GOBIERNO DE MÉXICO | EDUCACIÓN SECRETARÍA DE EDUCACIÓN PÚBLICA | TECNOLÓGICO NACIONAL DE MÉXICO

TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE DURANGO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

"Detección e identificación de fallas en un reactor químico heterogéneo para prevenir la emisión de contaminantes a la atmósfera"

TESIS

Como parte de los requisitos para obtener el grado de

MAESTRO EN SISTEMAS AMBIENTALES

Presenta

Ing. Leopoldo Campos Carrillo

Director de tesis:

Dr. Sergio Valle Cervantes

Victoria de Durango, Dgo., México Noviembre, 2019

CONACYT Consejo Nacional de Ciencia y Tecnología

“Detección e identificación de fallas en un reactor químico heterogéneo para prevenir la emisión de contaminantes a la atmósfera”

Por

Ing. Leopoldo Campos Carrillo

COMITÉ TUTORIAL

DIRECTOR DE TESIS

Dr. Sergio Valle Cervantes _____

ASESORES

Dr. Armando de la Peña Arellano _____

M. C. Rafael Lucho Chigo _____

"2019, Año del Caudillo del Sur, Emiliano Zapata"

Victoria de Durango, Dgo., a 28 / Octubre / 2019.

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
DEPI / C / 491 / 19.
ASUNTO: Autorización de Impresión de Tesis de Maestría.

ING. LEOPOLDO CAMPOS CARRILLO
No. DE CONTROL G97040211
P R E S E N T E .

De acuerdo al reglamento en vigor y tomando en cuenta el dictamen emitido por el jurado que le fue asignado para la revisión de su trabajo de tesis para obtener el **Grado de Maestro en Sistemas Ambientales,** esta División de Estudios de Posgrado e Investigación le autoriza la impresión del mismo, cuyo título es:

"DETECCIÓN E IDENTIFICACIÓN DE FALLAS EN UN REACTOR QUÍMICO HETEROGÉNEO PARA PREVENIR LA EMISIÓN DE CONTAMINANTES A LA ATMÓSFERA"

Sin otro particular de momento, quedo de Usted.

A T E N T A M E N T E .

"La Técnica al Servicio de la Patria"

C. LUZ ARACELI OCHOA MARTÍNEZ
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

LAOM'ammc.

"2019, Año del Caudillo del Sur, Emiliano Zapata"

Victoria de Durango, Dgo., a 31 / Octubre / 2019.

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
DEPI / C / 505 / 19.
ASUNTO: Autorización de Tema de Tesis de Maestría.

ING. LEOPOLDO CAMPOS CARRILLO
No. DE CONTROL G97040211
P R E S E N T E .

Con base en el Reglamento en vigor y teniendo en cuenta el dictamen emitido por el Jurado que le fue asignado, se le autoriza a desarrollar el tema de tesis para obtener el **Grado de Maestro en Sistemas Ambientales** cuyo título es:

"DETECCIÓN E IDENTIFICACIÓN DE FALLAS EN UN REACTOR QUÍMICO HETEROGÉNEO PARA PREVENIR LA EMISIÓN DE CONTAMINANTES A LA ATMÓSFERA"

CONTENIDO:

Sin otro asunto en particular, quedo de Usted.

A T E N T A M E N T E .

"La Técnica al Servicios de la Patria"

C. LUZ ARACELI OCHOA MARTÍNEZ
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

LAOM'ammc.

Resumen

En esta investigación se desarrolla un sistema de detección e identificación de fallas en un reactor multi-tubular heterogéneo para prevenir la emisión de contaminantes en la atmósfera utilizando la técnica de la descomposición de valores singulares (SVD). Dicha técnica permitió obtener las principales variables dentro del proceso de producción de formaldehído dentro del reactor catalítico. Después de la depuración y aplicación de la técnica SVD se elaboraron las gráficas de contribución que permitieron la identificación de las variables del comportamiento anómalo. El análisis llevado a cabo permitió crear un modelo y determinar las características normales de la planta cuando el reactor se desempeña en un estado estable.

Abstract

In this research, a fault detection and identification system in a heterogeneous multi-tubular reactor is developed to prevent the emission of pollutants into the atmosphere using the singular value decomposition (SVD) technique. This techniques allows to obtain the main variables within the formaldehyde production process within the catalytic reactor. After the purification and application of the SVD technique, the contribution graphs were developed that allowed the identification of the anomalous behavior variables. The analysis carried out allowed to create a model and determine the normal characteristics of the plant when the reactor performs in steady state.

Índice General

Lista de Figuras.

Lista de Símbolos y Nomenclatura.

FDI	Detección e Identificación de Fallas.
SVD	Descomposición de Valores Singulares.
PCA	Análisis de Componentes Principales.
SPE / Q	Error Cuadrático de Predicción.
PRESS	Suma de Cuadrados del Error Residual Pronosticado.
T^2	Valor estadístico de Hotelling.
M	Matriz de datos para el modelo PCA.
N	Matriz de datos ajustados y centrados.
x_{ij}	Dato i-ésimo perteneciente a la matriz **M**.
$\overline{x}_i$	Media aritmética del conjunto de datos de cada variable.
σ_i	Desviación estándar del conjunto de datos de cada variable.
$\widetilde{x}_{ij}$	Dato escalado perteneciente a la matriz **N.**
E	Matriz residual del modelo PCA**.**
γ_k / φ_k	Valor del componente principal**.**
P	Matriz de predicción de los datos ajustados.

Lista de Tablas.

A mi esposa Dany, mis hijos Iker e Ilhan,
a Mamá Rosita† y mi Padrino Rubén†.

Por su amor y comprensión…

Capítulo 1 Introducción.

1.1. Antecedentes

El desarrollo histórico de los diversos métodos de supervisión, detección de fallas y diagnóstico es difícil de describir porque las contribuciones originales están muy distribuidas en la literatura técnica. La comprobación de límites es probablemente tan antigua como la instrumentación de máquinas que se remonta a finales del siglo XIX. Para la supervisión de plantas, el uso de tinta y grabadoras de puntos posteriores fue un equipo estándar desde 1935. Más tarde, alrededor de 1960, los controladores analógicos con amplificadores basados en transistores (amplificadores de operación) y controladores secuenciales, con dispositivos cableados, estuvieron disponibles y luego se usaron comprobación de límites. Los métodos basados en modelos de señal como el análisis espectral podrían realizarse con filtros de paso de banda analógicos y con osciloscopios (Isermann, 2006).

La implementación de computadoras en proceso operativo en línea en 1960 abrió el camino para mejores métodos de supervisión, como el análisis de tendencias. En 1968 se programó por primera vez: se introdujeron controladores lógicos para reemplazar los controladores cableados por relés electromecánicos. Esto facilitó la realización de sistemas de protección. El advenimiento de la microcomputadora en 1971 y su creciente aplicación en sistemas de automatización de procesos descentralizados desde l975 fue el comienzo de una computación más involucrada, supervisión basada en software y algoritmos de detección de fallas. Aparecieron primeras publicaciones sobre métodos de detección de fallas basados en modelos de procesos en conexión con sistemas aeroespaciales y con plantas químicas. Varios de estos primeros conceptos se pueden clasificar como enfoques de relación de paridad, verificando la consistencia de las lecturas de los instrumentos

o los balances de masa o materia. Los residuos del balance de masa se aplicaron, por ejemplo, para la detección de fugas de tuberías (Isermann, 2006).

La diagnosis de fallos basada en modelos matemáticos comenzó con los trabajos pioneros de Clark (Harman *et al*., 1975) entre otros. Desde entonces, se ha desarrollado una gran cantidad de investigación en esta área. Como resultado, hoy en día existen un conjunto de métodos que conforman la base de este campo y pueden ser considerados como los cimientos de los métodos más avanzados. Los métodos base de la detección de fallos se suelen clasificar en: basados en ecuaciones de paridad (Gertler *et al*., 1998), basados en observadores (Chen *et al*., 2012) y basados en estimación paramétrica (Isermann, 2006). Las relaciones entre todos estos métodos han sido establecidas por varios autores (Gertler *et al*., 1998).

Se han realizado diversas investigaciones en el campo de la detección de fallas y sobre todo utilizando métodos matemáticos y estadísticos. En la Tabla 1.1 se muestra un resumen de algunos trabajos previos realizados utilizando dichos métodos para detección de fallas.

Tabla 1.1. Investigaciones realizadas de detección de fallas utilizando modelos y métodos matemáticos

Método	Investigación	País	Referencia
Modelo múltiple	*Fault diagnosis of the 3 tank system using fuzzy multiple inference modelling*	Reino Unido	*López-Toribio et al., 1999*
PCA	*Plant-wide Monitoring of Processes Under Closer-loop Control*	E.U.A.	*Valle-Cervantes, 2001*

Modelos cuantitativos	*A review of process fault detection and diagnosis: Part I: Quantitative model-based methods*	E.U.A.	*Venkatasubramanian et al., 2003*
PCA	*A study on the number of principal components and sensitivity of fault detection using PCA*	Japón	*Tamura et al., 2007*
SVD y Distancia Euclidiana	Nuclear power plant sensor fault detection using singular value decomposition-based method	India	Mandal et al., 2017

1.2. Justificación

La industria química es una industria muy importante a nivel nacional, representó el 2.5% de la industria, según el censo económico 2014 INEGI (2015) (Figura 1.1). Su gran impacto en la vida diaria de las personas se puede constar en todos los productos que se consumen diariamente ya sea en los hogares, en las empresas o en las propias industrias. La industria química provee de las materias primas para la fabricación de otros productos y también puede proporcionar el producto de consumo final.

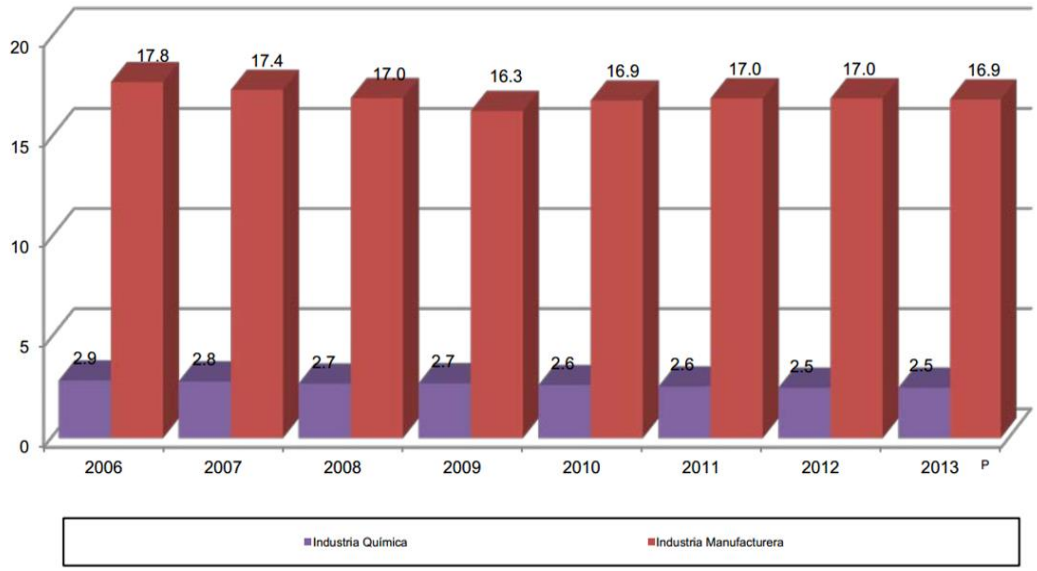


Figura 1.1. Participación del valor agregado bruto en valores básicos de la industria manufacturera y de la industria química en el total de 2006 a 2013, por ciento (INEGI, 2015).

Dentro de estos productos químicos se puede encontrar uno muy importante por su amplia gama de aplicaciones, el formaldehído. La producción a nivel industrial de este producto requiere del empleo de plantas industriales con equipos de gran potencia y volumen, equipos que deben de estar al máximo de capacidad y de eficiencia. La reacción necesaria para la obtención del formaldehído es a través de la oxidación catalítica del metanol (Guerrero *et al*., 2009). La reacción anterior requiere de un proceso muy peligroso por su riesgo de explosión y toxicidad. Por esta razón es importante que los equipos involucrados en el proceso de producción no fallen.

La detección e identificación de fallas, el aislamiento y la reconfiguración (FDI) es un problema muy importante y desafiante en muchas disciplinas, como ingeniería química, ingeniería nuclear, ingeniería aeroespacial y sistemas automotrices (Hwang *et al.*, 2010).

La detección temprana permitirá la predicción de la evolución de la falla y la estimación de la vida útil restante del componente defectuoso antes de que la falla progrese a un estado que ponga en peligro la integridad operacional del sistema. La detección temprana suele ir acompañada de un número inaceptable de falsas alarmas. Un detector de fallas confiable y preciso, por lo tanto, debe ser capaz de declarar una falla solo cuando se alcanza un grado de confianza específico con una tasa de falsa alarma prescrita (Zhang *et al*., 2011).

El principal resultado a obtener es el de las gráficas de contribución que permita detectar en forma oportuna las situaciones anómalas en el sistema. También otro resultado a obtener son los valores singulares de la información proporcionada por la lectura de los sensores y actuadores que se encuentran en la planta dentro del proceso donde se involucra el reactor químico y con esto se puedan identificar las principales variables del sistema.

Este trabajo proporciona a la industria, formas novedosas para mejorar la operación del proceso de producción de formaldehido utilizando la descomposición de valores singulares (SVD) para la identificación de los factores principales que afectan al reactor en la producción de formaldehído.

Impacto ambiental.

Esta investigación busca evitar cualquier perjuicio al medio ambiente, al prevenir la emisión de contaminantes a la atmósfera al permitir tener un mayor control en el proceso de producción de formaldehído, disminuyendo e identificando las fallas en dicho proceso.

Dado que la detección e identificación de fallas (FDI) es un proceso de control y que previene las emisiones de contaminantes a la atmósfera, es congruente con la línea de investigación de *análisis y control de sistemas ambientales*.

1.3. Objetivo general.

Detección e identificación de fallas en un reactor químico heterogéneo para prevenir la emisión de contaminantes a la atmósfera.

1.4. Objetivos específicos

- Proporcionar a la industria formas novedosas para mejorar la operación del proceso de producción de formaldehido utilizando la descomposición de valores singulares (SVD), para la identificación de los factores principales que afectan al reactor en la producción de formaldehído.

- Aplicar la estrategia desarrollada al reactor heterogéneo de la planta química, que permita la disminución de fallos y esto permita a su vez, disminuir cualquier impacto negativo al medio ambiente.

1.5. Descripción del contenido de los capítulos.

En el primer capítulo se hablará de los antecedentes de la detección de fallas utilizando métodos matemáticos, la justificación de la presente tesis y los objetivos.

En el capítulo dos se describirá el formaldehído y su proceso de obtención a nivel industrial, así como algunas de sus aplicaciones, también se explicará la detección de fallas y en que consiste el método de la descomposición de valores singulares.

En el capítulo tres se explicará la metodología utilizada para la detección e identificación de fallas utilizando el análisis de componentes principales, así como los algoritmos utilizados.

En el capítulo cuatro se mostraran los resultados obtenidos de una muestra real de la planta de producción de formaldehído y las fallas detectadas en el periodo muestreado y como esto previene la emisión de contaminantes a la atmósfera.

Por último se mostraran las conclusiones y recomendaciones relativas a la detección e identificación de fallas en el reactor y que mejoras o posible seguimiento se podrá realizar para mejorar esta metodología.

Capítulo 2 Marco teórico.

Formaldehído

El formaldehído se produce por oxidación catalítica de metanol en fase de vapor, siendo uno de los productos químicos más importantes a nivel mundial por su bajo costo, alta pureza y la amplia gama de aplicaciones, en especial para la producción de resinas termoestables, principalmente resinas fenólicas, de urea, y melamina que son utilizadas como adhesivos y aglutinantes en la industria (Guerrero *et al*., 2009).

Algunos usos del formaldehído

La International Programme on Chemical Safety (IPCS, 1991), enumera algunos usos importantes del formaldehído:

- Nutrición animal y agricultura.
- Aminoplásticos (resinas de urea formaldehído y resinas de formaldehído de melamina).
- Plásticos fenólicos (resinas de fenol formaldehído).
- Polioximetileno (plásticos de poliacetal).
- Intermedio químico.
- Productos farmacéuticos.
- Cosméticos.
- Otros bienes de consumo.

Proceso de producción de formaldehído en un reactor químico heterogéneo.

Hoy en día existen varios procesos industriales autorizados para convertir el metanol al formaldehído. Todos ellos son procesos catalíticos heterogéneos, basados en catalizadores de plata y / o catalizadores de hierro – molibdeno (Figura 2.1).

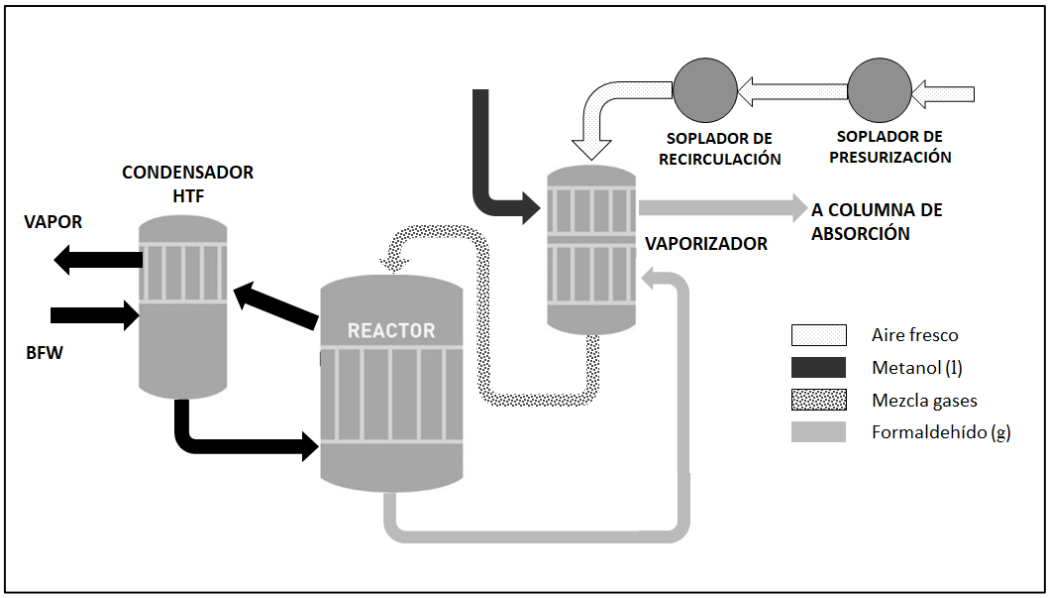


Figura 2.1. Diagrama simplificado del proceso en una planta de producción de formaldehído, elaboración propia basada en (Soares et al., 2005).

En catalizadores de plata el proceso de producción de formaldehído es producido por la oxidación de metanol y la des hidrogenación de metanol. Ninguna de las reacciones es dominante y el proceso global es exotérmico (Soares e*t al.*, 2005).

$$CH_3OH + \frac{1}{2}O_2 \rightarrow CH_2O + H_2O \qquad \Delta H = -157\,{}^{kJ}/_{mol} \qquad (2.1)$$

Sobre los catalizadores de molibdato de hierro, el formaldehído se produce solo por oxidación con metanol. En el proceso de estos catalizadores, se usa un exceso de aire para asegurar una conversión casi completa y evitar los límites explosivos de metanol (6.7-36.5% vol., en aire). La temperatura de reacción debe ser inferior

a 400 ° C para asegurar la estabilidad del catalizador y limitar las reacciones secundarias. Para los catalizadores de Mo-Fe-O, el rendimiento de formaldehído puede ser tan alto como 95% y la conversión tan alta como 98-99%. La vida media de un catalizador industrial es de aproximadamente 6 a 12 meses (Soares e*t al.*, 2005).

Detección e identificación de fallas (FDI):

El propósito fundamental de un sistema FDI es generar una alarma cuando ocurre una falla y de ser posible aislarla y estimar la magnitud de la falla e indicar el sensor o actuador con falla (Tudón *et al*., 2011).

Dentro del control automático de los sistemas técnicos, las funciones de supervisión sirven para indicar estados de proceso no deseados o no permitidos, y para tomar las medidas apropiadas a fin de mantener la operación y evitar daños o accidentes. Se pueden distinguir las siguientes funciones (Isermann, 2006):

1. *monitoreo:* las variables medibles se verifican con respecto a las tolerancias, y las alarmas se generan para el operador;
2. *protección automática:* en el caso de un estado de proceso peligroso, la función de supervisión inicia automáticamente una respuesta adecuada;
3. *supervisión con diagnóstico de fallas:* en función de las variables medidas, se calculan las características, los síntomas se generan a través de la detección de cambios, se realiza un diagnóstico de fallas y se toman decisiones para contrarrestarlas.

La gran ventaja de los métodos de supervisión basados en el valor límite clásico 1) y 2) es su simplicidad y fiabilidad. Sin embargo, solo pueden reaccionar después de un cambio relativamente grande de una característica, es decir, después de

una falla repentina grande o una falla de aumento gradual y duradero. Además, generalmente no es posible realizar un diagnóstico de fallas en profundidad. Por lo tanto, se necesitan métodos avanzados de supervisión y diagnóstico de fallas que satisfagan los siguientes requisitos (Isermann, 2005):

i. Detección temprana de fallas pequeñas con comportamiento temporal incipiente o abrupto;
ii. Diagnóstico de fallas en el actuador, componentes del proceso o sensores;
iii. Detección de fallas en circuitos cerrados;
iv. Supervisión de procesos en estados transitorios.

Isermann (2005), presenta un estudio general de los métodos de supervisión, detección de fallas y diagnóstico. En el siguiente modelo se consideran los métodos de detección de fallas, que permiten una visión profunda del comportamiento del proceso.

Métodos de detección de fallos basados en modelos.

La tarea consiste en la detección de fallas en los procesos, actuadores y sensores mediante el uso de las dependencias entre diferentes señales medibles. Estas dependencias se expresan mediante modelos de procesos matemáticos.

La Figura 2.2 muestra la estructura básica de la detección de fallas basada en modelos. En base a las señales de entrada medidas $\boldsymbol{U}$ y las señales de salida $\boldsymbol{Y}$, los métodos de detección generan residuos $\boldsymbol{r}$, estimaciones de parámetros $\boldsymbol{\Theta}$ o estimaciones de estado $\hat{\boldsymbol{x}}$, que se llaman características. En comparación con las características normales, se detectan cambios de características que conducen a síntomas analíticos $\boldsymbol{S}$ (Isermann, 2005).

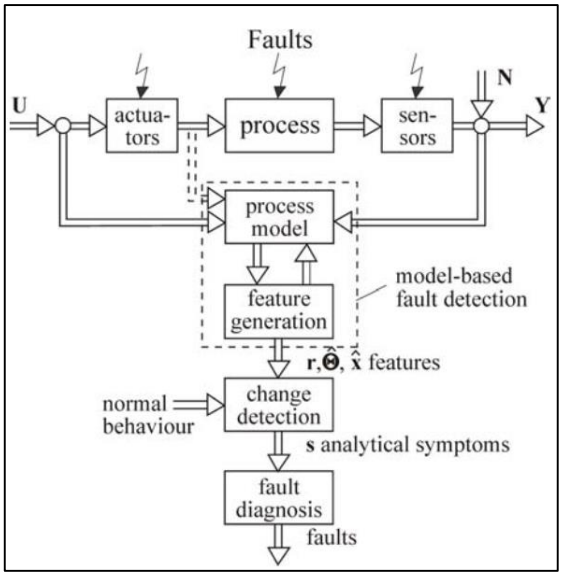


Figura 2.2. Esquema general de detección y diagnóstico de fallas basado en el modelo de proceso (Isermann, 2005).

Isermann (2005), enumera los métodos basados en modelos para la FDI:

1. Modelos de proceso y modelado de fallas.
2. Detección de fallas con estimación de parámetros
3. Detección de fallas con observadores
4. Detección de fallas con ecuaciones de paridad.
5. Detección de fallas con modelos de señal

Descomposición de valores singulares (SVD)

La descomposición en valores singulares de una matriz fue propuesta en forma independiente por Beltrami en 1873 y Jordan en 1874. Su generalización a espacios de dimensión infinita fue desarrollada en el contexto de ecuaciones integrales por Smith (1907) y Weyl (1912). Sin embargo, esta factorización no fue

popular hasta fines de la década del 60, cuando Golub y otros autores mostraron cómo calcularla numéricamente y usarla como herramienta para muchos algoritmos estables (Golub *et al.*, 1989) (Trefethen *et al.*, 1997) (Buffo *et al.,* 2015).

