:

[X, FVAL, EXITFLAG, OUTPUT] = FMINCON (FUN, X0, ...)

Donde X es la variable a optimizar

FVAL nos muestra el valor final obtenido para la función objetivo

EXITFLAG nos da la posibilidad de ver como se han dado las iteraciones, lo que pues ser de utilidad en el caso de que no se encuentre una solución optima.

OUTPUT nos muestra el numero total de iteraciones necesarias hasta obtener la solución final.

Para el caso se va a desarrollar la función fmincon se define de la siguiente manera:

```
[x,FVAL,EXITFLAG,output]=fmincon(@funcionobjetivo1,puntoinicial,[],[],[],[],
    bimin,[],@definicion_de_restricciones,options);
```

Con esto conseguiremos que la función nos muestre una estructura de salida con diferente información, como el numero de iteraciones que se han necesitado para encontrar la solución óptima y el valor final de la función objetivo que hemos definido, además deberemos definir únicamente la función objetivo y las restricciones del problema, que se harán en programas diferentes, el punto inicial y el valor mínimo de aumento de las iteraciones. Deberemos definir también los parámetros de optimización:

```
options=optimset('LargeScale','off');
puntoinicial=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];
bimin=[0.001;0.001;0.001;0.001;0.001;0.001;0.001;0.001];
```

A continuación se muestran los programas empleados para la solución de la etapa 1 de la metodología:

Primeramente se muestra el programa perteneciente a la etapa 1 para la generación de la matriz de covarianzas. Este programa no se utiliza de manera directa durante la etapa 1, sino que, tal y como se ha explicado a lo largo del trabajo, necesitamos contar con una matriz de covarianzas en los datos de partida, que en el caso de los ejemplos que se han desarrollado no podemos obtenerla directamente del proceso de fabricación por lo que se generará de manera aleatoria

Se generarán datos aleatorios de distribuciones normales multivariantes para generar la matriz de covarianzas de las variables X. Se generaran tantas matrices de covarianzas como piezas tenga el ensamblaje, además la dimensión de cada una de estas matrices vendrá determinada por el numero de variables correladas que existan en cada pieza.

En el ejemplo que se está desarrollando el programa para generar la matriz de covarianzas se desarrollaría de la siguiente manera:

```
% Este programa generara datos aleatorios de
% distribuciones normales multivariantes
% En este ejmplo el enamblaje consta de dos piezas,
% por lo que se generarán dos matrices de covarianzas:
% una de 5x5 y otra de 2x2


dim_pieza_sup=5; % Adaptamos la matriz al numero de variables correladas
                 % que definen la pieza inferior, 5 variables
dim_pieza_inf=3; % Adaptamos la matriz al numero de variables correladas
                 % que definen la pieza superior, 3 variables

m=1000; %Definimos un numero elevado de iteraciones

b1=[0.1 0.1 0.1 0.1 0.1]; % Valores para el comienzo de la iteración.
b2=[0.1 0.1 0.1]; % Valores para el comienzo de la iteración.

x1=zeros(m,dim_pieza_sup); % Generamos dos matrices de ceros,
                           % cada una de ellas de m filas
x2=zeros(m,dim_pieza_inf); % y columnas acordes con las dimensiones
                           % que se han definido anteriormente
%-------------------------------------------------------
```

```
25      %-------------------------------------------------------
26
27 -    x1(:,1)=randn(m,1)*b1(1);
28
29 -    for i=2:dim_pieza_sup
30 -        x1(:,i)=rand*x1(:,1)+randn(m,1)*b1(i)*0.5;
31 -    end
32 -    Matcorr1=corrcoef(x1);
33 -    MatCov1=cov(x1)
34 -    [autovectores1,au21]=eig(MatCov1);
35 -    Sigma1=sqrt(diag(MatCov1));
36 -    Propau1=sort(diag(au21)./sum(diag(au21))*100,'descend');
37
38      %-------------------------------------------------------
39 -    x2(:,1)=randn(m,1)*b2(1);
40
41 -    for i=2:dim_pieza_inf
42 -        x2(:,i)=rand*x2(:,1)+randn(m,1)*b2(i)*0.3;
43 -    end
44 -    Matcorr2=corrcoef(x2);
45 -    MatCov2=cov(x2)
46 -    [autovectores2,au22]=eig(MatCov2);
47 -    Sigma2=sqrt(diag(MatCov2));
48 -    Propau2=sort(diag(au22)./sum(diag(au22))*100,'descend');
49
50      % Estos dos programas generan las dos matrices de covarianzas iniciales de
51      % x que se utilizaran el los posteriores programas.
52
53      %-------------------------------------------------------
```

```
56      %-------------------------------------------------------
57
58 -    save CovIniciales MatCov1 MatCov2 % Por último guardamos las dos
59                                        % matrices que hemos generado
60                                        % para utilizarlas posteriormente
61
62      %-------------------------------------------------------
```

A continuación se muestra el programa que nos define la función objetivo de nuestro problema, en el definirá la función objetivo que se va a desarrollar, es decir, tal y como se ha definido a lo largo del proyecto lo que se busca es maximizar las tolerancias:

$$FO: \max\ t_i;\ i = 1,2,...,K = \max \sum_{i=1}^{K} t_i$$

sujeto a: $p \le \alpha$

Durante la etapa 1, cuando la función fmincon necesite utilizar la función objetivo llamara al programa que se describe a continuación:

```
[x,FVAL,EXITFLAG,output]=fmincon(@funcionobjetivo1,puntoinicial,[],[],[],[],
    bimin,[],@definicion_de_restricciones,options);
```

```
function f=funcionobjetivo1(x)

%En este programa se define la funcion objetivo

bi=x;  % valores por los que iremos multiplicando los autovectores para
       % modificarlos hasta conseguir los valores óptimos que cumplan la
       % función objetivo que estamos definiendo

p=0.9973; % definimos el porcentaje de piezas defectuosas que consideramos
          % aceptable

load CovIniciales % Cargamos el programa que hemos realizado antes para
                  % generar la matriz de covarianzas


covarianzasx=zeros(8,8); % la dimensión de la matriz de covarianzas sera la
                         % correspondiente al numero de variables X totales
                         % que tiene el ensamblaje

covarianzasx(1:5,1:5)=MatCov1; % dependiendo del numero de variables corre-
covarianzasx(6:8,6:8)=MatCov2; % ladas que tengamos obtendremos una
                               % matriz de covarianzas total,utilizando las
                               % dos matrices que se han obtenido en el
                               % programa anterior para generar la matriz
                               % de covarienzas

%-------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------
a=length(covarianzasx);

[C, autov]=eig(covarianzasx); % Obtiene los autovectores y los
% autovalores de la matriz d covarianzas de x

D=diag(autov);% Con los autovalores obtenidos en el comando anterior
% generamos una matriz D, diagonal de autovalores de la matriz
% de covarianzas

newD=bi.*D; % Obtenemos una nueva matriz D modificando los autovalores
%multiplicandola por los factores bi
```

```
newcovarianzasx=C*diag(newD)*C'; % con la nueva matriz D obtenemos una
% nueva matriz de covarianzas de x

proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)); % calculamos la
% proyección de la diagonal de la matriz de covarianzas de x que
% estos son los valores ti

f=-sum(proyecx); % la función objetivo sera buscar la minima diferencia
% entre las proyecciones de la matriz de covarianzas de x y las tolerancias
% de x

%--------------------------------------------------------------------------

save NewX newcovarianzasx
```

A continuación se muestra el programa que nos define las restricciones del problema, estas restricciones implican que no se superen las especificaciones de Y y que las tolerancias de X sean mayores que las tolerancias mínimas que se han definido para X. Además se comprueba que el determinante de las nuevas matrices de covarianzas de X e Y sea mayor que cero.

Al igual que para el programa anterior, este programa lo utiliza la función fmincon cuando necesita comprobar las restricciones del problema

```
[x,FVAL,EXITFLAG,output]=fmincon(@funcionobjetivo1,puntoinicial,[],[],[],[],
    bimin,[],@definicion_de_restricciones,options);
```

```
function [restricciones,nleq]=definicion_de_restricciones(x)

%Aquí se definen las restricciones del problema

load Newx  % esta leyendo newcovarianzasx

bi=x;
p=0.9973;

a=length(newcovarianzasx); % tamaño de la matriz de covarianzas de x

A=[0 0 0 1 1 0 0 0;-1 1 0 0 0 0 1 -1;0 1 -1 0 0 -1 1 0;0 0 -1 1 0 -1 0 0];
% Matrz de relaciones defindа por las especificaciones del montaje mecánico
% (Y=f(X))

tolminx=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];
% tolerancias mínimas para x

toleranciasy=[0.13;0.0924;0.0746;0.0924];
% toleranciasy =( Valores de Y [(y1<=127.13 , y2>=0.0076 , y3>= 0.0254
% y4>=0.0076)]- valores nominales de Y [127, 0.1 , 0.1, 0.1])

%--------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------
% restricción 1
restriccion1=det(newcovarianzasx); % calculamos el determinante de la nueva
% matriz de covarianzas de x que hemos obtenido

%-------------------------------------------------------------------------
% restricción 2
proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)); % calculamos la
% proyección de la matriz de varianzas de x que acabamos de obtener

restriccion2=-(proyecx-tolminx); % calculamos la diferencia entre la
% proyección de la matriz de varianzas de x y las tolerancias minimas de x
% que hemos definido

%-------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------
% restricción 3
newcovarianzasy=A*newcovarianzasx*A'; %obtenemos la nueva matriz de
% covarianzas de y a traves de la nueva matriz de covarianzas de x que
%hemos obtenido en el paso anterior

b=length(newcovarianzasy); % tamaño de la matriz de covarianzas de y
proyecy=sqrt(chi2inv(p,b)*diag(newcovarianzasy));% calculamos la proyección
% de la matriz de varianzas de y

restriccion3=proyecy-toleranciasy; % calculamos la diferencia entre la
% proyección de la matriz de varianzas de y las tolerancias de y

%-------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------
restricciones=[-restriccion1;restriccion2;restriccion3];
nleq=[];
% Las restricciones del problema serán: que no se superen las tolerancias
% de y ( utilizamos diferencias d y), que las tolerencias de x sean mayores
% que las tolerancias minima de x que hemos definido (utilizamos
% diferenciasx) y que los determinantes de las matrices de covarianzas de x
% e y sean >0 (utilizamos restriccion1 y restriccion2)

%-------------------------------------------------------------------------
```

Por último se muestra el programa base con el que se realizará la primera etapa de la metodología, en el que se hace uso de la función fmincon de Matlab, que llamará para su utilización a los programas que se han mostrado antes

```
% Se quiere asignar tolerancias a 8 variables para cumplir las
% restricciones de 4 variables Y. Etapa1
%-------------------------------------------------------------------------
options=optimset('LargeScale','off');
puntoinicial=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];
bimin=[0.001;0.001;0.001;0.001;0.001;0.001;0.001;0.001];

% Para realizar la optimización debemos definir un punto inicial en el que
% comenzará la optimización y un valor mínimo de aumento en cada iteración

[x,FVAL,EXITFLAG,output]=fmincon(@funcionobjetivo1,puntoinicial,[],[],[],[],...
    bimin,[],@definicion de restricciones,options);

%Se define la variable a optimizar, en este caso x, la función de
%optimización que será la que hemos definido con
%anterioridad en funcionobjetivo, el punto inicial, el valor mínimo y las
%restricciones que seran las definidas con anterioridad en
%definicion_de_restricciones

% Ejecuta el algoritmo de optimización hasta encontar la solución óptima
```

```
%-------------------------------------------------------------------------

%aquí se lee la solución para ver los valores obtenidos, se calcularan los
%valores de las matrices de covarianzas de x e y óptimos

bi=x;  % valores óptimos de bi

p=0.9973;
alpha=1-p;

load CovIniciales % Se leen las matrices de covarianzas y se unen en una
                  % sola de 8x8
covarianzasx=zeros(8,8);
covarianzasx(1:5,1:5)=MatCov1;
covarianzasx(6:8,6:8)=MatCov2;

a=length(covarianzasx); % tamaño de la matriz de covarianzas de x

A=[0 0 0 1 1 0 0 0;-1 1 0 0 0 0 1 -1;0 1 -1 0 0 -1 1 0;0 0 -1 1 0 -1 0 0];
% Matriz de relaciones definda por las especificaciones del montaje
% mecánico(Y=f(X)

tolminx=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01]; % es el valor de
% tolerancias mínimas para x

toleranciasy=[0.13;0.0924;0.0746;0.0924];
% toleranciasy =( Valores de Y [(y1<=127.13 , y2>=0.0076 , y3>= 0.0254
% y4>=0.0076)]- valores nominales de Y [127, 0.1 , 0.1, 0.1])

%-------------------------------------------------------------------------
```

```
57 -    [C, autov]=eig(covarianzasx); % Obtiene los autovectores y los
58      % autovalores de la matriz d covarianzas de x
59
60 -    D=diag(autov);% Con los autovalores obtenidos en el comando anterior
61      % generamos una matriz D, diagonal de autovalores de la matriz
62      % de covarianzas
63
64 -    newD=bi.*D; % Obtenemos una nueva matriz D modificando los autovalores
65      %multiplicandola por los factores bi
66
67 -    newcovarianzasx=C*diag(newD)*C'; % con la nueva matriz D obtenemos una
68      % nueva matriz de covarianzas de x
69
70 -    restriccion1=det(newcovarianzasx)
71
72 -    proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)) % calculamos la proyección
73      % de la matriz de varianzas de x que acabamos de obtener
74
75 -    Suma_ti=sum(proyecx)
76
77 -    restriccion2=-(proyecx-tolminx) % Realizamos la comparación entre la
78                                      % proyección y las tolerancias de x
79
80 -    newcovarianzasy=A*newcovarianzasx*A'; % obtenemos la nueva matriz de
81      % covarianzas de y a traves de la nueva matriz de covarianzas de x que
82      %hemos obtenido en el paso anterior
```

```
84 -    b=length(newcovarianzasy); % tamaño de la matriz de varianzas de y
85
86 -    proyecy=sqrt(chi2inv(p,b)*diag(newcovarianzasy)); % calculamos la
87      % proyección de la diagonal de la matriz de covarianzas de y
88
89 -    restriccion3=proyecy-toleranciasy % calculamos la diferencia entre la
90      % proyección de la diagonal de la matriz de covarianzas de y
91      % las tolerancias de y
92
93      %--------------------------------------------------------------------------
```

```
93      %--------------------------------------------------------------------------
94
95 -    save Etapa1 newcovarianzasx bi % Grabamos la solución,
96      %  en este caso será las matrices de covarianzas de x  óptimas para esta
97      % etapa asi como los valores óptimos de bi que se han utilizado.
```

A continuación vemos la matriz de covarianzas de x que se obtiene al final de esta etapa cuando concluye la optimización realizada con la función fmincon y que se utilizará como punto inicial para la etapa 2.

```
newcovarianzasx =

    0.0012    0.0007    0.0007    0.0000    0.0009         0         0         0
    0.0007    0.0005    0.0006    0.0002    0.0003         0         0         0
    0.0007    0.0006    0.0010    0.0005   -0.0001         0         0         0
    0.0000    0.0002    0.0005    0.0003   -0.0005         0         0         0
    0.0009    0.0003   -0.0001   -0.0005    0.0013         0         0         0
         0         0         0         0         0    0.0001    0.0001    0.0001
         0         0         0         0         0    0.0001    0.0001    0.0000
         0         0         0         0         0    0.0001    0.0000    0.0002
```

4.1.3.2. Etapa 2

En esta segunda etapa se realiza el cálculo del porcentaje de piezas defectuosas y se comprueba si este porcentaje es adecuado o no, en este caso la optimización no se realizará mediante una función propia de Matlab como era el caso de la función fmincon en la etapa 1 sino utilizando dos métodos de optimización diferentes que se aplicaran escribiendo sus algoritmos en el propio programa.

En primer lugar se utiliza el método de la bisección, que se basa en el Teorema de Bolzano y en el del Valor Intermedio y los utiliza para aproximar ceros de una función determinada.

Supóngase que queremos encontrar los ceros de una función f(x) continua. dados dos puntos a y b tal que f(a) y f(b) tengan signos distintos, sabemos por el Teorema de Bolzano que f(x) debe tener, al menos, una raíz en el intervalo [a; b]. El método de bisección divide el intervalo en dos, usando un tercer punto c =(a + b)/2. En este momento, existen dos posibilidades: f(a) y f(c), ó f(c) y f(b) tienen distinto signo. El método de bisección se aplica al subintervalo donde el cambio de signo ocurre. Este proceso puede aplicarse tantas veces como sea necesario para alcanzar la precisión que se requiera.

También se utilizará el método de la secante, este es un método que también sirve para encontrar los ceros de una función de forma iterativa.

Es una variación del método de Newton-Raphson donde en vez de calcular la derivada de la función en el punto de estudio, teniendo en mente la definición de derivada, se aproxima la pendiente a la recta que une la función evaluada en el punto de estudio y en el punto de la iteración anterior. Este método, es mas sencillo que el de la bisección ya que solo requiere de 2 puntos al principio, y después el mismo método se va retroalimentando. Lo que hace básicamente es ir tirando rectas secantes a la curva de la ecuación que se tiene originalmente, y va chequeando la intersección de esas rectas con el eje de las X para ver si es la raíz que se busca.

Estos dos métodos de optimización se utilizan en la etapa 2 para encontrar un porcentaje de defectuosos de α=0.0027.

Al igual que pasaba en el caso de la etapa 1, será necesario escribir una función a parte que se utilizará para calcular el porcentaje de defectuosos. Una vez calculado este porcentaje, en el programa de la etapa 2 se comprobará si este valor es válido.

Primeramente se muestra la función que será necesario utilizar a lo largo de esta etapa y que nos cálcula el porcentaje de defectuosos.

```
function [pro_def]=Propdefecto(RE,LEI,LES)

fi=size(LEI,1);
if fi==1
    LEI=LEI';
end

fi=size(LES,1);
if fi==1
    LES=LES';
end


[n,m]=size(RE);
MLTI=ones(n,1)*LEI';
MLTS=ones(n,1)*LES';
esdef=(RE>MLTS)|(RE<MLTI);
if m>1
    Esdef1=(sum(esdef')>0);
else
    Esdef1=esdef;
end
pro_def=mean(Esdef1);
```

Una vez que tenemos esta función podemos proceder al desarrollo de la etapa 2:

```
2
3     % En este prgramas se realiza la Etapa 2.
4     % necesito cargar la solución obtenida en la Etapa 1
5
6 -   load Etapa1  %lee newcovarianzasc bi
7     % las variables que lee son los óptimos: newcovarianzasy newcovarianzasx bi
8
9 -   A=[0 0 0 1 1 0 0 0;-1 1 0 0 0 0 1 -1;0 1 -1 0 0 -1 1 0;0 0 -1 1 0 -1 0 0];
10    % Matrz de relaciones defindа por las especificaciones del montaje mecánico
11    % (Y=f(X))
12
13 -   toleranciasy=[0.13;0.0924;0.0746;0.0924];
14    % toleranciasy =( Valores de Y [(y1<=127.13 , y2>=0.0076 , y3>= 0.0254
15    % y4>=0.0076)]- valores nominales de Y [127, 0.1 , 0.1, 0.1])
16
17 -   bi0=bi; % tomamos como valores iniciales los valores de bi y las matrices
18 -   covx=newcovarianzasx; % de covarianzas de x e y obtenidos en la etapa 1
19
20 -   mediax=[25.4 50.8 76.2 101.6 25.4 25.3 50.8 76.1]'; % Valores nominales
21                                                         % de x en columna
22
23 -   mediay=A*mediax; % (127 0.1 0.1 0.1)% Valores nominales de y
24
25 -   a=length(mediax); % número de variables x
26
27 -   b=length(mediay) ;% número de variables y
28
29 -   h=mediay-toleranciasy; % h=(0.13;0.0924;0.0746;0.0924
30
31    %Las especificaciones de Y son: Y1<=127.13, Y2>=0.01, Y3>=0.025, Y4>=0.01
32 -   LES=[127.13 1000 1000 1000]; % En el caso de tener solo límite inferior
33    % ponemos un valor elevado para el límite superior (caso de y1,y2,y3)
34
35 -   LEI=zeros(1,4); % en el caso de tener solo límite inferior le damos valor
36 -   LEI(2:4)=h(2:4);%0 al límte inferior (caso de y1)
37
38    %--------------------------------------------------------------------------
```

```
37
38    %--------------------------------------------------------------------------
39
40 -   epsilom=0.00005;
41 -   p=0.9973;
42 -   alpha=0.0027;
43 -   FMAX=30.1111;
44 -   FMIN=0.000001;
45
46    % Empieza a iterar para buscar la solución que implique un alfa de 0.0027.
47    % Genera valores aleatorios de X según su Cov y media, los trunca en 99.73%
48    % y con esos valores truncados calcula los valores de Y según Y=F(X).
49    % Estima la proporción de Y fuera de tolerancias por MonteCarlo.
50
51    %--------------------------------------------------------------------------
52
```

```
54 -    bi=1
55 -    newcovarianzasx=bi*covx
56 -    T=zeros(10000,8);
57
58      % En este algoritmo se utiliza el metodo de la bisección
59 -    for j=1:10
60 -        n=10^5;
61 -        proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx));
62
63 -        simX=mvnrnd(mediax,newcovarianzasx,n); %genera valores aleatorios con
64          %distribución de media de x y matriz de covarianzas de x
65 -        Tolsupx=mediax+proyecx;
66 -        Tolinfx=mediax-proyecx;
67
68 -        clear T
69 -        T=(simX>=(ones(n,1)*Tolinfx'))&(simX<=(ones(n,1)*Tolsupx'));
70 -        Tn=double(T);
71 -        Tnp=prod(Tn')';    % Buscamos los valores que estan dentro de las
72                             % tolerancias de x obtenidas como proyección
73 -        indiX=find(Tnp==1);
74 -        simXtrun=simX(indiX,:); % Esta matriz contiene las observaciones x,
75                                  % ya comprobadas
76 -        ntrun=length(simXtrun);
77
78 -        simYtrun=zeros(ntrun,b);
79 -        simYtrun=simXtrun*A'; % Calcula las y para todos los x seleccionados,
80                                % le la función de propdefecto
81 -        [Vpdefec(j)]=Propdefecto(simYtrun,LEI,LES);
82 -    end
```

```
86
87 -    propdefec1=mean(Vpdefec)
88 -    propdefec=propdefec1
89
90
91 -    if propdefec>=alpha
92 -        fmax=bi;
93 -        pmax=propdefec;
94 -        fmin=FMIN;
95 -        pmin=0;
96 -    else
97 -        fmin=bi;
98 -        pmin=propdefec;
99 -        fmax=FMAX;
100 -       pmax=1;
101 -   end
102     % Hasta aqui calcula para matrices finales obtenidas en la última iteración
103     % de la Etapa 1 el porcentaje de defectuosos p
104
105     %--------------------------------------------------------------------------
```

```
106
107     % A partir de aqui se empieza a iterar hasta que el porcentaje de
108     % defectuosas sea alpha
109     % En este algoritmo se utiliza el método de la secante
110
111 -   difer=abs(propdefec-alpha)
112 -   conta=0;
113 -   while difer>epsilom
114 -       conta=conta+1
115 -       if conta<2
116 -           bi=fmin+(fmax-fmin)/(pmax-pmin)*(alpha-pmin);
117 -       else
118 -           bi=(fmin+fmax)/2;
119 -       end
120
121 -       newcovarianzasx=bi*covx;
122 -       for j=1:10
123 -           n=10^5;
124 -           newvarianzasx=diag(newcovarianzasx);
125 -           proyecx=sqrt(chi2inv(p,a)*newvarianzasx);
126
127 -           simX=mvnrnd(mediax,newcovarianzasx,n);
128 -           Tolsupx=mediax+proyecx;
129 -           Tolinfx=mediax-proyecx;
130 -           clear T
131 -           T=(simX>=ones(n,1)*Tolinfx')&(simX<=ones(n,1)*Tolsupx');
132 -           Tn=double(T);
133 -           Tnp=prod(Tn')';
134 -           indiX=find(Tnp==1);
135 -           simXtrun=simX(indiX,:);
136 -           ntrun=length(simXtrun);
137 -           simYtrun=zeros(ntrun,b);
138 -           simYtrun=simXtrun*A';
139 -           [Vpdefec(j)]=Propdefecto(simYtrun,LEI,LES);
140 -       end
141 -       propdefec=mean(Vpdefec);
142
```

```
142 -       if propdefec>=alpha
143 -           fmax=bi;
144 -           pmax=propdefec;
145 -       else
146 -           fmin=bi;
147 -           pmin=propdefec;
148 -       end
149
150 -       difer=abs(propdefec-alpha);
151 -       if (conta>5)&(abs(bi-FMAX)<0.00005);break;end
152 -       if (conta>5)&(abs(bi-FMIN)<0.00005);break;end
153 -       if sqrt(bi)<0.009; bi=0.009^2;break;end
154 -       if sqrt(bi)>25;bi=25.555^2;break;end
155 -       if conta>100;conta,break;end
156
157 -   end
158 -   factorb=bi; % este es el factor de la Etapa 2
159 -   nonconforming=propdefec;
160
161 -   factorfinalCPx=bi*bi0;  %este son los factores de Etapa2*Etapa1
162
163 -   %-------------------------------------------------------------------------
```

```
165
166      %-------------------------------------------------------------------
167 -    newcovarianzasx=covx*bi;  %con esta matriz calculo los límites de tolerancia de X
168 -    proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx));
169
170 -    LTSX=mediax+proyecx;
171 -    LTIX=mediax-proyecx;
172
```

A continuación se muestran los resultados obtenidos al ejecutar el programa en Matlab para la última iteración:

```
nonconforming =

    0.0027


LTSX =

   25.6169
   50.9339
   76.4016
  101.7179
   25.6314
   25.3674
   50.8750
   76.1790


LTIX =

   25.1831
   50.6661
   75.9984
  101.4821
   25.1686
   25.2326
   50.7250
   76.0210


>> sum(proyecx)

ans =

    1.1232
```

Esta última imagen nos da los valores obtenidos para las tolerancias de las variables X al finalizar la optimización.

Podemos concluir que las tolerancias de las variables X que definen el problema son las siguientes:

$$X_1 = 25.4mm \pm 0.2169mm \Rightarrow X_1 \in [25.1831, 25.6169]mm$$
$$X_2 = 50.8mm \pm 0.1339mm \Rightarrow X_2 \in [50.661, 50.9339]mm$$
$$X_3 = 76.2mm \pm 0.2016mm \Rightarrow X_3 \in [75.9984, 76.4016]mm$$
$$X_4 = 101.6mm \pm 1.179mm \Rightarrow X_4 \in [101.4821, 101.7179]mm$$
$$X_5 = 25.4mm \pm 0.2314mm \Rightarrow X_5 \in [25.1686, 25.6314]mm$$
$$X_6 = 25.3mm \pm 0.0674mm \Rightarrow X_6 \in [25.2326, 25.3674]mm$$
$$X_7 = 50.8mm \pm 0.0750mm \Rightarrow X_7 \in [50.7250, 50.8750]mm$$
$$X_8 = 76.1mm \pm 0.0790mm \Rightarrow X_8 \in [76.0210, 76.1790]mm$$

4.1.3.3. Comparación con el caso de independencia

Por último se muestran los resultados que obtendríamos si consideráramos el problema como independiente para poder comparar los resultados con los que hemos obtenido utilizando la metodología propuesta en este trabajo en la que se consideran las variables dependientes:

```
LTSX =

   25.4887
   50.8443
   76.2667
  101.7037
   25.5677
   25.3666
   50.8442
   76.1887


LTIX =

   25.3113
   50.7557
   76.1333
  101.4963
   25.2323
   25.2334
   50.7558
   76.0113
```

```
>> sum(proyecx)

ans =

    0.6705
```

$$X_1 = 25.4mm \pm 0.887mm \Rightarrow X_1 \in [25.3113, 25.4887]mm$$
$$X_2 = 50.8mm \pm 0.0443mm \Rightarrow X_2 \in [50.757, 50.8443]mm$$
$$X_3 = 76.2mm \pm 0.0667mm \Rightarrow X_3 \in [75.1333, 76.2667]mm$$
$$X_4 = 101.6mm \pm 1.037mm \Rightarrow X_4 \in [101.4963, 101.7037]mm$$
$$X_5 = 25.4mm \pm 1.677mm \Rightarrow X_5 \in [25.2323, 25.5677]mm$$
$$X_6 = 25.3mm \pm 0.0666mm \Rightarrow X_6 \in [25.2334, 25.366]mm$$
$$X_7 = 50.8mm \pm 0.0442mm \Rightarrow X_7 \in [50.7558, 50.8442]mm$$
$$X_8 = 76.1mm \pm 0.0887mm \Rightarrow X_8 \in [76.0113, 76.1887]mm$$

Si comparamos los resultados obtenidos con los que obtuvimos para el caso de dependencia se puede observar que la hipótesis de independencia de X tiene un efecto importante en sus tolerancias, la suma de las proyecciones es mayor en el caso de dependencia, lo que supone un aumento general de las tolerancias, lo cual supondrá una disminución en los costes de fabricación.

4.2. EJEMPLO 2 .ASIGNACIÓN DE TOLERANCIAS A UNA BALLESTA

4.2.1. PLANTEAMIENTO DEL PROBLEMA

Este ejemplo ilustra la aplicación del método desarrollado a lo largo del proyecto para el diseño de tolerancias para un conjunto mecánico. Esta relacionado con la asignación de tolerancias de una ballesta y fue propuesto por Lee y Tang (2000) en el contexto de la asignación de tolerancias en virtud de la independencia de X. La ballesta tiene dos características funcionales: EL radio de la hoja, Y_1 , y la carga máxima admisible, Y_2. Estas características dependen del espesor X_1, del ancho X_2 y de la longitud de la ballesta X_3.

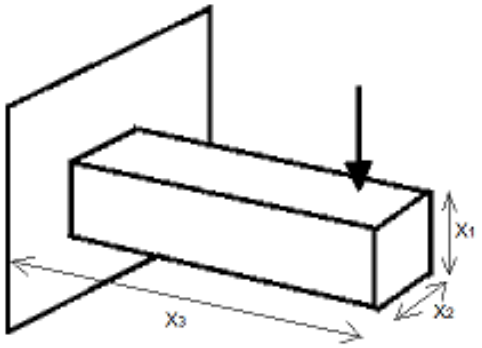


Figura 4.5. Representación grafica de la ballesta

Estas variables X se consideran variables dependientes a causa del proceso de fabricación. A raíz de Lee y Tang (2000) la relación $\mathbf{Y} = f(\mathbf{X})$ puede escribirse como:

$$Y_1 = \frac{EX_2X_1^3}{4X_3^3}$$

$$Y_2 = \frac{\sigma_b X_2 X_1^3}{6X_3}$$

Donde, E=21000Kg/mm2 es el módulo de elasticidad y $\sigma_b = 49.2Kg/mm^2$ es el rendimiento basado en el estrés de los materiales utilizados en la ballesta.

El problema es la asignación de las tolerancias máximas para as variables $\mathbf{X} = (X_1, X_2, X_3)'$. Los valores nominales vienen especificados por condiciones de diseño y son los siguientes:

$$\mathbf{VN}_X = (5, 40, 50)'$$

Estos valores nominales serán las medias de las variables, ya que se considera que el proceso está centrado en esos valores, Por tanto:

$$\boldsymbol{\mu}_X = (5, 40, 50)'$$

Tal y como se ha explicado las especificaciones de diseño para este conjunto mecánico se definen en término de dos variables **Y**, denotadas como $\mathbf{Y} = (Y_1, Y_2)'$.

