UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Sistemas Informáticos y Computación

Reducción del Tiempo de Simulación de Redes de Distribución de Agua, mediante el Método de Mallas y la Computación de Altas Prestaciones

Tesis Doctoral

Autor:
Fernando Alvarruiz Bermejo

Directores:
Fernando Martínez Alzamora
Antonio Manuel Vidal Maciá

Enero 2016

Agradecimientos

Quiero agradecer aquí a mis directores de tesis, Antonio Vidal y Fernando Martínez, la ayuda que me han prestado. Su orientación, sus consejos y comentarios han ayudado en buena medida a hacer que esta tesis sea mejor.

David, José Miguel y Pedro merecen una mención, por haber colaborado conmigo en el proyecto en el que nació esta tesis. Los tres son excelentes compañeros. Además, David me ha ayudado a menudo de forma generosa, sin necesidad siquiera de pedírselo. También agradezco a Salva su disposición a ayudar y compartir, y su torrente inagotable de chistes y juegos de palabras.

Doy las gracias a Vicente Hernández, antiguo líder del grupo GRyCAP, por contar conmigo para ser miembro del grupo.

También quiero darle las gracias a Carmen, porque me ha ayudado a organizarme, y siempre me ha escuchado y dado buenos consejos.

Y por supuesto a ti, Cristina, por apoyarme, ser comprensiva y aguantarme.

Resumen

La simulación por computador de las redes de distribución de agua potable, mediante el uso de modelos matemáticos, es hoy en día una herramienta indispensable para el diseño y la explotación de dichas redes. La simulación se utiliza tanto en el diseño de nuevos abastecimientos y en ampliaciones o modificaciones de abastecimientos existentes, como en las tareas de operación normales de cualquier red. Se puede diferenciar entre dos tipos de simulación: la simulación hidráulica, que permite obtener las presiones y caudales que se registran en la red, y la simulación de la calidad del agua, cuyo objetivo es obtener información sobre concentraciones de sustancias químicas.

A menudo la necesidad de simulación surge dentro de un problema más amplio de optimización o de análisis de fiabilidad, que requiere llevar a cabo un gran número de simulaciones, con lo que el proceso completo resulta de una complejidad computacional considerable. Esto, añadido al hecho de que el tamaño y nivel de detalle de los modelos de redes crece constantemente, como consecuencia de la incorporación automática de datos contenidos en Sistemas de Información Geográfica, hace que las prestaciones del *solver* de simulación tengan un gran impacto en el tiempo total de cálculo necesario.

En este contexto, esta tesis considera y explora distintas vías para mejorar las prestaciones de la simulación de redes de distribución de agua. La primera de estas vías consiste en realizar algunas aportaciones al método de simulación hidráulica conocido como *método de Newton-Raphson de mallas* (o simplemente *método de mallas*), el cual se basa en la consideración de caudales correctores asociados a un conjunto de mallas independientes definidas sobre la red. Aunque el método conocido como *Algoritmo del Gradiente Global* (GGA) goza de mayor aceptación, el método de mallas tiene el potencial de ser más rápido, debido al menor tamaño de los sistemas lineales subyacentes. En esta tesis se presentan aportaciones para mejorar las prestaciones del método de mallas de simulación hidráulica. En concreto, en primer lugar, se desarrollan algoritmos eficientes para la selección de un conjunto de mallas adecuado, que conduzca a un sistema altamente disperso. En segundo lugar se desarrollan métodos para la modelización eficiente de válvulas, y especialmente válvulas reductoras/sostenedoras de presión.

La segunda vía explorada es la introducción de la *computación de altas prestaciones* en la simulación hidráulica usando plataformas de memoria distribuida. En particular, se parte del código de Epanet, un software de simulación de redes de am-

plia aceptación, y se introducen en él algoritmos paralelos de simulación, usando la herramienta *Message Passing Interface* (MPI) para la comunicación entre procesos. Como resultado de este trabajo, se presenta en primer lugar un algoritmo paralelo para la simulación de caudales y presiones por medio del método GGA, haciendo uso de algoritmos multifrontales para la resolución paralela de los sistemas lineales subyacentes. En segundo lugar, se describe un algoritmo paralelo para la simulación de la calidad del agua mediante el *Método de Elementos Discretos de Volumen* (DVEM), particionando la red por medio de algoritmos de bisección recursiva multinivel. En tercer lugar, se presenta un método paralelo para resolver un problema de minimización de fugas mediante la determinación de las consignas óptimas de una serie de válvulas reductoras de presión.

En las plataformas de memoria distribuida, la sobrecarga de comunicación y sincronización puede ser excesiva, contrarrestando a veces la ganancia derivada de la división del cómputo entre los procesadores. Este efecto es menos acusado en las plataformas de memoria compartida como los sistemas *multicore*, que han ganado popularidad en los últimos años. Este hecho motiva la tercera de las vías exploradas en la tesis, que es el desarrollo de algoritmos paralelos para la simulación de presiones y caudales en sistemas *multicore*. Se utiliza la herramienta OpenMP para la paralelización, tanto del software Epanet y de su implementación del método GGA, como del método de mallas, con las aportaciones al mismo que se han realizado en el contexto de esta tesis.

Abstract

Computer simulation of water distribution networks by means of mathematical models is nowadays an indispensable tool for the design and exploitation of those networks. Simulation is used not only for the design of new supply systems, or modifications and extensions of existing systems, but also for the normal operation tasks carried out in any network. Two main types of simulation can be differentiated: hydraulic simulation, by means of which the pressures and flows registered in the network are computed, and water quality simulation, the objective of which is to obtain information about chemical substance concentrations.

The need for simulation comes often in the context of a wider problem of optimization or reliability analysis, which requires performing a large number of simulations, thus resulting in a process with considerable computational complexity. This fact, added to the growing size and level of detail of network models, as a consequence of the automatic incorporation of data coming from Geographical Information Systems, means that the performance of the simulation solver has a great impact in the overall computing time.

