Actas de las XXXVI Jornadas de Automática, 2 - 4 de septiembre de 2015. Bilbao
ISBN 978-84-15914-12-9

Detección De fallos En Una Planta De Tratamiento De Aguas Residuales Mediante PCA Distribuido

A. Sanchez-Fernández, M.J. Fuente y G.I. Sainz-Palmero

Resumen

En este trabajo se propone un método de detección y diagnóstico de fallos para una planta basado en el Análisis de Componentes Principales (PCA). El método se basa en la descomposición de la planta en múltiples bloques según la topología de la instalación en cada uno de esos bloques se calcula un modelo PCA y los resultados se envían al nodo central para fusionar la información y para detectar y diagnosticar fallos en toda la planta. Para poder evaluar su desempeño, se compara éste método con un método PCA centralizado y con algunos métodos de análisis de componentes principales distribuidos (DPCA), aplicándolos a una planta de tratamiento de aguas residuales (EDAR). El objetivo es comprobar cuál de los métodos distribuidos implementados es el mejor en términos de detectar fallos y además minimice el coste de la comunicación entre los bloques. Los resultados de las pruebas realizadas demuestran que el método DPCA basado en modelos locales tiene un rendimiento muy bueno.

Palabras Clave: Detección e identificación de fallos, Análisis de componentes principales distribuido, Estación depuradora de aguas residuales

1. INTRODUCCIÓN

La monitorización on-line de los procesos industriales está cobrando cada vez más importancia para evaluar el rendimiento de los procesos de cara a mejorar su eficiencia, y para garantizar un funcionamiento seguro y eficaz de las instalaciones de una planta industrial. Una detección e identificación rápida de un fallo puede ayudar a evitar costosas averías y graves incidentes. Normalmente, en el seguimiento de procesos se realizan dos tareas: (1) la detección de fallos, que da una indicación de que algo va mal en el proceso, asociado con la falla de algún equipo, el desgaste del mismo, alteraciones en el proceso, etc. y (2) el diagnóstico de fallos que determina la causa del fallo. Hay gran diversidad dentro de los métodos de detección y diagnóstico de fallos, y se pueden clasifican como métodos de detección de fallos basados en datos y métodos basados en modelos.

En la primera categoría, a la que pertenecen métodos como el Análisis de Componentes Principales (PCA) ([1], [2]) o métodos de análisis de componentes independientes (ICA) ([3], [4]), un modelo estadístico implícito se construye a partir de datos del sistema en condiciones normales de funcionamiento y se detectan problemas cuando se observan desviaciones del comportamiento normal. En el enfoque basado en modelos ([5], [6]) se construye un modelo del sistema lo más exacto posible para tener una representación del comportamiento del sistema en condiciones normales. Una vez que se detecta un fallo, es necesario realizar un diagnóstico para poder saber qué es lo que hay que corregir para volver al funcionamiento normal. Este diagnóstico se puede llevar a cabo por diferentes métodos, como el Análisis del discriminante de Fisher ([7], [8]), redes neuronales, diagrama de contribuciones [9], etc.

Para las plantas de pequeño tamaño, en las que el número de sensores es reducido, es razonable suponer que todas las medidas se pueden enviar a un nodo central donde se llevan a cabo la detección y el diagnóstico de fallos para toda la planta. Esto se denomina enfoque centralizado. Pero para un proceso grande o redes a gran escala, resulta más complicado enviar tanta cantidad de datos a una ubicación. En este caso se puede realizar un enfoque distribuido o descentralizado ([10], [11]), en el que el sistema se descompone en bloques y en cada uno de ellos se llevan a cabo cálculos con datos locales, y se envía el resultado al nodo central para que tome una decisión a nivel global.

En este trabajo, se aplica el método PCA para detectar fallos en una planta a gran escala, una planta de tratamiento de aguas residuales [12]. Como el número de variables a supervisar en la planta es muy alto, se utiliza un método de PCA distribuido (DPCA). Con este método la planta se descompone en bloques según la topología de proceso y se genera un método PCA para detectar fallos en cada bloque usando sólo información local. Después, la información obtenida a nivel local se fusiona en el nodo central para obtener una decisión global. De cara a comparar este método, se han estudiado otros métodos de PCA distribuido, ([13], [14],

ISBN 978-84-15914-12-9

[15], [16]), en los que cada bloque realiza algunos cálculos y envía información local al nodo central, que la procesa de cara a realizar la detección y diagnóstico. Por lo tanto, se va a comparar entre el método PCA distribuido propuesto aquí y otros métodos de PCA distribuido, así como con un PCA global, para ver el rendimiento de cada uno.