Sea A una matriz de $m \times n$ con rango $\boldsymbol{r}$. Entonces existe una matriz Σ de tamaño $m \times n$, donde las entradas diagonales de D son los $\boldsymbol{r}$ primeros valores singulares de A, y existe una matriz U ortogonal $m \times m$ y una matriz V ortogonal $n \times n$, tales que (Lay, 1999)

$$A = U\Sigma V^T \tag{2.2}$$

Aplicaciones de la SVD.

Lay (1999), enumera algunas aplicaciones de la descomposición de valores singulares:

1) Cálculo del rango.
2) Cálculo de la pseudoinversa.
3) Procesamiento de imágenes.
4) Estadística.
5) Cálculo del número de condición.

Análisis de Componentes Principales (PCA):

El Análisis de Componentes Principales (PCA) es una técnica usada ampliamente por investigadores para el monitoreo de procesos, el cual fue introducido por Pearson (1901) y desarrollado por Hotelling (1993) (Jolliffe, 1986). Cuando solo se las variables de procesos están disponibles, la técnica del PCA es una mejor

opción para el monitoreo de procesos y la detección de fallas (Valle-Cervantes, 2001).

El PCA es una técnica de análisis multivariable (Harrou et al., 2013). Está basada en la descomposición de la matriz de covarianza o de correlación de los datos (Valle-Cervantes, 2001). Intenta encontrar las combinaciones lineales de las variables que maximizan la varianza de los datos (componentes) (Valle-Cervantes, 2001). Intenta encontrar las combinaciones lineales de las variables que maximizan la varianza de los datos (componentes) (Valle-Cervantes, 2001).

Capítulo 3 Metodología.

Obtención de datos:

Se deben de obtener los datos de las variables de la planta mediante la descarga de los reportes que proporciona el sistema de control y monitoreo, la información será de los registros diarios de todas las variables involucradas (Flujo, presión, niveles, temperatura, análisis y misceláneas) en el proceso de producción del formaldehído. Los reportes tienen un formato de hoja de cálculo y texto delimitado.

Pre-procesamiento de los datos:

Utilizando el software de Matlab® se importaran los datos obtenidos previamente del sistema de monitoreo de la planta con la ayuda de la interfaz desarrollada para este proceso. En la importación se seleccionan únicamente las variables del sistema del reactor previamente definidas y se guardan en una matriz en formato de Matlab® (.m), lo que permite realizar los cálculos y operaciones necesarios para la FDI, en una forma más eficiente y rápida.

El índice Kaiser-Meyer-Olkin (KMO) se utiliza para determinar la correlación entre las variables del sistema del reactor y si son lo suficientemente grandes como para aplicar la técnica de PCA. El índice KMO (Tabla 3.1) proporciona una escala cuando se aplica al conjunto de datos, si el valor obtenido es cercano a 1, las variables están más correlacionadas, por el contrario, si el valor obtenido es cercano a 0, la técnica PCA no puede usarse (Kaiser, 1974) (Kassouf et al., 2018).

Tabla 3.1. Índice KMO para un conjunto de datos correlacionados.

Índice	Grado de variación común
0.9 - 1	*Excelente*
0.8 - 0.89	*Muy buena*
0.7 - 0.79	*Buena*
0.6 - 0.69	*Regular*
0.5 - 0.59	*Mala*
> 0.5	*Inaceptable*

Centrado y escalado de datos:

Los datos registrados por el sistema de la planta contienen diferentes tipos de unidades lo que complica la aplicación del método de PCA, por ser un método basado en la variabilidad de los datos. El centrado de datos consiste en ajustar los datos a valores cercanos a cero, utilizando la media como base del centrado restándola de cada punto de datos, tal que (van den Berg *et al*., 2006).

$$x_{ij} = x_{ij} - \overline{x_i} \qquad (3.1)$$

El escalado de datos se realiza dividiendo las variables por un factor, el factor de escala, que permite normalizar las unidades a una sola. El método de escalado que se utiliza es el de la desviación estándar utilizada como factor de escala, tal que (van den Berg *et al*., 2006).

$$\tilde{x}_{ij} = \frac{x_{ij} - \overline{x_i}}{\sigma_i} \qquad (3.2)$$

Generar el modelo PCA:

Validación cruzada.

Con la eigen-descomposición, se procede a generar el modelo PCA de la falla, pero primero se debe seleccionar la cantidad de componentes que se van a retener (Bro *et al*., 2008) (Josse *et al.,* 2012), para esto, se utiliza la técnica conocida como validación cruzada, método de Prediction Residual Error Square Sum (PRESS), se calcula la matriz de predicción de error como sigue (Diana *et al.,* 2002):

$$E_t^{(k)} = E_t^{(k-1)} - \tilde{\psi}_{ik}^{(t)} \tilde{\gamma}_{jk}^{(t)} = \left| e_{ij}^{(k)} \right| \qquad (3.3)$$

donde (i,j) son tales que $x_{i,j} \in E_t$

$$PRESS_t(k) = \sum_{\{i,j:x_{i,j} \in E_t\}} \left| e_{i,j}^{(k)} \right|^2 \qquad (3.4)$$

Selección de componentes.

Utilizando una técnica de detección e identificación de fallas (FDI) (Tudón *et al*., 2011; Sanchez-Fernández *et al*., 2015) se genera el modelo, seleccionando los principales componentes con base en la variación capturada por cada componente y aplicando la distribución *T^2 de Hotelling*, conjuntamente con otras técnicas estadísticas como el Error Cuadrático de Predicción (*SPE ó Q*) (Sanchez-Fernández *et al*., 2015) , se obtiene el modelo del sistema del reactor catalítico en un sistema ideal.

Comparación de modelo con muestra.

Se selecciona una muestra ($\mathbf{M}$) con los datos registrados durante un mes en el sistema de reactor catalítico para compararse con el modelo generado. Los datos $\mathbf{M}$ tienen que tener un pre-procesamiento y pre-tratamiento (van den Berg *et al.*, 2006) (escalado y centrado de datos) utilizando las medias y desviaciones estándar del modelo PCA. Se calculan los valores de *T^2 de Hotelling y Q-residual* para la muestra (Mujica *et al.*, 2010), se grafican los valores de cada valor

estadístico tiendo como límite los valores del modelo. Las valores en las muestras que superen dicho límite, representan las fallas en el sistema del reactor (Sanchez-Fernández *et al.*, 2015; Hu *et al*., 2016).

Elaboración de las gráficas de contribución.

En este paso se deben identificar las variables que más contribuyen con el sistema (matriz $\mathbf{C}$), para conocer principalmente los comportamientos anormales de dichas variables y lograr una FDI en el sistema del reactor que permita la toma de decisiones con anticipación (Scheible 1995).

$$\mathbf{C}_j = \left(\mathbf{N}_{kd,j} - \mathbf{P}_{kd,j}\right)^2 \qquad (3.5)$$

donde $\mathbf{N}$ es la matriz de datos ajustados y centrados, $\mathbf{P}$ la matriz de predicción obtenida con el producto de la matriz de puntuaciones y cargas del modelo PCA, kd la posición *í-ésima* del elemento de la matriz de residuos mayores a el valor de $\mathbf{Q}$.

Algoritmo.

En la Figura 3.1 se muestra el algoritmo de la metodología con los pasos necesarios para la generación del modelo PCA del reactor químico con los datos de la planta en operación normal. En la Figura 3.2 se muestra el algoritmo para la detección e identificación de fallas en los datos de la muestras también obtenidos del reactor químico y comparados con el modelo PCA.

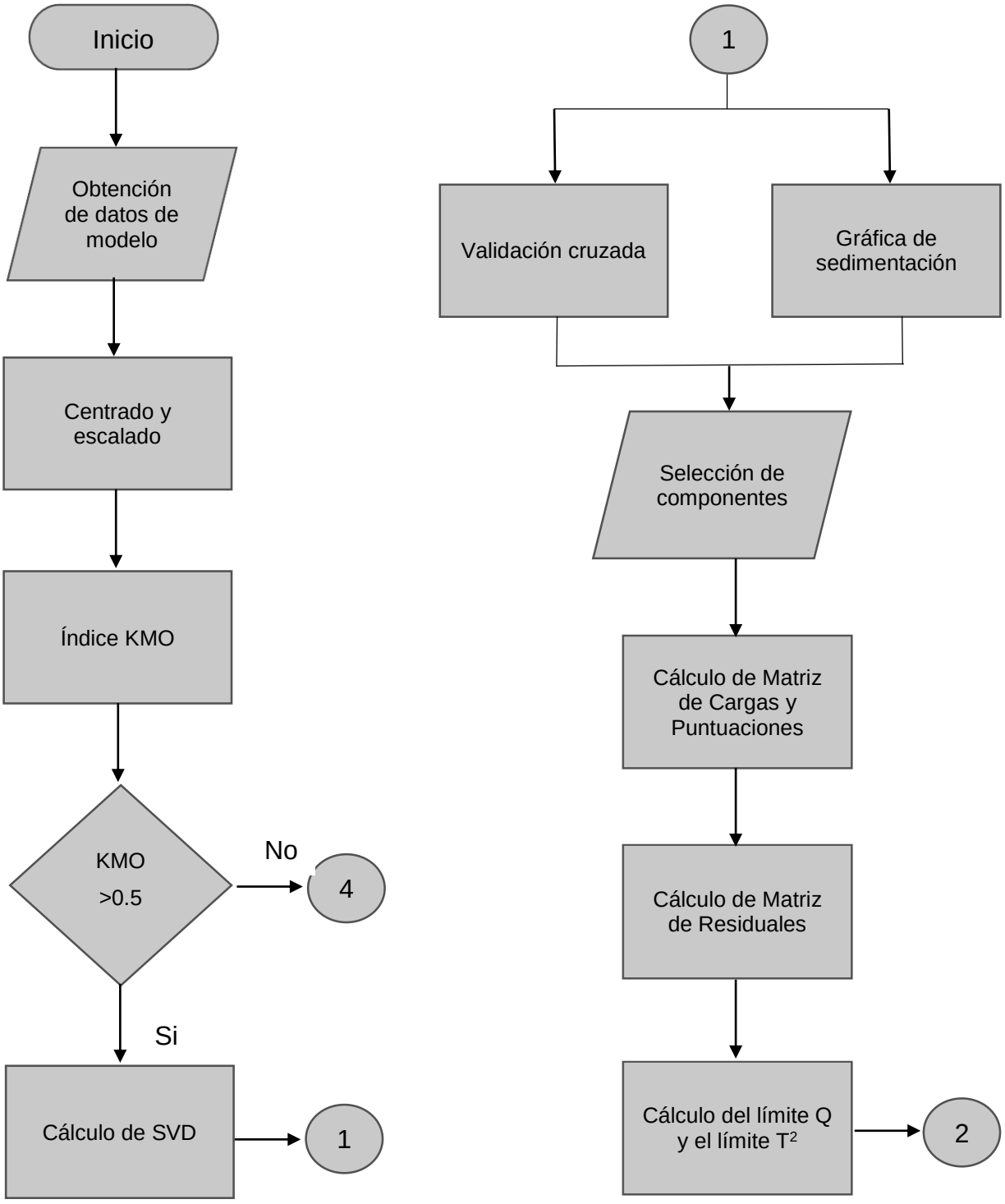


Figura 3.1. Algoritmo para el cálculo del modelo PCA del reactor.

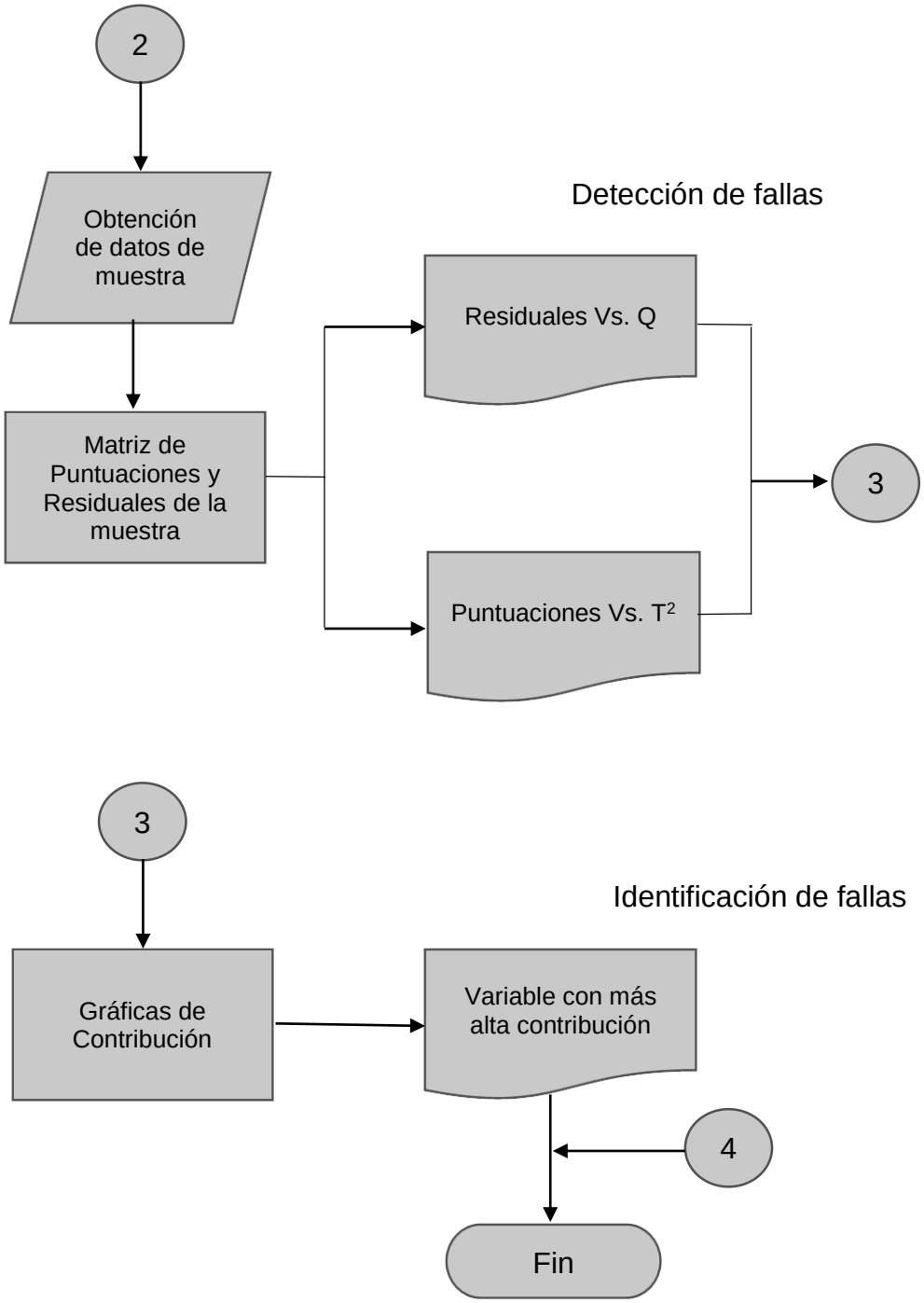


Figura 3.2. Algoritmo para la detección e identificación de fallas en la muestra, basado en el modelo PCA generado.

Capítulo 4 Resultados y discusión.

Datos obtenidos:

El sistema automatizado de la planta, tiene una interface que permite obtener toda la información de los sensores y actuadores. La interface gráfica permite el control e inspección visual de todo el sistema, también se pueden generar reportes detallados de las actividades de todo el sistema (sensores y válvulas) (Figura 4.1). Es en esta parte, donde se obtuvo la información requerida para el proyecto.

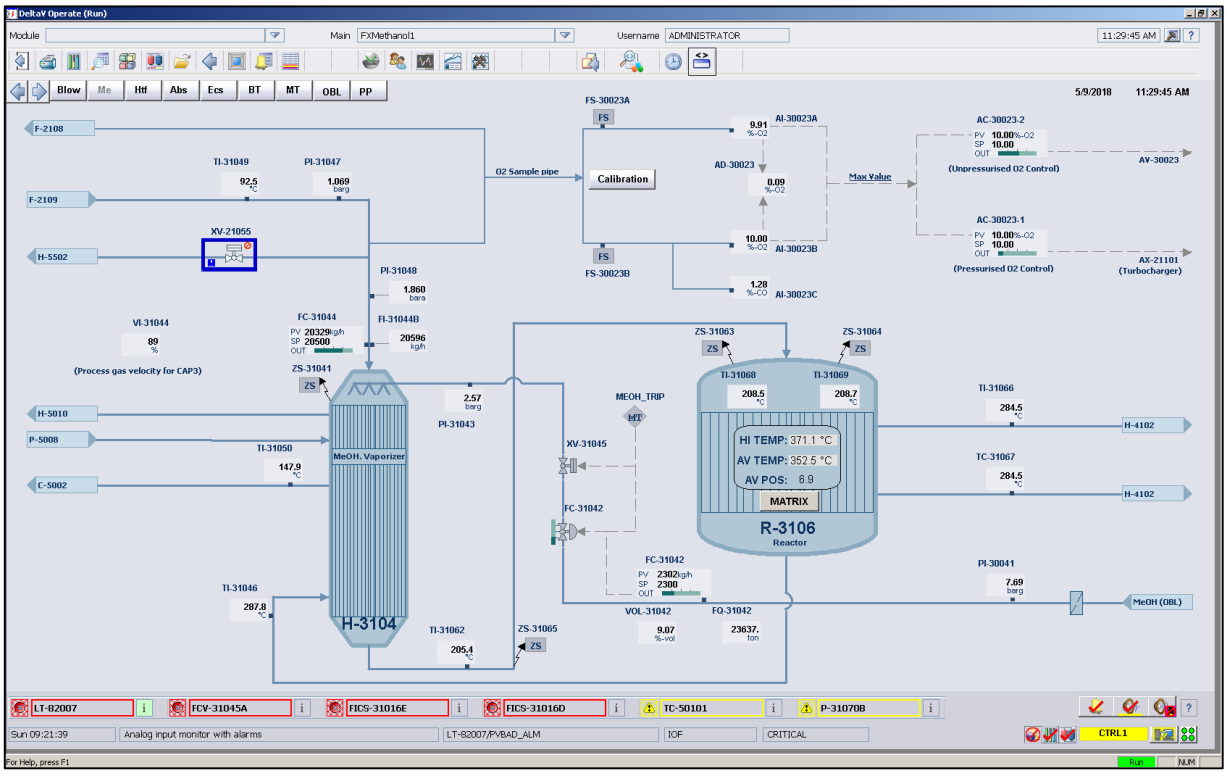


Figura 4.1. Captura de pantalla tomada del sistema de control (D.R. MASISA, 2018)

Se obtuvieron los registros del sistema de control del reactor de los meses de noviembre y diciembre de 2018, los cuales fueron registrados en forma diaria en ciclos de 4 horas (6 registros por día) siendo un total de 366 registros de 220 variables que

conforman la planta de producción de formaldehído. Para la muestra se seleccionaron 186 registros correspondientes al mes de enero del 2019 (Tabla 4.1)

Tabla 4.1. Registros utilizados para la generación del modelo PCA y para la muestra.

Modelo PCA		**Muestra**	
Mes	**Total de Registros**	**Mes**	**Total de Registros**
Noviembre 2018	180	Enero 2019	186
Diciembre 2018	186		
Total	**366**	**Total**	**186**

Con el primer pre-tratamiento de datos se depuraron las variables de la planta quedando solo 58 variables (Tabla 4.2 y Tabla 4.3) que conforman solo el sistema del reactor (soplantes, flujo de metanol, condensador, reactor, etc.).

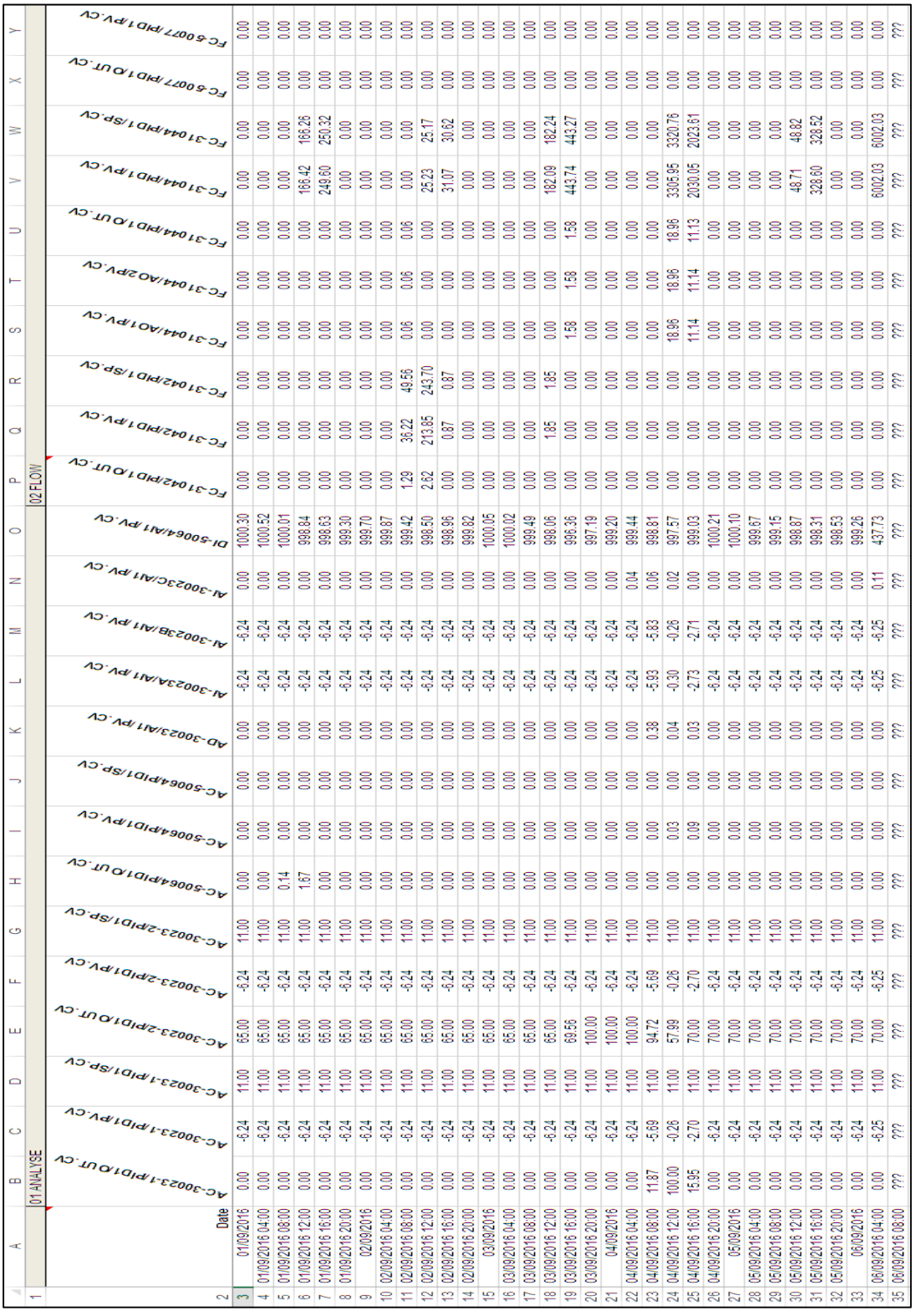
Tabla 4.2. Selección de variables de la planta de producción de formaldehído.

Variables de la planta de producción de formaldehído	**220**
Variables del Sistema del Reactor	**58**

El sistema de la planta permite obtener reportes mensuales de todo el sistema con información registrada cada 4 horas en un turno de 24 horas por cada día en que la planta esté produciendo, dichos reportes se exportaron en formato de hoja de cálculo (Figura 4.2). Después de la obtención de los reportes, se procedió al análisis y depuración de los datos. En esta parte es dónde comenzó el proceso utilizando el software Matlab®.

Tabla 4.3. Variables utilizadas del reactor de la planta de producción de formaldehído.

