Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Corrección de datos basada en modelos dinámicos. Aplicación a medición de caudales

Autor: Guillermo Teno Fernández

Tutor: David Muñoz de la Peña

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Corrección de datos basada en modelos dinámicos. Aplicación a medición de caudales

Autor:

Guillermo Teno Fernández

Tutor:

David Muñoz de la Peña Sequedo

Dep. de Ingeniería de Sistemas y Automática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2015

Índice

Introducción

Medidas de caudales en las redes de distribución

La medición de caudales en tuberías de gran diámetro ha jugado un papel importante en el campo de la instrumentación. Se pueden considerar como grandes tuberías aquellas cuyo diámetro es superior a 1000 mm. El límite superior alcanza en algunas instalaciones los 10 metros. El uso de caudalímetros para este tipo de conducciones aparece en distintas aplicaciones como:

- Conducciones de agua para el abastecimiento de agua potable en núcleos urbanos.
- Circuitos principales de refrigeración de centrales térmicas clásicas y nucleares.
- Entradas y salidas a estaciones depuradoras de aguas residuales.
- Tuberías forzadas para alimentación de grupos hidroeléctricos.

En el contexto de las redes de abastecimiento de agua la medición de caudal tiene las siguientes funcionalidades:

- Cálculo de estadísticos e informes asociados al agua abastecida a una determinada población.
- Facturación de volúmenes de agua aportados.
- Estimación de las pérdidas en la red.
- Monitorización de la red: detección de averías, fraudes, etc.

La funcionalidad de los puntos de medida instalados en una determinada red de abastecimiento se ve comprometida por la precisión de estos. Los errores de medida tienen un impacto directo sobre las distintas funcionalidades arriba mencionadas. Por ejemplo, si se requiere estimar las pérdidas de agua en un tramo de la subred, éstas se pueden estimar como la diferencia de caudales medidos a la entrada y salida del tramo. Los errores de medición entran de forma inexorable en dicho cómputo. De hecho, si los errores de medida son superiores a las pérdidas en el tramo, el resultado de la estimación podría ser que las pérdidas son negativas. De igual forma, la precisión en el cómputo del volumen de agua abastecido a una determinada población vendrá limitado por la precisión de los caudalímetros empleados.

La precisión de un determinado caudalímetro viene determinada por distintos factores. Entre ellos se encuentra la tecnología empleada para su diseño. En los últimos años el desarrollo tecnológico en disciplinas como simulación numérica de flujos, óptica, acústica,

electromagnetismo y electrónica ha implicado una mejora neta en la precisión de dicha instrumentación.

Independientemente de la tecnología empleada para la medición, existe una serie de factores que pueden afectar a la medida realizada. En el contexto de la medición de caudales en redes de abastecimiento se encuentran como factores de influencia:

- Calibración del equipo.
- Temperatura y presión.
- Condiciones de instalación.
- Deformación del perfil de velocidades.
- Presencia de pulsaciones y fluctuaciones en el fluido.
- Presencia de burbujas y sólidos en suspensión.

La calibración del equipo juega un papel esencial en la medida de grandes caudales y presenta una complejidad elevada. La calibración del equipo instalado se ve dificultada por el hecho de que generalmente es imposible comparar el caudal medido con el correspondiente a otro caudalímetro de referencia de mayor precisión. La calibración final del punto de medida está condenada pues a ser parcial e inexacta en muchas ocasiones. Esto tiene un impacto muy negativo en la precisión final del equipo de medida.
El proceso de calibración no se circunscribe exclusivamente al momento de instalación del caudalímetro. Dado que los equipos tienen derivas a largo plazo y la característica instalada puede cambiar con el tiempo se hace necesaria la recalibración periódica del equipo. Esto implica que las dificultades relativas a la calibración del equipo acompañan al mismo durante toda su vida útil (Flowmeter Calibration: How, Why, and Where, J. Yoder, Control for the Process Industry, 2000).

Existe una amplia gama de tipos de caudalímetros (Sensor Technology Handbook, chapter 10, Elsevier 2005):

- Deprimógenos o de presión diferencial.
- Rotámetros.
- Electromagnéticos.
- Ultrasonidos.
- Turbina.
- Vórtex.
- Másico.
- Caudalímetros y Contadores de desplazamiento positivo.

- Caudalímetros de inserción: Sondas tipo tubo de Pitot, tubos tipo Darcy, tubos de Bery, sondas Annubar, etc.

La medida de caudal de agua en las redes de aducción y abastecimiento de agua potable está condicionada por varios factores a saber:

1. Las conducciones de transporte pueden trabajar a lámina libre (canales) o bajo presión (tubería)
2. El tamaño de las conducciones es muy variable, en Emasesa, oscila entre 2.500 mm y 100 mm.
3. La velocidad de circulación del agua en las conducciones se diseña para que sea inferior a 1 m/s, el motivo es para disminuir las pérdidas de carga por rozamiento, optimizando así el consumo de energía necesaria para realizar el transporte del agua.
4. Los caudales medidos cambian en función del horario y de los hábitos de consumo, existiendo relaciones de 12:1 entre caudales punta en la mañana y mínimos nocturnos durante la noche.

Estos factores restringen la selección del tipo de caudalímetro para la medida de agua potable a dos principios de funcionamiento básicos, caudalímetro por ultrasonidos y caudalímetro electromagnético. El resto de principios de funcionamiento se autoexcluyen al no satisfacer los factores anteriormente descritos. Por ejemplo, no son aptos:

- Caudalímetros que restringen el paso del agua por la tubería ya que generan pérdida de carga y obstrucciones, como los elementos deprimógenos tipo Venturi, tubos Pitot, placas de orificio, turbinas o contadores de élices. , etc,.
- Caudalímetros que miden la velocidad del agua en un solo punto, generan incertidumbres de medida no admisibles, como los Vortex, caudalímetros de inserción, caudalímetros por dispersión térmica, molinetes, etc.
- Caudalímetros másicos por efecto Coriolis, debido a su limitación por tamaño, solo se fabrican para diámetro inferiores a 400 mm.
- Secciones de control junto a medidores de nivel, solo se aplican en canales a lámina libre, provocan pérdidas de carga y la relación altura-caudal puede cambiar con el tiempo, debido a las modificaciones en las paredes del canal.

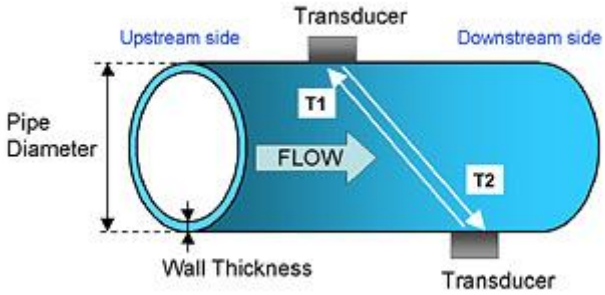
Los dos tipos de caudalímetros más empleado en el tratamiento y transporte del agua potable, son, como se ha visto, los tipos electromagnéticos y ultrasonidos (Baker 2000, Flow measurement handbook : industrial designs, operating principles, performance, and applications, Cambridge University Press, 2000). Sus principales ventajas son:

- Permiten el paso libre del agua en las conducciones sin provocar obstrucciones.
- Se pueden instalar en tuberías de cualquier diámetro, desde 150 mm hasta más de 7 m en el caso de ultrasonidos, y desde 1 mm hasta 2,5 m en el caso electromagnético.
- El coste de montaje de los ultrasonidos es similar para una conducción pequeña o grande.
- Las incertidumbres de medida obtenida son muy buenas si la instalación se realiza correctamente.
- El caudalímetro electromagnético tiene una precisión en la medida mejor que el caudalímetro por ultrasonidos, aunque a un precio más elevado.

Caudalímetros de ultrasonidos

El principio general de método de medida se basa en la variación que se produce en la velocidad del sonido en un fluido medida en un receptor fijo, al variar la velocidad del fluido (Patente 4308754, Norman E. Pedersen, James E. Bradshaw, James E. Matson, Lawrence C. Lynnworth, 1982). Las emisiones de ondas ultrasónicas (1 MHz o más) se realizan desde unas sondas cerámicas que son excitadas por señales senoidales. La misma sonda puede actuar alternativamente como emisor y receptor. Este tipo de medida se aplica tanto en tubería cerrada como en canal abierto. Una de las ventajas de este tipo de caudalímetros es que presentan una pérdida de carga nula y la no existencia de partes móviles. Los diámetros oscilan entre 25 mm hasta 6 y 8 metros de diámetro. La velocidad medible se sitúa en el intervalo de 0,3 hasta 15m/s.

Un equipo con un par de sondas efectúa la medición de la velocidad únicamente en el plano que contiene a las sondas y, por tanto, no calcula la velocidad media del fluido. Por tanto, el caudal no se puede inferir simplemente de la multiplicación de la medida de la velocidad por la sección. De aquí se deduce una importante limitación ya que si el perfil de velocidades está deformado (por ejemplo por la existencia de singularidades aguas arriba), resultará que la velocidad medida por el equipo no es la representativa de la sección. A su vez, si el perfil de velocidades está deformado en el plano en que se efectúa la medición se plantea el mismo problema.



La medición del caudal es más precisa cuando la distancia a la toma del embalse es suficiente como para considerar que el flujo está totalmente desarrollado o establecido. En todas las aplicaciones de caudal, cuando se habla de condiciones ideales de medición se asume flujo totalmente desarrollado.

Dentro de la familia de medidores de caudal de ultrasonidos se consideran dos tipos según su principio de operación:

- Doppler.
- Tiempo de Tránsito.

A su vez, el último tipo abarca otros diferenciados por el tratamiento de la señal empleado para calcular el tiempo de tránsito:

- Dominio temporal: en este caso el tiempo de tránsito se infiere directamente del tiempo transcurrido entre la emisión y recepción de la señal.
- Dominio frecuencial: en este caso la medida del tiempo de tránsito se obtiene de forma indirecta comparando la diferencia de frecuencias de las señales implicadas en la medición.

Los caudalímetros de tipo Doppler se basan en la variación de la frecuencia del sonido, transmitida desde una fuente móvil al ser detectada por un observador fijo. El empleo del equipo implica que el fluido objeto de medición contenga partículas en suspensión o burbujas de aire que reflejen parcialmente el tren de ondas transmitido desde una sonda emisor/receptor o bien desde dos sondas situadas en oposición.

Los caudalímetros de tiempo de tránsito se basan en la disposición de dos sondas enfrentadas, formando un ángulo con el eje de la tubería. En la versión más común las sondas son a la vez emisor y receptor. Cada sonda envía un pulso que es recibido por la opuesta con adelanto o retraso en función de que vaya a favor o contracorriente. Teniendo en cuenta el tiempo de

tránsito, el ángulo de los sensores con el eje de la tubería y el diámetro de la misma es posible inferir la velocidad en el plano que contiene a los sensores.

Estos equipos se adaptan bien a variaciones bruscas de caudal o a flujos pulsantes, ya que la determinación de la velocidad puede realizarse con una frecuencia alta. Una revisión completa de esta tecnología se puede encontrar en (Considine, D.M. 1993. "Ultrasonic Flowmeters," Process/Industrial Instruments & Controls Handbook, 4th Ed., McGraw-Hill, New York, NY:4.115-4.119)(Spitzer, David W. 1990. Industrial Flow Measurement, "Ultrasonic Flowmeters," Instrument Society of America, Research Triangle Park, NC.).
Los inconvenientes de estos caudalímetros son (Integrated Water Meter Management, F. Arregui, E. Cagrera and R. Cobacho, IWA Publishing, 2006):

- Los basados en cómputo de tiempo de tránsito son muy sensibles a la presencia de burbujas o sólidos en suspensión
- Muy sensibles a la presencia de singularidades aguas arriba.
- El posicionamiento de las sondas en diámetros grandes es delicado y require de personal especializado.
- Opacidad de algunos fabricantes a transmitir detalles de cómo funcionan realmente sus equipos. Esto supedita el adecuado mantenimiento y recalibración periódica del equipo a la existencia de un adecuado servicio técnico post-venta.
- Dependencia (en algunos casos) de la celeridad con la presión y la temperatura.

La mejor precisión se obtiene con los caudalímetros basados en tiempo de tránsito y dominio temporal. En condiciones ideales, la precisión de estos varía entre 1% a 3%. Sin embargo, cuando se requiere el concurso tanto de la medida de velocidad como del nivel del conducto para inferir el caudal, la precisión se degrada sustancialmente. Asimismo, estos equipos requieren una calibración especialmente compleja, lo que se traduce en muchas situaciones en un sustancial incremento de los errores de medida.

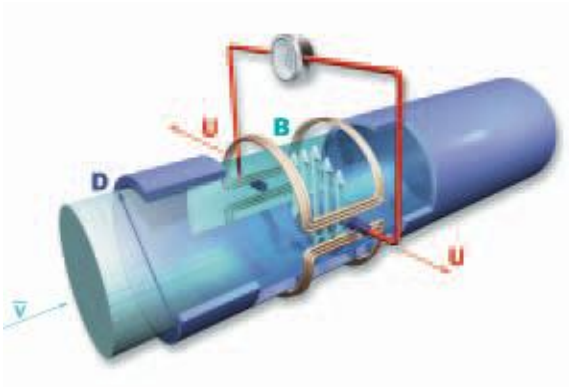
Caudalímetros magnéticos

Se basan en la aplicación de la Ley de Faraday a un líquido conductor. Cuando a un líquido conductor que está en movimiento se le aplica un campo magnético, este genera una fuerza electromotriz que es proporcional a la velocidad.

El campo magnético se genera por medio de dos bobinas dispuestas a lo largo de un tramo de tubería convenientemente aislado y la fuerza electromotriz se detecta por medio de dos

electrodos en contacto con el fluido. La densidad de campo magnético tiene, normalmente un valor fijado para un tamaño dado del caudalímetro.

El caudalímetro electromagnético consta de dos elementos: uno primario y otro secundario. El primario está compuesto de una bobina y de un núcleo de material ferromagnético, el tubo de medida, los electrodos y una derivación para la señal de referencia. La señal que se detecta en los electrodos es extremadamente baja, lo cual es una fuente de error puesto que a caudales bajos es difícil discernir entre señal debida a velocidad de flujo o señal debido a ruido electromagnético. Esta razón es la que hace que la precisión del equipo en términos porcentuales se reduzca a bajos caudales.



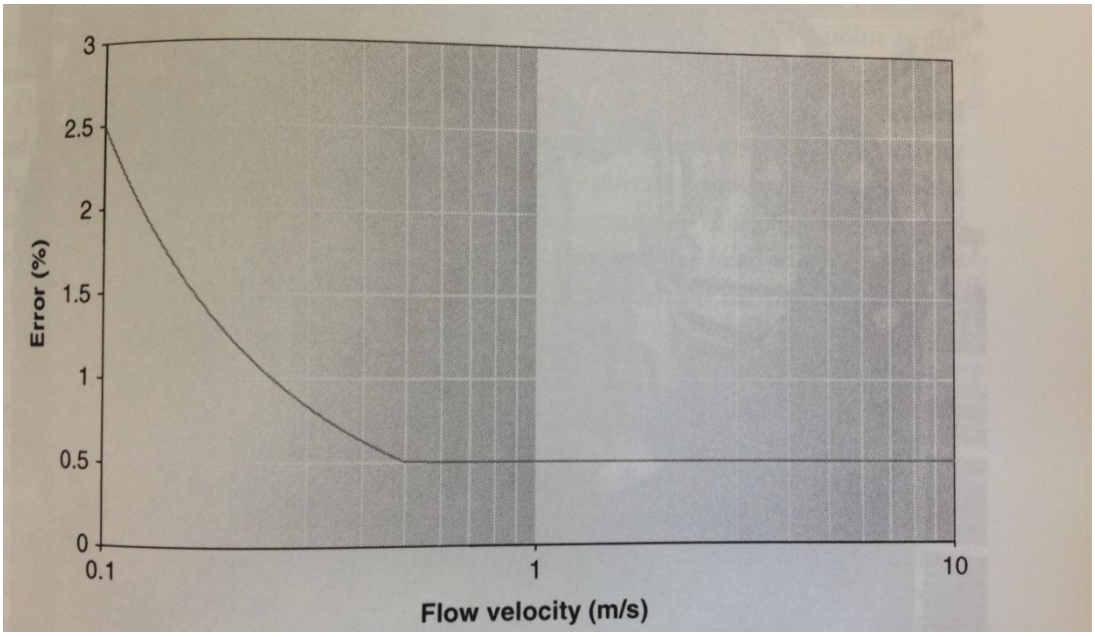
El elemento secundario realiza las siguientes funciones: amplificar y procesar las señales de electrodos y referencia, eliminar señales espurias, compensar las variaciones de tensión y frecuencia, minimizar las variaciones de amplitud del campo magnético en el elemento primario, etc. Los nuevos diseños intentan incorporar bobinas y electrodos que tiendan a reducir el tratamiento de compensación de señales, mejorando su sensibilidad respecto a la distorsión de flujo (Electromagnetic flowmeters: a state of the art review, Hemp, J.; Sanderson, M.L., BHRA fluid engineering conference on flow measuring techniques, Warwick, UK, 9 Sep 1981).

Una ventaja sustancial de este tipo de caudalímetro es que la velocidad de flujo medida abarca completamente el área de conducción por lo que ésta es una velocidad media, directamente proporcional al caudal, sin dependencias de perfil, condiciones hidrodinámicas, etc.

Entre los inconvenientes de este tipo de caudalímetros se encuentran:

- El encastramiento o deposición de sólidos en los electrodos puede modificar las prestaciones e introducir errores.
- Requiere mantenimiento por especialistas del fabricante.

- Precio especialmente elevado en grandes diámetros, que además plantean problemas de calibración.



Curva de error de un caudalímetro magnético en función de la velocidad del fluido.

Introducción a las técnicas de reconciliación de datos

En toda planta química o de procesado o refinado petroquímico, cientos o incluso miles de variables son rutinariamente medidas y automáticamente guardadas con propósito de un proceso de control, optimización online, o evaluación económica continua. Los ordenadores y los sistemas de adquisición de datos modernos facilitan la recolección y el procesamiento de un gran volumen de datos, en ocasiones muestreados con una frecuencia del orden de minutos o incluso segundos.

Además, el uso de ordenadores también permite la eliminación de errores presentes en la recolección manual. Esto ha mejorado la precisión y una consecuente validación del procesamiento de datos. Con esto, somos capaces de mejorar sustancialmente las medidas de los caudalímetros de nuestra red sin necesidad de cambiarlos, lo que supone un ahorro muy importante de dinero, puesto que tanto su coste de compra como el de mantenimiento son bastante elevados.

Los procesos de medida están inevitablemente corrompidos por errores durante la medida, procesamiento y transmisión de la señal medida. El error total tiene una doble contribución, el

error aleatorio y los llamados *gross errors*. Estos errores en la medición de datos pueden provocar fallos en los sistemas de control, falsear procesos de optimización, o incluso llevar el proceso a un punto antieconómico o peligroso.
El error aleatorio implica que ni la magnitud ni el signo del error puede ser predicho con certeza, en otras palabras, si se repite la medición con el mismo instrumento y con idénticas condiciones del proceso, se puede obtener un valor distinto dependiendo del error aleatorio. Estos errores normalmente corresponden a componentes de alta frecuencia en la señal medida, y normalmente son pequeños en magnitud excepto por algunos picos ocasionales.

Los *gross errors* se producen por eventos no aleatorios como un mal funcionamiento del instrumento, un fallo de calibración, humedad o corrosión en los sensores, y restos sólidos. Si se repite la medida con el mismo instrumento bajo condiciones idénticas, la contribución de este error en la medida va a ser la misma. Estos errores se pueden evitar por mantenimiento de las instalaciones. Aunque los *gross errors* ocurren con menos frecuencia, sus magnitudes son normalmente mayores que las de los errores aleatorios.

Investigadores y desarrolladores del área de acondicionamiento de señal han avanzado en el diseño de filtros analógicos y digitales que atenúan el efecto del ruido de alta frecuencia en las medidas. Métodos más sofisticados incluyen *controles de calidad estadísticos* que se utilizan para detectar errores significantes en el procesamiento de datos (outliers). Estos métodos suelen ser aplicados a cada variable de forma separada. Sin embargo, aunque estos métodos mejoran la precisión de las medidas, no utilizan el modelo del sistema y con ello no aseguran una consistencia con los datos ni las interrelaciones de las distintas variables del proceso. De todas formas, estas técnicas deben ser usadas como un primer paso para reducir el efecto de errores aleatorios y grandes *gross errors* en los datos almacenados.

Las técnicas de reconciliación de datos y de detección de *gross errors*, eliminan *gross errors* sistemáticos y errores aleatorios de las bases de datos utilizando las relaciones entre las variables del proceso teniendo en cuenta un modelo del sistema. Ambas técnicas, suelen aplicarse juntas para mejorar la calidad de las medidas.

Las técnicas de reconciliación de datos y de detección de *gross errors* consiguen reducir los errores aprovechando la redundancia en las medidas. En todo proceso, las variables están relacionadas entre ellas por medio de ecuaciones de balance de masa o de energía. A partir de estas restricciones es necesario un mínimo de medidas sin errores para poder calcular los parámetros del sistema y sus variables. Si se supera este mínimo el sistema está sobredeterminado y se puede aplicar la reconciliación de datos, a esta redundancia se la conoce como redundancia espacial. Otro tipo de redundancia existente en las medidas es la redundancia temporal. Este efecto aparece debido a que las medidas se producen continuamente en el tiempo con una alta relación de muestras, produciendo más información de la necesaria para determinar un proceso estático. Si suponemos que el proceso se encuentra en un estado estático podemos aprovechar esta redundancia temporal simplemente realizando una media de las medidas, y aplicamos reconciliación de datos estática para los valores medios. Si el proceso es dinámico, la evolución del proceso se describe por medio de ecuaciones diferenciales correspondientes a balances de masa y de energía, que inherentemente capturan la redundancia temporal y espacial de las variables medidas.

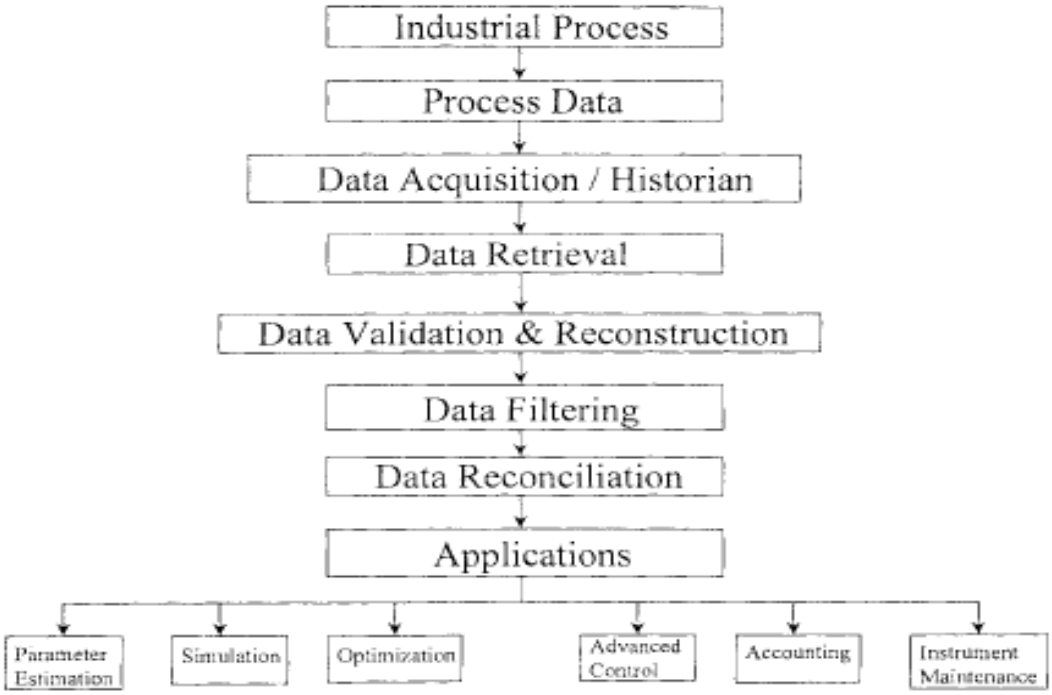


Figure 1-1. Online data collection and conditioning system.

Formulación del problema de reconciliación

La cantidad de ajustes realizadas a las medidas se minimiza puesto que se espera que los errores aleatorios en las medidas sean muy pequeños. En el caso general, no todas las variables del proceso son medidas debido a limitaciones técnicas o económicas. Tanto la estimación de estas variables no medidas como los parámetros del modelo se obtienen a partir del problema de reconciliación. La estimación de estos valores no medidos se conoce como *data coaptation*.

En general, la reconciliación de datos se formula como un problema de optimización por mínimos cuadrados.

$$\underset{x_i,u_j}{\text{Min}} \sum_{i=1}^{n} w_i (y_i - x_i)^2$$

subject to

$$g_k(x_i,u_j) = 0 \qquad k = 1,\ldots, m$$

El término w_i corresponde a los pesos de cada variable, y_i es la medida, x_i es la reconciliación estimada para la variable i, y u_j son las estimaciones de las variables no medidas. La segunda ecuación define el conjunto de restricciones del modelo. Los pesos w_i se eligen dependiendo de la precisión de las diferentes medidas. Las restricciones del modelo suelen ser balances de energía o de masa, pero se pueden incluir inecuaciones impuestas por la factibilidad de las operaciones del proceso. Forzar las medidas para que cumplan relaciones inexactas puede provocar imprecisiones en la solución de la reconciliación de datos y fallos en el diagnóstico de *gross errors.*

Las ecuaciones de balance se pueden expresar de la siguiente forma:

$$entradas - salidas + generado - consumido - acumulado = 0$$

Las magnitudes para las que está definida la anterior ecuación pueden ser flujos de material, flujos de componentes individuales, o el flujo de energía. Si no hay acumulaciones en ninguna de las anteriores magnitudes, las restricciones son algebraicas y se define como una operación estática. Para un proceso dinámico, los términos de acumulaciones no pueden ser despreciados y las restricciones son ecuaciones diferenciales.

Beneficios de la reconciliación de datos y la detección de gross errors

El desarrollo de la reconciliación de datos y la detección de *gross errors* y la implementación practica en un sistema es una tarea difícil y costosa que no se justifica sin unos beneficios considerables para la aplicación en particular. La justificación viene de las grandes aplicaciones que provienen de mejorar el método de adquisición de datos en determinados procesos:

- Una aplicación directa de la reconciliación de datos es la evaluación de los rendimientos de un proceso o en el consumo de recursos en diferentes procesos. Los valores corregidos proporcionan unas estimaciones más fiables que los valores de las medidas raw.
- Aplicaciones como la optimización y simulación de equipos existentes dependen de un modelo del equipo. Estos modelos normalmente contienen parámetros que necesitan ser estimados a través de las bases de datos de la planta. Esto es conocido como *model tuning*, para el cual son esenciales los datos. El uso de medidas erróneas e imprecisas en el *model tuning* puede dar lugar a modelos incorrectos que pueden anular los beneficios procedentes de la optimización.
- La reconciliación de datos puede ser muy útil a la hora de programar el mantenimiento de los equipos del proceso. Por medio de la reconciliación se puede obtener información clave sobre los parámetros del equipo.
- Muchas estrategias de control avanzadas requieren estimaciones de calidad de las variables a controlar. Las técnicas dinámicas de reconciliación de datos se pueden usar para deducir estimaciones de control más precisas.
- La detección de *gross errors* no solo mejora la calidad de las estimaciones procedentes de la reconciliación de datos, sino que también es útil a la hora de identificar problemas en la instrumentación que requieren un mantenimiento especial. Una prematura detección de errores en la instrumentación puede reducir los costes de mantenimiento y un mejor funcionamiento de la planta.

Reconciliación estática

En este apartado se va a proceder a explicar el objeto de un anterior proyecto sobre reconciliación estática de datos aplicada a la red de abastecimiento de EMASESA en Sevilla. La red corresponde a la conducción de agua desde los pantanos de El Gergal y La Minilla hasta la estación de tratamiento del Carambolo y su posterior distribución hacia Sevilla.

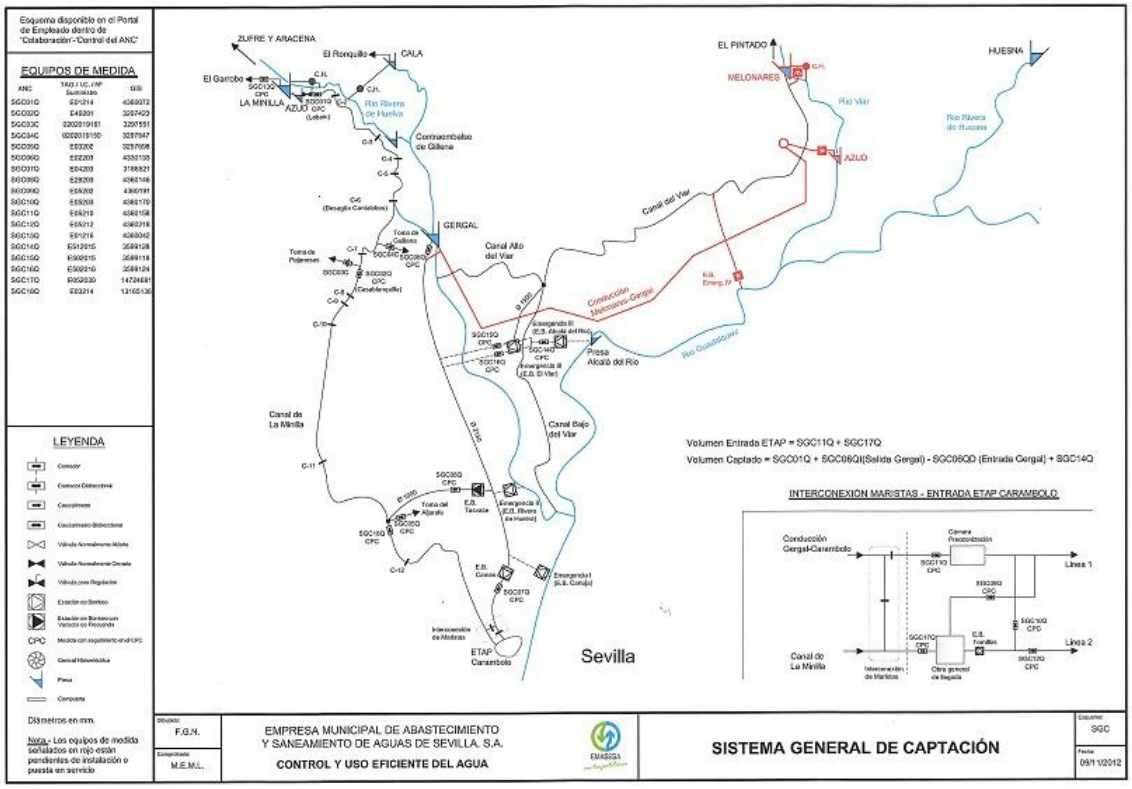


Imagen correspondiente a la conducción de agua desde los pantanos de El Gergal y La Minilla hasta la estación de tratamiento del Carambolo y su posterior distribución hacia Sevilla.

En la subred de abastecimiento de agua de EMASESA considerada se identifican seis ecuaciones de balance de masa. La ecuación j-ésima viene definida por tres conjuntos denotados E_j^e, E_j^s y E_j^a. La descripción de cada uno de dichos conjuntos es la siguiente:

- **Conjunto de flujos entrantes:** Este conjunto, denotado E_j^e, viene dado por los índices de los flujos entrantes.
- **Conjunto de flujos salientes:** Dicho conjunto se denota como E_j^s y viene dado por los índices de los flujos salientes.
- **Conjunto de niveles:** Este conjunto se denota E_j^a y viene dado por los índices de las variables de nivel. Este conjunto es vacío para las ecuaciones de equilibrio de masa donde no sea necesario considerar niveles de depósitos.

Se consideran 6 ecuaciones de balance de masas, que vienen dadas por los conjuntos E_j^e, E_j^s, $E_j^a, j = 1, \dots, 6$. Si t_1 y t_2 son dos instantes temporales expresados en minutos (es decir, t2 –t1 hace referencia al número de minutos transcurridos entre t_1 y t_2), las ecuaciones de balance de masa se obtienen de igualar el volumen entrante obtenido entre el periodo entre t_1 y t_2 con el volumen saliente más el posible volumen acumulado en los depósitos) para ese mismo periodo de tiempo. Las ecuaciones de balance de masa se pueden ser más o menos imprecisas en función del efecto de posibles pérdidas de la red, asimismo las ecuaciones de balance de masa

pueden estar desvirtualizadas debido a la dinámica asociada al transporte de agua en el canal y/o tratamiento en la estación del carambolo.