El documento está organizado de la siguiente manera. La siguiente sección contiene una breve descripción del método PCA centralizado utilizado para detectar fallos, y después se dará una explicación de los métodos PCA distribuidos utilizados. La sección 4 presenta una breve descripción de la planta de tratamiento de aguas residuales, y en el apartado 5 los métodos propuestos se prueban en la EDAR para obtener una comparación entre todos ellos. Finalmente, en la última sección se presentan las conclusiones y las futuras líneas de trabajo.

2. PCA

Al extraer datos de una planta se obtiene una matriz de datos X, de n filas y m columnas (matriz de n observaciones y m variables observadas), que debe ser normalizada a media 0 y varianza 1: X^{norm}.

Una vez hecho esto, se calcula la matriz de covarianza:

$$S = \frac{1}{(n-1)}(X^{norm})^T X^{norm}$$

Y se descompone en valores singulares:

$$S = V \Lambda V^T$$

Siendo Λ (m x m) una matriz diagonal que contiene los valores singulares de X^{norm} (que son, también, los valores propios de la matriz de covarianza, S), que representan los valores de varianza del conjunto de datos. Las columnas de V (m x m) son los vectores propios. Si se elige un número a de elementos de la diagonal de Λ que representen un porcentaje suficientemente alto (usualmente un 60 % o 70 %) de la varianza total de los datos, obtenemos una matriz Λ_a. Tomando un número a de entre las primeras columnas de V, resulta una matriz P, llamada *Matriz de cargas*. La matriz T, de componentes principales de X, se calcula como:

$$T = X^{norm} * P$$

Para detectar fallos se dispone de las herramientas estadísticas T^2 y Q. El estadístico T^2 se calcula según la expresión siguiente:

$$T^2 = z^T P \Lambda_a^{-1} P^T z$$

Siendo z una nueva observación. Este estadístico da una medida de la variabilidad del proceso capturada por los componentes principales. Si el valor de este parámetro está por debajo de un umbral, se puede decir que el sistema trabaja en condiciones normales. Este umbral, para un nivel de significación α, es:

$$T_\alpha^2 = \frac{n^2-1}{n(n-a)} F_\alpha(a, n-a)$$

donde $F_\alpha(a, n-a)$ es el valor crítico superior de la distribución F de Fischer-Snedecor, con n y $n-a$ grados de libertad.

Para estudiar los $m-a$ valores singulares no contemplados en el modelo PCA, se utiliza el estadístico Q:

$$Q = r^T r$$

donde

$$r = (I - PP^T) z^T$$

siendo r el vector de residuos, calculado como la proyección de la observación z en el espacio de los residuos, I es la matriz identidad. El valor que no debe superar el parámetro Q para que el proceso esté bajo control es:

$$Q_\alpha = g * \chi^2(\alpha, h) \tag{1}$$

con:

$$g = \sigma_Q^2/(2 * \bar{Q}),\ h = (2 * \bar{Q}^2)/\sigma_Q^2 \tag{2}$$

donde $\chi^2(\alpha, h)$ es la función de distribución χ^2 acumulada con h grados de libertad y probabilidad α, $\bar{Q}$ es la media de los valores de Q y σ_Q^2 su varianza.

La alarma saltará para T^2 o Q cuando uno de ellos sobrepase su umbral un número determinado de observaciones consecutivas, indicando que el sistema está funcionando de forma anómala. El número de observaciones consecutivas para activar la alarma debe ser lo suficientemente alto para evitar falsas alarmas pero no ser tan elevado que provoque grandes retrasos en avisar del fallo.

Los valores de los umbrales calculados antes son teóricos, y pueden servir como una primera aproximación, pero es posible obtener unos umbrales más realistas con los datos disponibles. Para ello, se toman los datos del sistema funcionando sin

ISBN 978-84-15914-12-9

fallo y se se busca un valor umbral para cada estadístico, de modo que sólo un $\alpha\,\%$ de las observaciones, ese umbral sea superado. Para este proceso se aplica validación cruzada dando a α un valor del $5\,\%$. Finalmente, se elegirá entre el valor teórico y el obtenido por validación cruzada, el de más alto valor.

Cuando se produzcan estas alarmas, interesa saber qué variable ha contribuido más para provocar el fallo. Esto se puede hacer aplicando el diagrama de contribución [9, 8].