Los valores nominales de las variables **Y** son:

$$VN_y = (0.288, 18.22)'$$

Los límites de especificación requeridos para las variables **Y** son:

$$Y_1 \in [0.268, 0.308]$$
$$Y_2 > 17.62_3$$

En este caso las función Y_1 e Y_2 guardan una relación no lineal con las variables **X**, por tanto, aplicaremos una expansión de Taylor para poder linealizarlas de la forma:

$$Y = a_0 + a_1 X_1 + a_2 X_2 + a_3 X_3$$

Para ello utilizaremos la siguiente expansión de Taylor:

$$Y = VN_{Y_3} + \sum_{i=1}^{12} \frac{\partial Y_3}{\partial X_i}\Big|_{VN_i} (X_i - VN_i)$$

$$Y = VN_{Y_3} + \frac{\partial Y_3}{\partial X_1}\Big|_{VN} (X_1 - VN_1) + \frac{\partial Y_3}{\partial X_2}\Big|_{VN} (X_2 - VN_2) + \frac{\partial Y_3}{\partial X_3}\Big|_{VN} (X_3 - VN_3)$$

Comenzamos linealizando la función Y_1:

$$Y_1 = \frac{EX_2X_1^3}{4X_3^3} \Rightarrow Y_1 = a_0 + a_1X_1 + a_2X_2 + a_3X_3$$

Comenzaremos calculando los coeficientes

$$a_1 = \left.\frac{\partial Y_1}{\partial X_1}\right|_{VN} = \frac{3X_1^2EX_2}{4X_3^3} = \frac{3\cdot 5^2\cdot E\cdot 40}{4\cdot 50^3} = 0.17284$$

$$a_2 = \left.\frac{\partial Y_1}{\partial X_2}\right|_{VN} = \frac{EX_1^3}{4X_3^3} = \frac{E\cdot 5^3}{4\cdot 50^3} = 0.0072$$

$$a_3 = \left.\frac{\partial Y_1}{\partial X_3}\right|_{VN} = \frac{-12X_3^2EX_2X_1^3}{(4X_3^3)^2} = \frac{-12\cdot 50^2\cdot E\cdot 40\cdot 5^3}{(4\cdot 50^3)^2} = -0.01728$$

Utilizando la fórmula de la expansión de Taylor quedaría:

$$Y_1 = VN_{Y_1} + \left.\frac{\partial Y_3}{\partial X_1}\right|_{VN}(X_1 - VN_1) + \left.\frac{\partial Y_3}{\partial X_2}\right|_{VN}(X_2 - VN_2) + \left.\frac{\partial Y_3}{\partial X_3}\right|_{VN}(X_3 - VN_3)$$

$$Y_1 = 0.288 + 0.17284(X_1 - 5) + 0.0072(X_2 - 40) - 0.01728(X_3 - 50)$$

$$Y_1 = 0.288 + 0.17284X_1 - 0.8642 + 0.0072X_2 - 0.288 - 0.01728X_3 + 0.864$$

$$Y_3 = 0.17284X_1 + 0.0072X_2 - 0.01728X_3$$

Por lo tanto, una vez linealizada la función Y_1 quedaría:

$$Y_3 = 0.17284X_1 + 0.0072X_2 - 0.01728X_3$$

A continuación linealizamos la función Y_2:

$$Y_2 = \frac{\sigma_b X_2X_1^3}{6X_3} \Rightarrow Y_2 = a_0 + a_1X_1 + a_2X_2 + a_3X_3$$

Comenzaremos calculando los coeficientes

$$a_1 = \left.\frac{\partial Y_2}{\partial X_1}\right|_{VN} = \frac{3X_1^2\sigma_b X_2}{6X_3} = \frac{3\cdot 5^2\cdot\sigma_b\cdot 40}{6\cdot 50} = 10.93$$

$$a_2 = \left.\frac{\partial Y_2}{\partial X_2}\right|_{VN} = \frac{\sigma_b X_1^3}{6X_3} = \frac{\sigma_b\cdot 5^3}{6\cdot 50} = 0.4553$$

$$a_3 = \left.\frac{\partial Y_2}{\partial X_3}\right|_{VN} = \frac{-6\sigma_b X_2 X_1^3}{(6X_3)^2} = \frac{-3\cdot\sigma_b\cdot 40\cdot 5^3}{(6\cdot 50)^2} = -0.364$$

Utilizando la fórmula de la expansión de Taylor quedaría:

$$Y_2 = VN_{Y_2} + \left.\frac{\partial Y_3}{\partial X_1}\right|_{VN}(X_1 - VN_1) + \left.\frac{\partial Y_3}{\partial X_2}\right|_{VN}(X_2 - VN_2) + \left.\frac{\partial Y_3}{\partial X_3}\right|_{VN}(X_3 - VN_3)$$

$$Y_2 = 18.22 + 10.93(X_1 - 5) + 0.4553(X_2 - 40) - 0.364(X_3 - 50)$$

$$Y_1 = 18.22 + 10.93X_1 - 54.65 + 0.4553X_2 - 18.212 - 0.364X_3 + 18.2$$

$$Y_3 = -36.442 + 10.93X_1 + 0.4553X_2 - 0.364X_3$$

Por lo tanto, una vez linealizada la función Y_1 quedaría:

$$Y_3 = -36.442 + 10.93X_1 + 0.4553X_2 - 0.364X_3$$

Luego, las relaciones entre las variables **X** e **Y** quedarían como:

$$Y_1 = 0.17284X_1 + 0.0072X_2 - 0.01728X_3$$
$$Y_2 = -36.44 + 10.93X_1 + 0.4553X_2 - 0.364X_3$$

De donde, se deduce que la matriz **A** es igual a:

$$\mathbf{A} = \begin{bmatrix} 0.17284 & 0.0072 & -0.01728 \\ 10.93 & 0.4553 & -0.364 \end{bmatrix}$$

Una vez linealizada la función, y con la matriz **A** que se ha obtenido podemos obtener el vector de medias para las variables **Y**

$$\boldsymbol{\eta}_Y = a + \mathbf{A}\boldsymbol{\mu}_X =$$

$$= \begin{bmatrix} 0 \\ -36.442 \end{bmatrix} + \begin{bmatrix} 0.17284 & 0.0072 & -0.01728 \\ 10.93 & 0.4553 & -0.364 \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 40 \\ 50 \end{bmatrix} = \begin{bmatrix} 0.2882 \\ 18.22 \end{bmatrix}$$

Luego, el vector de medias es $\boldsymbol{\eta}_Y = (0.2882, 18.22)'$

La matriz de covarianzas de **X** se simulará al igual que en el ejemplo anterior, generando una matriz de covarianzas aleatoria ya que al ser un ejemplo teórico no tenemos datos reales del proceso bajo control. En este ejemplo al ser las variables X correladas, pertenecientes a una sola pieza se generará una única matriz de covarianzas:

$$\boldsymbol{\Sigma}_x = \begin{bmatrix} 0.0102 & 0.0090 & 0.0019 \\ 0.0090 & 0.0103 & 0.0018 \\ 0.0019 & 0.0018 & 0.0029 \end{bmatrix}$$

El objetivo del problema es asignar las mayores tolerancias de **X** de manera que la proporción de defectuosas sea menor o igual que 0.0027, es decir $p \leq \alpha = 0.0027$

Con el fin de poder resolver el problema de optimización, además de ésta restricción se considerará una restricción adicional, que las tolerancias de **X** deben ser superiores a 0.01mm, es decir, $t_i \geq 0.001;\ i = 1,2,...,12$ De esta manera evitamos que las tolerancias tomen valores iguales o muy próximos a cero.

4.2.2. SOLUCIÓN DEL PROBLEMA UTILIZANDO EL PROGRAMA DE CÁLCULO MATLAB

4.2.2.1. Etapa 1

Para desarrollar la etapa 1 empezamos, como en el ejemplo anterior describiendo los programas necesarios para esta etapa. Todos los programas son prácticamente iguales a los utilizados en el ejemplo 1, se han realizado sólo pequeñas modificaciones para ajustarlos a los datos de este ejemplo.

El primero de ellos es el necesario para generar una matriz de covarianzas aleatoria:

```
% Este programa generara datos aleatorios de
% distribuciones normales multivariantes
% En este ejmplo el enamblaje consta de una piezas,
% por lo que se generarán una sola matriz de covarianzas de 3x3

dim_pieza_sup=3; % Adaptamos la matriz al numero de variables correladas
                 % que definen la pieza 3 variables

m=1000; %Definimos un numero elevado de iteraciones

b1=[0.1 0.1 0.1]; % Valores para el comienzo de la iteración.

x1=zeros(m,dim_pieza_sup); % Generamos una matrices de ceros,de m filas

%-------------------------------------------------------

%-------------------------------------------------------

x1(:,1)=randn(m,1)*b1(1);

for i=2:dim_pieza_sup
    x1(:,i)=rand*x1(:,1)+randn(m,1)*b1(i)*0.5;
end
Matcorr1=corrcoef(x1);
MatCov1=cov(x1)
[autovectores1,au21]=eig(MatCov1);
Sigma1=sqrt(diag(MatCov1));
Propau1=sort(diag(au21)./sum(diag(au21))*100,'descend');

%-------------------------------------------------------

% Este programa genera la  matriz de covarianzas iniciales de
% x que se utilizara el los posteriores programas.

%-------------------------------------------------------

save CovIniciales MatCov1  % Por último guardamos la matriz que hemos
                           %generado para utilizarla posteriormente
```

También necesitaremos definir la función objetivo:

```
function f=funcionobjetivo1(x)

%En este programa se define la funcion objetivo

bi=x;  % valores por los que iremos multiplicando los autovectores para
       % modificarlos hasta conseguir los valores óptimos que cumplan la
       % función objetivo que estamos definiendo

p=0.9973; % definimos el porcentaje de piezas defectuosas que consideramos
          % aceptable

load CovIniciales % Cargamos el programa que hemos realizado antes para
                  % generar la matriz de covarianzas


covarianzasx=zeros(3,3); % la dimensión de la matriz de covarianzas sera la
                         % correspondiente al numero de variables X totales
                         % que tiene el ensamblaje

covarianzasx(1:3,1:3)=MatCov1; %La matriz de covarianzas es directamente la
%que hemos obtenido en el primer programa, ya que tenemos una unica pieza


%--------------------------------------------------------------------------
```

```

%--------------------------------------------------------------------------
a=length(covarianzasx);

[C, autov]=eig(covarianzasx); % Obtiene los autovectores y los
% autovalores de la matriz d covarianzas de x

D=diag(autov);% Con los autovalores obtenidos en el comando anterior
% generamos una matriz D, diagonal de autovalores de la matriz
% de covarianzas

newD=bi.*D; % Obtenemos una nueva matriz D modificando los autovalores
%multiplicandola por los factores bi

newcovarianzasx=C*diag(newD)*C'; % con la nueva matriz D obtenemos una
% nueva matriz de covarianzas de x

proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)); % calculamos la
% proyección de la diagonal de la matriz de covarianzas de x que
% estos son los valores ti

```

```
43
44 -    f=-sum(proyecx); % la función objetivo sera buscar la minima diferencia
45      % entre las proyecciones de la matriz de covarianzas de x y las tolerancias
46      % de x
47
48      %--------------------------------------------------------------------------
49
50 -    save NewX newcovarianzasx
```

Definimos las restricciones:

```
1       function [restricciones,nleq]=definicion_de_restricciones(x)
2
3       %Aquí se definen las restricciones del problema
4
5 -     load Newx  % esta leyendo newcovarianzasx
6
7 -     bi=x;
8 -     p=0.9973;
9
10 -    a=length(newcovarianzasx); % tamaño de la matriz de covarianzas de x
11
12 -    A=[0.17284 0.0072 -0.01728;10.93 0.4553 -0.364];
13      % Matrz de relaciones defindа por las especificaciones del montaje mecánico
14      % (Y=f(X))
15
16 -    tolminx=[0.01;0.01;0.01];
17      % tolerancias mínimas para x
18
19 -    toleranciasy=[0.02;0.6];
20      % toleranciasy =( Valores de Y [(y1=[0.268,0.308] , y2>=17.2]-
21      %valores nominales de Y [0.288,18.22])
22
23      %--------------------------------------------------------------------------
```

```
25      %--------------------------------------------------------------------------
26      % restricción 1
27 -    restriccion1=det(newcovarianzasx); % calculamos el determinante de la nueva
28      % matriz de covarianzas de x que hemos obtenido
29
30      %--------------------------------------------------------------------------
31      % restricción 2
32 -    proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)); % calculamos la
33      % proyección de la matriz de varianzas de x que acabamos de obtener
34
35 -    restriccion2=-(proyecx-tolminx); % calculamos la diferencia entre la
36      % proyección de la matriz de varianzas de x y las tolerancias minimas de x
37      % que hemos definido
38
```

```
%--------------------------------------------------------------------------
% restricción 3
newcovarianzasy=A*newcovarianzasx*A'; %obtenemos la nueva matriz de
% covarianzas de y a traves de la nueva matriz de covarianzas de x que
%hemos obtenido en el paso anterior

b=length(newcovarianzasy); % tamaño de la matriz de covarianzas de y
proyecy=sqrt(chi2inv(p,b)*diag(newcovarianzasy));% calculamos la proyección
% de la matriz de varianzas de y

restriccion3=proyecy-toleranciasy; % calculamos la diferencia entre la
% proyección de la matriz de varianzas de y las tolerancias de y

%--------------------------------------------------------------------------
```

```
%--------------------------------------------------------------------------
restricciones=[-restriccion1;restriccion2;restriccion3];
nleq=[];
% Las restricciones del problema serán: que no se superen las tolerancias
% de y ( utilizamos diferencias d y), que las tolerencias de x sean mayores
% que las tolerancias minima de x que hemos definido (utilizamos
% diferenciasx) y que los determinantes de las matrices de covarianzas de x
% e y sean >0 (utilizamos restriccion1 y restriccion2)

%--------------------------------------------------------------------------
```

Por ultimo se muestra el programa principal en el que se desarrolla la etapa 1 mediante la utilización de la función fmincon:

```

% Se quiere asignar tolerancias a 3 variables para cumplir las
% restricciones de 2 variables Y. Etapa1
%--------------------------------------------------------------------------
options=optimset('LargeScale','off');
puntoinicial=[0.01;0.01;0.01];
bimin=[0.001;0.001;0.001];

% Para realizar la optimización debemos definir un punto inicial en el que
% comenzará la optimización y un valor minimo de aumento en cada iteración

[x,FVAL,EXITFLAG,output]=fmincon(@funcionobjetivo1,puntoinicial,[],[],[],[],...
    bimin,[],@definicion_de_restricciones,options);

%Se define la variable a optimizar, en este caso x, la función de
%optimización que será la que hemos definido con
%anterioridad en funcionobjetivo, el punto inicial, el valor minimo y las
%restricciones que seran las definidas con anterioridad en
%definicion_de_restricciones

% Ejecuta el algoritmo de optimización hasta encontar la solución óptima

```

```
25
26      %-------------------------------------------------------------------------
27
28      %aquí se lee la solución para ver los valores obtenidos, se calcularan los
29      %valores de las matrices de covarianzas de x e y óptimos
30
31 -    bi=x;  % valores óptimos de bi
32
33 -    p=0.9973;
34 -    alpha=1-p;
35
36 -    load CovIniciales % Se lee la matriz de covarianzas
37
38 -    covarianzasx=zeros(3,3);
39 -    covarianzasx(1:3,1:3)=MatCov1;
40
41
42 -    a=length(covarianzasx); % tamaño de la matriz de covarianzas de x
43
44 -    A=[0.17284 0.0072 -0.01728;10.93 0.4553 -0.364];
45      % Matriz de relaciones definda por las especificaciones del montaje
46      % mecánico(Y=f(X)
47
48 -    tolminx=[0.01;0.01;0.01]; % es el valor de
49      % tolerancias mínimas para x
50
51 -    toleranciasy=[0.02;0.6];
52      % toleranciasy =( Valores de Y [(y1=[0.268,0.308] , y2>=17.62]
53      %- valores nominales de Y [0.288,18.22])
54
55      %-------------------------------------------------------------------------
```

El resto del programa es igual al ejemplo 1.

A continuación se muestra la matriz de covarianzas obtenida al finalizar esta etapa

```
newcovarianzasx =

    0.0001    0.0008   -0.0008
    0.0008    0.0071   -0.0075
   -0.0008   -0.0075    0.0079
```

4.2.2.2. Etapa 2.

Para el cálculo de la proporción de piezas defectuosas se hace uso de la misma función usada en el ejemplo 1.

El programa correspondiente a la etapa 2 sólo cambian en algunas líneas con respecto al programa del ejemplo 1:

```
% En este prgramas se realiza la Etapa 2.
% necesito cargar la solución obtenida en la Etapa 1

load Etapa1  %lee newcovarianzasc bi
% las variables que lee son los óptimos: newcovarianzasy newcovarianzasx bi

A=[0.17284 0.0072 -0.01728;10.93 0.4553 -0.364];
% Matrz de relaciones defindа por las especificaciones del montaje mecánico
% (Y=f(X))

toleranciasy=[0.02;0.6];
% toleranciasy =( Valores de Y [(y1=[0.268,0.308] , y2>=17.62]
%- valores nominales de Y [0.288,18.22])

bi0=bi; % tomamos como valores iniciales los valores de bi y las matrices
covx=newcovarianzasx; % de covarianzas de x e y obtenidos en la etapa 1

mediax=[5 40 50]'; % Valores nominales de x en columna

mediay=A*mediax; % (0.288,18.22)% Valores nominales de y

a=length(mediax); % número de variables x

b=length(mediay) ;% número de variables y

h=mediay-toleranciasy;

%Las especificaciones de Y son: Y1=[0.268,0.308], Y2>17.62
LES=[0.308 1000]; % En el caso de tener solo límite inferior
% ponemos un valor elevado para el límite superior (caso de y2)

LEI=[0.268 17.2]; % en este caso las dos variables tienen definido el
                  % límite de especificación inferior)

%-------------------------------------------------------------------------
```

El resto del programa es igual que en la etapa 1

Los resultados obtenidos son:

```
nonconforming =

    0.0027
```

```
LTSX =

    5.0653
   40.5685
   50.5979


LTIX =

    4.9347
   39.4315
   49.4021

>> sum(proyecx)

ans =

    1.2317
```

Esta última imagen nos da los valores obtenidos para las tolerancias de las variables X al finalizar la optimización.

Podemos concluir que las tolerancias de las variables X que definen el problema son las siguientes:

$$X_1 = 5mm \pm 0.0652mm \Rightarrow X_1 \in [4.9348, 5.0652]mm$$
$$X_2 = 40mm \pm 0.5672mm \Rightarrow X_2 \in [39.4328, 40.5672]mm$$
$$X_3 = 50mm \pm 0.5966mm \Rightarrow X_3 \in [49.4034, 50.5966]mm$$

4.2.2.3. Comparación con el caso de independencia

Al igual que en el ejemplo anterior comparamos los resultados con los obtenidos para el caso de independencia:

```
LTSX =

    5.0130
   41.2115
   51.3584


LTIX =

    4.9870
   38.7885
   48.6416
```

```
>> sum(proyecx)

ans =

    2.5925
```

En este caso si comparamos los resultados obtenidos para el caso de dependencia e independencia , la dependencia entre las variables influye significativamente en la asignación de los límites de tolerancia, especialmente para las variables X_2 y X_3, en este caso las suma de las proyecciones es mayor para el caso de independencia, en diferencia con el caso anterior la solución no supone un aumento de las tolerancias, sino una disminución, considerar la dependencia de las variables supone que el proceso sea mas adecuado, aun incrementando los costes debido a la disminución de la tolerancia.

4.3. ASIGNACIÓN DE TOLERANCIAS A UN CONJUNTO MECÁNICO

4.3.1. PLANTEAMIENTO DEL PROBLEMA

Este ejemplo ilustra la aplicación del método desarrollado a lo largo del proyecto para el diseño de tolerancias para un conjunto mecánico. Este ejemplo fue propuesto por Lee y Woo (1993). El conjunto mecánico está compuesto por 2 piezas como se muestra en la figura 3.11 cuya relación variables X que definen las piezas y las variables Y que definen la relación entre las variables X es no lineal.

Las variables **X** del problema son un conjunto de longitudes de la pieza como se muestra en la figura 3.11. Estas variables son 12 y se denotan como $\mathbf{X} = (X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10})'$. Los valores nominales vienen especificados por condiciones de diseño y son los siguientes:

$$\mathbf{VN}_X = (50, 20.05, 9.9985, 9.9985, 30, 10, 30, 30, 40, 50)'$$

Estos valores nominales serán las medias de las variables, ya que se considera que el proceso está centrado en esos valores, Por tanto:

$$\boldsymbol{\mu}_X = (50, 20.05, 9.9985, 9.9985, 30, 10, 30, 30, 40, 50)'$$

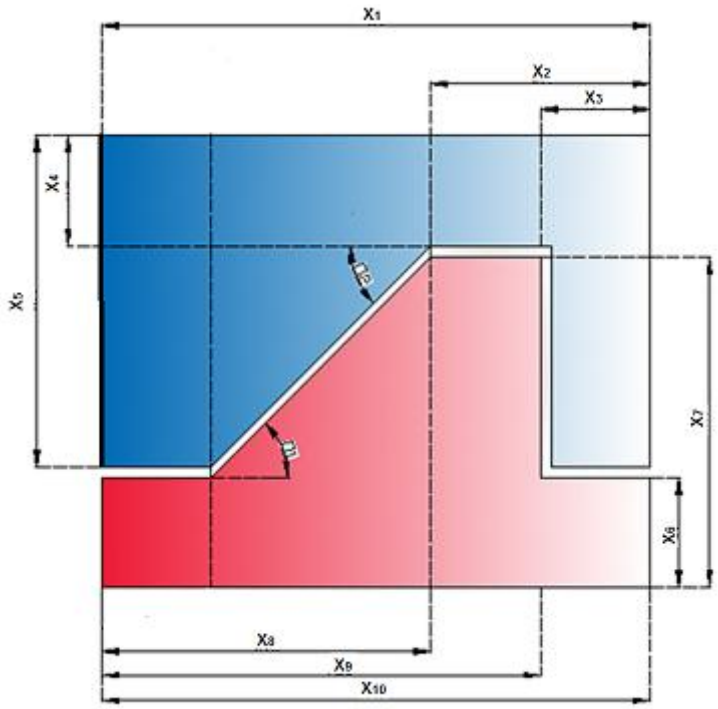


Figura 4.5. Conjunto mecánico de 2 piezas

Las especificaciones de diseño para este conjunto mecánico se definen en término de una serie de variables **Y**, denotadas como $\mathbf{Y} = (Y_1, Y_2, Y_3)´$.

Las funciones de diseño Y_1 e Y_2 representan las condiciones de holgura, tanto vertical como horizontal entre las dos partes.

La función de diseño Y_3 nos da los requisitos necesarios para la diferencia de de tamaño de las dos partes.

La relación $\mathbf{Y} = f(\mathbf{X})$ está definida como:

$$Y_1 = (X_5 - X_4) - (X_7 - X_8)$$
$$Y_2 = (X_2 - X_3) - (X_9 - X_8)$$
$$Y_3 = X_1 - X_{10}$$

Los valores nominales de las variables **Y** son:

$$VN_y = (0.0015, 0.0515, 0)$$

Los límites de especificación requeridos para las variables **Y** son:

$$Y_1 > 0mm$$
$$Y_2 > 0mm$$
$$Y_4 = 0mm \pm 0.01$$

Simplificando las relaciones entre las variables **X** e **Y** quedarían como:

$$Y_1 = -X_4 + X_5 + X_6 - X_7$$
$$Y_2 = X_2 - X_3 + X_8 - X_9$$
$$Y_3 = X_1 - X_{10}$$

De donde, se deduce que la matriz **A** es igual a:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

Una vez obtenida la matriz A podemos obtener el vector de medias de Y:

$$\boldsymbol{\eta}_Y = a + \mathbf{A}\boldsymbol{\mu}_X =$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 50 \\ 20.05 \\ 9.9985 \\ 9.9985 \\ 30 \\ 10 \\ 30 \\ 30 \\ 40 \\ 50 \end{bmatrix} = \begin{bmatrix} 0.0015 \\ 0.0515 \\ 0 \end{bmatrix}$$

Luego, el vector de medias es $\boldsymbol{\eta}_Y = (0.0015, 0.0515, 0)'$

La matriz de covarianzas de **X** se simulará al igual que en el ejemplo anterior, generando una matriz de covarianzas aleatoria ya que al ser un ejemplo teórico no tenemos datos reales del proceso bajo control. Igual que en el ejemplo anterior, al existir dos partes individuales en el conjunto mecánico, se deben considerar los dos grupos de variables correladas. El primer grupo corresponde a las variables X_1, X_2, X_3, X_4, X_5 que definen la pieza de arriba, y el segundo grupo corresponde a las variables $X_6, X_7, X_8, X_9, X_{10}$ que definen la pieza de abajo.

Así, en primer lugar se genera una matriz de covarianzas para cada grupo de variables y luego se construye la matriz total de covarianzas. El bloque de la matriz resultante Σ_x que será usada en este ejemplo es la siguiente:

$$\boldsymbol{\Sigma}_x = \begin{bmatrix} 0.0099 & 0.0066 & 0.0039 & 0.0092 & 0.0059 & 0 & 0 & 0 & 0 & 0 \\ 0.0066 & 0.0070 & 0.0026 & 0.0061 & 0.0039 & 0 & 0 & 0 & 0 & 0 \\ 0.0039 & 0.0026 & 0.0040 & 0.0037 & 0.0024 & 0 & 0 & 0 & 0 & 0 \\ 0.0092 & 0.0061 & 0.0037 & 0.0111 & 0.0056 & 0 & 0 & 0 & 0 & 0 \\ 0.0059 & 0.0039 & 0.0024 & 0.0056 & 0.0061 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0102 & 0.0077 & 0.0085 & 0.0038 & 0.0006 \\ 0 & 0 & 0 & 0 & 0 & 0.0077 & 0.0066 & 0.0064 & 0.0029 & 0.0005 \\ 0 & 0 & 0 & 0 & 0 & 0.0085 & 0.0064 & 0.0080 & 0.0032 & 0.0005 \\ 0 & 0 & 0 & 0 & 0 & 0.0038 & 0.0029 & 0.0032 & 0.0024 & 0.0002 \\ 0 & 0 & 0 & 0 & 0 & 0.0006 & 0.0005 & 0.0005 & 0.0002 & 0.0009 \end{bmatrix}$$

El objetivo del problema es asignar las mayores tolerancias de **X** de manera que la proporción de defectuosas sea menor o igual que 0.0027, es decir $p \leq \alpha = 0.0027$

Con el fin de poder resolver el problema de optimización, además de ésta restricción se considerará una restricción adicional, que las tolerancias de **X** deben ser superiores a 0.01mm, es decir, $t_i \geq 0.001;\ i = 1,2,...,10$ De esta manera evitamos que las tolerancias tomen valores iguales o muy próximos a cero.

4.3.2. SOLUCIÓN DEL PROBLEMA UTILIZANDO EL PROGRAMA DE CÁLCULO MATLAB

Para este ejemplo la parte de los programas que se muestran es la que corresponde con los pasos que se deben cambiar para cada ejemplo en concreto, el resto de los programas es igual que en el ejemplo 1, los programas completos para cada ejemplo se pueden ver en los Anexos del trabajo.

4.3.2.1. Etapa 1

Tal y como se ha hecho en los ejemplos anteriores comenzamos definiendo la martriz de covarianzas de X.

```
1
2     % Este programa generara datos aleatorios de
3     % distribuciones normales multivariantes
4     % En este ejmplo el enamblaje consta de dos piezas,
5     % por lo que se generarán dos matrices de covarianzas:
6     % una de 5x5 y otra de 5x5
7
8
9  -  dim_pieza_sup=5; % Adaptamos la matriz al numero de variables correladas
10                    % que definen la pieza inferior, 5 variables
11 -  dim_pieza_inf=5; % Adaptamos la matriz al numero de variables correladas
12                    % que definen la pieza superior, 5 variables
13
14 -  m=1000; %Definimos un numero elevado de iteraciones
15
16 -  b1=[0.1 0.1 0.1 0.1 0.1]; % Valores para el comienzo de la iteración.
17 -  b2=[0.1 0.1 0.1 0.1 0.1]; % Valores para el comienzo de la iteración.
18
19 -  x1=zeros(m,dim_pieza_sup); % Generamos dos matrices de ceros,
20                               % cada una de ellas de m filas
21 -  x2=zeros(m,dim_pieza_inf); % y columnas acordes con las dimensiones
22                               % que se han definido anteriormente
```

A continuación definimos la función objetivo:

```
function f=funcionobjetivo1(x)

%En este programa se define la funcion objetivo

bi=x;  % valores por los que iremos multiplicando los autovectores para
       % modificarlos hasta conseguir los valores óptimos que cumplan la
       % función objetivo que estamos definiendo

p=0.9973; % definimos el porcentaje de piezas defectuosas que consideramos
          % aceptable

load CovIniciales % Cargamos el programa que hemos realizado antes para
                  % generar la matriz de covarianzas


covarianzasx=zeros(10,10); % la dimensión de la matriz de covarianzas sera la
                           % correspondiente al numero de variables X totales
                           % que tiene el ensamblaje

covarianzasx(1:5,1:5)=MatCov1; % dependiendo del numero de variables corre-
covarianzasx(6:10,6:10)=MatCov2 % ladas que tengamos obtendremos una
                                % matriz de covarianzas total,utilizando las
                                % dos matrices que se han obtenido en el
                                % programa anterior para generar la matriz
                                % de covarienzas
```

Definimos las restricciones:

```
function [restricciones,nleq]=definicion_de_restricciones(x)

%Aquí se definen las restricciones del problema

load Newx  % esta leyendo newcovarianzasx

bi=x;
p=0.9973;

a=length(newcovarianzasx); % tamaño de la matriz de covarianzas de x

A=[0 0 0 -1 1 1 -1 0 0 0;0 1 -1 0 0 0 0 1 -1 0;1 0 0 0 0 0 0 0 0 -1];
% Matrz de relaciones defindа por las especificaciones del montaje mecánico
% (Y=f(X))

tolminx=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];
% tolerancias mínimas para x

toleranciasy=[0.1;0.15;0.1];
% toleranciasy =( Valores de Y [(y1>0 , y2>0 ,y3=0+-0.01]-
%valores nominales de Y [0.0015, 0.0515 , 0])

%-----------------------------------------------------------------------
```

Por ultimo definimos el programa base para la etapa 1 donde se utiliza la función fmincon que hace uso de los programas que se han definido con anterioridad.

```
clear all

% Se quiere asignar tolerancias a 8 variables para cumplir las
% restricciones de 4 variables Y. Etapa1
%-------------------------------------------------------------------------
options=optimset('LargeScale','off');
puntoinicial=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];
bimin=[0.001;0.001;0.001;0.001;0.001;0.001;0.001;0.001;0.001;0.001];

% Para realizar la optimización debemos definir un punto inicial en el que
% comenzará la optimización y un valor minimo de aumento en cada iteración

[x,FVAL,EXITFLAG,output]=fmincon(@funcionobjetivo1,puntoinicial,[],[],[],[],
    bimin,[],@definicion_de_restricciones,options);

%Se define la variable a optimizar, en este caso x, la función de
%optimización que será la que hemos definido con
%anterioridad en funcionobjetivo, el punto inicial, el valor minimo y las
%restricciones que seran las definidas con anterioridad en
%definicion_de_restricciones

% Ejecuta el algoritmo de optimización hasta encontar la solución óptima
```

```
%-------------------------------------------------------------------------

%aquí se lee la solución para ver los valores obtenidos, se calcularan los
%valores de las matrices de covarianzas de x e y óptimos

bi=x;  % valores óptimos de bi

p=0.9973;
alpha=1-p;

load CovIniciales % Se leen las matrices de covarianzas y se unen en una
                  % sola de 8x8
covarianzasx=zeros(10,10);
covarianzasx(1:5,1:5)=MatCov1;
covarianzasx(6:10,6:10)=MatCov2;

a=length(covarianzasx); % tamaño de la matriz de covarianzas de x

A=[0 0 0 -1 1 1 -1 0 0 0;0 1 -1 0 0 0 0 1 -1 0;1 0 0 0 0 0 0 0 0 -1];
% Matriz de relaciones defindа por las especificaciones del montaje
% mecánico(Y=f(X)

tolminx=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];
% es el valor de tolerancias minimas para x


toleranciasy=[0.1;0.15;0.1];
% toleranciasy =( Valores de Y [(y1>0 , y2>0 , y4=0+-0.01]-
%valores nominales de Y [0.0015, 0.0515 , 0])

%-------------------------------------------------------------------------
```

La matriz de covarianzas obtenida al finalizar esta etapa es la siguiente:

```
newcovarianzasx =

  Columns 1 through 9

    0.0001    0.0001    0.0000   -0.0000    0.0000         0         0         0         0
    0.0001    0.0010   -0.0002   -0.0015   -0.0008         0         0         0         0
    0.0000   -0.0002    0.0002    0.0001    0.0001         0         0         0         0
   -0.0000   -0.0015    0.0001    0.0031    0.0016         0         0         0         0
    0.0000   -0.0008    0.0001    0.0016    0.0009         0         0         0         0
         0         0         0         0         0    0.0010    0.0009    0.0009    0.0008
         0         0         0         0         0    0.0009    0.0008    0.0008    0.0007
         0         0         0         0         0    0.0009    0.0008    0.0008    0.0007
         0         0         0         0         0    0.0008    0.0007    0.0007    0.0006
         0         0         0         0         0    0.0008    0.0007    0.0007    0.0006

  Column 10

         0
         0
         0
         0
         0
    0.0008
    0.0007
    0.0007
    0.0006
    0.0006
```

4.3.2.2. Etapa 2

Por ultimo, al igual que hemos hecho para los dos ejemplos anteriores definimos el programa necesario para ejecutar la etapa 2, la función para el cálculo de defectuosos no se escribirá en este caso ya que es la misma para todos los ejemplos.

```
1  -  clear all
2
3     % En este prgramas se realiza la Etapa 2.
4     % necesito cargar la solución obtenida en la Etapa 1
5
6  -  load Etapa1  %lee newcovarianzasc bi
7     % las variables que lee son los óptimos: newcovarianzasy newcovarianzasx bi
8
9  -  A=[0 0 0 -1 1 1 -1 0 0 0;0 1 -1 0 0 0 0 1 -1 0;1 0 0 0 0 0 0 0 0 -1];
10    % Matrz de relaciones defindа por las especificaciones del montaje mecánico
11    % (Y=f(X))
12
13 -  toleranciasy=[0.1;0.15;0.1];
14    % toleranciasy =( Valores de Y [(y1>=0.0015 , y2>=0.0515 , y3=
15    % 0.0049+-0.0175 y4=0+-0.01])- valores nominales de Y
16    %[0.0015, 0.05151 ,0.0049, 0])
17
```

```
17
18
19 -    bi0=bi; % tomamos como valores iniciales los valores de bi y las matrices
20 -    covx=newcovarianzasx; % de covarianzas de x e y obtenidos en la etapa 1
21
22 -    mediax=[50 20.05 9.9 9.9 30 10 30 30 40 50]';
23      % Valores nominales de x en columna
24
25 -    mediay=[0.1 0.15 0]; %Valores nominales de y
26
27 -    a=length(mediax); % número de variables x
28
29 -    b=length(mediay) ;% número de variables y
30
31
32 -    h=[0 0 0.1];
33
34      %Las especificaciones de Y son: y1>=0.0015 , y2>=0.0515 ,
35      %y3=0.0049+-0.0175 y4=0+-0.01]
36
37 -    LES=[1000 1000 0.1]; % En el caso de tener solo límite inferior
38      % ponemos un valor elevado para el límite superior (caso de y1,y2)
39
40 -    LEI=[0 0 -0.1];% en este caso tenemos limit inferior
41      % para todas las variables
42
43      %-------------------------------------------------------------------------
```

Por último se muestran los resultados obtenidos al finalizar la etapa:

```
nonconforming =

    0.0027

>> sum(proyecx)

ans =

    1.7020
```

```
LTSX =

   50.0459
   20.2425
    9.9951
   10.2386
   30.1795
   10.1921
   30.1750
   30.1723
   40.1551
   50.1558


LTIX =

   49.9541
   19.8575
    9.8049
    9.5614
   29.8205
    9.8079
   29.8250
   29.8277
   39.8449
   49.8442
```

Esta última imagen nos da los valores obtenidos para las tolerancias de las variables X al finalizar la optimización.