In this context, this thesis considers and explores different strategies to improve the performance of water distribution network simulation. The first strategy consists of making some contributions to the hydraulic simulation method known as *Looped Newton-Raphson* (or more simply the *loop method*), which is based on the consideration of flow corrections associated to a set of independent loops within the network. Even though the method known as *Global Gradient Algorithm* (GGA) is more widely used and accepted, the loop method has the potential to be faster, owing to the smaller size of the underlying linear systems. In this thesis some contributions are presented to improve the performance of the loop method for hydraulic simulation. Firstly, efficient algorithms are developed for the selection of a suitable set of independent loops, leading to a highly sparse linear system. Secondly, methods are developed for efficient modeling of hydraulic valves, and especially pressure reducing/sustaining valves.

The second strategy explored is the introduction of *high performance computing* in the hydraulic simulation using distributed memory platforms. In particular, the code of Epanet, a widely accepted water distribution network simulation software, is taken as the starting point for the introduction of parallel simulation algorithms, using the *Message Passing Interface* (MPI) tool for inter-process communications. As a result of this work, firstly a parallel algorithm is presented for the simulation

of flows and pressures by means of the GGA method, making use of multifrontal algorithms for the parallel solution of the underlying linear systems. Secondly, a parallel algorithm for water quality simulation by means of the *Discrete Volume Element Method* (DVEM) is described, based on partitioning the network by means of multilevel recursive bisection algorithms. Thirdly, a parallel method is presented for leakage minimization by finding the optimal pressure settings for a set of pressure-reducing valves.

In distributed memory platforms the overhead due to communication and synchronization can be excessively high, counterbalancing the gain derived from the division of the computation among the processors. This effect is less pronounced in shared memory platforms such as multicore systems, which have gained popularity over the last years. This fact motivates the third strategy explored in this thesis, which is the development of parallel algorithms for simulation of flows and pressures using multicore systems. OpenMP is the tool used for the parallelization, both of the method GGA as implemented in Epanet software and of the loop method with the contributions on it that have been made in the context of this thesis.

Resum

La simulació per computador de les xarxes de distribució d'aigua potable, per mitjà de l'ús de models matemàtics, es hui en dia una ferramenta indispensable per al disseny i l'explotació d'abastiments d'aigua. La simulació s'utilitza tant per al disseny de nous abastiments o ampliacions i modificacions d'abastiments existents, com per a les tasques d'operació normals en qualsevol xarxa. Es pot diferenciar entre dos tipus de simulació: la simulació hidràulica, que permet obtindre les pressions i cabals que es produeixen en la xarxa, i la simulació de la qualitat de l'aigua, l'objectiu de la qual és obtindre informació sobre concentracions de substàncies químiques.

Sovint la necessitat de simulació sorgeix dins d'un problema més ampli d'optimització o d'anàlisi de fiabilitat, que requereix dur a terme un gran nombre de simulacions, amb la qual cosa el procés complet resulta d'una complexitat computacional considerable. Això, afegit al fet de que la grandària i nivell de detall del models de xarxes creix constantment, com a conseqüència de la incorporació automàtica de dades contingudes en Sistemes d'Informació Geogràfica, fa que les prestacions del *solver* de simulació tinguen un gran impacte en el temps total de càlcul necessari.

En este context, esta tesi considera i explora diferents vies per a millorar les prestacions de la simulació de xarxes de distribució d'aigua. La primera d'estes vies consisteix en realitzar algunes contribucions al mètode de simulació hidràulica conegut com *mètode de Newton-Raphson de malles* (o simplement *mètode de malles*), el qual es basa en la consideració de cabals correctors associats a un conjunt de malles independents definides en la xarxa. Encara que el mètode conegut com *Algorisme del Gradient Global* (GGA) gaudeix de major acceptació, el mètode de malles té el potencial de ser més ràpid, degut a la menor grandària dels sistemes lineals subjacents. En esta tesi es presenten contribucions per a millorar les prestacions del mètode de malles de simulació hidràulica. En concret, en primer lloc es desenvolupen algorismes eficients per a la selecció d'un conjunt de malles adequat, que conduïsca a un sistema lineal altament dispers. En segon lloc es desenvolupen mètodes per a la modelització eficient de vàlvules, i especialment vàlvules reductores/sostenidores de pressió.

La segona via explorada és la introducció de la *computació d'altes prestacions* en la simulació hidràulica utilitzant plataformes de memòria distribuïda. En concret, es parteix del codi d'Epanet, un programari de simulació de xarxes de distribució d'aigua d'amplia acceptació, i s'hi introdueixen algorismes paral·lels de simulació, utilitzant la ferramenta *Message Passing Interface* (MPI) per a la comunicació en-

tre processos. Com a resultat d'este treball, es presenta en primer lloc un algorisme paral·lel per a la simulació de cabals i pressions per mitjà del mètode GGA, fent ús d'algorismes multifrontals per a la resolució en paral·lel dels sistemes lineals subjacents. En segon lloc, es descriu un algorisme paral·lel per a la simulació de la qualitat d'aigua amb el *Mètode d'Elements Discrets de Volum* (DVEM), particionant la xarxa per mitjà d'algoritmes de bisecció recursiva multinivell. En tercer lloc es presenta un mètode paral·lel per a resoldre un problema de minimització de fugues mitjançant la determinació de les consignes òptimes d'una sèrie de vàlvules reductores de pressió.

En les plataformes de memòria distribuïda, la sobrecàrrega de comunicació i sincronització pot ser excessiva, contrarestant a vegades el guany derivat de la divisió del còmput entre els processadors. Este efecte és menys acusat en les plataformes de memòria compartida, com els sistemes *multicore*, els quals han guanyat popularitat en els últims anys. Este fet motiva la tercera de les vies explorades en esta tesi, que és el desenvolupament d'algorismes paral·lels per a la simulació de pressions i cabals en sistemes *multicore*. S'utilitza la ferramenta OpenMP per a la paral·lelització, tant del programari Epanet i de la seua implementació del mètode GGA, com del mètode de malles, amb les contribucions al mateix que s'han realitzat en el context d'esta tesi.