No.	Variable	Descripción	Group	System	Trip
1	**AC-30023-1/PID1/PV.CV**	Valvula del control de presion del oxígeno	01 ANALYSE	Reactor / Sopladores	Plant trip
2	**AC-30023-2/PID1/PV.CV**	Valvula del control de presion del oxígeno de entrada de aire fresco	01 ANALYSE	Reactor / Sopladores	Plant trip
3	**AD-30023/AI1/PV.CV**	Analizador de diferencial de oxígeno	01 ANALYSE	Reactor	Plant trip
4	**AI-30023A/AI1/PV.CV**	Contenido de oxígeno Bajo / Alto	01 ANALYSE	Reactor	MeOH trip
5	**AI-30023B/AI1/PV.CV**	Contenido de oxígeno Bajo / Alto	01 ANALYSE	Reactor	MeOH trip
6	**AI-30023C/AI1/PV.CV**	Contenido de monóxido de carbono Bajo / Alto	02 FLOW	Reactor	Plant trip
7	**FC-31042/PID1/PV.CV**	Flujo de MeOH	02 FLOW	Reactor	MeOH trip
8	**FC-31044/PID1/PV.CV**	Flujo de gas de proceso	02 FLOW	Reactor / Sopladores	MeOH trip
9	**FI-31044B/AI1/PV.CV**	Flujo de gas de proceso	02 FLOW	Reactor	MeOH trip
10	**FI-40081/AI1/PV.CV**	Flujo HTF	02 FLOW	Condensador	Plant trip
11	**FI-92044/AI1/PV.CV**	Flujo BFW - condensador HTF	03 FLOWTOTAL	Condensador	Plant trip
12	**LC-41023/PID1/PV.CV**	Nivel de BFW en el condensador HTF	04 LEVEL	Condensador	MeOH trip
13	**LI-40042/AI1/PV.CV**	Indicador de nivel de tanque de HTF	04 LEVEL	Condensador	Plant trip
14	**LI-41029/AI1/PV.CV**	Nivel del condensador HTF	05 PRESSURE	Condensador	MeOH trip
15	**PC-21364/PID1/PV.CV**	Valvula de la presion de lubricacion de TC	05 PRESSURE	Sopladores	Plant trip
16	**PC-41026/PID1/PV.CV**	Presión condensador HTF	05 PRESSURE	Condensador	Blower trip
17	**PC-55022/PID1/PV.CV**	Sistema presurizado Bajo / Alto	05 PRESSURE	Sopladores	Blower trip
18	**PI-21024/AI1/PV.CV**	Indicador de presion de entrada de aire freso al TC	05 PRESSURE	Sopladores	Plant trip
19	**PI-21042/AI1/PV.CV**	Indicador de presion de aire de salida del TC	05 PRESSURE	Sopladores	Plant trip
20	**PI-30041/AI1/PV.CV**	Presion de metanol despues del filtro antes del vaporizador (after strain)	05 PRESSURE	Reactor	Plant trip
21	**PI-31043/AI1/PV.CV**	Presion de metanol antes del vaporizador	05 PRESSURE	Reactor	Plant trip
22	**PI-31047/AI1/PV.CV**	Presion del gas de proceso a la entrada del vaporizador	05 PRESSURE	Reactor / Sopladores	Plant trip
23	**PI-31048/AI1/PV.CV**	Presion del gas de proceso a la entrada del vaporizador despues de las recirculaciones	05 PRESSURE	Reactor / Sopladores	Plant trip
24	**PI-55024/AI1/PV.CV**	Presión de la columna de absorción	05 PRESSURE	Sopladores	Blower trip
25	**PI-55066/AI1/PV.CV**	Presion del gas de proceso antes del TC	05 PRESSURE	Sopladores	Plant trip
26	**PI-55067/AI1/PV.CV**	Presion del gas de proceso despues del TC	05 PRESSURE	Sopladores	Plant trip
27	**PI-80004/AI1/PV.CV**	Presion ambiental dentro del cuarto de soplantes	05 PRESSURE	Sopladores	Plant trip
28	**PI-92048/AI1/PV.CV**	Indicador de presion de entrada al condensador BFW	06 TEMP1	Condensador	Plant trip
29	**TC-21321/ALM1/PV.CV**	Temperatura del aceite de lubricante del TC	06 TEMP1	Sopladores	Plant trip
30	**TC-31067/ALM1/PV.CV**	Temperatura del vapor HTF	06 TEMP1	Reactor / Condensador	MeOH trip
31	**TC-40083/ALM1/PV.CV**	Temperatura de HTF a la salida de los calentadores	06 TEMP1	Reactor / Condensador	Plant trip
32	**TI-21043/AI1/PV.CV**	Temperatura de aire a la salida del turbocargador	06 TEMP1	Sopladores	Plant trip
33	**TI-21363/AI1/PV.CV**	Temperatura del aceite lubricante del TC	06 TEMP1	Sopladores	Plant trip
34	**TI-31046/AI1/PV.CV**	Temperatura PG después del reactor FA	06 TEMP1	Reactor	Blower trip
35	**TI-31049/AI1/PV.CV**	Temperatura del gas de proceso despues de soplantes	07 TEMP2	Reactor	Plant trip
36	**TI-31050/AI1/PV.CV**	Tempreatura del PG antes del Absorbedor	07 TEMP2	Reactor	Blower trip
37	**TI-31062/AI1/PV.CV**	Temperatura del PG antes del reactor FA	07 TEMP2	Reactor	Blower trip
38	**TI-31066/AI1/PV.CV**	Temperatura del vapor HTF	07 TEMP2	Reactor / Condensador	MeOH trip
39	**TI-31068/AI1/PV.CV**	Temperatura del Reactor alto	07 TEMP2	Reactor	Blower trip
40	**TI-31069/AI1/PV.CV**	Temperatura del Reactor alto	07 TEMP2	Reactor	Blower trip
41	**TI-40041/AI1/PV.CV**	Temperatura del tanque de almacenamiento HTF	07 TEMP2	Condensador	Plant trip
42	**TI-40082A/AI1/PV.CV**	Temperatura de calentadores de HTF	07 TEMP2	Condensador	Plant trip
43	**TI-40082B/AI1/PV.CV**	Temperatura de calentadores de HTF	07 TEMP2	Condensador	Plant trip
44	**TI-40082C/AI1/PV.CV**	Temperatura de calentadores de HTF	07 TEMP2	Condensador	Plant trip
45	**TI-41021/AI1/PV.CV**	Temperatura HTF dentro condensador baja	07 TEMP2	Condensador	Plant trip
46	**TI-41025/AI1/PV.CV**	High temp HTF condensor top	07 TEMP2	Condensador	MeOH trip
47	**TI-50091/AI1/PV.CV**	Temperatura de aire de recirculacion del adsorbedor a los sopladores	07 TEMP2	Reactor	Plant trip
48	**TI-55064/AI1/PV.CV**	Temperatura después de TC	07 TEMP2	Reactor	Blower trip
49	**TI-55065/AI1/PV.CV**	Temperatura después de la cama catlítica	07 TEMP2	Reactor	Blower trip
50	**TI-80001/AI1/PV.CV**	Temperatura del cuarto de soplantes	08 MISC	Sopladores	Plant trip
51	**JI-31044A/AI1/PV.CV**	Consumo de soplante F-2108	08 MISC	Sopladores	Plant trip
52	**JI-31044B/AI1/PV.CV**	Consumo de soplante F-2109	08 MISC	Sopladores	Plant trip
53	**SI-31044A/AI1/PV.CV**	Revoluciones por minuto de F-2108	08 MISC	Sopladores	Plant trip
54	**SI-31044B/AI1/PV.CV**	Revoluciones por minuto de F-2109	08 MISC	Sopladores	Plant trip
55	**VOL-31042/AI1/PV.CV**	Inlet MeOH	08 MISC	Reactor	MeOH trip
56	**XI-21084/AI1/PV.CV**	Vibracion de soplante F-2108	08 MISC	Sopladores	Plant trip
57	**XI-21094/AI1/PV.CV**	Vibracion de soplante F-2109	08 MISC	Sopladores	Plant trip
58	**XI-21103/AI1/PV.CV**	Torque del TC	08 MISC	Sopladores	Plant trip



Date	01 ANALYSE																02 FLOW												
	AC-3002 3-1/PID1/OUT.CV	AC-3002 3-1/PID1/PV.CV	AC-3002 3-1/PID1/SP.CV	AC-3002 3-2/PID1/OUT.CV	AC-3002 3-2/PID1/PV.CV	AC-3002 3-2/PID1/SP.CV	AC-5006 4/PID1/OUT.CV	AC-5006 4/PID1/PV.CV	AC-5006 4/PID1/SP.CV	AD-3002 3/AI1/PV.CV	AI-3002 3A/AI1/PV.CV	AI-3002 3B/AI1/PV.CV	AI-3002 3C/AI1/PV.CV	DI-5006 4/AI1/PV.CV	FC-3104 2/PID1/OUT.CV	FC-3104 2/PID1/PV.CV	FC-3104 2/PID1/SP.CV	FC-3104 4/AO1/PV.CV	FC-3104 4/AO2/PV.CV	FC-3104 4/PID1/OUT.CV	FC-3104 4/PID1/PV.CV	FC-3104 4/PID1/SP.CV	FC-5007 7/PID1/OUT.CV	FC-5007 7/PID1/PV.CV
01/09/2016	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	1000.30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01/09/2016 04:00	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	1000.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01/09/2016 08:00	0.00	-6.24	11.00	65.00	-6.24	11.00	0.14	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	1000.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01/09/2016 12:00	0.00	-6.24	11.00	65.00	-6.24	11.00	1.67	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	998.84	0.00	0.00	0.00	0.00	0.00	0.00	0.00	166.42	166.26	0.00	0.00	0.00	0.00	0.00
01/09/2016 16:00	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	998.63	0.00	0.00	0.00	0.00	0.00	0.00	0.00	249.60	250.32	0.00	0.00	0.00	0.00	0.00
01/09/2016 20:00	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	999.30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
02/09/2016	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	999.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
02/09/2016 04:00	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	999.87	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
02/09/2016 08:00	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	999.42	1.29	36.22	49.56	0.06	0.06	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
02/09/2016 12:00	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	998.50	2.62	213.85	243.70	0.00	0.00	0.00	0.00	25.23	25.17	0.00	0.00	0.00	0.00	0.00
02/09/2016 16:00	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	998.96	0.00	0.87	0.87	0.00	0.00	0.00	0.00	31.07	30.62	0.00	0.00	0.00	0.00	0.00
02/09/2016 20:00	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	999.82	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
03/09/2016	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	1000.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
03/09/2016 04:00	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	1000.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
03/09/2016 08:00	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	999.49	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
03/09/2016 12:00	0.00	-6.24	11.00	65.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	998.06	0.00	1.85	1.85	0.00	0.00	0.00	0.00	182.09	182.24	0.00	0.00	0.00	0.00	0.00
03/09/2016 16:00	0.00	-6.24	11.00	69.56	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	996.36	0.00	0.00	1.58	1.58	1.58	1.58	1.58	443.74	443.27	0.00	0.00	0.00	0.00	0.00
03/09/2016 20:00	0.00	-6.24	11.00	100.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	997.19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
04/09/2016	0.00	-6.24	11.00	100.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.00	999.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
04/09/2016 04:00	0.00	-6.24	11.00	100.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	-6.24	0.04	999.44	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
04/09/2016 08:00	11.87	-5.69	11.00	94.72	-5.69	11.00	0.00	0.00	0.00	0.38	-5.93	-5.83	0.06	998.81	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
04/09/2016 12:00	100.00	-0.26	11.00	57.99	-0.26	11.00	0.00	0.03	0.00	0.04	-0.30	-0.26	0.02	997.57	0.00	0.00	0.00	18.96	18.96	18.96	18.96	18.96	3305.95	3320.76	0.00	0.00	0.00	0.00	0.00
04/09/2016 16:00	16.95	-2.70	11.00	70.00	-2.70	11.00	0.00	0.09	0.00	0.03	-2.73	-2.71	0.00	999.03	0.00	0.00	0.00	11.14	11.14	11.14	11.14	11.13	2030.05	2023.61	0.00	0.00	0.00	0.00	0.00
04/09/2016 20:00	0.00	-6.24	11.00	70.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	0.00	1000.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
05/09/2016	0.00	-6.24	11.00	70.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	0.00	1000.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
05/09/2016 04:00	0.00	-6.24	11.00	70.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	0.00	999.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
05/09/2016 08:00	0.00	-6.24	11.00	70.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	0.00	999.15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
05/09/2016 12:00	0.00	-6.24	11.00	70.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	0.00	998.87	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	48.71	48.82	0.00	0.00	0.00	0.00	0.00
05/09/2016 16:00	0.00	-6.24	11.00	70.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	0.00	998.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	328.60	328.52	0.00	0.00	0.00	0.00	0.00
05/09/2016 20:00	0.00	-6.24	11.00	70.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	0.00	998.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
06/09/2016	0.00	-6.24	11.00	70.00	-6.24	11.00	0.00	0.00	0.00	0.00	-6.24	-6.24	0.00	999.26	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
06/09/2016 04:00	0.00	-6.25	11.00	70.00	-6.25	11.00	0.00	0.00	0.00	0.00	-6.25	-6.25	0.11	437.73	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6002.03	6002.03	0.00	0.00	0.00	0.00	0.00
06/09/2016 08:00	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???	???

Figura 4.2 Reporte mensual generado por el sistema de control de la planta de formaldehído (MASISA, 2018).

Importación de los datos obtenidos.

Se desarrolló una interface en Matlab® que facilita la realización de los procesos de depurado y análisis de datos (Figura 4.3) (ver Anexo1).

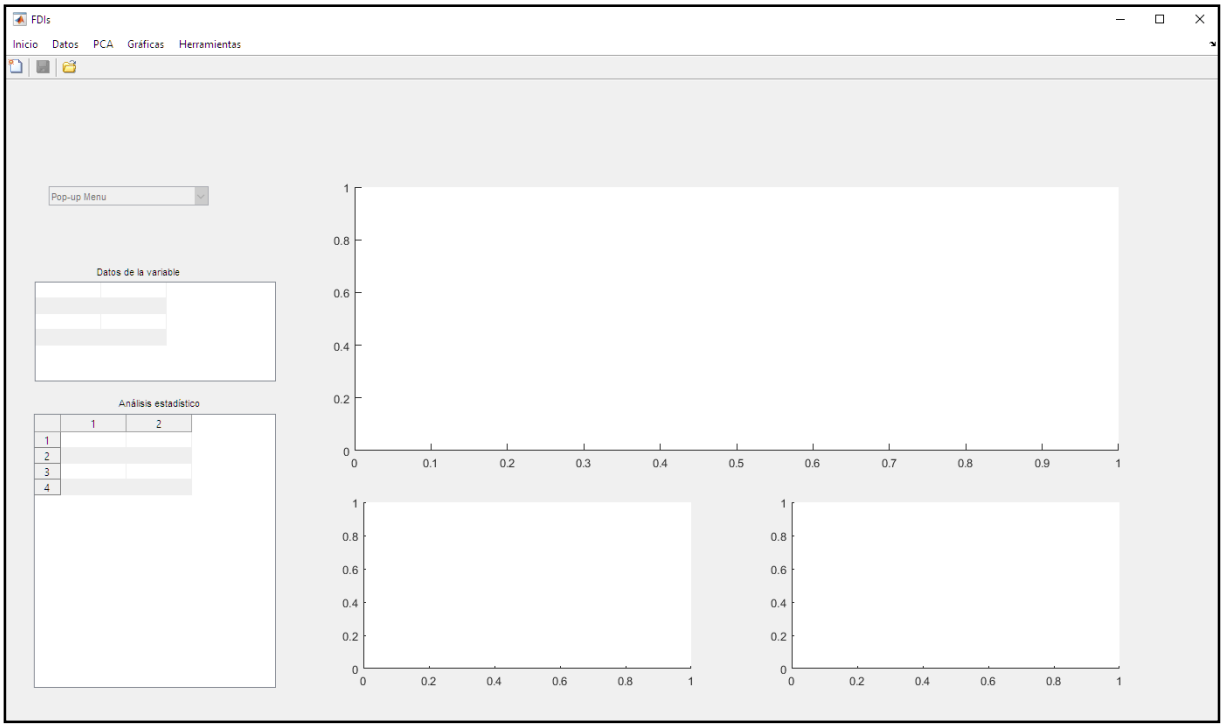


Figura 4.3 Interface para la detección e identificación de fallas basado en PCA.

Con los datos recolectados en la planta de formaldehído, se utilizó la interface que permitió la interacción con la información recabada en la planta (reportes mensuales). Como la información generada por el sistema de la planta incluye todas las variables del sistema, se realizó una selección preliminar de las variables a considerar solamente para el proyecto. Ésta selección fue hecha con base en conversaciones con ingenieros y operadores de la planta y de nuestro sistema de interés: el sistema de reactor de la planta.

Identificadas las variables que se requieren para el análisis y depuración de datos, se procedió a ejecutar el algoritmo que permitió importar únicamente las variables requeridas, que se encuentran dentro del reporte. Esto se hizo a través de una opción dentro de la interface desarrollada que permite seleccionar uno o más archivos para importar al ambiente de trabajo dentro de la interface. Se almacenó la matriz importada dentro de un archivo para su posterior uso.

Prueba Kaiser-Meyer-Olkin (KMO).

Se aplicó la prueba KMO al conjunto de datos recolectados para conocer si estaban los suficientemente correlacionados para utilizarse la metodología de PCA. El resultado se muestra en la Tabla 4.4.

Tabla 4.4. Resultados de la medida de adecuación muestral a los datos para el modelo.

Total de registros	Total de variables	Resultado	**Escala** (Kaiser, 1974)
58	**366**	0.9236	La prueba de KMO produce un grado de variación común excelente.

Ajuste de datos.

Se procedió al pre-procesamiento y pre-tratamiento de los datos, en primer lugar con los datos del modelo, ya que con los valores de la media y desviación estándar de los datos del modelo se realizó el ajuste y centrado de los datos de la muestra, los datos están expresados en diferentes unidades como se muestra en la Figura 4.5 (datos modelo) y en la Figura 4.5 (datos muestra). Se realizó correctamente este ajuste y centrado de los datos para eliminar el error de unidades (Figura 4.6 y Figura 4.7), los rangos donde se ajustaron los datos se muestran en la

Tabla 4.5.

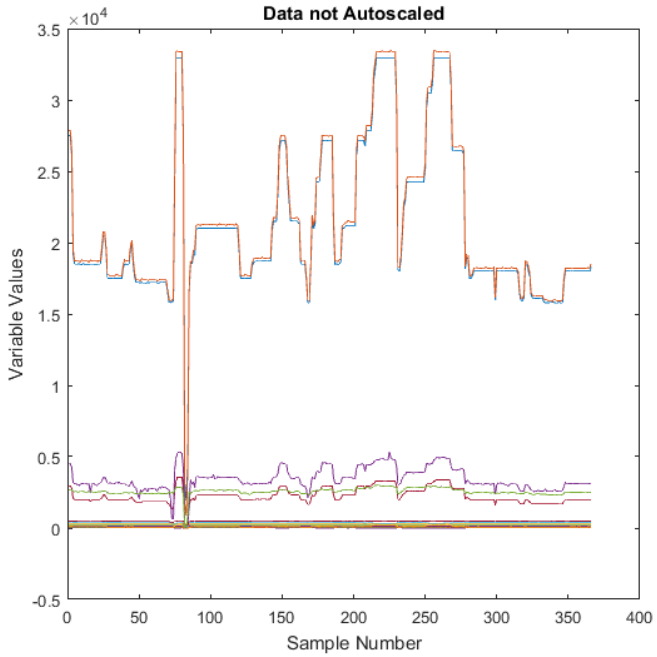


Figura 4.4. Datos para el modelo antes del pre-procesamiento.

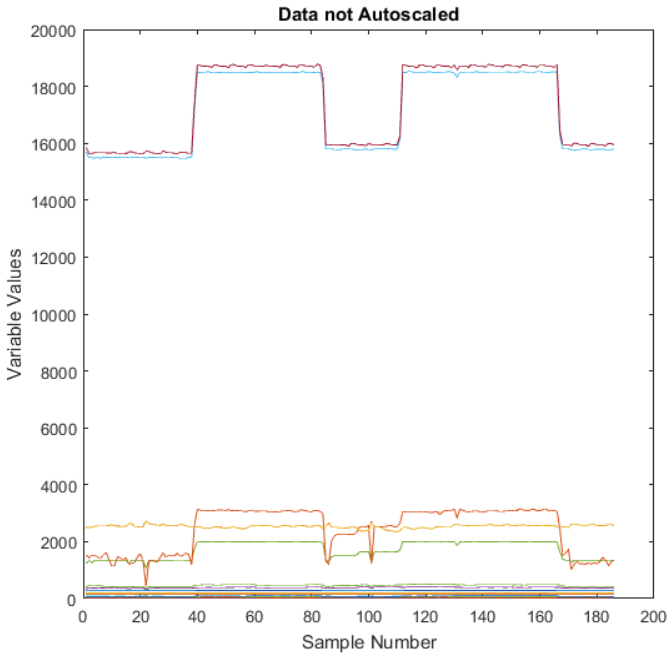


Figura 4.5. Datos de la muestra antes del pre-procesamiento.

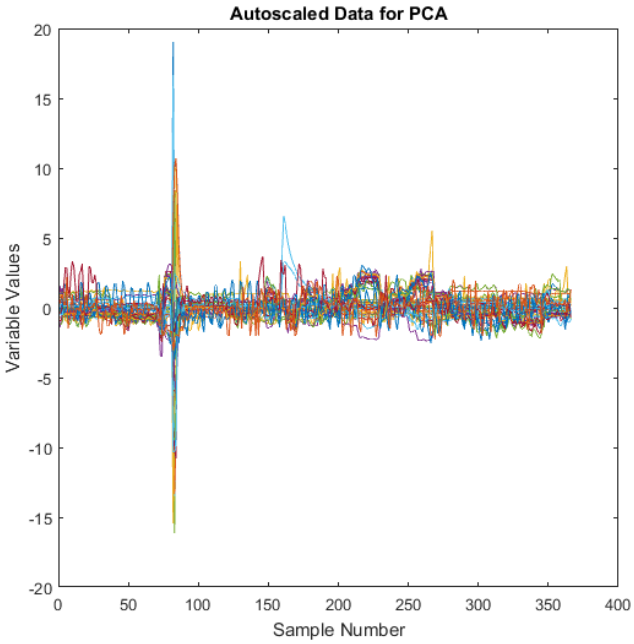


Figura 4.6. Datos para el modelo después del pre-procesamiento.

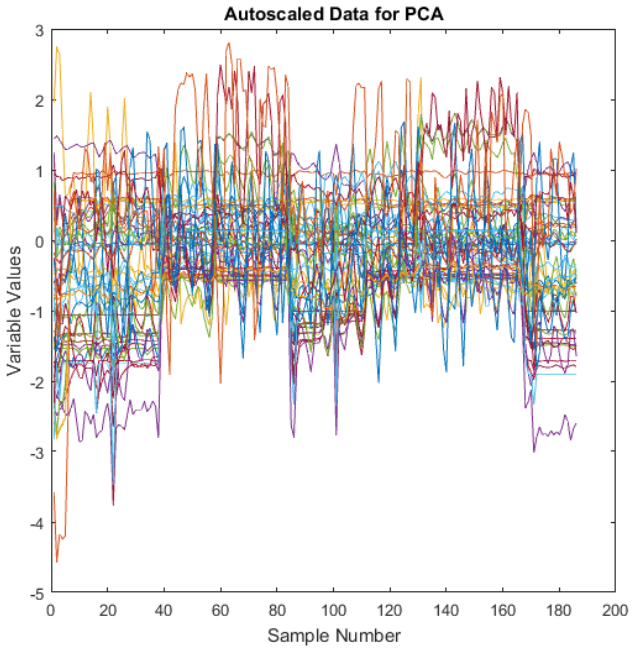


Figura 4.7. Datos de la muestra después del pre-procesamiento.

Tabla 4.5. Valores máximos y mínimos obtenidos del ajuste de los datos del modelo y la muestra.

Datos	Valor max.	Valor mín.
Modelo	19	-16
Muestra	8	-4

Se puede observar que los valores máximos y mínimos del modelo son mayores que la muestra, esto se debe principalmente a que en el conjunto de datos existen algunas anormalidades en el registro de los datos, pero que no afectan el comportamiento normal del conjunto de datos del modelo.

Eigenvalores y eigenvectores.

Con base a la metodología SVD se calcularon los eigenvalores y eigenvectores del conjunto de datos del modelo para seleccionar el número de componentes (**¡Error! No se encuentra el origen de la referencia.**). El número de componentes a seleccionar depende del porcentaje de varianza acumulada total, que para este caso es del 85%, lo cual se observa que en el componente número 5 se tiene el valor del 86.1306% acumulado. Es importante tomar en cuenta que los primeros tres componentes son los que acumulan la mayor varianza en la mayoría de los casos, en esto radica la importancia de esta metodología de la descomposición de valores singulares.

Tabla 4.6. Eigenvalores y varianza acumulada por componente.

PC Number	Eigenvalue of Cov(X)	% Variance Captured This PC	% Variance Captured Total
1	27.3183	47.2513	47.2513
2	14.6830	25.3965	72.6478
3	2.9942	5.1789	77.8267
4	2.8304	4.8957	82.7223
5	1.9705	3.4083	86.1306
6	1.5333	2.6520	88.7827
7	1.2292	2.1262	90.9088
8	0.9712	1.6799	92.5887
9	0.8138	1.4076	93.9964
10	0.7972	1.3790	95.3753
11	0.5779	0.9995	96.3748
12	0.5309	0.9182	97.2931
13	0.3857	0.6672	97.9602
14	0.2838	0.4908	98.4510
15	0.2402	0.4155	98.8665
16	0.1845	0.3191	99.1856
17	0.1792	0.3099	99.4955
18	0.1304	0.2256	99.7211

En la Figura 4.8 se puede observar los valores de los eigenvalores por número de componente y que hasta el componente 5 el eigenvalor es mayor a 1. Por lo cual se selecciona este número de componentes.