Cada una de las ecuaciones viene descrita a través de la tabla

DESCRIPCIÓN DE LAS ECUACIONES DE BALANCE DE MASA

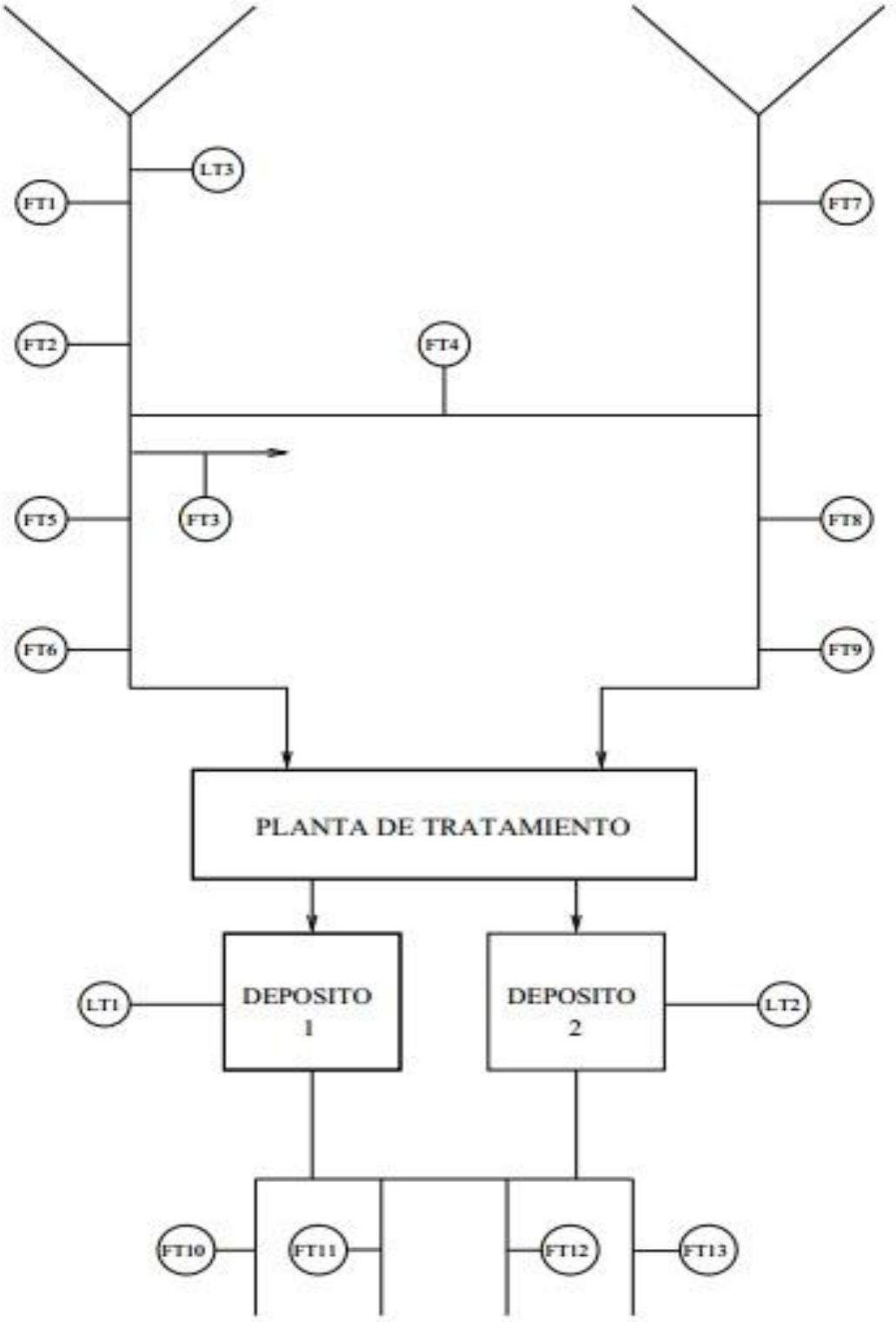
Núm. igualdad	Descripción	Flujos Entrantes	Flujos Salientes	Niveles
1	$\int_{t_1}^{t_2} F_1(t)dt \approx \int_{t_1}^{t_2} F_2(t)dt$	$E_1^e = \{1\}$	$E_1^s = \{2\}$	$E_1^a = \emptyset$
2	$\int_{t_1}^{t_2} (F_2(t) + F_4(t))\, dt \approx \int_{t_1}^{t_2} (F_3(t) + F_5(t))\, dt$	$E_2^e = \{2, 4\}$	$E_2^s = \{3, 5\}$	$E_2^a = \emptyset$
3	$\int_{t_1}^{t_2} F_5(t)dt \approx \int_{t_1}^{t_2} F_6(t)dt$	$E_3^e = \{5\}$	$E_3^s = \{6\}$	$E_3^a = \emptyset$
4	$\int_{t_1}^{t_2} F_7(t)dt \approx \int_{t_1}^{t_2} (F_4(t) + F_8(t))\, dt$	$E_4^e = \{7\}$	$E_4^s = \{4, 8\}$	$E_4^a = \emptyset$
5	$\int_{t_1}^{t_2} F_8(t)dt \approx \int_{t_1}^{t_2} F_9(t)dt$	$E_5^e = \{8\}$	$E_5^s = \{9\}$	$E_5^a = \emptyset$
6	$\int_{t_1}^{t_2} (F_6(t) + F_9(t))\, dt = \int_{t_1}^{t_2} (F_{10}(t) + F_{11}(t) + F_{12}(t) + F_{13}(t))\, dt + S_1 (L_1(t_2) - L_1(t_1)) + S_2 (L_2(t_2) - L_2(t_1))$	$E_6^e = \{6, 9\}$	$E_6^s = \{10, 11, 12, 13\}$	$E_6^a = \{1, 2\}$

Nótese que todas las ecuaciones de equilibrio de masa se pueden escribir en términos de los conjuntos E_j^e, E_j^s y E_j^a. En efecto, todas las ecuaciones se ajustan a la siguiente expresión:

$$\int_{t_1}^{t_2} \left(\sum_{m \in E_j^e} F_m(t) \right) dt \approx \int_{t_1}^{t_2} \left(\sum_{m \in E_j^s} F_m(t) \right) dt + \sum_{m \in E_j^a} S_m(L_m(t_2) - L_m(t_1)) \quad j = 1, \ldots, 6.$$

Nótese que en las ecuaciones de masa se han sustituido los signos de igualdad por los signos de aproximación. Esto es así porque dichas ecuaciones están sujetas a distintas fuentes de error:

- Errores de medida en los caudalímetros.
- Posibles pérdidas en la red.
- Dinámicas de transporte.
- Variaciones de volumen de agua acumulada en la estación de tratamiento de agua.
- Etc.



Red de abastecimiento desde los pantanos La Minilla (izquierda) y El Gergal (derecha) hacia estación Carambolo y posterior distribución.

En lo que sigue, se considerará que los distintos valores t_1 y t_2 considerados para la obtención de las distintas ecuaciones de balance de masa son enteros. Asimismo se considerará que el instante t=1 hace referencia al primer dato del fichero de datos, es decir, a las 00:00 horas del 01/01/2009. Bajo esta asunción, y teniendo en cuenta que los flujos están expresados en metros cúbicos por minuto y la diferencia entre t_1 y t_2 son minutos, las integrales que aparecen en cada una de las ecuaciones de balance de masa se pueden aproximar por sumatorios:

$$\sum_{k=t_1}^{t_2-1}\left(\sum_{m\in E_j^e} F_m(k)\right) \approx \sum_{k=t_1}^{t_2-1}\left(\sum_{m\in E_j^s} F_m(k)\right) + \sum_{m\in E_j^a} S_m(L_m(t_2)-L_m(t_1)) \quad j=1,\ldots,6.$$

A la hora de analizar los posibles errores de medida de los caudalímetros y/o fugas en la red resulta necesario considerar las distintas ecuaciones de balance de masa en distintos intervalos temporales. El número de intervalos N a considerar así como su duración dependerá del análisis particular que se desee realizar. En todo caso, dichos intervalos se pueden considerar almacenados en una matriz T de tantas filas como intervalos temporales se quieran considerar y con dos columnas. La primera columna hace referencia al instante temporal de inicio y la segunda al instante final. Como se ha comentado anteriormente, únicamente se considerarán instantes temporales enteros, es decir, sin decimales. De esta forma, la matriz T tiene la forma:

$$T = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \\ \vdots & \vdots \\ t_{N1} & t_{N2} \end{bmatrix}$$

Por tanto, de la matriz T se pueden obtener las siguientes $6N$ ecuaciones:

$$\sum_{k=t_{i1}}^{t_{i2}-1}\left(\sum_{m\in E_j^e} F_m(k)\right) = \sum_{k=t_{i1}}^{t_{i2}-1}\left(\sum_{m\in E_j^s} F_m(k)\right) + \sum_{m\in E_j^a} S_m(L_m(t_{i2})-L_m(t_{i1})) \quad j=1,\ldots,6;\ \ i=1,\ldots,N.$$

En lo que sigue denotaremos como $v_m^{(i)}$ el volumen de agua en el periodo $[t_{i1}\ t_{i2}]$ asociado al flujo $F_m(\cdot)$. Es decir,

$$v_m^{(i)} = \sum_{k=t_{i1}}^{t_{i2}-1} F_m(k), \quad m=1,\ldots,13.$$

Nótese que estos volúmenes se obtienen directamente de las medidas $F_m(k)$ de los caudalímetros. Así mismo, el volumen de agua acumulado en los depósitos 1 y 2 en el periodo $[t_{i1}\ \ t_{i2}]$ se denotará $v_d^{(i)}$. Esto es:

$$v_d^{(i)} = S_1\big(L_1(t_{i2}) - L_1(t_{i1})\big) + S_2\big(L_2(t_{i2}) - L_2(t_{i1})\big)$$

Con esta notación, las ecuaciones de masa se pueden reescribir como

$$v_1^{(i)} \approx v_2^{(i)}$$

$$v_2^{(i)} + v_4^{(i)} \approx v_3^{(i)} + v_5^{(i)}$$

$$v_5^{(i)} \approx v_6^{(i)}$$

$$v_7^{(i)} \approx v_4^{(i)} + v_8^{(i)}$$

$$v_8^{(i)} \approx v_9^{(i)}$$

$$v_6^{(i)} + v_9^{(i)} \approx v_{10}^{(i)} + v_{11}^{(i)} + v_{12}^{(i)} + v_{13}^{(i)} + v_d^{(i)}$$

Con $i = 1, \dots, N$

Nótese que se están considerando 6 ecuaciones de equilibrio de masa. Podemos definir el error absoluto $e_j^{(i)}$ cometido en el periodo de tiempo *i*-ésimo en la ecuación *j*-ésima (en metros cúbicos) como sigue:

$$e_1^{(i)} = v_1^{(i)} - v_2^{(i)}$$

$$e_2^{(i)} = v_2^{(i)} + v_4^{(i)} - v_3^{(i)} - v_5^{(i)}$$

$$e_3^{(i)} = v_5^{(i)} - v_6^{(i)}$$

$$e_4^{(i)} = v_7^{(i)} - v_4^{(i)} - v_8^{(i)}$$

$$e_5^{(i)} = v_8^{(i)} - v_9^{(i)}$$

$$e_6^{(i)} = v_6^{(i)} + v_9^{(i)} - v_{10}^{(i)} - v_{11}^{(i)} - v_{12}^{(i)} - v_{13}^{(i)} - v_d^{(i)}$$

Con $i = 1, \dots, N$

Dado que las ecuaciones de balance de masa se forman como *suma de volúmenes entrantes igual a suma de volúmenes salientes más acumulados*, los errores vienen dados como *suma de volúmenes entrantes menos suma de volúmenes salientes y acumulados.* Nótese que los términos de pérdida en la red aparecen con signo positivo (la suma de los volúmenes entrantes es superior a la suma de los volúmenes salientes y acumulados). De aquí se infiere que el signo de los errores constituye un indicio de la calidad de las medidas volumétricas. En efecto, discrepancias positivas pueden deberse a términos de pérdidas o a errores en la medida de los caudalímetros. Sin embargo, los errores negativos se deben fundamentalmente a errores de medida puesto que los términos de pérdidas tienen signo positivo. Concluimos pues que la presencia de errores de signo negativo es un claro indicio de errores en la medida de los

caudalímetros asociados (bajo la asunción de que los periodos de tiempo considerados son lo suficientemente grandes como para poder despreciar los errores debidos a la dinámica de transporte).

En la tabla siguiente se presentan los errores absolutos para cada una de las ecuaciones considerando que los intervalos consignados en la matriz *T* corresponden a los meses de enero a junio 2009 (es decir, el intervalo $[t_{11}\ \ t_{12}]$ hace referencia al mes de enero, el intervalo $[t_{21}\ \ t_{22}]$ al mes de febrero y así consecutivamente hasta terminar con el mes de junio). Nótese que la última línea hace referencia al valor medio de dicho error por ecuación. Es decir, a:

$$\frac{1}{N}\sum_{i=1}^{N} e_j^{(i)}$$

ERROR ABSOLUTO CALCULADO POR MESES (EN DECÁMETROS CÚBICOS)

MES /ECUACIÓN	1	2	3	4	5	6
Enero	41.47	96.71	277.02	7.34	-23.05	-474.55
Febrero	-11.56	105.59	73.78	19.79	2.46	-295.44
Marzo	23.34	259.16	-154.05	95.39	51.96	-108.51
Abril	149.72	290.36	-138.80	119.63	60.24	17.77
Mayo	39.65	285.09	54.06	-17.96	9.16	-351.94
Junio	156.81	165.56	98.79	-3.09	-4.79	-437.50
Media	66.57	200.41	35.13	36.85	15.99	-275.03

Se observa en la tabla que los errores asociados a la ecuación 6 son de signo negativo en todos los meses excepto Abril. Debido a que este error es negativo, no se puede buscar una justificación basada exclusivamente en la existencia de posibles pérdidas puesto que esto significaría que se está generando agua de forma espontánea en la red. Asimismo, el intervalo de tiempo considerado es lo suficientemente grande como para que los errores debidos a la dinámica de transporte no jueguen un papel relevante. Se infiere pues que la fuente de error se encuentra fundamentalmente en los errores de medida de los caudalímetros.

También es posible definir un error relativo. Para ello se puede normalizar el error absoluto por una cantidad que refleje el valor medio del volumen entrante en dicho periodo.

Con dicho objetivo se denota m_j el volumen medio de entrada por periodo correspondiente a cada ecuación *j:*

$$m_j = \frac{1}{N}\sum_{i=1}^{N}\sum_{m\in E_j^e} v_m^{(i)}, \quad j = 1,\ldots,6.$$

Esto se traduce en los siguientes volúmenes de entrada medios:

$$\begin{aligned}
m_1 &= \frac{1}{N}\sum_{i=1}^{N} v_1^{(i)} \\
m_2 &= \frac{1}{N}\sum_{i=1}^{N} \left(v_2^{(i)} + v_4^{(i)}\right) \\
m_3 &= \frac{1}{N}\sum_{i=1}^{N} v_5^{(i)} \\
m_4 &= \frac{1}{N}\sum_{i=1}^{N} v_7^{(i)} \\
m_5 &= \frac{1}{N}\sum_{i=1}^{N} v_8^{(i)} \\
m_6 &= \frac{1}{N}\sum_{i=1}^{N} \left(v_6^{(i)} + v_9^{(i)}\right).
\end{aligned}$$

Objetivos del proyecto

A la luz de los resultados del apartado anterior sobre la aplicación de la reconciliación estática sobre la red de EMASESA, se ha propuesto tratar de mejorar los resultados anteriores añadiendo una componente dinámica a la red.

El proyecto anterior estudiaba la red desde un punto de vista puramente estático, las ecuaciones de balance se definían para unos caudales de entrada y salida en un mismo intervalo de tiempo, sin tener en cuenta los momentos anteriores. Estos resultados eran válidos para canales de corto recorrido, en los cuales no había apenas tiempo de transporte del agua desde un sensor de caudal a otro. En cambio, atendiendo al mapa de la red, los primeros canales de agua recorren largas distancias, para lo cual no tiene mucho sentido que se desprecie la dinámica del canal y se establezca una relación estática entre el sensor de salida y de entrada del mismo.

Consideremos un canal de unos 10 km de longitud con dos sensores de caudal colocados a los extremos del mismo. En el proyecto anterior, las ecuaciones de balance establecían que la cantidad de agua que pasaba por el segundo sensor entre un tiempo t_1 y t_2 tenía que ser la misma

cantidad de agua que pasaba por el primer sensor entre esos mismos instantes de tiempo menos una cantidad correspondiente a las pérdidas en el canal. Esto no tiene mucho sentido, porque se desprecia el tiempo de transporte del agua entre los 10 km que separan a ambos sensores, que puede resultar incluso mayor al intervalo de tiempo entre t_1 y t_2.

En este proyecto, con el fin de mejorar los resultados del proyecto anterior, sí se va a tener en cuenta dicho tiempo de transporte, haciendo que en la ecuación de balance, la cantidad de agua que pase por el sensor dos dependa del histórico de medidas del primer sensor y no solo de la medida actual de ese sensor.

$$e_1^{(i)} = v_1^{(i)} - v_2^{(i)} \qquad \text{Ecuación de balance estática}$$

$$e_1^{(k)} = \sum_{i=0}^{m11} \theta_{11}^{i}\, v_1^{(k-LM_{11}-i)} - v_2^{(k)} \qquad \text{Ecuación de balance dinámica}$$

Al despreciar la dinámica, la corrección estática añadía un error adicional debido a que las ecuaciones de balance no eran del todo correctas, y al forzar su cumplimiento se cometía un error de modelado que termina en un resultado que no es del todo correcto.

Teniendo en cuenta la dinámica de cada canal de la red, identificaremos un modelo para cada uno de ellos. Con esto, se pretende mejorar la estimación de la corrección estática, al menos en aquellas ecuaciones de balance donde los caudales que intervengan tengan una dinámica más acusada. Presumiblemente, las ecuaciones con una dinámica más acusada serán las primeras, puesto que los canales que intervienen son los más largos, y con ello tienen un mayor tiempo de transporte del agua.
Una vez identificado el modelo, necesitamos adaptar el algoritmo de optimización, desarrollado por el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Sevilla, para poder corregir las medidas de acuerdo a nuestro modelo identificado. Con ello, compararemos los resultados obtenidos con el modelo dinámico con los obtenidos de forma estática, es decir, sin el modelo.

Primeramente, antes de identificar el modelo, se ha desarrollado en este proyecto un algoritmo para integrar los caudales medidos por los sensores en la red. El algoritmo ha sido programado en lenguaje *C++* empleando *Visual studio 2012 ultimate* y se han utilizando también las librerías de optimización desarrolladas por el Departamento.

Los objetivos del proyecto son:

- Desarrollo de un algoritmo de integración de caudales para obtener los litros totales que ha medido un sensor en un intervalo de tiempo.

- Identificación de un modelo dinámico de los distintos canales de la red de EMASESA.
- Corrección dinámica de los datos obtenidos de integrar los caudales de la red de EMASESA en un intervalo de tiempo.
- Comparación de los resultados de la corrección dinámica con los de la corrección estática.

Este proyecto aporta una solución novedosa para la mejora sistemática de la calidad de las medidas proporcionadas por la instrumentación. Por otra parte, la implantación de esta herramienta, reduciría los costes asociados a la adquisición de nueva instrumentación, ya que permitiría obtener estimaciones volumétricas sustancialmente mejores sin tener que recurrir a la adquisición e instalación de instrumentación de mayor precisión.

Por otra parte, entre las mejoras aportadas por la herramienta a desarrollar, se encuentra posibilidad de llevar a cabo tanto, una estimación de pérdidas en red como un recalibrado automático de la instrumentación. Por último, la herramienta proporcionará de forma automatizada, los distintos informes diarios, semanales y mensuales asociados a los volúmenes transportados en red, incrementando la fiabilidad de los mismos.

El proyecto se ha organizado en cinco partes. En la primera de ellas se hace un repaso de las distintas funciones que componen las librerías *OPLib* y *CWLib*, desarrolladas por el Departamento. La segunda parte corresponde al proceso de identificación del modelo dinámico de las distintas ecuaciones de balance aplicadas a los nodos. En la tercera se explica el método de corrección estática, implementado en el proyecto anterior, y dinámica, desarrollado a lo largo de este proyecto. A continuación, en la cuarta parte, se resume el proceso de aplicación de dichos algoritmos de corrección sobre los datos de la red, y obtención de resultados. Finalmente, en la quinta parte, correspondiente a las conclusiones del proyecto, se discutirán los resultados obtenidos y se valorará el cumplimiento de los objetivos.

Bibliotecas OPLib y CWLib

La mayor parte del proyecto está programa en lenguaje *C++* utilizando *Visual Studio 2012 Ultimate*. En este proyecto se van a manejar unas grandes cantidades de datos almacenados en matrices. Para poder operar con ellos de forma más eficiente es necesario emplear herramientas más potentes que los clásicos *"bucles for",* para movernos entre matrices.

El Departamento de Ingeniería de Sistemas y Automática de la Escuela Superior de Ingenieros de Sevilla, ha desarrollado una librería en *C++* que permite, por medio de la programación orientada a objeto, una forma de operar más sencilla e intuitiva con matrices.

En concreto, se ha desarrollado una clase llamada *"CMatrix"*, que permite crear matrices y operar de forma muy simple con todos los objetos creados de la misma clase. Para poder operar entre las diferentes *"CMatrix"* es necesario definir una serie de funciones dentro de la clase, que son las que nos permiten operar con las matrices de una forma muy parecida a *Matlab*.

Las librerías *"CMatrix"* se encuentran dentro de la biblioteca *OPLib*, donde también otecase encuentran una serie de algoritmos de optimización, que emplearemos a lo largo del proyecto.

Además de la *OPLib*, hay desarrollada una biblioteca de corrección de medidas de caudal basada en modelos estáticos llamada *CWLib*. En este proyecto añadiremos a esta biblioteca la posibilidad de utilizar un nuevo tipo de modelo dinámico para modelar la red. Dentro de la biblioteca *CWLib* se desarrollan las funciones que empleamos para analizar la red, integrar las cantidades de litros, cargar la *network* o definir los problemas de optimización.

A continuación se van a describir las bibliotecas y las funciones más empleadas dentro de ellas.

OPLib

Esta es la biblioteca donde se encuentra definida la clase *CMatrix*, junto con los distintos algoritmos de optimización.

Para definir la clase *CMatrix* hay dos archivos de código fuente, *"CMatrix"* y *"CMatrixPlus"*. El primero de ellos es el que define la clase junto con las ecuaciones de clase, el segundo de ellos, define una serie de funciones cuyo argumento de salida es una *CMatrix*.

El archivo de código de fuente donde se encuentran implementados los distintos algoritmos de optimización se llama *"Optimizacion".*

-CMatrix

Empecemos por definir la clase *CMatrix*. Entre los miembros privados de esta clase se encuentran el número de filas y el número de columnas. Al ser elementos privados de una clase, no se puede acceder directamente a ellos, es necesaria una función pública que te permita modificarlos.

La clase *CMatrix* cuenta con cuatro constructores distintos, dependiendo de cómo declaremos un objeto de la clase, se utilizará uno u otro.

Estos cuatro constructores nos permiten:

-*declarar un conjunto vacío.* Es decir, simplemente declarar un objeto tipo *CMatrix* sin especificar tamaño ni valor de sus elementos. *Ej.- CMatrix A.*

-*declarar una matriz con un tamaño deseado.* Es decir, cuando declaro el objeto tipo *CMatrix*, también le paso las dimensiones especificadas. *Ej.- CMatrix A(nfilas,ncolum)*

-*declarar una matriz con un tamaño especificado y valor de todos sus elementos.* Igual que la anterior pero dándole un valor inicial a todos los elementos de la matriz. *Ej.- CMatrix A(nfilas,ncolumnas,valor_inicial).*

-*copiar una matriz original en otra nueva.* Especificando la matriz tipo *CMatrix* de origen, crea una copia de la original en una nueva. *Ej.- CMatrix A(B).*

También, se han sobrecargado una serie de operadores de manera que se puede operar con las matrices *CMatrix* de una manera muy simple e intuitiva. Por ejemplo, si queremos incrementar todos los elementos de la matriz en un valor simplemente tenemos que poner *(CMatrix) A += 4*. O si queremos hacer una suma o resta de matrices elemento a elemento, simplemente ponemos *C = A + B*. Entendiéndose *A, B y C* como matrices tipo *CMatrix*.

Otros operadores que se han implementado son: la multiplicación y división elemento a elemento de matrices, la multiplicación de matrices, el producto y la división de todos los elementos por un escalar y los operadores de incremento y decremento de todos los elementos en un valor.

Hay una gran cantidad de funciones que están implementadas en *CMatrix*, como por ejemplo una que realiza la descomposición de Cholesky u otra que resuelve un problema de mínimos cuadrados. Las más utilizadas a lo largo de este proyecto han sido:

-unsigned int getNrows() y **unsignedint getNcols()** que nos devuelven el número de filas y de columnas que tiene nuestra matriz.

-inline void resize esta función nos permite cambiar el número de filas y de columnas de nuestra matriz (recordemos que eran parámetros privados), y además asignar si se desea un valor a todos los elementos de la matriz.

-void display(void) esta función se utiliza para imprimir por pantalla nuestra matriz.

Además, hay funciones que permiten sacar una submatriz a partir de una matriz. Con todas estas funciones se consigue un tratamiento de matrices muy parecido a *Matlab*.

-CMatrixPlus

En este archivo se incluyen una serie de funciones cuyo argumento de salida es una función CMatrix.

-CMatrix Eye (const unsigned int dim) matriz identidad de dimensión especificada como argumento de entrada.

-CMatrix TriR (const unsigned int nrows, const unsigned ncols, const double val) matriz triangular superior de dimensión y valor especificados.

-CMatrix TriL (const unsigned int nrows, const unsigned ncols, const double val) matriz triangular inferior de dimensión y valor especificados.

-CMatrix Max (CMatrix& matA, CMatrix matB) matriz que contiene los máximos entre dos matrices comparadas elemento a elemento.

-CMatrix Min (CMatrix& matA, CMatrix matB) matriz que contiene los mínimos entre dos matrices comparadas elemento a elemento.

-CMatrix Max (CMatrix& matA, const double val) matriz con los elementos de otra matriz acotados inferiormente por *val*.

-CMatrix Min (CMatrix& matA, const double val) matriz con los elementos de otra matriz acotados superiormente por *val*.

-CMatrix Abs (CMatrix& mat) matriz con los valores absolutos de los elementos de otra matriz.

-CMatrix Sum (CMatrix& mat) vector fila con los sumatorios de las columnas de otra matriz.

-CMatrix Log (CMatrix& mat) matriz con los logaritmos neperianos de los elementos de otra matriz.

-Optimizacion

En este fichero se encuentran una serie de algoritmos de corrección y optimización de datos. En concreto, a lo largo de este proyecto se ha empleado solo uno, tanto para el problema de corrección estática como para el problema de corrección dinámica.

La función utilizada ha sido:

TOptSolution StrictlyConvexFastGradientCorrection(CMatrix& vecv, CMatrix& matA, CMatrix& vecb, CMatrix& matW, CMatrix& vecLB, CMatrix& vecUB);

Función de corrección de un vector *v* para obtener otro *z* sujeto a *A*z=b, LB<=z<=UB*, usando una aproximación suave para el valor absoluto y un gradiente rápido.

CWLib

En esta biblioteca se define la estructura de la *network*, que es la estructura donde definimos nuestra red. En ella se definen todas las ecuaciones de balance de masas y las variables (sensores) que aparecen en las mismas.

También, es en esta librería donde se convierten los datos de la red a formato matriz y se integran los caudales que pasan por cada sensor para obtener los litros totales que han pasado en un intervalo de tiempo.

Por último, pero no menos importante, en esta librería se definen las funciones que dimensionan los problemas de corrección dinámica y estática, que después se introducirán en el algoritmo de optimización de la biblioteca *OPLib*.

-*Estructuras*

La estructura **Network**, que se explicará en un apartado posterior, es donde se condensa la información de la red. A través de los campos conseguimos definir nuestra red, esto es las ecuaciones de balance de masas con sus variables asociadas. Para poder definir la red la estructura **Network** se apoya de otras dos estructuras, la estructura **Equation** y la estructura **Variable.**

La estructura **Equation** es donde se define cada ecuación, hay una estructura por cada ecuación. En ella se especifica que variables intervienen en cada ecuación, cuales actúan como entrada y cuales como salida, y como están ordenadas.

La estructura **Variable** describe cada variable de nuestra red. Especifica si la variable en cuestión es un sensor de flujo o un sensor de nivel situado en un depósito.

-*Funciones*

Son muchas las funciones incluidas en esta librería, en esta memoria se va a proceder a describir las usadas a lo largo del proyecto.

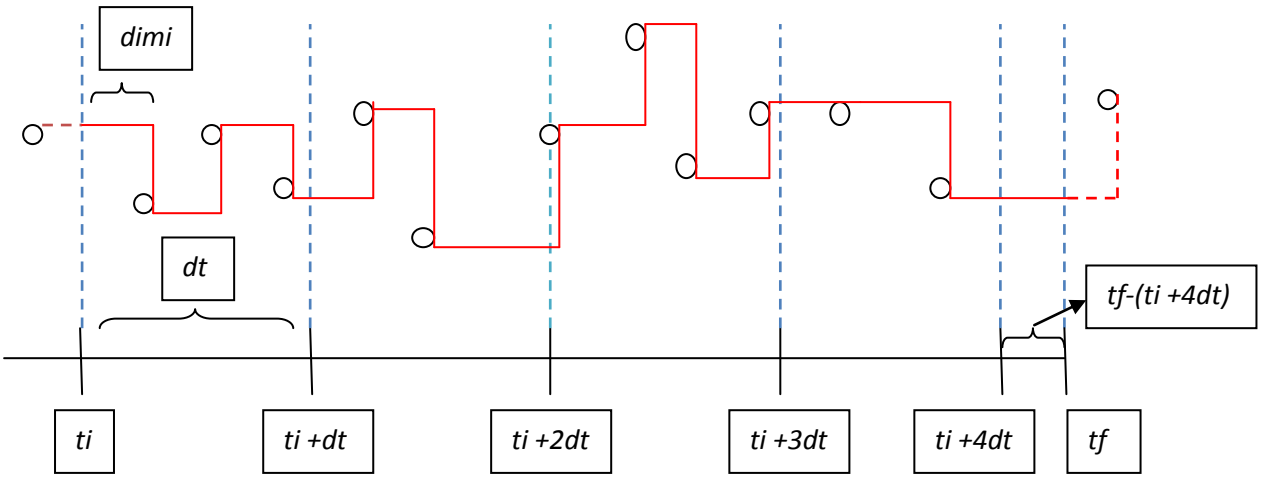
-void integraF(double tk, double tf, double dt, CMatrix& m, CMatrix& mout)

Esta función integra el flujo de litros por segundo que han circulado entre *tf* y *tk* con un paso de integración *dt*, por una variable de caudal (sensor) de la red, obteniendo un resultado en litros.

La matriz *m* contiene los datos en bruto de la variable que queremos integrar, es decir, contiene para cada variable, las medidas que se han tomado y cuando se ha tomado esa medida. Las medidas no están equiespaciadas, es decir, a lo mejor un sensor toma medidas cada cuatro minutos y después las toma cada cinco.

En la matriz *mout* se almacenan las cantidades de litros que han pasado entre dos fechas separadas entre ellas un tiempo *dt*. Al igual que en la matriz anterior, la primera columna corresponde a las fechas en milisegundos, y la segunda columna la medida en litros.

La cantidad de litros que circulan en un tiempo *dt* corresponde al área bajo la línea roja en dicho *dt*.



Cada círculo representa una muestra tomada por el sensor en un instante de tiempo. Como se puede observar, varía el tiempo entre muestras. Estos son los datos que hay en la matriz *m*, las medidas y la fecha en la que fueron tomadas.

Para poder sacar la cantidad de litros en cada *dt*, dividimos nuestro *dt* en trocitos tan anchos como la distancia que haya entre las medidas del sensor dentro de nuestro *dt*, y luego, sumar las áreas de cada trocito.

Para sacar el área de cada trocito tenemos que buscar la medida justo anterior a donde empieza nuestro *dt* (en el caso del primer *dt* implica buscar la mayor medida con un tiempo anterior a *ti*). Una vez localizada esa medida, arrastramos el valor de esa medida hasta la medida siguiente, y calculamos el área del primer trocito como la diferencia de tiempo entre el comienzo del *dt* y la primera medida, multiplicado por el valor de la medida anterior al dt, a este trocito de área se le va a conocer como *dimi*. Los *dimi* siguientes se calculan de forma similar, arrastrando el valor de la medida anterior hasta la siguiente, y multiplicando por la diferencia de tiempo entre medidas. En el último dimi, también se arrastra el valor de la medida anterior, pero el tiempo no se calcula como la diferencia con la siguiente medida, si no que se calcula como la diferencia con el final del *dt* que estamos analizando.

Esto que hemos hecho a pequeña escala dentro de un *dt* se aplica al conjunto total, es posible que entre *ti* y *tf* no haya un número entero de intervalos de integración. Si esto ocurre, el trocito final que queda se calcularía de forma similar a los *dt* anteriores, con la salvedad que en vez de tener una duración de *dt* tendría una duración de *tf-(ti+ndt)*.

-void integraD(double tk,double tf,double dt,CMatrix& m,CMatrix& mout, double& voli)

Esta función saca el número de litros que han pasado entre *tf* y *tk* con un paso de integración *dt*, por una variable correspondiente a un depósito. La función consigue sacar el número de litros midiendo la diferencia de nivel que ha sufrido el depósito en ese diferencial de tiempo. El proceso es similar al anterior, con la salvedad que no necesitamos multiplicar por la diferencia de tiempos, puesto que lo que nos interesa es la diferencia de nivel en los depósitos, es decir, las medidas. Posteriormente, multiplicaremos por el área del depósito para sacar el volumen de agua

-void CWnetworkmodel(Network& network, CMatrix& vecEqu, CMatrix& vecVar, CMatrix& Am, CMatrix& bm, CMatrix& Ac, CMatrix& bc, CMatrix& Tc)

Esta función es la encargada de definir el problema de corrección estático. Se explicará en apartados posteriores.

-void CWnetworkmodelDinamica(Network& network, CMatrix& vecEqu, CMatrix& vecVar, CMatrix& MD, CMatrix& b, CMatrix& A, CMatrix& LB, CMatrix& UB, CMatrix& W, CMatrix& Ws, CMatrix& v,long long ti, long long tf, double Dt)

Esta función, que ha sido elaborada a lo largo del proyecto, es la encargada de definir el problema de corrección dinámica. Es una función mucho más compleja que la anterior y se describirá en apartados posteriores.