Índice general

Capítulo 1

Introducción y objetivos

1.1. Contexto

La simulación por computador de las redes de distribución de agua potable, mediante el uso de modelos matemáticos, es hoy en día una herramienta indispensable para el diseño y la explotación de dichas redes. Mediante la simulación es posible determinar, por ejemplo, si los usuarios recibirán los caudales de agua requeridos con una presión adecuada, si existen reservas de agua suficientes de las que disponer en caso de necesidad, o bien si puede haber problemas con la calidad del agua suministrada. Por ello, la simulación por computador se utiliza tanto en el diseño de nuevos abastecimientos y en ampliaciones o modificaciones de abastecimientos existentes, como en las tareas de operación normales de cualquier red, permitiendo en este último caso averiguar el comportamiento de la red ante determinados escenarios hipotéticos (por ejemplo, aumento del consumo de agua por parte de los usuarios; cierre de algunas tuberías por trabajos de mantenimiento o por avería, etc.).

Uno de los problemas básicos en el análisis de redes de agua es la *simulación hidráulica*, por medio de la cual se obtienen las presiones y caudales que se van a registrar durante un periodo determinado de la operación de la red (por ejemplo un día). La simulación hidráulica es un problema dinámico, pero puede tratarse de forma simplificada como una sucesión de problemas estáticos, que se enlazan a través de los balances de masa en los depósitos.

Un segundo problema importante es la *simulación de la calidad del agua*, por medio de la cual se obtiene información respecto a parámetros tales como la concentración de una determinada sustancia química (por ejemplo, cloro, o un contaminante determinado), el tiempo de permanencia del agua en la red, o el porcentaje de agua que proviene de una determinada fuente. Este tipo de información resulta esencial, dado que existe una preocupación creciente por garantizar niveles aceptables de la calidad del agua que se consume.

Además, la simulación de redes de agua mediante modelos matemáticos es la base para la consideración de otros objetivos más ambiciosos, como el diseño óptimo de redes, calibración del modelo, o la determinación de estrategias óptimas de operación

de la red, teniendo en cuenta factores como los costes energéticos de operación, la minimización de las fugas de agua, y/o el mantenimiento de una adecuada calidad del agua. Estos problemas se basan en la resolución de un gran número de problemas de simulación, con lo que resultan de una complejidad computacional considerable.

Por otro lado, el tamaño y nivel de detalle de los modelos de redes está creciendo, como consecuencia de la incorporación automática de datos contenidos en Sistemas de Información Geográfica (SIG). Todo ello hace que las tareas computacionales relacionadas con el análisis de redes de agua sean cada vez más complejas, por lo que las prestaciones del *solver* de simulación hidráulica y calidad del agua tienen un gran impacto en el tiempo total de cálculo necesario.

Una de las vías mediante las cuales se puede buscar la aceleración de los procesos de simulación es la Computación de Altas Prestaciones. El procesamiento paralelo ha experimentado un auge importante, tanto en el cálculo científico de altas prestaciones como en aplicaciones de propósito más general. El uso de plataformas de memoria distribuida, está bastante extendido, y en los últimos años han ganado aceptación sistemas multicore, así como las GPUs (*Graphics Processing Units*).

Algunos trabajos que utilizan la computación paralela se basan en lanzar múltiples simulaciones diferentes en paralelo, por ejemplo en contextos de optimización mediante algoritmos genéticos [53, 48, 57, 49]. Sin embargo, para ello se requiere que exista un número suficiente de simulaciones independientes, lo que no siempre ocurre, dado que va a depender del algoritmo utilizado y del problema a resolver. En este sentido, resulta necesario mejorar las prestaciones en la resolución del problema básico de simulación hidráulica, considerado de forma aislada, así como del problema de simulación de calidad del agua, lo cual podría combinarse con la ejecución de múltiples simulaciones en paralelo [33]. Otros trabajos consideran la utilización de GPUs, o bien de las instrucciones vectoriales de los procesadores actuales, como herramientas para acelerar las simulaciones [33, 34, 19].

Otra vía diferente para la reducción del tiempo de cálculo consiste en introducir mejoras en los métodos de simulación existentes para las redes de distribución de agua. En este sentido, el método que goza de mayor aceptación es el Algoritmo del Gradiente Global (GGA) [68], implementado en Epanet [58], un software de dominio público desarrollado por la Agencia de Protección Ambiental de Estados Unidos, y que es aún hoy día un referente para la simulación de sistemas de distribución de agua. En los últimos años ha habido diferentes trabajos considerando otros métodos, y en particular ha habido un renovado interés [16, 24, 10] en el método propuesto por Epp y Fowler [25], basado en la consideración de un conjunto de mallas en la red, cada una con un caudal corrector. Dicho interés se justifica por el hecho de que el método da lugar a un sistema de ecuaciones de menor dimensión, aunque presenta algunos problemas, dado que es necesario encontrar un conjunto de mallas independientes sobre la red, y el conjunto elegido determina la dispersidad del sistema obtenido, y por tanto condiciona fuertemente la eficiencia del método. Además, la consideración de algunos tipos de válvulas, o incluso el cierre de tuberías, obligaría a redefinir el conjunto de mallas, lo que supone una carga computacional importante.

En este contexto, en la tesis se pretende ahondar en las dos vías mencionadas para conseguir una reducción de tiempo de los procesos de simulación de redes de

abastecimiento de agua, tal como se detalla en el siguiente apartado.

1.2. Objetivo de la tesis

De acuerdo con el contexto presentado en la sección anterior, en esta tesis se abordan distintos caminos para mejorar las prestaciones de los procesos de simulación de redes de distribución de agua, y se lleva a cabo una evaluación de la adecuación de las vías exploradas. Se puede desglosar los objetivos de la siguiente manera:

- Desarrollo, implementación y evaluación de algoritmos para la simulación hidráulica eficiente mediante el método de mallas propuesto por Epp y Fowler [25]. En particular, se presentarán métodos para la obtención de un conjunto de mallas adecuado, que conduzcan a un sistema suficientemente disperso. Igualmente, se abordará la modelización de los distintos tipos de válvulas utilizados comúnmente, prestando especial atención, por su dificultad, a las válvulas reductoras de presión. Se presentarán métodos para llevar a cabo dicha modelización sin tener que redefinir el conjunto de mallas establecido inicialmente.

- Desarrollo, implementación y evaluación de algoritmos paralelos sobre plataformas de memoria distribuida para resolver problemas de simulación de redes de distribución de agua, considerando la simulación tanto de caudales y presiones como de la calidad del agua. A diferencia de otros trabajos mencionados anteriormente como [53, 48, 57, 49], donde se considera la realización en paralelo de distintas simulaciones independientes, lo que se persigue en esta tesis es la paralelización de una simulación de forma aislada, lo que constituye un problema de mayor complejidad, pero aplicable en un mayor rango de problemas.