3. MÉTODOS PCA DISTRIBUIDOS

El método presentado en la sección anterior se considera como un método PCA centralizado, en el que todas las medidas de los sensores se envían a un nodo central y se procesan para detectar fallos en toda la planta. En esta sección, se mostrarán métodos PCA distribuidos. El primer paso en todos los métodos es descomponer la planta en diferentes bloques, que realizarán algunos cálculos con datos locales y enviarán cierta información al nodo central con el fin de detectar problemas. Este enfoque necesita una descomposición de la planta en s bloques, que se lleva cabo según la topología de la misma y el conocimiento del proceso.

3.1. Método DPCA con modelos PCA locales

El método PCA distribuido propuesto consiste en descomponer la planta en varios bloques y realizar un modelo local de PCA en cada uno de ellos. De forma que la información que sale de cada bloque es: una señal binaria que indica la existencia o no del fallo, $f_i, i = 1, ..., s$, el tiempo de detección y el análisis de contribuciones (la variable que más contribuye al fallo y el valor de su contribución) con el fin de diagnosticar el fallo. Esa información se envía desde cada bloque al nodo central, y éste toma una decisión global combinando las salidas de cada bloque. En el nodo central se detecta un fallo cuando cualquiera de las variables binarias f_i es igual a uno. Para hacer el diagnóstico, se toman en consideración dos salidas de cada bloque: el tiempo de detección y el valor de la contribución. Se analizan los datos de los bloques que hayan detectado primero el fallo, y se busca, entre ellos, el máximo valor de contribución, que pertenecerá a la variable o variables responsables del mal funcionamiento.

3.2. Análisis de Componentes Principales Colectivo (CPCA)

Esta técnica construye un modelo PCA global a partir de los datos enviados por cada bloque, con un costo de comunicación mínimo [15].

En primer lugar, es necesario normalizar cada matriz de datos local X_i $(i = 1...s)$ para obtener columnas con media cero y varianza unidad. A continuación, se genera en cada bloque un modelo PCA individual, por lo que, si hay s bloques, cada uno tendrá su matriz de cargas P_i, con $i = 1...s$. Con estas matrices, es posible obtener los componentes principales locales: $T_i = X_i P_i$. En el procesador central, se forma una nueva matriz T de la forma:

$$T = [T_1, T_2, ..., T_s] \tag{3}$$

Esta matriz es una transformación lineal de la matriz de datos globales: X: $T = XP$, with:

$$P = \begin{bmatrix} P_1 & 0 & . & . & . & 0 \\ 0 & P_2 & . & . & . & 0 \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & & . \\ 0 & 0 & . & . & . & P_s \end{bmatrix} \tag{4}$$

Es posible tener una aproximación de la matriz de datos global: X: $\hat{X} = TP^T$. Ahora se construye un modelo PCA global usando esta matriz $\hat{X}$. El modelo PCA global permite obtener los umbrales de los estadísticos T^2 y Q, respectivamente.

Ahora, para detectar fallos en línea, cada nueva observación medida en cada bloque se proyecta en el modelo local PCA que le corresponda para obtener los nuevos componentes principales T_i^{nueva}, que se envían al procesador central donde se unen en una nueva matriz T^{nueva}. Esta matriz se transforma en $\hat{X}^{nueva}$, que se utiliza para calcular los valores de los estadísticos T^2 y Q, con las matrices globales. Se detecta una anomalía si alguno de los límites T_a^2 o Q_a se superan.

3.3. Método Merging Distributed PCA (MPCA)

El tercer método [16] también genera modelos locales de PCA en cada bloque y envía algunas matrices al procesador central donde se construye nuevamente un modelo PCA global. Este método sólo se puede utilizar si todos los bloques tienen el mismo número de variables.

En el primer paso, se crea un modelo PCA local para cada bloque, partiendo de datos en condiciones normales de operación. Para ello, cada nodo

ISBN 978-84-15914-12-9

debe normalizar sus datos locales de modo que queden en un rango de $[0, 1]$, utilizando su máximo y mínimo locales. Después, cada columna es normalizada a media cero.

El proceso continúa con la construcción de cada matriz local de covarianza S_i, que es descompuesta usando la descomposición SVD:

$$n_i S_i = U_i \Lambda_i U_i^T \tag{5}$$

for $i = 1...s$. Como siempre. tomando a_i elementos de U_i y de Λ_i, se obtienen las matrices $\tilde{P}_i$ y $\tilde{\Lambda}_i$. Estas se envían al nodo central dónde se usan para obtener una aproximación de la matriz global de covarianza:

$$n\tilde{S} = \sum_{i=1}^{s} \tilde{P}_i \tilde{\Lambda}_i^2 \tilde{P}_i^T + \sum_{i=1}^{s} n_i (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T \tag{6}$$

donde n_i es el número de observaciones en el nodo i, $\tilde{P}_i$ y $\tilde{\Lambda}_i$ son las matrices locales, $\bar{x}_i$ es el vector de medias locales del bloque i y $\bar{x}$ es la media de las medias locales.

y n es la suma del número de variables n_i de cada nodo.