-unsigned int CWvarindex(Network& network, IdVar id)

Esta función se utiliza en las funciones que crean los problemas de corrección para poder ordenar las variables en la matriz *"A"* de salida de acuerdo con la *network*.

-int CWreadRawData(BDProcess& instance, Network& network, unsigned int idDatabase, unsigned int idProvider, double ti, double tf, double dt, CMatrix& MI, CMatrix& MD, CMatrix& PD, string& errorCorr)

Cuando se llama a esta función se obtienen los datos de la red en forma de matriz por medio de una función llamada **getDataAsMatrix** (de la librería BDLib). Una vez obtenidos los datos, se integran entre el periodo *ti* y *tf* con un diferencial de tiempo *dt*. Para realizar la integración llama a las funciones **integraF** e **integraD**.

Otras librerías

Aparte de estas dos librería existen 3 librerías más con las que conformamos nuestro proyecto, estas librerías son *TestCWLib, BDLib y XMLGrafica*.

TestCWLib simplemente aplica un ejemplo de red de distribución a la que se le calculan las medidas en litros de la red y luego se corrigen los datos.

BDLib es una librería estática donde se almacenan las bases de datos que vamos a necesitar a la hora de ejecutar el ejemplo.

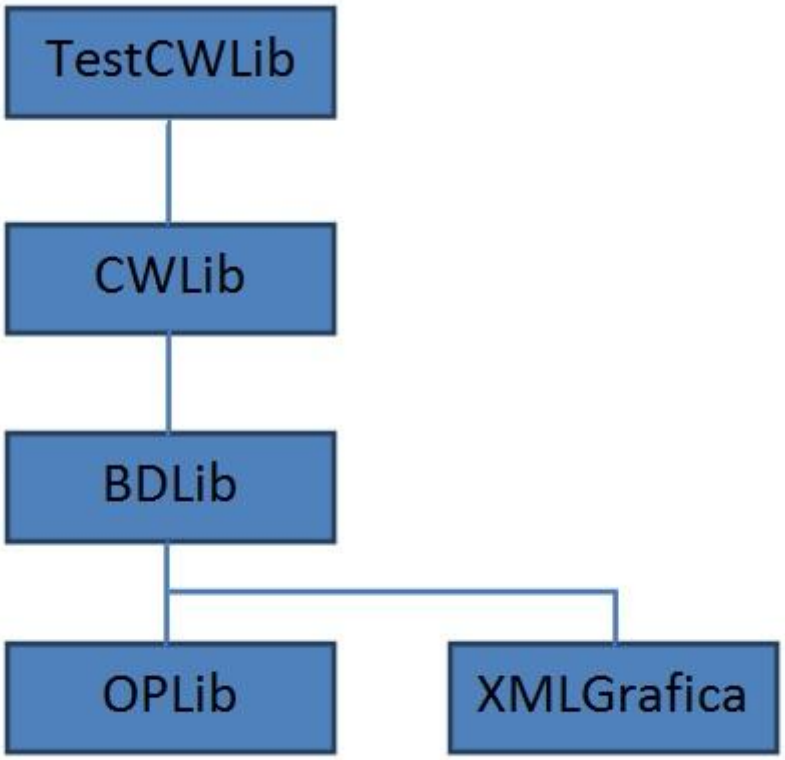
La utilidad principal de la librería *XMLGrafica* es la de visualizar gráficas en una ventana emergente. Sin embargo, *BDLib* no la necesita por este motivo, sino porque hace uso de unas funciones *rapidXML* que también incluye, con objeto de poder trabajar también con archivos *XML*. La aplicación global utiliza este formato, entre otras cosas, para guardar la lista de archivos de registros que contiene la base de datos.

Los proyectos *BDLib*, *CWLib* y *TestCWLib* precisan la inclusión de la librería 'boost' para su funcionamiento.

Dependencias entre librerías

Para poder ejecutar el proyecto satisfactoriamente, es necesario especificar las dependencias entre librerías en el compilador, de forma que no haya ningún error de compilación.

La librería TestCWLib solo depende de CWLib, asimismo, CWLib solo depende de BDLib. Sin embargo, BDLib tiene una doble dependencia con OPLib y XMLGrafica.



- OPLib: librería estática algebraica y de optimización. Contiene también otras cosas, pero en principio no son de utilidad.
- BDLib: librería estática de base de datos.
- CWLib: librería estática de procesado de datos para la obtención de caudales coherentes de agua.
- TestCWLib: aplicación ejemplo.

Identificación de la dinámica del sistema

El proyecto se centra en la identificación de la dinámica de una red de distribución de aguas y posterior corrección de los datos adquiridos por los caudalímetros a lo largo de la misma.

En un proyecto anterior, se estudiaron las pérdidas que había en la red por medio de unas ecuaciones de balance de masas en los nodos de la red. Posteriormente, estas pérdidas y las medidas de los caudalímetros se corrigieron para sacar un resultado óptimo. El problema es que no se tuvieron en cuenta ni los tiempos de transporte del agua a lo largo de los canales, ni la dinámica propia del transporte del agua a través de los canales.

Debido a que las distancias que recorre el agua de un nodo a otro son en algunos casos muy grandes, hace necesario que el problema se aborde desde otro ángulo que tenga en cuenta la dinámica del sistema, atendiendo al tiempo de transporte y la dinámica del sistema.

Para ello, vamos a obtener un modelo dinámico para cada caudal de entrada de cada nodo. Este modelo, se va a usar únicamente en los caudales de entrada de cada nodo, despreciándose así la dinámica de los caudales de salida. Hacemos la suposición de que el caudalímetro a la salida de los nodos están situados en una zona muy próxima al nodo.

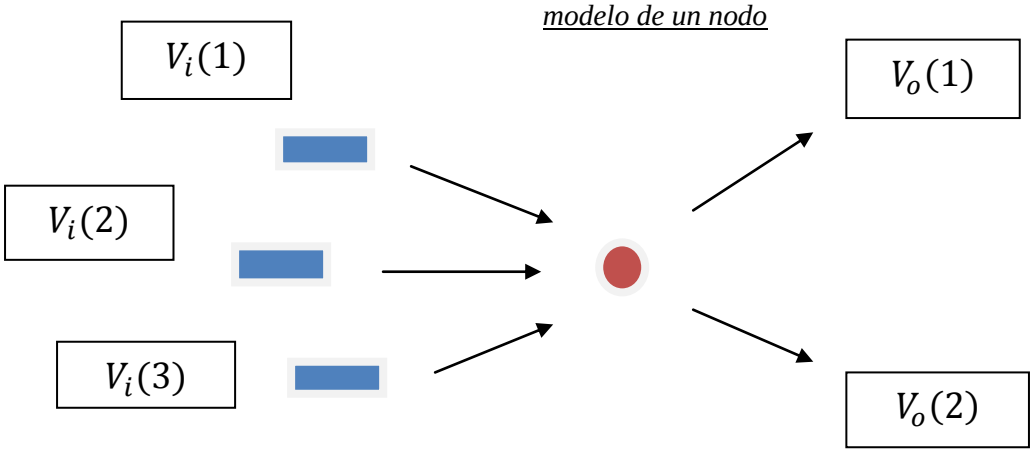
El modelo de la red se obtiene de aplicar las ecuaciones de balance de basa en los nodos de la misma, suponiendo que el agua entrante depende de las medidas de caudal realizadas aguas arriba como un sistema de primero orden con retraso y ganancia estática unidad.

Por lo tanto, para cada nodo, el modelo es:

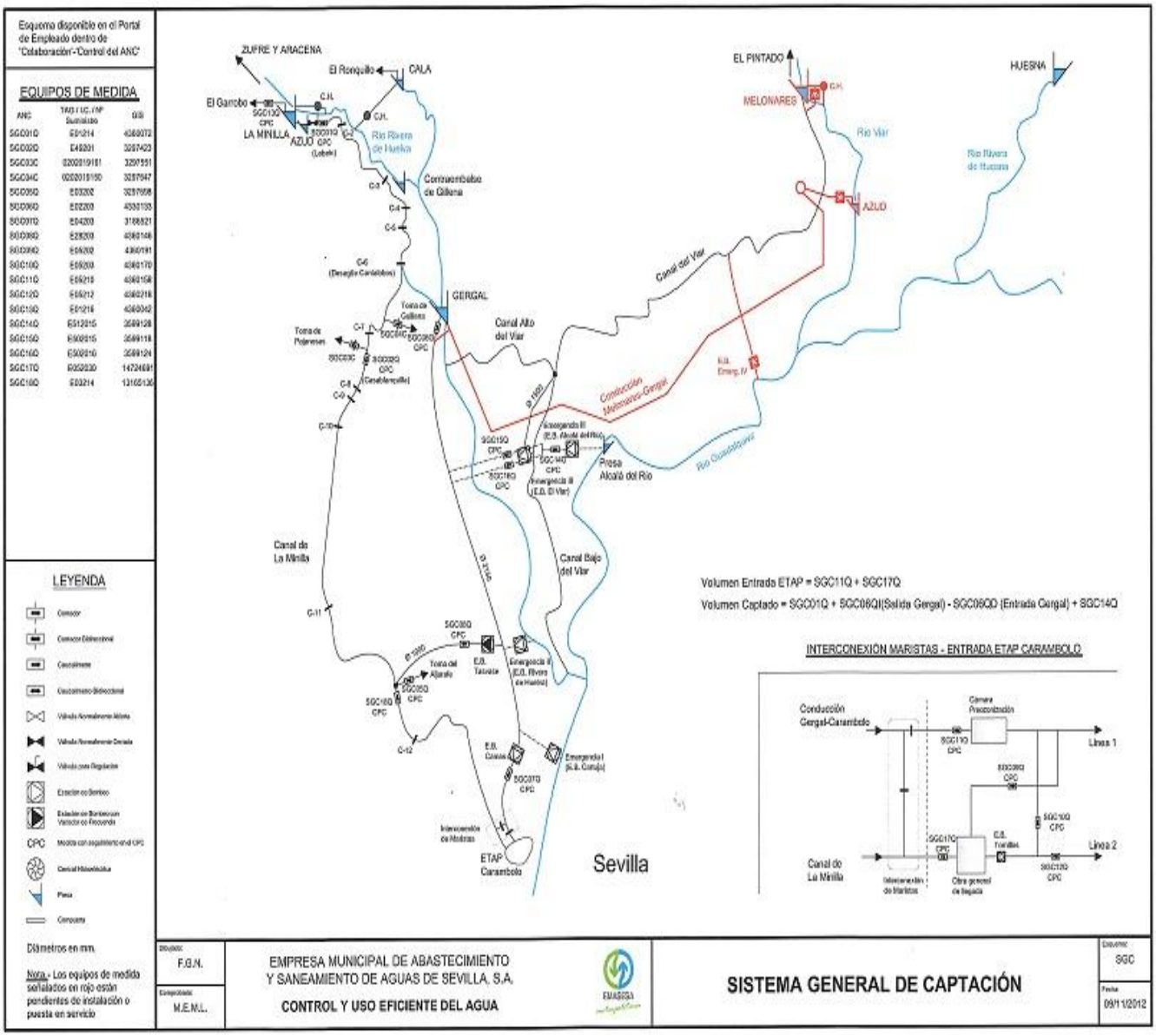
$$V_o(s) + E(s) = V_i(s)\frac{1}{\tau s + 1} \cdot e^{-sL}$$

Siendo $V_o(s)$ el conjunto de salidas del nodo, $E(s)$ las pérdidas en dicha ecuación, y habiendo un término $V_i(s)\frac{1}{\tau s+1} \cdot e^{-sL}$ para cada entrada del nodo.

modelo de un nodo



**cada caja es un sistema de primer orden*



Esta imagen corresponde a la red de distribución de aguas sobre la que versa el proyecto. En esta imagen se pueden apreciar las grandes distancias que recorre el agua y de ahí, el por qué de la necesidad de atender a la dinámica de la red.

Obtención de los parámetros del modelo discreto

El modelo escogido, para cada rama que entra en un nodo, es un modelo no autoregresivo basado en la respuesta impulsional de un sistema de primer orden retrasado con ganancia estática unidad, muestreada y truncada a *m* términos.

Dado un retraso *L*, una constante de tiempo *tau*, y un tiempo de muestreo en minutos, el modelo identificado está definido por los parámetros *m*, *LM* y θ_i con *i*=1,…,*m*.

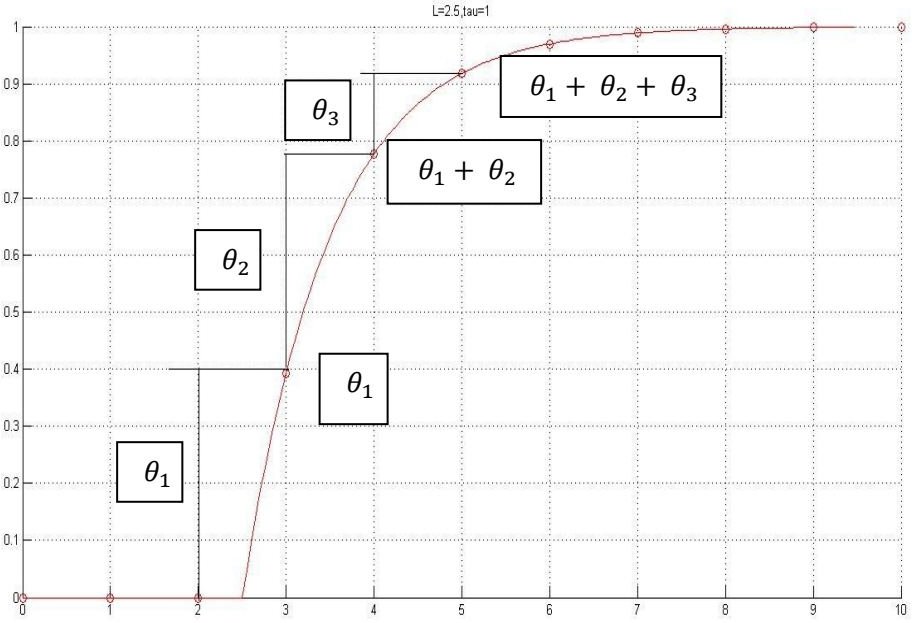
El valor del número de tiempos de muestreo *LM* de retraso es igual a la división entera del retraso *L* entre el tiempo de muestreo *Dt* redondeada hacia abajo.

$$LM = floor\left(\frac{L}{Dt}\right);$$

El retraso no tiene que ser múltiplo de *Dt*. Puede tomar cualquier valor. El efecto del retraso se tiene en cuenta al calcular la respuesta impulsional muestreada.

Cada parámetro θ_i se obtiene de la respuesta impulsional de un sistema de primer orden con ganancia 1, constante de tiempo *tau* y retraso *EL=L-LM*Dt* (el resto del retraso).

$$\theta_i = e^{-\max((i-1)Dt-EL,0)/tau} - e^{-(iDt-EL)/tau}$$



La figura muestra la respuesta de un sistema continuo de primer orden con una constante de tiempo unidad, un retraso de 2.5 minutos y ganancia unidad. Los círculos rojos muestran la respuesta muestreada con tiempo de muestreo de un minuto. El valor θ_i*se obtiene como la diferencia entre el valor de la respuesta en i y en el instante anterior i-1.*

El número de parámetros del modelo *m* es el menor valor que cumple:

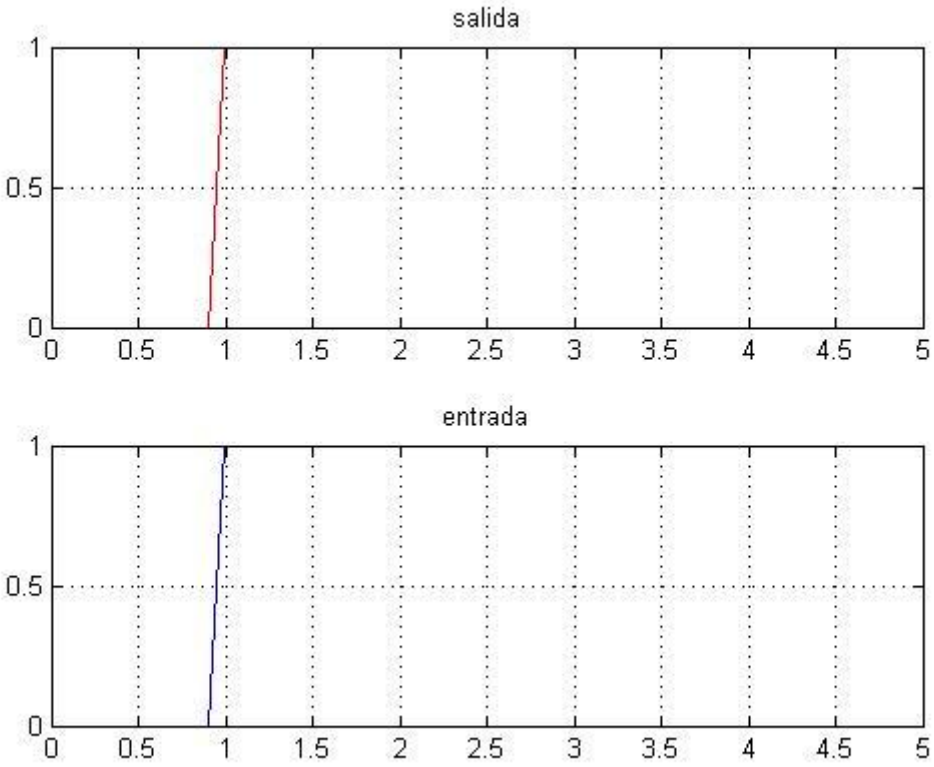
$$mDt - EL > 6tau$$

Para garantizar que

$$\sum_{i=1,\dots,m} \theta_i \in [0.9975,1]$$

El valor del retraso *L* y de la constante de tiempo *tau* que caracterizan al canal son independientes del tiempo de muestreo, pudiendo ser utilizados para evaluar las pérdidas en diferentes periodos.

En el caso en el que la dinámica del sistema fuese despreciable, es decir, *L=0* y *tau=0*, existiría una única θ, que tendría un valor de 1. Se entendería que la respuesta de la salida a un cambio en la entrada es inmediata y con un comportamiento parecido a un escalón.



Ejemplo del caso en el que L=0 y tau=0.

Modelo dinámico discreto

Nuestra red viene definida por las siguientes ecuaciones de masa:

$$v_1^{(i)} \approx v_2^{(i)}$$

$$v_2^{(i)} + v_4^{(i)} \approx v_3^{(i)} + v_5^{(i)}$$

$$v_5^{(i)} \approx v_6^{(i)}$$

$$v_7^{(i)} \approx v_4^{(i)} + v_8^{(i)}$$

$$v_8^{(i)} \approx v_9^{(i)}$$

$$v_6^{(i)} + v_9^{(i)} \approx v_{10}^{(i)} + v_{11}^{(i)} + v_{12}^{(i)} + v_{13}^{(i)} + v_d^{(i)}$$

$$v_{10}^{(i)} + v_{11}^{(i)} + v_{12}^{(i)} \approx v_{16}^{(i)} + v_{17}^{(i)} + v_{18}^{(i)}$$

Con $i = 1, \dots, N$

Nótese que en las 7 ecuaciones de equilibrio de masa consideradas los términos de la izquierda corresponden a las entradas de los nudos y los términos de la izquierda a las salidas. Podemos definir el error absoluto $e_j^{(i)}$cometido en el periodo de tiempo *i-ésimo* en la ecuación *j-ésima* como sigue:

$$e_1^{(i)} = v_1^{(i)} - v_2^{(i)}$$

$$e_2^{(i)} = v_2^{(i)} + v_4^{(i)} - v_3^{(i)} - v_5^{(i)}$$

$$e_3^{(i)} = v_5^{(i)} - v_6^{(i)}$$

$$e_4^{(i)} = v_7^{(i)} - v_4^{(i)} - v_8^{(i)}$$

$$e_5^{(i)} = v_8^{(i)} - v_9^{(i)}$$

$$e_6^{(i)} = v_6^{(i)} + v_9^{(i)} - v_{10}^{(i)} - v_{11}^{(i)} - v_{12}^{(i)} - v_{13}^{(i)} - v_d^{(i)}$$

$$e_7^{(i)} = v_{10}^{(i)} + v_{11}^{(i)} + v_{12}^{(i)} - v_{10}^{(i)} - v_{16}^{(i)} - v_{17}^{(i)} - v_{18}^{(i)}$$

Dado que las ecuaciones de balance de masa se forman como suma de volúmenes entrantes igual a suma de volúmenes salientes más acumulados, los errores vienen dados como suma de volúmenes entrantes menos suma de volúmenes salientes y acumulados. Nótese que los términos de pérdida en la red aparecen con signo positivo (la suma de los volúmenes entrantes es superior a la suma de los volúmenes salientes y acumulados). De aquí se infiere que el signo de los errores constituye un indicio de la calidad de las medidas volumétricas. En efecto, discrepancias positivas pueden deberse a términos de pérdidas o errores en la medida de los caudalímetros. Sin embargo, los errores negativos se deben fundamentalmente a errores de medida puesto que los términos de pérdidas deben tener signo positivo.

Atendiendo a las ecuaciones anteriores elaboramos las ecuaciones del modelo dinámico, añadiendo dinámica únicamente a los caudales de entrada. Nuestro modelo escogido es un modelo no autoregresivo basado en la respuesta impulsional muestreada truncada a *m* términos.

Suponemos que solo los términos de entrada de cada ecuación van a ser modelados en las ecuaciones de cálculo de errores y que las variables que aparecen como entrada en una ecuación solo actúa como entrada en dicha ecuación.

Las ecuaciones anteriores quedan :

$$e_1^{(k)} = \sum_{i=0}^{m11} \theta_{11}^i \, v_1^{(k-LM_{11}-i)} - v_2^{(k)}$$

$$e_2^{(k)} = \sum_{i=0}^{m22} \theta_{22}^i \, v_2^{(k-LM_{22}-i)} + \sum_{i=0}^{m24} \theta_{24}^i \, v_4^{(k-LM_{24}-i)} - v_3^{(k)} - v_5^{(k)}$$

$$e_3^{(k)} = \sum_{i=0}^{m35} \theta_{35}^i \, v_5^{(k-35-i)} - v_6^{(k)}$$

$$e_4^{(k)} = \sum_{i=0}^{m47} \theta_{47}^i \, v_7^{(k-47-i)} - v_4^{(k)} - v_8^{(k)}$$

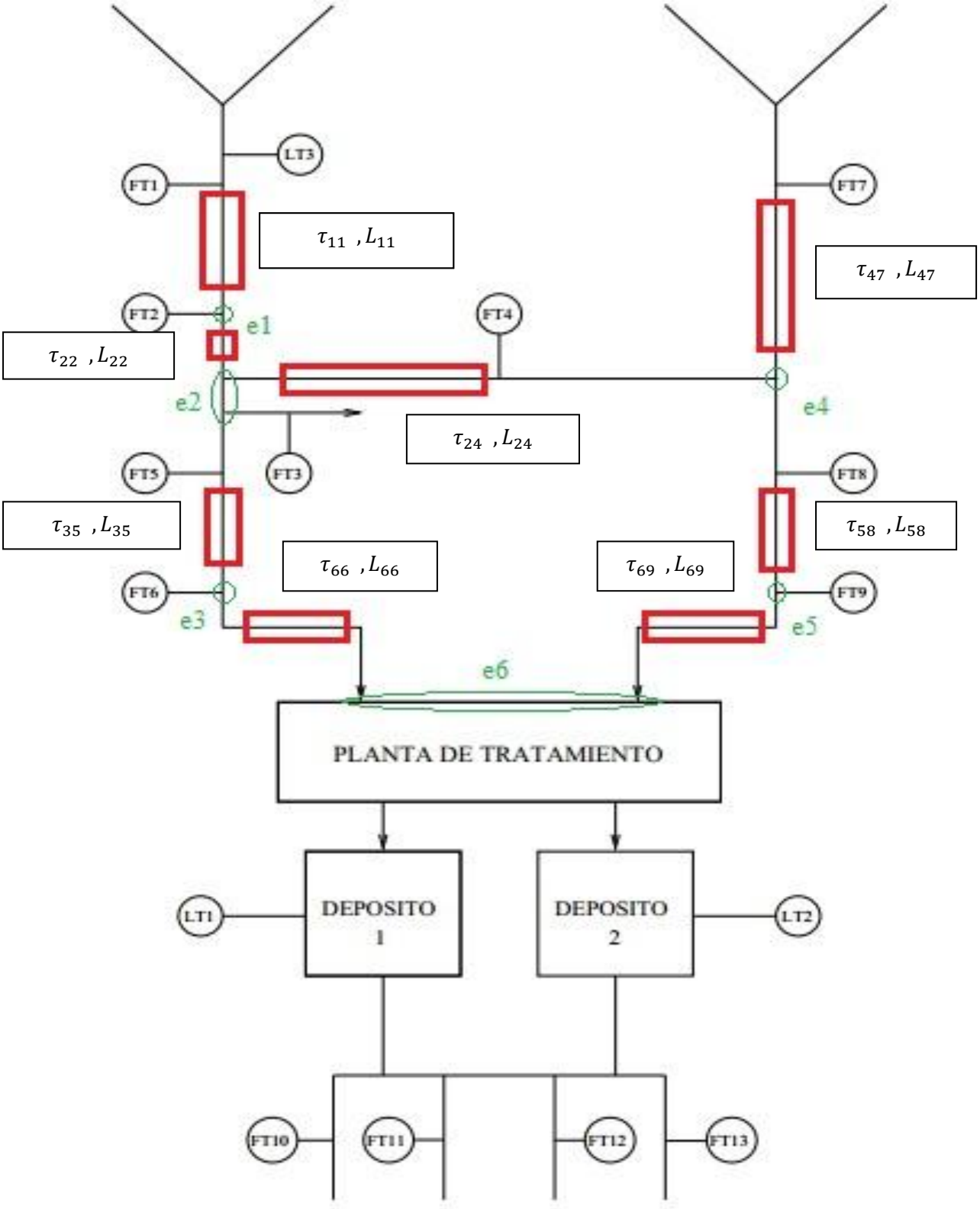
$$e_5^{(k)} = \sum_{i=0}^{m58} \theta_{58}^i \, v_8^{(k-LM_{58}-i)} - v_9^{(k)}$$

$$e_6^{(k)} = \sum_{i=0}^{m66} \theta_{66}^i \, v_6^{(k-LM_{66}-i)} + \sum_{i=0}^{m69} \theta_{69}^i \, v_9^{(k-LM_{69}-i)} - v_{10}^{(k)} - v_{11}^{(k)} - v_{12}^{(k)} - v_{13}^{(k)} - v_d^{(k)}$$

$$e_7^{(k)} = \sum_{i=0}^{m710} \theta_{710}^i \, v_{10}^{(k-LM_{710}-i)} + \sum_{i=0}^{m711} \theta_{711}^i \, v_{11}^{(k-LM_{711}-i)} + \sum_{i=0}^{m712} \theta_{712}^i \, v_{12}^{(k-LM_{712}-i)} - v_{10}^{(k)} - v_{16}^{(k)} - v_{17}^{(k)} - v_{18}^{(k)}$$

Con $k = 1, \dots, N$

Los parámetros LM y θ^i se obtienen del retraso del sistema en minutos pasado a muestras y de la respuesta impulsional del sistema respectivamente.



La figura anterior muestra un esquema de la red de distribución de aguas de emasesa con sus respectivos sensores. Sobre ella, se muestran los tramos que tienen dinámica y que parámetros corresponden a cada tramo.

Función para obtener un modelo de primer orden

El objetivo es identificar un modelo dinámico que relacione los caudales medidos a la entrada con los caudales medidos a la salida. Los datos utilizados son los almacenados en archivos de texto creados a través de visual studio empleando las librerías OPlib y CWlib. Estos archivos contienen las medidas integradas cada media hora de los distintos sensores de nuestra red en un periodo de tiempo de un mes.

Se ha desarrollado una función de Matlab para calcular el error cuadrático de una serie de modelos de primer orden con retardo puro y ganancia estática unidad. La función, a partir de un histórico de datos de las entradas y un tiempo de muestreo especificado, calcula el error cuadrático para cada par *L* y *tau*. Obteniéndose así una matriz de errores cuadráticos que nos permitirá ver que pareja de *tau* y *L* nos da un mejor resultado de la identificación.

function E=iden_Ltau2_1(MD,Dt,L,tau)

argumentos de entrada:

- **MD** Matriz de datos de las medidas de los sensores, integradas cada media hora durante un mes, que se obtiene de ejecutar los archivos de texto.
- **Dt** Tiempo de muestreo en minutos del modelo identificado.
- **L** vector con retraso en minutos para probar
- **tau** vector con tiempo característico en minutos para probar

argumentos de salida:

- **E** Error cuadrático para cada par de L, tau.
  - $E(i,j) = E\big(L(i), tau(j)\big) = \sum abs(v2(k) - v2m(k))$

El modelo identificado cumple la siguiente ecuación:

$$v2m(k) = \sum_{i=0,\dots,m-1} \theta_i v1(k - LM - i)$$

Donde $v1(k)$ es el volumen que ha pasado por la el caudal de entrada en Dt minutos en el instante de muestreo k, $v2(k)$ es el volumen que a pasado por el caudal de salida en Dt minutos en el instante de muestreo k, y $v2m(k)$ es ese mismo valor estimado con el modelo a partir del histórico de datos de $v1$.

El modelo es un sistema de primer orden lineal con retardo con entrada $v1$ y salida $v2$.

Pruebas de identificación

Ecuación 1

Se ha validado un modelo a partir de los datos de los meses de Octubre, Noviembre y Diciembre de 2013, y Enero de 2014.

Se ha calculado el error para un tiempo de muestreo de 30 minutos.

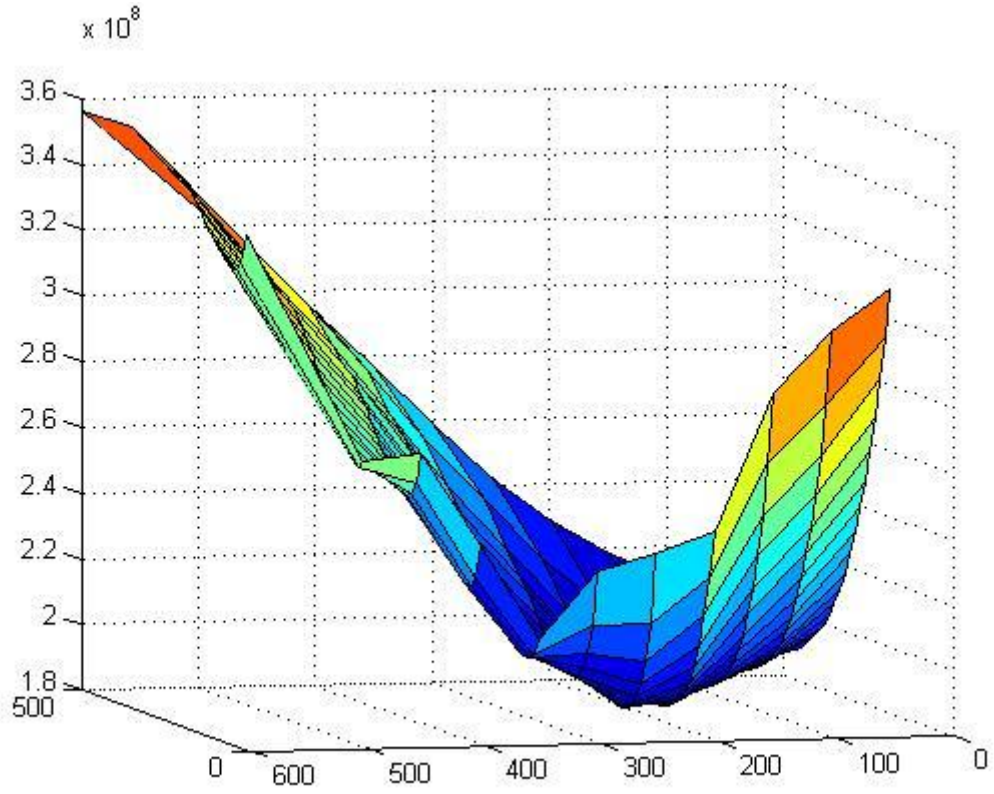
Los valores de retraso y tiempo característico son los siguientes

$$L = [140\ 170\ 200\ 230\ 250\ 280\ 310\ 340]$$

$$tau = [60\ 90\ 120\ 150\ 165\ 180\ 210\ 240\ 270]$$

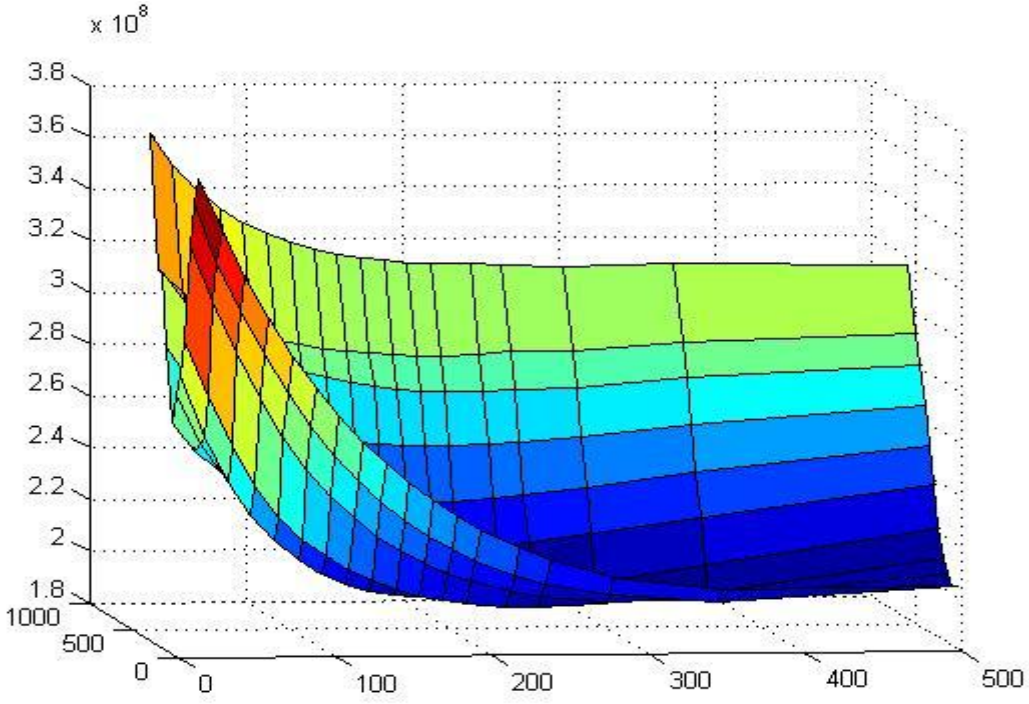
Los modelos con menor suma de los errores absolutos para cada mes fueron:

-Octubre: L1=200, tau1=165

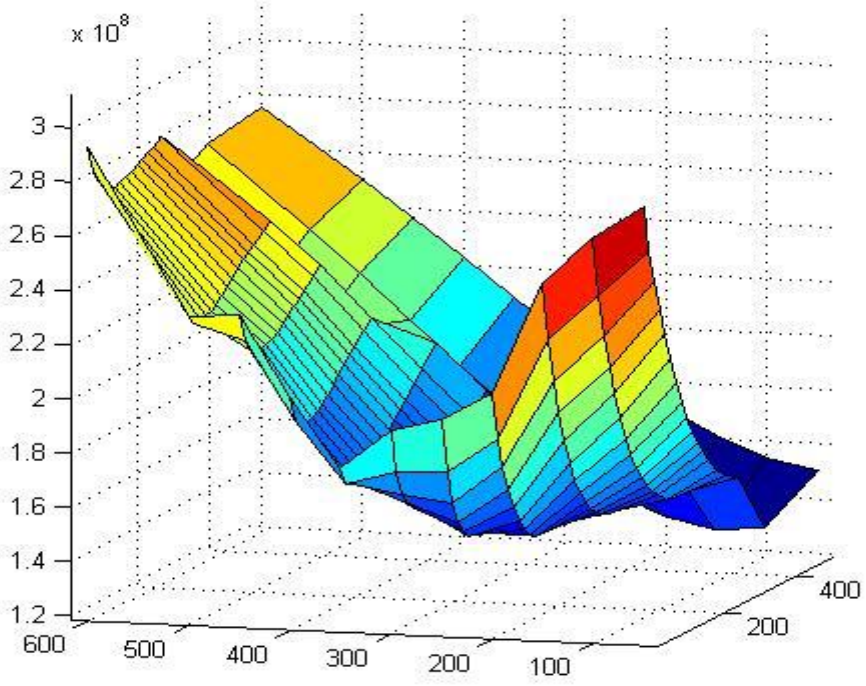


Representación del error en función del retraso L y el tiempo característico tau. Se puede obsevar una tendencia a tener un mínimo a lo largo de una dirección L+tau=cte, ya que el retraso y un tiempo característico largo se pueden compensar.