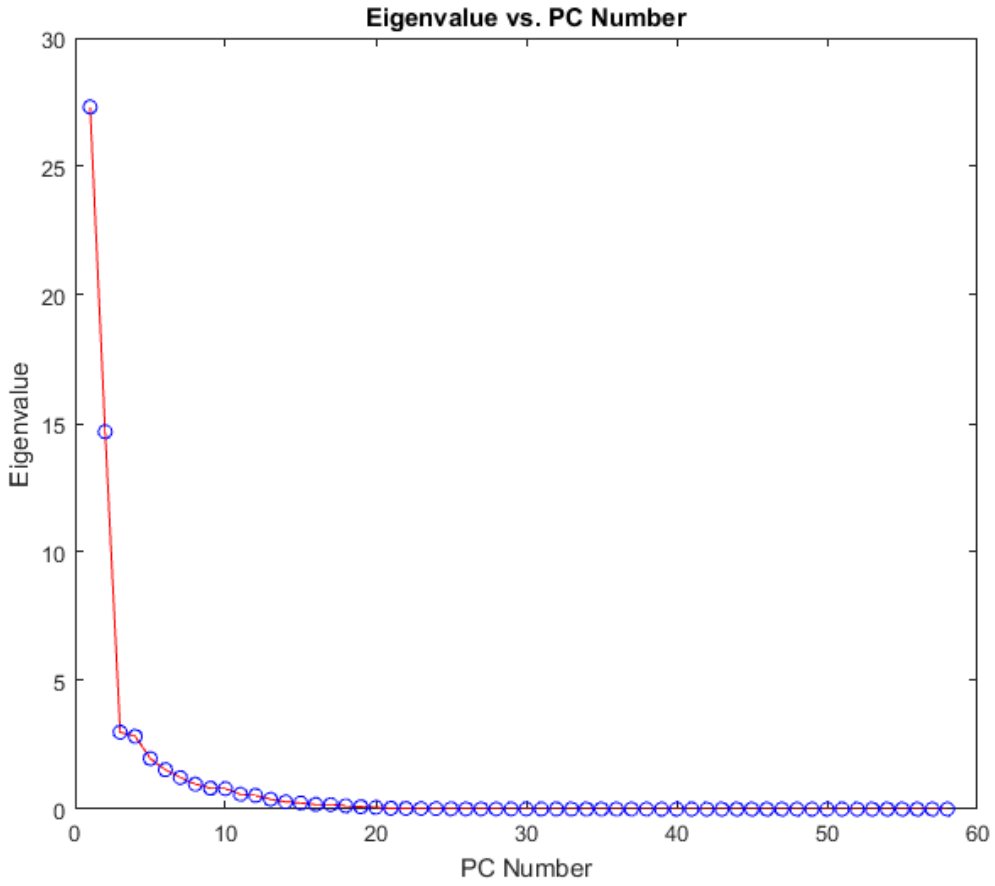


Figura 4.8. Varianza acumulada por componente.

Suma de Cuadrados del Error Residual Pronosticado (PRESS).

Se realizó el cálculo del PRESS y se graficaron los resultados y con estos valores se pudo seleccionar el número más adecuado para el modelo. Como se muestra en la Figura 4.9 el número de componente que tiene el valor más bajo de PRESS acumulado es el componente 4, lo que este sería el número de componente a seleccionar utilizando este método.

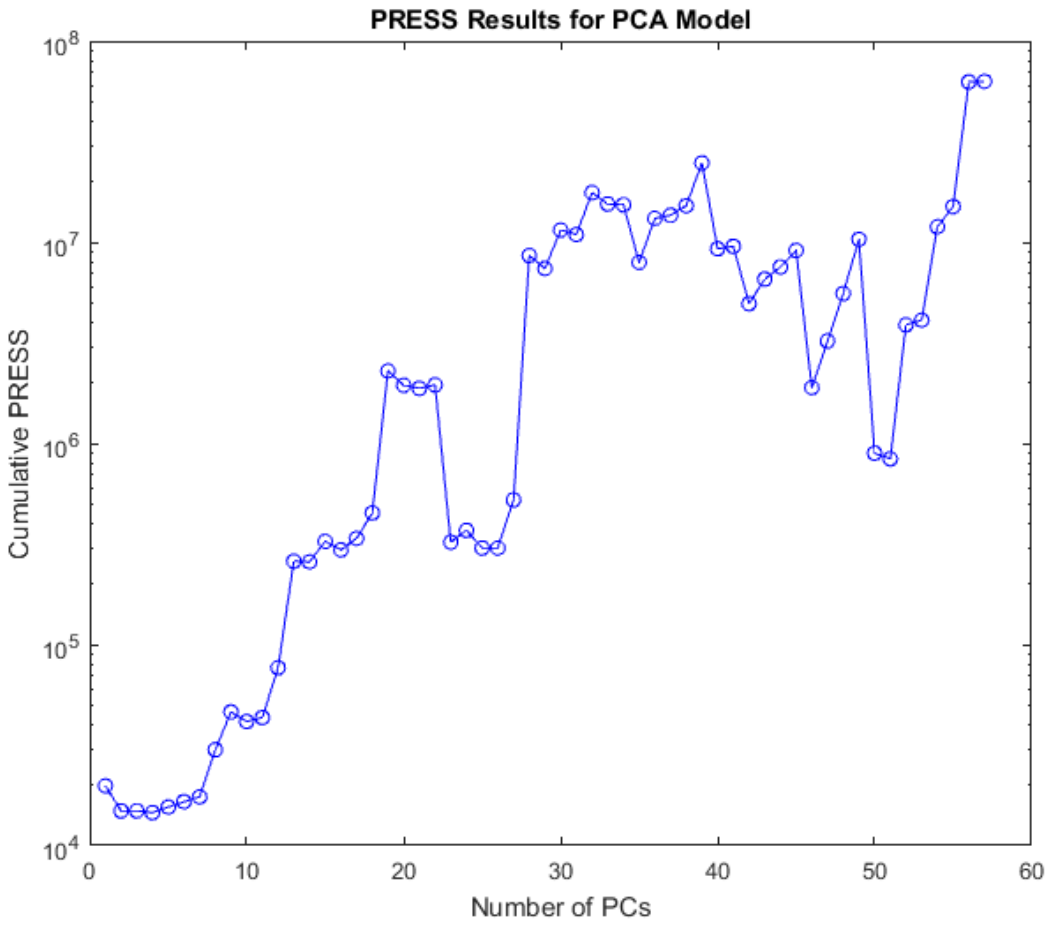


Figura 4.9. PRESS acumulado por componente.

Error de Predicción Cuadrático (SPE).

Se calculó el valor de SPE y se graficaron los valores de los datos que se utilizaron para generar el modelo.

Se observa en la Figura 4.10 que los datos del modelo, desde el 72 al 88, se encuentran fuera del valor del SPE (20.5056), estos valores pueden representar una desviación del conjunto de datos normalizados, probablemente una operación anormal en el reactor de la planta.

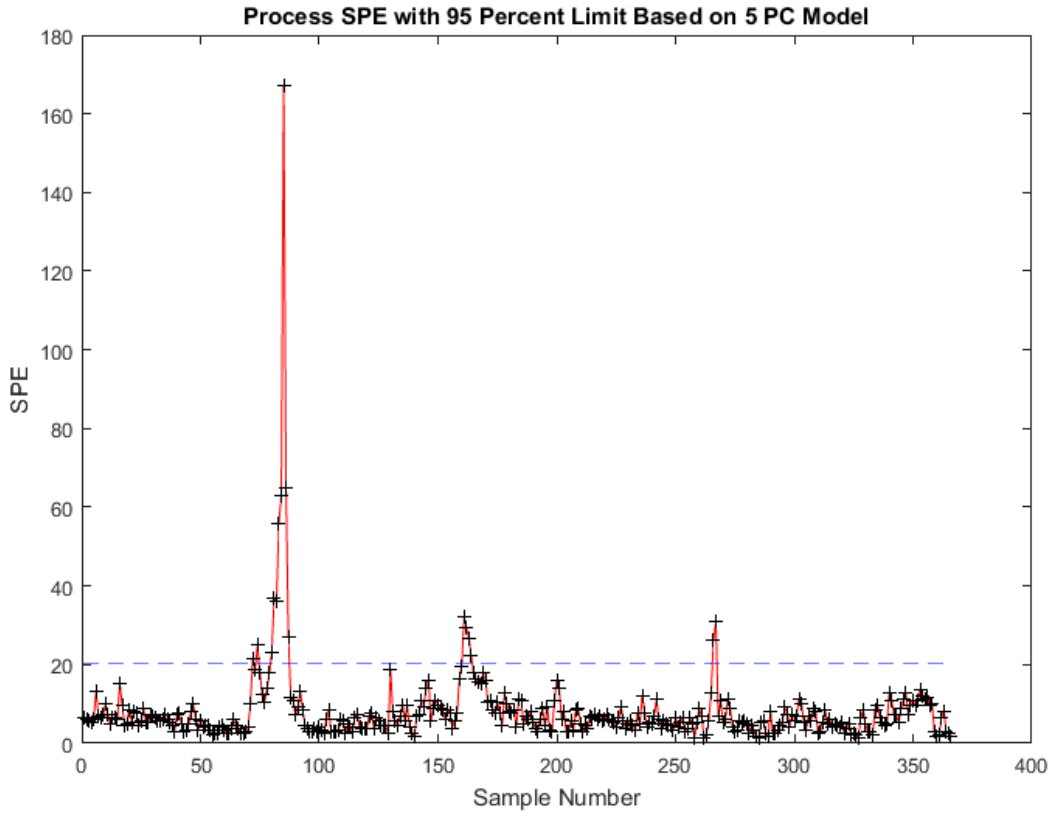


Figura 4.10. Valores de las variables respecto al SPE.

Modelo PCA.

Con la selección de componentes, se generó el modelo PCA, obteniendo el valor de Hotelling y el del Error Cuadrático de Predicción (SPE) (**¡Error! No se encuentra el origen de la referencia.**). Se consideran que los valores que se encuentran dentro del límite de Q – residual, representan los valores del modelo PCA del reactor.

Tabla 4.7. Modelo PCA del sistema del reactor.

MODELO PCA	Valor
Número de Componentes	**5**
Porciento de varianza acumulada	**86.13**
T^2 *de Hotelling*	**15.5711**
Error Cuadrático de Predicción (Q)	**20.5056**

T^2 de Hotelling.

Además del valor del SPE, se obtuvo además el valor estadístico de la T^2 de Hotelling y se graficó este valor para compararlo con los valores de los datos del modelo.

Como se puede observar en la Figura 4.11 los valores de los datos del modelo están dentro del valor de la T^2 de Hotelling, exceptuando los valores que se encuentran entre los datos 80 y 86, que son muy similares a los mostrados por la Figura 4.10.

Matriz de puntuaciones del modelo.

Con la matriz de puntuaciones de los componentes 1 al 5, los datos se ajustan con el modelo PCA, como se muestran en las figuras de la 4.12 a la 4.16.

Como se puede observar la matriz de puntuaciones de los datos se encuentran dentro de los límites (95%) del modelo en todos los cinco componentes, solo una cantidad muy mínima se salió de los límites.

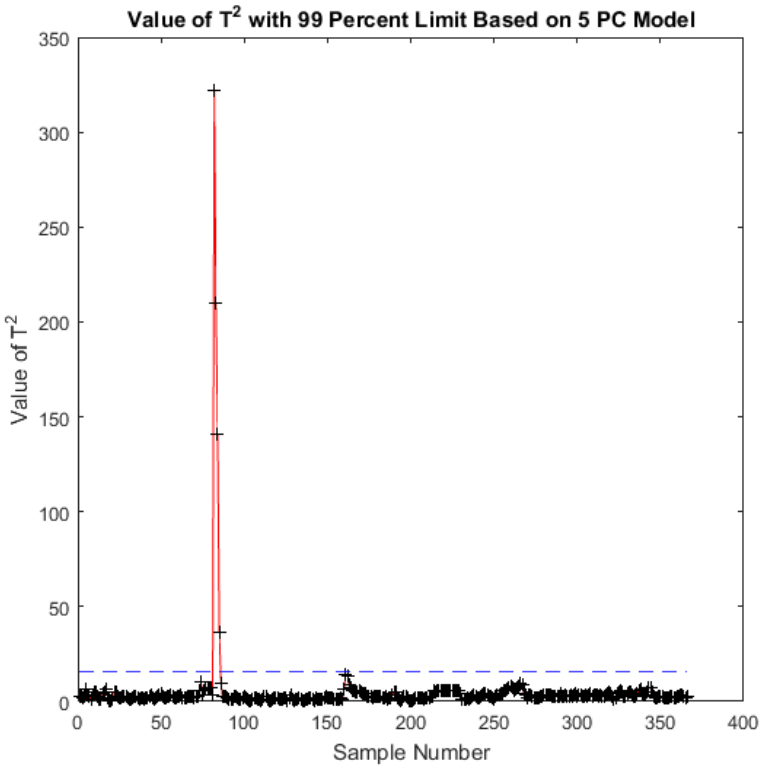


Figura 4.11. Valores de la T2 de Hotelling.

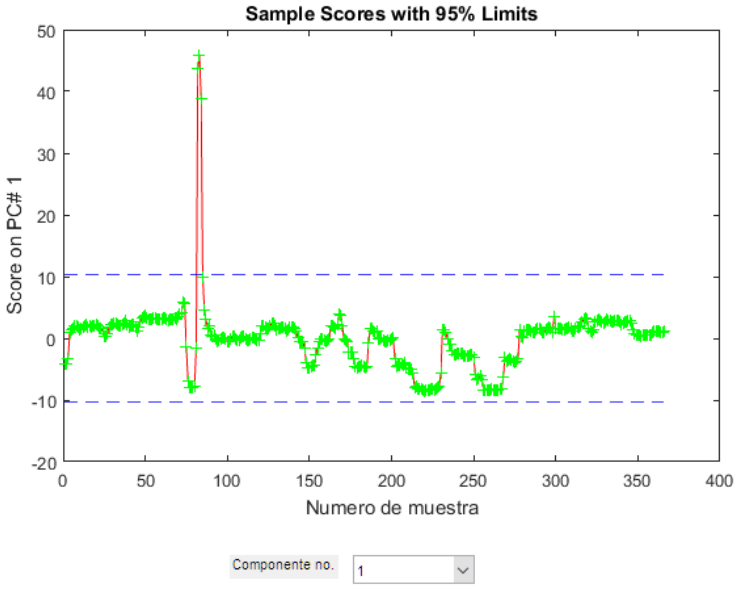


Figura 4.12. Matriz de puntuaciones del componente 1.

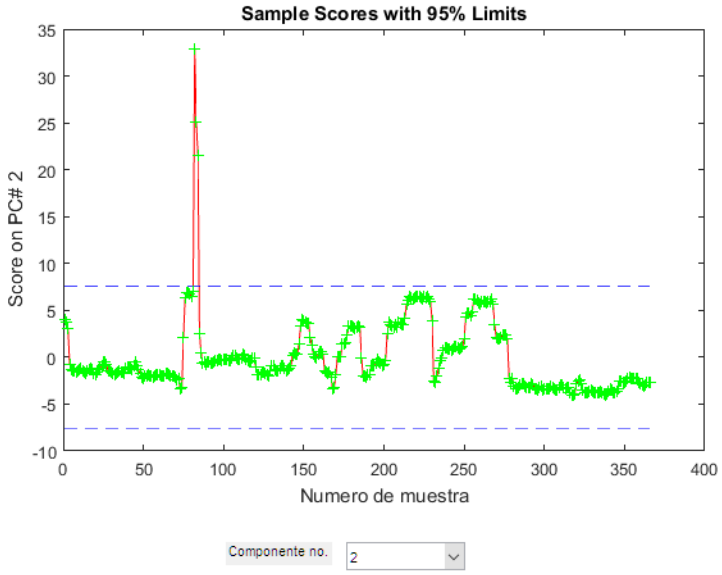


Figura 4.13. Matriz de puntuaciones del componente 2.

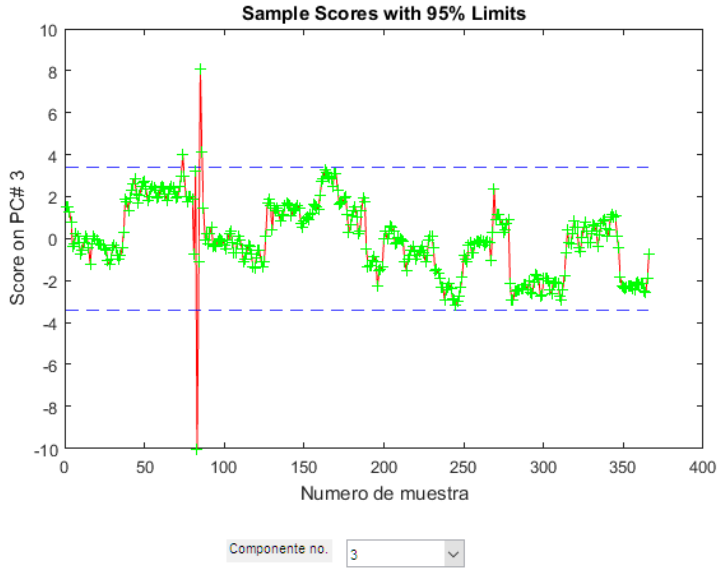


Figura 4.14. Matriz de puntuaciones del componente 3.

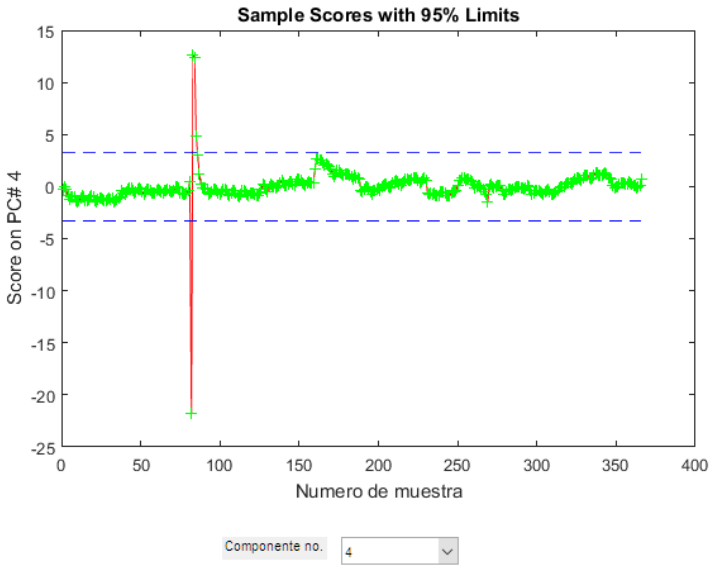


Figura 4.15. Matriz de puntuaciones del componente 4.

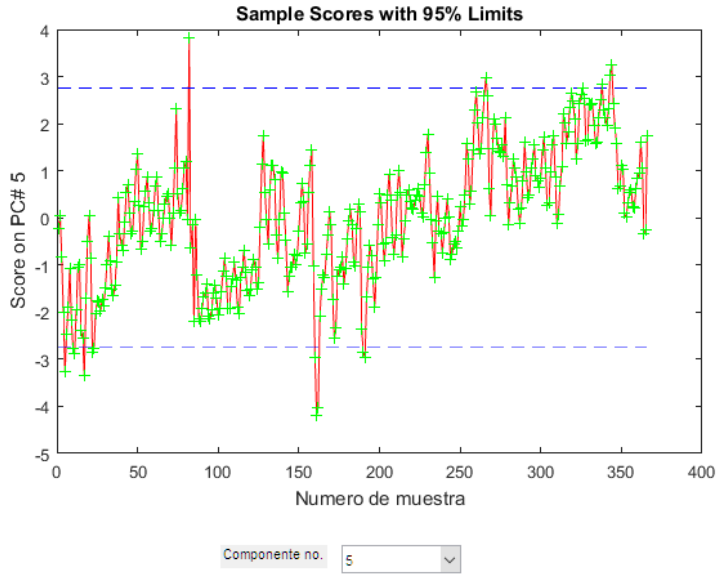


Figura 4.16. Matriz de puntuaciones del componente 5.

Matriz de cargas del modelo.

Las matrices de cargas obtenidas con el modelo PCA se graficaron para comprobar el comportamiento de las variables por cada componente de la figura 4.17 a la 4.21.

Como se observa en las gráficas de las matrices de carga, los componentes 1 y 2 representan la mayor correlación de las variables del modelo y también se puede observar cuales variables son las que contribuyen más al modelo (se observa con mayor detalle con las gráficas de contribución).

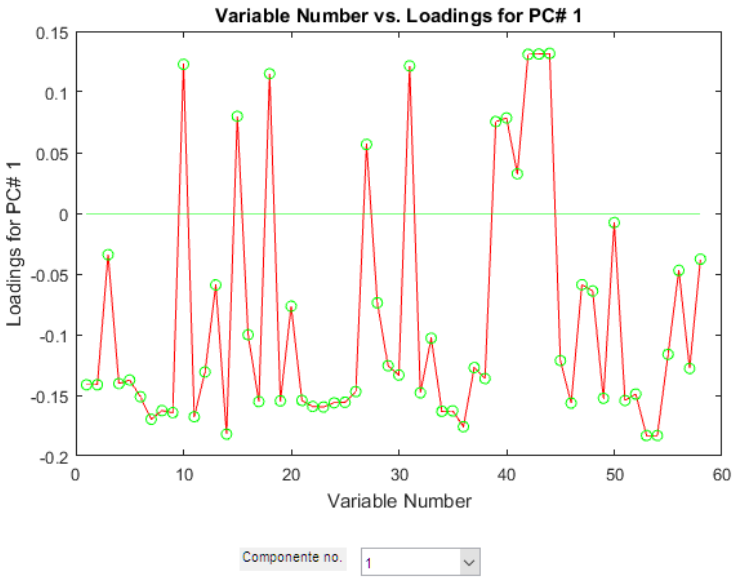


Figura 4.17. Matriz de cargas del componente 1 de las variables del modelo.

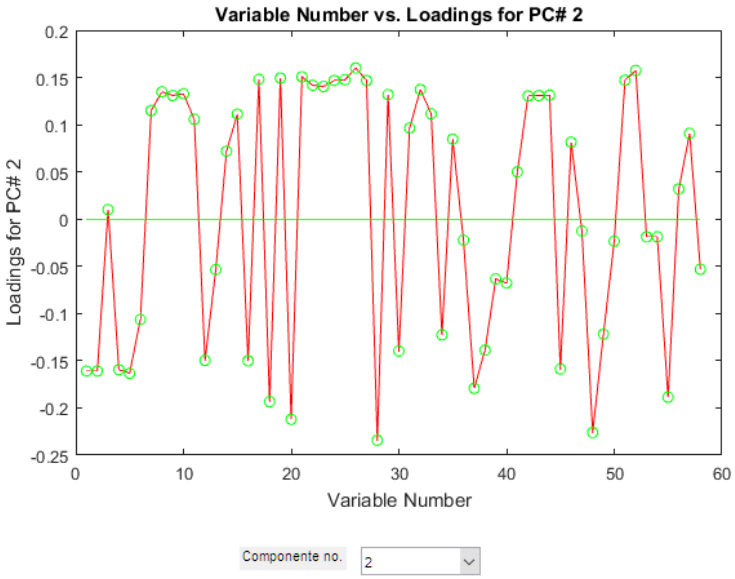


Figura 4.18. Matriz de cargas del componente 2 de las variables del modelo.

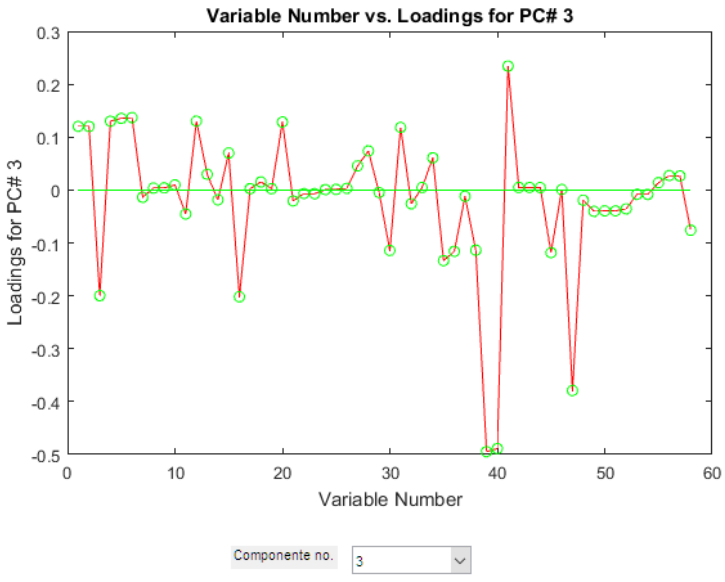


Figura 4.19 Matriz de cargas del componente 3 de las variables del modelo.