Esta matriz permite el cálculo de un modelo global PCA con a_g componentes principales, es decir, $\tilde{S} = U\Lambda U^T$ y tomando los a_g vectores propios y valores propios más altos de U y Λ, respectivamente, se obtienen la matrices globales P y Λ_{a_g}. Estas matrices P y Λ_{a_g}, junto con la media global $\bar{x}$, se transmiten a cada nodo. Ahora, en cada bloque, es posible calcular los componentes principales $\hat{T} = X_i P$. Además, con el fin de detectar fallos, cuando llega una nueva observación, los procesadores locales usan estas matrices globales (P y λ_{a_g}) para obtener los estadísticos y compararlos con sus respectivos umbrales. Estos umbrales se calculan individualmente en cada nodo tal como se indicó en la sección anterior.

3.4. DPCA method with QR factorization

Otro método [13] utilizado realiza una factorización QR con los datos de cada bloque y después combina cada factorización local en el nodo central, de modo que se obtiene un modelo PCA global. La tarea de detección se lleva a cabo en cada bloque utilizando este PCA global.

Se hace una normalización por columnas de cada matriz de datos local para que los datos estén dentro del rango $[0, 1]$. Después, en cada bloque, se calculan el vector de medias $\bar{x}_i$ y del número de filas n_i de X_i, y se normaliza cada matriz X_i:

$$\bar{X}_i = X_i - e_{n_i} \bar{x}_i^T \tag{7}$$

Luego, se hace la descomposición QR de esta matriz:

$$\bar{X}_i = Q_i^{(0)} R_i^{(0)} \tag{8}$$

En el siguiente paso, cada nodo i, con $s/2 \leq i \leq (s-1)$, envía su $R_i^{(0)}$ al nodo $i-(s/2)$, y todos los nodos envían cada n_i y $\bar{x}_i$ al procesador central. En el nodo i, con $0 \leq i \leq (s/2)$, el método calcula la descomposición:

$$\begin{pmatrix} R_i^{(0)} \\ R_{i+s/2}^{(0)} \end{pmatrix} = Q_i^{(1)} R_i^{(1)} \tag{9}$$

Ahora, cada nodo i, con $s/4 \leq i \leq (s/2)$ envía $R_i^{(1)}$ al nodo $(i-s/4)$. El proceso continuará hasta llegar a $l = log_2(s)$ pasos. En la última etapa, el nodo $i = 0$ lleva a cabo el cálculo:

$$\begin{pmatrix} R_0^{(l-1)} \\ R_1^{(l-1)} \end{pmatrix} = Q_0^{(l)} R_0^{(l)} \tag{10}$$

En el procesador central,se calcula la media global:

$$\bar{x} = \frac{1}{n} \sum_{i=0}^{s-1} n_i \bar{x}_i \tag{11}$$

y se construye y se descompone la siguiente matriz:

$$\begin{pmatrix} \sqrt{n_0}(\bar{x}_0 - \bar{x}) \\ \sqrt{n_1}(\bar{x}_1 - \bar{x}) \\ \cdot \\ \cdot \\ \sqrt{n_{s-1}}(\bar{x}_{s-1} - \bar{x}) \\ R_0^{(l)} \end{pmatrix} = QR \tag{12}$$

La matriz de covarianza es $nS = R^T R$. Si se le aplica la descomposición SVD a la matriz S se puede obtener: $S = U\Lambda V^T$. Ahora las matrices Λ y V se utilizan para calcular el modelo global PCA (P_a and Λ_a, umbrales de T^2 y Q, etc.). Esta información se envía a los bloques, y luego cada uno de ellos calcula sus estadísticos T^2 y Q para cada nueva observación. Se detecta una anomalía si se exceden los límites T_α^2 o Q_α. Cuando esto sucede, la variable responsable del fallo se identifica usando el análisis de la contribuciones [9].

ISBN 978-84-15914-12-9

3.5. Método DPCA con clustering

La siguiente técnica [14] utiliza una factorización QR de forma similar al método anterior para construir un modelo PCA global y la detección de fallos se realiza mediante el agrupamiento de los datos en clusters, de modo que la distancia entre cada cluster y el primer componente principal del modelo determinará si existe fallo o no.