-Noviembre: L1=200, tau1=165

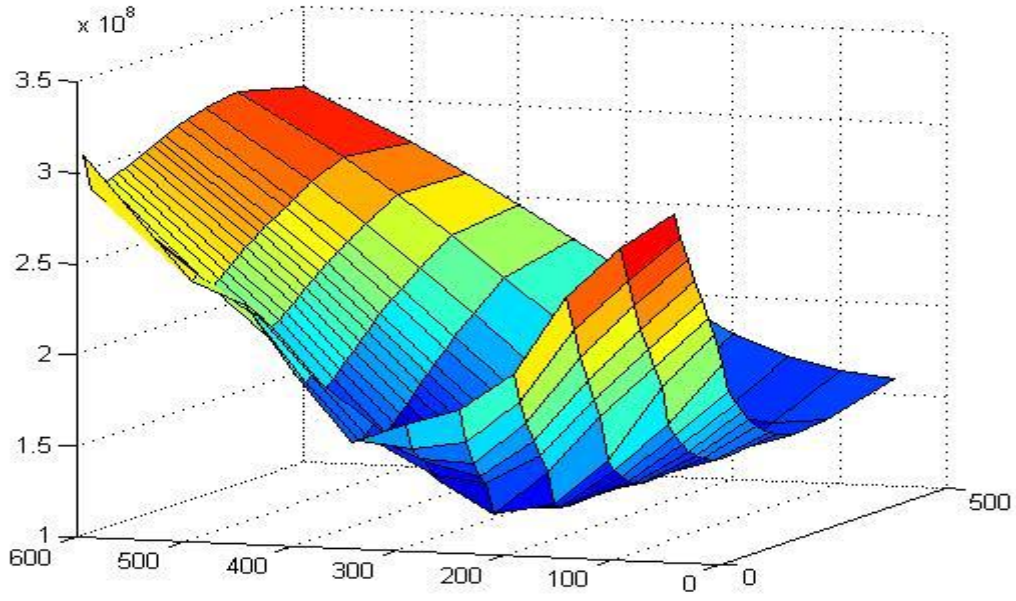


-Diciembre: L1=200, tau1=180



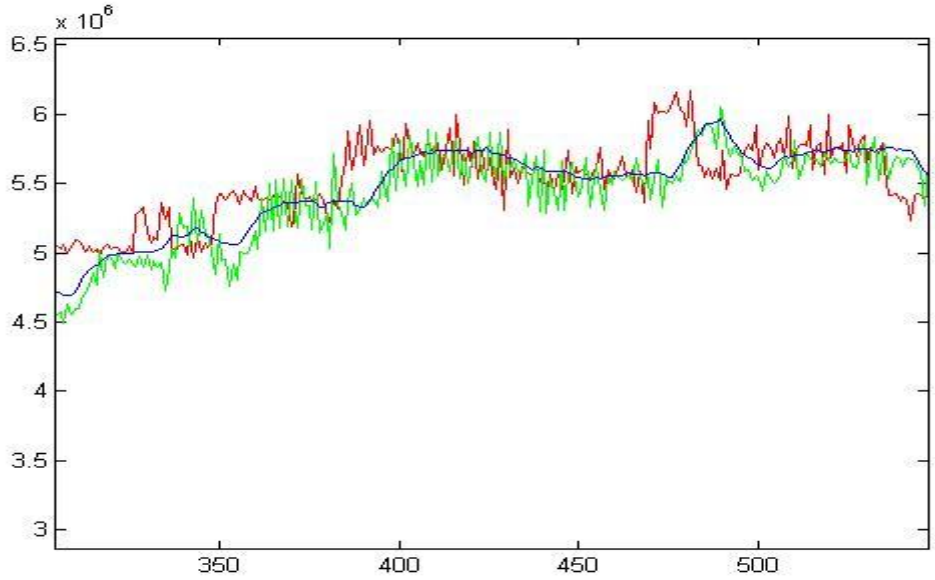
Representación del error en función del retraso L y el tiempo característico tau. Se puede obsevar una tendencia a tener un mínimo a lo largo de una dirección L+tau=cte, ya que el retraso y un tiempo característico largo se pueden compensar.

-Enero: L1=250, tau=90

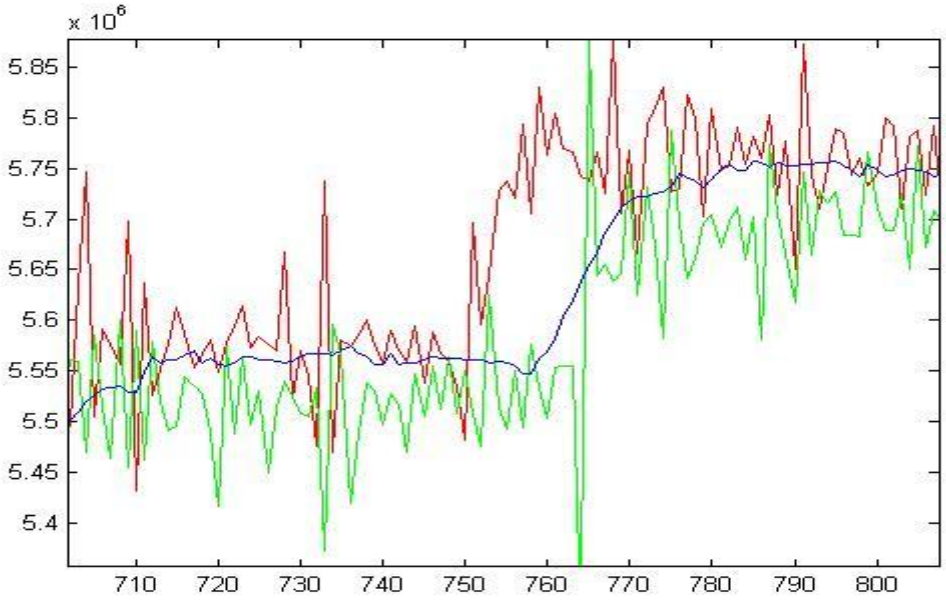


Representación del error en función del retraso L y el tiempo característico tau. Se puede obsevar una tendencia a tener un mínimo a lo largo de una dirección L+tau=cte, ya que el retraso y un tiempo característico largo se pueden compensar.

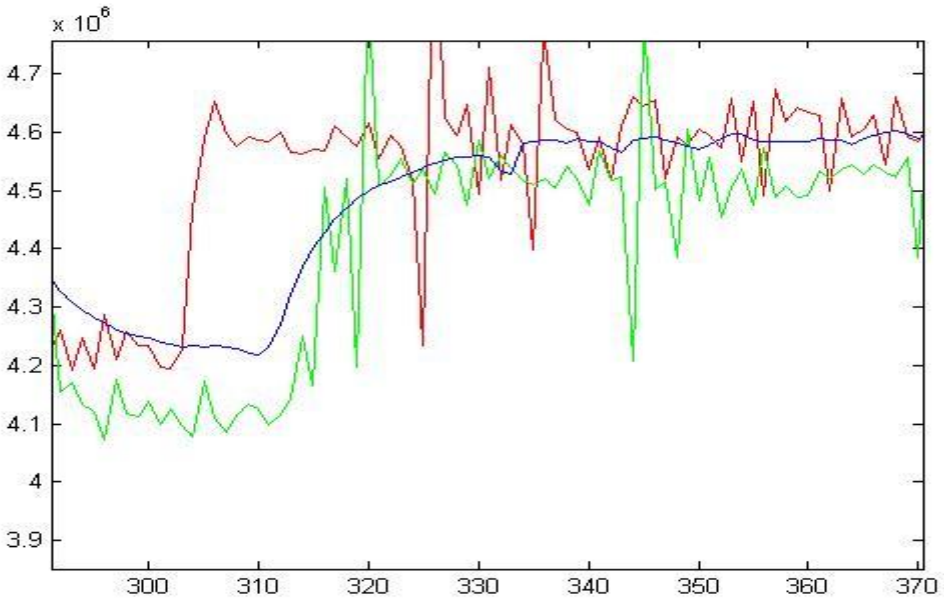
A la vista de estos resultados se ha decidido validar el modelo con L1=200, tau1=180. El error absoluto es de 1.4862e+08.



La imagen anterior muestra la validación del modelo en el mes de Diciembre con tiempo de muestreo de 30 minutos, L1=200 y tau1=180.En rojo está representada la entrada (v1), en verde la salida (v2) y en azul el modelo.



La imagen anterior muestra la validación del modelo en el mes de Diciembre con tiempo de muestreo de 30 minutos, L1=200 y tau1=180.En rojo está representada la entrada (v1), en verde la salida (v2) y en azul el modelo.



La imagen anterior muestra la validación del modelo en el mes de Octubre con tiempo de muestreo de 30 minutos, L1=200 y tau1=180.En rojo está representada la entrada (v1), en verde la salida (v2) y en azul el modelo.

Ecuación 2

Se ha validado un modelo a partir de los datos de los meses de Octubre y Noviembre de 2013.

Se ha calculado el error para un tiempo de muestreo de 30 minutos.

Los valores de retraso y tiempo característico son los siguientes

$$L = [20\ 50\ 80\ 110\ 140\ 180\ 200\ 230\ 250\ 260]$$

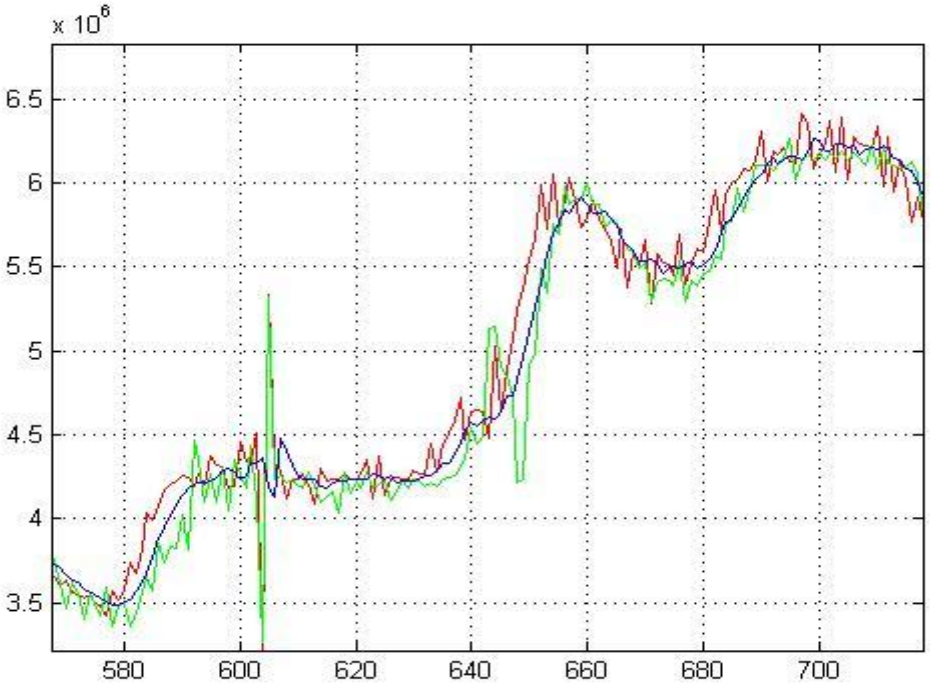
$$tau = [20\ 50\ 60\ 90\ 120\ 150\ 165\ 180\ 200\ 240\ 270]$$

Los modelos con menor suma de los errores absolutos para cada mes fueron:

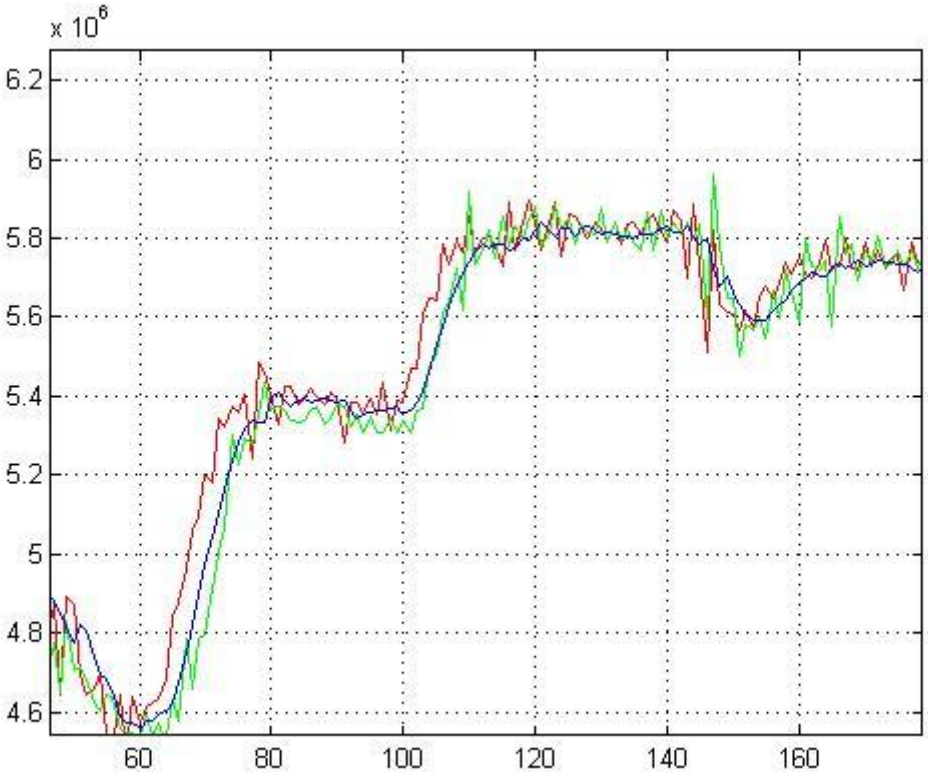
-Octubre: L2=20, tau2=60, L4=250, tau4=200

-Noviembre: L2=50, tau2=50, L4=180, tau4=200

A la vista de estos resultados se ha decidido validar el modelo con L2=50, tau2=50, L4=180, tau4=200. El error absoluto es de 9.8429e+07.



La imagen anterior muestra la validación del modelo en el mes de Octubre con tiempo de muestreo de 30 minutos, L2=50, tau2=60, L4=250 y tau4=200. En rojo está representada la entrada (v2+v4), en verde la salida (v3+v5) y en azul el modelo.



La imagen anterior muestra la validación del modelo en el mes de Octubre con tiempo de muestreo de 30 minutos, L2=50, tau2=60, L4=250 y tau4=200. En rojo está representada la entrada (v2+v4), en verde la salida (v3+v5) y en azul el modelo.

Ecuación 3

Se ha validado un modelo a partir de los datos de los meses de Octubre, Noviembre y Diciembre de 2013, y Enero de 2014.

Se ha calculado el error para un tiempo de muestreo de 30 minutos.

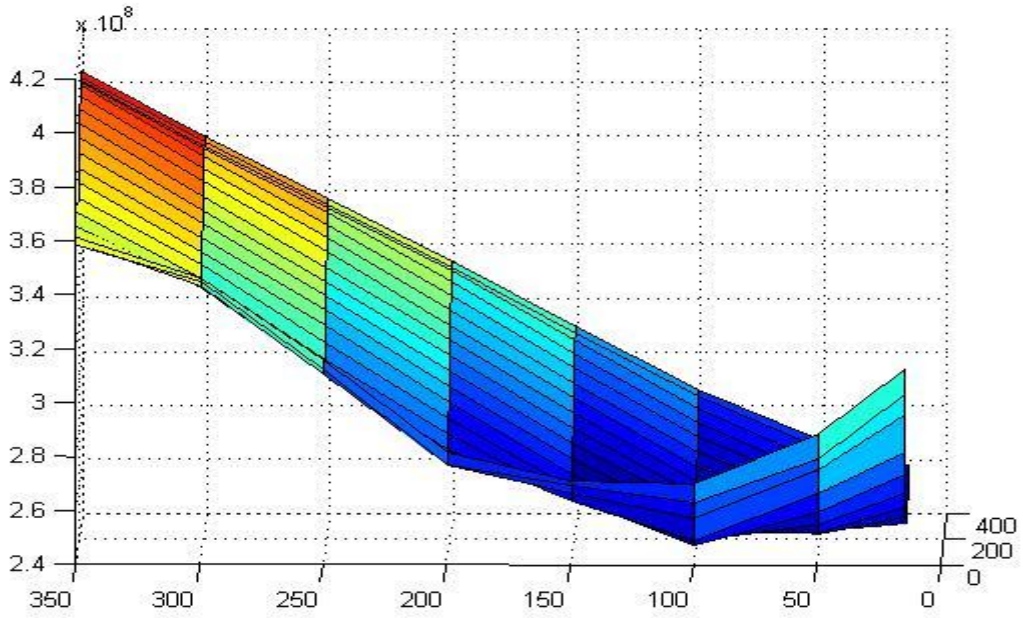
Los valores de retraso y tiempo característico son los siguientes

$$L = [15\ 30\ 50\ 75\ 100\ 130\ 150]$$

$$tau = [20\ 45\ 60\ 90\ 120\ 140\ 165\ 180\ ]$$

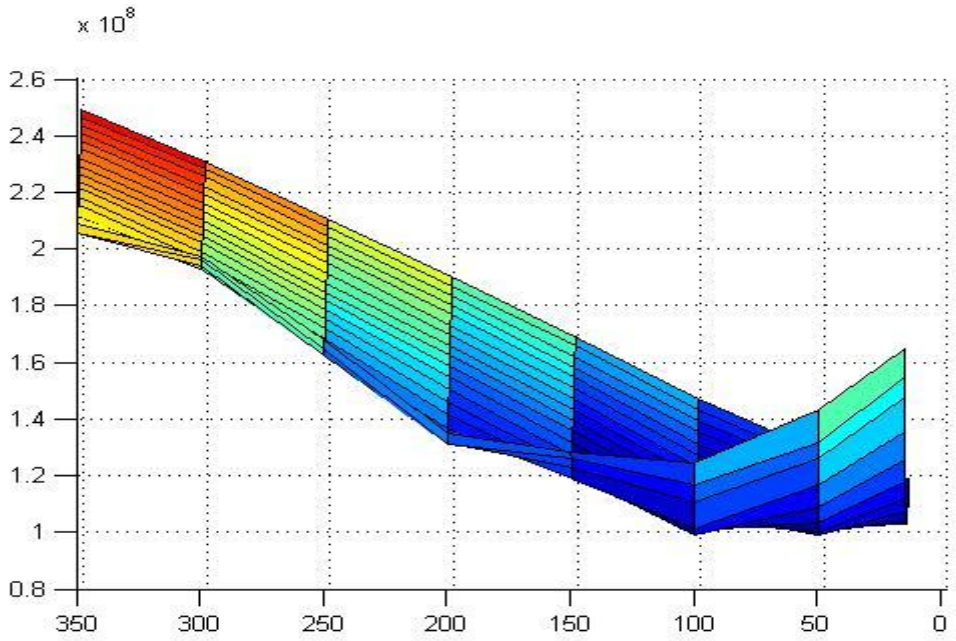
Los modelos con menor suma de los errores absolutos para cada mes fueron:

-Octubre: L5=100, tau5=45



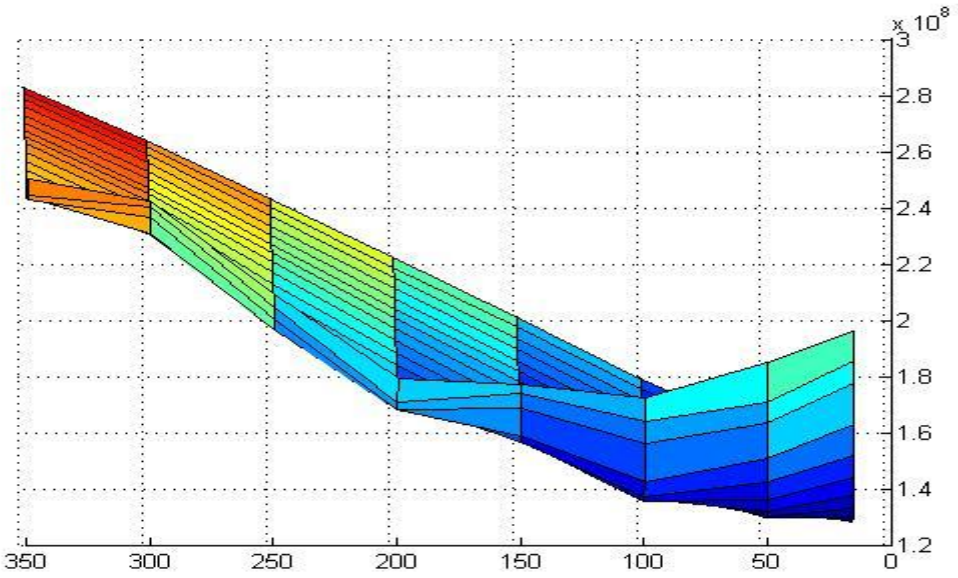
Representación del error en función del retraso L y el tiempo característico tau. Se puede obsevar una tendencia a tener un mínimo a lo largo de una dirección L+tau=cte, ya que el retraso y un tiempo característico largo se pueden compensar.

-Noviembre: L5=100, tau5=45



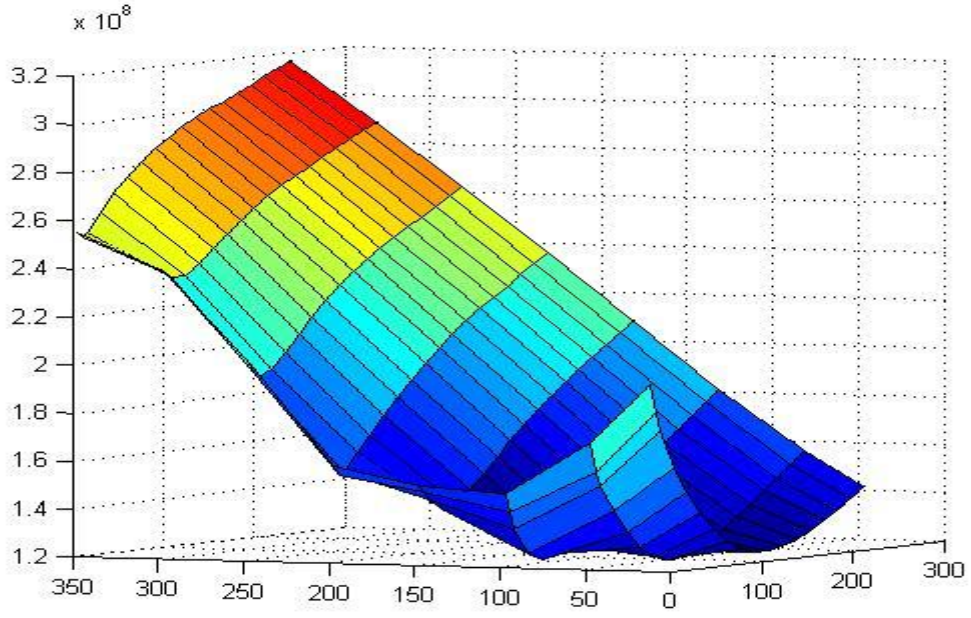
Representación del error en función del retraso L y el tiempo característico tau. Se puede obsevar una tendencia a tener un mínimo a lo largo de una dirección L+tau=cte, ya que el retraso y un tiempo característico largo se pueden compensar.

-Diciembre: L5=15, tau5=135



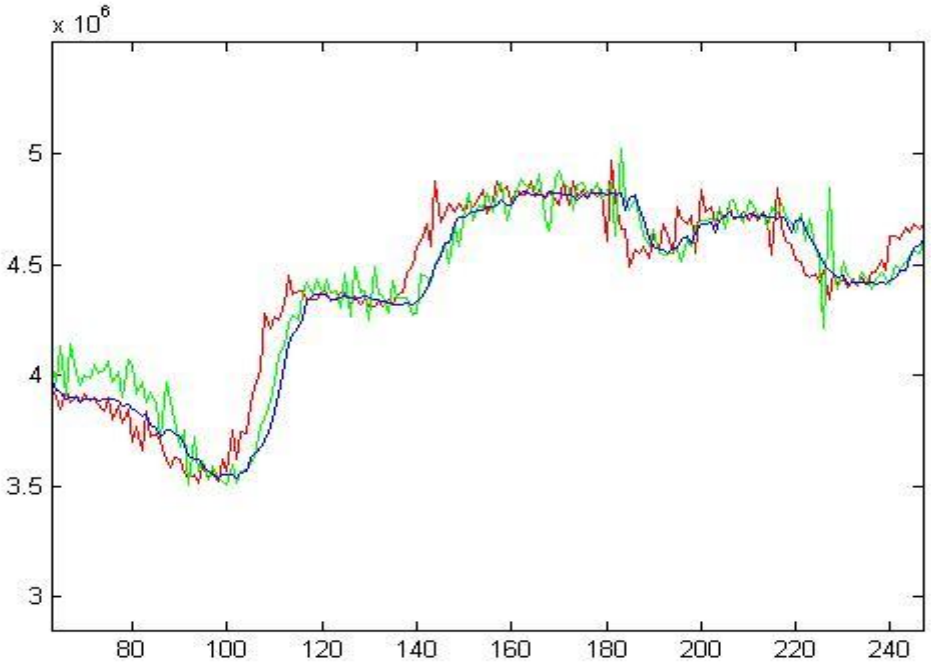
Representación del error en función del retraso L y el tiempo característico tau. Se puede obsevar una tendencia a tener un mínimo a lo largo de una dirección L+tau=cte, ya que el retraso y un tiempo característico largo se pueden compensar.

-Enero: L5=50, tau5=90

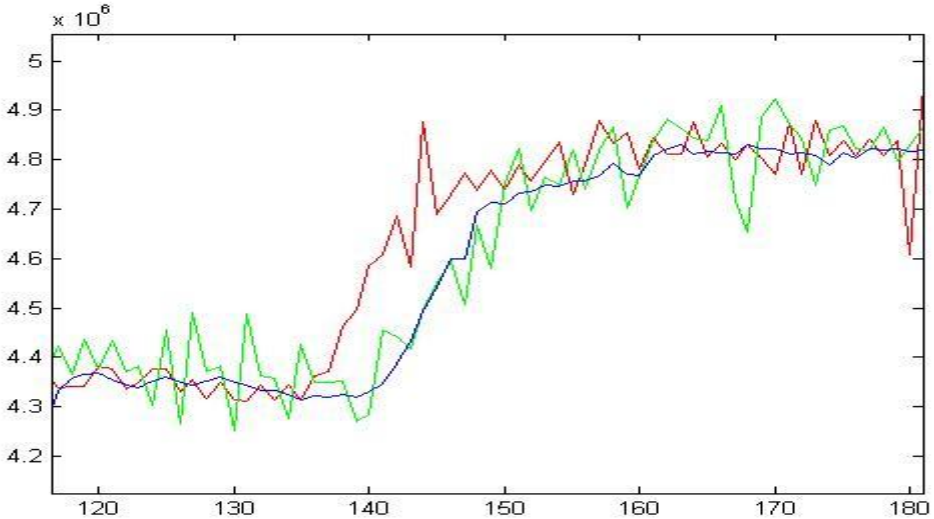


Representación del error en función del retraso L y el tiempo característico tau. Se puede obsevar una tendencia a tener un mínimo a lo largo de una dirección L+tau=cte, ya que el retraso y un tiempo característico largo se pueden compensar.

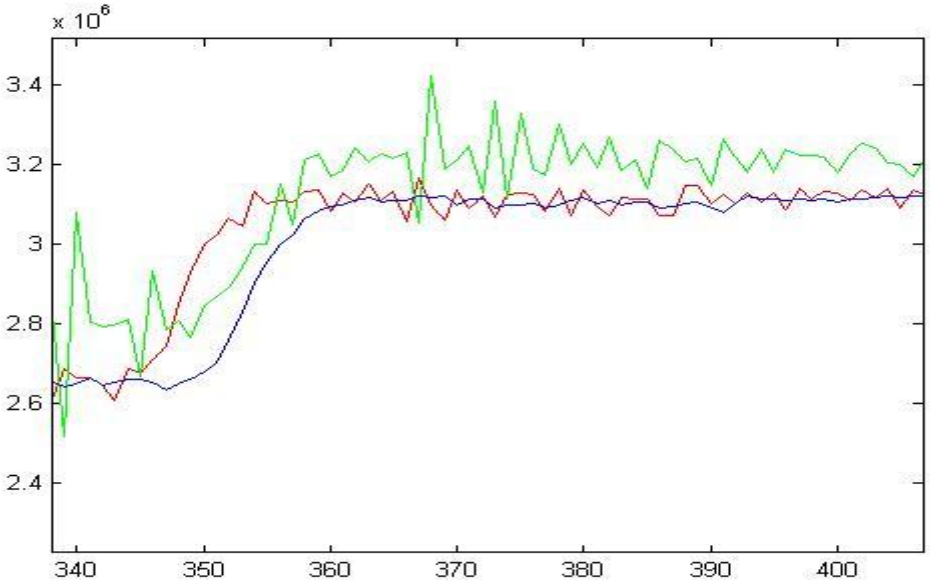
A la vista de estos resultados se ha decidido validar el modelo con L5=100, tau5=45.El error absoluto es de 9.9143e+07.



La imagen anterior muestra la validación del modelo en el mes de Noviembre con tiempo de muestreo de 30 minutos, L5=100, tau5=45. En rojo está representada la entrada (v5), en verde la salida (v6) y en azul el modelo.



La imagen anterior muestra la validación del modelo en el mes de Noviembre con tiempo de muestreo de 30 minutos, L5=100, tau5=45. En rojo está representada la entrada (v5), en verde la salida (v6) y en azul el modelo.

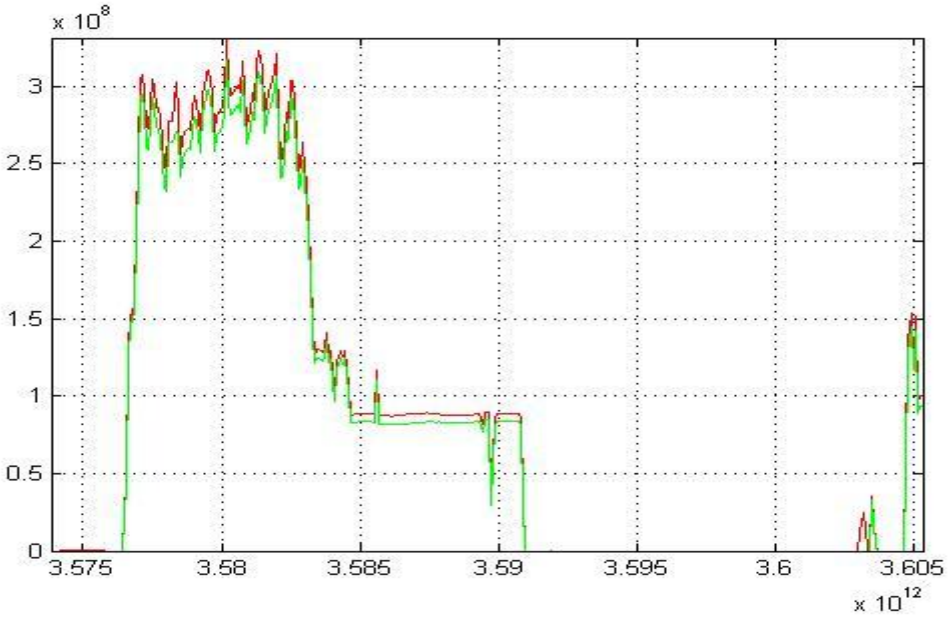


La imagen anterior muestra la validación del modelo en el mes de Octubre con tiempo de muestreo de 30 minutos, L5=100, tau5=45. En rojo está representada la entrada (v5), en verde la salida (v6) y en azul el modelo.