- Desarrollo, implementación y evaluación de algoritmos paralelos sobre plataformas de memoria compartida para problemas de simulación de caudales y presiones en redes de distribución de agua. En particular, se considerarán sistemas *multicore*.

1.3. Organización de la tesis

Esta tesis se estructura en 7 capítulos. Hay que tener en cuenta que esta tesis implica dos disciplinas diferentes: ingeniería hidráulica y computación de altas prestaciones, por lo que se ha juzgado conveniente introducir algunos conceptos básicos de ambos campos, para facilitar la lectura a los especialistas en una u otra área. A continuación se describe brevemente el contenido de cada capítulo.

El capítulo 2 se dedica a revisar el estado del arte en el área de redes de distribución de agua. En primer lugar se introduce el problema de la simulación hidráulica y se revisan los métodos existentes para llevarla a cabo, incidiendo especialmente en los más relevantes desde el punto de vista de la tesis. Estos son el método del

Gradiente Global (GGA) y el método de mallas, para la simulación de caudales y presiones, y el método de Elementos Discretos de Volumen, para la simulación de la calidad del agua.

El capítulo 3 revisa aspectos relativos a la computación de altas prestaciones, presentando las plataformas *hardware* y herramientas disponibles para la programación.

En el capítulo 4 se presentan aportaciones para mejorar las prestaciones de la simulación hidráulica mediante el método de mallas. Se describen en primer lugar dos métodos nuevos para la selección de un conjunto de mallas independientes. Se aborda seguidamente la problemática de la modelización de ciertos tipos de válvulas, y se describen distintas formulaciones que permiten llevar a cabo dicha modelización sin necesidad de redefinir el conjunto de mallas. Por supuesto, se evalúan los distintos métodos presentados.

A continuación, en el capítulo 5 se describe la implementación de algoritmos paralelos sobre memoria distribuida para la simulación de redes de distribución de agua, teniendo en cuenta tanto la simulación hidráulica (caudales y presiones), como la de la calidad del agua. También se considera el problema de la minimización de fugas mediante la determinación de las consignas óptimas de una serie de válvulas reductoras de presión. Se lleva a cabo una evaluación de las implementaciones de los distintos métodos descritos en el capítulo.

Seguidamente, en el capítulo 6 se presenta la implementación de algoritmos paralelos sobre sistemas *multicore* para la simulación de caudales y presiones en redes de distribución de agua. Se busca con ello mejorar las prestaciones obtenidas con los correspondientes algoritmos sobre memoria distribuida. No sólo se considera en este capítulo la paralelización del método GGA de simulación hidráulica, sino que también se lleva a cabo la paralelización del método de mallas, con las incorporaciones al mismo introducidas en el capítulo 4. Los desarrollos planteados son evaluados sobre un conjunto de redes de test.

Para finalizar, el capítulo 7 expone las conclusiones de la tesis, las aportaciones originales y el trabajo futuro.

Capítulo 2

Contexto sobre simulación de redes de distribución de agua

En este capítulo se describe el contexto relacionado con la simulación de redes de distribución de agua. Como se expone en la sección 2.1, existen distintos tipos de modelos para la simulación de los caudales y presiones, de entre los cuales los relevantes en esta tesis son los modelos dinámicos no inerciales, que permiten tener en cuenta variaciones con el tiempo de los consumos, posiciones de los elementos de regulación y alturas en los depósitos. Además, la variación temporal y espacial de la concentración residual de sustancias químicas (reactivas o no) se puede llevar a cabo mediante modelos de calidad del agua, los cuales precisan haber obtenido previamente los caudales circulantes mediante los modelos hidráulicos correspondientes.

Después de introducir en el apartado 2.2 los distintos elementos que componen los modelos hidráulicos, el capítulo se centra en primer lugar en describir los modelos de simulación hidráulica, que determina la evolución de caudales y presiones, presentando las ecuaciones de equilibrio que gobiernan el proceso, y los distintos métodos de resolución. De entre esos métodos, se profundiza en dos de ellos por su importancia y por su relación con la tesis: el algoritmo del gradiente global (GGA), y el método de Newton-Raphson por mallas.

A continuación se pasa a considerar los modelos de simulación de la calidad del agua, donde de nuevo se presentan las ecuaciones de equilibrio correspondientes, y los métodos existentes para su resolución. En este caso se incide especialmente en el método de volúmenes discretos (DVEM, del inglés *Discrete Volume-Element Method*), cuya paralelización se aborda en en el capítulo 5 de esta tesis.

2.1. Modelos matemáticos hidráulicos

El estado de una red hidráulica a presión se encuentra en continuo cambio. Las válvulas son maniobradas, las bombas paran o arrancan, el nivel de agua en depósitos va cambiando, y todo ello para, entre otras cosas, dar respuesta a unas demandas

que varían en el tiempo y que, consecuentemente, provocan diferentes condiciones de suministro.

Muy pocas veces puede considerarse que la red hidráulica es algo "estático" en un sentido estricto, dado que las variables y los elementos hidráulicos (caudales circulantes, consumos, presiones, niveles de los depósitos, estado de bombas y válvulas, etc.) modifican sus valores o posiciones a lo largo del tiempo. En función del tratamiento que se dé a la variable temporal, podemos clasificar los modelos en estáticos o dinámicos, y a su vez, dentro de estos últimos, en inerciales y no inerciales. A continuación se describen brevemente los distintos tipos de modelos, de entre los cuales los modelos dinámicos no inerciales son el foco de atención de esta tesis.

Los *modelos estáticos* son aquellos que representan el estado de la red en un instante dado. En este tipo de modelos las variables características del sistema no varían con el tiempo. Se utilizan con fines estratégicos y de planificación, para comprobar cómo se va a comportar la red ante un determinado estado de carga, como por ejemplo ante unas demandas futuras, un corte del servicio por avería o mantenimiento, una ampliación de la red, o una nueva estrategia de regulación.