En primer lugar, se divide la instalación en bloques, y cada bloque normaliza su matriz de datos en condiciones normales, al rango $[0, 1]$, usando los máximos y mínimos locales. Aplicando el algoritmo anterior (descomposición QR) se obtiene una matriz global R. Ahora es posible calcular la matriz de covarianza:

$$S = \frac{1}{n} R^T R \tag{13}$$

Después de la descomposición de esta matriz utilizando el método SVD, se calculan los componentes principales. El primer componente principal φ se utiliza como una referencia, de modo que es posible calcular la distancia entre este componente y cualquier vector de datos x_k^i:

$$d_p(x_k^i, \varphi) = \sqrt{\|x_k^i - \bar{x}\|^2 - (\varphi^T (x_k^i - \bar{x})^2)} \tag{14}$$

Ahora, todas las observaciones locales se agrupan en clústers de radio ω_c. En cada nodo, la primera observación se utiliza para crear el primer clúster, con su centroide en las coordenadas de este primer punto, y radio ω_c. Para la siguiente observación, si la distancia entre este vector de datos y el centroide del primer clúster es menor que ω_c, este nuevo punto se añade a ese clúster; si no, se crea un nuevo clúster con la última observación como su centro de gravedad, y el radio ω_c. El procedimiento se repite para las siguientes observaciones. Es importante recordar que cuando se añade un punto a un clúster existente, el centro de gravedad de este clúster cambia, y tiene que ser ajustado al punto medio de todos los puntos contenidos en el clúster, incluyendo el nuevo. Cuando se concluye este proceso, todos los clúster ajustan su radio: cada clúster cambia el radio a la distancia entre su centro y el punto más externo contenido en él. Así que, después de todo, se tiene en cada nodo i, un conjunto de l_i clúster que agrupan todas las observaciones de este bloque, y estos conjuntos representan el sistema de trabajo en condiciones de operación normales. Cada grupo estará representado por su radio r_j^i y su centroide c_j^i.

Ahora, es posible calcular la distancia entre cada clúster C_j^i (para $i = 1...s$, $j = 1...L_i$) y la primera componente principal φ:

$$d(C_j^i, \varphi) = d_p(c_j^i, \varphi) + r_j^i \tag{15}$$

Ahora, el máximo de estos valores: d_{max} se puede utilizar para detectar fallos. De tal forma que cuando se reciban nuevas observaciones y se agrupen en clústers, si la distancia de alguno de ellos está por encima de d_{max}, se detecta una anomalía en la instalación. Esta etapa se desarrollará con ventanas de tiempo deslizantes: cuando se adquiere un nuevo vector de datos, el proceso utilizará este vector y los últimos v vectores para crear un nuevo conjunto de clústers. Entonces, las distancias entre estos clústers y el primer componente principal será comparado con d_{max}. Para la próxima observación, el primer punto de la ventana anterior se descarta, y la nueva observación se incluye.

Cuando se detecta una anomalía en algunos bloques, el número de observaciones incluido en cada clúster que esté por encima del límite se cuenta para cada uno de los bloques de la instalación. Y el bloque con el mayor número de observaciones por encima del umbral se considerará el responsable del fallo, posteriormente, se hará un análisis de la contribución de las variables de ese bloque en el momento de la detección del fallo con el fin de identificar la variable responsable del problema.

4. PLANTA DE TRATAMIENTO DE AGUAS RESIDUALES

El método propuesto en este documento ha sido probado en una planta de tratamiento de aguas residuales simulada, figura 1. Este modelo corresponde al modelo BSM2 (Benchmark Simulation Model 2) desarrollado por los Grupos de Trabajo COST Action 682 y 624, y el IWA Task Group on Benchmarking of control Strategies for WWTP ([17], [12]).

Una planta de tratamiento de aguas residuales (EDAR) es una instalación diseñada para eliminar los productos químicos y biológicos del agua, lo que hace posible utilizar este agua tratada para otros fines. En esta instalación hay diferentes partes (Fig 1.): un clarificador, reactores de fangos activados, un sedimentador (o clarificador secundario), y los elementos de tratamiento de fangos: espesante, digestor, tanque de almacenamiento, unidad de deshidratación, etc. Todos estos elementos están unidos por tuberías, y controlados por válvulas, bombas, etc. El objetivo es reducir la toxicidad y aumentar la calidad del agua.