Ecuación 4

Se ha validado un modelo a partir de los datos del mes de Octubre de 2013.

Para esta ecuación solo se han usado datos de un mes debido a que durante una gran parte del año el suministro de agua por este canal permanece cortado.



Representación de la dinámica de la ecuación 4 desde abril de 2013 hasta abril de 2014. En rojo se dibuja la entrada (v7) y en verde la salida (v4 + v8).

Se ha calculado el error para un tiempo de muestreo de 30 minutos.

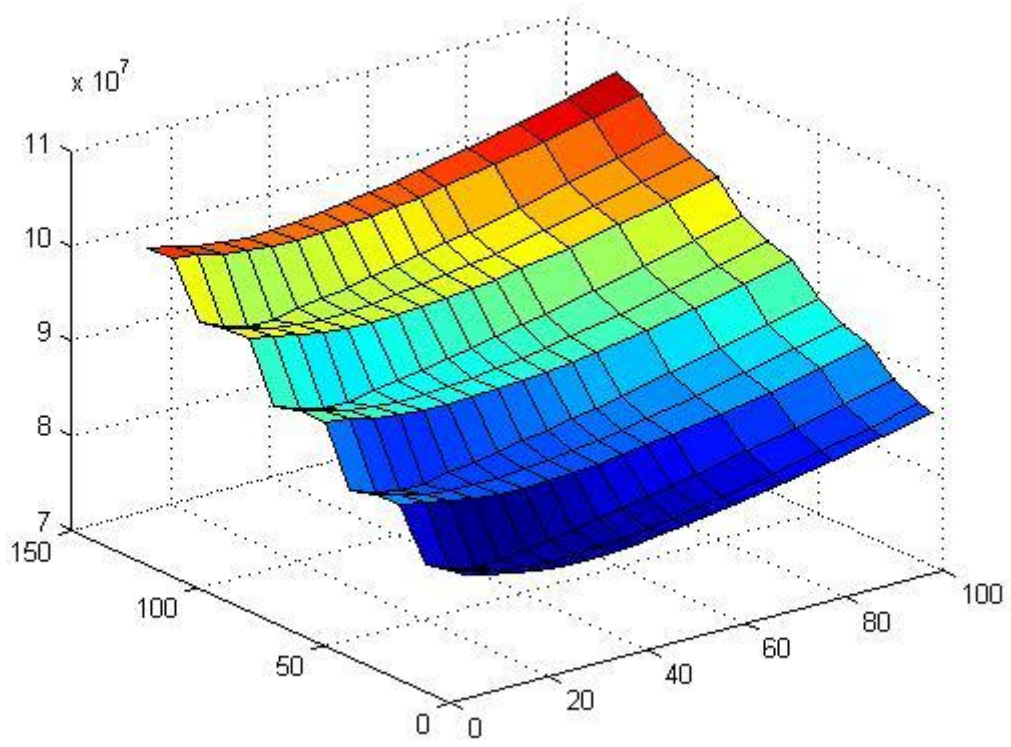
Los valores de retraso y tiempo característico son los siguientes

$$L = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 15\ 20\ 25\ 30\ ]$$

$$tau = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 15\ 20\ 25\ 30\ ]$$

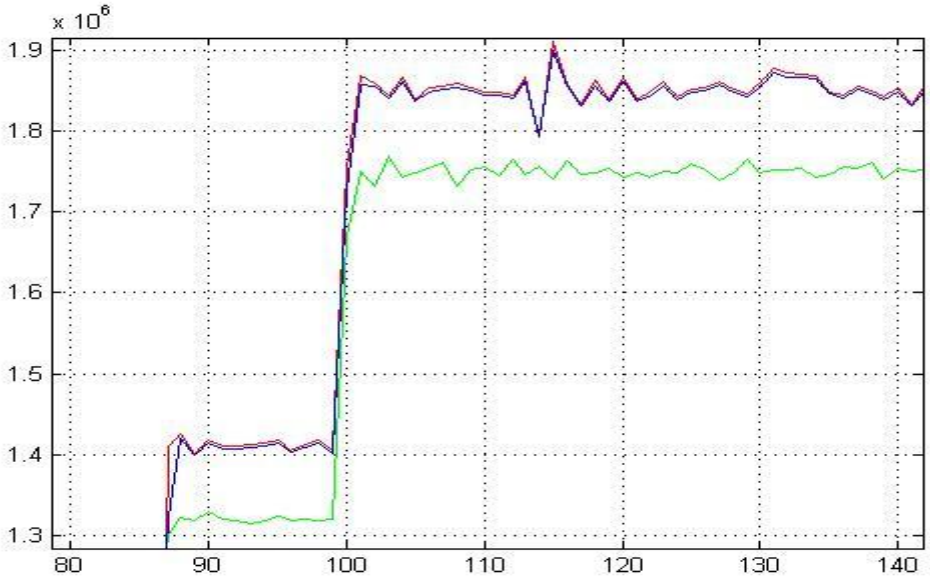
El modelo con menor suma de los errores absolutos fue:

-Octubre: L7=5, tau7=9



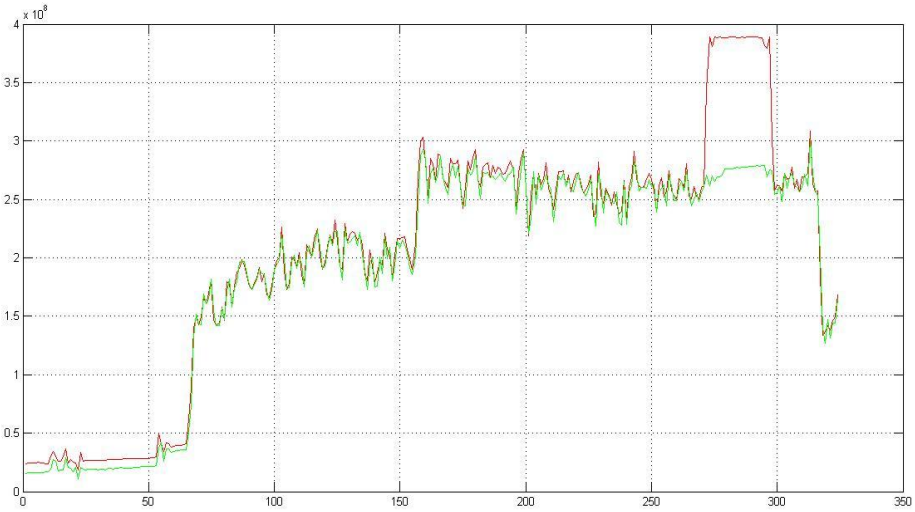
Representación del error en función del retraso L y el tiempo característico tau. Se puede obsevar una tendencia a tener un mínimo a lo largo de una dirección L+tau=cte, ya que el retraso y un tiempo característico largo se pueden compensar.

Dado que salen unos datos de tau7 y L7 muy bajos y utilizamos un tiempo de muestreo de 30 se podría decir que dicho sistema tiene una dinámica prácticamente nula. Una aproximación de tau7 y L7 ambos igual a 0 sería aceptada. El sumatorio de los errores resulta 6.5248e+12.



La imagen anterior muestra la validación del modelo en el mes de Octubre con tiempo de muestreo de 30 minutos, L7=0, tau7=0. En rojo está representada la entrada (v7), en verde la salida (v4+v8) y en azul el modelo.

En esta imagen, además, se puede apreciar que durante algunos periodos de tiempo sacan agua del canal, la salida sigue la dinámica de la entrada pero no llega a alcanzar el valor de agua de la entrada. Podría pensarse que es debido a un fallo de alguno de los sensores, pero en imágenes donde se representan más instantes de tiempo se puede apreciar como la salida sigue exactamente a la entrada menos en algunas zonas donde suponemos que sacan agua del canal en cuestión.



Esta imagen representa la entrada de la ecuación 1 en rojo y la salida en verde desde abril de 2013 hasta abril de 2014. Al final de la gráfica se puede apreciar un ejemplo de extracción de agua en un canal.

Ecuación 5

Se ha validado un modelo a partir de los datos de los meses de Agosto y Septiembre, y de Abril y Mayo, de 2013.

Se ha calculado el error para un tiempo de muestreo de 30 minutos.

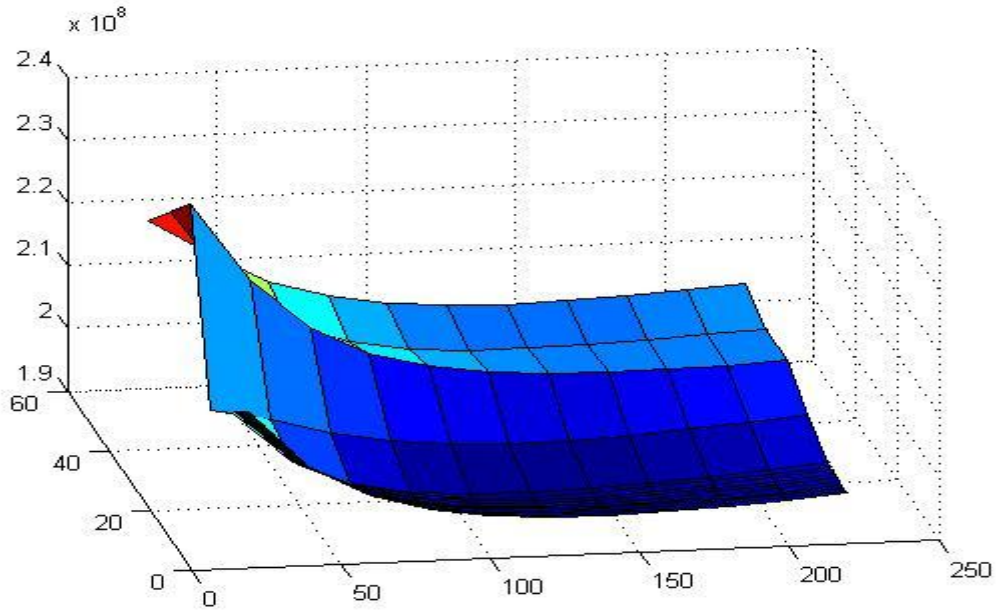
Los valores de retraso y tiempo característico son los siguientes

$$L = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 15\ 20\ 25\ 30\ ]$$

$$tau = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 15\ 20\ 25\ 30\ 40\ 50\ 60\ 70\ 80\ ]$$

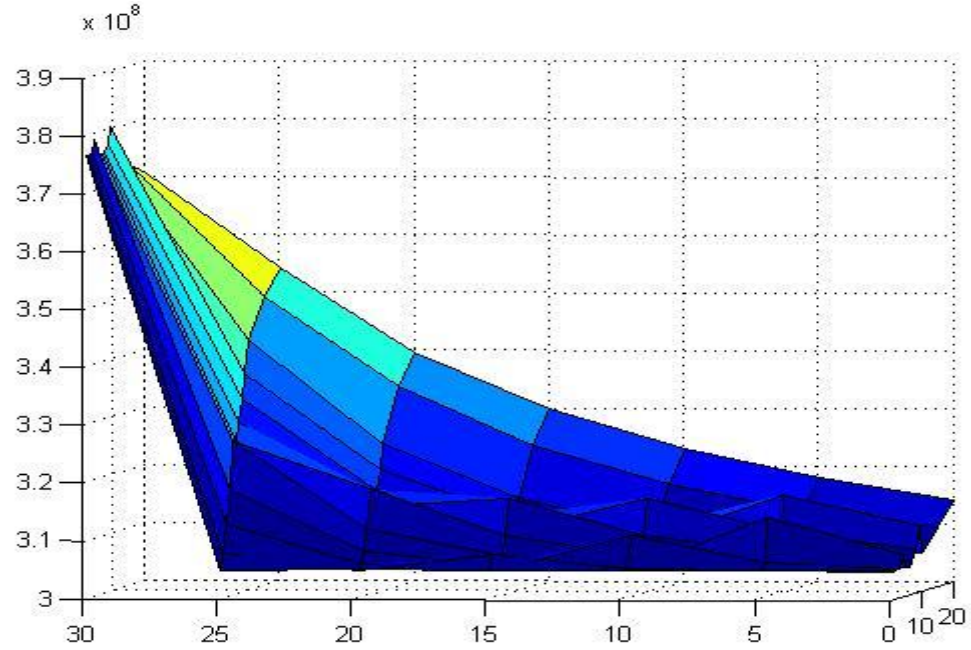
Los modelos con menor suma de los errores absolutos para cada mes fueron:

-Agosto y Septiembre: L8=8, tau8=120



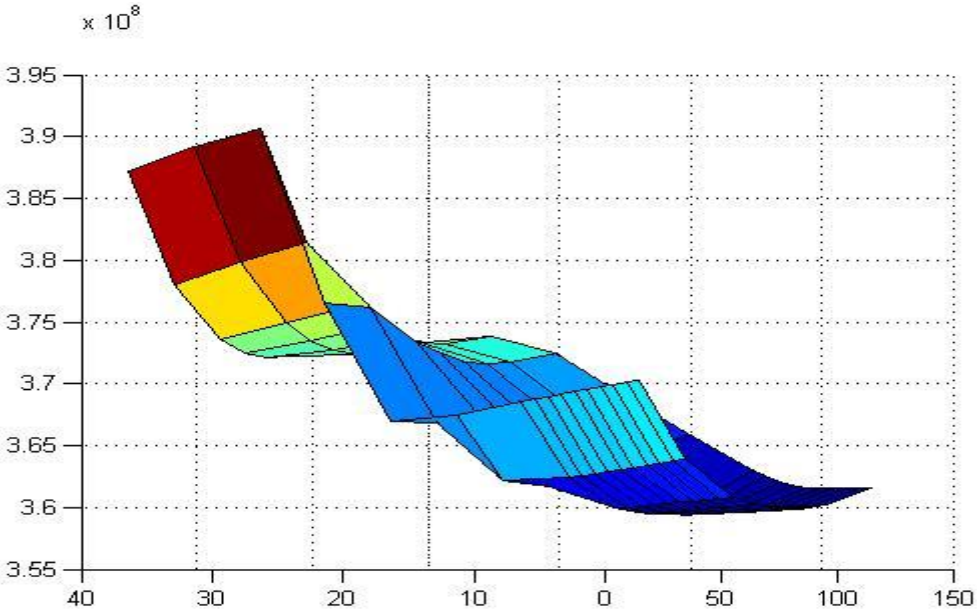
Representación del error en función del retraso L y el tiempo característico tau. Se puede obsevar una tendencia a tener un mínimo a lo largo de una dirección L+tau=cte, ya que el retraso y un tiempo característico largo se pueden compensar.

-Abril y Mayo: L8=8, tau8=80



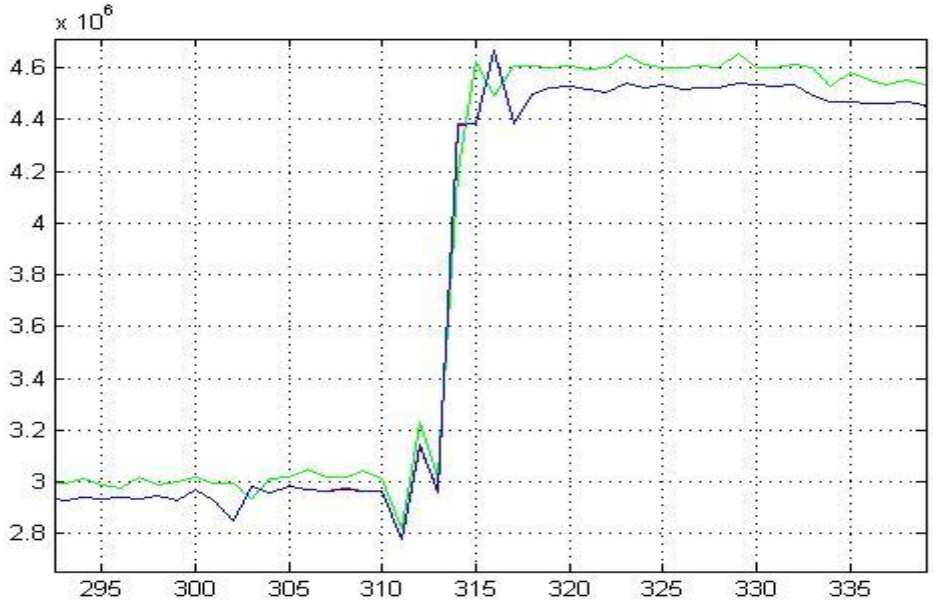
Representación del error en función del retraso L y el tiempo característico tau. Se puede obsevar una tendencia a tener un mínimo a lo largo de una dirección L+tau=cte, ya que el retraso y un tiempo característico largo se pueden compensar.

-Mayo: L8=25, tau8=1



Representación del error en función del retraso L y el tiempo característico tau. Se puede obsevar una tendencia a tener un mínimo a lo largo de una dirección L+tau=cte, ya que el retraso y un tiempo característico largo se pueden compensar.

No tiene sentido que un sistema con un retraso tan pequeño tenga un tiempo de establecimiento tan grande, luego validamos el modelo para L8=25 y tau8=1, pero al igual que en la ecuación anterior podemos despreciar la dinámica y establecer L8=0 y tau8=0. El sumatorio de las pérdidas resulta de 3.0465e+08.



La imagen anterior muestra la validación del modelo en el mes de Octubre con tiempo de muestreo de 30 minutos, L8=0, tau8=0. En rojo está representada la entrada (v8), en verde la salida (v9) y en azul el modelo.

Ecuación 6

Se ha validado un modelo a partir de los datos del mes de Diciembre de 2013.

Se ha calculado el error para un tiempo de muestreo de 30 minutos.

Los valores de retraso y tiempo característico son los siguientes

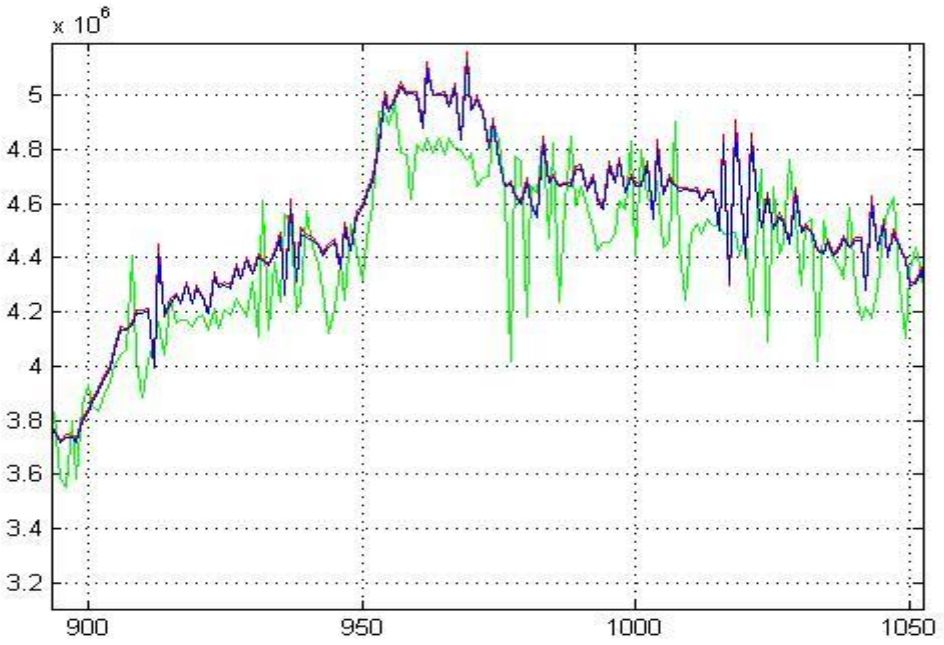
$$L = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 15\ 20\ 25\ 30\ ]$$

$$tau = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 15\ 20\ 25\ 30]$$

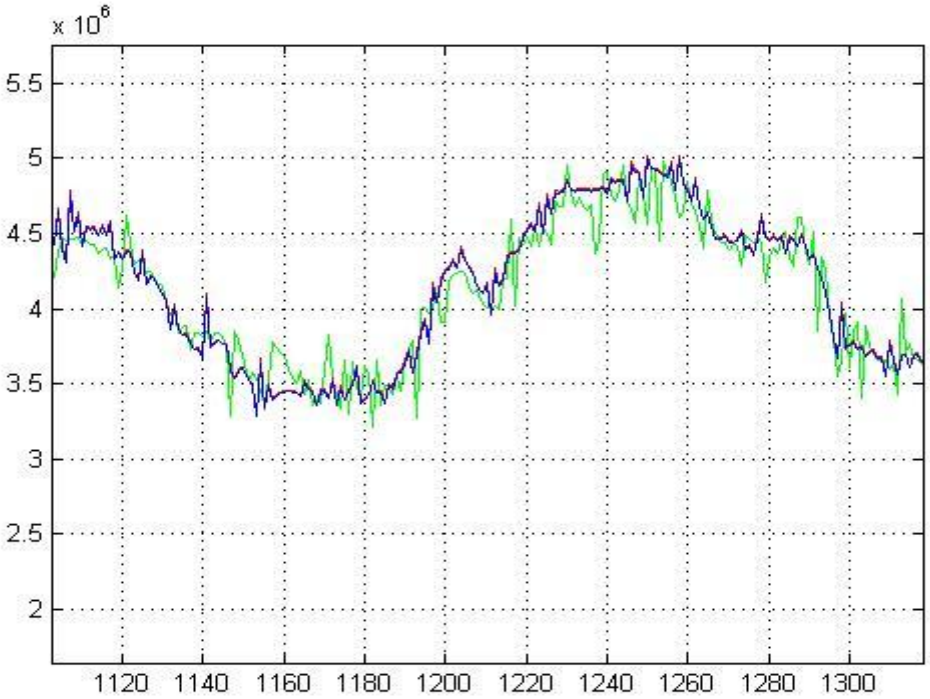
El modelo con menor suma de los errores absolutos fue:

-Diciembre: L6=6, tau6=9, L9=0, tau9=6

Al igual que en ecuaciones anteriores, despreciamos la dinámica y validamos con el modelo con L6, tau6, L9 y tau9 iguales a cero. Para estos parámetros las pérdidas son de 3.4568e+08.



La imagen anterior muestra la validación del modelo en el mes de Octubre con tiempo de muestreo de 30 minutos, L6=0, tau6=0 ,L9=0, tau9=0. En rojo está representada la entrada (v6+v9), en verde la salida (v10+v11+v12+v13+vd) y en azul el modelo.



La imagen anterior muestra la validación del modelo en el mes de Diciembre con tiempo de muestreo de 30 minutos, L6=0, tau6=0 ,L9=0, tau9=0. En rojo está representada la entrada (v6 +v9), en verde la salida (v10 +v11+v12+v13+vd) y en azul el modelo.

Ecuación 7

Se ha validado un modelo a partir de los datos del mes de Octubre de 2013.

Se ha calculado el error para un tiempo de muestreo de 30 minutos.

Los valores de retraso y tiempo característico son los siguientes

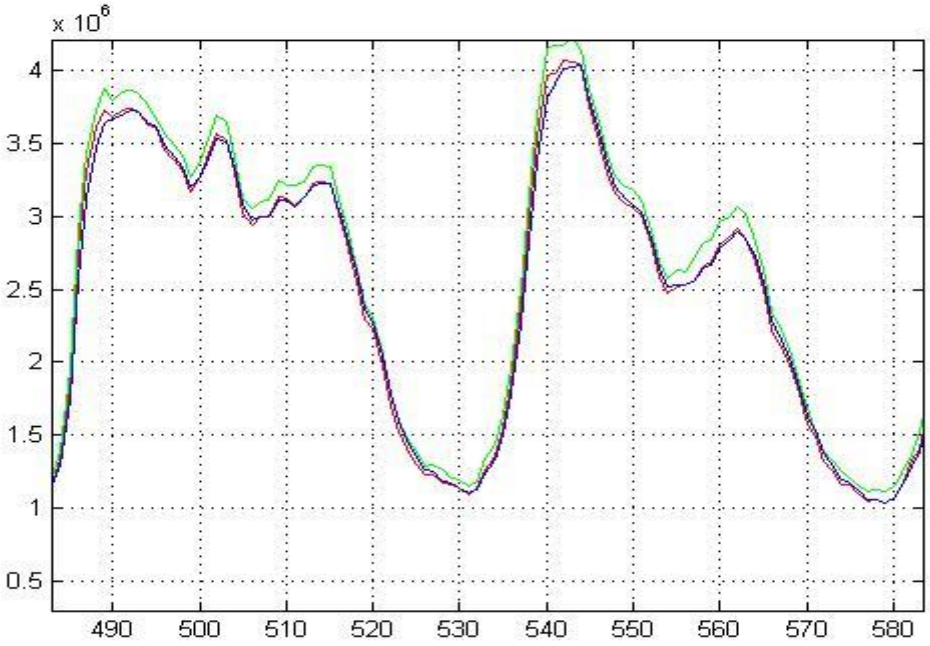
$$L = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 15\ 20\ 25\ 30\ ]$$

$$tau = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 15\ 20\ 25\ 30\ 35\ 40]$$

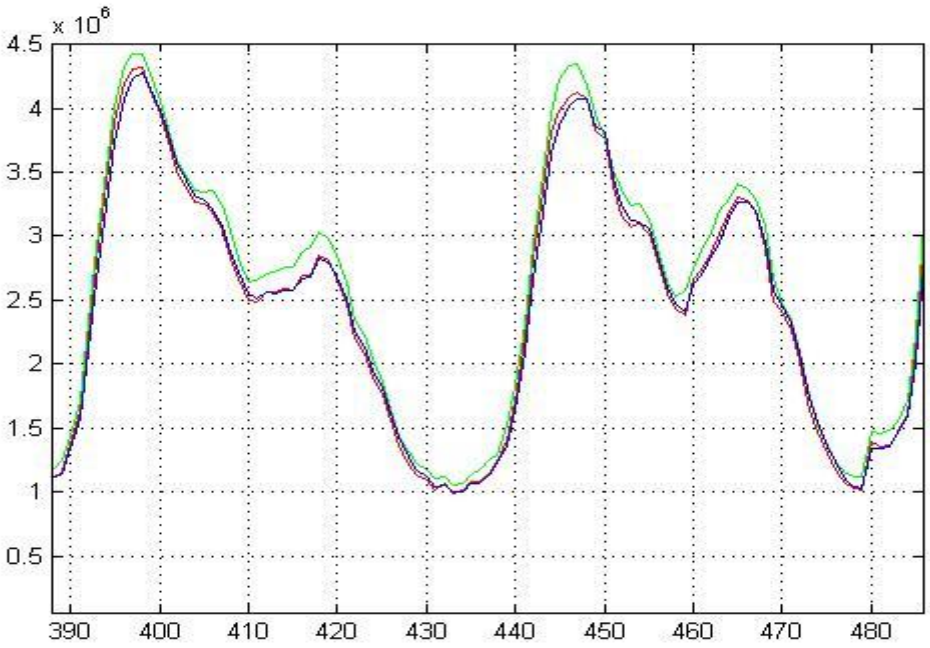
El modelo con menor suma de los errores absolutos fue:

-Octubre: L10=0 tau10=1 L11=15 tau11=40 L12=0 tau12=5

Al igual que en los últimos casos despreciamos la dinámica y validamos el modelo con los parámetros L10, tau10, L11, tau11, L12 y tau12 iguales a cero. Las pérdidas totales resultan de 4.095e+09.



La imagen anterior muestra la validación del modelo en el mes de Octubre con tiempo de muestreo de 30 minutos, L10=0, tau10=0 ,L11=0, tau11=0, L12=0, tau12=0. En rojo está representada la entrada (v10+v11+v12), en verde la salida (v16+v17+v18) y en azul el modelo.



La imagen anterior muestra la validación del modelo en el mes de Noviembre con tiempo de muestreo de 30 minutos, L10=0, tau10=0 ,L11=0, tau11=0, L12=0, tau12=0. En rojo está representada la entrada (v10+v11+v12), en verde la salida (v16+v17+v18) y en azul el modelo.

En resumen, salvo las tres primeras ecuaciones, en el resto de ecuaciones se puede despreciar el efecto de la dinámica en el modelo.

Los resultados han sido :

- Ecuación 1:
  - L1=200
  - tau1=180
- Ecuación 2:
  - L2=50 L4=180
  - tau2=50 tau4=200
- Ecuación 3:
  - L5=100
  - tau5=45
- Ecuación 4:
  - L7=0
  - tau7=0
- Ecuación 5
  - L8=0
  - tau8=0
- Ecuación 6
  - L6=0 L9=0
  - tau6=0 tau9=0
- Ecuación 7
  - L10=0 L11=0 L12=0
  - tau10=0 tau11=0 tau12=0

Corrección de datos

La corrección de datos ha sido realizada en *C++* empleando *Visual Studio 2012 Ultimate* y las librerías *CWLib* y *OPLib*. En particular se ha usado la función *"CWnetworkmodel"*, que ya estaba implementada, para el dimensionado del problema de optimización con corrección estática, y se ha diseñado la función *"CWnetworkmodelDinamica"* para el dimensionado del problema de optimización con corrección dinámica. Ambas funciones se encuentran en la librería *CWLib* mencionada anteriormente.

Una vez se ha dimensionado el problema de optimización, la función *"StrictlyConvexFastGradientCorrection"* resuelve el problema de optimización que se le pasa en la entrada. Esta función se encuentra en la librería *OPLib* mencionada anteriormente. La función de optimización corrige un vector de entrada *"v"* para que cumpla una serie de restricciones, devolviendo un vector corregido *"z"* tal que se minimiza la distancia entre *"v"* y *"z"*. Las restricciones obligan a que el vector *"z"* se mueva entre un límite superior y otro inferior y que, además, las pérdidas tienen que ser positivas.

Matemáticamente:

$$\min_{LB \le z \le UB} (z - v)' \, W \, (z - v)$$

$$s.a. \quad A\, z = b$$

La estructura Network

Antes de comenzar a explicar las funciones que definen el problema de optimización y la función de optimización misma, es necesario explicar la estructura y el uso de la *"struct"* que define nuestra red. Esta estructura es la *"struct Network"*, en ella se recogen todas las ecuaciones de balance de masa de nuestra red, y el orden de las variables que intervienen en dichas funciones.

Los campos de esta estructura son:

- **string** *name* este campo se utiliza para ponerle un nombre a la red.
- **unsigned short int** *Nvar* en este campo se introducen el número de variables (caudalímetros) de los que dispone nuestra red.
- **unsigned short int** *Nequ* en este campo se introducen el número de ecuaciones de balance de masa que componen nuestra red.
- **Variable** *VariableList[100]* vector de estructuras tipo "Variable" que sirve para definir las distintas variables que componen la red.
- **Equation** *EquationList[100]* vector de estructuras tipo "Equation" que sirve para definir las distintas ecuaciones que componen la red.

Accederemos a esta estructura siempre que queramos conocer algún dato de la red o como están ordenadas las variables dentro de cada ecuación.

La estructura “Variable” contiene toda la información correspondiente a las distintas variables de nuestra red. En ella se recoge la dirección “id” de la variable, que sirve para poder ordenarlas en cada ecuación. Aparte, en esta estructura se define si la variable correspondiente es un flujo o un depósito.

Los campos de esta estructura son:

- **IdVar** *id* campo que nos identifica la dirección de la variable en cuestión. Es una estructura de tipo *“IdVar”*. Estas estructuras tienen tres campos, *“group, subgroup* e *id”*, en ellas se identifica la dirección de la variable.
- **string** *name* para ponerle un nombre a la variable en cuestión.
- **Char** *type* aquí se especifica si la variable es un flujo o un depósito. Si es un flujo vale *‘F’* y si es un depósito *‘D’*.
- **double** *aux* valor auxiliar. En caso de variables tipo *‘D’* es la superficie en metros cuadrados.

La estructura “Equation” se utiliza para registrar toda la información de cada una de las ecuaciones, desde que variables intervienen, definiendo si actúan como entrada o como salida, hasta el orden de dichas variables dentro de la ecuación.

Los campos de esta estructura son:

- **string** *name* para ponerle un nombre identificativo a la ecuación.
- **unsigned short int** *Nvar* para conocer el número de variables en cuestión que intervienen en la ecuación.
- **short int** *Tvar[100]* vector que se utiliza para definir si la variable que ocupa la posición *n* actúa como entrada o salida del nodo. Si la *posición n* vale 1, la variable que ocupe la posición *n* en el vector *“var”* actúa como entrada, si por el contrario vale -1, actúa como una salida.
- **IdVar** *var[100]* en este vector se recoge el orden que van a llevar las variables en la ecuación. Para las “Nvar” primeras posiciones recoge el IdVar de la variable en cuestión. Sirve para identificar qué variable está ocupando qué posición y así saber qué variable actúa como entrada y cual como salida.
- **short int** *L[100]* en este vector se establecen los tiempos de retraso en minutos de cada variable de la ecuación.
- **short int** *tau[100]* en este vector se establecen los tiempos característicos en minutos de cada variable de la ecuación.