Los *modelos dinámicos*, por su parte, son aquellos que simulan el comportamiento de la red a lo largo del tiempo. En ellos, las variables principales (caudal y presión) se consideran variables temporales. Dentro de los modelos dinámicos están aquellos que tratan el fenómeno transitorio que se produce en las maniobras bruscas de los elementos de regulación, las cuales exigen considerar el agua com un medio compresible. Sin embargo, dichos fenómenos en redes urbanas sólo adquieren cierta importancia en las tuberías de traída desde los puntos de producción. En el resto de la red los transitorios se disipan en las muchas ramificaciones existentes, por lo que no se suelen tener en cuenta. Dentro de los modelos dinámicos que consideran el agua como un fluido incompresible encontramos los modelos inerciales y los no inerciales.

Los *modelos dinámicos inerciales* tienen en cuenta, como su propio nombre indica, la inercia del fluido, aunque no los cambios bruscos de presión provocados por la elasticidad del fluido. En definitiva, permiten simular el comportamiento dinámico de la red siempre que no haya cambios bruscos. La consideración del término de inercia, complica notablemente la formulación del problema, por lo que son poco utilizados.

Finalmente, los *modelos dinámicos no inerciales*, cuasi-estáticos o en *período extendido*, no tienen en cuenta los fenómenos inerciales, sino que simulan el comportamiento de la red como una sucesión de diferentes estados permanentes a lo largo del tiempo, teniendo en cuenta tan solo las variaciones en los consumos, posiciones de los elementos de regulación y alturas en los depósitos. Para la mayoría de casos prácticos, esta simplificación es perfectamente admitida, y son por ello los más utilizados.

Por otra parte, hay un creciente interés en el estudio de la calidad del agua en sistemas de distribución usando modelos matemáticos. La preocupación por la calidad del agua potable mientras permanece en la red de distribución, una vez ha abandonado la planta de tratamiento, surgió en EEUU en los años ochenta como consecuencia del endurecimiento por parte de la EPA (*Environmental Protection*

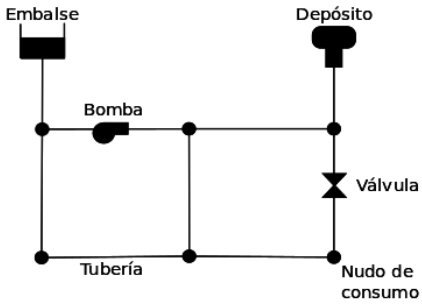

Figura 2.1: Componentes de una red de abastecimiento de agua.

Agency) de los requisitos que debía tener el agua al llegar a los puntos de consumo.

Los *modelos de calidad* se componen básicamente de dos partes: un modelo hidráulico y un modelo de calidad del agua propiamente dicho. Los modelos hidráulicos determinan los caudales y las presiones en la red bajo unas determinadas condiciones iniciales y de contorno, mientras que los modelos de calidad predicen la variación temporal y espacial de la concentración residual de cualquier sustancia química (reactiva o no) o fuente contaminante existente en la red, usando para ello la información de la distribución de caudales en el sistema.

Los modelos de calidad del agua se aplican al estudio del transporte de contaminantes en las redes de distribución de agua potable, determinando sus rutas, procedencias y tiempos de permanencia en la misma. También permiten analizar el transporte y reacción de los contaminantes bajo condiciones variables en el tiempo, como sucede en la realidad con las demandas, cambios de nivel en los depósitos, cierre y apertura de válvulas, arranque y paro de bombas, etc.

2.2. Componentes del modelo. Ecuaciones de comportamiento

A continuación describimos los elementos que componen un modelo de una red de distribución de agua. Se puede obtener más información en [58].

Una red de abastecimiento de agua consta de una serie de *líneas* conectadas a *nudos*. Las líneas pueden ser *tuberías*, *bombas* o *válvulas*, mientras que los nudos pueden ser *depósitos*, *embalses* o *nudos de consumo*. Mientras que los depósitos y embalses almacenan agua, los nudos de consumo (a los que llamaremos también *uniones*) son simplemente extremos de las líneas o puntos de conexión entre ellas, que pueden tener un caudal entrante/saliente asociado, conocido como *demanda* (negativa si es entrante). La figura 2.1 muestra un esquema de una red de ejemplo, con los distintos elementos presentados, que pasamos a describir a continuación.

Nudos de consumo. Las propiedades básicas asociadas a este tipo de nudos son la cota respecto a un nivel de referencia (usualmente el nivel del mar), la demanda de agua (flujo que abandona la red) y la calidad inicial del agua.

Los resultados obtenidos para los nudos en cada uno de los pasos de tiempo de simulación son la altura piezométrica (energía interna por unidad de peso del fluido, o bien suma de la cota más la altura de presión), la presión y la calidad del agua.

Los nudos de consumo pueden también presentar una demanda variable en el tiempo, tener asignados distintos tipos de demanda (doméstica, industrial, etc), presentar una demanda negativa, indicando que el caudal entra a la red a través del nudo, ser punto de entrada de un contaminante a la red o tener asociado un emisor (o hidrante) cuyo caudal de salida depende de la presión.

Embalses. Los embalses son nudos que representan una fuente externa de caudal, o bien un sumidero, de capacidad ilimitada. Se utilizan para modelizar elementos como lagos, captaciones desde ríos, acuíferos subterráneos, o también puntos de entrada a otros subsistemas. Los embalses pueden utilizarse también como puntos de entrada de contaminantes.

Las propiedades básicas de un embalse son su altura piezométrica (que coincidirá con la cota de la superficie libre del agua si éste se encuentra a la presión atmosférica), y la calidad del agua en el mismo en el caso de realizar un análisis de calidad.

Dado que un embalse actúa como un elemento de contorno del sistema, su altura o calidad del agua no se verán afectadas por lo que pueda ocurrir en la red.

Depósitos. Los depósitos son nudos con cierta capacidad de almacenamiento, en los cuales el volumen de agua almacenada puede variar con el tiempo durante la simulación. Los datos básicos de un depósito son la cota de solera (para la cual el nivel del agua es cero), el diámetro (o su valor equivalente si no es cilíndrico), el nivel inicial del agua, así como el mínimo y máximo, y la calidad inicial del agua.

Los principales resultados asociados a un depósito, a lo largo de la simulación, son la altura piezométrica (cota de la superficie libre) y la calidad del agua. El nivel del agua en los depósitos debe oscilar entre el nivel mínimo y el nivel máximo.