Podemos concluir que las tolerancias de las variables X que definen el problema son las siguientes.

$$X_1 = 50mm \pm 0.0459\text{mm} \Rightarrow X_1 \in [49.9541, 50.0459]\text{mm}$$
$$X_2 = 20.05mm \pm 0.1925\text{mm} \Rightarrow X_2 \in [19.8575,\ 20.2425]\text{mm}$$
$$X_3 = 9.9985mm \pm 0.0951\text{mm} \Rightarrow X_3 \in [9.8049, 9.9951]\text{mm}$$
$$X_4 = 9.9985mm \pm 0.3386\text{mm} \Rightarrow X_4 \in [9.5614, 10.2386]\text{mm}$$
$$X_5 = 30mm \pm 0.1795\text{mm} \Rightarrow X_5 \in [29.8205, 30.1795]\text{mm}$$
$$X_6 = 10mm \pm 0.1921\text{mm} \Rightarrow X_6 \in [9.8079, 10.1921]\text{mm}$$
$$X_7 = 30mm \pm 0.1750\text{mm} \Rightarrow X_7 \in [29.8250, 30.1750]\text{mm}$$
$$X_8 = 30mm \pm 0.1723\text{mm} \Rightarrow X_8 \in [29.8227, 30.1551]\text{mm}$$
$$X_9 = 40mm \pm\ 0.1551\text{mm} \Rightarrow X_9 \in [39.8449, 40.1551]\text{mm}$$
$$X_{10} = 50mm \pm 0.1558\ \text{mm} \Rightarrow X_{10} \in [49.8442, 50.1558]\text{mm}$$

4.3.2.3. Comparación con el caso de independencia

```
LTSX =

   50.5352
   20.6168
   10.4671
   10.2787
   30.3781
   10.3778
   30.3783
   30.5680
   40.5674
   50.5346


LTIX =

   49.4648
   19.4832
    9.3329
    9.5213
   29.6219
    9.6222
   29.6217
   29.4320
   39.4326
   49.4654

>> sum(proyecx)

ans =

    4.8519
```

Al igual que pasaba en el ejemplo anterior, para el caso de independencia obtenemos unas tolerancias mas amplias, esto supone que el sistema sea mas adecuado, aun cuando los costes del proceso podrían aumentar, ya que el considerar dependencia nos da un resultado mas real que si no consideramos.

4.3. RESUMEN PARA LA APLICACIÓN DE LA METODOLOGIA CON EL PROGRAMA MATLAB

Para terminar este capítulo se va a realizar un resumen de los cambios que se realizan en cada programa de Matlab de los necesarios para resolver los diferentes problemas mediante el método que se ha desarrollado en el proyecto para que se pueda cerca de manera sencilla:

PROGRAMA	MODIFICACIONES
Generación de la matriz de covarianzas	• Dimensiones de las piezas que forman el ensamblaje • Valores de los vectores que definen el comienzo de la iteración • Hay que tener en cuanta que este paso no sería necesario si el proceso fuera real ya que en este caso la matriz de covarianzas se obtendría del propio proceso de fabricación y no se obtendría de manera aleatoria.
Definición de la función objetivo	• Dimensión de la matriz total de varianzas y covarianzas • Definición de la matriz de covarianzas total en función del numero y tamaño las matrices obtenidas en el programa anterior
Definición de las restricciones	• Matriz A • Valor de las tolerancias mínimas para las variables X • Valor de las tolerancias de las variables Y
Etapa 1	• Vector para el punto inicial de la optimización • Vector del valor mínimo de aumento en cada iteración • Matriz A • Valor de las tolerancias mínimas para las variables X • Valor de las tolerancias de las variables Y

Calculo de la proporción de defectuosas	No se realiza ningún cambio,
Etapa 2	• Matriz A • Valor de las tolerancias de las variables Y • Vector con el valor nominal de las variables X • Vector con el valor nominal de las variables Y • Límites de especificación inferior y superior de las variables Y

Capítulo 5

CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN

CAPITULO 5: CONCLUSIONES Y FUTURAS LINEAS DE INVESTIGACIÓN

5.1. CONCLUSIONES

Se ha cumplido el objetivo principal del proyecto que era presentar de forma detallada, incluyendo ejemplos, la metodología propuesta por González y Sánchez (2008). El documento elaborado describe paso a paso la metodología y se han incluido tres ejemplos para facilitar su comprensión. Asimismo, se ha incluido los programas de Matlab utilizados para la resolución de los ejemplos, que pueden ser usados con pequeñas modificaciones para nuevos casos.

Tal y como se ha explicado a lo largo del proyecto, los métodos que existían anteriormente para el diseño óptimo de tolerancias no consideraban la correlación entre las variables, por lo que podemos concluir que el método que se ha desarrollado en este trabajo nos dará unos resultados mas precisos ya que sí consideramos esta relación. Los ejemplos que se han utilizado ilustran la importancia de tener en cuenta la dependencia de las variables, mostrando grandes desviaciones con respecto al caso de independencia. En algunos casos el considerar la correlación de las variables supuso mayores tolerancias lo cual es bueno, ya que implica una disminución de los costes de fabricación.

En cuanto al algoritmo de optimización utilizado para la resolución del problema, se puede concluir que la toolbox de optimización de Matlab es bastante potente y adecuada. En concreto la función “fmincon”. El poder usar Matlab facilita la aplicación de la metodología presentada en este proyecto, ya que el usuario no tendrá que trabajar minuciosamente con los algoritmos de optimización En el documento que se ha elaborado se incluyen los programas de Matlab utilizados. Estos pueden ser utilizados para otros casos similares realizando únicamente pequeños cambios en los programas.

5.1. FUTURAS LINEAS DE INVESTIGACIÓN.

El método desarrollado a lo largo de este trabajo permite muchas líneas de ampliación o desarrollo, a continuación se nombran algunas de ellas:

En primer lugar, una línea de ampliación de esta metodología podría ser el diseño de tolerancias considerando correlación entre las variables en el que además de modificar las tolerancias se modifiquen los valores nominales de las variables. Este caso se ha estudiado en el proyecto fin de carrera presentado por Jesús González (2010). Este método se basa en modificar, además de los autovalores de la matriz de covarianzas (que modifica el tamaño del elipsoide tal y como se hace en este trabajo), el valor de las media de las variables, lo que implica que además de modificar el tamaño del elipsoide lo desplaza en el espacio hasta encontrar el tamaño y lugar óptimo.

Otro caso para el que se podría estudiar una modificación o ampliación del método estudiado en este trabajo es aplicarlo para el caso de conjuntos mecánicos en el que las variables de las piezas que lo componen no siga una distribución normal multivariante como es el caso de este trabajo si no que pudiera seguir otro tipo de distribución para la cual los datos deberían ser tratados de manera diferente, o se debiera transformar esa distribución hasta convertirla en una distribución normal.

También podríamos considerar como otra línea de investigación aplicar otros algoritmos de optimización como por ejemplo los métodos de búsqueda en malla, métodos de búsqueda selectiva, métodos de simulación de cristalización o métodos de optimización basados en algoritmos genéticos.

Por último, sería de gran utilidad desarrollar una interfaz gráfica que permita asignar tolerancias a una pieza cualquiera, introduciendo previamente los datos necesarios. Esta interfaz podría desarrollarse en Matlab usando como base los programas presentados en este proyecto. La interfaz sería más fácil de utilizar ya que el usuario sería guiado para la aplicación de la metodología.

ANEXOS Y BIBLIOGRAFIA

BIBLIOGRAFIA Y ANEXOS

BIBLIOGRAFIA

- Pérez Díaz, J.L y Palacios Cuenca, S. Expresión Gráfica en la Ingeniería. Introducción al Dibujo Industrial. Universidad Carlos III de Madrid. Prentince Hall. 1998.

- Pestaña, D. ,Rodríguez, J.M. , Romera, E. ,Álvarez, V. y Portilla, A. Curso Práctico de Cálculo y Precálculo. Barcelona. Editorial Ariel, S.A. 2000.

- Peña, D. Análisis de Componentes Principales. Madrid. McGraw-Hill. 2002.

- Peña, D. Fundamentos de Estadística. Madrid. Alianza Universidad. 2005.

- González, I. y Sánchez, I. "Optimal design of Tolerances with correlated Variables" Mechanism and Machine Theory 44, pp. 1097-1107 (2009).

- Alfaro Navarro, J.L. "Métodos multivariantes en control estadístico de la calidad". Universidad de Castilla La Mancha. Albacete.

- D.B. Parkison. "The application of a robust design method to tolerancing"

- Woo-Jong Lee, Tony C. Woo, Shuo-Yan Chow. "Tolerance Synthesis for nonlinear Systems based on nolinear programming". IIE Transactions. University of Michigan, Ann Harbor, Michigan. Taylor&Francis. 1993.

- Manual de Matlab. Matlab. The language of Technical computing. The Math Works

- Hong Y.S. and Chang T.C., (2002). "A comprehensive review of tolerancing research". International Journal of Production Research, 40, pp. 2424-2459.

- Lee J. and Johnson G.E., (1993). "Optimal tolerance allotment using genetic algorithm and truncated Monte Carlo simulation". Computer Aided Design, 25, pp. 601-610.

- Lee, W.J. and Woo. T.C., (1990). "Tolerances: Their analysis and synthesis". Journal of Engineering Industry, 112, pp. 113-121.

- Lee, C. and Tang, G., (2000). “Tolerance design for products with correlated characteristics”. Mechanism and Machine Theory, 35, pp. 1675-1687.

- Nickerson, D., (1994) “Construction of a conservative confidence region from projections of an exact confidence region in multiple linear regression”. The American Statistician, 48, pp 120-13.

- Wu, C., Chen Z. znd Tang, G., (1998). “Component tolerance design for minimum quality loss and manufacturing cost”. Computers in Industry, 35, pp. 223-232.

- González García, J., (2010) Proyecto fin de Carrera. “Metodología para la asignación de tolerancias y valores nominales a un conjunto de variables correlativas”. Universidad Carlos III de Madrid

ANEXO 1. ARTÍCULO EJEMPLOS 1 Y 2.

“Optimal Design of Tolerances with correlated variables”

Mechanism and Machine Theory 44 (2009) 1097–1107

Contents lists available at ScienceDirect

Mechanism and Machine Theory

journal homepage: www.elsevier.com/locate/mechmt

Statistical tolerance synthesis with correlated variables

Isabel González [a,*,1], Ismael Sánchez [b]

[a] Department of Mechanical Engineering, Universidad Carlos III de Madrid. Avd. de la Universidad 30, 28911, Leganés, Madrid, Spain
[b] Department of Statistics, Universidad Carlos III de Madrid. Avd. de la Universidad 30, 28911, Leganés, Madrid Spain

ARTICLE INFO

Article history:
Received 16 August 2008
Received in revised form 8 October 2008
Accepted 18 October 2008
Available online 6 December 2008

Keywords:
Tolerance synthesis
Tolerance allocation
Correlated variables

ABSTRACT

Optimal tolerance design aims at assigning tolerances such that the functionality requirements are achieved with minimum cost. Classical tolerance design procedures are based on the assumption of independence of variables. This assumption might not be realistic, leading to the assign of non-optimal tolerances. This paper proposes an innovative methodology to allocate optimal statistical tolerances to dependent variables, where the dependence structure is estimated from the manufacturing process. The methodology is based on the assumption that the multivariate process variability is consequence of a set of independent factors. Hence, the tolerance assignment should be based on the statistical properties of these factors. The tolerance design takes the independence of the factors as a restriction, and tolerances are optimally assigned according to the variability of them.

1. Introduction

The quality and functionality of parts are usually related with one or more variables $\mathbb{Y} = (Y_1, Y_2, \ldots, Y_J)'$, such as length, width, weight, voltage, and so on. The functionality of the part imposes some specification limits on $\mathbb{Y}$. These variables can, in turn, be a function of other variables $\mathbb{X} = (X_1, X_2, \ldots, X_K)'$. For example, the quality of a part could be defined by the variable Y_1 = maximum allowable load. This variable, in turn, depends on the length X_1, width X_2, and thickness X_3, of the part. Similarly, in a mechanical assembly, there are often certain characteristics of quality $\mathbb{Y}$, that are a function of the dimensions of the individual components that compose the mechanical assembly. In cases like these, the specification limits of $\mathbb{Y}$ imposes restrictions on the values of $\mathbb{X}$, denoted as tolerance limits. The tolerances of $\mathbb{X}$ should be assigned so that the specification limits of $\mathbb{Y}$ are met with minimum cost. In this sense, whereas specifications of $\mathbb{Y}$ are imposed by the functionality requirements, tolerances of $\mathbb{X}$ should be optimally designed. In the literature, this tolerance design of $\mathbb{X}$ is denoted as tolerance synthesis or tolerance allocation.

Tolerances of manufactured components have a significant impact on quality and manufacturing costs. Tight tolerances assures the functionality requirements of $\mathbb{Y}$ with very high probability; but it might lead to an increase in manufacturing costs. That is, too tight tolerances impose an expensive production process with very low variability. In contrast, loose tolerances may lead to increased waste and assembly problems, which lead to high costs due to lack of quality (quality costs). For this reason, tolerance synthesis is frequently formulated as an optimization problem for the optimal design of a product in terms of functionality and cost.

It is important to note that optimal tolerance synthesis is linked to the variability of the process, both with the variances and the covariances. Hence, an estimation of the covariance matrix of $\mathbb{X}$ is a critical input to reach a feasible tolerance design.

* Corresponding author.
E-mail addresses: ismael@est-econ.uc3m.es (I. González), ismael@est-econ.uc3m.es (I. Sánchez).
URL: http://turan.uc3m.es/uc3m/dpto/IN/dpin11/fabdis/imgfaria.html (I. González).
[1] Tel.: +34 916245860; fax: +34 916249430.

0094-114X/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mechmachtheory.2008.10.006

This estimation of the covariance matrix of $\mathbb{X}$ has to be done with an initial sample of data of $\mathbb{X}$, in a similar fashion as in the design of control charts.

The most used method for tolerance synthesis is based on a statistical approach (statistical tolerance synthesis). Under this approach, a statistical distribution of the variables $\mathbb{X} = (X_1, X_2, \ldots, X_K)'$ is assumed as a result of the variability of the production process. An important aspect of this statistical approach is the dependence of the variables $\mathbb{X}$, which is consequence of the manufacturing process. The statistical distribution of $\mathbb{Y}$ is derived from the distribution of $\mathbb{X}$ and can be used to compute the proportion of parts out of specification limits; that is, the non-conforming proportion of parts p. The tolerances of $\mathbb{X}$ are assigned so that they are as wide as possible but keeping the non-conforming proportion p in minimal acceptable values. This statistical criterion is more efficient than using a deterministic criterion. Under a deterministic criterion, also known as worst case tolerancing, the tolerances are assigned so that the 100% of the parts meet all the specifications. This practice leads to tighter tolerances, and hence higher manufacturing costs.

In the field of statistical tolerance synthesis, there is a large number of research works for the case of independence of $\mathbb{X}$. For example, the methodologies proposed in Lee and Woo [1], Lee and Johnson [2], Wu et al. [3], Feng and Balasu [4], Lee and Tang [5], Ye and Salustri [6] and Huang et al. [7]. Although in some cases this assumption could be valid, there are many other cases where variables $\mathbb{X}$ are related. For example, in processes like molding or press stamping the length, width and thickness of a part are all related. In spite of the key importance of the dependence of $\mathbb{X}$ in the tolerance design, the literature based on the dependence of $\mathbb{X}$ is rather scarce.

In Chen [8], a set of independent variables $\mathbb{X}$ become dependent as a result of a selective assembly. For example, a device could be assembled in such a way that parts with high values of X_1 are assembled with parts that have a low value of X_2; that is to say, a negative relationship between X_1 and X_2 is imposed. This selective assembling provokes a correlation that should be taken into account in the optimal tolerance design. Lee et al. [9] propose an optimal tolerance synthesis where the correlation matrix of $\mathbb{X}$ is taken into account. Their non-linear programming algorithm finds the optimal solution by modifying the standard deviations σ_i independently according to so-called feasible directions and keeping the correlation matrix constant. As will be proven in Section 3, this treatment of the dependence structure might not be realistic. In that section, it will be seen that in order to preserve the characteristics of the manufacturing process, the standard deviations σ_i can not be changed independently, since they are linked by the eigenvectors of the covariance matrix.

In this paper, we propose a methodology to allocate optimal statistical tolerances to dependent variables $\mathbb{X}$, where the dependence structure of $\mathbb{X}$ is inherent to the manufacturing process. The manufacturing process is assumed to be defined by a set of independent (latent) factors that explain the dependence structure of $\mathbb{X}$. Under the assumption of normality, these independent factors can easily be computed using principal component analysis (PCA). These independent factors are related to the engineering design of the process and may not be under discussion in the tolerance design. Consequently, the optimal tolerance design should be based on the statistical properties of these independent factors.

The outline of the article is as follows. Section 2 describes briefly the common criteria considered in the current procedures of tolerance synthesis under independence of $\mathbb{X}$. These criteria will be used as a reference point in the proposed methodology. Section 3 describes the basic considerations of the proposed methodology. Section 4 describes the steps of the procedure. Section 5 applies the proposed methodology to two examples cited in the literature, and compares the results with those obtained by assuming independence of $\mathbb{X}$. Section 6 concludes.

2. A review of tolerance synthesis under independence of *X*

2.1. General considerations

Many of the methodologies proposed in the literature under the assumption of independence of $\mathbb{X}$ have some aspects in common. That is the case, for example, of the methodologies proposed in Wu et al. [3], Feng and Balasu [4], Lee and Tang [5], Ye and Salustri [6], Huang et al. [7] and Chen [8]. These common aspects will be described in this section. They will serve as a reference point for the methodology proposed in this article.

Let Y be a characteristic that ensures the functionality of a part. A part is conforming if the variable Y is inside some specifications; that is, $Y \in [LSL, USL]$, where LSL and USL are the lower and the upper specification limits, respectively. These limits are considered symmetric about the mean of Y. It is assumed that under optimal conditions, the non-conforming proportion p is not larger than some small value α; that is

$$p \equiv P(Y \notin [LSL, USL]) \leqslant \alpha \tag{1}$$

The variable Y is a function of $\mathbb{X}$, that represents a set of dimensions of a part or mechanical assembly; that is, $Y = f(X_1, X_2, \ldots, X_K)$. The traditional tolerance synthesis problem consists on finding the optimal tolerances of K independent variables $\mathbb{X} = (X_1, X_2, \ldots, X_K)'$, where the tolerances are of type $X_i \in [LTL_i, UTL_i]$, $i = 1, 2, \ldots, K$. Here, LTL_i is the lower tolerance limit and UTL_i is the upper tolerance limit and both define a symmetric interval about the mean of X_i. These tolerance intervals form a rectangular tolerance region T_X defined as

$$T_X = \left\{ \mathbb{X} \in R^K : (LTL_i \leqslant X_i \leqslant UTL_i),\ i = 1, \ldots, K \right\}. \tag{2}$$

The tolerances are assigned to X_i such that the total cost (manufacturing and quality) is minimized and (1) is attained.

In the existing literature, it is assumed that $X_i \sim N(\mu_i, \sigma_i^2)$, where μ_i is the mean and σ_i^2 is the variance. The manufacturing process is assumed centered on an adequate value (for example, a nominal value). Then, the means μ_i, $i = 1,\ldots,K$, will be kept constant and the tolerances of X_i can be expressed as a function of the standard deviations σ_i. It is customary to use the so-called six-sigma approach (Harry and Stewart [10]; Hong and Chang [11]. Under this approach, and under the assumption of symmetric tolerances, the tolerance limits of X_i are expressed in terms of a number of standard deviations σ_i from the mean. If it is also assumed that the capability index of the variable X_i is $C_p = 1$, then the tolerances for a given X_i would be at $\mu_i \pm 3\sigma_i$; that is

$$X_i \in [LTL_i, UTL_i] = [\mu_i - 3\sigma_i, \mu_i + 3\sigma_i] = [\mu_i - t_i, \mu_i + t_i]. \tag{3}$$

Although some authors have proposed different cost models, here, and for the sake of simplicity, costs are considered the same for all variables and proportional to the length of the tolerances. Under this assumption, the optimization of tolerances consists on the optimization (maximization) of $\mathbf{t} = (t_1,\ldots,t_k)'$ in (3) subject to the restriction (1). This can be expressed as

$$\text{Max} \sum_{i=1}^{K} t_i, \quad \text{subject to } p \leqslant \alpha, \text{ and } \mathbf{t} \geqslant 0. \tag{4}$$

Since $t_i = 3\sigma_i$ the optimization can be expressed as the optimization (maximization) of the K standard deviations σ_i as

$$\text{Max} \sum_{i=1}^{K} \sigma_i, \quad \text{subject to } p \leqslant \alpha, \text{ and } \sigma_i \geqslant 0; \quad i = 1,\ldots,K. \tag{5}$$

Note that σ_i does not need to be the real variability of the manufacturing process. If the optimal σ_i are larger than the variability of the manufacturing process, then we would have $C_p > 1$. Conversely, if the optimal σ_i are smaller than the actual ones, the process is not capable and should be improved.

In order to verify the constraint (1), it is necessary to estimate the distribution of Y. Then the relationship $Y = f(X_1, X_2,\ldots,X_K)$ should be defined. In general, it is assumed that $f(\cdot)$ has a known analytical form, and that it is a linear relationship of the form $Y = a_0 + a_1X_1 + \cdots + a_KX_K$, or $Y = a_0 + \mathbf{a}'\mathbb{X}$, where $\mathbf{a} = (a_1, a_2,\ldots,a_K)'$. Under this assumption, $Y \sim N(\eta_Y, \sigma_Y^2)$, where

$$\eta_Y = a_0 + a_1\mu_1 + \cdots + a_K\mu_K, \tag{6}$$

and, by the independence of X_i

$$\sigma_Y = \sqrt{a_1^2\sigma_1^2 + a_2^2\sigma_2^2 + \cdots + a_K^2\sigma_K^2}. \tag{7}$$

If the linear relationship is only approximately true, it is possible to approximate $f(\cdot)$ using a first-order Taylor expansion. In this case, the coefficients $a_1, a_2,\ldots,a_K$ are equal to the first derivative of Y with respect to each variable X_i, that is, $a_i = \delta Y/\delta X_i$. The derivatives are evaluated at $X_i = \mu_i$. When $f(\cdot)$ is highly non-linear, an extended Taylor series expansion can be applied. Some related methods can be seen in Evans [12]. Once the distribution of Y has been determined, the proportion p could be computed according to normal probability theory.

2.2. The general procedure

Now we will describe the steps that are most common in the literature under the independence assumption. The steps of the procedure are as follows, where, for simplicity it is assumed that $\mu_i = 0$, $i = 1, 2,\ldots,K$, and hence $\eta_Y = 0$.

1. Set $r = 0$. Start from some initial values of $\sigma_i^{(r)}$, $i = 1, 2,\ldots,K$ (for instance, the estimations from a sample of data).
2. Compute $\sigma_Y^{(r)}$ by replacing the values of $\sigma_i^{(r)}$ in (7).
3. Compute the non-conforming proportion $p = P(Y \notin [LSL, USL] \mid Y \sim N(0, \sigma_Y^{2(r)}).)$.
4. If $\alpha - p > \varepsilon$, where ε is a small value, set $r = r + 1$. Assign a new value to each $\sigma_i^{(r)}$ and go to step 2. Otherwise, go to step 5.
5. Compute $t_i = 3\sigma_i^{(r)}$. Then, tolerance limits are $LTL_i = \mu_i - t_i$ and $UTL_i = \mu_i + t_i$.

The step 4 can be made using different optimization methods. For example, gradient-based methods like Lagrange multipliers, SQP algorithm, or direct search methods like genetic algorithm, neural networks and simulated annealing. This procedure can be extended to the case where both μ_i and σ_i are optimized, as it is the case of the tolerance design of integrated circuits.

This problem can also be extended to the case of multivariate $\mathbb{Y} = (Y_1, Y_2,\ldots,Y_J)'$, and consequently there is a set of J specification limits $Y_j \in [LSL_j, USL_j]$. Authors as Chen [8] and Lee and Woo [1] analyzed this case. In these articles the variables $\mathbb{Y}$ are treated as independent. The non-conforming proportion p is then computed for each variable Y_j. This practice do not assure a total non-conforming proportion $p \leqslant \alpha$, due to the problem of simultaneous confidence regions (see, for instance, Nickerson [13]). In order to alleviate this problem, Lee and Woo [1] apply a correction to adjust the desired non-conforming proportion α by using and individual α_J for each Y_j so that $\alpha_J = 1 - (1 - \alpha)^{(1/J)}$. Although independence of $\mathbb{Y}$ is assumed, this assumption may not be valid, since $\mathbb{Y}$ depends on the same variables $\mathbb{X}$.

3. Basic concepts of the proposed methodology assuming dependence of $\mathbb{X}$

This section describes the basis of the proposed methodology for the allocation of tolerances to dependent variables $\mathbb{X}$. This methodology also considers a set of correlated variables $\mathbb{Y}$ where such correlation comes from $\mathbb{X}$. The objective is to find the optimal tolerance region T_X for a set of K dependent variables $\mathbb{X} = (X_1, X_2, \ldots, X_K)'$ so that the set of m dependent variables $\mathbb{Y} = (Y_1, Y_2, \ldots, Y_m)'$, verifies

$$p \equiv P(\mathbb{Y} \notin S) \leqslant \alpha, \tag{8}$$

where S is the rectangular specification region defined by

$$S = \left\{ \mathbb{Y} \in R^m : (LSL_j \leqslant Y_j \leqslant USL_j),\ j = 1, \ldots, m \right\}. \tag{9}$$

Here, LSL_j and USL_j are symmetric limits about the mean of Y_j. These specifications can be generalized to unilateral specifications, where LSL_j or USL_j are infinite. For the sake of simplicity, the costs are considered equal for all the variables and therefore the tolerance allocation is a problem of maximization of tolerances of $\mathbb{X}$, subject to (8), as in (4). Multivariate normality for $\mathbb{X}$ is assumed; that is, $\mathbb{X} \sim N_K(\mu_x, \Sigma_x^0)$, where μ_x is the mean vector and Σ_x^0 is the covariance matrix.

Both μ_x and Σ_x^0, depend on the manufacturing process and should be estimated from data. In absence of data, Σ_x^0 could not be estimated, and the tolerance allocation could initially be made assuming independence of $\mathbb{X}$. As is customary, the tolerance are optimized as a function of the covariance matrix. In a more general setting, the optimization would also involve the mean vector.

A known linear relationship between $\mathbb{X}$ and $\mathbb{Y}$ is assumed. Therefore, each variable Y_j can be expressed as $Y_j = a_{0j} + a_{1j}X_1 + \cdots + a_{Kj}X_K$. When the linear relationship is only approximately true, a Taylor expansion can be applied. In this case, the coefficients a_{ij} will be equal to $a_{ij} = \delta Y_j / \delta X_{ij}$. Under the linearity assumption, it holds that $\mathbb{Y} \sim N_m(\eta_y, \Sigma_y^0)$. The mean vector η_y verifies

$$\eta_y = \mathbf{a} + \mathbf{A}\mu_x', \tag{10}$$

where $\mathbf{a} = (a_{01}, a_{02}, \ldots, a_{0j})$ and $\mathbf{A}$ is the matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{k1} \\ a_{12} & a_{11} & & a_{k2} \\ \vdots & & \ddots & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{km} \end{bmatrix},$$

and Σ_y^0 verifies

$$\Sigma_y^0 = \mathbf{A}\Sigma_x^0\mathbf{A}'. \tag{11}$$

The basis of the proposed methodology is similar to the one used in the independent case. In that case, the maximization of t_i was made through the maximization of σ_i (see 3). Now, t_i depends on all the elements of the covariance matrix of $\mathbb{X}$. Then, the maximization of the tolerances t_i is obtained through some optimization of a covariance matrix Σ_x. Σ_x can be interpreted as the covariance matrix of an artificial process that verifies (8) with the largest possible tolerances of $\mathbb{X}$. Note that Σ_x is not (necessarily) the covariance matrix of the process, which is Σ_x^0. Σ_x is obtained after some manipulation of Σ_x^0 with the goal of searching for maximum tolerances.

The problem then is how to find this matrix Σ_x. It will be seen in next subsection that the searching of the optimal Σ_x should be made according to some restrictions. The main idea is that although the variables $\mathbb{X}$ might not be independent, they may be considered as a combination of independent factors $\mathbb{Z} = (Z_1, Z_2, \ldots, Z_K)'$ that might not be directly measurable. In multivariate analysis, these independent factors are usually known as latent factors. These independent factors can be interpreted as the primary source of the variability and dependence structure of $\mathbb{X}$. Therefore, any change in Σ_x will necessarily be provoked by variations in the variance of the independent factors $\mathbb{Z}$, which will always be independent. Hence, the optimization of the matrix Σ_x should be based on the latent factors $\mathbb{Z}$.

Consequently, the methodology for the tolerance allocation in the dependent case should solve the following three problems: (a) the relationship between tolerances and Σ_x, (b) a procedure to modify Σ_x to find the optimal one but taking into account the dependence structure of $\mathbb{X}$ (through the independent factors $\mathbb{Z}$); and (c) a procedure to evaluate the restriction (8). Next subsections are devoted to these problems.

3.1. Relationship between tolerances and Σ_x

Under multivariate normality of $\mathbb{X}$, the region that concentrates the $100(1-\beta)\%$ of the distribution is a hyperellipsoid $H_{X,\beta}$ of dimension K, defined as

$$H_{X;\beta} = \left\{ \mathbb{X} \in R^K \mid (\mathbb{X} - \mu_x)'\Sigma_x^{-1}(\mathbb{X} - \mu_x) = \chi^2_{K;1-\beta} \right\}, \tag{12}$$

where $\chi^2_{K;1-\beta}$ is the $100(1-\beta)\%$ point of the chi-square distribution with K degrees of freedom. Note that this hyperellipsoid depends not only on σ_i but on the covariances of $\mathbb{X}$. As in (3) for the independent case, the tolerance limits $[LTL_i, UTL_i]$ need to be expressed as some distance from the mean, where that distance is related to $\mathbf{\Sigma}_x$.