**los campos L y tau no existían previamente puesto que en proyectos anteriores no se contemplaba la dinámica. Se han creado a lo largo de este proyecto para poder definir la dinámica de los caudales de entrada en los nodos. Esta dinámica la suponemos independiente del valor del tiempo de muestreo, ya que representa el tiempo de transporte del agua.*

Definición del problema de corrección estática

Para realizar la corrección estática se ha utilizado la función *"CWnetworkmodel"* existente en la librería *CWLib*. Esta función se encarga de crear el problema de optimización que se le pasará al algoritmo de optimización. Para ello, a partir de unos datos iniciales que se le suministrará, transformará una serie de matrices para definir nuestro problema. En el problema estático solo se necesita la información de un periodo de integración, pues las correcciones se realizan teniendo en cuenta únicamente el presente tiempo de integración.

void CWnetworkmodel(Network& network, CMatrix& vecEqu, CMatrix& vecVar, CMatrix& Am, CMatrix& bm, CMatrix& Ac, CMatrix& bc, CMatrix& Tc)

- **Network& network** variable de tipo Network por la que le pasaremos todos los datos de nuestra red. En la network se incluyen tanto el número de ecuaciones, como el número de variables y cuales actúan como entrada o salida.
- **CMatrix& vecEqu** este argumento de entrada es un vector de tamaño igual al número de ecuaciones. La primera posición corresponde a la primera ecuación y la última posición a la última ecuación. Todas las componentes de este vector valdrán uno salvo en aquellas en las que no se quiera tener en cuenta dicha ecuación en el algoritmo de optimización, en su lugar valdrán cero.
- **CMatrix& vecVar** vector de tamaño igual al número de variables que indica que variables se quieren corregir. Al igual que el anterior, si vale uno significa que se quiere corregir esa variable y si vale cero la deja tal cual.
- **CMatrix& Am** variable de tipo *CMatrix* que se debe pasar por referencia. La función se encarga de modificarla. Es la matriz que define las ecuaciones de balance sin pérdidas.
- **CMatrix& bm** variable de tipo *CMatrix* que se debe pasar por referencia. La función se encarga de modificarla. Es el término independiente de las ecuaciones de balance sin pérdidas.
- **CMatrix& Ac** variable de tipo *CMatrix* que se debe pasar por referencia. La función se encarga de modificarla. Es la matriz que define las ecuaciones de balance de masa con pérdidas y que posteriormente se le pasará a la función de optimización.
- **CMatrix& bc** variable de tipo *CMatrix* que se debe pasar por referencia. La función se encarga de modificarla. Junto con *"Tc"* forman el término independiente.
- **CMatrix& Tc** variable de tipo *CMatrix* que se debe pasar por referencia. La función se encarga de modificarla. Junto con *"bc"* forman el término independiente.

*el vector *"b"* que se le debe pasar a la función de optimización viene de *"bc + v*Tc"*.

La matriz *"Ac"* es una matriz que tiene tantas filas como ecuaciones de balance de masa tiene la red, y de tantas columnas como variables (sensores) más las pérdidas (una por ecuación). En esta matriz, cada fila corresponde a una ecuación. En dicha fila, las columnas correspondientes a cada variable valdrán 0, 1 ó -1. Cuando una variable no aparezca en dicha ecuación estará a cero, si esa variable aparece en la ecuación y actúa como una entrada estará a 1, y, si esa variable aparece en la ecuación pero actúa como una salida, valdrá -1. Las columnas correspondientes a las pérdidas valdrán 1 cuando esté en la fila de la ecuación en cuestión. Las variables(columnas de la matriz) vendrán ordenadas según venga establecido en la network.

A la matriz *"Ac"* se le pueden añadir más filas en función del valor del vector *"vecVar"*. Si alguna variable del vector *"vecVar"* vale cero significa que no queremos modificar el valor de dicha variable. Eso se consigue añadiendo una nueva ecuación (una fila más) a la matriz *"Ac"*.

Ejemplo.-

Las ecuaciones a considerar serán

$$v1 = v2$$

$$v2 + v3 = v4 + v5$$

**consideramos que tanto el vector vecVar como vecEqu valen 1 todas sus componentes*

Matriz Ac

	v1	*v2*	*v3*	*v4*	*v5*	*e1*	*e2*
Ecuación 1	*1*	*-1*	*0*	*0*	*0*	*1*	*0*
Ecuación 2	*0*	*1*	*1*	*-1*	*-1*	*0*	*1*

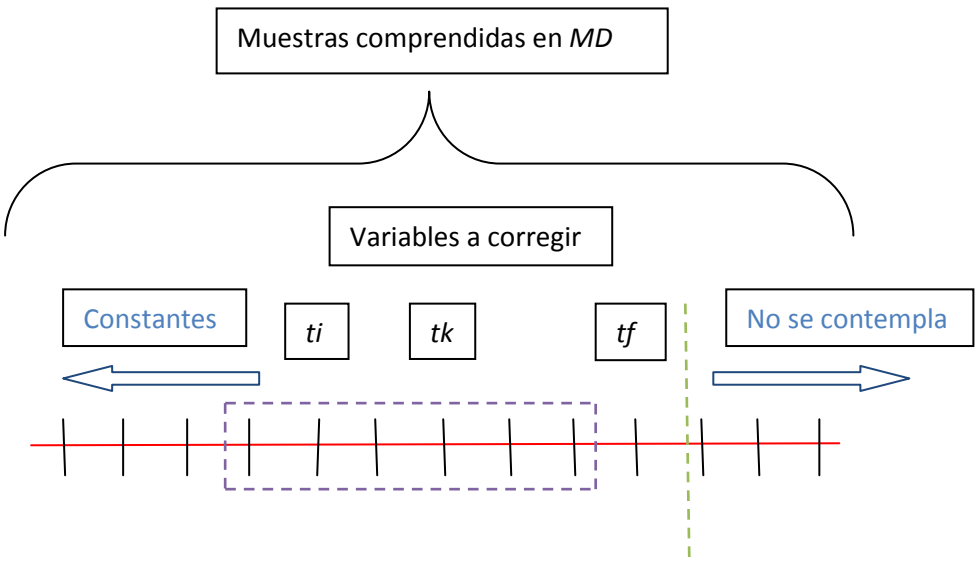
Definición del problema de corrección dinámica

Para realizar la corrección dinámica se ha programado una función llamada *"CWnetworkmodelDinamica"* que se encarga de diseñar el problema de optimización. Esto quiere decir que esta función, a partir de unos datos iniciales que se le tendrá que pasar, será capaz de dimensionar las matrices que se le deben pasar a la función *StrictlyConvexFastGradientCorrection"* para que realice la corrección de datos. Esta función es bastante más compleja que la anterior, pues en las ecuaciones no se tiene en cuenta únicamente el valor de la entrada para el instante en el que se calcula la matriz, si no que también depende de los valores anteriores, lo que resulta en que la dimensión es de la matriz *"A"* es notablemente superior. Además, está función calcula la matriz de pesos *"Ws"* a partir de un vector de pesos *"W"*; creará los vectores que hacen de límites de las variables, *"UB"* y *"LB"*; y, finalmente también diseñará el vector *"v"* que se le pasará a la función de optimización.

A diferencia de la función anterior, donde únicamente se tenían en cuenta los valores de las variables en el presente tiempo de muestreo para el que se quiere realizar la corrección, esta necesita acceder a la matriz *"MD"* donde se almacenan las cantidades de litros integrados para cada diferencial de tiempo (que en nuestro caso serán de treinta minutos), puesto que para realizar la corrección en un instante de tiempo, necesitamos establecer una ventana temporal para acceder a las muestras anteriores.

A nuestra función se le deberá dar de entrada entre que dos fechas se quiere realizar la optimización, extrayendo los datos que sean necesarios de la matriz *"MD"* para poder dimensionar el problema. Además, le deberemos de pasar por referencia tres vectores que la función transformará en *"UB","LB"* y *"v"*, y también una matriz que será dimensionada como la matriz de pesos *"Ws"*.

En el problema estático, la corrección se hacía solo para un vector de la matriz *"MD"* correspondiente a un instante de tiempo. Ahora, tenemos que corregir en bloque, es decir, corregir varios vectores a la vez, tantos vectores como muestras haya entre *tf* y *ti*.



Cuando queremos realizar una corrección en un instante de tiempo concreto, el modelo dinámico nos fuerza a tener en cuenta muestras anteriores. Es por ello que es necesario pasar como argumento de entrada la matriz "MD". Si en alguna de las muestras, en alguna de las variables tenemos que retroceder tanto en el tiempo que se sale de *ti*, almacenaremos el valor de esa variable en ese instante anterior a *ti* como una constante dentro del vector b.

No se va a tener en cuenta a la hora de realizar la corrección, que en muestras cercanas a *tf* el valor de esa corrección influya en correcciones a muestras posteriores a *tf*, lo que forzaría a que a la hora de realizar la corrección también se tuviera que tomar como constantes los valores posteriores a *tf*.

void CWnetworkmodelDinamica(Network& network, CMatrix& vecEqu, CMatrix& vecVar, CMatrix& MD, CMatrix& b, CMatrix& A, CMatrix& LB, CMatrix& UB, CMatrix& W, CMatrix& Ws, CMatrix& v, long long ti, long long tf, double Dt)

- **Network& network** variable de tipo Network por la que le pasaremos todos los datos de nuestra red. En la network se incluyen tanto el número de ecuaciones, como el número de variables y cuales actúan como entrada o salida. La network debe contener para cada ecuación los valores de los retrasos y tiempos característicos de las distintas variables de entrada.
- **CMatrix& vecEqu** este argumento de entrada es un vector de tamaño igual al número de ecuaciones. La primera posición corresponde a la primera ecuación y la última posición a la última ecuación. Todas las componentes de este vector valdrán uno salvo en aquellas en las que no se quiera tener en cuenta dicha ecuación en el algoritmo de optimización, en su lugar valdrán cero.
- **CMatrix& vecVar** vector de tamaño igual al número de variables que indica que variables se quieren corregir. Al igual que el anterior, si vale uno significa que se quiere corregir esa variable y si vale cero la deja tal cual.
- **CMatrix& MD** contiene los litros de agua integrados para cada diferencial de tiempo (en nuestro caso treinta minutos) y para cada variable.
- **CMatrix& b** variable de tipo *CMatrix* que se debe pasar por referencia. La función se encarga de modificarla. Representa el término independiente de las ecuaciones de balance de masa. Almacena la suma de los términos de la variable de entrada retrasados con respecto a la muestra que se está estudiando guardados con signo negativo.
- **CMatrix& A** variable de tipo *CMatrix* que se debe pasar por referencia. La función se encarga de modificarla. Es la matriz que define las ecuaciones de balance de masa con pérdidas y que posteriormente se le pasará a la función de optimización.
- **CMatrix& LB** variable de tipo *CMatrix* que se debe pasar por referencia. La función se encarga de modificarla. Este vector va a representar el límite inferior de los términos del vector *"z"*.
- **CMatrix& UB** variable de tipo *CMatrix* que se debe pasar por referencia. La función se encarga de modificarla. Este vector va a representar el límite superior de los términos del vector *"z"*.
- **CMatrix& W** vector que para cada posición representa el peso de esa variable a la hora de ser optimizada. Generalmente va a ser un vector de unos, menos en aquellas variables que quieras modificar menos, en cuyo lugar se pondrá un número mayor al resto.
- **CMatrix& Ws** variable de tipo *CMatrix* que se debe pasar por referencia. La función se encarga de modificarla. Representa la matriz de pesos del sistema calculada a partir del vector *"W"*.
- **CMatrix& v** variable de tipo *CMatrix* que se debe pasar por referencia. La función se encarga de modificarla. Vector de dimensión (número de variables + número de ecuaciones)*(número de muestras que van entre *"ti"* y *"tf"*). Almacena el valor de las variables y de las pérdidas para cada muestra desde "ti" hasta *"tf"*.
- **long long ti** tiempo en milisegundos desde el que se debe empezar la optimización.
- **long long tf** tiempo en milisegundos hasta el que se debe realizar la optimización.
- **double Dt** tiempo de muestreo en milisegundos con el que se han integrado los caudales de la red. En nuestro caso es de 30 minutos.

La matriz "A" tiene tantas filas como número de ecuaciones tiene la red, multiplicado por el número de tiempos de muestreo que se van a optimizar. El número de instantes de muestreo se calcula como *"ceil((tf-ti)/dt)"*. Al igual que antes, cada fila corresponde a una ecuación de balance de masas pero con la salvedad de que cada ecuación aparecerá tantas veces como instantes de muestreo.

La matriz *"A"* tiene tantas columnas como número de variables más número de pérdidas (una por cada ecuación), multiplicadas por el número de tiempos de muestreo que se van a optimizar. Al igual que antes, cuando en la ecuación aparezca dicha variable (para el instante de muestreo de la ecuación) se pondrá un 1 ó un -1 en función de si actúa como entrada o salida. Para saber que variable ocupa cada columna es necesario acceder a la *network*.

Al igual que antes, dependiendo de los valores de *"vecVar"* y "*vecEqu"*, la matriz *"A"* podrá tener más o menos filas.

El vector *"b"* es un vector columna cuya dimensión debe ser igual al número de filas de *"A"*. Para cada posición almacena el sumatorio de las muestras retrasadas de las variables de entrada que hay que tener en cuenta en la ecuación que corresponda, multiplicadas por su *theta* correspondiente, y almacenándose con signo negativo debido a que debe cumplirse *A*z=b*.

**Nótese que aunque los términos que componen b son también las variables de la red, estas no serán optimizadas, debido a que en el vector b se comportan como constantes. De esta forma, solo los términos incluidos en la matriz A serán corregidos.*

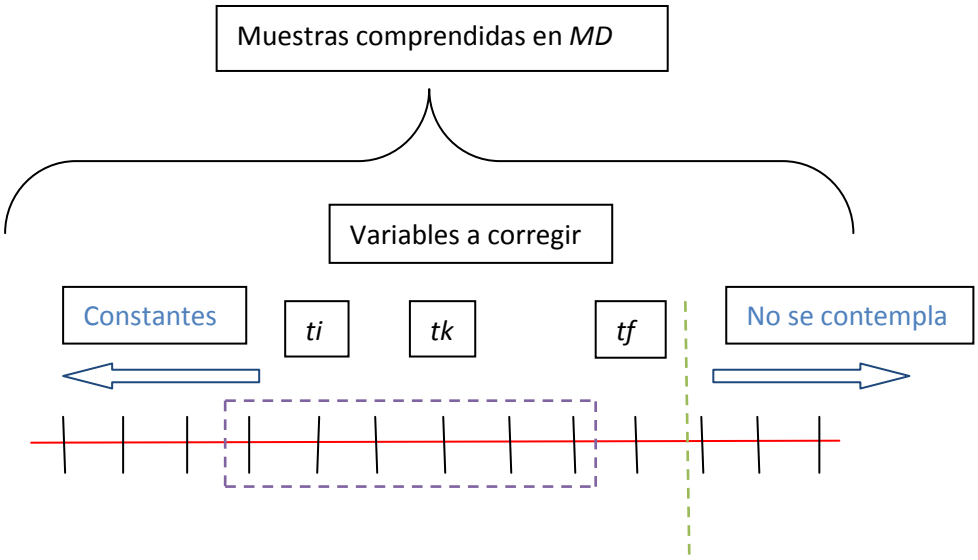
El vector *"LB"* es un vector columna cuya dimensión debe ser igual al número de columnas de *"A"*. Las *"Nvar"* primeras posiciones de este vector corresponden a las distintas variables de nuestra red, ordenadas de acuerdo con la *network*. Seguidamente, las *"Nequ"* posiciones siguientes corresponden a las pérdidas en cada ecuación. Esta estructura se repite tantas veces como instantes de muestreo haya comprendidos entre *ti* y *tf*. Para cada posición registra el límite inferior de la variable que corresponda. El límite inferior tanto para las pérdidas como para las medidas de los sensores debe ser cero, menos en aquellas variables que sean depósitos, cuyas medidas corresponden a incrementos o decrementos del nivel, que pueden ser negativas. En el caso de los depósitos, el límite inferior será *"-MAX_double"*, que correponde a un hipotético menos infinito

El vector *"UB"* es un vector columna cuya dimensión debe ser igual al número de columnas de *"A"*. Las *"Nvar"* primeras posiciones de este vector corresponden a las distintas variables de nuestra red, ordenadas de acuerdo con la *network*. Seguidamente, las *"Nequ"* posiciones siguientes corresponden a las pérdidas en cada ecuación. Esta estructura se repite tantas veces como instantes de muestreo haya comprendidos entre *ti* y *tf*. Para cada posición registra el límite superior de la variable que corresponda. El límite superior tanto para las pérdidas como para las medidas de los sensores es "*MAX_double"*, que corresponde a un hipotético infinito.

El vector *"v"* es un vector columna cuya dimensión debe ser igual al número de columnas de *"A"*. Las *"Nvar"* primeras posiciones de este vector corresponden a las distintas variables de nuestra red, ordenadas de acuerdo con la *network*. Seguidamente, las *"Nequ"* posiciones siguientes corresponden a las pérdidas en cada ecuación. Esta estructura se repite tantas veces como instantes de muestreo haya comprendidos entre *ti* y *tf*.

Para cada posición "*Nvar"* registra el valor de la variable correspondiente, extraído de la matriz *"MD"*, en el tiempo de muestreo *"k"*. En el caso de las pérdidas, las *"Nequ"* siguientes posiciones, deberían calcularse como *"muestras necesarias de las variables de entrada en la ecuación "nequ" en k + valor de b para la fila del tiempo de muestreo k - variables de salida en k"*.

$$e_1^{(k)} = \sum_{i=0}^{m11} \theta_{11}^{i}\, v_1^{(k-LM_{11}-i)} - v_2^{(k)}$$



El término de $v_2^{(k)}$ lo sacamos directamente de *"MD"*, al igual que los términos del sumatorio que estén comprendidos entre *ti* y *tf*. El resto de términos del sumatorio corresponden a la fila del vector *b* correspondiente a la ecuación de la pérdida que estamos calculando, en el instante correspondiente.

El vector corregido *"z"* tendrá la misma estructura que *"v"*, pero todos los términos serán mayores que cero menos en aquellas variables que correspondan a las medidas de un depósito, debido a que los sensores en un depósito miden incrementos de nivel, y estos pueden resultar negativos.

La matriz *"Ws"* es una matriz diagonal cuadrada de tamaño el mismo número de columnas que *"A"*. Básicamente consiste en poner el vector *"W"* de entrada en la diagonal de *"WS"* tantas veces como instantes de muestreo.

-*Descripción del algoritmo*

El algoritmo, empieza calculando la *"matriz de thetas"*. Esta matriz almacena para cada columna los valores de las *thetas* de cada variable, calculadas a partir de los valores *L* y *tau* de dichas variables. La correspondencia entre columna-variable está ordenada según establece la *network*, al igual que con la matriz *A*. Realmente esta matriz es una gran matriz de ceros, de tantas columnas como variables y de 100 filas. Para cada columna, se rellenan tantas filas como thetas sean calculadas para esa variable, empleando el mismo método de cálculo que en la identificación dinámica. Si alguna variable tiene una dinámica despreciable (*L* y *tau* son cero) únicamente tendrá una fila rellena con valor 1 ($\theta_0 = 1$).

```
//calculo de la matriz de thetas, cada columna contiene las thetas correspondientes a las variables de entrada
for (i=0;i<nequ;i++)
{
    for (j=0; j<network.EquationList[i].Nvar; j++)
        {
            if (network.EquationList[i].Tvar[j]==1)
            {
                k=1;
                L=network.EquationList[i].L[j]*60*1000;
                tau=network.EquationList[i].tau[j]*60*1000;
                LM(0,CWvarindex(network,network.EquationList[i].var[j]))=floor(L/Dt);
                EL=L-LM(0,CWvarindex(network,network.EquationList[i].var[j]))*Dt;
                while (k<=((6*tau+EL)/Dt))
                {
                    if((k-1)*Dt>=EL)
                    {
                    theta(k-1,CWvarindex(network,network.EquationList[i].var[j]))=exp(-(double)((k-1)*Dt-EL)/tau)-
                        exp(-(double)(k*Dt-EL)/tau);
                    }
                    else
                    {
                    theta(k-1,CWvarindex(network,network.EquationList[i].var[j]))=1-exp(-(k*Dt-EL)/tau);
                    }
                    k++;
                }
```

La figura anterior corresponde a una extracción del código de la función donde se muestra la sección del cálculo de la "matriz de thetas".

Una vez calculada *"la matriz de thetas"* se procede a calcular la matriz *A*. Esta matriz, para cada instante de muestreo entre *ti* y *tf*, recorre el vector de ecuaciones, y para cada ecuación, recorre el vector de variables. Comprueba si la variable es de entrada (valor 1 en la network), si es así, calcula a que muestra retrasada corresponde esa variable. Si esa muestra está comprendida entre *ti* y *tf*, en la columna correspondiente a esa variable en el instante adecuado, escribe el valor de *theta* correspondiente. Si esa muestra no está comprendida entre *ti* y *tf* tiene que buscar el valor de dicha variable en dicha muestra en la matriz *MD*, multiplicarla por la *theta* correspondiente, y sumarla a la fila correspondiente del vector *b*. este proceso se repetirá tantas veces como número de *thetas* tenga en la *"matriz de thetas"* dicha variable.

Si la variable es de salida (vale -1 en la *network*), simplemente se pondrá un -1 en la columna correspondiente a esa variable en el instante que corresponda.

Una vez se ha recorrido el vector de variables, se pone un -1 en la columna correspondiente a la ecuación que se está estudiando en ese momento, en el instante de muestreo correspondiente.

Para generar los vectores *"LB"* y *"UB"* tenemos que recorrer las *"Nvar"* primeras posiciones, que corresponden a las variables ordenadas según la *network*. Luego las *"Nequ"* siguientes posiciones, que corresponden a las pérdidas de cada ecuación. Este proceso lo tendremos que repetir tantas veces como instantes de muestreo haya entre *ti* y *tf*.

```
//vectores UB y LB

for(tk=0;tk<dimk;tk++)
{
    for(i=0;i<nvar+nequ;i++)
    {
        if((network.VariableList[i].type == 'F') || i>=nvar)
        {
            LB(i+tk*(nvar+nequ),0)=0;
            UB(i+tk*(nvar+nequ),0)=MAX_double;
        }
        if(network.VariableList[i].type == 'D')
        {
            LB(i+tk*(nvar+nequ),0)=-MAX_double;
            UB(i+tk*(nvar+nequ),0)=MAX_double;
        }
    }
}
```

La figura anterior corresponde a un extracto del código de la función donde se calculan los vectores UB y LB.

Para generar el vector "v" tenemos que recorrer las *"Nvar"* primeras posiciones, que corresponden a las variables ordenadas según la *network*. Luego las *"Nequ"* siguientes posiciones, que corresponden a las pérdidas de cada ecuación. Este proceso lo tendremos que repetir tantas veces como instantes de muestreo haya entre *ti* y *tf*.

Para ordenar las variables según la network se emplea una función de CWLib llamada CWVarindex. Esta función recibe como argumento de entrada la network y la id de la variable que se quiere colocar. La función busca en el vector VariableList de la network la id correspondiente a la variable que quiero colocar, cuando la encuentra, devuelve como argumento de salida la posición de ese vector, y esa posición es la posición que ocupa esa variable tanto en las columnas de la matriz A, como en los vectores v, LB y UB.

Para las "Nvar" posiciones sacamos los valores correspondientes de la matriz "MD" en el instante de tiempo correspondiente y los guardamos en "v".

Para las "Nequ" posiciones es necesario calcular dicho valor según la ecuación:

$$e_1^{(k)} = \sum_{i=0}^{m11} \theta_{11}^{i}\, v_1^{(k-LM_{11}-i)} - v_2^{(k)}$$

El término de $v_2^{(k)}$ lo sacamos directamente de *"MD"*, al igual que los términos del sumatorio que estén comprendidos entre *ti* y *tf*. El resto de términos del sumatorio corresponden a la fila

del vector *b* correspondiente a la ecuación de la pérdida que estamos calculando, en el instante correspondiente.

```
//vector v de salida
for (k=0,j=0;j<nfil&&!k;j++)
{
    if(ti==MD(j,0))
    {
        k=j;
    }
}
for(tk=0;tk<dimk;tk++)
{
    for(i=0;i<nvar+nequ;i++)
    {
        if (i<nvar)
        {
        v(i+tk*(nequ+nvar),0)=MD(k+tk,i+1);
        }
        else
        {
            for(j=0;j<(nvar+nequ)*dimk;j++)
            {
                if (j!=i+(nvar+nequ)*tk)
                {
                v(i+tk*(nequ+nvar),0)=v(i+tk*(nequ+nvar),0)+A(i-nvar+tk*nequ,j)*v(j,0);
                }
            }
            v(i+tk*(nequ+nvar),0)=v(i+tk*(nequ+nvar),0)-b(i-nvar+tk*nequ,0);
        }
    }
}
```

La figura anterior corresponde a un extracto del código de la función donde se calcula el vector v.

La matriz *"Ws"* simplemente se calcula repitiendo el vector *"W"* de entrada, en la diagonal de matriz, tantas veces como instantes de muestreo hay entre *ti* y *tf*.

```
//Matriz de pesos Ws, suponemos que el vector W de entrada es un vector columna

for(tk=0;tk<dimk;tk++)
{
    for(i=0;i<nvar+nequ;i++)
    {
        for (j=0;j<nvar+nequ;j++)
        {
            if(i==j)
            {
                Ws(i+(nvar+nequ)*tk,j+(nvar+nequ)*tk)=W(i,0);
            }
        }
    }
}
```

La figura anterior corresponde a un extracto del código de la función donde se calcula la matriz de pesos Ws.

Algoritmo de optimización

Una vez definido el problema de cualquiera de las dos formas, procedemos a introducirlo en la función que realiza la corrección de datos. Esta función se encuentra en la librería *OPLib*. La función pasa como argumento de salida una estructura tipo *'struct ToptSolution'*. El campo *'Z'* de esta estructura corresponde al vector *'v'* que se le pasa a la función como argumento de entrada, pero ya corregido.

TOptSolution StrictlyConvexFastGradientCorrection(CMatrix& vecv, CMatrix& matA, CMatrix& vecb, CMatrix& matW, CMatrix& vecLB, CMatrix& vecUB)

- **CMatrix& vecv** referencia al vector *"v"* que hay que corregir. Para el problema estático es necesario crearlo aparte de la función. Para el problema dinámico es un argumento de salida de la función.
- **CMatrix& matA** referencia a la matriz *"A"* del problema, que debe tener el mismo número de columnas que la longitud de *"v"*. Esta matriz es *"Ac"* para el caso estático y *"A"* para el dinámico.
- **CMatrix& vecb** referencia al vector *"b"* del problema, que debe tener tantos términos como filas de *"A"*. Para el caso dinámico, este vector se introduce tal cual, es un argumento de salida de la función. En el caso estático se calcula como "*bc+ v*Tc"*.
- **CMatrix& matW** referencia a la matriz de pesos *"W"* del problema. Se asume que *"W"* es diagonal de dimensión igual a la longitud del vector *"v"*. Puede pasarse como matriz, como un vector con la diagonal de dicha matriz, u omitirse (con lo que tomaría un valor por defecto igual a la identidad). En el caso dinámico es el argumento de salida de la función *"Ws"*.
- **CMatrix& vecLB** referencia al vector *"LB"* del problema, que contiene los limites inferiores para el vector solución *"z"*. Debe tener una longitud igual a la del vector *"v"*. Si no se introduce significa que no se aplican. En el caso dinámico, es el argumento de salida *"LB"*.
- **CMatrix& vecUB** referencia al vector *"UB"* del problema, que contiene los límites superiores para el vector solución *"z"*. Debe tener una longitud igual a la del vector *"v"*. Si no se introduce toma un valor por defecto de *"MAX_double"* (a efectos prácticos, como si no existiese límite superior). En el caso dinámico, es el argumento de salida *"UB"*.

*No importa si los vectores son vectores fila o columna, el algoritmo asume internamente que son todos vectores columna. En caso de que los limites introducidos sean incompatibles (es decir, que "*UB[i] < LB[i]"* para algún i) se primara el valor de *"LB"* sobre el de *"UB"* y se forzara a que *"UB[i]"* sea igual a *"LB[i]"*.

Esta función corrige un vector *"v"*, devolviendo un vector corregido *"z"* tal que se minimiza la distancia entre *"v"* y *"z"*, y tanto las pérdidas como las variables que no sean depósitos son mayores que cero.

Matemáticamente:

$$\min_{LB \le z \le UB} (z - v)' \, W \, (z - v)$$

$$s.a. \quad A \, z = b$$

Ejemplo de corrección estática.-

La red viene definida por las siguientes ecuaciones

$$e1 = v1 - v2$$

$$e2 = v2 + v3 - v4 - v5$$

La red después de la corrección será

$$e1c = v1c - v2c$$

$$e2c = v2c + v3c - v4c - v5c$$

Los vectores v y z serán

$$v = \begin{bmatrix} v1(k) \\ \vdots \\ v5(k) \\ e1(k) \\ e2(k) \end{bmatrix} \quad z = \begin{bmatrix} v1c(k) \\ \vdots \\ v5c(k) \\ e1c(k) \\ e2c(k) \end{bmatrix}$$

La matriz W, que representa la matriz de pesos, será diagonal, cuya diagonal será

$$\begin{bmatrix} w1 \\ \vdots \\ w7 \end{bmatrix}$$

Se quiere que la distancia al cuadrado entre v y z sea mínima, y que a su vez, los términos vengan limitados tanto por arriba como por debajo, por unos vectores llamados UB y LB respectivamente.

$$\min_{LB \le z \le UB} (z - v)' \, W \, (z - v)$$

Para nuestro caso, no existe límite superior, pero queremos que nuestras variables no sean menores que cero salvo en los casos que correspondan a un depósito.

Ejemplo de corrección dinámica.-

La red viene definida por las siguientes ecuaciones

$$e1(k) = 0.6 \cdot v1(k-1) + 0.4 \cdot v1(k-2) - v2(k)$$

$$e2(k) = v2(k-1) - v3(k)$$

La red después de la corrección será

$$e1c(k) = 0.6 \cdot v1c(k-1) + 0.4 \cdot v1c(k-2) - v2c(k)$$

$$e2c(k) = v2c(k-1) - v3c(k)$$

Teniendo en cuenta que hay "kf" instantes de muestreo, los vectores v y z serán

$$v = \begin{bmatrix} v1(k0) \\ \vdots \\ v3(k0) \\ e1(k0) \\ e2(k0) \\ v1(k1) \\ \vdots \\ v3(k1) \\ e1(k1) \\ e2(k1) \\ \vdots \\ v1(kf) \\ \vdots \\ v3(kf) \\ e1(kf) \\ e2(kf) \end{bmatrix} \qquad z = \begin{bmatrix} v1c(k0) \\ \vdots \\ v3c(k0) \\ e1c(k0) \\ e2c(k0) \\ v1c(k1) \\ \vdots \\ v3c(k1) \\ e1c(k1) \\ e2c(k1) \\ \vdots \\ v1c(kf) \\ \vdots \\ v3c(kf) \\ e1c(kf) \\ e2c(kf) \end{bmatrix}$$

La matriz W, que representa la matriz de pesos, será diagonal, cuya diagonal será

$$W = \begin{bmatrix} w1 \\ \vdots \\ w3 \\ \vdots\ kf\ veces \\ w1 \\ \vdots \\ w3 \end{bmatrix}$$

Se quiere que la distancia al cuadrado entre v y z sea mínima, y que a su vez, los términos vengan limitados tanto por arriba como por debajo, por unos vectores llamados UB y LB respectivamente.

$$\min_{LB \leq z \leq UB} (z - v)' \, W \, (z - v)$$

Para nuestro caso, no existe límite superior, pero queremos que nuestras variables no sean menores que cero salvo en los casos que correspondan a un depósito. Las dimensiones de los vectores UB y LB son iguales a las de v y z.

Resultados de las correcciones

En este apartado vamos a recoger y analizar los resultados de haber aplicado el algoritmo *StrictlyConvexFastGradientCorrection* tanto al caso estático como al dinámico.

Para ello, primeramente se ha obtenido la matriz *"MD"* de medidas integradas entre una fecha inicial y otra final, con un paso de integración de 30 minutos. Realmente, la matriz *"MD"* empieza con las medidas integradas de un día anterior a la fecha inicial. Esto es debido a que para aplicar el algoritmo de corrección al caso dinámico, va a ser necesario mirar las muestras anteriores a *ti*, por ello necesitamos tenerlas en la matriz *"MD"*.

En nuestro caso, la matriz *"MD"* va a contener las medidas integradas de todo un mes, definido por la fecha inicial y la fecha final. Teniendo en cuenta que va a haber una muestra cada media hora, en un día habrá 48 muestras, y suponiendo que un mes tiene 30 días esto hace un total de 1440 muestras. Cada muestra corresponde a una fila en la matriz *"MD"*, y dado que hay 18 variables, la matriz *"MD"* tiene 19 columnas (la primera es para la fecha). El gran tamaño de esta matriz provoca que si se quiere corregir de golpe dinámicamente (con la función *CWnetworkmodelDinamica*) todo el mes, se dimensione un problema enorme que el algoritmo de corrección no será capaz de resolver.