Tuberías. Las tuberías son líneas que transportan el agua de un nudo a otro. Se debe asumir que las tuberías están completamente llenas en todo momento, esto es, que el flujo es a presión. La dirección del flujo es siempre del nudo de mayor altura piezométrica al de menor altura piezométrica. Los principales parámetros de una tubería son: los nudos inicial y final, el diámetro, la longitud, el coeficiente de rugosidad (para calcular las pérdidas de carga) y su estado, esto es, si se encuentra abierta, cerrada, o posee válvula de retención. El parámetro de estado permite simular el hecho de que una tubería posea o no válvulas de corte o válvulas de retención sin tener que modelar estos elementos explícitamente.

Los datos de una tubería relacionados con los modelos de calidad son: el coeficiente de reacción en el medio y el coeficiente de reacción en la pared.

En relación a los resultados de la simulación, los asociados a las tuberías son: el caudal de circulación, la velocidad del flujo, la pérdida de carga unitaria, el factor de fricción para la fórmula de Darcy-Weisbach (descrita a continuación), la velocidad media de reacción a lo largo de la tubería y la calidad media del agua a lo largo de la conducción.

La pérdida de carga (o de altura piezométrica) en una tubería debida a la fricción por el paso del agua, puede calcularse utilizando tres fórmulas de pérdidas diferentes [58]:

- La fórmula de Hazen-Williams.
- La fórmula de Darcy-Weisbach.
- La fórmula de Chezy-Manning.

La fórmula de Hazen-Williams es la más utilizada en EEUU. Sin embargo, no puede utilizarse para líquidos distintos del agua, y fue desarrollada originalmente sólo para flujo turbulento. Desde el punto de vista académico, la fórmula de Darcy-Weisbach es la más correcta, y es aplicable a todo tipo de líquidos y regímenes. Finalmente, la fórmula de Chezy-Manning es utilizada usualmente para canales y tuberías de gran diámetro, donde la turbulencia está muy desarrollada.

Todas las fórmulas emplean la misma ecuación básica para calcular la pérdida de carga entre el nudo de entrada y el de salida. Esta viene dada por la suma de un término principal más una *pérdida menor*, esto es:

$$\phi_k = r_k q_k^{\beta} + \tilde{\phi}_k$$

donde ϕ_k es la pérdida de carga total (en unid. longitud) de la tubería k, $\tilde{\phi}_k$ es la pérdida menor, q_k es el caudal de la tubería, r_k es el coeficiente de resistencia, y β el exponente del caudal.

El coeficiente de resistencia se calcula de forma diferente dependiendo de la fórmula empleada, y el valor de β también es distinto en cada caso. En concreto:

- Hazen-Williams: $r_k = 10{,}674\, C_k^{-1{,}852} d_k^{-4{,}871} L_k, \quad \beta = 1{,}852$
- Darcy-Weisbach: $r_k = 0{,}0827\, f(\varepsilon_k, d_k, q_k)\, d_k^{-5} L_k, \quad \beta = 2$
- Chezy-Manning: $r_k = 10{,}294\, \nu_k^2 d_k^{-5{,}33} L_k, \quad \beta = 2$

donde $C_k, \varepsilon_k, \nu_k$, son los coeficientes de rugosidad de Hazen-Williams, Darcy-Weisbach (m) y Manning, respectivamente, d_k es el diámetro de la tubería (m), L_k la longitud (m), q_k el caudal (m^3/s), y f es el factor de fricción de Darcy-Weisbach, que depende de ε_k, d_k y q_k.

Cada fórmula utiliza un coeficiente de rugosidad distinto, el cual debe determinarse empíricamente. En la práctica hay que ser conscientes de que el valor de estos coeficientes puede cambiar considerablemente con la edad de las tuberías. Además, es destacable que en los casos de Hazen-Williams y Chezy-Manning, el coeficiente de resistencia r_k es constante durante toda la simulación, dado que sólo depende del

coeficiente de rugosidad correspondiente, el diámetro y la longitud. Sin embargo, en el caso de Darcy-Weisbach el coeficiente r_k depende también del caudal, por lo que debe recalcularse cada vez que se quiera obtener la pérdida de carga.

Por otra parte, la pérdida de carga menor $\tilde{\phi}_k$ es debida al incremento de la turbulencia cuando el flujo pasa por un codo o un accesorio. La importancia de incluir o no tales pérdidas depende del tipo de red modelada y de la precisión deseada de los resultados. Para tenerlas en cuenta hay que incluir entre los datos de la tubería el coeficiente de pérdidas menores. El valor de la pérdida será el producto de dicho coeficiente por la altura dinámica en la tubería, esto es:

$$\tilde{\phi}_k = \tilde{\rho}_k \frac{v_k^2}{2g} = \frac{8}{g\pi^2 d_k^4} \tilde{\rho}_k q_k^2 = \rho_k q_k^2$$

donde $\tilde{\rho}_k$ es el coeficiente de pérdidas menores (adimensional), v_k la velocidad del flujo, y g la aceleración de la gravedad.

Como conclusión, teniendo en cuenta las fórmulas presentadas, y el hecho de que el caudal en una tubería puede ser negativo respecto al sentido arbitrario asignado a la misma, la pérdida de carga entre el nudo inicial i y el final j de una tubería k se puede expresar mediante la fórmula general:

$$h_i - h_j = \phi_k = q_k(r_k|q_k|^{\beta-1} + \rho_k|q_k|) \tag{2.1}$$

Bombas. Las bombas son líneas que comunican energía al fluido elevando su altura piezométrica. Los datos principales de una bomba son sus nudos de aspiración e impulsión y su curva característica (o relación entre caudal trasegado y la altura comunicada). El resultado principal asociado a una bomba es el incremento de altura comunicada al fluido. Se asume que el caudal a través de la bomba es de sentido único, siempre de su nudo inicial a su nudo final.