We should remind that the tolerance region of $\mathbb{X}$ is defined as the intervals $[LTL_i, UTL_i]$, forming a hyperrectangle of dimension K, centered on μ_x and with sides parallel to $X_1, X_2, \ldots, X_K$. Consequently, in order to relate the tolerances of $\mathbb{X}$ with $\mathbf{\Sigma}_x$ the hyperrectangle should be related to the hyperellipsoid (12). This can be attained using as tolerance hyperrectangle the projection of the hyperellipsoid (12) on the axes $X_1, X_2, \ldots, X_K$. The semi-lengths of the sides of such hyperrectangle will be the tolerances t_i. Those semi-lengths are determined by solving the systems of equations of first derivative, with respect to each X_i of the quadratic form in (12), leading to

$$t_i = \sqrt{\left(\mathbf{\Sigma}_x^{-1}\right)_{ii}\chi^2_{K;1-\beta}}, \tag{13}$$

where the sub-index ii denotes the ith element of the diagonal of the inverse of $\mathbf{\Sigma}_x$ (see, for instance, Hardle and Simar [14, Chapter 2]). The tolerance limits for each variable X_i, $i = 1, 2$, are then

$$X_i \in [LTL_i, UTL_i] = [\mu_i - t_i, \mu_i + t_i]. \tag{14}$$

Fig. 1 illustrates this projection in two dimensions. Region A represents the ellipse with $100(1-\beta)\%$ coverage of the distribution of $\mathbb{X}$, and region B represents the rectangle obtained from the projection of the ellipse. That rectangle defines the tolerances for each variable X_1, X_2. Eqs. (13) and (14) show us the tolerances that correspond to a given matrix $\mathbf{\Sigma}_x$. The next problem is to find the optimal matrix $\mathbf{\Sigma}_x$; i.e., the matrix that maximizes the tolerances t_i subject to (8).

3.2. Procedure to modify $\mathbf{\Sigma}_x$

In the case of independence of $\mathbb{X}$, the standard deviations σ_i can be optimized independently of each other. Now, in the case of dependence of the variables $\mathbb{X}$, the tolerance limits depend on the whole covariance matrix and not only on the variance σ_i^2 of the diagonal. Therefore, in this case, the whole covariance matrix $\mathbf{\Sigma}_x$ needs to be optimized. At each iteration of the optimization process, it will be necessary to change $\mathbf{\Sigma}_x$. This change in $\mathbf{\Sigma}_x$ will imply a new set of tolerance limits using (13) and a new $\mathbf{\Sigma}_y$ using (11), which in turn will be used to compute the non-conforming proportion p. Fig. 2 shows an example of this idea in two dimensions. Fig. 2a represents an ellipse defined by $\mathbf{\Sigma}_x$ and the tolerance region obtained using (13) and (14). Fig. 2b represents the new ellipse obtained by transforming $\mathbf{\Sigma}_x$ to $\mathbf{\Sigma}_x^*$, and the resulting larger tolerance region.

As mentioned above, the variables $\mathbb{X}$ are not independent, but may be considered as a combination of independent latent factors $\mathbb{Z}$ that might not be directly measurable. Therefore, any change in $\mathbf{\Sigma}_x$ will necessarily be provoked by variations in the variance of the independent factors $\mathbb{Z}$. Under normality, the independent factors can be easily obtained using principal component analysis (PCA). PCA allows to obtain the variances of $\mathbb{Z}$ and the relationship between these variances and $\mathbf{\Sigma}_x$. Since the procedure takes into account the structure of the latent factors, the resulting $\mathbf{\Sigma}_x$ will be compatible with the characteristics of the manufacturing process. Next, we describe the steps of the procedure to obtain $\mathbf{\Sigma}_x$ from $\mathbf{\Sigma}_x^0$. This procedure is used in each iteration of the optimization algorithm shown in Section 4.

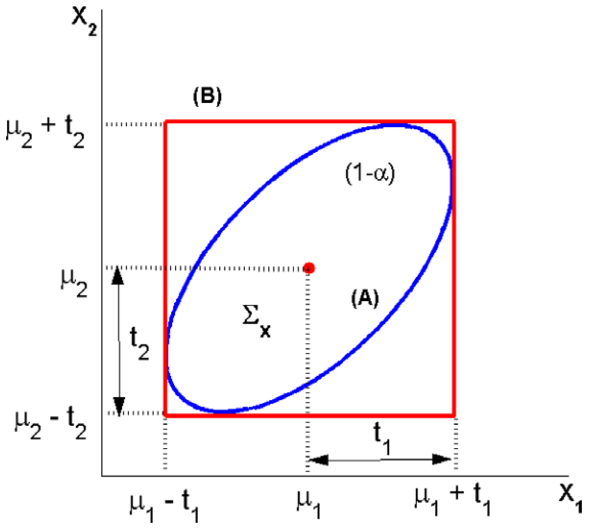


Fig. 1. Region (A): elliptical region defined by Σ_x. Region (B): tolerance region.

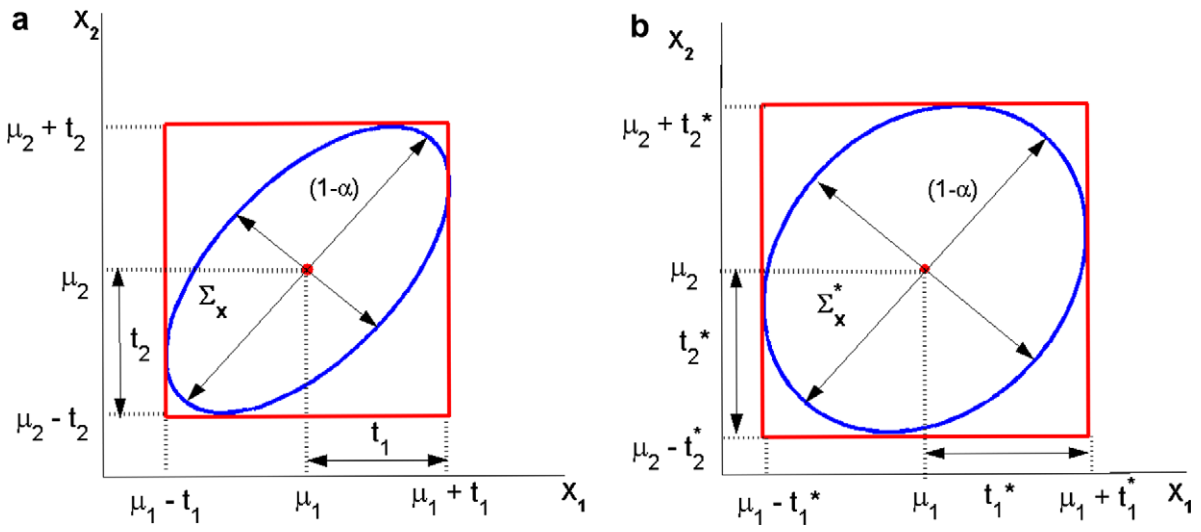


Fig. 2. Ellipse and tolerance region defined by: (a) Σ_x. (b) Σ_x^*.

STEP 1: Obtain the singular value decomposition of the covariance matrix $\mathbf{\Sigma}_x^0$, that can be written as

$$\mathbf{\Sigma}_x^0 = \mathbf{C}\mathbf{D}^0\mathbf{C}', \tag{15}$$

where $\mathbf{C}$ is the matrix of eigenvectors with columns $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K$, and $\mathbf{D}^0$ is the diagonal matrix with the eigenvalues $\lambda_1^0, \lambda_2^0, \ldots, \lambda_K^0$. The matrix $\mathbf{D}^0$ is the covariance matrix of the independent factors $\mathbb{Z}$, and the eigenvalues λ_i^0 are the variances of $\mathbb{Z}$. The eigenvectors represent the directions of the axes of the hyperellipsoid defined by $\mathbf{\Sigma}_x^0$ and the eigenvalues represent the length of these axes. Then, a change in a eigenvalue means a change in the length of one axis of the hyperellipsoid. Conversely, a change in the eigenvectors means a change in the directions of the axes, which is a dramatic change in the process. Therefore, if we look for changes that preserve the dependence structure, we should only modify the eigenvalues, and not the directions of the axes (eigenvectors). Changes in the eigenvectors would only take place if the manufacturing process is redesigned, for instance allowing that originally independent factors become dependent.

STEP 2: Modify the variance of the independent factors $\mathbb{Z}$, which means to modify the values of $\lambda_1^0, \lambda_2^0, \ldots, \lambda_K^0$. A new set of eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_K$, can be obtained as

$$\lambda_i = b_i\lambda_i^0, \tag{16}$$

where b_i are positive coefficients. Since the eigenvalues represents the size of the axes of the hyperellipsoid, different coefficients b_i will imply that the hyperellipsoid change its shape, maintaining the original direction of its axes. The use of the same coefficient b for all eigenvalues will imply that the hyperellipsoid increases or decreases its size, maintaining its initial shape.

STEP 3: Obtain a new covariance matrix $\mathbf{\Sigma}_x$. Let us denote as $\mathbf{D}$ the diagonal matrix with elements $\lambda_1, \lambda_2, \ldots, \lambda_K$. Hence, by (15)

$$\mathbf{\Sigma}_x = \mathbf{C}\mathbf{D}\mathbf{C}'. \tag{17}$$

By this procedure, the changes in $\mathbf{\Sigma}_x$ depends on the coefficients b_i. Therefore, the optimization of $\mathbf{\Sigma}_x$ can be treated as the optimization of these K coefficients b_i. In each iteration of the optimization process, the b_i are changed independently. This procedure leads to an optimization problem similar to the case of independence of $\mathbb{X}$, where the optimization variables were the K independent standard deviations σ_i.

This idea of searching for a new matrix $\mathbf{\Sigma}_x$ but preserving the eigenvectors $\mathbf{C}$ is a key element in the design of realistic tolerances. Failing to do it will produce tolerances that ignore the actual dependence of the variables. The resulting tolerances might be difficult to attain in a real situation. González and Sánchez [15] uses this idea to simulate out of control situations in multivariate processes.

3.3. Procedure to verify the criterion $p \leqslant \alpha$

Since the optimization problem consists on maximizing the tolerance of $\mathbb{X}$ subject to obtaining a non-conforming proportion $p \leqslant \alpha$, a procedure to verify this criteria, at each iteration of the optimization process, must be defined. For a given $\mathbf{\Sigma}_x$ obtained in (17), and the corresponding new $\mathbf{\Sigma}_y$, the restriction (8) should be verified. The verification of this restriction by

the computation of p may require complicated numerical integration or slow Monte Carlo simulations. Consequently, it is interesting to apply efficient procedures to verify (8) that avoid this complexity. To this aim, two procedures for the evaluation of the restriction (8) are proposed. The first one is a simple procedure that allows a fast evaluation of (8) that does not need to compute p. This procedure is approximate, but leads to the allocation of conservative tolerances to $\mathbb{X}$ in the sense that $p < \alpha$. The second procedure is more accurate but computationally more expensive, and it is only used to assure that $p = \alpha$. We describe now these procedures.

3.3.1. A simple procedure to verify $p \leqslant \alpha$

The non-conforming proportion is $p = P(\mathbb{Y} \notin S \mid \mathbb{Y} \sim N_m(\eta_y, \mathbf{\Sigma}_y))$, where η_y and $\mathbf{\Sigma}_y$ are obtained as in (10) and (11) for a given $\mathbf{\Sigma}_x$. Under the normality assumption, there is a hyperellipsoid $H_{Y;\alpha}$ of dimension m of coverage $100(1-\alpha)\%$ defined as

$$H_{Y;\alpha} = \left\{ \mathbb{Y} \in \mathbb{R}^m \mid (\mathbb{Y} - \eta_y)' \mathbf{\Sigma}_y^{-1} (\mathbb{Y} - \eta_y) = \chi^2_{m;1-\alpha} \right\}.$$

We will denote $H_{Y;\alpha}$ as process hyperellipsoid. Following the same arguments as in (13) for the case of variables $\mathbb{X}$, we can obtain a hyperrectangle that is tangent to this process hyperellipsoid. The semi-lengths of the sides of this hyperrectangle corresponding to $\mathbf{\Sigma}_y$ and of coverage $100(1-\alpha)\%$, denoted as l_j, are

$$l_j = \sqrt{\left(\mathbf{\Sigma}_y^{-1}\right)_{jj} \chi^2_{m,1-\alpha}}; \tag{18}$$

where the sub-index jj denotes the jth diagonal element of the inverse of $\mathbf{\Sigma}_y$. The sides of this process hyperrectangle, that will be denoted as $[LL_j, UL_j], j = 1, 2, \ldots, m$, can be obtained as $LL_j = \eta_j - l_j$ and $UL_j = \eta_j + l_j$. We will denote this hyperrectangle as the process hyperrectangle.

The procedure is then very simple and consists in checking if this process hyperrectangle is inside the specification hyperrectangle S in (9). Should that be the case, restriction (8) would hold. It is important to note that the process hyperrectangle is a conservative region. Because the process hyperrectangle completely contains the process hyperellipsoid, the probability of being outside the process hyperrectangle is lower than α. Consequently, even if $[LL_j, UL_j] = [LSL_j, USL_j], j = 1, 2, \ldots, m$, the non-conforming proportion p will be lower than α. Fig. 3a shows an example for $m = 2$. In this figure, the process ellipse has a coverage of $100(1-\alpha)\%$ and hence the tangent process rectangle is a region of coverage greater than $100(1-\alpha)\%$. Since this process rectangle is completely inside the rectangular specification region, it holds that $p \leqslant \alpha$.

3.3.2. Exact procedure to verify $p \leqslant \alpha$

In this procedure, we compute p using Monte Carlo simulations. In this simulation, it should be taken into account that only those parts inside the tolerance region T_X are used. Hence, the non-conforming proportion p can be written as

$$p = P(\mathbb{Y} \notin S \mid \mathbb{Y} = f(\mathbb{X}); \mathbb{X} \in T_X)). \tag{19}$$

The procedure can be summarized as follows. For a given $\mathbf{\Sigma}_x$ and applying (13) and (14) we compute the corresponding tolerance region T_X. Then, we generate Monte Carlo simulation from $\mathbb{X} \sim N_K(mu_x, \mathbf{\Sigma}_x)$ and select only the replications inside T_X. We will denote as $\mathbf{X}_T$ to these multivariate replications. Next, we compute the corresponding values of $\mathbb{Y}$ as $\mathbf{Y}_T = f(\mathbf{X}_T)$. Finally, we compute p as the sampling proportion of replications outside S.

4. The tolerance synthesis procedure

The procedure for the optimal tolerance design has two stages. In the first stage, a conservative tolerances are obtained using a fast procedure. These tolerances are conservatives in the sense that the resulting non-conforming proportion is $p < \alpha$. In the second stage, the tolerances are increased as much as possible until $p = \alpha$.

4.1. Stage 1

1. Set the counter $r = 0$. Estimate the initial covariance matrix $\mathbf{\Sigma}_x^{(0)}$, and the mean vector μ_x from the manufacturing process data. Obtain the mean vector η_y by replacing μ_x in (10).
2. By using this matrix $\mathbf{\Sigma}_x^{(0)}$ in (15), obtain the eigenvectors matrix $\mathbf{C}$, and the initial diagonal matrix $\mathbf{D}^{(0)}$ with eigenvalues $\lambda_1^{(0)}, \lambda_2^{(0)}, \ldots, \lambda_K^{(0)}$.
3. Compute the tolerances $t_i^{(r)}$ according to (13) using $\mathbf{\Sigma}_x^{(r)}$ and a desired value of β, for instance $\beta = 0.0027$.
4. Obtain the matrix $\mathbf{\Sigma}_y^{(r)}$ by replacing $\mathbf{\Sigma}_x^{(r)}$ in (11).
5. Obtain the semi-lengths $l_j^{(r)}, j = 1, \ldots, m$, by using the inverse of the matrix $\mathbf{\Sigma}_y^{(r)}$ and the value $\chi^2_{m,1-\alpha}$, in (18) for a desired α. Compute the limits of the process hyperrectangle as $LL_j^{(r)} = \eta_j - l_j^{(r)}$ and $UL_j^{(r)} = \eta_j + l_j^{(r)}$.
6. Check if the goal (4) is attained. The restriction $p \leqslant \alpha$ is verified using the simple method in Section 3.3.1; that is, we check the restriction

$$\left(LL_j^{(r)} \geqslant LSL_j\right) \cap \left(UL_j^{(r)} \leqslant USL_j\right), \quad j = 1, \ldots, m.$$

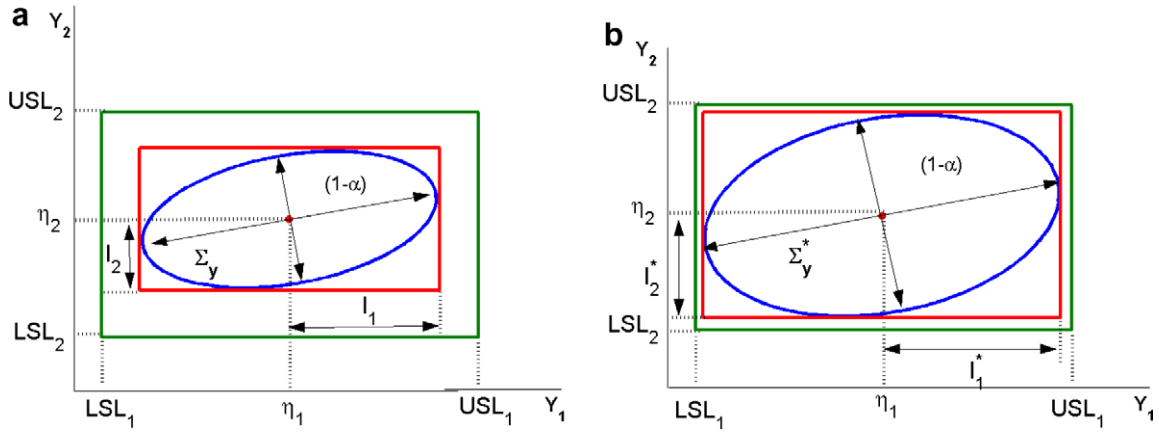


Fig. 3. (a) Initial ellipse inside S. (b) Ellipse inside S with larger volume obtained in the next iteration.

7. If maximum is attained, go to Stage 2 below. Otherwise set $r = r + 1$. Obtain a new covariance matrix $\mathbf{\Sigma}_x^{(r)}$ following the procedure exposed in Section 3.2. That is, compute a new set of eigenvalues as $\lambda_i^{(r)} = \lambda_i^{(0)} b_i^{(r)}$. These new eigenvalues will be the diagonal elements of a new matrix $\mathbf{D}^{(r)}$. By replacing the matrix $\mathbf{C}$ and the new matrix $\mathbf{D}^{(r)}$ in (15), obtain the new covariance matrix $\mathbf{\Sigma}_x^{(r)}$. Go to step 3. The values $b_i^{(r)}$ are the variables to be optimized. Traditional optimization methods can be used. In the examples below, the sequential quadratic programming (SQP) method has been used.

A geometric interpretation of this stage is that we are searching for a hyperellipsoid defined by $\mathbf{\Sigma}_x$ with maximum volume such that the process hyperrectangle defined by the corresponding $\mathbf{\Sigma}_y$ is inside S. The changes in the volume of the hyperellipsoid defined by $\mathbf{\Sigma}_x$ are performed by changing its shape but keeping the direction of its axes. Fig. 3 shows an example of two iterations in two dimensions. Here, the larger rectangle represents the specification region S. Fig. 3a shows an iteration with a covariance matrix $\mathbf{\Sigma}_y$ corresponding to the $\mathbf{\Sigma}_x$ illustrated in Fig. 2a. Fig. 3b shows a consecutive iteration with a new covariance matrix $\mathbf{\Sigma}_y^*$, corresponding the new $\mathbf{\Sigma}_x^*$ illustrated in Fig. 2b. It can be seen that at this iteration tolerances of $\mathbb{X}$ are larger, and still the restriction $p < \alpha$ holds.

4.2. Stage 2

In Stage 2, and as opposed to Stage 1, the procedure used to verify (8) will be based on the exact computation of p exposed in Section 3.3.2. The steps are the following.

1. Set $r = 0$ and $b^{(r)} = 1$. Let us denote as $\mathbf{\Sigma}_x^{(R)}$ to the covariance matrix obtained at the end of the Stage 1.
2. Obtain a new matrix $\mathbf{\Sigma}_x^{(r)}$. We will use a similar procedure as in step 7 of the Stage 1. However, since we are already close to the optimal, and for the sake of simplicity, the same value of $b^{(r)}$ in (16) is used for all the eigenvalues. This procedure is equivalent to multiply the covariance matrix of $\mathbb{X}$ by b. Hence, we set $\mathbf{\Sigma}_x^{(r)} = b^{(r)} \mathbf{\Sigma}_x^{(R)}$.
3. Compute the tolerance limits of $\mathbb{X}$, $[LTL_i^{(r)}, UTL_i^{(r)}]$, $i = 1, \ldots, K$, by using (13) and (14) and the matrix $\mathbf{\Sigma}_x^{(r)}$.
4. Generate B replications of a normal $N_K(\mu_x, \mathbf{\Sigma}_x^{(r)})$ and select the values $\mathbf{X}_T^{(r)}$ inside the tolerance region.
5. Obtain $\mathbf{Y}_T^{(r)} = f(\mathbf{X}_T^{(r)})$ and compute the proportion $p^{(r)}$ of values outside S.
6. Check if $p^{(r)} = \alpha$. If not, set $r = r + 1$, set a new value $b^{(r)}$ and go to 2. Otherwise, the search is finished and the tolerance limits for X_i, $i = 1, \ldots, K$, are $[LTL_i^{(r)}, UTL_i^{(r)}]$.

In this stage, the optimization variable is only b. The optimization process can hence be performed as a univariate optimization problem and can be solved using traditional univariate searching methods.

5. Illustrative examples

5.1. Example 1

This example is related with the tolerance allocation of a leaf spring and was proposed by Lee and Tang [5] in the context of tolerance allocation under independence of $\mathbb{X}$. The leaf spring has two functional characteristics: spring rate, Y_1, and the maximum allowable load, Y_2. These characteristics depend on the thickness X_1, the width X_2, and the length X_3 of the leaf spring. These variables $\mathbb{X} = (X_1, X_2, X_3)'$ are considered dependent variables because of the manufacturing process. Following Lee and Tang [5] the relationship $\mathbb{Y} = f(\mathbb{X})$ can be written as

$$Y_1 = \frac{EX_2X_1^3}{4X_3^3}, \quad Y_2 = \frac{\sigma_b X_2 X_1^2}{6X_3}, \tag{20}$$

where E = 21,000 kg/mm^2 is the modulus of elasticity and σ_b = 49.2 kg/mm^2 is the yield stress based of the material used in the leaf spring. The problem is the allocation of the maximum tolerances for $\mathbb{X}$ so that the non-conforming proportion is α = 0.0027. The specification requirements are $Y_1 \in [0.268, 0.308]$ and $Y_2 > 17.62$. Besides these specification requirements, the tolerance design is restricted to obtain tolerances of $\mathbb{X}$ higher than 0.01 mm. Following Lee and Tang [5], the nominal values of $\mathbb{X}$ are $\mathrm{NV}_x = (5, 40, 450)'$ and hence the nominal values of $\mathbb{Y}$ are $\mathrm{NV}_y = (0.288, 18.22)'$. The normality of $\mathbb{X}$ is assumed as $\mathbb{X} \sim N_3(\mu_x, \mathbf{\Sigma}_x)$. For simplicity let us assume that the process is centered at the nominal value. Therefore, $\mu_x = (5, 40, 450)'$ and $\eta_y = (0.288, 18.22)'$. In practice, the covariance matrix $\mathbf{\Sigma}_x^{(0)}$ should be estimated from the manufacturing process data. In this illustrative example, this covariance has been randomly generated, obtaining

$$\mathbf{\Sigma}_x^{(0)} = \begin{bmatrix} 0.0003 & 0.0012 & 0.0009 \\ 0.0012 & 0.0089 & -0.0011 \\ 0.0009 & -0.0011 & 0.0089 \end{bmatrix}.$$

Since the relationship (20) is not linear, a first-order Taylor expansion is applied, and then $\mathbb{Y} \approx .\mathbf{a}_0 + \mathbf{A}\mathbb{X}$, where $\mathbf{a}_0 = (0, -18.23)$ and the matrix $\mathbf{A}$ is equal to

$$\mathbf{A} = \begin{bmatrix} 0.17284 & 0.0072 & -0.00192 \\ 7.2889 & 0.4556 & -0.04049 \end{bmatrix}.$$

Therefore, the mean vector of $\mathbb{Y}$ can be written as $\eta_y = \mathbf{a}_0 + \mathbf{A}\eta_y = (0.288, 18.22)'$, and the initial covariance matrix of $\mathbb{Y}$ can be obtained using (11) as

$$\mathbf{\Sigma}_y^{(0)} = \begin{bmatrix} 0.000012 & 0.00055 \\ 0.00055 & 0.02528 \end{bmatrix},$$

and hence the variables $\mathbb{Y}$ are also dependent. Following the proposed methodology, the Stage 1 will consists on the optimization of the three coefficients b_1, b_2 and b_3 in (16). For this example, a sequential quadratic programming (SQP) method is used as implemented in the function fmincon in Matlab. The SQP method approximates the Hessian of the Langragian function by using a quasi-Newton updating method. This matrix is then used to generate a quadratic subproblem whose solution is used to form a search direction for a line search procedure. An overview of SQP can be found in Fletcher [16].

Once the factors b_1, b_2 and b_3 have been optimized, the Stage 2 is performed as an univariate optimization process. In this case, the coefficient b is the optimization variable. An searching algorithm implemented by the authors, that combines the methods of secant and bisection, has been used. Finally, the optimal tolerances of $\mathbb{X}$ are equal to: $X_1 = 5 \pm 0.037$ mm, $X_2 = 40 \pm 1.098$ mm and $X_3 = 450 \pm 1.063$ mm.

The same problem has been solved for the case of independence of $\mathbb{X}$. The methodology described in Section 2 has been used. In this case, $\mathbf{\Sigma}_x^{(0)}$ is a diagonal matrix with elements 0.0003, 0.0089 and 0.0089, respectively. In this case, the tolerances of $\mathbb{X}$ are equal to: $X_1 = 5 \pm 0.01$ mm, $X_2 = 40 \pm 0.067$ mm and $X_3 = 450 \pm 9.381$ mm. As can be seen from the results, the dependence between variables significantly influences the allocation of the tolerances, especially for X_2 and X_3.

5.2. Example 2

This example illustrates the tolerance design for a mechanical assembly and was proposed in Lee and Woo [1] and Lee and Johnson [2]. The mechanical assembly has two individual parts as shown in Fig. 4. The involved variables $\mathbb{X} = (X_1, X_2, \ldots, X_8)'$ are the lengths showed in Fig. 4. The mechanical assembly specifications are defined in terms of the variables $\mathbb{Y} = (Y_1, Y_2, Y_3, Y_4)$. The variable Y_1 represents a size condition for the base part. The other variables Y_2, Y_3, Y_4 reflect the clearance conditions between the two parts. The relationship $\mathbb{Y} = f(\mathbb{X})$ is

$$\begin{aligned} Y_1 &= X_4 + X_5, \\ Y_2 &= X_2 - X_1 - X_8 + X_7, \\ Y_3 &= X_7 - X_6 - X_3 + X_2, \\ Y_4 &= X_4 - X_3 - X_6. \end{aligned}$$

Hence, the matrix $\mathbf{A}$ can be written as

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 \end{bmatrix}.$$

The specification requirements are (in mm) $Y_1 \leqslant 127.13$, $Y_2 \geqslant 0.0076$, $Y_3 \geqslant 0.0254$ and $Y_4 \geqslant 0.0076$. Besides these specification requirements, the tolerance of $\mathbb{X}$ must be higher than 0.01 mm. The nominal values of $\mathbb{X}$ and $\mathbb{Y}$ are $\mathrm{NV}_x = (25.4, 50.8, 76.2, 101.6, 25.4, 25.3, 50.8, 76.1)'$ and $\mathrm{NV}_y = (127, 0.1, 0.1, 0.1)'$. As in Example 1, it is assumed that

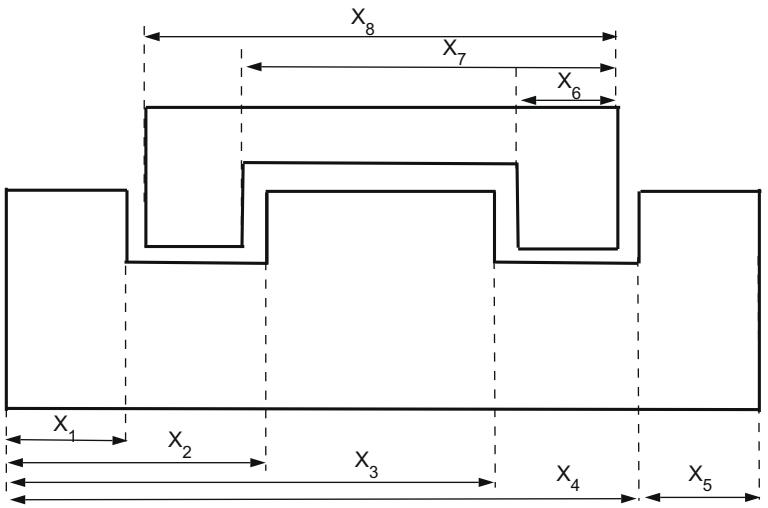


Fig. 4. Mechanical assembly of Example 2.

$\mathbb{X} \sim N_8(\mu_x, \Sigma_x)$, $\mu_x = \mathrm{NV}_x$ and $\eta_y = \mathrm{NV}_y$. Also, as is Example 1, the initial covariance matrix $\Sigma_x^{(0)}$ has been obtained randomly. In a practical situation, this matrix is estimated from data. In this case, there are two individual parts so that two groups of correlated variables are considered. The first group corresponds to the variables X_1, X_2, X_3, X_4 and X_5, and the second one corresponds to the variables X_6, X_7 and X_8. Firstly, a covariance matrix for each group of variables is randomly generated. Then, the total covariance matrix is constructed. The resulting block matrix $\Sigma_x^{(0)}$ is

$$\Sigma_x^{(0)} = \begin{bmatrix} 0.0102 & 0.0034 & 0.0066 & 0.0078 & 0.0107 & 0 & 0 & 0 \\ 0.0034 & 0.0020 & 0.0022 & 0.0025 & 0.0036 & 0 & 0 & 0 \\ 0.0066 & 0.0022 & 0.0051 & 0.0050 & 0.0068 & 0 & 0 & 0 \\ 0.0078 & 0.0025 & 0.0050 & 0.0108 & 0.0086 & 0 & 0 & 0 \\ 0.0107 & 0.0036 & 0.0068 & 0.0086 & 0.0433 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0027 & 0.0010 & 0.0007 \\ 0 & 0 & 0 & 0 & 0 & 0.0010 & 0.0005 & 0.0002 \\ 0 & 0 & 0 & 0 & 0 & 0.0007 & 0.0002 & 0.0004 \end{bmatrix}.$$

The tolerances of $\mathbb{X}$ should be assigned in order to obtain $p = 0.0027$.

The same algorithms used in Example 1 has been applied in this example. The optimal tolerances of $\mathbb{X}$ are equal to: $X_1 = 25.4 \pm 0.217$ mm, $X_2 = 50.8 \pm 0.076$ mm, $X_3 = 101.6 \pm 0.151$ mm, $X_4 = 76.2 \pm 0.236$ mm, $X_5 = 25.4 \pm 0.243$ mm, $X_6 = 25.3 \pm 0.113$ mm, $X_7 = 50.8 \pm 0.046$ mm and $X_8 = 76.1 \pm 0.030$ mm. In order to compare these results, the tolerances assuming independence of $\mathbb{X}$ are also obtained. The tolerances under independence are: $X_1 = 25.4 \pm 0.055$ mm, $X_2 = 50.8 \pm 0.027$ mm, $X_3 = 101.6 \pm 0.041$ mm, $X_4 = 76.2 \pm 0.064$ mm, $X_5 = 25.4 \pm 0.103$ mm, $X_6 = 25.3 \pm 0.041$ mm, $X_7 = 50.8 \pm 0.027$ mm and $X_8 = 76.1 \pm 0.055$ mm. It can be seen that the hypothesis of independence of $\mathbb{X}$ has an important effect in their tolerances.

6. Conclusion

The traditional methods for tolerance design ignore the statistical properties of the multivariate process. Without the use of this important information, tolerances will not be optimal. This article uses the actual multivariate dependence in the optimization process. Hence, it requires some initial data set to estimate the dependence structure. As opposed to some others methods in the literature, tolerances are design taking into account the singular value decomposition of the dependence structure of data. This property is a desired one since, as shown in the article, the dependence structure of the manufacturing process heavily relies on the eigenvectors and eigenvalues. Some examples illustrate the importance of taking the dependence into account, showing large departures with respect to the independent case. Besides, the methodology is simple to use. Some simple computations, that can be based on Monte Carlo simulations and some standard optimization algorithm, suffice to attain the optimal values.

Acknowledgements

The authors wish to express their appreciation to an anonymous referees for valuable comments. This research was supported in part by CICYT Grants SEJ2007-64500, CCG07-UC3M/DPI and DPI2005-08018.