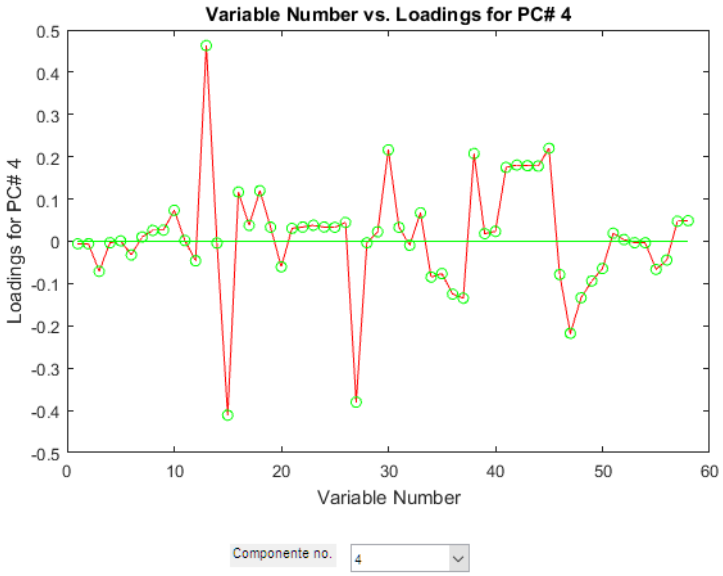


Figura 4.20 Matriz de cargas del componente 4 de las variables del modelo.

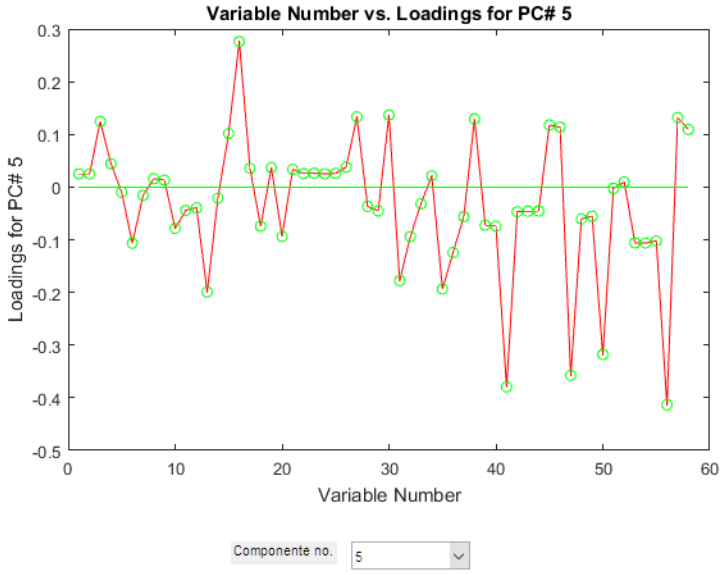


Figura 4.21 Matriz de cargas del componente 5 de las varables del modelo.

Gráfica biplot.

Se utilizó la gráfica biplot para representar las combinaciones de las variables y los valores de los datos del modelo, se pueden utilizar los cinco componentes, pero como se mencionó anteriormente, los componentes 1 y 2 son los que representan mayormente el modelo del reactor, porque contiene la mayor correlación de las variables.

En la Figura 4.22 se identifican las variables (color azul) que más afectan al modelo del sistema del reactor, estas se encuentran en el cuadrante superior derecho, Los datos del modelo (color rojo) se concentran al centro de la gráfica, exceptuando los datos que se localizan en el cuadrante inferior izquierdo (observaciones 82, 83 y 84), esto coincide con el resultado obtenido en la gráfica de la T^2 de Hotelling.

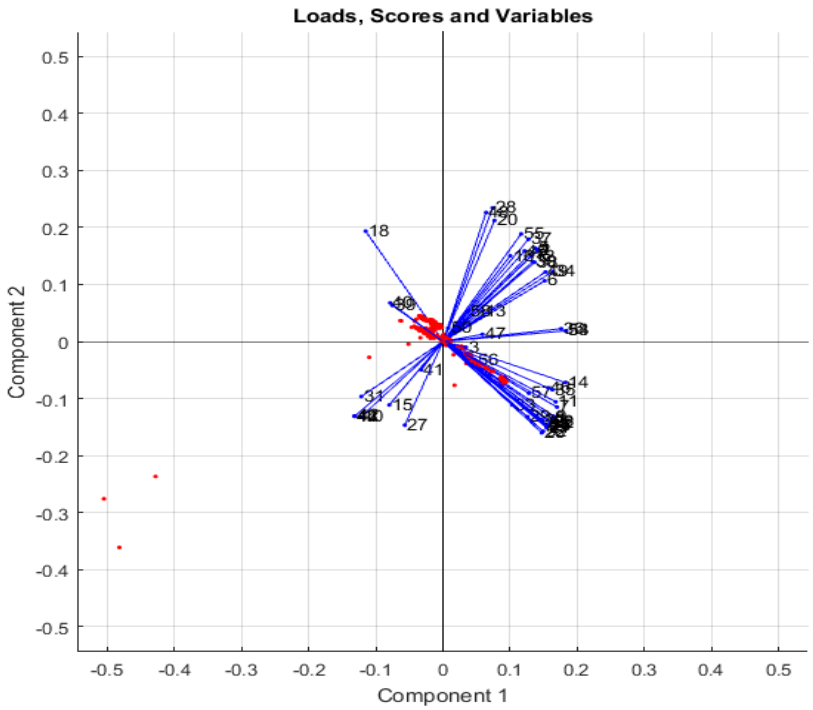


Figura 4.22. Gráfica biplot de las variables y datos del modelo utilizando los componentes 1 y 2.

Las variables que se identifican como las que más afectan al modelo se muestran en la Tabla 4.8.

Tabla 4.8. Variables del sistema de reactor que más afectan al modelo.

No.	Variable	Units	Condition	Group	System	Trip
1	AC-30023-1/PID1/PV.CV	% oxygen	Oxygen pressure control valve 1	01 ANALYSE	Reactor / Sopladores	Plant trip
2	AC-30023-2/PID1/PV.CV	% oxygen	Oxygen pressure control valve 2	01 ANALYSE	Reactor / Sopladores	Plant trip
4	AI-30023A/AI1/PV.CV	% oxygen	Low / High oxygen content A	01 ANALYSE	Reactor	MeOH trip
5	AI-30023B/AI1/PV.CV	% oxygen	Low / High oxygen content B	01 ANALYSE	Reactor	MeOH trip
6	AI-30023C/AI1/PV.CV	% CO	Low / High CO content	02 FLOW	Reactor	Plant trip
7	FC-31042/PID1/PV.CV	kg/h	High flow MeOH	02 FLOW	Reactor	MeOH trip
12	LC-41023/PID1/PV.CV	% level	Low BFW level HTF condensor	04 LEVEL	Condensador	MeOH trip
13	LI-40042/AI1/PV.CV	% level	Low BFW level HTF tank	04 LEVEL	Condensador	Plant trip
16	PC-41026/PID1/PV.CV	bara	High pressure HTF	05 PRESSURE	Condensador	Blower trip
20	PI-30041/AI1/PV.CV	barg	Methanol pressure after the filter before the vaporizer (after strain)	05 PRESSURE	Reactor	Plant trip
28	PI-92048/AI1/PV.CV	barg	Input pressure indicator to the BFW condenser	06 TEMP1	Condensador	Plant trip
30	TC-31067/ALM1/PV.CV	°C	High temp HTF vapour	06 TEMP1	Reactor / Condensador	MeOH trip
34	TI-31046/AI1/PV.CV	°C	High temp PG after FA reactor	06 TEMP1	Reactor	Blower trip
36	TI-31050/AI1/PV.CV	°C	High temp PG before Abosrber	07 TEMP2	Reactor	Blower trip
37	TI-31062/AI1/PV.CV	°C	High temp PG before FA reactor	07 TEMP2	Reactor	Blower trip
38	TI-31066/AI1/PV.CV	°C	High temp HTF vapour	07 TEMP2	Reactor / Condensador	MeOH trip
45	TI-41021/AI1/PV.CV	°C	High temp HTF condensor bottom	07 TEMP2	Condensador	Plant trip
47	TI-50091/AI1/PV.CV	°C	Recirculating air temperature from adsorber to blowers	07 TEMP2	Reactor	Plant trip
48	TI-55064/AI1/PV.CV	°C	High temp after TC trubine	07 TEMP2	Reactor	Blower trip
49	TI-55065/AI1/PV.CV	°C	High temp after catalytic bed	07 TEMP2	Reactor	Blower trip
50	TI-80001/AI1/PV.CV	°C	Blower room temperature	08 MISC	Sopladores	Plant trip
53	SI-31044A/AI1/PV.CV	rpm	Revolutions per minute F-2108	08 MISC	Sopladores	Plant trip
54	SI-31044B/AI1/PV.CV	rpm	Revolutions per minute F-2109	08 MISC	Sopladores	Plant trip
55	VOL-31042/AI1/PV.CV	% vol	High inlet MeOH	08 MISC	Reactor	MeOH trip
58	XI-21103/AI1/PV.CV	N-m	Torque TC	08 MISC	Sopladores	Plant trip

Comparación del modelo contra la muestra.

Después de generar el modelo PCA y validar los datos, se procedió a comparar los datos de la muestra (enero 2019) contra el modelo PCA generado. Se repitió el mismo proceso de tratamiento de los datos de la muestra como el proceso de datos del modelo, se ajustan y se centran los datos de la muestra con los valores estadísticos de los datos del modelo.

T^2 de Hotelling del modelo contra la muestra.

Se compara el valor de T^2 de Hotelling con los valores de la muestra para la identificación de fallas.

En la Figura 4.23 se observó que ninguno de los valores de la muestra estuvo por arriba del valor de T^2, lo que implica que no se detectó ninguna falla en los datos de la muestra.

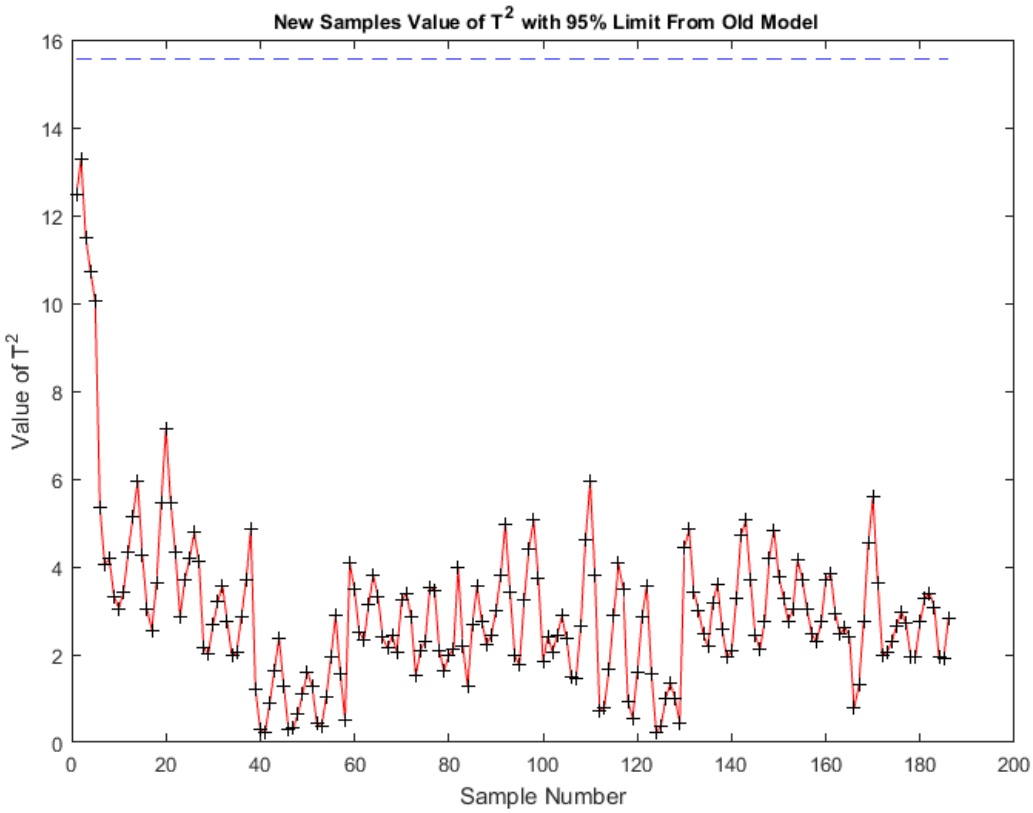


Figura 4.23. Comparación del valor de T2 de Hotelling de la muestra con el valor del 95% de límite del modelo PCA.

SPE del modelo contra la muestra.

Se comparó el valor de *Q–residual* con los valores de la muestra, para identificar las fallas que pasen éste valor. En la Figura 4.24 se pueden identificar siete fallas, las cuales fueron corroboradas con los datos de la planta y efectivamente sucedieron paros de la planta por fallas en el sistema del reactor, las cuales se identificaron con más detalle posteriormente con la ayuda de las gráficas de contribución.

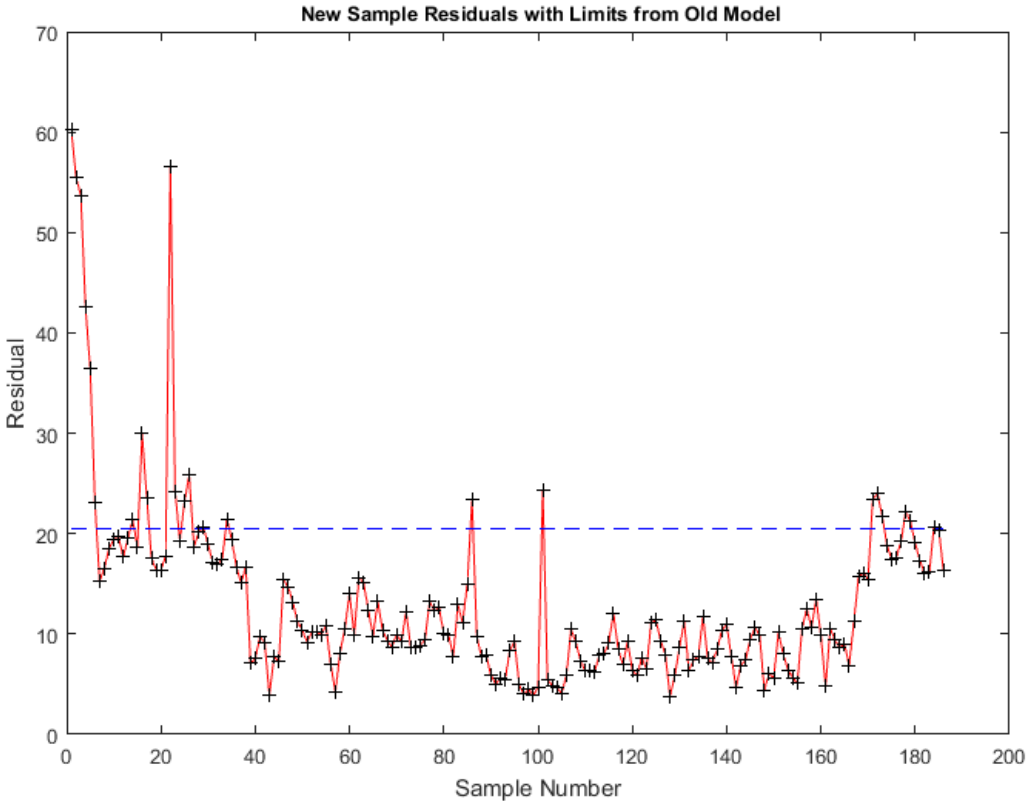


Figura 4.24. Comparación del valor de Q-residual del modelo con el valor de las muestras.

Gráfica de dispersión.

Se graficaron los tres primeros componentes utilizando una gráfica de dispersión, la cual permite observar la matriz de puntuaciones, tanto del modelo como de la muestra en una misma gráfica y así se identificaron si el modelo se ajusta a la muestra.

Como se observa en la Figura 4.25 el comportamiento de los datos de muestra (color rojo) se ajustan en su mayoría a los datos del modelo (color azul), sin embargo, al igual que con el valor de SPE que se comparó con la muestra, algunos valores de la muestra están fuera del modelo, estos valores son los que representan la falla en el reactor.

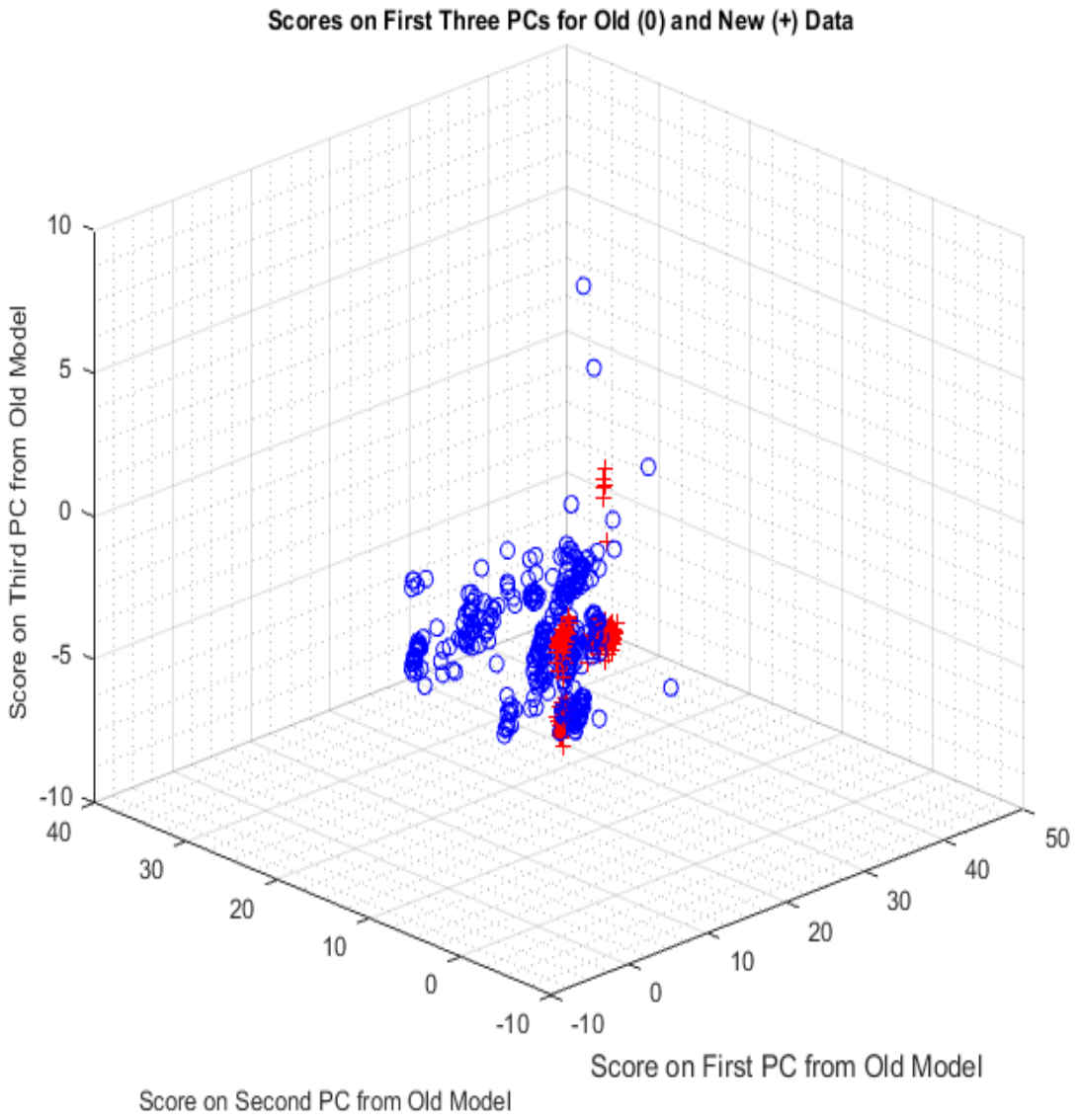


Figura 4.25. Comparación de las puntuaciones del modelo contra las puntuaciones de la muestra.

Gráficas de contribución.

Para identificar las fallas dentro del sistema del reactor se utilizaron las gráficas de contribución, con estas gráficas se pudo conocer donde ocurrió la falla en el sistema del reactor. Las gráficas de contribución son: variable con la más alta contribución, contribución al SPE por muestra y la gráfica de reconstrucción y proyección de las variables.

Variable con la más alta contribución.

Esta gráfica permitió identificar las variables que más contribuyen al sistema del reactor basado en el modelo PCA generado y la muestra. Los valores que se encontraron arriba del valor del SPE se muestran en la Figura 4.26 y permitieron identificar cuales variables fueron las que más contribuyeron a la falla. En este caso las que más contribuyeron son las variables 3, 16, 55 y 56.

Contribución al SPE.

Esta gráfica permitió ver la contribución al SPE de cada variable por muestra específica.

Se graficó la contribución al SPE de la muestra 171 (Figura 4.27), ya que dentro de esta muestra ocurrió una falla dentro del reactor, y muestra que variables fueron las que más contribuyeron al SPE cuando ocurre una falla. En este caso, como se observa en la gráfica la variable que más contribuye es la variable de la válvula de entrada del metanol (no. 55).

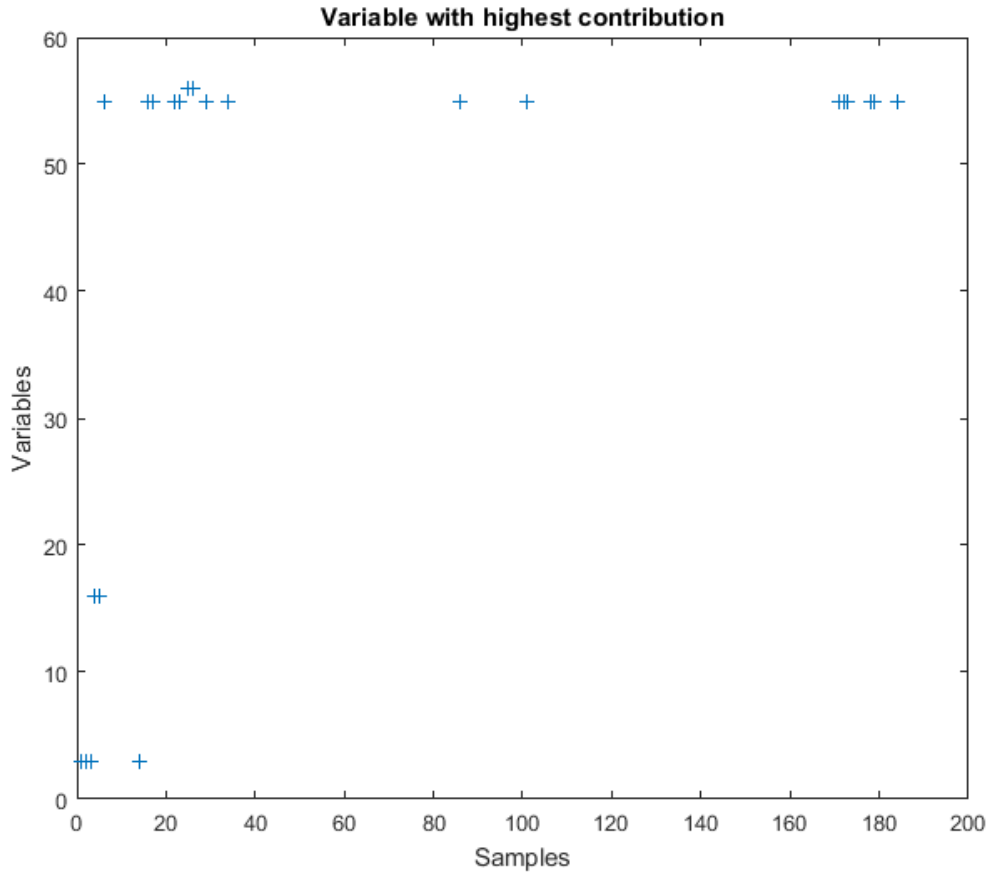


Figura 4.26. Variable con la más alta contribución.