En el caso estático no se da este problema puesto que la corrección se hace muestra por muestra, es decir, fila por fila. Para ello, es necesario hacer un *"bucle for"* que recorra todas las filas de *MD*, y para cada fila, genera un vector *"v"* cuyos primeras *"Nvar"* posiciones corresponden a las *"Nvar"* columnas de la matriz *MD*. Las *"Nequ"* posiciones siguientes corresponden al cálculo de las pérdidas según la ecuación de la que se trate. Una vez se crea el vector *"v"*, se llama a la función *CWnetworkmodel*, y después, con los argumentos de salida de esta función, se llama a la función *StrictlyConvexFastGradientCorrection*. Esta función genera como argumento de salida una estructura tipo *TOptSolution*, donde uno de sus campos es el vector *"z"*, que es el vector "*v"* corregido. Las *"Nvar"* primeras posiciones se guardan en la matriz *"MS"*, que es una matriz de las mismas dimensiones de *MD* pero que guarda las medidas ya corregidas estáticamente. Las *"Nequ"* siguientes se guardan en la matriz *"PS"*, que es una matriz con tantas filas como *"MD"* y *"Nequ"* columnas, donde guardan los resultados de las pérdidas ya corregidas estáticamente. Cada columna corresponde a una ecuación distinta, siendo la primera columna para la primera ecuación y la séptima columna para la séptima ecuación.

Tanto para el caso estático como para el caso dinámico, los vectores *"vecVar"* y *"vecEqu"* que añaden restricciones al problema de corrección van a ser iguales. *vecVar* me permite fijar una variable para que el algoritmo de corrección la deje como estaba inicialmente y no la corrija. En nuestro caso, vamos a fijar las variables correspondientes a los depósitos (variables 14 y 15) debido a que el algoritmo de corrección tiende a modificarlas mucho. El algoritmo de corrección las modifica mucho debido a que son las únicas variables que no tienen restricciones en cuanto a su valor, pueden ir desde menos infinito hasta más infinito, puesto que la medida en dichas variables representa una variación de nivel. Además, solo aparecen en la ecuación 6, con lo cual modificar el valor de esas variables solo repercute en esa ecuación.

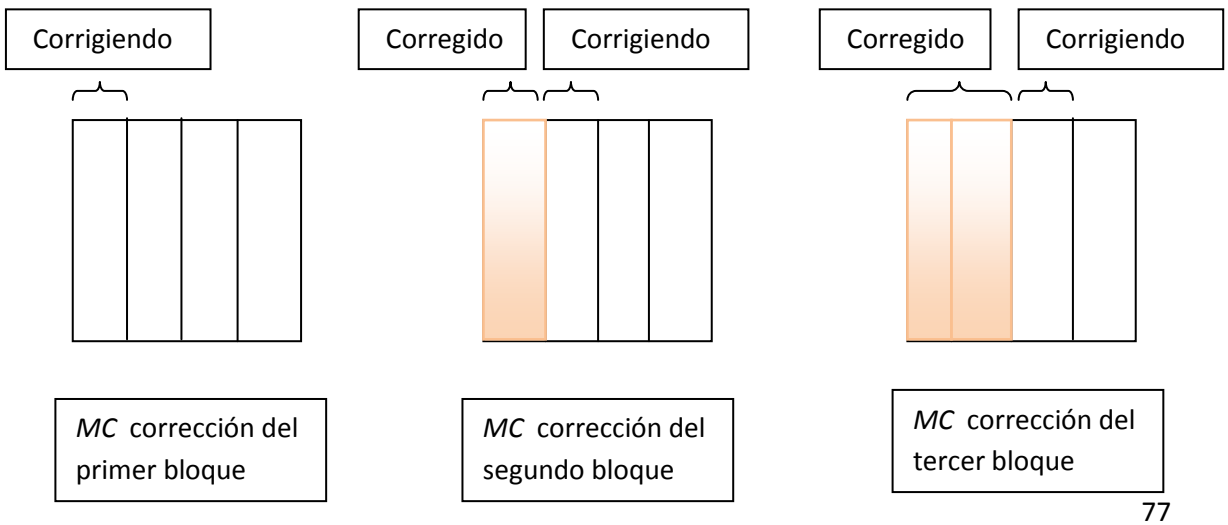
vecEqu me permite despreciar algunas ecuaciones en la corrección provocando que no sean corregidas. En nuestro caso vamos a tener todas en cuenta, luego va a ser un vector de unos.

```
//Corrección estática
    for(i=0;i<nfilas;i++)
    {
        for(j=0;j<nvar+nequ;j++)
            {
                if (j<nvar)
                {
                    vs(j,0)=MD(i,j+1);
                }
                else
                {
                    vs(j,0)=PD(i,j-nvar);
                    /*vs(j,0)=0;*/
                }
            }
        CWnetworkmodel(network,vecEqu,vecVar,
                    Am, bm,
                    Ac, bc,Tc);
        structuraopt=StrictlyConvexFastGradientCorrection(vs, Ac, bc+Tc*vs, W, LBs, UBs);
        for(j=0;j<nvar+nequ;j++)
        {
            if (j<nvar)
            {
                MS(i,j+1)=structuraopt.Z[j];
            }
            else
                PS(i,j-nvar)=structuraopt.Z[j];
        }
    }
```

La imagen anterior es un extracto del código de la librería de TestCWLib, donde se aprecia el procedimiento para realizar la corrección estática.

En el caso dinámico no se puede hacer la corrección del mes completo debido al tamaño del problema. Por ello, se hacen unas correcciones por bloques de unas cuantas horas. Cuando se corrige un bloque se fija en la matriz *"MC",* y para el cálculo del siguiente bloque se pasa como argumento de entrada a la función *CWnetworkmodelDinamica* la matriz *MC*, ya actualizada con la corrección anterior. Originariamente, la matriz *MC* es igual a la matriz *MD*, pero cada vez que se corrige un bloque, la matriz *MC* se actualiza, provocando que para el siguiente bloque, los datos iniciales de los que parte están parcialmente corregidos.

Para aplicar esto, realizamos un bucle que se recorra tantas veces como bloques vaya a haber. El número de bloques depende inversamente del tamaño de los mismos. Para cada iteración, llamamos a la función *CWnetworkmodelDinamica*, cuyos argumentos de salida son los de entrada de *StrictlyConvexFastGradientCorrection*. A continuación, usamos el vector *"Z"*, sacándolo de la estructura de salida del algoritmo de corrección, para actualizar la matriz *MC*. El vector *"Z"* tiene la misma estructura que el vector *"v"*. Para el caso dinámico esto es, las *"Nvar"* primeras posiciones corresponden a las medidas de las variables, y las siguientes *"Nequ"* posiciones corresponderían a las pérdidas, calculadas según el modelo. Toda esta estructura se repite tantas veces como muestras haya entre *ti* y *tf*, que en nuestro caso es el tamaño del bloque que vamos a corregir.



```
//correccion dinamica
vecEqu.resize(nequ,1,1.0);
vecVar.resize(nvar,1,1.0);
vecVar(13,0) = 0.0;
vecVar(14,0) = 0.0;
for (i=y; i<nfilas-k;i=i+k)
{
    ti=fechaIni+(i-y)*dt;
    tf=ti+k*dt;
    CWnetworkmodelDinamica(network,vecEqu,vecVar, MC, b, A, LB, UB,  W,  Ws,  v,ti, tf, dt);
    structuraopt=StrictlyConvexFastGradientCorrection(v, A, b, Ws, LB, UB);
    for (j=0;j<k;j++)
    {
        for (s=0;s<(nvar+nequ);s++)
        {
            if (s<nvar)
            {

                MC(i+j,s+1)=structuraopt.Z[s+j*(nvar+nequ)];
            }
            else
            {

                PC(i+j,s+1-nvar)=structuraopt.Z[s+j*(nvar+nequ)];
            }
        }
    }
}
```

La imagen anterior es un extracto del código de la librería de TestCWLib, donde se aprecia el procedimiento para realizar la corrección dinámica. La inicialización de "i" a "y" en el principal bucle for es porque "y" es la fila de la matriz MD correspondiente a la fecha deseada de inicio de la corrección.

Implementación de los algoritmos en diferentes problemas de corrección

Se han obtenido resultados para diferentes pruebas, tanto para la estática como para la dinámica.

> *-Prueba 1: corrección de la matriz MD tanto de forma dinámica como estática, anulando la corrección en los términos de los depósitos.*
>
> *-Prueba 2: corrección de la matriz MD corrección de la matriz MD tanto de forma dinámica como estática, anulando la corrección en los términos de los depósitos, y partiendo de un valor inicial de las pérdidas igual a cero.*
>
> *-Prueba 3: corrección de la matriz MD tanto de forma dinámica como estática, anulando la corrección en los términos de los depósitos, y modificando la matriz de pesos de forma que las pérdidas tengan mucho menos peso que el resto.*

En las dos primeras pruebas, la matriz de pesos *W* tenía un valor de *1* para todas las componentes, dándole así la misma importancia a cada variable.

Los vectores *vecVar* y *vecEqu* tienen un valor de 1 en todas sus posiciones, salvo las posiciones que corresponden a las depósitos en *vecVar*, que son cero para evitar que sean corregidos.

Los límites inferiores son cero para todas las variables menos para los depósitos, dado que representa un incremento de nivel que puede resultar negativo.

Los límites superiores son todos *"MAX_value"*, que se supone infinito.

Para todas las pruebas se debe cumplir que las pérdidas corregidas sean mayores que cero, así como las variables que no sean depósitos. También, se debe cumplir la igual *A*b=z*.

La mejor forma para comprobar la condición de las perdidas es, que a la hora de representar las ecuaciones, la gráfica de la entrada corregida siempre esté por encima de la de la salida corregida, provocando que las pérdidas sean mayores de cero.

La condición *A*b* está garantizada siempre que el algoritmo de corrección haya convergido antes de alcanzar el número máximo de iteraciones.

Un indicador para saber como de buena ha sido la corrección es el de la *suma de las modificaciones en cada variable al cuadrado*. Este indicador nos va a servir para poder comparar las distintas pruebas que vamos a realizar. Cuanto menos haya tenido que tocar el algoritmo de corrección las variables para alcanzar el resultado óptimo, mejor va a ser esa corrección.

La gran suposición y fundamentación de este proyecto es, que considerando la dinámica, nos debería de salir una corrección mejor que de forma estática. Es decir, que la *suma de las modificaciones en cada variable al cuadrado* debería de salir más pequeña en el caso dinámico que en el estático, al menos, para las ecuaciones donde haya una mayor dinámica.

$$SMC = \sum_{i=1}^{M}\sum_{j=1}^{N}(MD(i,j) - MC(i,j)$$

Siendo *M* y *N* el número de filas y de columnas de la matriz *MD*.

Prueba 1.-

En esta prueba vamos a corregir la matriz *MD* simplemente evitando las correcciones en los términos de los depósitos (*vecVar).* La matriz de pesos va a ser una matriz identidad que propicia que todas las variables tengan el mismo peso en la corrección. Las pérdidas iniciales con las que se va a iniciar la corrección, es decir, las del vector "*v"*, van a tener un valor según las ecuaciones de balance de masa en los nodos, ya sea dinámicamente o estáticamente.

Se ha calculado el valor del *SMC* tanto para el caso dinámico como para el estático para diferentes ecuaciones, en el mes de Mayo de 2013, siendo los resultados:

Resultados para las dos primeras ecuaciones

Corrección dinámica=5.3931e+15

Corrección estática=8.3919e+15

Resultados para las tres primeras ecuaciones

Corrección dinámica= 5.4904e+15

Corrección estática= 1.1021e+16

Resultados para todas las ecuaciones

Corrección dinámica= 4.0910e+16

Corrección estática= 2.4584e+16

A la vista de los resultados, se puede observar que calculando el *SMC* para toda la matriz, teniendo en cuenta todas las ecuaciones, el problema de corrección dinámica introduce un mayor número de modificaciones que el estático, cosa que no resulta del todo intuitiva.

Si observamos los datos con más detenimiento, vemos que en las tres primeras ecuaciones las modificaciones son menores para el caso dinámico que para el estático. Recordemos que las tres primeras ecuaciones son las únicas que tienen en cuenta la dinámica del sistema, puesto que el resto de ecuaciones tiene una dinámica despreciable. Esto quiere decir, que para las ecuaciones que realmente tienen un carácter dinámico, la corrección dinámica resulta mucho mejor que la estática.

Prueba 2.-

Para esta prueba, aparte de anular las correcciones en los términos de los depósitos, vamos a partir de un valor inicial de las pérdidas igual a cero. Esto se consigue forzando en el vector *"v"* a cero las posiciones correspondientes a las pérdidas de las ecuaciones.

Se ha calculado el valor del *SMC* tanto para el caso dinámico como para el estático para diferentes ecuaciones, en el mes de Mayo de 2013, siendo los resultados:

Resultados para las 2 primeras ecuaciones (son las q tienen dinámica) y todos los instantes de tempo

Corrección dinámica=5.9958e+15

Corrección estática=8.5680e+15

Resultados para las 3 primeras ecuaciones y todos los instantes de tiempo

Corrección dinámica =6.1373e+15

Corrección estática=1.1227e+16

Resultados para todas las ecuaciones y todos los instantes de tiempo

Corrección dinámica =4.2059e+16

Corrección estática=2.7397e+16

De estos datos se extrae las mismas conclusiones que en la prueba anterior, con la salvedad de que al forzar las pérdidas a un valor inicial nulo, parece que el algoritmo modifica todavía más las variables.

Prueba 3.-

En esta prueba, como en las dos anteriores, vamos a obligar al algoritmo de corrección a que no modifique los términos de los depósitos. Además, vamos a modificar la matriz de pesos *W*, dándole mucho más peso a las variables que a las pérdidas, de forma que le sea más fácil modificar las pérdidas que las variables. Concretamente, le hemos puesto un peso a las variables de 0.1 y un peso a las pérdidas de 1. Se ha intentado ampliar la diferencia de pesos, pero provocaba que el algoritmo de corrección no convergiera.

Se ha calculado el valor del *SMC* tanto para el caso dinámico como para el estático para diferentes ecuaciones, en el mes de Mayo de 2013, siendo los resultados:

Resultados para las 2 primeras ecuaciones y todos los instantes de tiempo

Corrección dinámica=5.5325e+15

Corrección estática=8.2662e+15

Resultados para las 3 primeras ecuaciones y todos los instantes de tiempo

Corrección dinámica=5.6161e+15

Corrección estática= 1.0880e+16

Resultados para todas las ecuaciones y todos los instantes de tiempo

Corrección dinámica=4.1144e+16

Corrección estática=2.4311e+16

Al igual que en las dos pruebas anteriores, las modificaciones totales son superiores en la dinámica que en la estática. Sin embargo, atendiendo a las tres primeras ecuaciones, se observa que la corrección dinámica es mejor que la estática. Las modificaciones en esta prueba son inferiores a la anterior, lo que hace pensar que este es un mejor método, pero no tan eficaz como el correspondiente a la primera prueba.

Obtención de resultados para el problema elegido

A la luz de los resultados anteriores, se ha decidido obtener las gráficas y unos resultados más detallados para el problema de la *Prueba 1*, es decir, simplemente anulando la corrección en los términos de los depósitos y no forzar las pérdidas iniciales a cero, ni dar más peso a una variable que a otra modificando la matriz *W*.

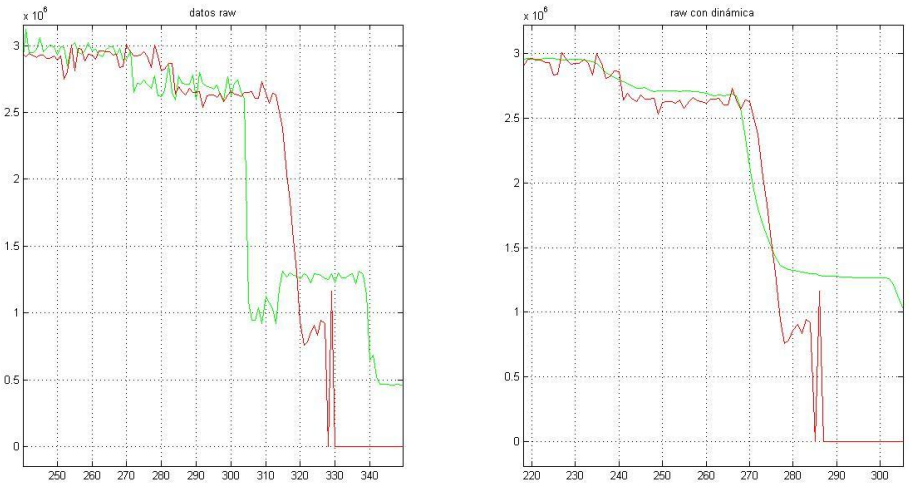
Se ha obtenido para cada ecuación el *SMC*. Los resultados de las ecuaciones corresponden a las medidas tomadas en el mes de Mayo de 2013, con un *dt* de 30 minutos.

Ecuación 1

Resultados del *SMC*

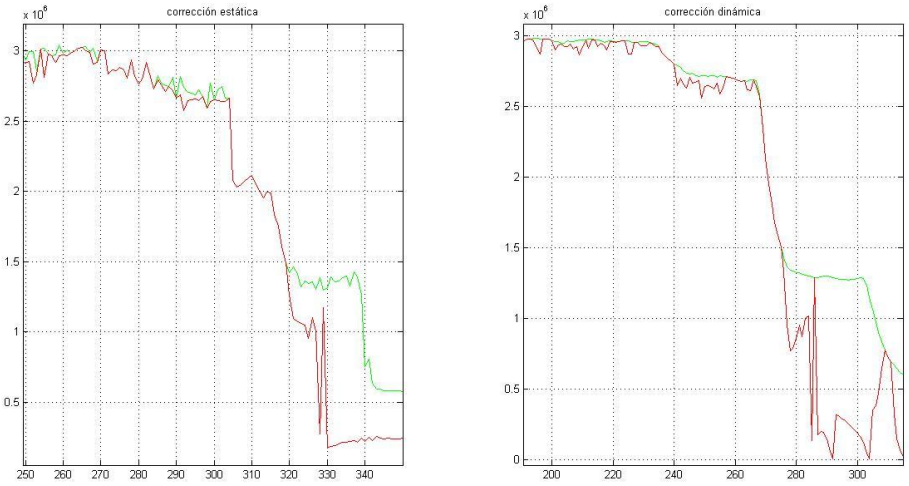
Corrección dinámica: 3.4379e+15

Corrección estática: 4.9870e+15



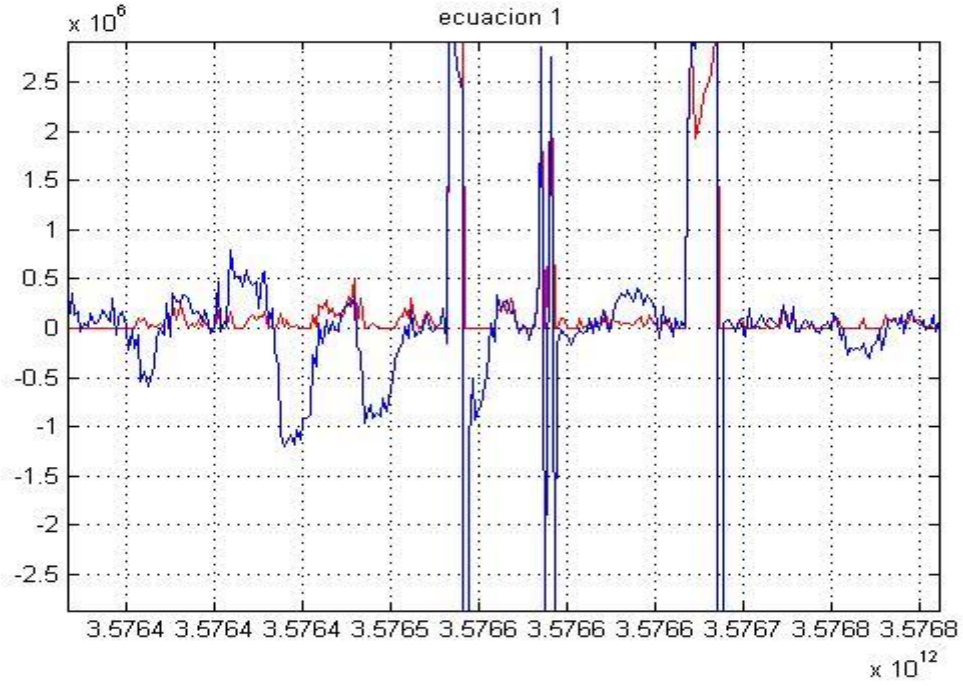
La imagen anterior muestra una representación de la ecuación 1. A la izquierda, usando los datos raw y de forma estática, y la derecha, usando los datos raw pero aplicando el modelo dinámico. En verde se representan los caudales de entrada y en rojo los de salida.

Como se puede observar en esta imagen, esta ecuación tiene un fuerte carácter dinámico. Se puede observar claramente un retraso en la salida (curva roja) respecto de la entrada (curva verde). Al aplicar el modelo, obtenemos un resultado mucho mejor, pero aún así, hay tramos donde las pérdidas son negativas (la salida está por encima de la entrada) por ello vamos a aplicar una corrección.



La imagen anterior muestra una representación de la ecuación 1. A la izquierda, usando los datos corregidos de forma estática, y la derecha, usando los datos corregidos pero aplicando el modelo dinámico. En verde se representan los caudales de entrada y en rojo los de salida.

En esta imagen se puede observar cómo se han corregido los datos respecto a la imagen anterior, de forma que ahora las pérdidas son positivas siempre. Observando los resultados del *SMC* se puede deducir que la corrección dinámica en esta ecuación es mejor que la estática, pues tiene un *SMC* menor. Este resultado es debido al fuerte carácter dinámico que tiene esta ecuación, con un retraso $L_1 = 200$ y un tiempo característico $\tau_1 = 180$ (minutos).



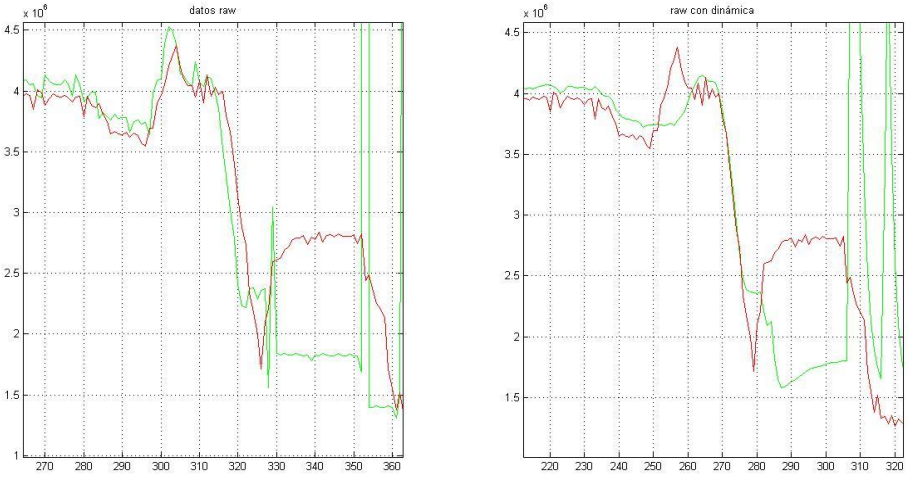
Esta imagen muestra en azul las pérdidas de la ecuación 1 antes de ser corregidas y en rojo las pérdidas de la ecuación 1 después de ser corregidas de forma dinámica. Nótese que después de la corrección las pérdidas siempre son mayores que cero.

Ecuación 2

Resultados del *SMC*

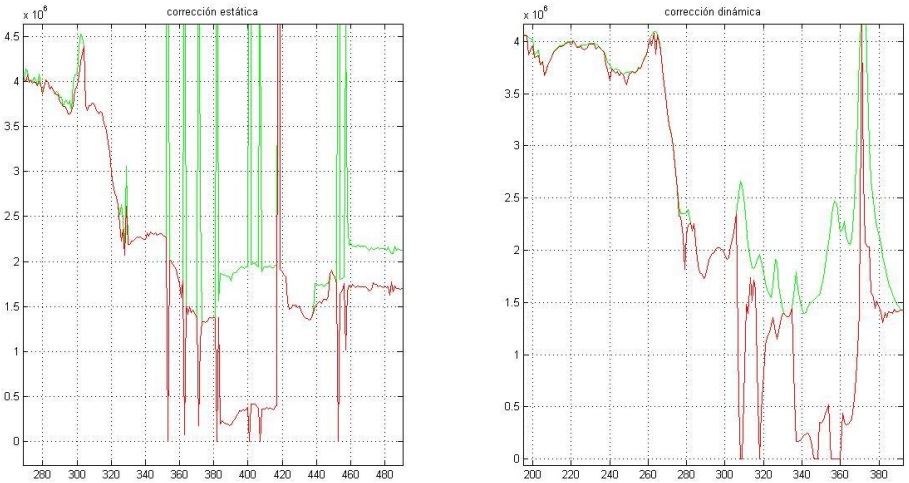
Corrección dinámica: 5.2696e+15

Corrección estática: 5.5353e+15



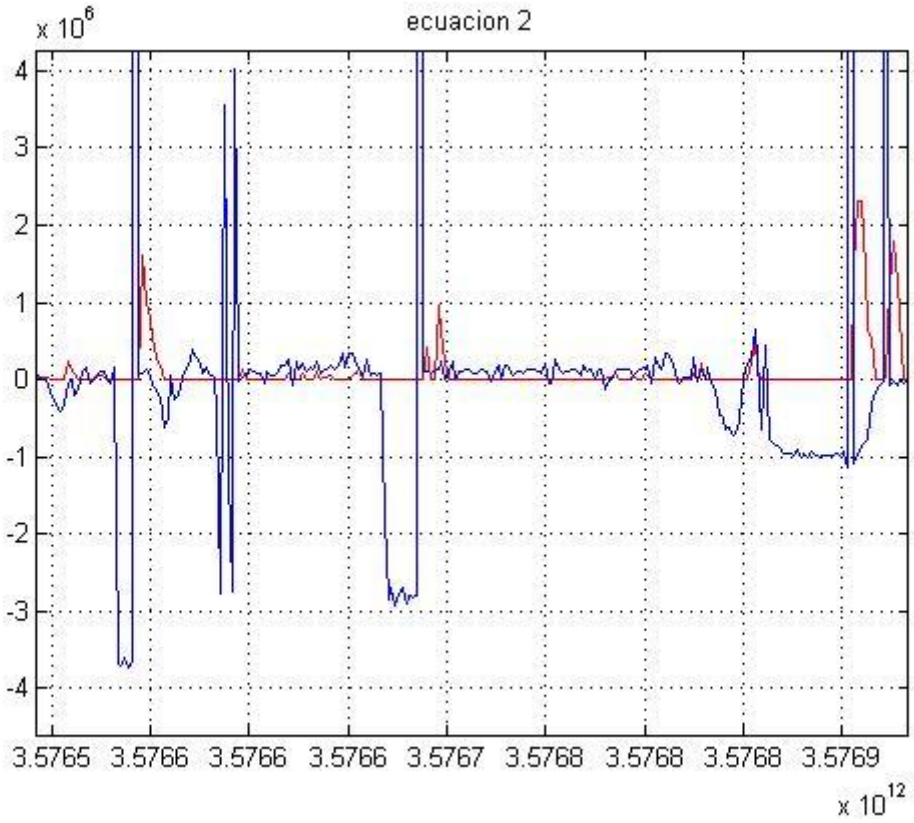
La imagen anterior muestra una representación de la ecuación 2. A la izquierda, usando los datos raw y de forma estática, y la derecha, usando los datos raw pero aplicando el modelo dinámico. En verde se representan los caudales de entrada y en rojo los de salida.

Como se puede observar en esta imagen, esta ecuación tiene un fuerte carácter dinámico. Se puede observar claramente un retraso en la salida (curva roja) respecto de la entrada (curva verde). Al aplicar el modelo, obtenemos un resultado mucho mejor, pero aún así, hay tramos donde las pérdidas son negativas (la salida está por encima de la entrada) por ello vamos a aplicar una corrección.



La imagen anterior muestra una representación de la ecuación 2. A la izquierda, usando los datos corregidos de forma estática, y la derecha, usando los datos corregidos pero aplicando el modelo dinámico. En verde se representan los caudales de entrada y en rojo los de salida.

En esta imagen se puede observar cómo se han corregido los datos respecto a la imagen anterior, de forma que ahora las pérdidas son positivas siempre. Observando los resultados del *SMC* se puede deducir que la corrección dinámica en esta ecuación es mejor que la estática, pues tiene un *SMC* menor. Este resultado es debido al fuerte carácter dinámico que tiene esta ecuación, con un retraso en los caudales de entrada de $L_2 = 50, L_4 = 180$ y unos tiempos característicos $\tau_2 = 50, \tau_4 = 200$ (minutos).



Esta imagen muestra en azul las pérdidas de la ecuación 2 antes de ser corregidas y en rojo las pérdidas de la ecuación 2 después de ser corregidas de forma dinámica. Nótese que después de la corrección las pérdidas siempre son mayores que cero.

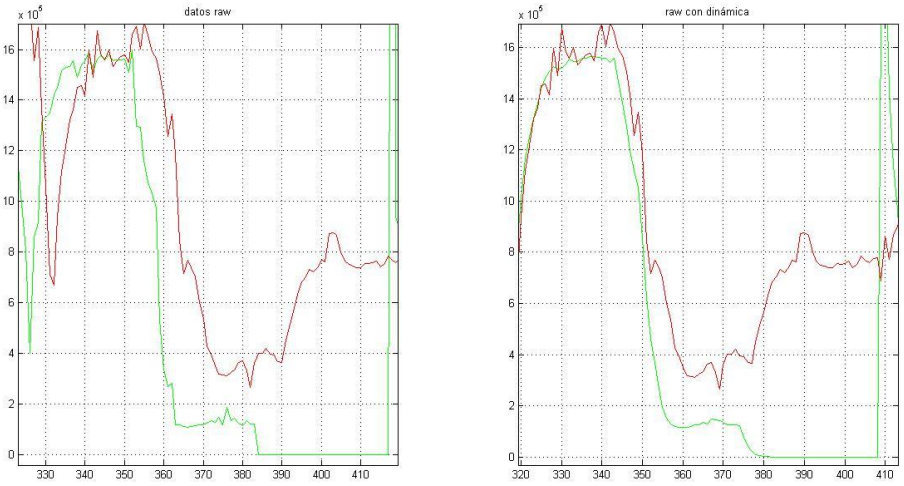
Ecuación 3

Resultados del *SMC*

Corrección dinámica: 2.1891e+14

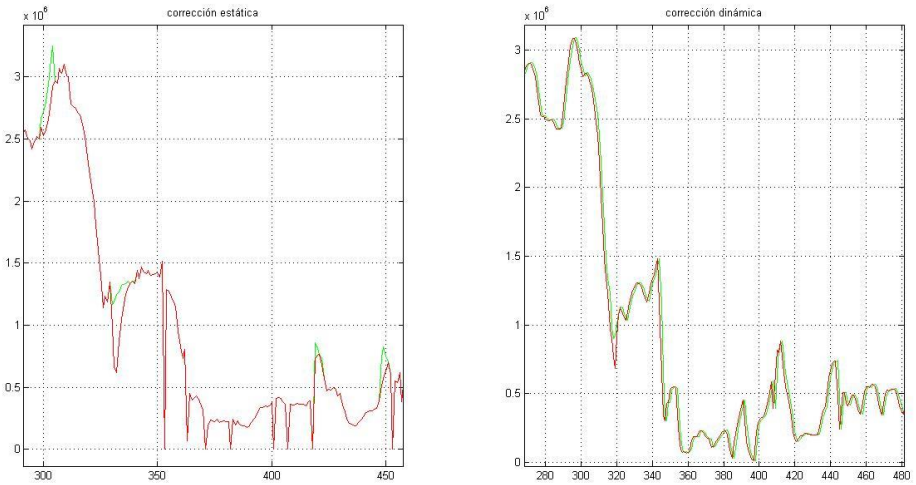
Corrección estática: 5.3050e+15

Esta es la ecuación donde se ve una mayor diferencia entre el *SMC* de la corrección dinámica y de la corrección estática.



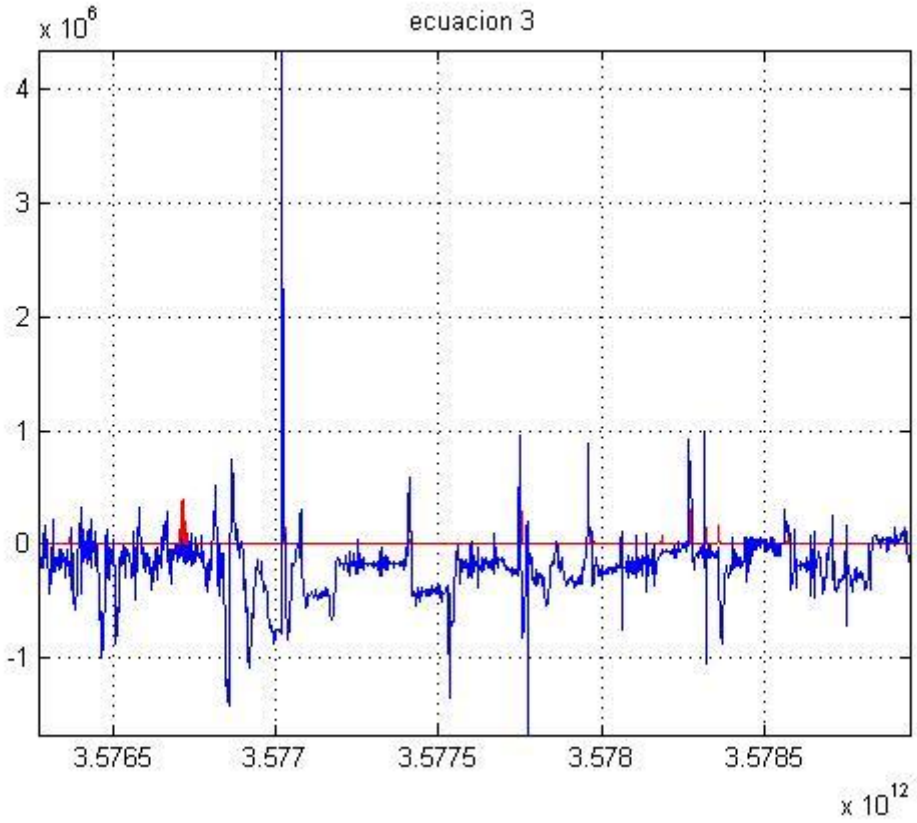
La imagen anterior muestra una representación de la ecuación 3. A la izquierda, usando los datos raw y de forma estática, y la derecha, usando los datos raw pero aplicando el modelo dinámico. En verde se representan los caudales de entrada y en rojo los de salida.