References

[1] W.J. Lee, T.C. Woo, Tolerances: their analysis and synthesis, Journal of Engineering Industry 112 (1990) 113–121.
[2] J. Lee, G.E. Johnson, Optimal tolerance allotment using genetic algorithm and truncated Monte Carlo simulation, Computer Aided Design 25 (1993) 601–610.
[3] C. Wu, Z. Chen, G. Tang, Component tolerance design for minimum quality loss and manufacturing cost, Computers in Industry 35 (1998) 223–232.
[4] C. Feng, R. Balasu, Robust tolerance design considering process capability and quality loss, Conceptual and Innovative Design for Manufacturing DE, vol. 103, ASME, New York, 1999. pp. 1–14.
[5] C. Lee, G. Tang, Tolerance design for products with correlated characteristics, Mechanism and Machine Theory 35 (2000) 1675–1687.
[6] B. Ye, F.A. Salustri, Simultaneous tolerance synthesis for manufacturing and quality, Research in Engineering Design 14 (2003) 98–106.
[7] M. Huang, Y. Zhong, Z. Xu, Concurrent process tolerance design based on minimum product manufacturing cost and quality loss, International Journal of Advanced Manufacturing Technology 25 (2005) 714–722.
[8] M.S. Chen, Optimising tolerance allocation for mechanical components correlated by selective assembly, International Journal of Advanced Manufacturing Technology 12 (1996) 349–355.
[9] W.J. Lee, T.C. Woo, S.Y. Chou, Tolerance synthesis for nonlinear systems based on nonlinear programming, IIE Transactions 25 (1993) 51–61.
[10] M.J. Harry, R. Stewart, Six-Sigma Mechanical Design Tolerancing, Motorola University Press, Shaumburg, IL, 1988.
[11] Y.S. Hong, T.C. Chang, A comprehensive review of tolerancing research, International Journal of Production Research 40 (2002) 2425–2459.
[12] D.H. Evans, Statistical tolerancing: the state of the art. Part II, Methods for estimating moments Journal of Quality Technology 7 (1975) 1–12.
[13] D. Nickerson, Construction of a conservative confidence region from projections of an exact confidence region in multiple linear regression, The American Statistician 48 (1994) 13–120.
[14] W. Hardle, L. Simar, Applied Multivariate Statistical Analysis, Springer-Verlag, Berlin, 2003.
[15] I. González, I. Sánchez, Principal alarms in multivariate statistical process control, Journal of Quality Technology 40 (2008) 19–30.
[16] R. Fletcher, Practical Methods of Optimization, John Wiley and Sons, 1987.

ANEXO 2. ARTÍCULO EJEMPLO 3.

“Tolerance Synthesis for Nonlinear Systems base don Nonlinear Programming”

TOLERANCE SYNTHESIS FOR NONLINEAR SYSTEMS BASED ON NONLINEAR PROGRAMMING

WOO-JONG LEE
Technical Center, Daewoo Motor Co., Incheon, Korea

TONY C. WOO AND SHUO-YAN CHOU
Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan 48109-2117

Automatic assignment of tolerances to dimensioned mechanical assemblies is studied as an optimization problem; the objective of which is to minimize the (manufacturing) cost, subject to the constraints of (design) functionality and (assembly) interchangeability. By associating a nominal dimension and a tolerance to the variance, a probabilistic approach is adopted.

Trigonometric functions relating the component geometries give rise to the nonlinearity in the system. Estimating an n-dimensional nonlinear integral by a polytope converts the probabilistic optimization formulation to a deterministic one. It also allows rapid evaluation of tolerance analysis embedded in tolerance synthesis.

Local optimality is ensured by analysis of convexity and quasi-concavity of the objective function and some of the constraints. Sensitivity analysis is performed to provide search directions for global optimality. An implementation is reported with an example.

■ Tolerances in an engineering design are intended to capture variations from the ideal, as introduced by the very process of realization such as manufacturing and assembly. Nominal dimensions specify idealized geometries by size, location and form. The range between the upper and lower limits of the variation from the nominal dimension is called *tolerance* [1].

At the design stage, functionality, performance and reliability are the major issues under consideration. Tolerance, or variation from the ideal, should be set as close to zero as possible to satisfy those considerations. However, high precision means tight tolerances which are usually associated with high cost; thus, looser tolerances are more desirable at the manufacturing stage. At the assembly stage, for the interchangeability of the components, tight tolerances are desirable. While design and assembly prefer tight tolerancing, the virtual components (from design) must be brought to realization by manufacturing before the physical components can be assembled. This requirement results in a three-way trade-off amongst design, manufacturing and assembly as shown in Figure 1.

The concurrent consideration of opposing criteria on tolerancing, between design and manufacturing, and between manufacturing and assembly, needs to be resolved. The result is effectively a synthesis of tolerances. This paper deals with formulations and computational techniques for tolerance synthesis by analysis.

Basic to manufacturing is cost. To assign cost-effective tolerances, a probabilistic approach is taken (rather than the deterministic approach) so as to perform trade-off analysis. The approach, simply stated, is to convert the designer's "function-oriented" specifications into manufacturable specifications by allocating the tolerances to the ideal dimensions such that the manufacturing cost is minimized. Models of tolerance-cost functions [10], [19], [22], [23], [25] are employed. Figure 2 illustrates a typical inverse relation of manufacturing costs to tolerances: the tighter the tolerance the higher the cost.

A major obstacle to probabilistic tolerancing has been the amount of CPU time required for simulation. To reduce the time for the computation, Parkinson [20] and the authors [14], [15] used the notion of reliability index [9] as an approximation. However, the computation of such an approximation is still intensive. In [14], a local circular search was used to find a global optimal solution. Michael and Siddall [19] decomposed the random variable space into orthogonal n-dimensional cubes, where n is the number of dimensions considered; that gives $O(2^n)$ cells to be tested. Other attempts which impose restrictions on the domain of the problem such as, linearity [2], [7], [11], [24], [25], [26] and single design constraint [22], have practical limitations.

The purpose of this paper is to provide a general framework for tolerance synthesis for nonlinear systems

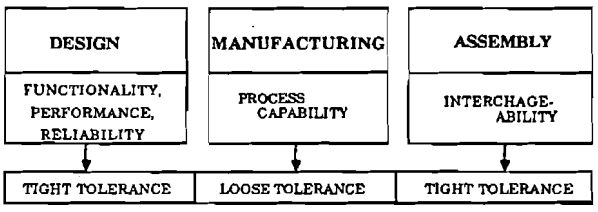


Figure 1. Concurrent consideration of tolerance

 0740-817X/93/$3.00x.00

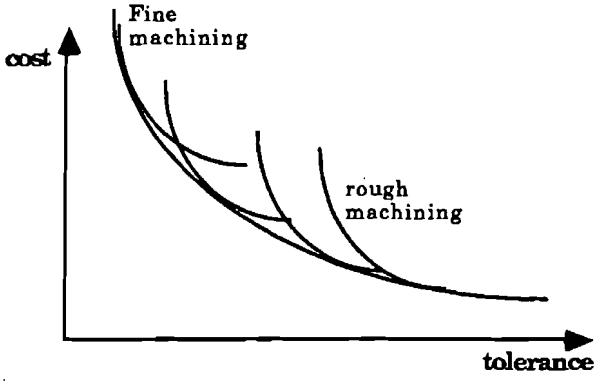


Figure 2. A typical tolerance-cost curve

with multiple, dependent design constraints. (The design constraints are termed "dependent" if they share the same dimension variables. For instance, the two design constraints $x_1 - x_2 - 0.1 \leq 0$ and $x_1 + x_3 - 0.05 \leq 0$ are dependent because they share the same variable x_1.) Least-cost tolerance synthesis is mathematically formulated as a nonlinear programming (NLP) problem. Algorithms are provided and illustrated by an example.

Basic Concepts

Design Function

Unlike manufacturing, whose principal concern is to produce in-spec individual component, design and assembly are concerned with inter-component relations. Typically, functionalities of a product are captured by specifying them as design and assembly requirements. Consider a simple assembly as shown in Figure 3. Suppose the clearance between the components is to be greater than or equal to 0.01 to achieve certain performance threshold and to obtain certain level of assemblability. This requirement can be expressed mathematically as:

$$x_1 - x_2 \geq 0.01, \tag{1}$$

where x_1 and x_2 denote the dimensions of the hole and the shaft respectively. Rewrite (1) as an inequality with all the terms on the left-hand side and let $F(x,c)$ be a function denoting the left-hand side:

$$F(x,0.01) = -x_1 + x_2 + 0.01 \leq 0. \tag{2}$$

Functions such as $F(x,0.01)$ are called *design functions*. These design functions describe the inter-component relations, and provide a mathematical basis for controlling functionality and interchangeability. 0.01 is referred to as a *design constant* of the design function $F(x,0.01)$; the design constants are typically specified by designers. A design function is not always linear. For example, as illustrated in Figure 4, some of the design functions take on trigonometric terms.

A linear system is one in which an assembly dimension y is defined by a linear combination of component dimensions $x_1, x_2, \ldots, x_n$:

$$y = \sum_{j=1}^{n} a_j x_j, \tag{3}$$

where a_j is a signed binary integer. Suppose x_j are random variables. As variations accumulate (or "stack-up," as it is commonly referred to), the tolerance analysis for linear systems becomes useful for understanding the probabilistic functionality and assemblability, and has been studied extensively [2], [7], [11], [24], [25], [26]. The key property that facilitates tolerance analysis is that the variance of the linear sum is the sum of the variances of component dimensions under the assumption of independence, that is,

$$\sigma_y^2 = \sum_{j=1}^{n} (a_j)^2 \sigma_{x_j}^2. \tag{4}$$

When tolerance t_j of dimension x_j is defined as $\pm 3\sigma_j$ from the mean μ_j, the stack-up tolerance t_y can be then expressed as

$$t_y = k_y \sqrt{\sum_{j=1}^{n} (a_j)^2 \left(\frac{t_j}{6}\right)^2}, \tag{5}$$

where k_y is a constant derived from the sustainable percentage λ_y of rejected product. In the case of normally distributed x_j, $k_y = 2 * \Phi^{-1}\left(1 - \frac{\lambda_y}{2}\right)$, where $\Phi(\cdot)$ is the is the cumulative distribution function of standard normal distribution. However, the analysis procedure for a linear system can not be extended to nonlinear systems because of the lack of a general rule for an aggregate such as equation (4).

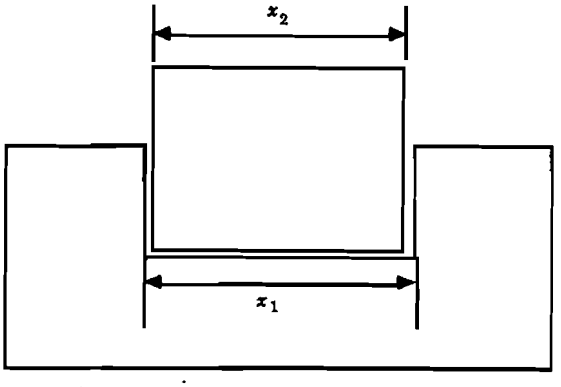
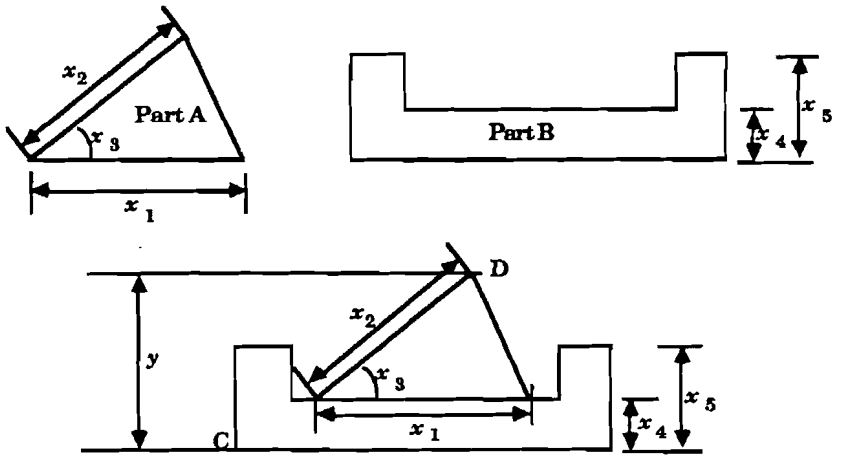


Figure 3. An assembly



y : vertical distance from C (of part B) to D (of part A) is equal to $x_4 + x_2 \sin x_3$

Figure 4. A non-linear design function

Yield

The dimension vector $\boldsymbol{x}$ is assigned to follow the multivariate normal distribution. (Indeed, Mansoor [18] shows that most manufacturing processes produce dimensions with normal distributions.) Let μ_j and σ_j denote the mean and the standard deviation of the normally distributed random variable x_j. The mean μ_j is typically fixed by the designer, whereas the standard deviation σ_j is chosen according to the precision of the control exercised over the manufacturing process. This parameter σ_j is therefore a function of the tolerance t_j. According to current standard practice, σ_j is normally set to $\frac{t_j}{6}$. Consequently, if t_j is given, x_j is a well defined normally distributed random variable.

Probabilistic tolerance analysis can be stated more succinctly in a mathematical formulation: Given tolerances t_j (or standard deviations σ_j), determine the probability such that the design function $F(\boldsymbol{x},c)$ is satisfied, i.e., $Pr(F(\boldsymbol{x},c) \leq 0)$. In other words, the *yield* can be obtained by evaluating the following integral,

$$\text{yield} \equiv Y(t) = \int_{F(\boldsymbol{x},c)\leq 0} f(\boldsymbol{x};\mu,t,\boldsymbol{R})\, d\boldsymbol{x} \qquad (6)$$

where $f(\boldsymbol{x};\mu,t,\boldsymbol{R})$ is the probability density function of multivariate normal vector $\boldsymbol{x}$ for which μ, t, and $\boldsymbol{R}$ denote the mean vector, the tolerance vector (standard deviation vector), and the correlation matrix of $\boldsymbol{x}$, respectively. (If the random variables x_j are independent, it will not be necessary to specify covariances.)

It helps the intuition to visualize the interaction between tolerances and the design function. Consider an assembly of two random variables, x_1 and x_2. A design function $F(\boldsymbol{x},c)$ partitions the two-dimensional space into two regions. The *safe region* $\mathbf{R}_S$ in Figure 5(a) corresponds to the region in which $F(\boldsymbol{x},c) \leq 0$. (The complement of the safe region is called the *failure region* $\mathbf{R}_F$.) Now, the given tolerances also prescribe a region in the same space. In Figure 5(b), the upper and lower limits of a random variable x_j define a strip. Intersecting the strips for x_1 and x_2 gives the *tolerance region* $\mathbf{R}_T$. It is noted that the size of $\mathbf{R}_T$ varies with tolerances, but $\mathbf{R}_S$ (or $\mathbf{R}_F$) is independent of tolerances. Combining $\mathbf{R}_T$ with $\mathbf{R}_S$ gives the *reliable region* $\mathbf{R}_R$ as shown in Figure 5(c). To be able to achieve the required reliability of production, $\mathbf{R}_R$ must be non-empty.

Tolerance Analysis

Yield of Linear Design Function

While this paper addresses systems with nonlinear design functions, discussion of the linear case facilitates subsequent development as non-linear functions will be approximated by hyperplanes. The yield of a linear design function $F(\boldsymbol{x},c)$ as computed from (6), involves multiple integration. An approximation method which can be extended to the nonlinear case adopts the notion of the *reliability index*, which was introduced by Hasofer and Lind [9]. (The observation that a linear combination of multivariate normal random variables follow a univariate normal distribution [6, p.56] does not extend readily to the case of nonlinear design functions.) Consider Figure 6, in which there are two independent random variables z_1 and z_2 both following the standard normal distribution $N(0,1)$. The (standardized) linear design function of this example is of the form $\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3$. The desired yield $Pr(\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 \leq 0)$ is equal to $\Phi(\beta)$, where

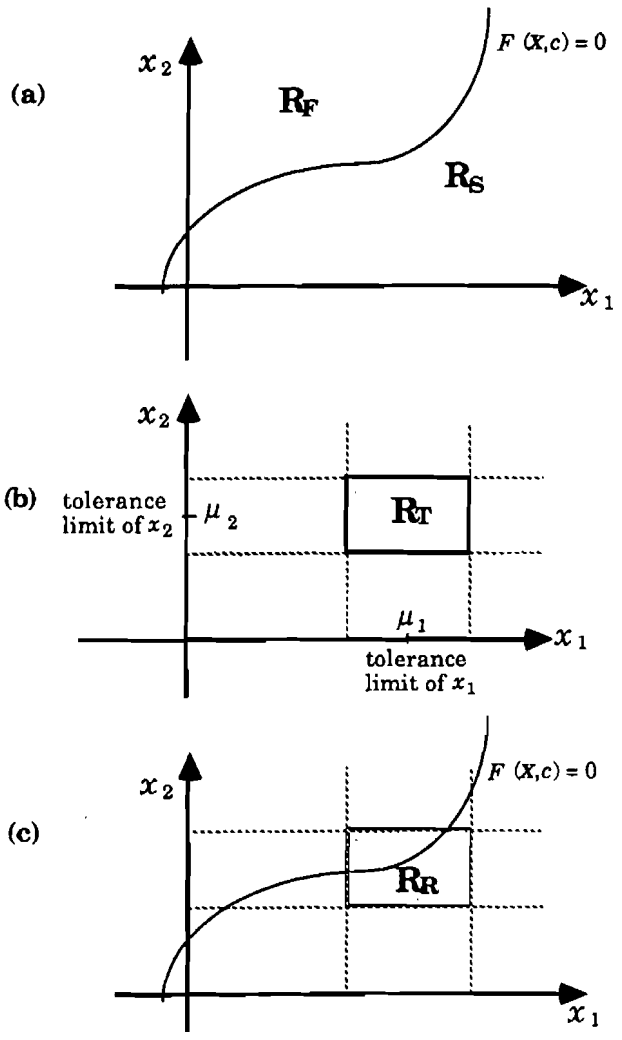


Figure 5. Safe, tolerance and reliable regions

$$\beta = \frac{|\alpha_3|}{\sqrt{\alpha_1^2 + \alpha_2^2}}$$

is the minimum distance from the design function to the origin. The distance β is referred to as the *reliability index* of the design function. Because of the rotational symmetry of the standard coordinates, the yield can be obtained by looking up the value of $\Phi(\beta)$ in the univariate standard normal distribution table. The above technique is next generalized for computing the yield of a linear design function with any number of random variables.

In general, tolerance analysis for a linear design function $F(x,c) = a^T x + c$ requires the following steps:

(i) **Standardization**: Transform the dependent normal variables to the independent standard normal variables by using

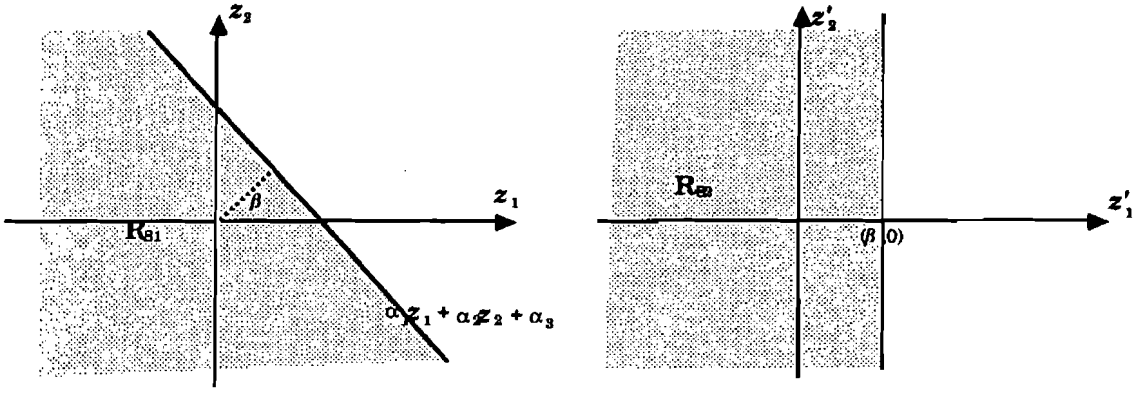
$$z = (PD)^{-1}(x - \mu) \qquad (7)$$

where z is the transformed standard normal vector and P is the orthogonal matrix for diagonalizing a given covariance matrix V such that $P^T V P = DD^T$. (The detailed transformation procedure is given in Appendix A.) The transformed z space is referred to as the *standard coordinates*.

(ii) **Reliability Index**: Compute the minimum distance β from the origin to the transformed design function in the standard coordinates. First, represent the design function in terms of z:

$$a^T x + c = a^T PDz + a\mu + c. \qquad (8)$$

The distance from the origin to the right hand side of equation (8) corresponds to β which can be then expressed as



$$Pr(R_{S1}) = Pr(\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 \leq 0) = \Phi(\beta) = Pr(R_{S2})$$

Figure 6. Rotational symmetry of standard coordinate

$$\beta = \frac{-a^T\mu - c}{\sqrt{(a^TPD)(a^TPD)^T}} = \frac{-a^T\mu - c}{\sqrt{a^TVa}} \quad (9)$$

since $PD(PD)^T = PDD^TP^T = P(P^TVP)P^T = V$. Note that the numerator of equation (9), $-a^T\mu - c$, is always positive since the given nominal dimensions, μ, are assumed to satisfy the design condition.

(iii) **Table Look-Up**: Look up the univariate standard normal distribution table for $\Phi(\beta)$.

Yield of Nonlinear Design Functions

To solve integral (6) with nonlinear function $F(x,c)$ and multivariate normal PDF, two techniques are considered: (i) Monte-Carlo simulation [5], [8] and (ii) approximation through the linearization of $F(x,c)$ [14], [15], [20].

Monte-Carlo simulation starts with generating N sets of random samples $(x_{11}, x_{21},\ldots,x_{n1}),\ldots,(x_{1N}, x_{2N},\ldots,x_{nN})$ from the given multivariate normal PDF, where x_{jl} denotes the l-th sample ($l = 1,\ldots,N$) of the j-th dimension ($j = 1,\ldots,n$). Then, each set is substituted into the design function and the sign of the functional value is checked. Suppose T sets out of N sets have a nonpositive sign. Then, the estimate of the yield is T/N. While Monte-Carlo simulation can be applied to the linear or nonlinear $F(x,c)$, its computation is time intensive because a large number of random samples needs to be taken to obtain an accurate result. This time intensive phenomenon is compounded in *tolerance synthesis*, whose algorithm engages iterative tolerance analysis to estimate the yield and the gradient.

In order to reduce the computational time for approximating yield, the linearization of nonlinear design functions is applied. This linearization is achieved through Taylor series expansion at the expansion point of the nonlinear design function. (The first order Taylor series expansion gives the tangent hyperplane passing through the expansion point.) Note that the probability density in the standard coordinate decreases exponentially as the distance from the origin increases. This suggests that to be a well-selected expansion point the (probabilistically) densest area should be preserved after linearization.

The "densest" area is, therefore, estimated by the minimum distance between the nonlinear function and the origin. Each design function is standardized by transformation (7), and the point on each standardized design function which has the minimum distance from the origin is selected as the expansion point. Consider two points p_1 and p_2 on $G(z,\gamma)$, the standardized $F(x,c)$, as candidates for expansion points. In Figure 7, lines L_1 and L_2 correspond to linearization of $G(z,\gamma)$ through p_1 and p_2.

The point on $G(z,\gamma) = 0$ that has the minimum distance from the origin is obtained by solving the following single constraint NLP:

$$\text{Min } \beta = \sqrt{z^Tz}, \text{ subject to } G(z,\gamma) = 0. \quad (10)$$

Based on equation (7), formulation (10) can be rewritten with the original variables:

$$\text{Min } \beta^2 = (x-\mu)^TV^{-1}(x-\mu),$$
$$\text{subject to } F(x,c) = 0. \quad (11)$$

Here, the objective function is expressed as β^2 instead of β since the positive definiteness of the covariance matrix V always guarantees the same solution. As a solution scheme for (11), the iterative method based on Newton-Raphson method [17] is used:

$$x^{(k+1)} = \mu + V\nabla F(x^{(k)},c)\frac{(x^{(k)}-\mu)^T\nabla F(x^{(k)},c) - F(x^{(k)},c)}{\nabla F(x^{(k)},c)^TV\nabla F(x^{(k)},c)} \quad (12)$$

where $x^{(k)}$ denotes the solution after the k-th iteration and $\nabla F(x,c)$ is the $n \times 1$ gradient vector of $F(x,c)$ at x. As an initial solution for (12), the mean nominal vector μ is used.

Tolerance analysis for nonlinear design function therefore takes the following steps:

(i) **Expansion Point**: Find the expansion point x^* by applying equation (12) iteratively.

(ii) **Reliability Index**: Compute the reliability index β as

$$\sqrt{(x^*-\mu)^TV^{-1}(x^*-\mu)}$$

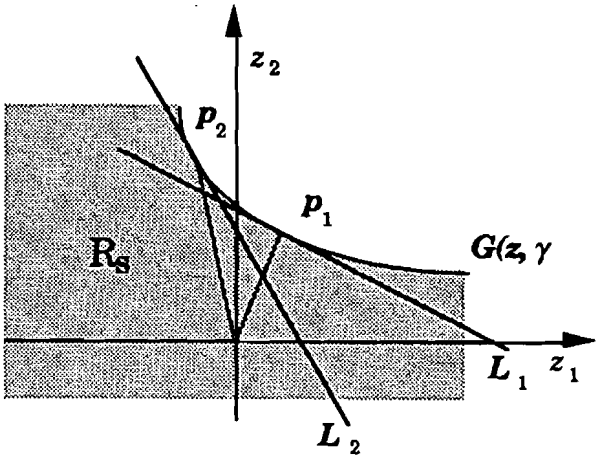


Figure 7. Expansion point for linearization

(iii) **Table Look-Up**: The yield $Y(t)$, i.e. $Pr(F(x,c) \leq 0)$, is approximated by $\Phi(\beta)$, which can be found by looking up in the univariate standard normal distribution table.

Least Cost Tolerance Synthesis

The problem in the context of tolerance analysis is: given the design functions and tolerances, compute the yield. Suppose, instead, the yield is given, how are the tolerances to be assigned to each dimension? Figure 8 illustrates the impact of two different sets of tolerances on the yield. In the figures, the concentric circles represent the contour of equal probability density. It is noted that the tighter set of tolerances (the one in Figure 8(a)) results in a higher yield as compared to the yield with the looser set of tolerances (as in Figure 8(b)).

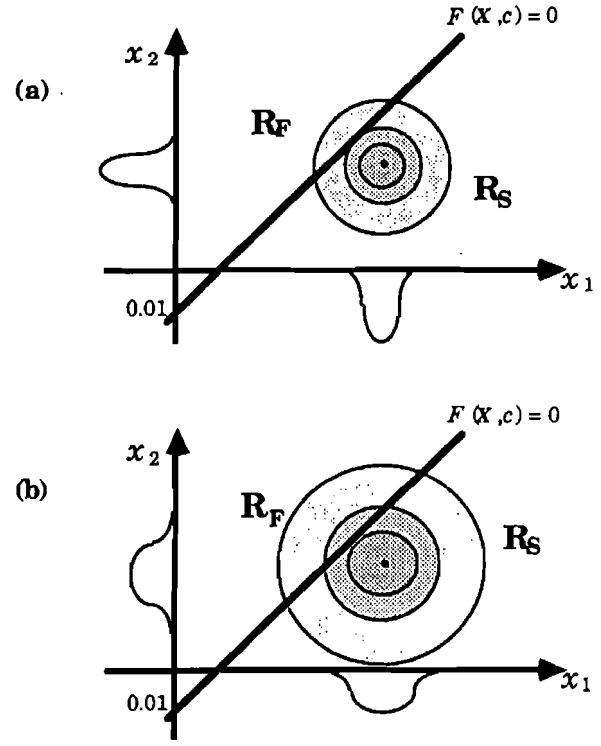


Figure 8. Impact of different tolerances on yield

A higher yield, while desirable, is achieved at the expense of other considerations such as cost and manufacturability. To resolve this conflict, heuristic rules have been proposed [2]: (i) equal tolerances, (ii) tolerances proportional to dimensions, and (iii) tolerances proportional to process deviations. While heuristics are practical, it would be intellectually satisfying to see if assignments of tolerances can be optimized.

Tolerance-Cost Model

As indicated in the introduction, it is generally accepted that there is an inverse relationship between tolerance and manufacturing cost. A number of cost models have been employed to fit manufacturing tolerance-cost sampled data [10], [19], [22], [23], [25], as shown in Table 1.

With such tolerance-cost models for the toleranced dimensions, the total manufacturing cost can be obtained by summing the individual manufacturing costs:

$$C(t) = \sum_{j=1}^{n} C(t_j). \tag{13}$$

Note that model (13) is based on the "throw-aways" strategy, that is, the repair cost for the defect is not considered. However, if the reworking cost is considered, the model becomes:

$$C(t) = \sum_{j=1}^{n} C(t_j) + C_r(p_1,\ldots,p_n)\{1 - Y(t)\} \tag{14}$$

where $C_r(\cdot)$ is the cost function due to reworking and p_j is the probability of reworking the j-th dimension in case that the design function is not satisfied. The difficulty of employing model (14) is in procuring the empirical values for p_j. In this paper, the throw-away cost model (13) is adopted.

Mathematical Formulation

Least-cost tolerance allocation involves the determination of a set of tolerances which minimizes the manufacturing cost and satisfies the performance requirement;

Table 1. Tolerance-Cost Models

Model Name	References	Cost Function*
Sutherland-Roth Model	[23]	$C(t) = a\,t^{-b} + f$
Reciprocal Squared Model	[10], [22]	$C(t) = a / t^2 + f$
Exponential Model	[25]	$C(t) = a \exp\{ - t / b \} + f$
Michael-Siddall Model	[19]	$C(t) = a\,t^{-b} \exp\{ - e\,t \} + f$

* a, b, e constants for variable manufacturing cost
f : constant for fixed manufacturing cost

the decision variables are the tolerances. We formulate this optimization problem with minimizing the manufacturing cost as the objective function and satisfying the functional requirements and non-negativity of tolerances as the constraints:

$$\text{Min } C(t), \text{ subject to } Y(t) \geq 1-\lambda, \text{ and } t \geq 0. \quad (15)$$

where $C(t)$ is the manufacturing cost function in terms of tolerance t, and $(1-\lambda)$ is the minimal satisfactory yield, i.e., the yield given by tolerances t should be greater than or equal to the pre-specified level $1-\lambda$. Formulation (15) indicates that, to perform tolerance synthesis, we should be able to perform tolerance analysis; that means, tolerance synthesis includes tolerance analysis.

With the total cost model as (13) and multiple design functions, formulation (15) is rewritten as

$$\text{Min } \sum_{j=1}^{n} C(t_j) \quad (16)$$

subject to

$$Y_i(t) \geq 1-\lambda_i \qquad \text{for } i = 1, 2,\ldots,m,$$

and

$$t \geq 0.$$

This probabilistic optimization problem can be simplified into a deterministic optimization problem through approximation of the yield by the reliability index.

This conversion into deterministic optimization differs from chance constrained programming due to Charnes and Cooper [3] in that the yield is approximated not at the origin of the standard coordinates but at the point of minimum distance from the origin. Thus, $Y_i(t)$ is approximated by $\Phi(\beta_i)$, as suggested in the section on nonlinear design functions. The constraints of (16) are rewritten as $\Phi(\beta_i) \geq 1-\lambda_i$. Furthermore, by the monotonic property of the function $\Phi(\cdot)$ the constraint $\Phi(\beta_i) \geq 1-\lambda_i$ can be inverted to the constraint $\beta_i \geq \Phi^{-1}(1-\lambda_i)$, and the formulation (16) becomes

$$\text{Min } \sum_{j=1}^{n} C(t_j) \quad (17)$$

subject to

$$\beta_i \geq \Phi^{-1}(1-\lambda_i) \qquad \text{for } i = 1, 2,\ldots,m,$$

and

$$t \geq 0.$$

Note that β_i is a function (with respect to σ_j), whereas $\Phi^{-1}(1-\lambda_i)$ is equal to a constant which can be obtained from the standard normal distribution table.

Recall that an expansion point x_j^* is obtained from solving formulation (11) with a fixed σ_j. However, the "optimal" σ_j obtained from solving formulation (17) is in general different from the initial σ_j. Therefore, in the solution process, as x_j^* changes, the constraints of (17) change as well. Consequently, σ_j may no longer be optimal or even feasible. To incorporate the changes in x_j^*, the Kuhn-Tucker necessary conditions of (11) which denote the local optima of (11) and the feasibility equations, are added to formulation (17) as constraints. Formulation (17) becomes:

$$\text{Min } \sum_{j=1}^{n} C(t_j) \quad (18)$$

subject to

$$\beta_i \geq \Phi^{-1}(1-\lambda_i), \qquad \text{for } i = 1, 2,\ldots,m \quad (18a)$$

$$\frac{\partial \beta_i}{\partial x} + u_i \frac{\partial F_i(x^*,c)}{\partial x} = 0, \quad \text{for } i = 1, 2,\ldots,m \quad (18b)$$

$$F_i(x^*,c) = 0, \quad (18c)$$

and

$$t \geq 0.$$

Algorithmic Analysis

In Nonlinear Programming, no existing algorithm guarantees a globally optimal solution unless the objective function and the constraints are of certain forms. The existence check for the special forms is performed as a sufficient condition for global optimality [27, pp. 43-44]: that the objective function is convex and the constraint functions are quasi-concave. This ensures that the locally optimal solution implied by the Kuhn-Tucker necessary conditions is also a globally optimal solution. In our model, however, no assumptions have been made on the nature of constraints.