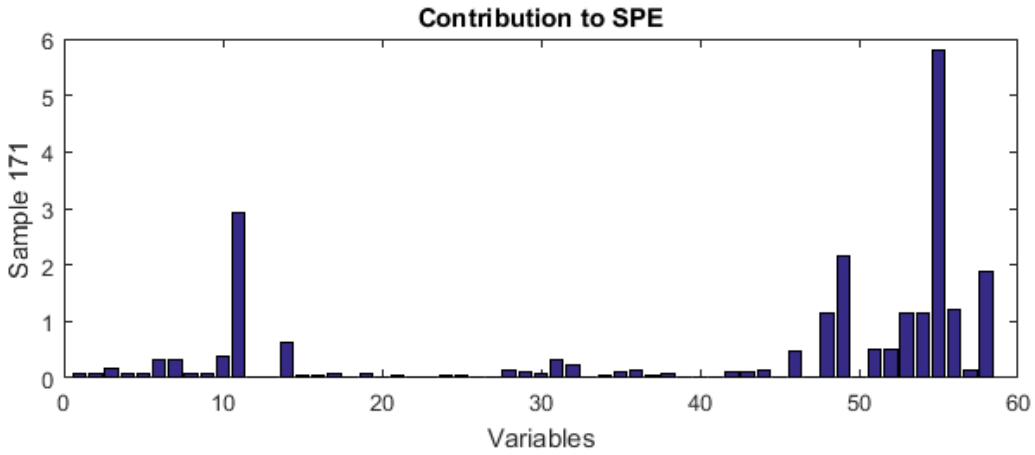


Figura 4.27. Contribución al SPE de la muestra 171.

Gráfica de datos, proyección y reconstrucción.

Esta gráfica mostró los datos, su proyección y reconstrucción de una muestra específica, para poder conocer si los valores de dicha muestra se ajustan al modelo y la proyección de los mismos basados en la matriz de cargas.

En la Figura 4.28 se observó el comportamiento de la proyección (color negro) y reconstrucción (color rojo) de los datos de la muestra 171 y la variable 55, como se ve en esta gráfica la variable se ajusta a los datos reales (color azul). La diferencia entre los datos reales y los datos proyectados y reconstrucción es debido a que estos últimos se ajustaron con la matriz de cargas y puntuaciones.

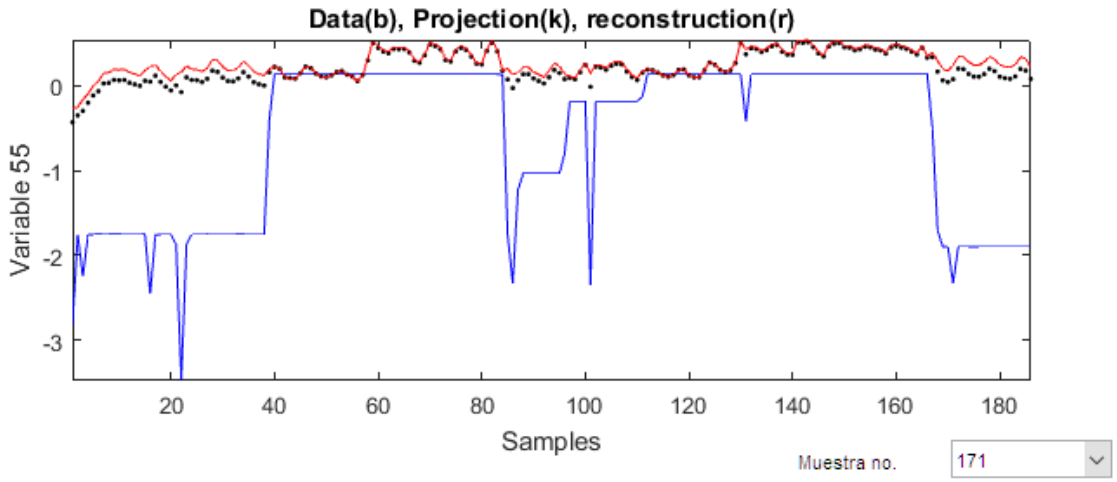


Figura 4.28. Datos, proyección y reconstrucción de datos de la muestra.

Detección e identificación de fallas.

Las fallas detectadas e identificadas en la muestra del mes de Enero de 2019 con base al modelo PCA generado con la información del periodo noviembre – diciembre de 2018, se muestran en la Tabla 4.9.

Tabla 4.9. Identificación de Fallas dentro del sistema del reactor.

Falla	Muestra no.	Fecha	Variable	Ubicación	Trip
1	1 - 6	01/01/2019	Analizador diferencial de oxígeno, Presión condensador HTF	Condensador	**Blower Trip**
2	14 - 17	03/01/2019	Válvula de entrada del metanol	Reactor	**MeOH Trip**
3	22 - 24	04/01/2019	Válvula de entrada del metanol	Reactor	**MeOH Trip**
4	34	06/01/2019	Válvula de entrada del metanol	Reactor	**MeOH Trip**
5	86	15/01/2019	Válvula de entrada del metanol	Reactor	**MeOH Trip**
6	101	17/01/2019	Válvula de entrada del metanol	Reactor	**MeOH Trip**
7	171 - 173	29/01/2019	Válvula de control de presión de oxígeno, Válvula de control de entrada de aire	Reactor / Sopladores	**Plant Trip**

Las variables identificadas que fueron más afectadas fueron en el sensor de entrada del metanol (variable 55), en el analizador de diferencial de oxígeno (variable 3), en la entrada superior del condensador (variable 16), en la válvula de control de presión de oxígeno (variable 1) y en la válvula de control de presión de oxígeno en la entrada de aire fresco (variable 2).

Prevención de emisión de contaminantes a la atmósfera.

En la detección e identificación de fallas se evitó que la operación del reactor se comprometiera y pasará a un estado crítico de operación, lo que evita fugas de formaldehído, potencialmente peligrosas al aire poniendo en riesgo la seguridad del personal de la planta y el ecosistema circundante de la planta.

Capítulo 5 Conclusiones y recomendaciones.

Se identificaron 25 variables más importantes del sistema del reactor que son críticos para el funcionamiento de la planta. Se detectaron 7 fallas en la muestra que se comparó contra el modelo, utilizando el valor del Error de Predicción Cuadrática. La variable que más contribuyó a la mayoría de las fallas fue la del sensor de entrada del metanol esto es entendible debido a que cualquier falla que suceda en la planta detiene el flujo del metanol hacia el reactor. El modelo del sistema del reactor debe actualizarse al menos cada mes, en el que se haya tenido una operación estable. Es recomendable el ajustado y centrado de datos para la normalización de los datos.

El sistema de control automatizado de la planta de formaldehído debe complementarse con un sistema de detección e identificación de fallas ya que actualmente las fallas no son detectadas hasta que el sistema automatizado realiza un paro automático de la planta cuando los valores de los sensores llegan a estados críticos.

Para una detección de fallas más oportuna, se recomienda utilizar la información que se registra cada segundo, en lugar de la registrada cada cuatro horas, debido a que el modelo se vuelve más sensible a los cambios en las variables con los registros por segundo a los modelos generados por los reportes mensuales.

La obtención de modelos matemáticos para la detección e identificación de fallas en un reactor químico, es una técnica eficiente y que permite un análisis oportuno cuando se tiene una gran cantidad de información y evita que se ponga en riesgo la operación y seguridad del reactor catalítico, lo que permite minimizar el posible daño al medio ambiente.

Bibliografía.

Bro, R., K. Kjeldahl, A. K. Smilde and H. A. L. Kiers (2008). "Cross-validation of component models: A critical look at current methods." Analytical and Bioanalytical Chemistry 390(5): 1241-1251.

Buffo Flavia, V. A (2015). *La decomposición en valores singulares: un enfoque geométrico y aplicaciones*. Retrieved from Argentina:

Chen, J., & Patton, R. J. (2012). *Robust Model-Based Fault Diagnosis for Dynamic Systems*: Springer US.

Diana, G. and C. Tommasi (2002). "Cross-validation methods in principal component analysis: A comparison." Statistical Methods and Applications 11(1): 71-82.

Gertler, J., & Luo, Q. (1998, 1998). *Direct identification of nonlinear parity relations. A way around the robustness problem.* Paper presented at the Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171).

G.H. Golub & Ch. F. Van Loan (1989). *Matrix Computations.*Johns Hopkins University Press, Baltimore, Maryland.

Guerrero Fajardo, C. A., & Sánchez Castellanos, F. J. (2009). Synthesising Fe-Mo catalysts supported on silica for the selective oxidation of methane to formaldehyde. *Ingeniería e Investigación; Vol 29, No 1 (2009)*.

H. Hotelling. Analysis of a complex of statistical variables into principal components. J. Educ. Psychol., 24(6):417-441, 1993.

Harman, T. L., Liebfried, T. F., Clark, J. W., & Hibbs, C. W. (1975). A Comparison of Two Methods for Determining the Extracellular Potential Field of an Isolated Purkinje Strand in a Volume Conductor. *IEEE Transactions on Biomedical Engineering, BME-22*(3), 174-183. doi:10.1109/TBME.1975.324557

Harrou, F., M. N. Nounou, H. N. Nounou and M. Madakyaru (2013). "Statistical fault detection using PCA-based GLR hypothesis testing." Journal of Loss Prevention in the Process Industries 26(1): 129-139.

Hu, Y., H. Chen, G. Li, H. Li, R. Xu and J. Li (2016). "A statistical training data cleaning strategy for the PCA-based chiller sensor fault detection, diagnosis and data reconstruction method." Energy and Buildings 112: 270-278

Hwang, I., Kim, S., Kim, Y., & Seah, C. E. (2010). A Survey of Fault Detection, Isolation, and Reconfiguration Methods. *IEEE Transactions on Control Systems Technology, 18*(3), 636-653. doi:10.1109/TCST.2009.2026285

INEGI. (2015). *La industria química en México 2014. Serie estadísticas sectoriales*. México, D.F.: Instituto Nacional de Estadística y Geografía.

IPCS, I. P. O. C. S. (1991). *Formaldehyde Health and Safety Guide*. (0259-7268). Geneva 1991: World Health Organization; United Nations Environment Programme; International Labour Organisation.

I.T. Jolliffe. Principal component analysis. Springer-Verlag,1986. New York.

Isermann, R. (2005). Model-based fault-detection and diagnosis – status and applications. *Annual Reviews in Control, 29*(1), 71-85. doi:https://doi.org/10.1016/j.arcontrol.2004.12.002

Isermann, R. (2006). *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*: Springer Berlin Heidelberg.

Josse, J. and F. Husson (2012). "Selecting the number of components in principal component analysis using cross-validation approximations." Computational Statistics & Data Analysis 56(6): 1869-1879.

K. Pearson. On lines and planes of closest fit to systems of points in space. Phil. Mag., 2(11):559-572,1901.

Kaiser, H. F. (1974). "An index of factorial simplicity." Psychometrika 39(1): 31-36.

Kassouf, A., D. Jouan-Rimbaud Bouveresse and D. N. Rutledge (2018). "Determination of the optimal number of components in independent components analysis." Talanta 179: 538-545.

Lay, D. C. (1999). *Álgebra lineal y sus aplicaciones* (Segunda Edición ed.). México: Addison Wesley Longman

Lopez-Toribio, C. J., R. J. Patton, C. Batt and J. Chen (1999). Fault diagnosis of the 3 tank system using fuzzy multiple inference modelling. 1999 European Control Conference (ECC)

L.N. Trefethen & D. Bau (1997). *Numerical Linear Algebra.* SIAM, Philadelphia.

Mandal, S., N. Sairam, S. Sridhar and P. Swaminathan (2017). "Nuclear power plant sensor fault detection using singular value decomposition-based method." Sādhanā 42(9): 1473-1480.

Mujica, L. E., J. Rodellar, A. Fernández and A. Güemes (2010). "Q-statistic and T2-statistic PCA-based measures for damage assessment in structures." Structural Health Monitoring 10(5): 539-553.

Sanchez-Fernández, A., M. J. Fuente and G. I. Sainz-Palmero (2015). Fault detection in wastewater treatment plants using distributed PCA methods. 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA).

Scheible, R. W. H. P. M. R. E. S. J. J. (1995). Method of Controlling a Manufacturing Process Using Multivariate Analysis. United States, Eastman Kodak Company: 22.

Soares, A. P. V., Portela, M. F., & Kiennemann, A. (2005). Methanol Selective Oxidation to Formaldehyde over Iron- Molybdate Catalysts. *Catalysis Reviews, 47*(1), 125-174. doi:10.1081/CR-200049088.

Tamura, M. and S. Tsujita (2007). "A study on the number of principal components and sensitivity of fault detection using PCA." Computers & Chemical Engineering 31(9): 1035-1046.

Tudón Martínez, J. C., R. Morales Menéndez, R. A. Ramírez Mendoza, L. E. Garza Castañón and A. Vargas Martínez (2011). "Fault Detection in a Heat Exchanger, Comparative Analysis between Dynamic Principal Component Analysis and Diagnostic Observers." Computación y Sistemas 14: 269-282.

Valle-Cervantes, S. (2001). Plant -wide monitoring of processes under closed - loop control. 3035991 Ph.D., The University of Texas at Austin.

van den Berg, R. A., H. C. J. Hoefsloot, J. A. Westerhuis, A. K. Smilde and M. J. van der Werf (2006). "Centering, scaling, and transformations: improving the biological information content of metabolomics data." BMC genomics 7: 142-142.

Venkatasubramanian, V., R. Rengaswamy, K. Yin and S. N. Kavuri (2003). "A review of process fault detection and diagnosis: Part I: Quantitative model-based methods." Computers & Chemical Engineering 27(3): 293-311.

Zhang, B., Sconyers, C., Byington, C., Patrick, R., Orchard, M. E., & Vachtsevanos, G. (2011). A Probabilistic Fault Detection Approach: Application to Bearing Fault Detection. *IEEE Transactions on Industrial Electronics, 58*(5), 2011-2018. doi:10.1109/TIE.2010.2058072

ANEXO 1

Código interface en Matlab para la detección e identificación de fallas en un reactor químico heterogéneo.