El modelo de la EDAR utilizado está implementado en Simulink (Matlab), y algunas partes de

ISBN 978-84-15914-12-9

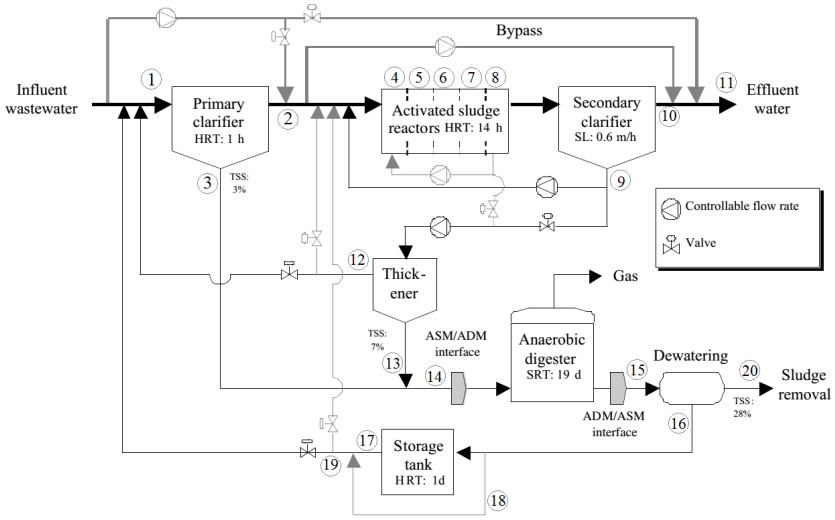

Figura 1: Planta BSM2

este modelo se han cambiado para producir deliberadamente anomalías o fallos en la instalación. El sistema de control integrado no se modificó y se mantiene activo a lo largo de la simulación. El modelo matemático tiene 16 variables de estado como: material inerte soluble, sustrato lentamente biodegradable, nitrato y nitrito, etc. [12] en cada punto de medición (de modo que hay un total de 320 en el sistema, ya que hay 20 puntos de medición, como se ve en Figura 1). Realmente en una planta de aguas residuales no es posible medir todo este conjunto de variables al instante, por lo que en este trabajo se ha trabajado con una serie de variables mucho más fáciles de medir instantáneamente en una planta real (Figura 2). El modelo de simulación se ejecuta a lo largo de 609 días, tomándose medidas cada 15 minutos. Así que al final del proceso se obtiene una matriz de observaciones de 58.464 filas y 7 columnas en cada punto de medición.

Como hay 20 puntos de medición, se ha dividido la instalación en 20 bloques, de modo que en cada uno de estos bloques se medirán las 7 variables definidas en la figura 2. Esto también se hace, debido a que varios métodos DPCA necesitan que todos los bloques de la instalación tengan el mismo número de variables, es decir, todas las matrices X_i deben tener el mismo número de columnas.

4.1. Fallos

En este caso, se han considerado cuatro tipos de fallo, que se han implementado con una magnitud que varía entre el 20 % y 200 %, dando como resultado un conjunto de 16 pruebas con las que probar los diferentes métodos de detección.

Used variables	Variables in the model
COD (Chemical Oxygen demand), g COD.m^{-3}	S_I(inert soluble material), S_S(readily biodegradable substrate), X_I(inert particulate material), X_S(slowly biodegradable substrate)
O_2, g (-COD).m^{-3}	S_O (dissolved oxygen)
Alk (alkalinity)	S_{Alk}(alkalinity)
N (nitrogen), g N.m^{-3}	S_{NO} (nitrate and nitrite), S_{NH} (ammonia and ammonium), S_{ND} (soluble organic nitrogen associated with S_S)
SS (solids suspended), g SS.m^{-3}	TSS (total suspended solids)
Flow, m^3.d^{-1}	Flow rate
Temp, $^{\circ}$C	Temperature

Figura 2: Variables medidas en cada bloque de la instalación

- El primer fallo (F_1) consiste en cambiar el valor medido por el sensor de oxígeno disuelto en el líquido del reactor 4 del bloque de reactores de lodos activados. El controlador de oxígeno, entonces, funciona con entradas erróneas y no introduce la cantidad correcta de oxígeno en los reactores.

- Otro fallo (F_2) consiste en cambiar el valor de la alcalinidad en el líquido que entra en la planta , para simular un cambio en la composición del agua a tratar en la planta.

- El tercer tipo de fallo (F_3) simula un mal funcionamiento de las válvulas de control que hay en la salida del clarificador secundario. El flujo de Q_u se divide en Q_W y Q_r, y el fallo consiste en simular un atasco en uno de los tubos que provoca un aumento de caudal en el otro.

- El último fallo (F_4) consiste en simular la reducción del flujo que circula por una tubería de modo que parezca que hay una fuga. Este fallo fue implementado a la salida del clarifi-

ISBN 978-84-15914-12-9

cador primario, reduciendo el flujo que llega al digestor.