Geometrically, the constraints of formulation (18) comprise a feasible region. A point in the feasible region corresponds to a feasible assignment of tolerances. Then, the total cost of a point with a particular combination of tolerances is obtained by evaluating the objective function, which is the cost function, at that point.

Feasible direction methods are commonly used for solving NLP problems. A general procedure of feasible direction methods is described. Suppose $\sigma^{(k)}$ is a point in the feasible region. A direction $d^{(k)}$ is identified such that, for a sufficiently small $\lambda > 0$, the following two properties are true: (i) $\sigma^{(k+1)} = \sigma^{(k)} + \lambda d^{(k)}$ is feasible, and (ii) the objective value at $\sigma^{(k+1)}$ is better than the objective value at $\sigma^{(k)}$. In each iteration of the feasible direction method, having determined a feasible direction, a one-dimensional optimization problem is solved

to maximize the improvement of the objective value.

In particular, the sensitivity of yield is employed to determine the search direction in the optimization process for tolerance assignment. The sensitivity of yield provides information on which tolerances are critical. With respect to a tolerance t_j, it is defined as

$$s(\beta,t_j) = \frac{\partial Y(t)}{\partial t_j}. \tag{19}$$

Now that $Y(t)$ is approximated by $\Phi(\beta)$ (by step (iii) in tolerance analysis for nonlinear design functions), (19) can be approximated by $\frac{\partial \Phi(\beta)}{\partial t_j}$. (For a linear design function, $\frac{\partial \Phi(\beta)}{\partial t_j}$ is the exact yield sensitivity.)

Expanding $s(\beta,t_j)$ by the chain rule reveals the search direction:

$$s(\beta,t_j) = \frac{\partial \Phi(\beta)}{\partial \beta}\frac{\partial \beta}{\partial \sigma_j}\frac{\partial \sigma_j}{\partial t_j}. \tag{20}$$

In (20), $\frac{\partial \beta}{\partial \sigma_j}$ is obtained by

$$\frac{\partial \beta}{\partial \sigma_j} = \frac{\partial \beta}{\partial \beta^2} * \frac{\partial \beta^2}{\partial \sigma_j} = \frac{1}{2\beta} * \frac{\partial \beta^2}{\partial \sigma_j}. \tag{21}$$

Recall that,

$$\beta^2 = (x^*-\mu)^T V^{-1}(x^*-\mu)$$

$$= (x^*-\mu)^T (DRD)^{-1}(x^*-\mu) = g^T R^{-1} g,$$

where D is a diagonal matrix whose the k-th diagonal element is $\frac{1}{\sigma_k}$, R is a $n \times n$ correlation matrix, and $g = \left(\frac{x_1^*-\mu_1}{\sigma_1},\ldots,\frac{x_n^*-\mu_n}{\sigma_n}\right)^T$. Therefore,

$$\frac{\partial \beta^2}{\partial \sigma_j} = \frac{\partial}{\partial \sigma_j} g^T R^{-1} g = \frac{\partial g^T}{\partial \sigma_j} R^{-1} g + g^T R^{-1}\frac{\partial g}{\partial \sigma_j}$$

$$= 2\frac{\partial g^T}{\partial \sigma_j} R^{-1} g = -2(0,\ldots,0,\frac{x_j^*-\mu_j}{\sigma_j^2},0,\ldots,0) R^{-1} g$$

$$= -2\frac{x_j^*-\mu_j}{\sigma_j^2}\left(\sum_{i=1}^{n} \hat{\varrho}_{ji}\frac{x_i^*-\mu_i}{\sigma_i}\right), \tag{22}$$

where $\hat{\varrho}_{ji}$ is the (i,j)-th element of matrix R^{-1}. Substituting (22) into (21) results in

$$\frac{\partial \beta}{\partial \sigma_j} = -\frac{x_j^*-\mu_j}{\beta\sigma_j^2}\left(\sum_{i=1}^{n} \hat{\varrho}_{ji}\frac{x_i^*-\mu_i}{\sigma_i}\right). \tag{23}$$

Finally, substituting (23) into (20), we get

$$s(\beta,t_j) = \phi(\beta) * \left\{-\frac{x_j^*-\mu_j}{\beta\sigma_j^2}\left(\sum_{i=1}^{n} \hat{\varrho}_{ji}\frac{x_i^*-\mu_i}{\sigma_i}\right)\right\} * \frac{1}{6}, \tag{24}$$

where $\hat{\varrho}_{ji}$ is the (i,j) element of the inverse matrix of the correlation matrix,

x_j^* is the j-th coordinate of the expansion point, and

$\phi(\cdot)$ is the PDF of the standard normal distribution, i.e.,

$$\phi(\beta) = \frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{\beta^2}{2}\right\}.$$

Under the assumption of independence among the dimensions, i.e., $\hat{\varrho}_{ji}=1$ if $j=i$, and $\hat{\varrho}_{ji}=0$ otherwise, the sensitivity of yield is rewritten as

$$s(\beta,t_j) = \phi(\beta) * \left\{-\frac{(x_j^*-\mu_j)^2}{\beta\sigma_j^3}\right\} * \frac{1}{6}. \tag{25}$$

Equation (25) shows that the $s(\beta,t_j)$ are always nonpositive. This means that there is a nonincreasing *monotonicity* between yield and tolerance. This monotonic property demonstrates the trade-off between tolerance and performance: the performance, which is implied by yield, increases as the tolerances are tightened.

Lagrangean Multiplier

The impact of modifying yield constraints on the manufacturing cost, i.e., the objective value, is considered in this section. This is performed as a post-optimality analysis.

The partial derivative $\frac{\partial C(\sigma_1,\ldots,\sigma_n)}{\partial \lambda_i}$ at the optimal solution $(\sigma_1^*,\ldots,\sigma_n^*)^T$ is decomposed into two parts using the chain rule:

$$\frac{\partial C(t)}{\partial \lambda_i} = \frac{\partial C(t)}{\partial q_i}\frac{\partial q_i}{\partial \lambda_i}. \tag{26}$$

The second part of this decomposition turns out to be:

$$\frac{\partial q_i}{\partial \lambda_i} = -\frac{\sqrt{2\pi}}{\exp\left\{-\frac{q_i^2}{2}\right\}}$$

since $\lambda_i = 1-\Phi(q_i)$. The first part of (26), $\frac{\partial C(t)}{\partial q_i}$, called the "Lagrangean multiplier" in NLP has the following characteristic: if the constraint is inactive at the optimal solution, then the corresponding Lagrangean multiplier should be zero. This means that for an inactive constraint at optimum the corresponding q_i can be modified slightly without incurring additional manufacturing cost.

In case that a constraint is active at optimum, the corresponding Lagrangean multiplier should be greater than zero. It is computed as:

$$\frac{\partial C(t)}{\partial q_i} = \sum_{j=1}^{n}\frac{\partial C(t)}{\partial \sigma_j}\frac{\partial \sigma_j}{\partial q_i} = \sum_{j=1}^{n}\frac{\partial C(t)}{\partial \sigma_j}\frac{\partial \sigma_j}{\partial \beta_i} \tag{27}$$

since the i-th constraint is assumed to be active, i.e. $q_i = \beta_i$. From equation (25), $\frac{\partial \beta_i}{\partial \sigma_j} = -\frac{(x_{ij}^*-\mu_j)^2}{\beta_i\sigma_j^3}$, where x_{ij}^* is the j-th expansion point for β_i, (27) becomes

$$\frac{\partial C(t)}{\partial q_i} = \sum_{j=1}^{n}\frac{\partial C(t)}{\partial \sigma_j}\left\{-\frac{\beta_i\sigma_j^3}{(x_{ij}^*-\mu_j)^2}\right\}. \tag{28}$$

Based on the inverse-squared model, i.e., $C(t) =$

$\sum_{k=1}^{n}\left\{\frac{a_k}{(6\sigma_k)^2}+f_k\right\}$, equation (28) becomes

$$\frac{\partial C(t)}{\partial q_t}=\sum_{j=1}^{n}\frac{-a_j}{18\sigma_j^3}\left\{-\frac{\beta_t\sigma_j^3}{(x_{ij}^*-\mu_j)^2}\right\}=\sum_{j=1}^{n}\frac{a_j\beta_t}{18(x_{ij}^*-\mu_j)^2}. \quad (29)$$

Implementation

An assembly of two parts with twelve dimensions is shown in Figure 9(a). Six design functions, linear and non-linear, are given in Figure 9(b). Design functions $F_1(X)$ and $F_2(X)$ represent the vertical and the horizontal clearance conditions of the two parts. Design functions $F_3(X)$ and $F_4(X)$ post the restriction on the difference between angles θ_1 and θ_2 to ensure feasibility of assembly. Design functions $F_5(X)$ and $F_6(X)$ give the requirements for the size difference of those two parts.

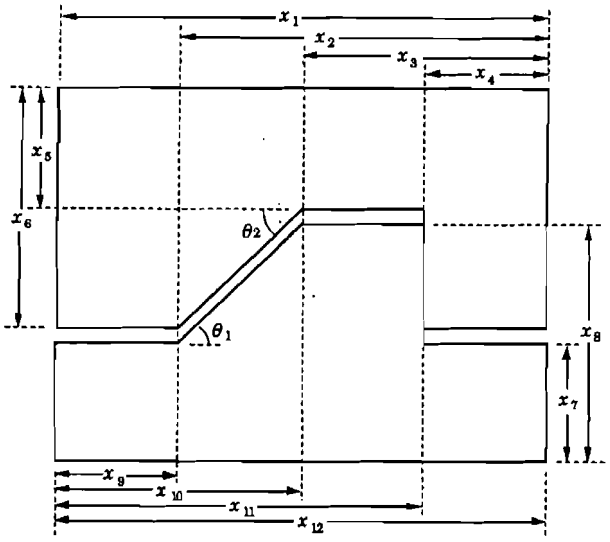
The nominal dimensions are given as X^T = (50.0, 40.00125, 20.05, 9.9985, 9.9985, 30.0, 10.0, 30.0, 10.05, 30.0, 40.0, 50.0). The cost function with respect to the tolerance of each dimension is

$$C(\sigma_i)=\frac{a_i\times 10^{-3}}{(6\sigma_i)^{b_i}}.$$

The coefficients of the respective cost functions are: $a_1 = 0.2$, $a_2 = 1.0$, $a_3 = a_4 = 0.015$, $a_5 = 0.008$, $a_6 = 0.009$, $a_7 = 0.008$, $a_8 = 0.006$, $a_9 = 1.0$, $a_{10} = 0.01$, $a_{11} = 0.015$, and $a_{12} = 0.2$; and $b_1 = \ldots = b_{12} = 2.0$.

The algorithm given in this paper is implemented in PASCAL and runs on the IBM PC. For the algorithms to be practical in an interactive design environment, attention is given to speed — in particular, the computation for the Jacobian matrix. The initial tolerances are assigned in accordance to ANSI-Y14.5M for loose fit.

It took 5.25 CPU seconds to obtain the optimal solution, and the resulting tolerances are shown in Table 2; each stack-up condition is satisfied with 95% confidence if the dimensions x_1 to x_{12} are manufactured within the tolerances obtained. To cross check the algorithm, dimensions x_2 and x_9 are intentionally assigned with higher manufacturing costs, with $a_2 = a_9 = 1.0$. The tolerances computed are consistent with the manufacturing costs as looser tolerances are assigned to dimensions x_2 and x_9.



(a)

$$F_1(X) = (x_6-x_5) - (x_8-x_7)$$

$$F_2(X) = (x_3-x_4) - (x_{11}-x_{10})$$

$$F_3(X) = (x_8-x_7)(x_2-x_3)-(x_6-x_5)(x_{10}-x_9) + \tan(\pi/180) * \{(x_{10}-x_9)(x_2-x_3) + (x_8-x_7)(x_6-x_5)\}$$

$$F_4(X) = (x_6-x_5)(x_{10}-x_9) - (x_8-x_7)(x_2-x_3) + \tan(\pi/180) * \{(x_{10}-x_9)(x_2-x_3) + (x_8-x_7)(x_6-x_5)\}$$

$$F_5(X) = -x_1 + x_{12} + 0.01$$

$$F_6(X) = x_1 - x_{12} + 0.01$$

(b)

Figure 9. Examples of linear and nonlinear design functions

Concluding Remarks

A procedure for tolerance synthesis has been presented. The algorithm for probabilistic tolerancing has been developed based on the notion of feasible directions. An analytic result of yield sensitivity is used to speed up the computation of the Jacobian matrix inside the optimization loop. Compared to a previously established landmark [14], this new algorithm produces a solution with more than 10 times reduction in CPU

Table 2. Result of the Example

Dimensions	Cost Function Coefficients		Tolerances
	a_i	b_i	
X_1	0.2	2.0	0.0093
X_2	1.0	2.0	0.6427
X_3	0.015	2.0	0.0337
X_4	0.015	2.0	0.0337
X_5	0.008	2.0	0.0010
X_6	0.009	2.0	0.0011
X_7	0.008	2.0	0.0010
X_8	0.006	2.0	0.0008
X_9	1.0	2.0	0.6433
X_{10}	0.01	2.0	0.0337
X_{11}	0.015	2.0	0.0337
X_{12}	0.2	2.0	0.0093

time. Also, the post-optimality analysis of the algorithm enables a designer to verify design intention.

Acknowledgement

The authors are grateful to the reviewers who provided insightful comments that substantially improved the presentation of this paper.

REFERENCES

[1] *Dimensioning and Tolerancing*, ANSI Standard Y14.5M-1982, American Society of Mechanical Engineering, New York (1982).

[2] Bjorke, O., *Computer Aided Tolerancing*, Tapir Press, Norway (1978).

[3] Charnes, A., and Cooper, W. W., "Deterministic Equivalents for Optimizing and Satisficing under Chance Constraints," *Ops. Res.*, Vol. 11, 18-39 (1963).

[4] Dong, Z., and Soom, A., "Automatic Tolerance Analysis from a CAD Database," ASME Paper No. 86-DET-36 (1986).

[5] Evans, D. H., "Statistical Tolerancing: The State of the Art, Part II, Methods for Estimating Methods," *Journal of Quality Technology*, Vol. 7, No. 1, 1-12 (January 1975).

[6] Graybill, F. A., *An Introduction to Linear Statistical Model—Volume 1*, McGraw-Hill Book Co., Inc., New York (1961).

[7] Greenwood, W. H., and Chase, K. H., "A New Tolerance Analysis Method for Designers and Manufacturers," *Trans. of ASME, Journal of Engineering for Industry*, Vol. 109, 112-116 (May 1987).

[8] Grossman, D. D., "Monte-Carlo Simulation of Tolerancing in Discrete Parts Manufacturing and Assembly," Report No. STAN-CS-76-555, Computer Science Department, Stanford University (May 1976)

[9] Hasofer, A. M., and Lind, N. C., "Exact and Invariant Second-Moment Code Format," *Journal of the Engineering Mechanics Division*, ASCE, Vol. 100, 111-121 (1974).

[10] Hiller, M. J., "A Systematic Approach to the Cost Optimization of Tolerances in Complex Assemblies," *Bulletin of Mechanical Engineering Education*, Vol. 5, 157-161 (1966).

[11] Huang, J., and Ostwald, P. F., "A Method for Optimal Tolerance Selection," ASME Paper No. 76-WA/DE-23.

[12] Latta, L. W., "The Assignment of Least Cost Tolerances," *Product Engineering*, Vol. 34, 111 (1963).

[13] Lee, W.-J., "Tolerancing—Computation on Geometric Uncertainties," Ph.D. Dissertation, Dept. of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan (1988).

[14] Lee, W.-J., and Woo, T. C., "Tolerances: Their Analysis and Synthesis," *Trans. of ASME, Journal of Engineering for Industry*, Vol. 112, 113-121 (May 1990).

[15] Lee, W.-J., and Woo, T. C., "Optimum Selection of Discrete Tolerances," *Trans. of ASME, Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 111, No. 2, 243-251 (June 1989).

[16] Luenberger, D. G., *Introduction to Linear and Nonlinear Programming*, Addison-Wesley (1965).

[17] Madsen, H. O., Krenk, S., and Lind, N. C., *Methods of Structural Safety*, Prentice-Hall, Inc., Englewood Cliffs, NJ 07632 (1986).

[18] Mansoor, E. M., "The Application of Probability to Tolerances Used in Engineering Designs," *Proc. Inst. Mech. Eng.*, Vol. 178, No. 1, 29-51 (1963).

[19] Michael, W., and Siddall, J. N., "The Optimal Tolerance Assignment with Less Than Full Acceptance," *Trans. of ASME, Journal of Mechanical Design*, Vol. 104, 855-860 (October 1982).

[20] Parkinson, D. B., "The Application of Reliability Methods to Tolerancing" *Trans. of ASME, Journal of Mechanical Design*, Vol. 104, 612-618 (July 1982).

[21] Peat, A. P., *Cost Reduction Charts for Designers and Production Engineers*, The Machining Publishing Co. Ltd., Brighton (1968).

[22] Spotts, M. F., "Allocation of Tolerances to Minimize Cost of Assembly," *Trans. of ASME, Journal of Engineering for Industry*, 762-764 (August 1973).

[23] Sutherland, G. H., and Roth, B., "Mechanism Design: Accounting for Manufacturing Tolerances and Costs in Function Generating Problems," *Trans. of ASME, Journal of Engineering for Industry*, 283-286 (1975).

[24] Turner, J. U., and Wozny, M. J., "Tolerances Analysis in a Solid Modeling Environment," *Computers in Engineering*, Vol. 2, ASME, 169-175 (1987).

[25] Wilde, D., and Prentice, E., "Minimum Exponential Cost Allocation of Sure-Fit Tolerances," ASME Paper No. 75-DET-93.

[26] Wu, Z., Elmaraghy, W. H., and Elmaraghy, H. A., "Evaluation of Cost-Tolerance Algorithms for Design Tolerance Analysis and Synthesis," *Manufacturing Review*, Vol. 1, No. 3, 168-179 (October 1988).

[27] Zangwill, W. I., *Nonlinear Programming—A Unified Approach*, Prentice-Hall Inc., Englewood Cliffs, New Jersey (1969).

Appendix

Transformation to the Standard Coordinates

The transformation into the standard coordinates is accomplished by taking the following four steps [6]:

Step 1: Translate to the origin by $x^o = x - \mu$;

Step 2: Diagonalize the given covariance matrix V through the operation $P^T V P = V_x$, where P is the orthogonal matrix;

Step 3: Orthogonally transform by $z^o = P^T x^o$;

Step 4: Standardize by $z = D^{-1} z^o$, where $DD^T = V_x$.

Woo-Jong Lee is with the Technical Center in Daewoo Motor Co., Korea. His principal research interests are in computer-aided design and manufacturing, solid modeling, and methods for analyzing and synthesizing tolerances in design and manufacturing. He received a B.S. degree in industrial engineering from Seoul National University in 1979, an M.S. degree in industrial engineering from Korea Advanced Institute of Science and Technology in 1981, and a Ph.D. in industrial and operations engineering from University of Michigan in 1988.

Tony C. Woo is Professor of Industrial and Operations Engineering at the University of Michigan, Ann Arbor, where he teaches courses in CAD/CAM. His research interest is in the development of geometric algorithms for design, manufacturing, and robotics.

Shuo-Yan Chou is a research assistant in the Department of Industrial and Operations Engineering at the University of Michigan, Ann Arbor. His research interest is in computational geometry and its applications in manufacturing, tolerancing, and solid modelling. He received a B.B.A. degree in industrial management science from Chen-Kung University in 1983, an M.S. degree in industrial engineering from University of Michigan in 1987, and he is currently working toward his Ph.D. degree in the Department of Industrial and Operations Engineering at University of Michigan.

Received June 1989; revised May 1990 and September 1990. Handled by the Editor-in-Chief.

ANEXO 3. PROGRAMAS DE MATLAB PARA EL EJEMPLO 1

```
clear all

% Se quiere asignar tolerancias a 8 variables para cumplir las
% restricciones de 4 variables Y. Etapa1
%-------------------------------------------------------------------------
options=optimset('LargeScale','off');
puntoinicial=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];
bimin=[0.001;0.001;0.001;0.001;0.001;0.001;0.001;0.001];

% Para realizar la optimización debemos definir un punto inicial en el que
% comenzará la optimización y un valor minimo de aumento en cada iteración

[x,FVAL,EXITFLAG,output]=fmincon(@funcionobjetivo1,puntoinicial,[],[],[],[],...
    bimin,[],@definicion_de_restricciones,options)

%Se define la variable a optimizar, en este caso x, la función de
%optimización que será la que hemos definido con
%anterioridad en funcionobjetivo, el punto inicial, el valor mínimo y las
%restricciones que seran las definidas con anterioridad en
%definicion_de_restricciones

% Ejecuta el algoritmo de optimización hasta encontar la solución óptima



%↙
----------------------------------------------------------------------------↙
---

%aquí se lee la solución para ver los valores obtenidos, se calcularan los
%valores de las matrices de covarianzas de x e y óptimos

bi=x;  % valores óptimos de bi

p=0.9973;
alpha=1-p;

load CovIniciales % Se leen las matrices de covarianzas y se unen en una
                  % sola de 8x8
covarianzasx=zeros(8,8);
covarianzasx(1:5,1:5)=MatCov1;
covarianzasx(6:8,6:8)=MatCov2;

a=length(covarianzasx); % tamaño de la matriz de covarianzas de x

A=[0 0 0 1 1 0 0 0;-1 1 0 0 0 0 1 -1;0 1 -1 0 0 -1 1 0;0 0 -1 1 0 -1 0 0];
% Matriz de relaciones defindа por las especificaciones del montaje
% mecánico(Y=f(X)

tolminx=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01]; % es el valor de
% tolerancias mínimas para x

toleranciasy=[0.13;0.0924;0.0746;0.0924];
% toleranciasy =( Valores de Y [(y1<=127.13 , y2>=0.0076 , y3>= 0.0254
% y4>=0.0076)]- valores nominales de Y [127, 0.1 , 0.1, 0.1])
```

```
%-------------------------------------------------------------------------

[C, autov]=eig(covarianzasx); % Obtiene los autovectores y los
% autovalores de la matriz d covarianzas de x

D=diag(autov);% Con los autovalores obtenidos en el comando anterior
% generamos una matriz D, diagonal de autovalores de la matriz
% de covarianzas

newD=bi.*D; % Obtenemos una nueva matriz D modificando los autovalores
%multiplicandola por los factores bi

newcovarianzasx=C*diag(newD)*C'; % con la nueva matriz D obtenemos una
% nueva matriz de covarianzas de x

restriccion1=det(newcovarianzasx);

proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)); % calculamos la proyección
% de la matriz de varianzas de x que acabamos de obtener

Suma_ti=sum(proyecx);

restriccion2=-(proyecx-tolminx); % Realizamos la comparación entre la
                                 % proyección y las tolerancias de x

newcovarianzasy=A*newcovarianzasx*A'; % obtenemos la nueva matriz de
% covarianzas de y a traves de la nueva matriz de covarianzas de x que
%hemos obtenido en el paso anterior

b=length(newcovarianzasy); % tamaño de la matriz de varianzas de y

proyecy=sqrt(chi2inv(p,b)*diag(newcovarianzasy)); % calculamos la
% proyección de la diagonal de la matriz de covarianzas de y

restriccion3=proyecy-toleranciasy; % calculamos la diferencia entre la
% proyección de la diagonal de la matriz de covarianzas de y
% las tolerancias de y

%-------------------------------------------------------------------------

save Etapa1 newcovarianzasx bi % Grabamos la solución,
%  en este caso será las matrices de covarianzas de x  óptimas para esta
% etapa asi como los valores óptimos de bi que se han utilizado.
```

```
% Este programa generara datos aleatorios de
% distribuciones normales multivariantes
% En este ejmplo el enamblaje consta de dos piezas,
% por lo que se generarán dos matrices de covarianzas:
% una de 5x5 y otra de 2x2


dim_pieza_sup=5; % Adaptamos la matriz al numero de variables correladas
                 % que definen la pieza inferior, 5 variables
dim_pieza_inf=3; % Adaptamos la matriz al numero de variables correladas
                 % que definen la pieza superior, 3 variables

m=1000; %Definimos un numero elevado de iteraciones

b1=[0.1 0.1 0.1 0.1 0.1]; % Valores para el comienzo de la iteración.
b2=[0.1 0.1 0.1]; % Valores para el comienzo de la iteración.

x1=zeros(m,dim_pieza_sup); % Generamos dos matrices de ceros,
                           % cada una de ellas de m filas
x2=zeros(m,dim_pieza_inf); % y columnas acordes con las dimensiones
                           % que se han definido anteriormente
%-------------------------------------------------------

%-------------------------------------------------------

x1(:,1)=randn(m,1)*b1(1);

for i=2:dim_pieza_sup
    x1(:,i)=rand*x1(:,1)+randn(m,1)*b1(i)*0.5;
end
Matcorr1=corrcoef(x1);
MatCov1=cov(x1);
[autovectores1,au21]=eig(MatCov1);
Sigma1=sqrt(diag(MatCov1));
Propau1=sort(diag(au21)./sum(diag(au21))*100,'descend');

%-------------------------------------------------------
x2(:,1)=randn(m,1)*b2(1);

for i=2:dim_pieza_inf
    x2(:,i)=rand*x2(:,1)+randn(m,1)*b2(i)*0.3;
end
Matcorr2=corrcoef(x2);
MatCov2=cov(x2);
[autovectores2,au22]=eig(MatCov2);
Sigma2=sqrt(diag(MatCov2));
Propau2=sort(diag(au22)./sum(diag(au22))*100,'descend');

% Estos dos programas generan las dos matrices de covarianzas iniciales de
% x que se utilizaran el los posteriores programas.

%-------------------------------------------------------
```

```
%-------------------------------------------------------

save CovIniciales MatCov1 MatCov2 % Por último guardamos las dos
                                  % matrices que hemos generado
                                  % para utilizarlas posteriormente

%-------------------------------------------------------
```

```
function f=funcionobjetivo1(x)

%En este programa se define la funcion objetivo

bi=x;  % valores por los que iremos multiplicando los autovectores para
       % modificarlos hasta conseguir los valores óptimos que cumplan la
       % función objetivo que estamos definiendo

p=0.9973; % definimos el porcentaje de piezas defectuosas que consideramos
          % aceptable

load CovIniciales % Cargamos el programa que hemos realizado antes para
                  % generar la matriz de covarianzas


covarianzasx=zeros(8,8); % la dimensión de la matriz de covarianzas sera la
                         % correspondiente al numero de variables X totales
                         % que tiene el ensamblaje

covarianzasx(1:5,1:5)=MatCov1; % dependiendo del numero de variables corre-
covarianzasx(6:8,6:8)=MatCov2; % ladas que tengamos obtendremos una
                               % matriz de covarianzas total,utilizando las
                               % dos matrices que se han obtenido en el
                               % programa anterior para generar la matriz
                               % de covarienzas

%-------------------------------------------------------------------------
a=length(covarianzasx);

[C, autov]=eig(covarianzasx); % Obtiene los autovectores y los
% autovalores de la matriz d covarianzas de x

D=diag(autov);% Con los autovalores obtenidos en el comando anterior
% generamos una matriz D, diagonal de autovalores de la matriz
% de covarianzas

newD=bi.*D; % Obtenemos una nueva matriz D modificando los autovalores
%multiplicandola por los factores bi

newcovarianzasx=C*diag(newD)*C'; % con la nueva matriz D obtenemos una
% nueva matriz de covarianzas de x

proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)); % calculamos la
% proyección de la diagonal de la matriz de covarianzas de x que
% estos son los valores ti

f=-sum(proyecx); % la función objetivo sera buscar la minima diferencia
% entre las proyecciones de la matriz de covarianzas de x y las tolerancias
% de x

%-------------------------------------------------------------------------

save NewX newcovarianzasx
```

```
function [restricciones,nleq]=definicion_de_restricciones(x)

%Aquí se definen las restricciones del problema

load Newx  % esta leyendo newcovarianzasx

bi=x;
p=0.9973;

a=length(newcovarianzasx); % tamaño de la matriz de covarianzas de x

A=[0 0 0 1 1 0 0 0;-1 1 0 0 0 0 1 -1;0 1 -1 0 0 -1 1 0;0 0 -1 1 0 -1 0 0];
% Matrz de relaciones defindа por las especificaciones del montaje mecánico
% (Y=f(X))

tolminx=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];
% tolerancias mínimas para x

toleranciasy=[0.13;0.0924;0.0746;0.0924];
% toleranciasy =( Valores de Y [(y1<=127.13 , y2>=0.0076 , y3>= 0.0254
% y4>=0.0076)]- valores nominales de Y [127, 0.1 , 0.1, 0.1])

%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% restricción 1
restriccion1=det(newcovarianzasx); % calculamos el determinante de la nueva
% matriz de covarianzas de x que hemos obtenido

%--------------------------------------------------------------------------



%--------------------------------------------------------------------------
% restricción 2
proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)); % calculamos la
% proyección de la matriz de varianzas de x que acabamos de obtener

restriccion2=-(proyecx-tolminx); % calculamos la diferencia entre la
% proyección de la matriz de varianzas de x y las tolerancias minimas de x
% que hemos definido

%--------------------------------------------------------------------------
% restricción 3
newcovarianzasy=A*newcovarianzasx*A'; %obtenemos la nueva matriz de
% covarianzas de y a traves de la nueva matriz de covarianzas de x que
%hemos obtenido en el paso anterior

b=length(newcovarianzasy); % tamaño de la matriz de covarianzas de y
proyecy=sqrt(chi2inv(p,b)*diag(newcovarianzasy));% calculamos la proyección
% de la matriz de varianzas de y

restriccion3=proyecy-toleranciasy; % calculamos la diferencia entre la
% proyección de la matriz de varianzas de y las tolerancias de y

%--------------------------------------------------------------------------
```

```
%--------------------------------------------------------------------------
restricciones=[-restriccion1;restriccion2;restriccion3];
nleq=[];
% Las restricciones del problema serán: que no se superen las tolerancias
% de y ( utilizamos diferencias d y), que las tolerencias de x sean mayores
% que las tolerancias minima de x que hemos definido (utilizamos
% diferenciasx) y que los determinantes de las matrices de covarianzas de x
% e y sean >0 (utilizamos restriccion1 y restriccion2)

%--------------------------------------------------------------------------
```

```
clear all

% En este prgramas se realiza la Etapa 2.
% necesito cargar la solución obtenida en la Etapa 1

load Etapa1  %lee newcovarianzasc bi
% las variables que lee son los óptimos: newcovarianzasy newcovarianzasx bi

A=[0 0 0 1 1 0 0 0;-1 1 0 0 0 0 1 -1;0 1 -1 0 0 -1 1 0;0 0 -1 1 0 -1 0 0];
% Matrz de relaciones defindа por las especificaciones del montaje mecánico
% (Y=f(X))

toleranciasy=[0.13;0.0924;0.0746;0.0924];
% toleranciasy =( Valores de Y [(y1<=127.13 , y2>=0.0076 , y3>= 0.0254
% y4>=0.0076)]- valores nominales de Y [127, 0.1 , 0.1, 0.1])

bi0=bi; % tomamos como valores iniciales los valores de bi y las matrices
covx=newcovarianzasx; % de covarianzas de x e y obtenidos en la etapa 1

mediax=[25.4 50.8 76.2 101.6 25.4 25.3 50.8 76.1]'; % Valores nominales
                                                    % de x en columna

mediay=A*mediax; % (127 0.1 0.1 0.1)% Valores nominales de y

a=length(mediax); % número de variables x

b=length(mediay) ;% número de variables y

h=mediay-toleranciasy; % h=(0.13;0.0924;0.0746;0.0924

%Las especificaciones de Y son: Y1<=127.13, Y2>=0.01, Y3>=0.025, Y4>=0.01
LES=[127.13 1000 1000 1000]; % En el caso de tener solo límite inferior
% ponemos un valor elevado para el límite superior (caso de y1,y2,y3)

LEI=zeros(1,4); % en el caso de tener solo límite inferior le damos valor
LEI(2:4)=h(2:4);%0 al límte inferior (caso de y1)

%-------------------------------------------------------------------------

epsilom=0.00005;
p=0.9973;
alpha=0.0027;
FMAX=30.1111;
FMIN=0.000001;

% Empieza a iterar para buscar la solución que implique un alfa de 0.0027.
% Genera valores aleatorios de X según su Cov y media, los trunca en 99.73%
% y con esos valores truncados calcula los valores de Y según Y=F(X).
% Estima la proporción de Y fuera de tolerancias por MonteCarlo.

%-------------------------------------------------------------------------


bi=1;
newcovarianzasx=bi*covx;
```

```
T=zeros(10000,8);

for j=1:10
    n=10^5;
    proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx));

    simX=mvnrnd(mediax,newcovarianzasx,n); %genera valores aleatorios con
    %distribución de media de x y matriz de covarianzas de x
    Tolsupx=mediax+proyecx;
    Tolinfx=mediax-proyecx;

    clear T
    T=(simX>=(ones(n,1)*Tolinfx'))&(simX<=(ones(n,1)*Tolsupx'));
    Tn=double(T);
    Tnp=prod(Tn')';   % Buscamos los valores que estan dentro de las
                      % tolerancias de x obtenidas como proyección
    indiX=find(Tnp==1);
    simXtrun=simX(indiX,:); % Esta matriz contiene las observaciones x,
                            % ya comprobadas
    ntrun=length(simXtrun);

    simYtrun=zeros(ntrun,b);
    simYtrun=simXtrun*A'; % Calcula las y para todos los x seleccionados,
                          % le la función de propdefecto
    [Vpdefec(j)]=Propdefecto(simYtrun,LEI,LES);
end

%-------------------------------------------------------------------------


propdefec1=mean(Vpdefec);
propdefec=propdefec1;


if propdefec>=alpha
    fmax=bi;
    pmax=propdefec;
    fmin=FMIN;
    pmin=0;
else
    fmin=bi;
    pmin=propdefec;
    fmax=FMAX;
    pmax=1;
end
% Hasta aqui calcula para matrices finales obtenidas en la última iteración
% de la Etapa 1 el porcentaje de defectuosos p

%-------------------------------------------------------------------------

% A partir de aqui se empieza a iterar hasta que el porcentaje de
% defectuosas sea alpha

difer=abs(propdefec-alpha);
conta=0;
```

```
while difer>epsilom
    conta=conta+1;
    if conta<2
        bi=fmin+(fmax-fmin)/(pmax-pmin)*(alpha-pmin);
    else
        bi=(fmin+fmax)/2;
    end

    newcovarianzasx=bi*covx;
    for j=1:10
        n=10^5;
        newvarianzasx=diag(newcovarianzasx);
        proyecx=sqrt(chi2inv(p,a)*newvarianzasx);

        simX=mvnrnd(mediax,newcovarianzasx,n);
        Tolsupx=mediax+proyecx;
        Tolinfx=mediax-proyecx;
        clear T
        T=(simX>=ones(n,1)*Tolinfx')&(simX<=ones(n,1)*Tolsupx');
        Tn=double(T);
        Tnp=prod(Tn')';
        indiX=find(Tnp==1);
        simXtrun=simX(indiX,:);
        ntrun=length(simXtrun);
        simYtrun=zeros(ntrun,b);
        simYtrun=simXtrun*A';
        [Vpdefec(j)]=Propdefecto(simYtrun,LEI,LES);
    end
    propdefec=mean(Vpdefec);


    if propdefec>=alpha
        fmax=bi;
        pmax=propdefec;
    else
        fmin=bi;
        pmin=propdefec;
    end

    difer=abs(propdefec-alpha);
    if (conta>5)&(abs(bi-FMAX)<0.00005);break;end
    if (conta>5)&(abs(bi-FMIN)<0.00005);break;end
    if sqrt(bi)<0.009; bi=0.009^2;break;end
    if sqrt(bi)>25;bi=25.555^2;break;end
    if conta>100;conta,break;end

end
factorb=bi; % este es el factor de la Etapa 2
nonconforming=propdefec;

factorfinalCPx=bi*bi0;  %este son los factores de Etapa2*Etapa1

%-------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------
newcovarianzasx=covx*bi;  %con esta matriz calculo los límites de tolerancia de X
proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx));

LTSX=mediax+proyecx;
LTIX=mediax-proyecx;
```

```
function [pro_def]=Propdefecto(RE,LEI,LES)

fi=size(LEI,1);
if fi==1
    LEI=LEI';
end

fi=size(LES,1);
if fi==1
    LES=LES';
end


[n,m]=size(RE);
MLTI=ones(n,1)*LEI';
MLTS=ones(n,1)*LES';
esdef=(RE>MLTS)|(RE<MLTI);
if m>1
    Esdef1=(sum(esdef')>0);
else
    Esdef1=esdef;
end
pro_def=mean(Esdef1);
```