Archivo: FDIs.m

```
function varargout = FDIs(varargin)
% FDIS MATLAB code for FDIs.fig
%      FDIS, by itself, creates a new FDIS or raises the existing
%      singleton*.
%
%      H = FDIS returns the handle to a new FDIS or the handle to
%      the existing singleton*.
%
%      FDIS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in FDIS.M with the given input arguments.
%
%      FDIS('Property','Value',...) creates a new FDIS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before FDIs_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to FDIs_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help FDIs

% Last Modified by GUIDE v2.5 09-Feb-2019 16:59:33

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @FDIs_OpeningFcn, ...
                   'gui_OutputFcn',  @FDIs_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before FDIs is made visible.
function FDIs_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to FDIs (see VARARGIN)

% Choose default command line output for FDIs
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

begin_ws();

% UIWAIT makes FDIs wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = FDIs_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes during object creation, after setting all properties.
function popupmenuY_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenuY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------
function import_data_Callback(hObject, eventdata, handles)
% hObject    handle to import_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in import_data.
begin_ws();
importdat(hObject, handles,1,1);

% --------------------------------------------------------------------
function import_data_hour_Callback(hObject, eventdata, handles)
% hObject    handle to import_data_hour (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
begin_ws();
importdat(hObject, handles,1,2);

% --------------------------------------------------------------------
function save_ses_Callback(hObject, eventdata, handles)
% hObject    handle to save_sesio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
save_sesion();

% --------------------------------------------------------------------
function save_sesio_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to save_sesio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
save_sesion();

% --------------------------------------------------------------------
function rest_ses_Callback(hObject, eventdata, handles)
% hObject    handle to rest_ses (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
begin_ws();
restore_sesion(hObject, eventdata, handles);

% --------------------------------------------------------------------
function exit_Callback(hObject, eventdata, handles)
% hObject    handle to exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close

% --------------------------------------------------------------------
function load_data_Callback(hObject, eventdata, handles)
% hObject    handle to load_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Cargamos la matriz de datos al espacio de trabajo
begin_ws();
loaddat(hObject, eventdata, handles);

% --------------------------------------------------------------------
function load_data_tb_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to load_data_tb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Cargamos la matriz de datos al espacio de trabajo
begin_ws();
loaddat(hObject, eventdata, handles);

% --------------------------------------------------------------------
function data_dep_Callback(hObject, eventdata, handles)
% hObject    handle to data_dep (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Depuramos los datos con el metodo de 3-sigma
depu_dat3s(hObject, eventdata, handles);

% --------------------------------------------------------------------
function adjust_var_Callback(hObject, eventdata, handles)
% hObject    handle to adjust_var (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
fdataj(hObject, eventdata, handles,1)

% --------------------------------------------------------------------
function t_mod_ajust_Callback(hObject, eventdata, handles)
% hObject    handle to t_mod_ajust (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
FigHandle = tdata(1);

% --------------------------------------------------------------------
function t_sam_ajust_Callback(hObject, eventdata, handles)
% hObject    handle to t_sam_ajust (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
FigHandle = tdata(2);

% --------------------------------------------------------------------
function eigenval_acum_Callback(hObject, eventdata, handles)
% hObject    handle to eigenval_acum (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
eigenval_table();

% --------------------------------------------------------------------
function gen_mod_Callback(hObject, eventdata, handles)
% hObject    handle to gen_mod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
gen_model(hObject, eventdata, handles);

% --------------------------------------------------------------------
function data_select_hour_Callback(hObject, eventdata, handles)
% hObject    handle to data_select_hour (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dat_sel(hObject, eventdata, handles,2,2);
pcapro();

% --------------------------------------------------------------------
function data_select_Callback(hObject, eventdata, handles)
% hObject    handle to data_select (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dat_sel(hObject, eventdata, handles,2,1);
pcapro();

% --------------------------------------------------------------------
function model_ajust_Callback(hObject, eventdata, handles)
% hObject    handle to model_ajust (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_ajust(1);

% --------------------------------------------------------------------
function sample_ajust_Callback(hObject, eventdata, handles)
% hObject    handle to sample_ajust (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_ajust(2);

% --------------------------------------------------------------------
function crossvalid_Callback(hObject, eventdata, handles)
% hObject    handle to crossvalid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_fcrossva(hObject, eventdata, handles)

% --------------------------------------------------------------------
function eigenval_Callback(hObject, eventdata, handles)
% hObject    handle to eigenval_acum (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_feigenval(hObject, eventdata, handles)

% --------------------------------------------------------------------
function scores_95lim_Callback(hObject, eventdata, handles)
% hObject    handle to scores_95lim (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_score_pc();

% --------------------------------------------------------------------
function graph_loads_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to graph_loads (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_loads_pc();

% --------------------------------------------------------------------
function graph_resq_Callback(hObject, eventdata, handles)
% hObject    handle to graph_resq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_res_q(hObject, eventdata, handles);

% --------------------------------------------------------------------
function graph_t2_Callback(hObject, eventdata, handles)
% hObject    handle to graph_t2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_t2(hObject, eventdata, handles);

% --------------------------------------------------------------------
function biplot_graph_Callback(hObject, eventdata, handles)
% hObject    handle to biplot_graph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_biplot(hObject, eventdata, handles)

% --------------------------------------------------------------------
function resd_q_Callback(hObject, eventdata, handles)
% hObject    handle to resd_q (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_resdq();

% --------------------------------------------------------------------
function scores_T2_Callback(hObject, eventdata, handles)
% hObject    handle to scores_T2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_scorest2();

% --------------------------------------------------------------------
function score_95lim_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to score_95lim (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_95limits();

% --------------------------------------------------------------------
function scater3d_Callback(hObject, eventdata, handles)
% hObject    handle to scater3d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_scatter3d();

% --------------------------------------------------------------------
function contr_var_Callback(hObject, eventdata, handles)
% hObject    handle to contr_var (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_contr_var();

% --------------------------------------------------------------------
function spe_contrib_Callback(hObject, eventdata, handles)
% hObject    handle to spe_contrib (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graph_spe_contrib();

% --------------------------------------------------------------------
function init_data_Callback(hObject, eventdata, handles)
% hObject    handle to init_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
exportxls(1);

% --------------------------------------------------------------------
function dep_mat_Callback(hObject, eventdata, handles)
% hObject    handle to dep_mat (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
exportxls(2);

% --------------------------------------------------------------------
function m_data_ajust_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to m_data_ajust (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
exportxls(3);

% --------------------------------------------------------------------
function m_scores_Callback(hObject, eventdata, handles)
% hObject    handle to m_scores (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
exportxls(4);

% --------------------------------------------------------------------
function m_loads_Callback(hObject, eventdata, handles)
% hObject    handle to m_loads (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
exportxls(5);

% --------------------------------------------------------------------
function m_res_Callback(hObject, eventdata, handles)
% hObject    handle to m_res (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
exportxls(6);

% --------------------------------------------------------------------
function ms_data_Callback(hObject, eventdata, handles)
% hObject    handle to ms_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
exportxls(7);

% --------------------------------------------------------------------
function ms_dep_Callback(hObject, eventdata, handles)
% hObject    handle to ms_dep (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
exportxls(8);

% --------------------------------------------------------------------
function ms_data_ajust_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to ms_data_ajust (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
exportxls(9);

% --------------------------------------------------------------------
function ms_resids_Callback(hObject, eventdata, handles)
% hObject    handle to ms_resids (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
exportxls(10)



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          FUNCIONES DEL SISTEMA                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%funcion para importar los datos de los reportes a la matriz principal
function importdat(hObject, handles, tipo_import, tipo_report)
[handles.import_files, handles.path_files] = uigetfile('*.xlsx','Sleccione los archivos a
importar','Multiselect','on');
if size(handles.import_files,2)>1;
   f = waitbar(0,'Please wait...');
   pause(.5)
   waitbar(.33,f,'Loading your data');
   pause(1)
   waitbar(.67,f,'Processing your data');
   pause(1)
   handles.M = import_data(handles.import_files, handles.path_files,tipo_import,
tipo_report);
   M = handles.M;
   waitbar(1,f,'Finishing');
   pause(1)
   handles.M = [];
   M=[];
   handles.popupmenuY.Enable = 'off';
   handles.data_dep.Enable = 'Off';
   delete_all(handles);
   close(f)
   uiwait(msgbox('Operación Completada', 'Success','modal'));
   guidata(hObject, handles)
```

```
   set(handles.popupmenuY,'callback', 'FDIs(''updateAxes'',gcbo,[],guidata(gcbo))')
end
% --------------------------------------------------------------------

%funcion que sirve para importar los datos desde excel a la matriz
%principal
function [A] = import_data(import_files, path_files, tipo_import, tipo_report)
sheet = 2;

A = []; %Inicializa la matriz principal que contendrá todos los datos.

%Verifica si se seleccionó un solo archivo o más archivos.
if isa(import_files,'cell') == 1    %se selecionaron dos o mas archivos
   sizeFiles = size(import_files, 2);
else
   % se seleecionó solo un archivo
   sizeFiles = 1;
   file = char(import_files);
   filename = strcat(char(path_files),char(import_files));
end;

%Se muestra el progreso del proceso de importación de archivos
progress = 0;
b = waitbar(0, 'Importando archivos...');
set(b,'Name','Por favor espere...');
set(b,'WindowStyle','modal');
pause(.5)

for i = 1:sizeFiles %comenzamos el ciclo por cada archivo seleccionado
   %Verificamos si solo se va a importar un archivo o más
   if isa(import_files,'cell') == 1
      file = char(import_files(i));
      filename = strcat(char(path_files),char(import_files(i)));
   end;
   fileimp = strcat('Importando archivo: ', file);
   waitbar(progress,b,fileimp);
   pause(1)

   %leemos el archivo de excel y lo guardamos en el espacio de trabajo
   [num,txt,~] = xlsread(filename,sheet);
   if tipo_report == 2
      num([1:3],:)=[]; %Le quitamos los tres primeros renglones al reporte por hora
```

```
  end

  sizeB = size(num,1); %verifica el numero de registros que contiene el archivo

%   try
    C = []; %Inicializa la matriz provisional que agrega a la matriz A

    % Llenamos la columna del no. de observaciones
    sizeA = size(A,1);
    C(:,1)=sizeA+1:sizeA+sizeB;

    % Obtener los datos de las fechas
    if tipo_report == 1
      txt([1:2],:)=[];                    %Eliminamos los renglones de los rotulos del
      formatIn = 'dd/mm/yyyy';            %reporte (para reporte mensual
    else
      txt([1:4],:)=[];                    %Eliminamos los renglones de los rotulos del
      formatIn = 'dd/mm/yyyy  HH:MM:SS';  %reporte (para reporte mensual
    end
    if size(txt,1)-2 > sizeB %Verificamos que todas las fechas contengan datos validos
      txt([sizeB+1:size(txt,1)],:)=[]; %truncamos todas las fechas que no contengan datos
validos
    end
    fecha = datenum(txt(:,1), formatIn); %disp(size(C,1));disp(size(fecha,1));
    C = [C fecha];
    clear txt; %limpiamos de la memoria la matriz txt

    %Agrega los datos de los rangos
    %Eliminamos las columnas de las variables que no necesitamos del
    %reporte

num(:,[1,3:4,6:9,14:15,17:20,22:41,44:48,50:57,59:68,71,73:77,79:80,82,93,96,98,100:115,1
17:130,145:153,155:158,161,163:170,173:176,183:220])=[];
    %Agregamos los datos a la matriz C
    C = [C num];

    %disp(size(A)); disp(size(C));
    if sizeA == 0;
      A = [C];
    else
      if size(A,2) == size(C,2)
        A = [A; C]; %concatenamos verticalmente
```

```
        else
          txt = strcat('Reporte con inconsistencia ',filename, '. No importado');
          uiwait(msgbox(txt,'Error','error','modal'));
        end;
      end
      progress = i/sizeFiles;
end

%Si se lograron importar datos calculamos los limites de control
if size(A,1) > 0
   %metodo 3-sigma (calculo de los limites de control)
   [LC, Ann] = lc_3s(A);
   %Guardamos la matriz importada y los LC en el archivo de trabajo
   waitbar(1,b,'Importación completada');
   if tipo_import == 1     %Se importaron datos para el modelo
      save('mtrxa.mat','A','LC','Ann');
   else                    %Se importaron datos para validar modelo
      B = A;
      LCb = LC;
      Bnn = Ann;
      save('mtrxb.mat','B','LCb','Bnn');
   end
else
   LC = [];
   Ann = [];
   waitbar(1,b,'No se importaron datos');
end;

pause(1)
close(b)

 % --------------------------------------------------------------------
%funcion para calcular el valor de los limites de control del metodo
%3-sigma
function [LC, Ann] = lc_3s(A)
%eliminamos valores nulos y negativos de la matriz
LC = [];
row=[];

%Depuramos la matriz de datos importados eliminando los registros que
%contienen valores nulos en sus variables.
```

```
[row,~]=find(isnan(A)); %Localizamos los renglones donde tenemos datos nulos en las
variables.
[row] = unique(row); %Eliminamos los renglones(posiciones) repetidos;
A(row,:)=[]; %Elimina los registros que contienen valores nulos en las variables

%Guardamos la matriz sin nulos y ceros en la matriz Ann
Ann = A;

A(:,1:2)=[]; %Quitamos las columnas de la fecha y consecutivo para calculas los LC

novar = size(A,2);
LC(:,1)= 2:novar+1;
for i = 1:novar
   y = [];
   y = A(:,i);
   vallsc = mean(y)+3*std(y);
   vallic = mean(y)-3*std(y);
   LC(i,2) = vallsc;
   LC(i,3) = vallic;
end

% --------------------------------------------------------------------
%funcion que permite graficar la matriz principal
function updateAxes(hObject, eventdata, handles)
xCol = 1;
yCol = get(handles.popupmenuY, 'value');
if yCol == 1
   yCol = 2;
   handles.popupmenuY.Value = 2;
end
[x, y] = readXlsCols(handles.M, xCol, yCol);

%depuramos la matriz y solo con valores validos (sin nulos)
y(isnan(y)) = [];

%Calculo del 3-sigma
LC = handles.LC;
LSC = [];
LIC = [];
b=LC(:,1);
k = find(b==yCol);
vallsc = LC(k,2);
```

```
vallic = LC(k,3);
for i=1:size(y,1)
   LSC = [LSC; vallsc];
   LIC = [LIC; vallic];
end;
%fin 3-sigma

%generamos una matriz x con las observaciones validas que estan en y
sizeX=size(y,1);
x=1:1:sizeX;
x=x';

%graficamos la grafica de observaciones y 3-sigma
plot (handles.axes1, x,y,x,LSC,x,LIC);
legend(handles.axes1,'Observ.','LSC','LIC');

%actualiza titulo y unidades del eje Y
load variables
yAxelabel = var_data(:,3);
graphtitle = var_data(:,4);
ylabel(handles.axes1,yAxelabel(yCol,1));
title(handles.axes1,graphtitle(yCol,1));

%graficamos el histograma con la curva de Gauss
histfit(y);

%Borramos la grafica anterior y graficamos la probabiliad de la
%distribucion normal
cla(handles.axes2,'reset');
probplot(handles.axes2, 'normal',y);

%mostramos la información de la variable y su analisis estadistico
info_var(handles, yCol, yAxelabel)
stat_analisys(handles, y, vallsc, vallic)

% --------------------------------------------------------------------
%funcion que sirve para leer el valor de la columna y renglo de la variable
%del control popup
function [x, y] = readXlsCols(A, xCol, yCol)
x = A(:, xCol);
y = A(:, yCol);
% --------------------------------------------------------------------
```

```
%funcion que sirve para poner el nombre de las variables en el control
%popup
function setPopupmenuString(hObject, eventdata, handles)
filename = handles.filename;
[numbers, colNames] = xlsread(filename,1);
set (hObject, 'string', colNames);
% --------------------------------------------------------------------

function fill_popmnu(hObject, eventdata, handles)
%Cargar los datos de las variables en los cuadros de Texto
load variables
colNam1 = var_data(:,1);
colNam2 = var_data(:,2);
colNames = strcat(colNam1,' : ',colNam2);
set (handles.popupmenuY, 'string', colNames);
delete_all(handles);

% --------------------------------------------------------------------
%funcion que realiza el calculo del analisis estadistico de la variable
function stat_analisys(handles, y, vallsc, vallic)
%Realiza el calculo estadisitico
handles.tstat.ColumnWidth = {143 80};
handles.tstat.ColumnName = {'Medida','Valor'};
medialb{1,1} = 'Tendencia Central';
medialb{2,1} = 'Media';
medialb{3,1} = 'Mediana';
medialb{4,1} = 'Moda';
medialb{5,1} = 'Dispersión';
medialb{6,1} = 'Rango intercuartil';
medialb{7,1} = 'Desviación media absoluta';
medialb{8,1} = 'Máx';
medialb{9,1} = 'Mín';
medialb{10,1} = 'Rango';
medialb{11,1} = 'Desviación standar';
medialb{12,1} = 'Varianza';
medialb{13,1} = 'No. de observaciones(n)';
medialb{14,1} = '3-sigma';
medialb{15,1} = 'LSC';
medialb{16,1} = 'LIC';
medialb{2,2} = mean(y);
medialb{3,2} = median(y);
```

```
medialb{4,2} = mode(y);
medialb{6,2} = iqr(y);
medialb{7,2} = mad(y);
medialb{8,2} = max(y);
medialb{9,2} = min(y);
medialb{10,2} = range(y);
medialb{11,2} = std(y);
medialb{12,2} = var(y);
medialb{13,2} = size(y,1);
medialb{15,2} = vallsc;
medialb{16,2} = vallic;
set (handles.tstat,'data',medialb);

% --------------------------------------------------------------------
%funcion para actualizar la información de la variable que se esta
%analizando
function info_var(handles, yCol, yAxelabel)
handles.tvar.Data = [];
handles.tvar.ColumnWidth = {143 200};
handles.tvar.FontWeight = 'bold';
varlb{1,1} = 'Descripción';
varlb{2,1} = 'Grupo';
varlb{3,1} = 'Sistema';
varlb{4,1} = 'Trip';
varlb{5,1} = 'Unidad medida';
varlb(1,2) = handles.V(yCol,2);
varlb(2,2) = handles.V(yCol,1);
varlb(3,2) = handles.V(yCol,3);
varlb(4,2) = handles.V(yCol,4);
varlb(5,2) = yAxelabel(yCol,1);
set (handles.tvar,'data',varlb);

% --------------------------------------------------------------------
%Funcion para guardar la sesion de trabajo
function save_sesion()
sel_path = uigetdir('default','Seleccione la ubicación donde guardar la sesión');
if sel_path ~=0
   err = 0;
   try
      f = fullfile(sel_path,'\','mtrxa.mat');
      copyfile('mtrxa.mat',f);
      f = fullfile(sel_path,'\','mtrxb.mat');
```

```
      copyfile('mtrxb.mat',f);
      f = fullfile(sel_path,'\','svd.mat');
      copyfile('svd.mat',f);
      f = fullfile(sel_path,'\','pca.mat');
      copyfile('pca.mat',f);
      f = fullfile(sel_path,'\','contrib.mat');
      copyfile('contrib.mat',f);
   catch
      uiwait(msgbox('Faltan algunas matrices de la sesión. Operación cancelada',
'Error','modal'));
      err = 1;
   end
   if err == 0;
       uiwait(msgbox('Operación Completada', 'Success','modal'));
   end
end

% --------------------------------------------------------------------
%Funcion para restaurar la sesion de trabajo
function restore_sesion(hObject, eventdata, handles)
begin_ws();
sel_path = uigetdir('default','Seleccione la ubicación desde donde restaurar la sesión');
if sel_path ~=0
   err = 0;
   try
      f = fullfile(sel_path,'\','mtrxa.mat');
      copyfile(f, 'mtrxa.mat');
      f = fullfile(sel_path,'\','mtrxb.mat');
      copyfile(f,'mtrxb.mat');
      f = fullfile(sel_path,'\','svd.mat');
      copyfile(f,'svd.mat');
      f = fullfile(sel_path,'\','pca.mat');
      copyfile(f,'pca.mat');
      f = fullfile(sel_path,'\','contrib.mat');
      copyfile(f,'contrib.mat');
   catch
      uiwait(msgbox('Faltan algunas matrices de la sesión. Operación cancelada',
'Error','modal'));
      err = 1;
   end
   if err == 0;
      loaddat(hObject, eventdata, handles)
```

```
      uiwait(msgbox('Operación Completada', 'Success','modal'));
   end
end

% --------------------------------------------------------------------
%funcion que sirve para cargar los datos al espacio de trabajo
function loaddat(hObject, eventdata, handles)
f = waitbar(.33,'Cargando los datos');
pause(1)
try
   load('mtrxa.mat');
   A(:,2)=[]; %Eliminamos la columna de la fecha
   handles.M = A;
   handles.LC = LC;
   waitbar(1,f,'Terminando');
   pause(1)
   fill_popmnu(hObject, eventdata, handles);
   load variables
   handles.V = var_data(:,5:8);
   guidata(hObject, handles);
   set(handles.popupmenuY,'callback', 'FDIs(''updateAxes'',gcbo,[],guidata(gcbo))');
   handles.popupmenuY.Enable = 'on';
   handles.data_dep.Enable = 'On';
   handles.save_ses.Enable = 'On';
   handles.save_sesio.Enable = 'On';
catch
   uiwait(msgbox('La matriz no contiene datos','Error','error','modal'));
end;
 close(f);

% --------------------------------------------------------------------
%funcion que depura los datos por el metodo del 3-sigma
function depu_dat3s(hObject, eventdata, handles)

%Cargamos el archivo con la matriz depurada de valores nulos
load('mtrxa.mat', 'Ann');
A = Ann;

f = waitbar(0,'Comenzando la depuración...');
pause(1)

%Calculo del 3-sigma
```

```
waitbar(.50,f,'Calculando los limites de control');
pause(1)
novar = size(A,2)-2;    %Calculamos el numero de variables
col=[];
pb = 0;
for i = 1:novar
   waitbar(pb,f,'Depurando con los limites de control');
   pause(.1)
   row=[];
   col=[];
   y3 = [];
   y3 = A(:,i+2);
   %Calculo del 3-sigma
   if size(y3,1) > 1 %Verificamos que al menos contenga un registro
      %Calculamos lo limites de control superior e inferior
      vallsc = mean(y3)+3*std(y3);
      vallic = mean(y3)-3*std(y3);
      %Depuramos los valores con los LSC y LIC
      [row,col] = find(y3>vallsc |y3<vallic);  %Verificamos que valores estan fuera de los LC en
la variable actual
      A(row,:)=[];  %Depuramos la matriz A con los registros que estan fuera de los LC
   end
   pb =  i / novar;
end
waitbar(1,f,'Proceso terminado');
pause(1)
close(f);
Ann=A; %guardamos la matriz depurada en la matriz Ann
save('mtrxa.mat', 'Ann'); %guardamos la matriz Ann en el archivo mtrxa.mat
A(:,2)=[]; %Eliminamos la columna de la fecha para actualizar las graficas en el espacio de
trabajo
handles.M = A; %Actualizamos la matriz del espacio de trabajo
guidata(hObject, handles);
set(handles.popupmenuY,'callback', 'FDIs(''updateAxes'',gcbo,[],guidata(gcbo))');
updateAxes(hObject, eventdata, handles);
% --------------------------------------------------------------------

%funcion para el calculo de ADJUST_VAR
function fdataj(hObject, eventdata, handles, tipo_data)

f = waitbar(0,'Comenzando el autojustado de datos...');
pause(2)
```

```
if tipo_data == 1
   %autoescalamos las variables
   load('mtrxa.mat','Ann');
   Ann(:,[1:2])=[]; %Eliminamos las columnas de consectuivo y fecha
   [data,meanpart1,stdspart1] = auto(Ann);
   data(isnan(data))=0; %Comprobamos que ne el ajuste no salieran datos tipo NaN
   save('pca.mat','data','meanpart1','stdspart1');
else
   load('mtrxb.mat', 'Bnn');
   load('pca.mat','meanpart1','stdspart1');
   Bnn(:,1:2)=[]; %Elimina la columna de fecha y consecutivo de los datos
   %[newdata,meanpart1,stdspart1] = auto(Bd);
   newdata = scale(Bnn,meanpart1,stdspart1);
   save('pca.mat','newdata','-append');
end

waitbar(1,f,'Proceso terminado');
uiwait(msgbox('Datos ajustados exitosamente','Success','success','modal'));
pause(1)
close(f);

% --------------------------------------------------------------------
% Calcula SVD para mostrar los eigenvalores
function eigenval_table()
FigHandle = pcavar;

% --------------------------------------------------------------------
function gen_model(hObject, eventdata, handles)
%Cargarmos las variables del SVD y definimos parametros
load('svd.mat','v','s','ssq');
load('pca.mat','data');
[m,n] = size(data);

%Seleccionar el numero de componentes para el modelo
prompt = {'Introduzca el número de PC:'};
titulo = 'Entrada';
dims = [1 35];
definput = {'5'};
answer = inputdlg(prompt,titulo,dims,definput);
lv = str2num(answer{1});
```

```
% Calculamos Scores y loads en base al numero de componentes seleccionados
%  Forma el Modelo PCA basado en el numerModel Based on the Number of PCs Chosen
loads  = v(:,1:lv);
scores = data*loads;
%  Calculate the residuals matrix and Q values
if lv < min([m n])
  resmat = (data - scores*loads')';
  res    = (sum(resmat.^2))';
else
  disp('Residuals not caluclated when number of PCs')
  disp(' Equals the number of samples or variables')
  disp(' (res and q = 0)')
  res    = zeros(m,1);
end

%  Calculate Q limit using unused eigenvalues
if lv < min([m n])
  temp = diag(s);
  if n < m
    emod = temp(lv+1:n,:);
  else
    emod = temp(lv+1:m,:);
  end
  th1 = sum(emod);
  th2 = sum(emod.^2);
  th3 = sum(emod.^3);
  h0 = 1 - ((2*th1*th3)/(3*th2^2));
  if h0 <= 0.0
    h0 = .0001;
    disp('  ')
    disp('Warning:  Distribution of unused eigenvalues indicates that')
    disp('          you should probably retain more PCs in the model.')
  end
  q = th1*(((2.33*sqrt(2*th2*h0^2)/th1) + 1 + th2*h0*(h0-1)/th1^2)^(1/h0));
else
  q = 0;
end
str = sprintf('The 99 Percent Q limit is %g',q);
uiwait(msgbox(str,'Success','modal'));

%  Calculate T^2 limit using ftest routine
if m > 300
```

```
  tsq = (lv*(m-1)/(m-lv))*ftest(.01,lv,300);
else
  tsq = (lv*(m-1)/(m-lv))*ftest(.01,lv,m-lv);
end

if lv > 1
%  Calculate the value of T^2 by normalizing the scores to
%  unit variance and summing them up
  temp2 = scores*inv(diag(ssq(1:lv,2).^.5));
  tsqs  = sum((temp2.^2)')';
else
  tsqs  = scores.^2/ssq(1,2);
end
str = sprintf('The 99 Percent T^2 limit is %g',tsq);
uiwait(msgbox(str,'Success','modal'));

uiwait(msgbox('Modelo PCA generado exitosamente','Success','modal'));

%Creamos el archivos de datos del modelo de PCA
save( 'pca.mat','ssq', 'scores', 'loads', 'lv', 'q', 'tsq', 'tsqs', 'res', '-append');

% --------------------------------------------------------------------
%Funcion para seleccionar los datos de la muestra
function dat_sel(hObject, eventdata, handles, tipo_import, tipo_report)
load('pca.mat','meanpart1','stdspart1');
err = 0;
try
   meanpart1;
catch
   err = 1;
   disp('Error');
end
if err == 0
   [files_new, path_files_new] = uigetfile('*.xlsx','Sleccione los archivos a
importar','Multiselect','on');
   if size(files_new,2)>1;
      f = waitbar(0,'Please wait...');
      pause(.5)
      waitbar(.33,f,'Loading your data');
      pause(1)
      waitbar(.67,f,'Processing your data');
      pause(1)
```

```
      import_data(files_new, path_files_new,tipo_import, tipo_report); %Importamos los
datos para validar modelo
      try
         fdataj(hObject, eventdata, handles, tipo_import);
      catch
         uiwait(msgbox('No se pudo cargar los datos','Error','error','modal'));
      end
      waitbar(1,f,'Finishing');
      pause(1)
      close(f)
      uiwait(msgbox('Operación Completada', 'Success','modal'));
   end
else
   uiwait(msgbox('No se han ajustado los datos del modelo. No se puede seleccionar
muestra','Error','modal'));
end

% ---------------------------------------------------------------------
% Funcion para calcular los Scores y las contribuciones de la muestra
function pcapro()
f = waitbar(0,'Validando Modelo vs. Muestra Por favor espere...');
pause(2)

%Cargamos los datos del modelo PCA
load('pca.mat','newdata','loads','ssq');

%Establecemos los parametros
try
   [m,n]     = size(loads);
   [ms,ns]   = size(newdata);

   %Verificamos el numero de variables en los datos nuevos
   if ns ~= m
     error('Number of variables in new data not consistent with loadings')
   end
   newscores    = newdata*loads;

   if ms > m
     resmat = newdata' - loads*loads'*newdata';
   else
     resmat = newdata' - loads*newscores';
   end
```

```
   resids = (sum(resmat.^2))';

   if n > 1
     tsqsn = sum((newscores.^2*inv(diag(ssq(1:n,2))))')';
   else
     tsqsn = (newscores.^2*inv(diag(ssq(1:n,2))));
   end

   waitbar(1,f,'Terminando');
   pause(1)
   close(f)

   save( 'pca.mat','newscores','resids','tsqsn','-append');

   calc_contribs();

   uiwait(msgbox('Operación Completada', 'Success','modal'));

catch
   uiwait(msgbox('La matriz no contiene datos. Verifique información','Error','error','modal'));
   close(f)
end;

% ---------------------------------------------------------------------
function calc_contribs()
f = waitbar(0,'Please wait...');
pause(.5)

waitbar(.33,f,'Loading your data');
pause(2)
load('pca.mat','data','newscores','loads','resids','q','newdata')

try
waitbar(.67,f,'Calculando predicciones');
pause(2)

%Calculamos las predicciones
prediction2 = newscores*loads';

% Definimos tamaños
[~,m] = size(data);
n2 = size(newdata,1);
```

```
%Encontrando variables con la mas alta contribución
b = waitbar(0, 'Calculando contribuciones...');
set(b,'Name','Por favor espere...');
set(b,'WindowStyle','modal');
progress = 0;
pause(.5)

% Variable con la mas alta contribución
(q);
k = find(resids >= q);
[row,col] = size(k);
cp_max=[];
for i = 1:row
   waitbar(progress,b,'Calculando contribuciones');
   pause(.01)
   %Algoritmo
   for j = 1:m
     contr(j) = (newdata(k(i,col),j) - prediction2(k(i,col),j))^2;
     contr2(i,j) = contr(j); %Almacenamos las contribuciones en una matriz
   end
   [~,y] = find(contr == max(contr));
   cp_max=[cp_max y];
   %
   progress = i/row;
end
close(b)
pause(1)

%Reconstruccion de variables
c = loads*loads';
rec1 = zeros(m);
for i = 1:m
          rec1(i,:) = [c(1:i-1,i); 0; c(i+1:m,i)]'./(1-c(i,i));
end
rec = (rec1*newdata')';

%Guardando datos en archivo
waitbar(.9,f,'Guardando cálculos');
pause(2)
save('contrib.mat','prediction2','rec','k','contr','contr2','cp_max','y')
waitbar(1,f,'Proceso terminado');
```

```
pause(1)
close(f);

catch
   uiwait(msgbox('La matriz no contiene datos. Verifique información','Error','error','modal'));
   close(f)
end;


%%%%%%%%%%%%%%%%%%
%%%%% GRAFICAS %%%%%%%%
%%%%%%%%%%%%%%%%%%
% --------------------------------------------------------------------
function graph_ajust(tipo_data)

if tipo_data == 1
   load('mtrxa.mat','Ann');
   load('pca.mat','data');
   try
      data;
      Ann;
   catch
      Ann=[];
      data=[];
   end
else
   load('mtrxb.mat','Bnn');
   load('pca.mat','newdata');
   try
      Ann = Bnn;
      data = newdata;
   catch
      Ann=[];
      data=[];
   end
end

if (size(Ann,1) > 0) && (size(data,1) > 0) %Verificamos si la matriz contiene datos
   Ann(:,1:2)=[]; %Elimina la columna de fecha y consecutivo de los datos
   blck1 = Ann; %Guardamos la matriz depurada en la matriz blck1
   %graficamos
   figure ('Name','Autoscaling Data','Color','white','Position',[350 100 590 560])
```

```
   subplot(2,2,[1,2])
   plot(blck1);
   title('Data not Autoscaled')
   xlabel('Sample Number')
   ylabel('Variable Values')

   %figure
   subplot(2,2,[3,4])
   plot(data);
   title('Autoscaled Data for PCA')
   xlabel('Sample Number')
   ylabel('Variable Values')
else
   uiwait(msgbox('La matriz no contiene datos','Error','error','modal'));
end;

% --------------------------------------------------------------------
%funcion para graficar la validacion cruzada
function graph_fcrossva(hObject, eventdata, handles)
load pca.mat;

if size(data,1) > 0 %Verificamos que ya se hayan ajustado los valores
   %validamos con la funcion de cross
   [press,cumpress] = crossval(data,[],'pca','con',58,3);
   figure ('Name','Cross Validation','Color','white','Position',[350 100 690 560])
   semilogy(cumpress,'-ob');
   title('PRESS Results for PCA Model')
   xlabel('Number of PCs')
   ylabel('Cumulative PRESS')
else
   uiwait(msgbox('La matriz no contiene datos','Error','error','modal'));
end

% --------------------------------------------------------------------
%funcion para calcular y graficar los eigen valores
function graph_feigenval(hObject, eventdata, handles)
load pca.mat;

figure ('Name','Eigenvalores','Color','white','Position',[350 100 690 560])

[m,n] = size(data);
if n < m
```

```
  cov = (data'*data)/(m-1);
  [u,s,v] = svd(cov);
  temp2 = (1:n)';
  escl = 1:n;
else
  cov = (data*data')/(m-1);
  [u,s,v] = svd(cov);
  v = data'*v;
  for i = 1:m
    v(:,i) = v(:,i)/norm(v(:,i));
  end
  temp2 = (1:m)';
  escl  = 1:m;
end
mns    = mean(data);
ssqmns = mns*mns';
ssqtot = sum(diag(cov));
if ssqtot/ssqmns < 1e10
  disp('  ')
  disp('Warning: Data does not appear to be mean centered.')
  disp(' Variance captured table should be read as sum of')
  disp(' squares captured.')
end
temp = diag(s)*100/(sum(diag(s)));
ssq  = [temp2 diag(s) temp cumsum(temp)];

%  This section calculates the standard errors of the
%  eigenvalues and plots them
mescl = length(escl);
clf
if mescl<21
  plot(escl,ssq(:,2),'-r',escl,ssq(:,2),'og')
else
  plot(escl(1:58),ssq(1:58,2),'-r',escl(1:58),ssq(1:58,2),'ob')
end
title('Eigenvalue vs. PC Number')
xlabel('PC Number'), ylabel('Eigenvalue')

% --------------------------------------------------------------------
%Grafica Scores por PC
function graph_score_pc()
FigHandle = pcavar2;
```

```
% --------------------------------------------------------------------
%Grafica loads por PC
function graph_loads_pc()
FigHandle = pcavar3;

% --------------------------------------------------------------------
%funcion para graficar el Q residual del modelo
function graph_res_q(hObject, eventdata, handles)
load pca;
load('svd.mat','q');

%  Create the scale vectors to plot against
[m,~] = size(data);
scl = 1:m;
scllim = [1 m];
mlimt = 1101; %limite de m
lim = [q q];

figure('Color','white','Position',[350 100 790 560])
if m < mlimt
   plot(scl,res,'-r',scl,res,'+k',scllim,lim,'--b')
else
   plot(scl,res,'-r',scllim,lim,'--b')
end
str = sprintf('Process SPE with 95 Percent Limit Based on %g PC Model',lv);
title(str), xlabel('Sample Number'), ylabel('SPE')
str = sprintf('The 99 Percent Q limit is %g',q);
uiwait(msgbox(str,'Success','modal'));

% --------------------------------------------------------------------
%Funcion para graficar la T^2 del modelo
function graph_t2(hObject, eventdata, handles)
load pca;

%  Create the scale vectors to plot against
[m,~] = size(data);
scl = 1:m;
scllim = [1 m];
mlimt = 1101; %limite de m

if lv > 1
```

```
   figure ('Name','Value of T^2','Color','white','Position',[350 100 590 560])
   tlim  = [tsq tsq];
   if m < mlimt
     plot(scl,tsqs,'-r',scl,tsqs,'+k',scllim,tlim,'--b')
          else
            plot(scl,tsqs,'-r',scllim,tlim,'--b')
   end
   str = sprintf('Value of T^2 with 99 Percent Limit Based on %g PC Model',lv);
   title(str), xlabel('Sample Number'), ylabel('Value of T^2')
else
   uiwait(msgbox('No se puede graficar con número de PC menor o igual a 1','Error','modal'));
end

str = sprintf('The 99 Percent T^2 limit is %g',tsq);
uiwait(msgbox(str,'Success','modal'));

% --------------------------------------------------------------------
%Funcion para la grafica biplot
function graph_biplot(hObject, eventdata, handles)
load('pca','loads','scores');
load variables;
vbls = var_data(2:59,1);
figure ('Name','Loads, Scores and Variables','Color','white','Position',[350 100 590 560])
str = 'Loads, Scores and Variables';
biplot(loads(:,1:3),'Scores',scores(:,1:3),'Varlabels',vbls);
title(str);

% --------------------------------------------------------------------
%Funcion para graficar los residuales de la muestra vs Q del modelo
function graph_resdq()
load('pca.mat','newdata','resids','q');
%Establecemos los parametros
[ms,~]   = size(newdata);
mlimt     = 1001;
scl       = 1:ms;
scllim    = [1 ms];
lim = [q q];

figure('Color','white','Position',[350 100 790 560])
plot(scl,resids,'-r',scllim,lim,'--b')
if ms < mlimt
   hold on, plot(scl,resids,'+k'),  hold off
```

```
end
title('\fontsize{10} New Sample Residuals with Limits from Old Model')
xlabel('Sample Number')
ylabel('Residual')

% -----------------------------------------------------------------
% Funcion para graficar los scores de la muestra vs T^2 del modelo
function graph_scorest2()
load('pca.mat','loads','newdata','tsq','tsqsn');
%Establecemos los parametros
[~,n]     = size(loads);
[ms,~]   = size(newdata);
mlimt     = 1001;
scl      = 1:ms;
scllim    = [1 ms];

if n == 1
   disp('T^2 not displayed when number of PCs = 1')
else
   figure('Color','white','Position',[350 100 790 560])
   tlim = [tsq tsq];
   plot(scl,tsqsn,'-r',scllim,tlim,'--b')
   if ms < mlimt
     hold 'on', plot(scl,tsqsn,'+k'), hold 'off'
   end
   title('\fontsize{10} New Samples Value of T^2 with 95% Limit From Old Model')
   xlabel('Sample Number')
   ylabel('Value of T^2')
end

% -----------------------------------------------------------------
%Funcion para graficar los scores de la muestra vs los 95% limites del
%modelo
function graph_95limits()
FigHandle = pcapr2;

% -----------------------------------------------------------------
%Funcion para graficar los Scores del modelo vs Scores de la muestra
function graph_scatter3d()
load('pca.mat','scores','newscores');

figure('Color','white','Position',[350 100 690 560])
```

```
hold on
grid on
grid minor
%plot3(scores(:,1),scores(:,2),scores(:,3),'og')
scatter3(scores(:,1),scores(:,2),scores(:,3),'ob')
title('\fontsize{10} Scores on First Three PCs for Old (0) and New (+) Data');
xlabel('Score on First PC from Old Model');
ylabel('\fontsize{10} Score on Second PC from Old Model');
zlabel('\fontsize{10} Score on Third PC from Old Model');
%plot3(newscores(:,1),newscores(:,2),newscores(:,3),'+r'), hold off
scatter3(newscores(:,1),newscores(:,2),newscores(:,3),'+r'), hold off
az = -44;
el = 25;
view(az, el);

% --------------------------------------------------------------------
function graph_contr_var()
load('contrib.mat','cp_max','k');
k = k';

figure('Color','white','Position',[350 100 690 560])
plot(k,cp_max,'+');
title('Variable with highest contribution')
xlabel('Samples')
ylabel('Variables')

% --------------------------------------------------------------------
function graph_spe_contrib()
FigHandle = gcontrib;

%%%%%%%%%%%%%%%%%%
%%%%% HERRAMIENTAS%%%%%%
%%%%%%%%%%%%%%%%%%%
function exportxls(matriz)

filter = {'*.xlsx';'*.csv'};
[file,path,~] = uiputfile(filter,'Seleccione el nombre del archivo para
exportar','export_data.xlsx');
filename = strcat(char(path),char(file));

load variables
```

```
try
f = waitbar(0,'Por favor espere...');
pause(1)
waitbar(.5,f,'Exportando a archivo');
pause(1)
switch matriz
   case 1
      labels = var_data(2:59,9);
      labels = labels';
      load('mtrxa.mat','A');
      A = A(:,3:60);
      xlswrite(filename,labels,1);
      xlswrite(filename,A,1,'A2');
   case 2
      labels = var_data(2:59,10);
      labels = labels';
      load('mtrxa.mat','Ann');
      Ann = Ann(:,3:60);
      xlswrite(filename,labels,1);
      xlswrite(filename,Ann,1,'A2');
   case 3
      labels = var_data(2:59,11);
      labels = labels';
      load('pca.mat','data');
      xlswrite(filename,labels,1);
      xlswrite(filename,data,1,'A2');
   case 4
      load('pca.mat','scores','lv');
      ren = lv + 1;
      labels = var_data(2:ren,12);
      labels = labels';
      xlswrite(filename,labels,1);
      xlswrite(filename,scores,1,'A2');
   case 5
      load('pca.mat','loads','lv');
      ren = lv + 1;
      labels = var_data(2:ren,13);
      labels = labels';
      xlswrite(filename,labels,1);
      xlswrite(filename,loads,1,'A2');
   case 6
      load('pca.mat','res');
```

```
      labels = 'Residuales';
      xlswrite(filename,labels,1);
      xlswrite(filename,res,1,'A2');
   case 7
      labels = var_data(2:59,11);
      labels = labels';
      load('mtrxb.mat','B');
      xlswrite(filename,labels,1);
      xlswrite(filename,B,1,'A2');
   case 8
      labels = var_data(2:59,11);
      labels = labels';
      load('mtrxb.mat','Bnn');
      xlswrite(filename,labels,1);
      xlswrite(filename,Bd,1,'A2');
   case 9
      labels = var_data(2:59,11);
      labels = labels';
      load('pca.mat','newdata');
      xlswrite(filename,labels,1);
      xlswrite(filename,newdata,1,'A2');
   case 10
      load('pca.mat','resids');
      labels = 'Residuales';
      xlswrite(filename,labels,1);
      xlswrite(filename,resids,1,'A2');
end
waitbar(1,f,'Terminando');
pause(1)
close(f);
uiwait(msgbox('Operación Completada', 'Success','modal'));
catch
   uiwait(msgbox('La matriz no contiene datos. Verifique información','Error','error','modal'));
end

% --------------------------------------------------------------------
%Funcion para reiniciar los archivos de las matrices
function begin_ws()
clear all
save mtrxb
save pca
save svd
```

```
save contrib

% --------------------------------------------------------------------
%funcion para reiniciar el espacio de trabajo
function delete_all(handles)
cla(handles.axes1, 'reset');
cla(handles.axes2, 'reset');
cla(handles.axes3, 'reset');
handles.tstat.Data = [];
handles.tvar.Data = [];
handles.tstat.ColumnWidth = {143 80};
handles.tvar.ColumnWidth = {143 200};
handles.tvar.FontWeight = 'bold';
handles.tstat.ColumnName = {'Medida','Valor'};
medialb{1,1} = 'Tendencia Central';
medialb{2,1} = 'Media';
medialb{3,1} = 'Mediana';
medialb{4,1} = 'Moda';
medialb{5,1} = 'Dispersión';
medialb{6,1} = 'Rango intercuartil';
medialb{7,1} = 'Desviación media absoluta';
medialb{8,1} = 'Máx';
medialb{9,1} = 'Mín';
medialb{10,1} = 'Rango';
medialb{11,1} = 'Desviación standar';
medialb{12,1} = 'Varianza';
medialb{13,1} = 'No. de observaciones(n)';
medialb{14,1} = '3-sigma';
medialb{15,1} = 'Media rango móvil';
medialb{16,1} = 'Amplitud';
medialb{17,1} = 'LSC';
medialb{18,1} = 'LIC';
varlb{1,1} = 'Descripción';
varlb{2,1} = 'Grupo';
varlb{3,1} = 'Sistema';
varlb{4,1} = 'Trip';
varlb{5,1} = 'Unidad medida';
set (handles.tstat,'data',medialb);
set (handles.tvar,'data',varlb);

handles.popupmenuY.Value = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Archivo: pcavar.m