Si se considera una bomba con velocidad de giro variable, la pérdida de carga (ganancia de carga en negativo) entre el extremo inicial i y el final j de la bomba k, queda en función de la relación ω_k entre la velocidad de giro en un determinado instante y su velocidad de giro nominal:

$$h_i - h_j = \phi_k = -\omega_k^2(h_{0k} - r_k(q_k/\omega_k)^{\gamma_k}) \tag{2.2}$$

donde h_{0k} es la altura a caudal nulo, ω_k es la velocidad relativa de giro, y r_k y γ_k son coeficientes de la curva de la bomba.

En lugar de dar la curva característica, el comportamiento de una bomba puede también modelarse suponiendo que aporta una cantidad de energía constante al fluido por unidad de tiempo (potencia constante), lo que permite determinar la altura comunicada al fluido en función del caudal de paso.

Al igual que las tuberías, las bombas deben poder pararse o arrancarse durante la simulación en instantes prefijados, o cuando se cumplan determinadas condiciones en la red. El modo de operación de una bomba puede también imponerse asignando una curva de modulación a su velocidad relativa.

Válvulas. Las válvulas son líneas que limitan la presión o el caudal en un punto determinado de la red. Las propiedades principales asociadas a una válvula son los nudos aguas arriba y aguas abajo, el diámetro, la consigna y su estado. Los resultados de la simulación asociados con una válvula son el caudal de paso y la pérdida de carga que origina.

Los tipos básicos de válvulas contemplados normalmente en un modelo son:

- Válvulas Reductoras de Presión (VRP) (en inglés PRV)
- Válvulas Sostenedoras de Presión (VSP) (en inglés PSV)
- Válvulas de Rotura de Carga (VRC) (en inglés PBV)
- Válvulas de Control del Caudal (VCQ) (en inglés FCV)
- Válvulas de Regulación (VRG) (en inglés TCV)
- Válvulas de Propósito General (VPG) (en inglés GPV).

Las VRP tratan de limitar la presión en el nudo aguas abajo de la válvula. La válvula puede estar en tres estados diferentes:

- Completamente abierta, si la altura piezométrica del punto de entrada (nudo aguas arriba) es inferior al valor de consigna.
- Cerrada, si las alturas piezométricas de los extremos de la válvula producirían un caudal inverso.
- Activa (o parcialmente abierta), en otro caso. En este estado la presión de salida es igual al valor de la consigna.

De manera similar, las VSP tratan de mantener la presión en el nudo aguas arriba de la válvula. La válvula puede estar también en uno de los tres estados siguientes:

- Completamente abierta, si la altura piezométrica del nudo aguas abajo es superior al valor de consigna.
- Cerrada, si las alturas piezométricas de los extremos de la válvula producirían un caudal inverso.
- Activa (o parcialmente abierta), en otro caso. En este estado la presión de entrada es igual al valor de la consigna.

Las válvulas VRC fuerzan la caída de presión a través de la válvula. El flujo a través de la válvula puede ser en cualquier dirección. Estas válvulas no representan a ningún componente físico, pero son muy útiles para modelar situaciones en las que la caída de presión a través de la válvula es conocida.

Las válvulas VCQ limitan el caudal de paso a través de la válvula a un valor prefijado. Son unidireccionales, y deben orientarse según el sentido del flujo a limitar.

Las válvulas VRG son bidireccionales y simulan una válvula parcialmente cerrada, cuyo comportamiento queda determinado por el valor del coeficiente de pérdidas

en la válvula. Usualmente los fabricantes proporcionan la relación entre dicho coeficiente y el grado de apertura de la válvula.

Las VPG se utilizan para representar una línea cuya relación pérdida-caudal es proporcionada por el usuario, en lugar de seguir el comportamiento típico de las válvulas, establecido por la ecuación de pérdidas. Pueden utilizarse para modelizar una turbina, el descenso dinámico de un pozo o una válvula reductora de presión controlada por caudal.

Las *válvulas de corte* (tipo compuerta) y las *válvulas de retención* (o *antirretorno*), cuya acción es abrir o cerrar totalmente el paso del flujo, no se consideran como líneas independientes, sino que se suelen incorporar como propiedades de la tubería en la cual se instalan.

2.3. Ecuaciones de equilibrio en la simulación hidráulica

En este apartado y los dos siguientes nos ocupamos de la simulación hidráulica (de caudales y presiones). La simulación de la calidad del agua se trata a partir del apartado 2.6.

El problema de simulación del comportamiento de una red de distribución de agua a lo largo de un periodo de simulación determinado se descompone en una serie de problemas en régimen permanente encadenados, en lo que se conoce como simulación en *periodo extendido*, como se ha comentado en el apartado 2.1 y veremos con más detalle en el 2.4.

Consideramos aquí el problema en régimen permanente. Éste consiste en obtener los caudales de la líneas y las alturas piezométricas (o bien presiones) en los nudos de consumo, dadas las demandas en éstos, y conocidas las características de las conducciones y otros elementos de regulación presentes, como bombas o válvulas, su estado abierto o cerrado, así como los niveles de los depósitos y embalses desde los cuales se alimenta la red o a los cuales fluye el agua a través de la misma.

Para plantear las ecuaciones de equilibrio correspondientes al problema en régimen permanente, hay que tener en cuenta que la diferencia entre las alturas piezométricas de los extremos de cada línea viene dada por las ecuaciones (2.1) y (2.2), para tuberías y bombas, respectivamente. De esta manera, se plantea el siguiente conjunto de m ecuaciones:

$$\phi_k(q_k) - h_{k_1} + h_{k_2} = 0, \qquad k = 1, 2, \ldots, m \tag{2.3}$$

donde m es el número de líneas de la red, $\phi_k(q_k)$ denota la pérdida de carga provocada por el paso del caudal q_k por la línea k, mientras que k_1 y k_2 representan el nudo inicial y final, respectivamente, de la misma línea. Cada uno de estos dos nudos puede ser un nudo de consumo o un depósito/embalse. En este último caso, la variable h asociada es conocida.

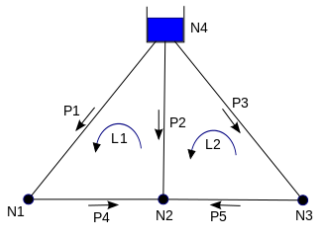

Figura 2.2: Red simple.