Como se puede observar en esta imagen, esta ecuación tiene un fuerte carácter dinámico. Se puede observar claramente un retraso en la salida (curva roja) respecto de la entrada (curva verde). Al aplicar el modelo, obtenemos un resultado mucho mejor, pero aún así, hay tramos donde las pérdidas son negativas (la salida está por encima de la entrada) por ello vamos a aplicar una corrección.



La imagen anterior muestra una representación de la ecuación 3. A la izquierda, usando los datos corregidos de forma estática, y la derecha, usando los datos corregidos pero aplicando el modelo dinámico. En verde se representan los caudales de entrada y en rojo los de salida.

En esta imagen se puede observar cómo se han corregido los datos respecto a la imagen anterior, de forma que ahora las pérdidas son positivas siempre. Observando los resultados del *SMC* se puede deducir que la corrección dinámica en esta ecuación es mejor que la estática, pues tiene un *SMC* menor. Este resultado es debido al fuerte carácter dinámico que tiene esta ecuación, con un retraso en el caudal de entrada de $L_5 = 100$ y un tiempo característico $\tau_5 = 45$ (minutos).



Esta imagen muestra en azul las pérdidas de la ecuación 3 antes de ser corregidas y en rojo las pérdidas de la ecuación 3 después de ser corregidas de forma dinámica. Nótese que después de la corrección las pérdidas siempre son mayores que cero.

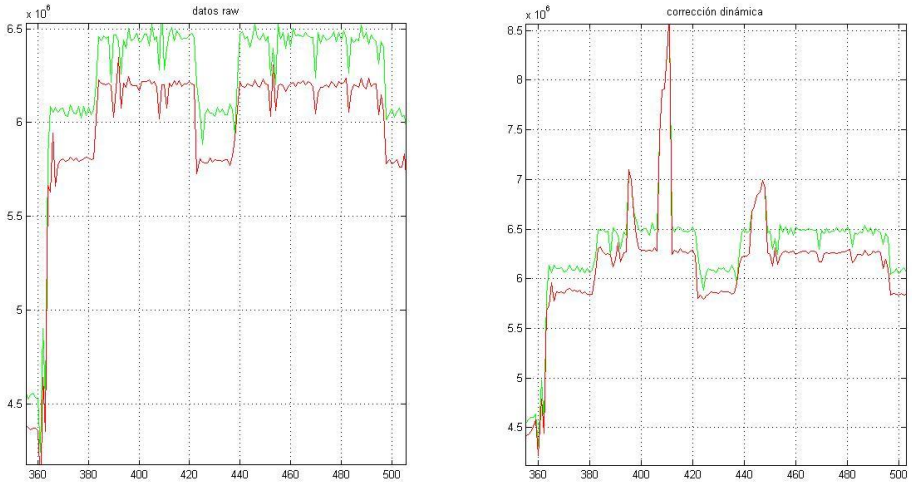
Ecuación 4

Resultados del *SMC*

Corrección dinámica: 2.1948e+16

Corrección estática: 8.3648e+15

A partir de esta ecuación aparece un cambio de tendencia en el que el *SMC* de las correcciones estáticas es inferior al de las dinámicas.

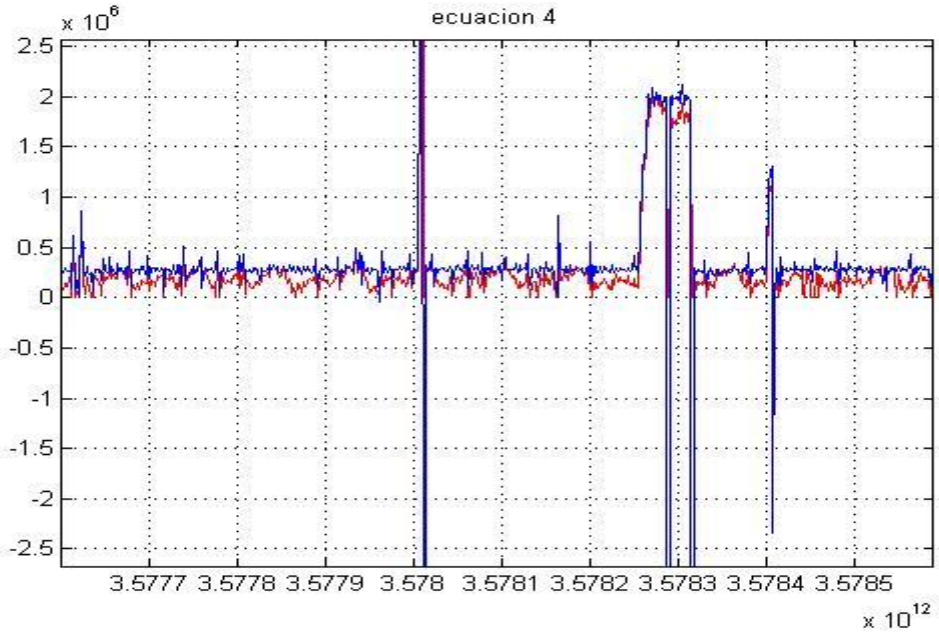


La imagen anterior muestra dos representaciones de la ecuación 4, siendo las gráficas rojas las representaciones de las salidas del nodo y las verdes las entradas al nodo. La gráfica de la izquierda es una representación de la ecuación 4 utilizando los datos raw de forma estática y la de la derecha es una representación de la ecuación 4 utilizando los datos corregidos del modelo dinámico y aplicando dicho modelo.

De la imagen de la izquierda se pueden observar varias cosas. La primera es que en esta ecuación la dinámica no está presente, no se observa apenas retraso entre las dos gráficas y son prácticamente igual de rápidas, por ello se identificó esta ecuación con $L_7 = 0\,, \tau_7 = 0$. La segunda cosa que podemos observar es un ejemplo de error de medida de los sensores de este caudal. Inexplicablemente hay una diferencia entre la entrada y la salida, la curva de la salida es igual a la de la entrada pero con menos cantidad de agua. Una explicación podría ser que estén sacando agua del caudal en un punto intermedio entre el sensor de entrada y el de salida.

La corrección parece rebajar la diferencia entre las dos curvas y con ello el supuesto error de medida que comete el sensor.

En esta ecuación el *SMC* es menor en el caso de corrección estático que en el dinámico, esto parece ser debido a que esta ecuación no tiene apenas carácter dinámico y la corrección estática parece ser mejor que la del modelo dinámico.



Esta imagen muestra en azul las pérdidas de la ecuación 4 antes de ser corregidas y en rojo las pérdidas de la ecuación 4 después de ser corregidas de forma dinámica. Nótese que después de la corrección las pérdidas siempre son mayores que cero.

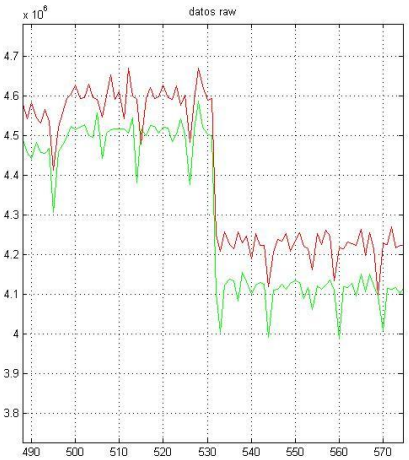
<u>Ecuación 5</u>

Resultados del *SMC*

Corrección dinámica: 2.2343e+16

Corrección estática: 6.7006e+15

De nuevo, el *SMC* para el caso dinámico es mayor que para el estático. De hecho, esta es la ecuación que tiene una mayor diferencia entre los dos.

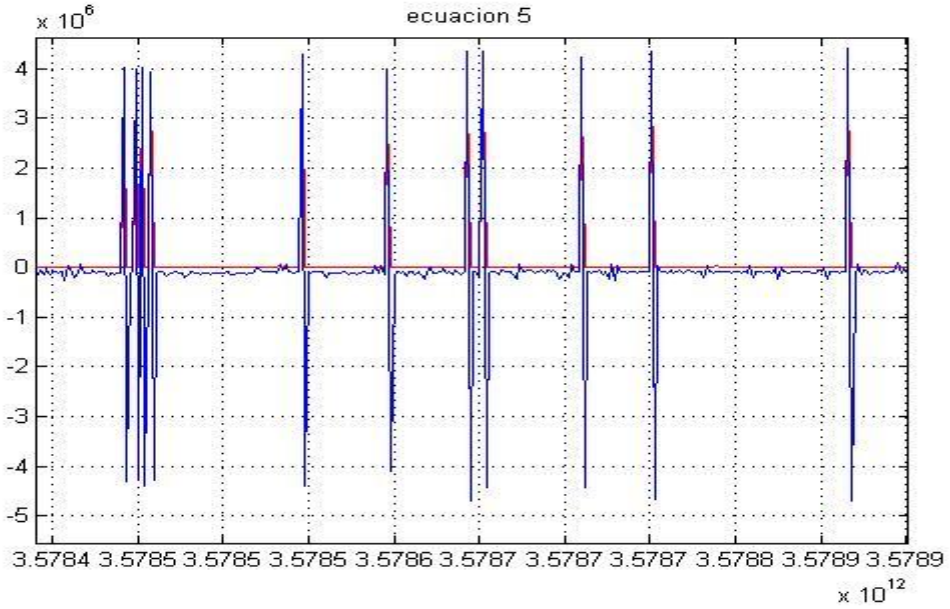


La imagen anterior muestra dos representaciones de la ecuación 5, siendo las gráficas rojas las representaciones de las salidas del nodo y las verdes las entradas al nodo. La gráfica de la izquierda es una representación de la ecuación 5 utilizando los datos raw de forma estática y la de la derecha es una representación de la ecuación 5 utilizando los datos corregidos del modelo dinámico y aplicando dicho modelo.

De la imagen de la izquierda se pueden observar varias cosas. La primera es que en esta ecuación la dinámica no está presente, no se observa apenas retraso entre las dos gráficas y son prácticamente igual de rápidas, por ello se identificó esta ecuación con $L_8 = 0\,, \tau_8 = 0$. La segunda cosa que podemos observar es un ejemplo de error de medida de los sensores de este caudal. Inexplicablemente hay una diferencia entre la entrada y la salida, la curva de la salida es igual a la de la entrada pero con una mayor cantidad de agua. En este caso no podríamos decir que están sacando agua puesto que parece ser que el sensor de la salida ha medido una mayor cantidad de litros que el sensor de la entrada, por ello, podríamos pensar que están echando agua en el caudal, pero eso no tiene sentido, luego debe haber un error de medida en alguno de los dos sensores.

En este caso la corrección es aún mayor que antes puesto que ya no solo trata de reducir la diferencia entre las dos curvas, al estar la curva roja por encima de la verde en todo momento las pérdidas son negativas, el algoritmo corregirá hasta que las perdidas sean positivas en todos los puntos.

En esta ecuación el *SMC* es bastante inferior en el caso de corrección estático que en el dinámico, esto parece ser debido a que esta ecuación no tiene apenas carácter dinámico y la corrección estática parece ser mejor que la del modelo dinámico.



Esta imagen muestra en azul las pérdidas de la ecuación 5 antes de ser corregidas y en rojo las pérdidas de la ecuación 5 después de ser corregidas de forma dinámica. Nótese que después de la corrección las pérdidas siempre son mayores que cero.

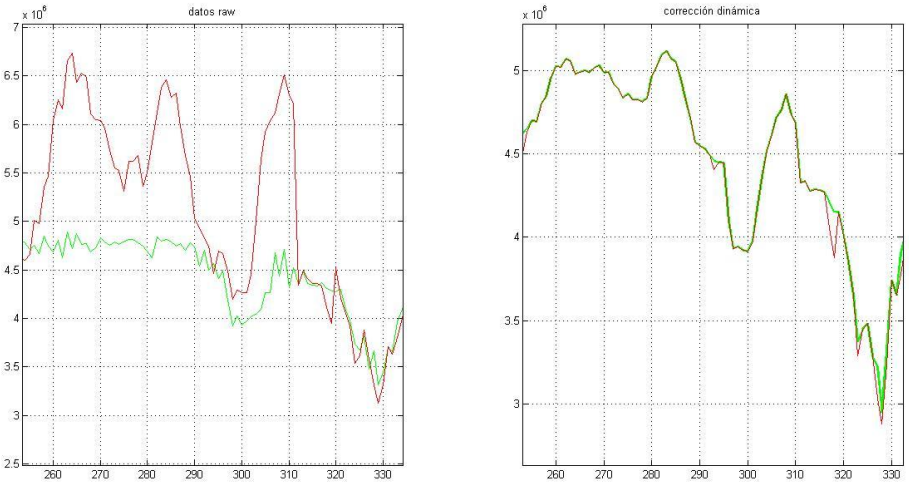
Ecuación 6

Resultados del *SMC*

Corrección dinámica: 1.1491e+16

Corrección estática: 6.2559e+15

Al igual que en las dos ecuaciones anteriores, la corrección estática parece ser mejor que la dinámica.

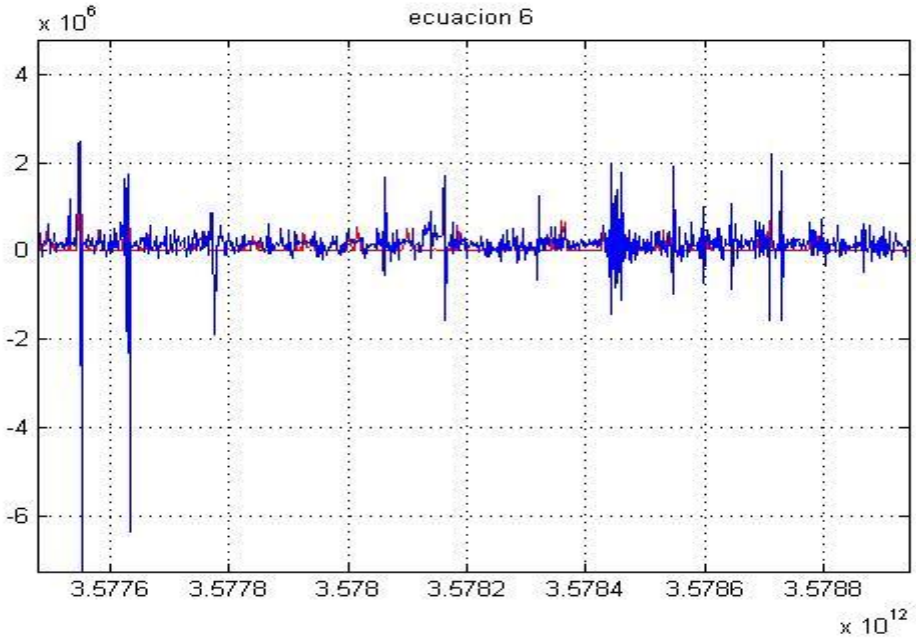


La imagen anterior muestra dos representaciones de la ecuación 6, siendo las gráficas rojas las representaciones de las salidas del nodo y las verdes las entradas al nodo. La gráfica de la izquierda es una representación de la ecuación 6 utilizando los datos raw de forma estática y la de la derecha es una representación de la ecuación 6 utilizando los datos corregidos del modelo dinámico y aplicando dicho modelo.

De la imagen de la izquierda se pueden observar varias cosas. La primera es que en esta ecuación la dinámica no está presente, no se observa apenas retraso entre las dos gráficas y son prácticamente igual de rápidas, por ello se identificó esta ecuación con $L_6 = L_9 = 0$, $\tau_6 = \tau_9 = 0$. Esta ecuación es la correspondiente a los depósitos, observando la imagen de la izquierda se puede observar, al igual que en ecuaciones anteriores, un error de medida de los sensores, ya sean los de nivel de los depósito o los de medida de caudal. De nuevo, tampoco parece indicar que sea por extracción de agua en el caudal porque la curva roja está por encima de la verde.

Es por ello que de nuevo va a haber un alto *SMC*, el algoritmo debe acercar las dos gráficas y además hacer que la verde esté por encima que la roja.

De nuevo, en esta ecuación el *SMC* es bastante inferior en el caso de corrección estático que en el dinámico, esto parece ser debido a que esta ecuación no tiene apenas carácter dinámico y la corrección estática parece ser mejor que la del modelo dinámico.



Esta imagen muestra en azul las pérdidas de la ecuación 6 antes de ser corregidas y en rojo las pérdidas de la ecuación 6 después de ser corregidas de forma dinámica. Nótese que después de la corrección las pérdidas siempre son mayores que cero.

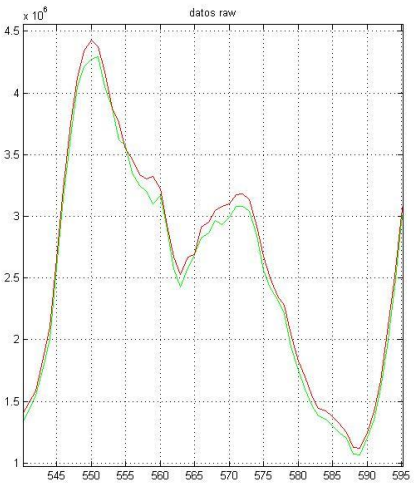
Ecuación 7

Resultados del *SMC*

Corrección dinámica: 2.3644e+15

Corrección estática: 2.0868e+15

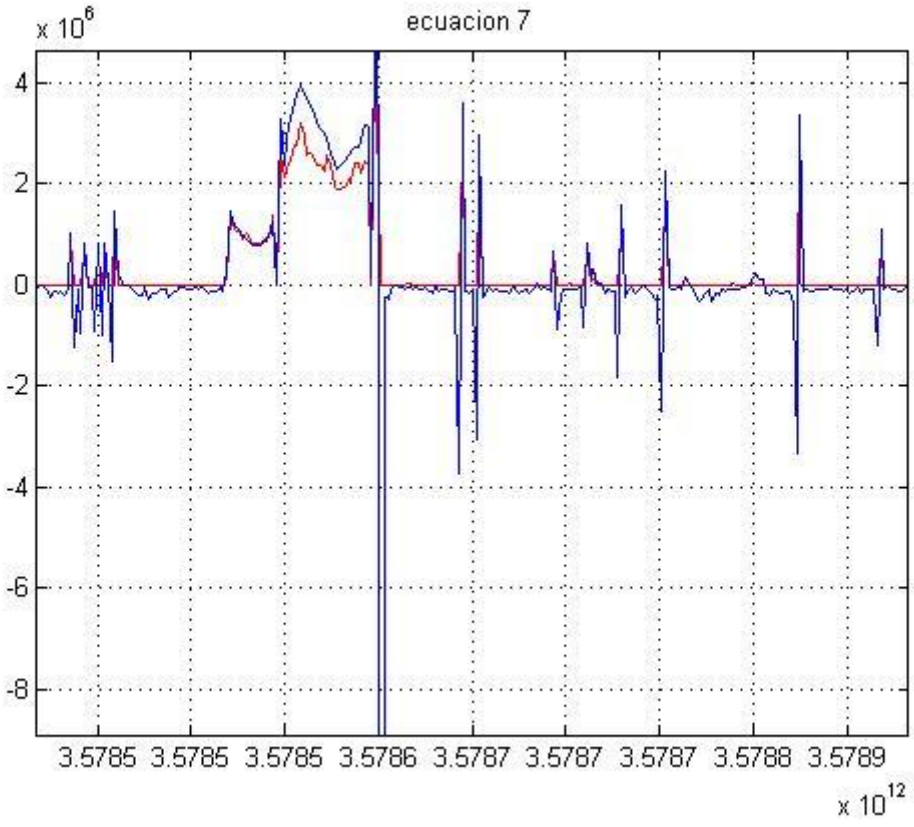
Al igual que en las tres ecuaciones anteriores, la corrección estática parece ser mejor que la dinámica, pero aquí parece haberse reducido bastante la diferencia.



La imagen anterior muestra dos representaciones de la ecuación 7, siendo las gráficas rojas las representaciones de las salidas del nodo y las verdes las entradas al nodo. La gráfica de la izquierda es una representación de la ecuación 7 utilizando los datos raw de forma estática y la de la derecha es una representación de la ecuación 7 utilizando los datos corregidos del modelo dinámico y aplicando dicho modelo.

De la imagen de la izquierda se pueden observar varias cosas. La primera es que en esta ecuación la dinámica no está presente, no se observa apenas retraso entre las dos gráficas y son prácticamente igual de rápidas, por ello se identificó esta ecuación con $L_{10} = L_{11} = L_{12} = 0$, $\tau_{10} = \tau_{11} = \tau_{12} = 0$. Aunque parece haber un error de medida en los sensores (gráfica roja por encima de la verde), este no parece ser tan severo como los otros, de ahí que los *SMC* en esta ecuación sean menos acusados.

De nuevo, en esta ecuación el *SMC* es inferior en el caso de corrección estático que en el dinámico, aunque en esta ecuación la diferencia no es tan grande como en anteriores. Esto parece ser debido a que esta ecuación no tiene apenas carácter dinámico y la corrección estática parece ser mejor que la del modelo dinámico.



Esta imagen muestra en azul las pérdidas de la ecuación 7 antes de ser corregidas y en rojo las pérdidas de la ecuación 7 después de ser corregidas de forma dinámica. Nótese que después de la corrección las pérdidas siempre son mayores que cero.

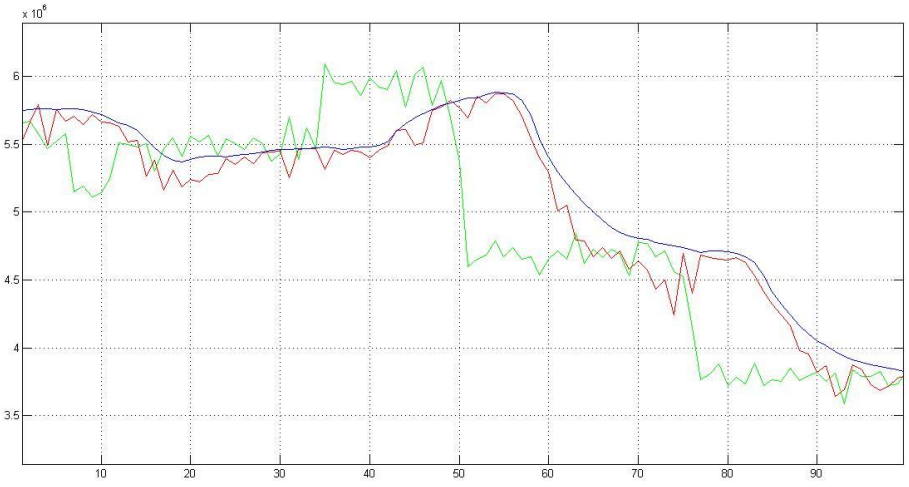
Conclusiones

En este apartado se va a hacer una vista general de los resultados obtenidos de aplicar el modelo dinámico a nuestra red, y que conclusiones se pueden sacar a partir de ellos.

Conclusiones sobre los resultados de la identificación del modelo dinámico de la red

El objetivo principal de este proyecto era obtener unos mejores resultados de la corrección con el modelo dinámico que con la corrección estática.

Para poder cumplir ese objetivo, primero teníamos que plantear un modelo que fuera válido y representara correctamente nuestra red. Se ha conseguido validar un modelo no autoregresivo basado en la respuesta impulsional de un sistema de primer orden retrasado con ganancia estática unidad, muestreada y truncada a *m* términos. Este modelo, como se puede observar en la gráfica siguiente, ha cumplido satisfactoriamente con las expectativas, sobretodo, en aquellas ecuaciones donde hay una dinámica más acusada (las tres primeras ecuaciones).



La imagen anterior es una representación de la ecuación 1. La curva verde corresponde a la entrada del nodo, la roja a la salida del nodo, y la azul es el modelo de la entrada en esta ecuación.

La ecuación primera, como se puede observar en la gráfica, tiene una dinámica bastante acusada, un retraso considerable entre la entrada (verde) y la salida (roja). Por medio del modelo dinámico de la entrada hemos conseguido reducir la diferencia entre la entrada modelada (azul) y la salida sin modelar (roja). Además, sin el modelo se pueden apreciar bastantes errores de medida, puesto que las pérdidas son negativas en algunos trozos (gráfica roja por encima de la verde). Al aplicar el modelo, dichos errores de medida se ven reducidos, incluso algunas veces eliminados, como en la imagen anterior.

Los resultados de la identificación han sido:

- Ecuación 1:
  - L1=200
  - tau1=180
- Ecuación 2:
  - L2=50 L4=180
  - tau2=50 tau4=200
- Ecuación 3:
  - L5=100
  - tau5=45
- Ecuación 4:
  - L7=0
  - tau7=0
- Ecuación 5
  - L8=0
  - tau8=0
- Ecuación 6
  - L6=0 L9=0
  - tau6=0 tau9=0
- Ecuación 7
  - L10=0 L11=0 L12=0
  - tau10=0 tau11=0 tau12=0

Parece ser, que las únicas ecuaciones que se ven realmente afectadas por la dinámica son las tres primeras, debido a que los caudales que intervienen en dichos nodos tienen un recorrido bastante mayor, lo que provoca que se vean muy influenciados por la dinámica de los mismos.

Conclusiones sobre los resultados de las correcciones

Una vez obtenido el modelo, podemos aplicar una corrección de forma que tenga en cuenta la dinámica de cada ecuación, para así tratar de obtener un resultado mejor que de forma estática.

En un principio, se pensaba que el modelo mejoraría la corrección en todos los aspectos, sobre todo en aquellas ecuaciones que tuvieran una dinámica más acusada (las tres primeras ecuaciones). Sin embargo, a la vista de los resultados se ha observado que esto no es así, la corrección dinámica solo ha mejorado las correcciones en aquellas ecuaciones que tienen una dinámica más acusada.

En apartados anteriores se ha corroborado que las correcciones se han realizado correctamente, las pérdidas tanto en la corrección dinámica como en la estática eran mayores que cero, y se cumplía que $A*z=b$.

El indicador que vamos a utilizar para valorar las diferencias entre la corrección estática y dinámica es el *Sumatorio de las Modificaciones al cuadrado.*

Este ha sido el resultado del *SMC* para cada ecuación:

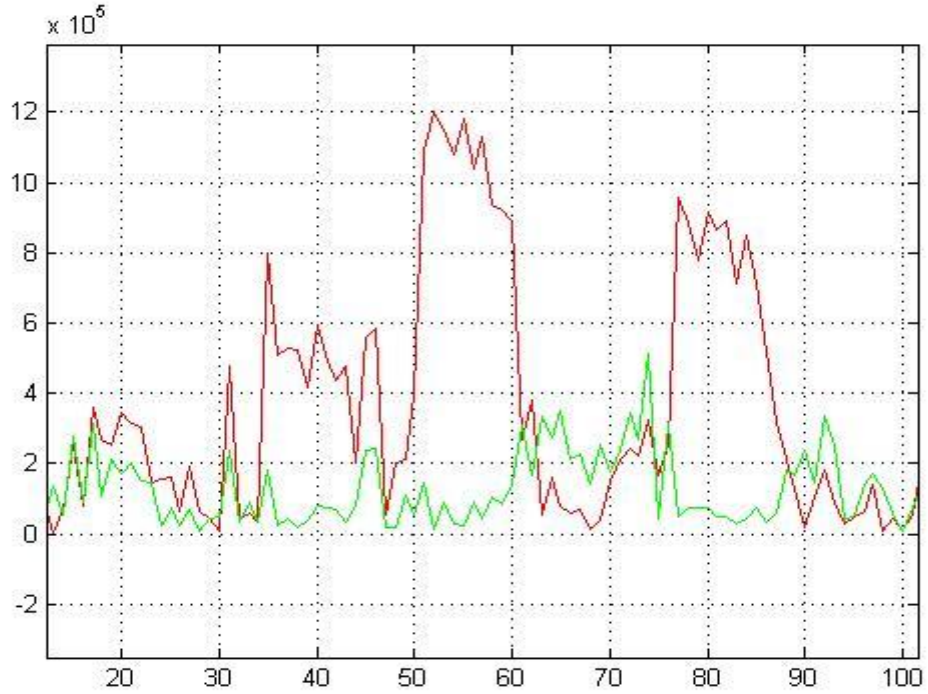
	Corrección dinámica (L^2)	**Corrección estática (L^2)**
Ecuación 1	$0.3438 \cdot 10^{16}$	$0.4987 \cdot 10^{16}$
Ecuación 2	$0.5270 \cdot 10^{16}$	$0.5535 \cdot 10^{16}$
Ecuación 3	$0.0219 \cdot 10^{16}$	$0.5305 \cdot 10^{16}$
Ecuación 4	$2.1948 \cdot 10^{16}$	$0.8365 \cdot 10^{16}$
Ecuación 5	$2.2343 \cdot 10^{16}$	$0.6701 \cdot 10^{16}$
Ecuación 6	$1.1491 \cdot 10^{16}$	$0.6256 \cdot 10^{16}$
Ecuación 7	$0.2364 \cdot 10^{16}$	$0.2087 \cdot 10^{16}$

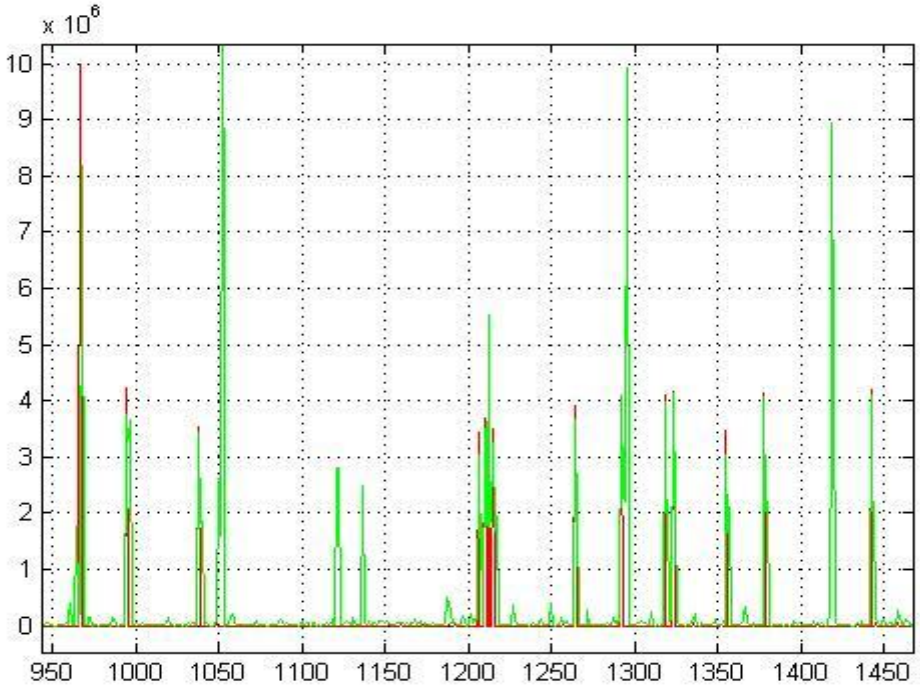
A la luz de estos resultados se confirma nuestra hipótesis inicial de que en las ecuaciones que tuvieran una mayor dinámica (las tres primeras) los resultados de nuestro modelo serían mejores que los anteriores sin modelo. Sin embargo, en el resto de ecuaciones la corrección estática resulta ser bastante mejor que la dinámica.

La diferencia en las ecuaciones que no tienen dinámica es tan grande que si sumamos los resultados de todas las ecuaciones, el *SMC* de la corrección dinámica resulta ser bastante mayor que el de la corrección estática.

Estos resultados nos hacen presuponer que en una red donde todos los caudales tengan una acusada dinámica el modelo de corrección dinámica tendría un resultado bastante mejor que el de la estática, pero dado que en nuestra red solo las tres primeras ecuaciones tienen una dinámica notable, los resultados no son tan buenos como se esperaba.

La diferencia entre las pérdidas de una ecuación con dinámica y una sin ella se puede observar en las siguientes imágenes





La primera imagen corresponde a la representación de las pérdidas corregidas de forma estática (roja) y las de la dinámica (verde) de la ecuación 1. La segunda imagen es la misma representación pero para la ecuación 5.

Viendo las imágenes anteriores se puede observar la diferencia mencionada anteriormente. En la primera imagen, correspondiente a la primera ecuación, se puede observar como las pérdidas corregidas de forma estática (diferencia entre la salida y la entrada) son mayores a las del modelo dinámico (diferencia entre la salida y la entrada modelada). Sin embargo, en la segunda imagen, correspondiente a la quinta ecuación, es decir, una ecuación sin dinámica, el resultado es totalmente el contrario, las pérdidas corregidas del modelo dinámico resultan ser bastante superiores a las de la corrección estática.

Reseña sobre el contenido del CD

En el CD que acompaña a esta memoria se incluye una copia de esta memoria en formato PDF, junto con los códigos de las funciones más importantes desarrolladas en este proyecto en formato .txt. El archivo "CWNetworkmodelDinamica" contiene la función desarrollada en lenguaje *C++* que define el problema de corrección dinámica. El archivo "identificacion" es una función en *Matlab* que crea una gráfica en tres dimensiones para poder sacar la *tau* y la *L* del sistema que da un menor error. El archivo "iden_Ltau" es la función en *Matlab* de la que se vale la función anterior para calcular los errores de identificación para cada par *L, tau.*