```
function varargout = pcavar(varargin)
% Calcula SVD para mostrar los eigenvalores
% PCAVAR MATLAB code for pcavar.fig
%      PCAVAR, by itself, creates a new PCAVAR or raises the existing
%      singleton*.
%
%      H = PCAVAR returns the handle to a new PCAVAR or the handle to
%      the existing singleton*.
%
%      PCAVAR('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PCAVAR.M with the given input arguments.
%
%      PCAVAR('Property','Value',...) creates a new PCAVAR or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before pcavar_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pcavar_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help pcavar

% Last Modified by GUIDE v2.5 15-Jan-2019 18:49:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pcavar_OpeningFcn, ...
                   'gui_OutputFcn',  @pcavar_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before pcavar is made visible.
function pcavar_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pcavar (see VARARGIN)

% Choose default command line output for pcavar
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

load pca.mat;

%Calculamos la descomposición de Valores Singulares
[ssq2] = pcav(data);

set(handles.uitable1, 'data', ssq2); %  Print variance information

% --- Outputs from this function are returned to the command line.
function varargout = pcavar_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


function [ssq2] = pcav(data);
%PCA Principal components analysis
```

```
%  PCA uses svd to perform pca on a data matrix. It is
%  assumed that samples are rows and variables are columns.
%  The input is the data matrix (data). Outputs are the scores
%  (scores), loadings (loads), variance info (ssq), residuals
%  (res), Q limit (reslm), T^2 limit (tsqlm), and T^2's (tsq).
%
%  Optional inputs are (plots) plots = 0 suppresses all plots,
%  plots = 1 [default] produces plots with no confidence limits,
%  plots = 2 produces plots with limits, plots = -1 plots the
%  eigenvalues only (without limits), a vector (scl) for
%  plotting scores against, (if scl = 0 sample numbers will
%  be used), and a scalar (lv) which specifies the
%  number of principal components to use in the model and
%  which suppresses the prompt for number of PCs.
%
%I/O: [scores,loads,ssq,res,reslm,tsqlm,tsq] = pca(data,plots,scl,lvs);
%
%  Note: with plots = 0 and lv specified, this routine requires
%  no interactive user input. If you would like to scale the data
%  before processing use the functions AUTO or SCALE.
%
%See also: EVOLVFA, EWFA, BIGPCA, PCAGUI, PCAPRO, PLTLOADS, PLTSCRS,
%         SCRPLTR, SIMCA, RESMTX, TSQMTX, PCADEMO

%Copyright Eigenvector Research, Inc. 1991-98
%BMW: 11/93,12/94,2/95,5/95,8/97
%NBG: 2/96,3/96,10/96
%Checked on MATLAB 5 by BMW  1/4/97

f = waitbar(0,'Calculando SVD. Por favor espere...');
pause(2)

%Calculo del SVD
[m,n] = size(data);
if n < m
  cov = (data'*data)/(m-1);
  [u,s,v] = svd(cov); %Calcula la descomposicion de valores singulares
  temp2 = (1:n)';
  n2 = 20; %Cambiamos esta linea para mostrar solo 20 componentes
  s2 = s(1:20,1:20); %Cambiamos esta linea para mostrar solo 20 componentes
  temp4 = (1:n2)';
else
```

```
  cov = (data*data')/(m-1);
  [u,s,v] = svd(cov);
  v = data'*v;
  for i = 1:m
    v(:,i) = v(:,i)/norm(v(:,i));
  end
  temp2 = (1:m)';
end

mns    = mean(data);
ssqmns = mns*mns';
ssqtot = sum(diag(cov));

if ssqtot/ssqmns < 1e10
  disp('  ')
  disp('Warning: Data does not appear to be mean centered.')
  disp(' Variance captured table should be read as sum of')
  disp(' squares captured.')
end

temp = diag(s)*100/(sum(diag(s)));
ssq  = [temp2 diag(s) temp cumsum(temp)];

temp3 = diag(s2)*100/(sum(diag(s2)));
ssq2  = [temp4 diag(s2) temp3 cumsum(temp3)];

waitbar(1,f,'Terminando');
pause(1)
close(f)
save('svd.mat','ssq', 'u', 's', 'v');
uiwait(msgbox('Operación Completada', 'Success','modal'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Archivo: pcavar2.m

```
function varargout = pcavar2(varargin)
% Grafica Scores por PC
% PCAVAR2 MATLAB code for pcavar2.fig
%      PCAVAR2, by itself, creates a new PCAVAR2 or raises the existing
%      singleton*.
%
%      H = PCAVAR2 returns the handle to a new PCAVAR2 or the handle to
%      the existing singleton*.
%
%      PCAVAR2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PCAVAR2.M with the given input arguments.
%
%      PCAVAR2('Property','Value',...) creates a new PCAVAR2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before pcavar2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pcavar2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help pcavar2

% Last Modified by GUIDE v2.5 04-Feb-2019 20:23:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pcavar2_OpeningFcn, ...
                   'gui_OutputFcn',  @pcavar2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before pcavar2 is made visible.
function pcavar2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pcavar2 (see VARARGIN)

% Choose default command line output for pcavar2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

fill_popmenu(hObject, eventdata, handles);
graph_scores(hObject, eventdata, handles,1); %grafica el primer componente

% UIWAIT makes pcavar2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = pcavar2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
nocomp=get(hObject,'Value');
graph_scores(hObject, eventdata, handles,nocomp); %grafica el primer componente

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

%funcion para llenar el popup menu con el numero de componentes
function fill_popmenu(hObject, eventdata, handles)
load('pca.mat','lv');
lista = 1:lv;
lista = lista';
set (handles.popupmenu1, 'string', lista);

%funcion para graficar los componentes
function graph_scores(hObject, eventdata, handles,nocomp)
load ('pca.mat','scores','data');
load ('svd.mat','s');
i = nocomp;
[m,~] = size(data);
%  Create the scale vectors to plot against
scl = 1:m;
scllim = [1 m];
temp = [1 1];
pclim = sqrt(s(i,i))*temp*ttestp(.025,m-i,2);

set(handles.figure1,'Name','Sample Scores with 95% Limits','Color','white');
plot(scl,scores(:,i),'-r',scllim,pclim,'--b',scllim,-pclim,'--b')
hold on, plot(scl,scores(:,i),'+g'), hold off
```

```
xlabel('Numero de muestra');
str = sprintf('Score on PC# %g',i);
ylabel(str);
title('Sample Scores with 95% Limits');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Archivo: pcavar3.m

```
function varargout = pcavar3(varargin)
% Grafica loads por PC
% PCAVAR3 MATLAB code for pcavar3.fig
%      PCAVAR3, by itself, creates a new PCAVAR3 or raises the existing
%      singleton*.
%
%      H = PCAVAR3 returns the handle to a new PCAVAR3 or the handle to
%      the existing singleton*.
%
%      PCAVAR3('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PCAVAR3.M with the given input arguments.
%
%      PCAVAR3('Property','Value',...) creates a new PCAVAR3 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before pcavar3_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pcavar3_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help pcavar3

% Last Modified by GUIDE v2.5 04-Feb-2019 20:23:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pcavar3_OpeningFcn, ...
                   'gui_OutputFcn',  @pcavar3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before pcavar3 is made visible.
function pcavar3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pcavar3 (see VARARGIN)

% Choose default command line output for pcavar3
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

fill_popmenu(hObject, eventdata, handles);
graph_loads(hObject, eventdata, handles,1); %grafica el primer componente

% UIWAIT makes pcavar3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = pcavar3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
nocomp=get(hObject,'Value');
graph_loads(hObject, eventdata, handles,nocomp); %grafica el primer componente

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

%funcion para llenar el popup menu con el numero de componentes
function fill_popmenu(hObject, eventdata, handles)
load('pca.mat','lv');
lista = 1:lv;
lista = lista';
set (handles.popupmenu1, 'string', lista);

%funcion para graficar los componentes
function graph_loads(hObject, eventdata, handles,nocomp)
load ('pca.mat','loads','data');
load ('svd.mat','s');
i = nocomp;
[~,n] = size(data);
%  Create the scale vectors to plot against
scl2 = 1:n;
mlimt = 501;

set(handles.figure1,'Name','Variable Number vs. Loadings for PC','Color','white');
if n < mlimt
   plot(scl2,loads(:,i),'-r',scl2,loads(:,i),'og',[1 n],[0 0],'-g')
else
   plot(scl2,loads(:,i),'-r',[1 n],[0 0],'-g')
```

```
end
xlabel('Variable Number')
str = sprintf('Loadings for PC# %g',i);
ylabel(str)
str = sprintf('Variable Number vs. Loadings for PC# %g',i);
title(str)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Archivo: pcapr2.m

```
function varargout = pcapr2(varargin)
%Funcion para graficar los scores de la muestra vs los 95% limites del
%modelo
% PCAPR2 MATLAB code for pcapr2.fig
%      PCAPR2, by itself, creates a new PCAPR2 or raises the existing
%      singleton*.
%
%      H = PCAPR2 returns the handle to a new PCAPR2 or the handle to
%      the existing singleton*.
%
%      PCAPR2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PCAPR2.M with the given input arguments.
%
%      PCAPR2('Property','Value',...) creates a new PCAPR2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before pcapr2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pcapr2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help pcapr2

% Last Modified by GUIDE v2.5 04-Feb-2019 18:22:12

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pcapr2_OpeningFcn, ...
                   'gui_OutputFcn',  @pcapr2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before pcapr2 is made visible.
function pcapr2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pcapr2 (see VARARGIN)

% Choose default command line output for pcapr2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

fill_popmenu(hObject, eventdata, handles);
graph_comp(hObject, eventdata, handles,1); %grafica el primer componente

% UIWAIT makes pcapr2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = pcapr2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
nocomp=get(hObject,'Value');
graph_comp(hObject, eventdata, handles,nocomp); %grafica el primer componente

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

%funcion para llenar el popup menu con el numero de componentes
function fill_popmenu(hObject, eventdata, handles)
load('pca.mat','lv');
lista = 1:lv;
lista = lista';
set (handles.popupmenu1, 'string', lista);

%funcion para graficar los componentes
function graph_comp(hObject, eventdata, handles,nocomp)
load('pca.mat','newdata','ssq','newscores');
%Establecemos los parametros
[ms,~]   = size(newdata);
mlimt    = 1001;
scl      = 1:ms;
scllim   = [1 ms];
temp     = [1 1];

i = nocomp;
pclim = temp*sqrt(ssq(i,2))*1.96;
plot(handles.axes1,scl,newscores(:,i),'-r',scllim,pclim,'--b',scllim,-pclim,'--b')
if ms < mlimt
```

```
   hold(handles.axes1, 'on'), plot(handles.axes1,scl,newscores(:,i),'+g'), hold(handles.axes1,
'off')
end
xlabel(handles.axes1,'Sample Number')
str = sprintf('Score on PC# %g',i);
ylabel(handles.axes1,str)
title(handles.axes1,'\fontsize{10}  New Sample Scores with 95% Limits from Old Model')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Archivo: tdata.m

```
function varargout = tdata(varargin)
% Tabla de datos
% TDATA MATLAB code for tdata.fig
%      TDATA, by itself, creates a new TDATA or raises the existing
%      singleton*.
%
%      H = TDATA returns the handle to a new TDATA or the handle to
%      the existing singleton*.
%
%      TDATA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TDATA.M with the given input arguments.
%
%      TDATA('Property','Value',...) creates a new TDATA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before tdata_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to tdata_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help tdata

% Last Modified by GUIDE v2.5 09-Feb-2019 16:25:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @tdata_OpeningFcn, ...
                   'gui_OutputFcn',  @tdata_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before tdata is made visible.
function tdata_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to tdata (see VARARGIN)

% Choose default command line output for tdata
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
show_table(handles, varargin);

% UIWAIT makes tdata wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = tdata_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Outputs from this function are returned to the command line.
function show_table(handles, tipo_data)

try
   f = waitbar(0,'Por favor espere...');
```

```
    pause(1)
    tipo_data = cell2mat(tipo_data);
    if tipo_data == 1;
       load('dep3s.mat','Ad');
       data = Ad;
    else
       load('dep3s.mat','Bd');
       data = Bd;
    end;
    No = data(:,1);
    fecha = datestr(data(:,2));
    Datos = data(:,[3:60]);
    T = table(No,fecha,Datos);
    waitbar(.50,f,'Generando la tabla.Por favor espere...');
    pause(1)
    writetable(T);
    T = readtable('T.txt');
    T = table2cell(T);
    set(handles.uitable1, 'data', T); %
    waitbar(.9,f,'Terminando...');
    pause(.5)
    close(f);
catch
    close(f);
    uiwait(msgbox('No se puede mostrar tabla, Verifique datos','Error','error','modal'));
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Archivo: gcontrib.m

```
function varargout = gcontrib(varargin)
% Grafica Scores por PC
% GCONTRIB MATLAB code for gcontrib.fig
%      GCONTRIB, by itself, creates a new GCONTRIB or raises the existing
%      singleton*.
%
%      H = GCONTRIB returns the handle to a new GCONTRIB or the handle to
%      the existing singleton*.
%
%      GCONTRIB('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GCONTRIB.M with the given input arguments.
%
%      GCONTRIB('Property','Value',...) creates a new GCONTRIB or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gcontrib_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gcontrib_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gcontrib

% Last Modified by GUIDE v2.5 04-Feb-2019 20:23:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gcontrib_OpeningFcn, ...
                   'gui_OutputFcn',  @gcontrib_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gcontrib is made visible.
function gcontrib_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gcontrib (see VARARGIN)

% Choose default command line output for gcontrib
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

load('contrib.mat','k');
nosample = k(1,1);
fill_popmenu(hObject, eventdata, handles);
graph_contrib(hObject, eventdata, handles,nosample); %grafica la variable con mayor
contribucion

% UIWAIT makes gcontrib wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = gcontrib_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
nosample=get(hObject,'Value');
graph_contrib(hObject, eventdata, handles,nosample); %grafica el primer componente

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

%funcion para llenar el popup menu con el numero de componentes
function fill_popmenu(hObject, eventdata, handles)
load('contrib.mat','k');
set (handles.popupmenu1, 'string', k);

%funcion para graficar los componentes
function graph_contrib(hObject, eventdata, handles,nosample)
load('pca.mat','newdata');
load('contrib.mat','contr','contr2','k','prediction2','y','rec')

[m] = size(contr,2);
[row] = size(k,1);

%Grafica de contribucion al SPE por muestra
contrt = contr2(nosample,:);
subplot(2,2,[1,2])
bar(contrt);
nosamp = k(nosample,1); %Numero de muestra del total de todas las muestras
s = sprintf('Sample %g',nosamp);
```

```
ylabel(s)
xlabel('Variables')
title('Contribution to SPE')

%Grafica de proyeccion
n2 = size(prediction2,1);

novariable = max(contr2(nosample,:));
[r c] = find(novariable == contr2(nosample,:));
subplot(2,2,[3,4])
plot(1:n2,newdata(:,c),'b-')
hold on
plot(1:n2,prediction2(:,c),'k.')
plot(1:n2,rec(:,c),'r-')
hold off
axis([1 n2 min([min(newdata(:,c)) min(prediction2(:,c)) min(rec(:,c))]) max([max(newdata(:,c))
max(prediction2(:,c)) max(rec(:,c))])]);
s = sprintf('Variable %g',c);
ylabel(s);
xlabel('Samples')
title('Data(b), Projection(k), reconstruction(r)')
```