5. RESULTADOS EXPERIMENTALES

Se ha trabajado con seis métodos en este artículo. El primer método es un PCA centralizado utilizado como referencia (que tiene en cuenta conjuntamente las 7 variables medidas en cada uno de los 20 puntos de medición, dando un total de 140 variables). Luego se trabaja con una serie de métodos distribuidos para los cuales se crean 20 bloques, uno para cada punto de medición, en los que se miden 7 variables. Todos los métodos se aplican con un porcentaje del $\alpha = 70\,\%$ de la varianza total representada por sus componentes principales, este valor fue elegido por proporcionar buen rendimiento y un número reducido de componentes principales. Para el método de clustering, se utilizó un radio para los clústers de $w_c = 0,1$.

Se van a medir durante las pruebas, el porcentaje de observaciones anómalas en ausencia de fallo mediante el índice OTI (Overall Type I risk),

$$OTI = \frac{n_f}{I_{NOC}} * 100$$

donde n_f es el número de observaciones por encima del límite cuando no hay fallo, y I_{NOC} es el número total de observaciones también en situación sin fallo. Además, para evitar falsas alarmas en condiciones normales, se impone que sólo si alguna de los estadísticos, sobrepasa los límites durante un cierto número de observaciones consecutivas se activa una alarma. Ese número se debe ajustar para no tener falsas alarmas durante la parte de las simulaciones en la que aún no hay fallo. Esto se mide usando el índice TI (Type I risk),

$$TI = \frac{n_{fa}}{n_s}$$

donde n_{fa} cuenta el número de simulaciones con falsas alarmas en el período de la simulación en el que aún no hay fallos, y n_s es el número de simulaciones realizadas.

Después de probar algunos valores se tomó el valor de 83 observaciones consecutivas por encima del límite de los estadísticos para que salte la alarma. Además, para el método DPCA con clustering, se ha establecido que para que salte la alarma en un bloque debe haber, al menos, un 10 % de los clúster por encima del umbral.

Los resultados obtenidos se agrupan en los cuadros: 1 y 2.

Cuadro 1: Fallos detectados e identificados, y tasa de falsas alarmas

Método	Fallos detectados		Fallos identificados		OTI	
	T^2	Q	T^2	Q	T^2	Q
PCA Centralizado	16	16	9	15	1.16	1.26
PCA Local	16	16	11	15	0.97	0.86
PCA Colectivo	16	15	0	0	3.67	0.84
Merged PCA	16	16	10	13	3.82	5.02
DPCA (QR)	13	16	6	12	4.18	9.56
DPCA & Clustering	16		11		0	

Cuadro 2: Tiempo de detección y OTI

Método	Retraso en la detección (no. observaciones)		TI (% de pruebas)	
	T^2	Q	T^2	Q
PCA Centralizado	1429.8	404.6	0 % (0/16)	6.25 % (1/16)
PCA Local	558.6	5.13	6.25 % (1/16)	6.25 % (1/16)
PCA Colectivo	2063.87	1151.93	0 % (0/16)	0 % (0/16)
Merged PCA	9.6	444.8	62.5 % (10/16)	100 % (16/16)
DPCA (QR)	108.92	166.8	25 % (4/16)	100 % (16/16)
DPCA & Clustering	28.73		0 % (0/16)	

Con estos resultados, es posible afirmar que, en general, el estadístico Q es más eficaz que el T^2, ya que puede identificar más defectos y es más rápido detectándolos. Además, es posible ver que el método distribuido (con modelos locales PCA) propuesto en este artículo es capaz de detectar todas las anomalías, e identificar un $93,75\,\%$ de ellas, con un índice de OTI por debajo del $5\,\%$. Y, además, es el método más rápido de todos a la hora de detectar fallos, con muy bajo número de falsas alarmas. El método centralizado también identifica casi todos los fallos, pero lo hace con un retraso muy grande. El PCA colectivo es incapaz de identificar ningún problema. El método Merged identifica (con Q) bastantes fallos, pero con mucho retraso; y con T^2 identifica menos variables, pero más rápido, aunque da un nivel de falsas alarmas demasiado alto. El método de factorización QR también identifica un buen número de fallos (con Q), pero presenta unos índices OTI y TI muy altos. Finalmente, el método DPCA con clustering, detecta la mayoría de las anomalías, con un tiempo de detección bueno, y sin falsas alarmas.