ANEXO 4. PROGRAMAS DE MATLAB PARA EL EJEMPLO 2

```
clear all

% Se quiere asignar tolerancias a 3 variables para cumplir las
% restricciones de 2 variables Y. Etapa1
%-------------------------------------------------------------------------
options=optimset('LargeScale','off');
puntoinicial=[0.01;0.01;0.01];
bimin=[0.001;0.001;0.001];

% Para realizar la optimización debemos definir un punto inicial en el que
% comenzará la optimización y un valor minimo de aumento en cada iteración

[x,FVAL,EXITFLAG,output]=fmincon(@funcionobjetivo1,puntoinicial,[],[],[],[],...
    bimin,[],@definicion_de_restricciones,options);

%Se define la variable a optimizar, en este caso x, la función de
%optimización que será la que hemos definido con
%anterioridad en funcionobjetivo, el punto inicial, el valor mínimo y las
%restricciones que seran las definidas con anterioridad en
%definicion_de_restricciones

% Ejecuta el algoritmo de optimización hasta encontar la solución óptima



%↙
----------------------------------------------------------------------------↙
---

%aquí se lee la solución para ver los valores obtenidos, se calcularan los
%valores de las matrices de covarianzas de x e y óptimos

bi=x;  % valores óptimos de bi

p=0.9973;
alpha=1-p;

load CovIniciales % Se lee la matriz de covarianzas

covarianzasx=zeros(3,3);
covarianzasx(1:3,1:3)=MatCov1;


a=length(covarianzasx); % tamaño de la matriz de covarianzas de x

A=[0.17284 0.0072 -0.01728;10.93 0.4553 -0.364];
% Matriz de relaciones defindа por las especificaciones del montaje
% mecánico(Y=f(X)

tolminx=[0.01;0.01;0.01]; % es el valor de
% tolerancias mínimas para x

toleranciasy=[0.02;0.6];
% toleranciasy =( Valores de Y [(y1=[0.268,0.308] , y2>=17.62]
%- valores nominales de Y [0.288,18.22])
```

```
%-----------------------------------------------------------------------

[C, autov]=eig(covarianzasx); % Obtiene los autovectores y los
% autovalores de la matriz d covarianzas de x

D=diag(autov);% Con los autovalores obtenidos en el comando anterior
% generamos una matriz D, diagonal de autovalores de la matriz
% de covarianzas

newD=bi.*D; % Obtenemos una nueva matriz D modificando los autovalores
%multiplicandola por los factores bi

newcovarianzasx=C*diag(newD)*C'; % con la nueva matriz D obtenemos una
% nueva matriz de covarianzas de x

restriccion1=det(newcovarianzasx);

proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)); % calculamos la proyección
% de la matriz de varianzas de x que acabamos de obtener

Suma_ti=sum(proyecx);

restriccion2=-(proyecx-tolminx); % Realizamos la comparación entre la
                                 % proyección y las tolerancias de x

newcovarianzasy=A*newcovarianzasx*A'; % obtenemos la nueva matriz de
% covarianzas de y a traves de la nueva matriz de covarianzas de x que
%hemos obtenido en el paso anterior

b=length(newcovarianzasy); % tamaño de la matriz de varianzas de y

proyecy=sqrt(chi2inv(p,b)*diag(newcovarianzasy)); % calculamos la
% proyección de la diagonal de la matriz de covarianzas de y

restriccion3=proyecy-toleranciasy; % calculamos la diferencia entre la
% proyección de la diagonal de la matriz de covarianzas de y
% las tolerancias de y

%-----------------------------------------------------------------------

save Etapa1 newcovarianzasx bi % Grabamos la solución,
%  en este caso será las matrices de covarianzas de x  óptimas para esta
% etapa asi como los valores óptimos de bi que se han utilizado.
```

```
% Este programa generara datos aleatorios de
% distribuciones normales multivariantes
% En este ejmplo el enamblaje consta de una piezas,
% por lo que se generarán una sola matriz de covarianzas de 3x3

dim_pieza_sup=3; % Adaptamos la matriz al numero de variables correladas
                 % que definen la pieza 3 variables

m=1000; %Definimos un numero elevado de iteraciones

b1=[0.1 0.1 0.1]; % Valores para el comienzo de la iteración.

x1=zeros(m,dim_pieza_sup); % Generamos una matrices de ceros,de m filas

%-------------------------------------------------------

%-------------------------------------------------------

x1(:,1)=randn(m,1)*b1(1);

for i=2:dim_pieza_sup
    x1(:,i)=rand*x1(:,1)+randn(m,1)*b1(i)*0.5;
end
Matcorr1=corrcoef(x1);
MatCov1=cov(x1);
[autovectores1,au21]=eig(MatCov1);
Sigma1=sqrt(diag(MatCov1));
Propau1=sort(diag(au21)./sum(diag(au21))*100,'descend');

%-------------------------------------------------------

% Este programa genera la  matriz de covarianzas iniciales de
% x que se utilizara el los posteriores programas.

%-------------------------------------------------------

save CovIniciales MatCov1  % Por último guardamos la matriz que hemos
                           %generado para utilizarla posteriormente









%-------------------------------------------------------
```

```
function f=funcionobjetivo1(x)

%En este programa se define la funcion objetivo

bi=x;  % valores por los que iremos multiplicando los autovectores para
       % modificarlos hasta conseguir los valores óptimos que cumplan la
       % función objetivo que estamos definiendo

p=0.9973; % definimos el porcentaje de piezas defectuosas que consideramos
          % aceptable

load CovIniciales % Cargamos el programa que hemos realizado antes para
                  % generar la matriz de covarianzas


covarianzasx=zeros(3,3); % la dimensión de la matriz de covarianzas sera la
                         % correspondiente al numero de variables X totales
                         % que tiene el ensamblaje

covarianzasx(1:3,1:3)=MatCov1; %La matriz de covarianzas es directamente la
%que hemos obtenido en el primer programa, ya que tenemos una unica pieza


%-------------------------------------------------------------------------
a=length(covarianzasx);

[C, autov]=eig(covarianzasx); % Obtiene los autovectores y los
% autovalores de la matriz d covarianzas de x

D=diag(autov);% Con los autovalores obtenidos en el comando anterior
% generamos una matriz D, diagonal de autovalores de la matriz
% de covarianzas

newD=bi.*D; % Obtenemos una nueva matriz D modificando los autovalores
%multiplicandola por los factores bi

newcovarianzasx=C*diag(newD)*C'; % con la nueva matriz D obtenemos una
% nueva matriz de covarianzas de x

proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)); % calculamos la
% proyección de la diagonal de la matriz de covarianzas de x que
% estos son los valores ti

f=-sum(proyecx); % la función objetivo sera buscar la minima diferencia
% entre las proyecciones de la matriz de covarianzas de x y las tolerancias
% de x

%-------------------------------------------------------------------------

save NewX newcovarianzasx
```

```
function [restricciones,nleq]=definicion_de_restricciones(x)

%Aquí se definen las restricciones del problema

load Newx  % esta leyendo newcovarianzasx

bi=x;
p=0.9973;

a=length(newcovarianzasx); % tamaño de la matriz de covarianzas de x

A=[0.17284 0.0072 -0.01728;10.93 0.4553 -0.364];
% Matrz de relaciones defindа por las especificaciones del montaje mecánico
% (Y=f(X))

tolminx=[0.01;0.01;0.01];
% tolerancias mínimas para x

toleranciasy=[0.02;0.6];
% toleranciasy =( Valores de Y [(y1=[0.268,0.308] , y2>=17.2]-
%valores nominales de Y [0.288,18.22])

%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% restricción 1
restriccion1=det(newcovarianzasx); % calculamos el determinante de la nueva
% matriz de covarianzas de x que hemos obtenido

%--------------------------------------------------------------------------
% restricción 2
proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)); % calculamos la
% proyección de la matriz de varianzas de x que acabamos de obtener

restriccion2=-(proyecx-tolminx); % calculamos la diferencia entre la
% proyección de la matriz de varianzas de x y las tolerancias minimas de x
% que hemos definido

%--------------------------------------------------------------------------
% restricción 3
newcovarianzasy=A*newcovarianzasx*A'; %obtenemos la nueva matriz de
% covarianzas de y a traves de la nueva matriz de covarianzas de x que
%hemos obtenido en el paso anterior

b=length(newcovarianzasy); % tamaño de la matriz de covarianzas de y
proyecy=sqrt(chi2inv(p,b)*diag(newcovarianzasy));% calculamos la proyección
% de la matriz de varianzas de y

restriccion3=proyecy-toleranciasy; % calculamos la diferencia entre la
% proyección de la matriz de varianzas de y las tolerancias de y

%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
restricciones=[-restriccion1;restriccion2;restriccion3];
```

```
nleq=[];
% Las restricciones del problema serán: que no se superen las tolerancias
% de y ( utilizamos diferencias d y), que las tolerencias de x sean mayores
% que las tolerancias minima de x que hemos definido (utilizamos
% diferenciasx) y que los determinantes de las matrices de covarianzas de x
% e y sean >0 (utilizamos restriccion1 y restriccion2)

%--------------------------------------------------------------------------
```

```
clear all

% En este prgramas se realiza la Etapa 2.
% necesito cargar la solución obtenida en la Etapa 1

load Etapa1  %lee newcovarianzasc bi
% las variables que lee son los óptimos: newcovarianzasy newcovarianzasx bi

A=[0.17284 0.0072 -0.01728;10.93 0.4553 -0.364];
% Matrz de relaciones defindа por las especificaciones del montaje mecánico
% (Y=f(X))

toleranciasy=[0.02;0.6];
% toleranciasy =( Valores de Y [(y1=[0.268,0.308] , y2>=17.62]
%- valores nominales de Y [0.288,18.22])

bi0=bi; % tomamos como valores iniciales los valores de bi y las matrices
covx=newcovarianzasx; % de covarianzas de x e y obtenidos en la etapa 1

mediax=[5 40 50]'; % Valores nominales de x en columna

mediay=A*mediax; % (0.288,18.22)% Valores nominales de y

a=length(mediax); % número de variables x

b=length(mediay) ;% número de variables y

h=mediay-toleranciasy;

%Las especificaciones de Y son: Y1=[0.268,0.308], Y2>17.62
LES=[0.308 1000]; % En el caso de tener solo límite inferior
% ponemos un valor elevado para el límite superior (caso de y2)

LEI=[0.268 17.2]; % en este caso las dos variables tienen definido el
                  % limite de especificación inferior)

%-------------------------------------------------------------------------

epsilom=0.00005;
p=0.9973;
alpha=0.0027;
FMAX=30.1111;
FMIN=0.000001;

% Empieza a iterar para buscar la solución que implique un alfa de 0.0027.
% Genera valores aleatorios de X según su Cov y media, los trunca en 99.73%
% y con esos valores truncados calcula los valores de Y según Y=F(X).
% Estima la proporción de Y fuera de tolerancias por MonteCarlo.

%-------------------------------------------------------------------------


bi=1;
newcovarianzasx=bi*covx;
T=zeros(10000,3);
```

```
for j=1:10
    n=10^5;
    proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx));

    simX=mvnrnd(mediax,newcovarianzasx,n); %genera valores aleatorios con
    %distribución de media de x y matriz de covarianzas de x
    Tolsupx=mediax+proyecx;
    Tolinfx=mediax-proyecx;

    clear T
    T=(simX>=(ones(n,1)*Tolinfx'))&(simX<=(ones(n,1)*Tolsupx'));
    Tn=double(T);
    Tnp=prod(Tn')';   % Buscamos los valores que estan dentro de las
                      % tolerancias de x obtenidas como proyección
    indiX=find(Tnp==1);
    simXtrun=simX(indiX,:); % Esta matriz contiene las observaciones x,
                            % ya comprobadas
    ntrun=length(simXtrun);

    simYtrun=zeros(ntrun,b);
    simYtrun=simXtrun*A'; % Calcula las y para todos los x seleccionados,
                          % le la función de propdefecto
    [Vpdefec(j)]=Propdefecto(simYtrun,LEI,LES);
end

%-------------------------------------------------------------------------


propdefec1=mean(Vpdefec);
propdefec=propdefec1;


if propdefec>=alpha
    fmax=bi;
    pmax=propdefec;
    fmin=FMIN;
    pmin=0;
else
    fmin=bi;
    pmin=propdefec;
    fmax=FMAX;
    pmax=1;
end
% Hasta aqui calcula para matrices finales obtenidas en la última iteración
% de la Etapa 1 el porcentaje de defectuosos p

%-------------------------------------------------------------------------

% A partir de aqui se empieza a iterar hasta que el porcentaje de
% defectuosas sea alpha

difer=abs(propdefec-alpha);
conta=0;
while difer>epsilom
```

```
    conta=conta+1;
    if conta<2
        bi=fmin+(fmax-fmin)/(pmax-pmin)*(alpha-pmin);
    else
        bi=(fmin+fmax)/2;
    end

    newcovarianzasx=bi*covx;
    for j=1:10
        n=10^5;
        newvarianzasx=diag(newcovarianzasx);
        proyecx=sqrt(chi2inv(p,a)*newvarianzasx);

        simX=mvnrnd(mediax,newcovarianzasx,n);
        Tolsupx=mediax+proyecx;
        Tolinfx=mediax-proyecx;
        clear T
        T=(simX>=ones(n,1)*Tolinfx')&(simX<=ones(n,1)*Tolsupx');
        Tn=double(T);
        Tnp=prod(Tn')';
        indiX=find(Tnp==1);
        simXtrun=simX(indiX,:);
        ntrun=length(simXtrun);
        simYtrun=zeros(ntrun,b);
        simYtrun=simXtrun*A';
        [Vpdefec(j)]=Propdefecto(simYtrun,LEI,LES);
    end
    propdefec=mean(Vpdefec);


    if propdefec>=alpha
        fmax=bi;
        pmax=propdefec;
    else
        fmin=bi;
        pmin=propdefec;
    end

    difer=abs(propdefec-alpha);
    if (conta>5)&(abs(bi-FMAX)<0.00005);break;end
    if (conta>5)&(abs(bi-FMIN)<0.00005);break;end
    if sqrt(bi)<0.009; bi=0.009^2;break;end
    if sqrt(bi)>25;bi=25.555^2;break;end
    if conta>100;conta,break;end

end
factorb=bi; % este es el factor de la Etapa 2
nonconforming=propdefec;

factorfinalCPx=bi*bi0; %este son los factores de Etapa2*Etapa1

%-------------------------------------------------------------------------


%-------------------------------------------------------------------------
```

```
newcovarianzasx=covx*bi;  %con esta matriz calculo los límites de tolerancia de X
proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx));

LTSX=mediax+proyecx;
LTIX=mediax-proyecx;
```

```
function [pro_def]=Propdefecto(RE,LEI,LES)

fi=size(LEI,1);
if fi==1
    LEI=LEI';
end

fi=size(LES,1);
if fi==1
    LES=LES';
end


[n,m]=size(RE);
MLTI=ones(n,1)*LEI';
MLTS=ones(n,1)*LES';
esdef=(RE>MLTS)|(RE<MLTI);
if m>1
    Esdef1=(sum(esdef')>0);
else
    Esdef1=esdef;
end
pro_def=mean(Esdef1);
```

ANEXO 5. PROGRAMAS DE MATLAB PARA EL EJEMPLO 3

```
clear all

% Se quiere asignar tolerancias a 8 variables para cumplir las
% restricciones de 4 variables Y. Etapa1
%-------------------------------------------------------------------------
options=optimset('LargeScale','off');
puntoinicial=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];
bimin=[0.001;0.001;0.001;0.001;0.001;0.001;0.001;0.001;0.001;0.001];

% Para realizar la optimización debemos definir un punto inicial en el que
% comenzará la optimización y un valor minimo de aumento en cada iteración

[x,FVAL,EXITFLAG,output]=fmincon(@funcionobjetivo1,puntoinicial,[],[],[],[],...
    bimin,[],@definicion_de_restricciones,options);

%Se define la variable a optimizar, en este caso x, la función de
%optimización que será la que hemos definido con
%anterioridad en funcionobjetivo, el punto inicial, el valor mínimo y las
%restricciones que seran las definidas con anterioridad en
%definicion_de_restricciones

% Ejecuta el algoritmo de optimización hasta encontar la solución óptima



%↙
----------------------------------------------------------------------------------↙
---

%aquí se lee la solución para ver los valores obtenidos, se calcularan los
%valores de las matrices de covarianzas de x e y óptimos

bi=x;  % valores óptimos de bi

p=0.9973;
alpha=1-p;

load CovIniciales % Se leen las matrices de covarianzas y se unen en una
                  % sola de 8x8
covarianzasx=zeros(10,10);
covarianzasx(1:5,1:5)=MatCov1;
covarianzasx(6:10,6:10)=MatCov2;

a=length(covarianzasx); % tamaño de la matriz de covarianzas de x

A=[0 0 0 -1 1 1 -1 0 0 0;0 1 -1 0 0 0 0 1 -1 0;1 0 0 0 0 0 0 0 0 -1];
% Matriz de relaciones defindа por las especificaciones del montaje
% mecánico(Y=f(X)

tolminx=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];
% es el valor de tolerancias mínimas para x


toleranciasy=[0.1;0.15;0.1];
% toleranciasy =( Valores de Y [(y1>0 , y2>0 , y4=0+-0.01]-
```

```
%valores nominales de Y [0.0015, 0.0515 , 0])

%-----------------------------------------------------------------------

[C, autov]=eig(covarianzasx); % Obtiene los autovectores y los
% autovalores de la matriz d covarianzas de x

D=diag(autov);% Con los autovalores obtenidos en el comando anterior
% generamos una matriz D, diagonal de autovalores de la matriz
% de covarianzas

newD=bi.*D; % Obtenemos una nueva matriz D modificando los autovalores
%multiplicandola por los factores bi

newcovarianzasx=C*diag(newD)*C'; % con la nueva matriz D obtenemos una
% nueva matriz de covarianzas de x

restriccion1=det(newcovarianzasx)

proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)) % calculamos la proyección
% de la matriz de varianzas de x que acabamos de obtener

Suma_ti=sum(proyecx)

restriccion2=-(proyecx-tolminx) % Realizamos la comparación entre la
                                % proyección y las tolerancias de x

newcovarianzasy=A*newcovarianzasx*A'; % obtenemos la nueva matriz de
% covarianzas de y a traves de la nueva matriz de covarianzas de x que
%hemos obtenido en el paso anterior

b=length(newcovarianzasy); % tamaño de la matriz de varianzas de y

proyecy=sqrt(chi2inv(p,b)*diag(newcovarianzasy)); % calculamos la
% proyección de la diagonal de la matriz de covarianzas de y
restriccion3=proyecy-toleranciasy;


%-------------------------------------------------------------------------

save Etapa1 newcovarianzasx bi % Grabamos la solución,
%  en este caso será las matrices de covarianzas de x  óptimas para esta
% etapa asi como los valores óptimos de bi que se han utilizado.
```

```
% Este programa generara datos aleatorios de
% distribuciones normales multivariantes
% En este ejmplo el enamblaje consta de dos piezas,
% por lo que se generarán dos matrices de covarianzas:
% una de 5x5 y otra de 5x5


dim_pieza_sup=5; % Adaptamos la matriz al numero de variables correladas
                 % que definen la pieza inferior, 5 variables
dim_pieza_inf=5; % Adaptamos la matriz al numero de variables correladas
                 % que definen la pieza superior, 5 variables

m=1000; %Definimos un numero elevado de iteraciones

b1=[0.1 0.1 0.1 0.1 0.1]; % Valores para el comienzo de la iteración.
b2=[0.1 0.1 0.1 0.1 0.1]; % Valores para el comienzo de la iteración.

x1=zeros(m,dim_pieza_sup); % Generamos dos matrices de ceros,
                           % cada una de ellas de m filas
x2=zeros(m,dim_pieza_inf); % y columnas acordes con las dimensiones
                           % que se han definido anteriormente
%--------------------------------------------------------

%--------------------------------------------------------

x1(:,1)=randn(m,1)*b1(1);

for i=2:dim_pieza_sup
    x1(:,i)=rand*x1(:,1)+randn(m,1)*b1(i)*0.5;
end
Matcorr1=corrcoef(x1);
MatCov1=cov(x1)
[autovectores1,au21]=eig(MatCov1);
Sigma1=sqrt(diag(MatCov1));
Propau1=sort(diag(au21)./sum(diag(au21))*100,'descend');

%--------------------------------------------------------
x2(:,1)=randn(m,1)*b2(1);

for i=2:dim_pieza_inf
    x2(:,i)=rand*x2(:,1)+randn(m,1)*b2(i)*0.3;
end
Matcorr2=corrcoef(x2);
MatCov2=cov(x2)
[autovectores2,au22]=eig(MatCov2);
Sigma2=sqrt(diag(MatCov2));
Propau2=sort(diag(au22)./sum(diag(au22))*100,'descend');

% Estos dos programas generan las dos matrices de covarianzas iniciales de
% x que se utilizaran el los posteriores programas.

%--------------------------------------------------------
```

```
%-------------------------------------------------------

save CovIniciales MatCov1 MatCov2 % Por último guardamos las dos
                                  % matrices que hemos generado
                                  % para utilizarlas posteriormente

%-------------------------------------------------------
```

```
function f=funcionobjetivo1(x)

%En este programa se define la funcion objetivo

bi=x;  % valores por los que iremos multiplicando los autovectores para
       % modificarlos hasta conseguir los valores óptimos que cumplan la
       % función objetivo que estamos definiendo

p=0.9973; % definimos el porcentaje de piezas defectuosas que consideramos
          % aceptable

load CovIniciales % Cargamos el programa que hemos realizado antes para
                  % generar la matriz de covarianzas


covarianzasx=zeros(10,10); % la dimensión de la matriz de covarianzas sera la
                           % correspondiente al numero de variables X totales
                           % que tiene el ensamblaje

covarianzasx(1:5,1:5)=MatCov1; % dependiendo del numero de variables corre-
covarianzasx(6:10,6:10)=MatCov2; % ladas que tengamos obtendremos una
                                 % matriz de covarianzas total,utilizando las
                                 % dos matrices que se han obtenido en el
                                 % programa anterior para generar la matriz
                                 % de covarienzas

%-------------------------------------------------------------------------
a=length(covarianzasx);

[C, autov]=eig(covarianzasx); % Obtiene los autovectores y los
% autovalores de la matriz d covarianzas de x

D=diag(autov);% Con los autovalores obtenidos en el comando anterior
% generamos una matriz D, diagonal de autovalores de la matriz
% de covarianzas

newD=bi.*D; % Obtenemos una nueva matriz D modificando los autovalores
%multiplicandola por los factores bi

newcovarianzasx=C*diag(newD)*C'; % con la nueva matriz D obtenemos una
% nueva matriz de covarianzas de x

proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)); % calculamos la
% proyección de la diagonal de la matriz de covarianzas de x que
% estos son los valores ti

f=-sum(proyecx); % la función objetivo sera buscar la minima diferencia
% entre las proyecciones de la matriz de covarianzas de x y las tolerancias
% de x

%-------------------------------------------------------------------------

save NewX newcovarianzasx
```

```
function [restricciones,nleq]=definicion_de_restricciones(x)

%Aquí se definen las restricciones del problema

load Newx  % esta leyendo newcovarianzasx

bi=x;
p=0.9973;

a=length(newcovarianzasx); % tamaño de la matriz de covarianzas de x

A=[0 0 0 -1 1 1 -1 0 0 0;0 1 -1 0 0 0 0 1 -1 0;1 0 0 0 0 0 0 0 0 -1];
% Matrz de relaciones defindа por las especificaciones del montaje mecánico
% (Y=f(X))

tolminx=[0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];
% tolerancias mínimas para x

toleranciasy=[0.1;0.15;0.1];
% toleranciasy =( Valores de Y [(y1>0 , y2>0 ,y3=0+-0.01]-
%valores nominales de Y [0.0015, 0.0515 , 0])

%--------------------------------------------------------------------------



%--------------------------------------------------------------------------
% restricción 1
restriccion1=det(newcovarianzasx); % calculamos el determinante de la nueva
% matriz de covarianzas de x que hemos obtenido

%--------------------------------------------------------------------------


%--------------------------------------------------------------------------
% restricción 2
proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx)); % calculamos la
% proyección de la matriz de varianzas de x que acabamos de obtener

restriccion2=-(proyecx-tolminx); % calculamos la diferencia entre la
% proyección de la matriz de varianzas de x y las tolerancias minimas de x
% que hemos definido

%--------------------------------------------------------------------------
% restricción 3
newcovarianzasy=A*newcovarianzasx*A'; %obtenemos la nueva matriz de
% covarianzas de y a traves de la nueva matriz de covarianzas de x que
%hemos obtenido en el paso anterior

b=length(newcovarianzasy); % tamaño de la matriz de covarianzas de y
proyecy=sqrt(chi2inv(p,b)*diag(newcovarianzasy));% calculamos la proyección
% de la matriz de varianzas de y

restriccion3=proyecy-toleranciasy; % calculamos la diferencia entre la
% proyección de la matriz de varianzas de y las tolerancias de y
```

```
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
restricciones=[-restriccion1;restriccion2;restriccion3];
nleq=[];
% Las restricciones del problema serán: que no se superen las tolerancias
% de y ( utilizamos diferencias d y), que las tolerencias de x sean mayores
% que las tolerancias minima de x que hemos definido (utilizamos
% diferenciasx) y que los determinantes de las matrices de covarianzas de x
% e y sean >0 (utilizamos restriccion1 y restriccion2)

%--------------------------------------------------------------------------
```

```
clear all

% En este prgramas se realiza la Etapa 2.
% necesito cargar la solución obtenida en la Etapa 1

load Etapa1  %lee newcovarianzasc bi
% las variables que lee son los óptimos: newcovarianzasy newcovarianzasx bi

A=[0 0 0 -1 1 1 -1 0 0 0;0 1 -1 0 0 0 0 1 -1 0;1 0 0 0 0 0 0 0 0 -1];
% Matrz de relaciones definda por las especificaciones del montaje mecánico
% (Y=f(X))

toleranciasy=[0.1;0.15;0.1];
% toleranciasy =( Valores de Y [(y1>=0.0015 , y2>=0.0515 , y3=
% 0.0049+-0.0175 y4=0+-0.01])- valores nominales de Y
%[0.0015, 0.05151 ,0.0049, 0])


bi0=bi; % tomamos como valores iniciales los valores de bi y las matrices
covx=newcovarianzasx; % de covarianzas de x e y obtenidos en la etapa 1

mediax=[50 20.05 9.9 9.9 30 10 30 30 40 50]';
% Valores nominales de x en columna

mediay=[0.1 0.15 0]; %Valores nominales de y

a=length(mediax); % número de variables x

b=length(mediay) ;% número de variables y


h=[0 0 0.1];

%Las especificaciones de Y son: y1>=0.0015 , y2>=0.0515 ,
%y3=0.0049+-0.0175 y4=0+-0.01]

LES=[1000 1000 0.1]; % En el caso de tener solo límite inferior
% ponemos un valor elevado para el límite superior (caso de y1,y2)

LEI=[0 0 -0.1];% en este caso tenemos límit inferior
% para todas las variables

%-------------------------------------------------------------------------

epsilom=0.00005;
p=0.9973;
alpha=0.0027;
FMAX=30.1111;
FMIN=0.000001;

% Empieza a iterar para buscar la solución que implique un alfa de 0.0027.
% Genera valores aleatorios de X según su Cov y media, los trunca en 99.73%
% y con esos valores truncados calcula los valores de Y según Y=F(X).
% Estima la proporción de Y fuera de tolerancias por MonteCarlo.
```

```
%-------------------------------------------------------------------------


bi=1
newcovarianzasx=bi*covx
T=zeros(10000,12);

for j=1:10
    n=10^5;
    proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx));

    simX=mvnrnd(mediax,newcovarianzasx,n); %genera valores aleatorios con
    %distribución de media de x y matriz de covarianzas de x
    Tolsupx=mediax+proyecx;
    Tolinfx=mediax-proyecx;

    clear T
    T=(simX>=(ones(n,1)*Tolinfx'))&(simX<=(ones(n,1)*Tolsupx'));
    Tn=double(T);
    Tnp=prod(Tn')';   % Buscamos los valores que estan dentro de las
                      % tolerancias de x obtenidas como proyección
    indiX=find(Tnp==1);
    simXtrun=simX(indiX,:); % Esta matriz contiene las observaciones x,
                            % ya comprobadas
    ntrun=length(simXtrun);

    simYtrun=zeros(ntrun,b);
    simYtrun=simXtrun*A'; % Calcula las y para todos los x seleccionados,
                          % le la función de propdefecto
    [Vpdefec(j)]=Propdefecto(simYtrun,LEI,LES);
end

%-------------------------------------------------------------------------


propdefec1=mean(Vpdefec)
propdefec=propdefec1


if propdefec>=alpha
    fmax=bi;
    pmax=propdefec;
    fmin=FMIN;
    pmin=0;
else
    fmin=bi;
    pmin=propdefec;
    fmax=FMAX;
    pmax=1;
end
% Hasta aqui calcula para matrices finales obtenidas en la última iteración
% de la Etapa 1 el porcentaje de defectuosos p

%-------------------------------------------------------------------------
```

```
% A partir de aqui se empieza a iterar hasta que el porcentaje de
% defectuosas sea alpha

difer=abs(propdefec-alpha)
conta=0;
while difer>epsilom
    conta=conta+1
    if conta<2
        bi=fmin+(fmax-fmin)/(pmax-pmin)*(alpha-pmin);
    else
        bi=(fmin+fmax)/2;
    end

    newcovarianzasx=bi*covx;
    for j=1:10
        n=10^5;
        newvarianzasx=diag(newcovarianzasx);
        proyecx=sqrt(chi2inv(p,a)*newvarianzasx);

        simX=mvnrnd(mediax,newcovarianzasx,n);
        Tolsupx=mediax+proyecx;
        Tolinfx=mediax-proyecx;
        clear T
        T=(simX>=ones(n,1)*Tolinfx')&(simX<=ones(n,1)*Tolsupx');
        Tn=double(T);
        Tnp=prod(Tn')';
        indiX=find(Tnp==1);
        simXtrun=simX(indiX,:);
        ntrun=length(simXtrun);
        simYtrun=zeros(ntrun,b);
        simYtrun=simXtrun*A';
        [Vpdefec(j)]=Propdefecto(simYtrun,LEI,LES);
    end
    propdefec=mean(Vpdefec);


    if propdefec>=alpha
        fmax=bi;
        pmax=propdefec;
    else
        fmin=bi;
        pmin=propdefec;
    end

    difer=abs(propdefec-alpha);
    if (conta>5)&(abs(bi-FMAX)<0.00005);break;end
    if (conta>5)&(abs(bi-FMIN)<0.00005);break;end
    if sqrt(bi)<0.009; bi=0.009^2;break;end
    if sqrt(bi)>25;bi=25.555^2;break;end
    if conta>100;conta,break;end

end
factorb=bi % este es el factor de la Etapa 2
nonconforming=propdefec
```

```
factorfinalCPx=bi*bi0  %este son los factores de Etapa2*Etapa1

%-------------------------------------------------------------------------


%-------------------------------------------------------------------------
newcovarianzasx=covx*bi  %con esta matriz calculo los límites de tolerancia de X
proyecx=sqrt(chi2inv(p,a)*diag(newcovarianzasx))

LTSX=mediax+proyecx
LTIX=mediax-proyecx
```

```
function [pro_def]=Propdefecto(RE,LEI,LES)

fi=size(LEI,1);
if fi==1
    LEI=LEI';
end

fi=size(LES,1);
if fi==1
    LES=LES';
end


[n,m]=size(RE);
MLTI=ones(n,1)*LEI';
MLTS=ones(n,1)*LES';
esdef=(RE>MLTS)|(RE<MLTI);
if m>1
    Esdef1=(sum(esdef')>0);
else
    Esdef1=esdef;
end
pro_def=mean(Esdef1);
```

ANEXO 6. MÉTODOS DE OPTIMIZACIÓN.

- **Función “fmincon” de Matlab**
- **Método de la bisección.**
- **Método de la secante**

```
>> help fmincon
 FMINCON finds a constrained minimum of a function of several variables.
    FMINCON attempts to solve problems of the form:
     min F(X)  subject to:  A*X  <= B, Aeq*X  = Beq (linear constraints)
      X                     C(X) <= 0, Ceq(X) = 0   (nonlinear constraints)
                               LB <= X <= UB        (bounds)

    FMINCON implements four different algorithms: interior point, SQP, active
    set, and trust region reflective. Choose one via the option Algorithm:
    for instance, to choose SQP, set OPTIONS = optimset('Algorithm','sqp'),
    and then pass OPTIONS to FMINCON.

    X = FMINCON(FUN,X0,A,B) starts at X0 and finds a minimum X to the
    function FUN, subject to the linear inequalities A*X <= B. FUN accepts
    input X and returns a scalar function value F evaluated at X. X0 may be
    a scalar, vector, or matrix.

    X = FMINCON(FUN,X0,A,B,Aeq,Beq) minimizes FUN subject to the linear
    equalities Aeq*X = Beq as well as A*X <= B. (Set A=[] and B=[] if no
    inequalities exist.)

    X = FMINCON(FUN,X0,A,B,Aeq,Beq,LB,UB) defines a set of lower and upper
    bounds on the design variables, X, so that a solution is found in
    the range LB <= X <= UB. Use empty matrices for LB and UB
    if no bounds exist. Set LB(i) = -Inf if X(i) is unbounded below;
    set UB(i) = Inf if X(i) is unbounded above.

    X = FMINCON(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON) subjects the minimization
    to the constraints defined in NONLCON. The function NONLCON accepts X
    and returns the vectors C and Ceq, representing the nonlinear
    inequalities and equalities respectively. FMINCON minimizes FUN such
    that C(X) <= 0 and Ceq(X) = 0. (Set LB = [] and/or UB = [] if no bounds
    exist.)

    X = FMINCON(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON,OPTIONS) minimizes with
    the default optimization parameters replaced by values in the structure
    OPTIONS, an argument created with the OPTIMSET function. See OPTIMSET
    for details. For a list of options accepted by FMINCON refer to the
    documentation.

    X = FMINCON(PROBLEM) finds the minimum for PROBLEM. PROBLEM is a
    structure with the function FUN in PROBLEM.objective, the start point
    in PROBLEM.x0, the linear inequality constraints in PROBLEM.Aineq
    and PROBLEM.bineq, the linear equality constraints in PROBLEM.Aeq and
    PROBLEM.beq, the lower bounds in PROBLEM.lb, the upper bounds in
    PROBLEM.ub, the nonlinear constraint function in PROBLEM.nonlcon, the
    options structure in PROBLEM.options, and solver name 'fmincon' in
    PROBLEM.solver. Use this syntax to solve at the command line a problem
    exported from OPTIMTOOL. The structure PROBLEM must have all the fields.

    [X,FVAL] = FMINCON(FUN,X0,...) returns the value of the objective
    function FUN at the solution X.

    [X,FVAL,EXITFLAG] = FMINCON(FUN,X0,...) returns an EXITFLAG that
    describes the exit condition of FMINCON. Possible values of EXITFLAG
```

```
and the corresponding exit conditions are listed below. See the
documentation for a complete description.

All algorithms:
  1  First order optimality conditions satisfied.
  0  Too many function evaluations or iterations.
 -1  Stopped by output/plot function.
 -2  No feasible point found.
Trust-region-reflective, interior-point, and sqp:
  2  Change in X too small.
Trust-region-reflective:
  3  Change in objective function too small.
Active-set only:
  4  Computed search direction too small.
  5  Predicted change in objective function too small.
Interior-point and sqp:
 -3  Problem seems unbounded.

[X,FVAL,EXITFLAG,OUTPUT] = FMINCON(FUN,X0,...) returns a structure
OUTPUT with information such as total number of iterations, and final
objective function value. See the documentation for a complete list.

[X,FVAL,EXITFLAG,OUTPUT,LAMBDA] = FMINCON(FUN,X0,...) returns the
Lagrange multipliers at the solution X: LAMBDA.lower for LB,
LAMBDA.upper for UB, LAMBDA.ineqlin is for the linear inequalities,
LAMBDA.eqlin is for the linear equalities, LAMBDA.ineqnonlin is for the
nonlinear inequalities, and LAMBDA.eqnonlin is for the nonlinear
equalities.

[X,FVAL,EXITFLAG,OUTPUT,LAMBDA,GRAD] = FMINCON(FUN,X0,...) returns the
value of the gradient of FUN at the solution X.

[X,FVAL,EXITFLAG,OUTPUT,LAMBDA,GRAD,HESSIAN] = FMINCON(FUN,X0,...)
returns the value of the exact or approximate Hessian of the Lagrangian
at X.

Examples
  FUN can be specified using @:
     X = fmincon(@humps,...)
  In this case, F = humps(X) returns the scalar function value F of
  the HUMPS function evaluated at X.

  FUN can also be an anonymous function:
     X = fmincon(@(x) 3*sin(x(1))+exp(x(2)),[1;1],[],[],[],[],[0 0])
  returns X = [0;0].

If FUN or NONLCON are parameterized, you can use anonymous functions to
capture the problem-dependent parameters. Suppose you want to minimize
the objective given in the function myfun, subject to the nonlinear
constraint mycon, where these two functions are parameterized by their
second argument a1 and a2, respectively. Here myfun and mycon are
M-file functions such as

     function f = myfun(x,a1)
     f = x(1)^2 + a1*x(2)^2;
```

```
        function [c,ceq] = mycon(x,a2)
        c = a2/x(1) - x(2);
        ceq = [];
```

To optimize for specific values of a1 and a2, first assign the values to these two parameters. Then create two one-argument anonymous functions that capture the values of a1 and a2, and call myfun and mycon with two arguments. Finally, pass these anonymous functions to FMINCON:

```
        a1 = 2; a2 = 1.5; % define parameters first
        options = optimset('Algorithm','interior-point'); % run interior-point↙
algorithm
        x = fmincon(@(x) myfun(x,a1),[1;2],[],[],[],[],[],[],@(x) mycon(x,a2),↙
options)
```

See also optimset, optimtool, fminunc, fminbnd, fminsearch, @, function_handle.

Reference page in Help browser

```
   doc fmincon
```

```
>>
```

EL MÉTODO DE LA BISECCIÓN

Teorema de Bolzano

Sea $f : [a, b] \subset \mathbb{R} \to \mathbb{R}$ *una función continua en* $[a, b]$ *tal que* $f(a) \cdot f(b) < 0$*, es decir, que tiene distinto signo en* a *y en* b*. Entonces, existe* $c \in (a, b)$ *tal que* $f(c) = 0$

El Teorema de Bolzano afirma que si una función es continua en un intervalo cerrado y acotado y en los extremos del mismo ésta toma valores con signos opuestos, entonces existe al menos una raíz de la función en el interior del intervalo.

Demostración:

Supongamos que $f(a) < 0$ *y* $f(b) > 0$*. Sea* T *el conjunto formado por todos los valores* x / $x \in [a, b]$ *para los que* $f(x) < 0$*. El conjunto* T *está acotado superiormente por* b *y, además, no es vacío ya que* a *pertenece a* T*. Por ello el conjunto* T *tiene un extremo superior* c*. Se cumple que* $f(c) = 0$*. Veamoslo:*

Si $f(c) > 0$*, entonces por la propiedad de la conservación del signo de las funciones continuas existiría un intervalo* $(c - \delta, c + \delta)$ *en el que la función sería también positiva. En este caso existirían valores menores que* c *que servirían de cota superior de* T *y por ello* c *no sería el extremo superior de* T *como hemos supuesto.*

Si $f(c) < 0$*, entonces existiría un intervalo* $(c - \delta, c + \delta)$ *en el que la función sería negativa y por tanto existirían valores de* x *a la derecha de* c *para los que la función sería negativa y por tanto* c *no sería el extremo superior de* T*. Por tanto* $f(c)$ *tiene que tomar el valor cero:* $f(c) = 0$*.*

Si $f(a) > 0$ *y* $f(b) < 0$ *el razonamiento es similar.*

De forma más general obtenemos **El Teorema del Valor Intermedio**:

El Teorema del Valor Intermedio

Sea $f : [a, b] \subset \mathbb{R} \to \mathbb{R}$ *continua en* $[a, b]$*, y tal que* $f(a) < f(b)$ *entonces, para cualquier* k *tal que* $f(a) < k < f(b)$ *existe* $x_0 \in (a, b)$ *tal que* $f(x_0) = k$

Básicamente el Teorema del Valor Intermedio nos dice que toda función contínua en un intervalo cerrado, una vez que alcanzó ciertos valores en los extremos del intervalo, entonces debe alcanzar todos los valores intermedios.

Demostración:

Para la demostración aplicamos el teorema de Bolzano en la función $g(x) = f(x) - k$*, la cual es continua, por serlo* $f(x)$*,* $g(a) < 0$ *y* $g(b) > 0$*. El teorema nos permite afirmar que existirá* $c \in (a, b)$ *tal que* $g(c) = 0$ *y en consecuencia* $f(c) = k$*.*

El método de la bisección se basa en estos teoremas y se emplea para aproximar ceros de funciones.

Supóngase que queremos encontrar los ceros de una función $f(x)$ continua. Dados dos puntos a y b tal que $f(a)$ y $f(b)$ tengan signos distintos, sabemos por el Teorema de Bolzano que $f(x)$ debe tener, al menos, una raíz en el intervalo $[a, b]$. El método de bisección divide el intervalo en dos, usando un tercer punto $c = \dfrac{(a+b)}{2}$. En este momento, existen dos posibilidades: $f(a)$ y $f(c)$, ó $f(c)$ y $f(b)$ tienen distinto signo. El método de bisección se aplica al subintervalo donde el cambio de signo ocurre. Este proceso puede aplicarse tantas veces como sea necesario para alcanzar la precisión que se requiera.

Páginas web relacionadas con el tema:

(1) http://www-ma3.upc.edu/users/carmona/teaching.html

Podréis encontrar un gráfico ilustrativo del método de la bisección.

(2) http://www.vitutor.com/fun/3/c 3.html

Podréis encontrar el enunciado del Teorema de Bolzano y ejercicios resueltos.

(3) http://www.matematicasbachiller.com/temario/sel/enuncia/seten07.htm

Los videos 7.09, 7.10 y 7.11 están relacionados con el Teorema de Bolzano. Aparece algo llamado la propiedad de Darboux que no es más que el Teorema de los valores intermedios.

Ejemplo 1.- Se considera la función $f : \mathbb{R} \to \mathbb{R}$ continua y acotada. Demostrar que la ecuación $f(x) - x = 0$ tiene al menos una raíz real.

Consideramos la función $h(x) = f(x) - x$. Dicha función es continua por ser diferencia de funciones continuas. Por ser f acotada en $\mathbb{R}$ existe un $M \in (0, +\infty)$ tal que

$$-M < f(x) < M \quad \text{para todo} \quad x \in \mathbb{R}.$$

Por tanto, para todo $x \in \mathbb{R}$ tenemos que $f(x) - M < 0$ y $f(x) + M > 0$ y en consecuencia

$$h(M) = f(M) - M < 0$$

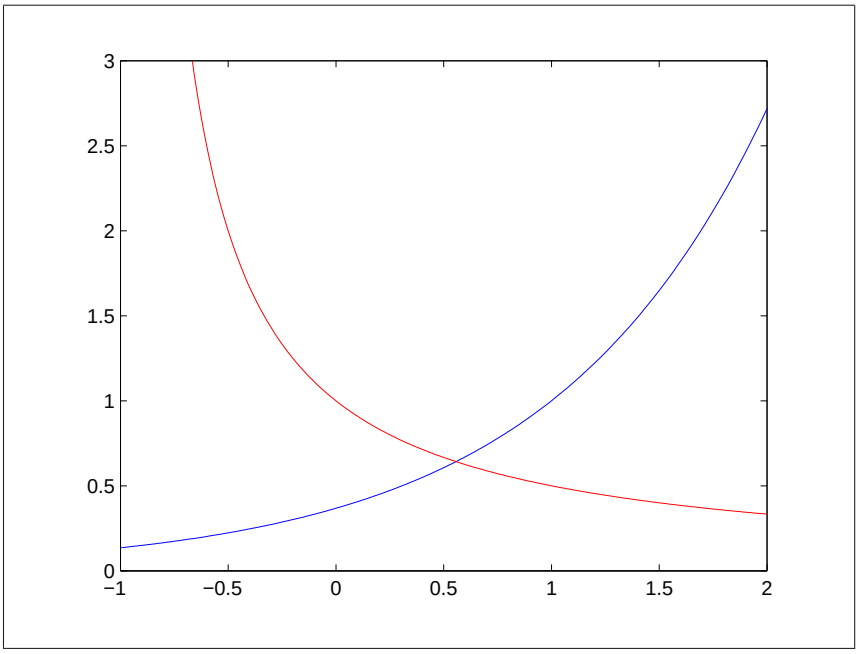
$$h(-M) = f(-M) + M > 0.$$

Por el Teorema de Bolzano existe $c \in [-M, M]$ tal que $h(c) = 0$.

Ejemplo 2.- Encontrar un intervalo en el que se pueda asegurar que existe alguna solución de la siguiente ecuación (Puedes ayudarte gràficamente).

$$e^{t-1} = \frac{1}{1+t}$$

Haciendo un dibujo aproximado podemos determinar un intervalo dónde encontrar la solución de la ecuación.



Podemos observar que la solución de la ecuación se encuentra entre t=0 y t=1. Aplicamos ahora el Teorema de Bolzano para demostrarlo. Consideramos $f(t) = e^{t-1} - \frac{1}{1+t}$. Se trata de una función continua en el intervalo $(-1, +\infty)$ ya que es suma y cociente de continuas y no se anula el denominador. Por otro lado, $f(0) = e^{-1} - 1 < 0$ y $f(1) = \frac{1}{2} > 0$. Por tanto, el Teorema de Bolzano asefura que existe $c \in (0, 1)$ tal que $f(c) = 0$.

Ejemplo 3.- Encontrar la raíz de $f(x) = e^{-x} - \ln(x)$ con un error más pequeño que media décima.

$f(x)$ es continua en $(0, +\infty)$. Buscamos x_1 y x_2 tal que $f(x_1) \cdot f(x_2) < 0$.

$$x_1 = 1 \qquad x_2 = 2$$

$$f(x_1) = 0{,}3679 \qquad f(x_2) = -0{,}5578$$

$$x_3 = \frac{x_1 + x_2}{2} = 1{,}5 \implies \varepsilon < 0{,}5, \text{ ya que } |1 - 1{,}5| = 0{,}5 \text{ y } |2 - 1{,}5| = 0{,}5$$

$$f(x_3) = -0{,}1823$$

$$x_1 = 1 \qquad x_3 = 1{,}5$$

$$f(x_1) = 0{,}3679 \qquad f(x_3) = -0{,}1823$$

$$x_4 = \frac{x_1 + x_3}{2} = 1{,}25 \implies \varepsilon < 0{,}25, \text{ ya que } |1{,}5 - 1{,}25| = 0{,}25 \text{ y } |1 - 1{,}25| = 0{,}25$$

$$f(x_4) = 0{,}0634$$

$$x_4 = 1{,}25 \qquad x_3 = 1{,}5$$

$$f(x_4) = 0{,}0634 \qquad f(x_3) = -0{,}1823$$

$$x_5 = \frac{x_3 + x_4}{2} = 1{,}375 \implies \varepsilon < 0{,}125$$

$$f(x_5) = -0{,}0656$$

$$x_4 = 1{,}25 \qquad x_5 = 1{,}375$$

$$f(x_4) = 0{,}0634 \qquad f(x_5) = -0{,}0656$$

$$x_6 = \frac{x_4 + x_5}{2} = 1{,}3125 \implies \varepsilon < 0{,}0625$$

$$f(x_6) = -0{,}0028$$

$$x_4 = 1{,}25 \qquad x_6 = 1{,}3125$$

$$f(x_4) = 0{,}0634 \qquad f(x_6) = -0{,}0028$$

$$x_7 = \frac{x_4 + x_6}{2} = 1{,}28125 \implies \varepsilon < 0{,}03125$$

$$f(x_7) = 0{,}0299$$

De esta manera obtenemos que $\boxed{1{,}28125 \pm 0{,}03125}$ es una raíz de $f(x)$.

Ejemplo 4.- Encontrar x con un error más pequeño que 0.05 el punto de corte de las funciones $h(x) = \text{sen}(x)$ y $g(x) = -x + 1$.

Dado que queremos encontrar la solucion de la ecuación $senx = -x + 1$ lo que vamos a hacer es definir la función $f(x) = \text{sen}(x) + x - 1$ y encontraremos sus ceros mediante el método de la bisección. Observamos que $f(x)$ es continua en $(-\infty, +\infty)$ por ser suma de funciones elementales. Buscamos x_1 y x_2 tal que $f(x_1) \cdot f(x_2) < 0$.

$$x_1 = 0 \qquad x_2 = 1$$

$$f(x_1) = -1 \qquad f(x_2) = 0{,}8415$$

$$x_3 = \frac{x_1 + x_2}{2} = 0{,}5 \implies \varepsilon < 0{,}5$$

$$f(x_3) = -0{,}0206$$

$$x_3 = 0{,}5 \qquad x_2 = 1$$

$$f(x_3) = -0{,}0206 \qquad f(x_2) = 0{,}8415$$

$$x_4 = \frac{x_3 + x_2}{2} = 0{,}75 \implies \varepsilon < 0{,}25$$

$$f(x_4) = 0{,}4316$$

$$x_3 = 0{,}5 \qquad x_4 = 0{,}75$$

$$f(x_3) = -0{,}0206 \qquad f(x_4) = 0{,}4316$$

$$x_5 = \frac{x_3 + x_4}{2} = 0{,}625 \implies \varepsilon < 0{,}125$$

$$f(x_5) = 0{,}2101$$

$$x_3 = 0{,}5 \qquad x_5 = 0{,}625$$

$$f(x_3) = -0{,}0206 \qquad f(x_5) = 0{,}2101$$

$$x_6 = \frac{x_3 + x_5}{2} = 0{,}5625 \implies \varepsilon < 0{,}0625$$

$$f(x_6) = 0{,}0958$$

$$x_3 = 0{,}5 \qquad x_6 = 0{,}5625$$

$$f(x_3) = -0{,}0206 \qquad f(x_6) = 0{,}0958$$

$$x_7 = \frac{x_3 + x_6}{2} = 0{,}53125 \implies \varepsilon < 0{,}03125$$

$$f(x_7) = 0{,}0379$$

Hemos encontrado que $\boxed{0{,}53125 \pm 0{,}03125}$ es solución de la ecuación y por tanto será el punto de corte de las dos funciones dadas.

RAÌCES DE ECUACIONES
Método de la Secante

Ing Yamil Armando Cerquera Rojas – **yacerque@gmail.com**
Especialista en Sistemas Universidad Nacional
Docente Universidad Surcolombiana
Neiva - Huila

Contenido

Método de la secante

Introducción

El principal inconveniente del método de Newton estriba en que requiere conocer el valor de la primera derivada de la función en el punto, lo cual puede llegar a resultar engorroso. Sin embargo, la forma funcional de $f(x)$ dificulta en ocasiones el cálculo de la derivada. El método de la secante es casi idéntico al de regula falsi salvo por un detalle: no se tiene en cuenta el signo de la función para estimar el siguiente punto. Se procede independientemente de los signos de la función. De todas maneras en algunos casos es más útil emplear el método de la secante.

Este método, a diferencia del de bisección y regla falsa, casi nunca falla ya que solo requiere de 2 puntos al principio, y después el mismo método se va retroalimentando. Lo que hace básicamente es ir tirando rectas secantes a la curva de la ecuación que se tiene originalmente, y va chequeando la intersección de esas rectas con el eje de las X para ver si es la raíz que se busca.

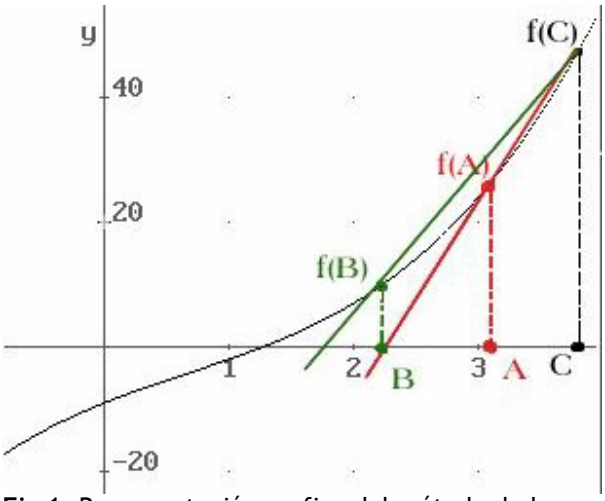


Fig 1. Representación grafica del método de la secante

Una forma de evitar el cálculo de $f'(x)$ consiste en considerar como aproximación a la derivada la recta que pasa por los valores de 2 iteraciones sucesivas (estima la tangente) es decir, la pendiente de la recta) $f'(x_0) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$. Esta variante se conoce con el nombre de método de la Secante. Sustituyendo esta expresión en la ecuación del método de Newton, se obtiene la expresión del método de la secante que proporciona el siguiente punto de iteración: $x_2 = x_1 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_1)$

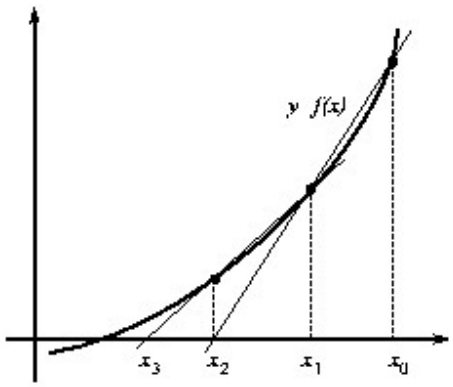


Fig 2. Representación geométrica de las iteraciones al aplicar el método de la secante

.

La sucesión queda expresada en términos generales como:

$$x_n = x_{n-1} - \left[\frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})} \right] f(x_{n-1})$$

A partir de ciertos valores x_0 y x_1 dados. El algoritmo deberá parar cuando $|x_{n+1} - x_n|$ sea menor que la precisión requerida. Obviamente, para poder arrancar el método se necesitan dos valores iniciales.

Forma de hacerlo:

Primero hay que definir algunos conceptos como:

x_n Es el valor actual de X
x_{n-1} Es el valor anterior de X
x_{n+1} Es el valor siguiente de X

Como su nombre lo dice, este método va trazando rectas secantes a la curva original, y como después del primer paso no depende de otras cantidades sino que solito va usando las que ya se obtuvieron, casi nunca falla porque se va acomodando y hará que encuentra la raíz.

Lo primero que se hace, igual que con otros métodos es dar 2 puntos cualesquiera que sean sobre el eje de las X que se llaman A y C.

Después se sustituyen esos puntos en la ecuación original para obtener f(A) y f(C). Una vez que se tienen todos esos datos se obtiene el punto B con la fórmula B=((Af(C))-(C(f(A)))/(f(C)-f(A)).

A diferencia del resto de los métodos, aquí no hay que acomodar en columnas cada uno de los datos, sino que se utiliza la simplificación de conceptos y como se simplifica la formula para seguir con el método.

Aquí solo se usan 2 columnas, una de x_n y otra de $f(x_n)$.

Ejemplo 1

Usar el método de la secante para calcular la raiz aproximada de la función $f(x)=x^2-4$. Comenzando con $x_0=4$, $x_1=3$ y hasta que $|\in_r|\leq 1\%$.

Aplicando para la primera iteración con la fórmula $x_2=x_1-\frac{x_1-x_0}{f(x_1)-f(x_0)}f(x_1)$, se tendría un valor para $x_2=2.2857$. Si se calcula el error relativo con los valores x_2 como valor real y x_1 como valor aproximado se tendrá: $\in_r=\left|\frac{3-2.2857}{2.2857}\right|*100\%=31.25\%$, Ver primer resultado en la figura 3:

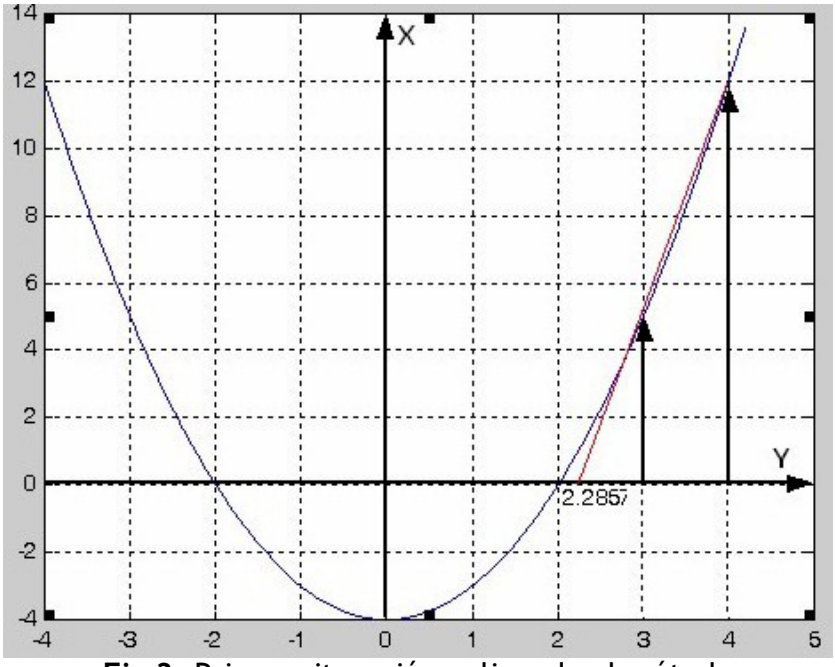


Fig 3. Primera iteración aplicando el método de la secante de la función $f(x)=x^2-4$

Ahora si se calcula en una segunda iteración $x_3 = x_2 - \frac{x_2 - x_1}{f(x_2) - f(x_1)} f(x_2)$, se tendría un valor para $x_3 = 2.0541$, con un error relativo $\in_r = \left| \frac{2.2857 - 2.0541}{2.0541} \right| * 100\% = 11.28\%$.
Ahora si se continúa realizando los cálculos iterativamente, se tendrán valores como los mostrados en la siguiente tabla.

Tabla 1. Resultados al aplicar el método de la Secante a la función $f(x) = x^2 - 4$. Con $x_0 = 4$ y $x_1 = 3$

i	x_i	x_{i+1}	x_{i+2}	$\in_a$	$\in_r$
0	4	3	2.2857	0.7143	31.25%
1	3	2.2857	2.0541	0.2316	11.28%
2	2.2057	2.0541	2.0036	0.0505	2.52%
3	2.0541	2.0036	2.0000	0.0036	0.18%

Se termina el proceso iterativo con la encontrada de la raíz para $x_5 = 2.0000$.

Ejemplo 2

Usar el método de la secante para aproximar la raíz de $f(x) = e^{-x^2} - x$, comenzando con $x_0 = 0$, $x_1 = 1$ y hasta que $|\in_r| \leq 1\%$.

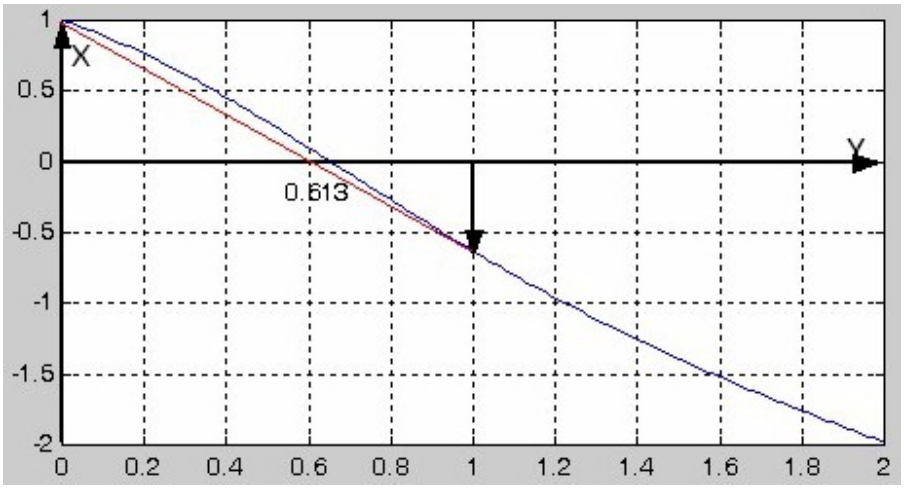


Fig 4. Primera iteración para la $f(x) = e^{-x^2} - x$, con $x_0 = 0$, $x_1 = 1$, aplicando el método de la secante

Solución

Se Tiene que $f(x_0) = 1$ y $f(x_1) = -0.632120558$, que se sustituye en la fórmula de la secante para calcular la aproximación x_2

$$x_n = x_{n-1} - \left[\frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})}\right] f(x_{n-1}) = 0.612699837$$

Con un error aproximado de:

$$|\in_r| = \left|\frac{x_2 - x_1}{x_1} 100\%\right| = 63.2\%$$

Como todavía no se logra el objetivo, continuamos con el proceso. Resumimos los resultados en la siguiente tabla:

Aprox. a la raíz	Error aprox.
0	
1	100%
0.612699837	63.2%
0.653442133	6.23%
0.652917265	0.08%

De lo cual se concluye que la aproximación a la raíz es:

$$x_4 = 0.652917265$$

Ejemplo 3

Usar el método de la secante para aproximar la raíz de $f(x) = \arctan(x - 2x + 1)$, comenzando con $x_0 = 0$, $x_1 = 1$, y hasta que el error relativo $|\in_r| \leq 1\%$.

Solución

Se tienen los valores $f(x_0) = 1$ y $f(x_1) = -0.214601836$, que se sustituyen en la fórmula del método de la secante para obtener la aproximación x_2:

$$x_n = x_{n-1} - \left[\frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})}\right] f(x_{n-1}) = 0.823315073$$

Con un error aproximado de:

$$|\in_r| = \left|\frac{x_2 - x_1}{x_1} 100\%\right| = 21.46\%$$

Como todavía no se logra el objetivo, se continúa con el proceso. Se resumen los resultados en la siguiente tabla:

Aprox. a la raíz	Error aprox.
0	
1	100%
0.823315073	21.4%
0.852330280	3.40%
0.853169121	0.09%

De lo cual se concluye que la aproximación a la raíz es:

$$x_4 = 0.853169121$$