6. CONCLUSIONES

En este trabajo se propone un método PCA distribuido para detectar e identificar fallos. El método consiste en hacer una descomposición de la planta en bloques generando modelos PCA locales en cada uno de ellos. En caso de detección de un fallo, la información que envía los bloques a un nodo central es: detección de un fallo, instante de la detección y el análisis de contribuciones. En ese

ISBN 978-84-15914-12-9

nodo central se toma una decisión global combinando la información de cada bloque. Este método se ha aplicado a un sistema a gran escala, concretamente, a una planta de tratamiento de aguas residuales.

Se han probado otros métodos para poder comparar con el método propuesto, y es posible concluir que el mejor rendimiento y simplicidad se logra con el PCA distribuido con modelos locales. Ya que es una técnica que trabaja con cálculos sencillos, es rápida en la detección de fallos, y al enviar poca cantidad de información al nodo central, tiene poco costes de comunicación.

Como trabajo futuro, se propone buscar una forma diferente de descomponer la planta en bloques, tratando de mejorar la eficiencia; por ejemplo, agrupando variables en función de la correlación entre ellas o buscando reducir los costes de comunicación al mínimo posible, etc.

Agradecimientos

Los autores desean agradecer a la Comisión Europea y al Ministerio de Economía y Competitividad de España por su apoyo financiero a través del proyecto DPI2012-39381-C02-02.

Referencias

[1] T. Kourti. Process Analysis and Abnormal Situation Detection: From Theory to Practice. *IEEE Control System Magazine*, pages 10–25, 2002.

[2] W. Ku, R.H. Storer, and C. Georgakis. Disturbance detection and isolation by dynamic principal component analysis. *Chemometrics and intelligent laboratory systems*, 30:179–196, 1995.

[3] J.M. Lee, C Yoo, and I.B. Lee. Statistical process monitoring with independent component analysis. *Journal of Process Control*, 14:467–485, 2004.

[4] T. Villegas, M.J. Fuente, and G.I. Sainz-Palmero. Fault diagnosis in a wastewater treatment plant using dynamic independent component analysis. In *18th Mediterranean Conference on Control & Automation (MED)*, 2010.

[5] S.X. Ding. *Model based Fault Diagnosis Techniques*. Springer, 2008.

[6] V. Venkatasubramanian, R. Rengaswamy, S.N. Kavuri, and K. Yin. A review of process fault detection and diagnosis. Part I: Quantitative model-based methods. *Computers & Chemical Engineering*, 27:291–311, 2003a.

[7] Q. He, S. Qin, and J. Wang. A new fault diagnosis method using fault directions in fisher discriminant analysis. *AIChE Journal*, 51(2):555–571, 2005.

[8] D. Garcia-Alvaez, M.J. Fuente, P. Vega, and G.I. Sainz. Fault detection and diagnosis using multivariate statistical techniques in a wastewater treatment plant. In *ADCHEM*, 2009.

[9] T. Kourti and J.F. MacGregor. Multivariate SPC Methods for Process and Product Monitoring. *Journal of Quality Technology*, 28:409–428, 1996.

[10] M. Grbovic, W. Li, P. Xu, A.K. Usadi, L. Somg, and S. Vucetic. Decentralized fault detection and diagnosis via sparse PCA based decomposition and maximum entropy decision fusion. *Journal of Process Control*, 22:738–750, 2012.

[11] Y. Zhang, H. Zhou, S.J. Qin, and T. Chai. Decentralized fault diagnosis of large-scale processes using multiblock kernel partial least squares. *IEEE Transactions on Industrial In*, 6(1):3–10, 2010.

[12] J. Alex, L. Benedetti, J. Copp, K.V. Gernay, U. Jeppsson, I. Nopens, M.N. Pons, C. Rosen, J.P. Steyer, and P. Vanrolleghem. Benchmark: Simulation model no. 2 (bsm2). Technical report, Dept. of Industrial Electrical Engineering and Automation. Lund University., 2008.

[13] Z.J. Bai, R.H. Chan, and F.T. Luk. Principal Component Analysis for distributed data sets with updting. In *6th International Workshoop on advanced parallel processing technologies*, 2005.

[14] M.A. Livani and M. Abadi. Distributed PCA-based anomaly detection in wireless sensor networks. In *2010 IEEE International Conference for Internet Technology and Secure Transactions*, 2010.

[15] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 2001.

[16] Y. Qu, G. Ostrouchov, N. Samatova, and A. Geist. Principal component analysis for dimension reduction in massive distributed data sets. *Proceedings of IEEE International Conference on Data Mining*, 2002.

[17] *The COST Simulation Benchmark: Description and Simulator Manual, http://www.benchmarkwwtp.org.*