Por ejemplo, considerando la red de la figura 2.2, tendríamos:

$$\left.\begin{array}{l}\phi_1(q_1)+h_1-h_4=0\\ \phi_2(q_2)+h_2-h_4=0\\ \phi_3(q_3)+h_3-h_4=0\\ \phi_4(q_4)+h_2-h_1=0\\ \phi_4(q_5)+h_2-h_3=0\end{array}\right\}$$

Por otra parte, en cada nudo de consumo se verifica que el caudal entrante debe ser igual al saliente, lo que da lugar a las siguientes n ecuaciones:

$$\sum_{k=1}^{m}\delta_{ki}q_k-c_i=0,\qquad i=1,2,\ldots,n \tag{2.4}$$

donde n es el número de nudos de consumo, c_i es la demanda del nudo i, mientras que $\delta_{ki}\in\{-1,0,+1\}$ indica la conexión entre la línea k y el nudo de consumo i, siendo igual a $+1$ si el nudo es el extremo final de la línea, -1 si es el extremo inicial, y 0 si no está conectado a la línea.

En el ejemplo de la figura 2.2 tendríamos:

$$\left.\begin{array}{r}q_1-q_4=0\\ q_2+q_4+q_5=0\\ q_3-q_5=0\end{array}\right\}$$

Alternativamente, se pueden expresar las ecuaciones (2.3) y (2.4) de forma matricial. En concreto, para (2.3) tendríamos:

$$\phi(q)+A_{12}\tilde{h}+A_{10}\hat{h}=0 \tag{2.5}$$

donde q es el vector de caudales, $\phi(q)=[\phi_1(q_1),\phi_2(q_2),\cdots\phi_m(q_m)]^T$ es la función de pérdidas de carga, $\tilde{h}$ y $\hat{h}$ son los vectores de alturas piezométricas de los nudos de consumo y depósitos/embalses, respectivamente, mientras que A_{12} y A_{10} son matrices de incidencias que representan las conexiones de las líneas con los nudos.

Más concretamente, A_{12} es una matriz (de tamaño $m\times n$) de incidencias entre líneas y nudos de consumo, cuyo elemento (i,j), igual a δ_{ij}, corresponde a la conexión

entre la línea i y el nudo de consumo j. De la misma manera, llamando n_s al número de depósitos/embalses, A_{10} es una matriz de tamaño $m \times n_s$, de incidencias entre líneas y depósitos/embalses, definida de manera análoga a A_{12}.

En el ejemplo de la figura 2.2, tendríamos:

$$A_{12} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix}, \qquad A_{10} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ 0 \\ 0 \end{bmatrix}$$

Por otra parte, el conjunto de ecuaciones (2.4) se puede expresar en forma matricial como:

$$A_{12}^T q - c = 0 \tag{2.6}$$

donde c es el vector de demandas de los nudos de consumo.

Las ecuaciones (2.3) y (2.4), o bien en forma matricial (2.5) y (2.6), constituyen un sistema de $n + m$ ecuaciones y el mismo número de incógnitas, formadas por el vector de caudales q (m elementos) y el vector de alturas piezométricas de los nudos de consumo $\tilde{h}$ (n elementos).

Sin embargo, es posible plantear otro conjunto adicional de ecuaciones, correspondientes a la conservación de energía en las mallas de la red. Es decir, en cada malla cerrada de la red debe cumplirse que la suma de las pérdidas de carga de las líneas que la forman debe ser nula, como consecuencia de la unicidad de las alturas h en cada nudo. Por tanto, dado un conjunto de n_m mallas independientes sobre una red, se verifica que:

$$\sum_{k=1}^{m} \mu_{ik} \phi_k(q_k) = 0, \qquad i = 1, 2 \ldots n_m \tag{2.7}$$

donde μ_{ik} es 0 si la línea k no forma parte de la malla i, y en caso contrario es $+1$ o -1, dependiendo de si la línea k sigue el sentido positivo o negativo de la malla i, respectivamente.

Volviendo al ejemplo de la figura 2.2, se pueden considerar las mallas L1 (formada por las tuberías P1, P4 y P2), y L2 (formada por P2, P5 y P3), asumiendo para las mismas el sentido antihorario. De acuerdo con esto, el conjunto de ecuaciones (2.7) sería en este caso:

$$\left.\begin{aligned} \phi_1(q_1) - \phi_2(q_2) + \phi_4(q_4) = 0 \\ \phi_2(q_2) - \phi_3(q_3) - \phi_5(q_5) = 0 \end{aligned}\right\}$$

En general, en una red se pueden encontrar conjuntos de hasta $m - (n + n_s) + 1$ mallas independientes, pudiendo haber múltiples conjuntos posibles.

En el caso de redes con varios depósitos/embalses, cabe considerar además que la suma de pérdidas de carga a lo largo de un camino que vaya de un depósito/embalse a otro debe ser igual a la diferencia de alturas piezométricas entre ambos depósitos/embalses, la cual es conocida de antemano. Eso supone expandir el conjunto de ecuaciones dado por (2.7), añadiendo ecuaciones correspondientes a *pseudo-mallas*

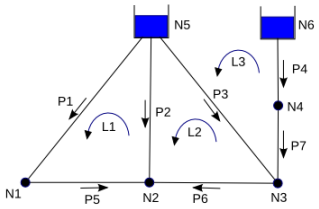

Figura 2.3: Red con dos depósitos y una pseudo-malla L3.

(o sea, caminos que van de un depósito/embalse a otro). En general, al conjunto de mallas reales considerado inicialmente se le pueden añadir $n_s - 1$ pseudo-mallas independientes, con lo que el sistema (2.7) se transforma en el siguiente conjunto de $m - n$ ecuaciones:

$$z_i + \sum_{k=1}^{m} \mu_{ik}\phi_k(q_k) = 0, \qquad i = 1, 2 \ldots m - n \tag{2.8}$$

donde $z_i = 0$ si la malla i es real, y $z_i = \hat{h}_{i_2} - \hat{h}_{i_1}$ si es una pseudo-malla que va del depósito i_1 al i_2.

Esto puede verse sobre la red de ejemplo de la figura 2.3, donde consideraríamos las mallas reales L1 (tuberías P1, P5 y P2) y L2 (tuberías P2, P6 y P3), junto con la pseudo-malla L3 (tuberías P3, P7 y P4). El correspondiente sistema de ecuaciones sería:

$$\left.\begin{aligned} \phi_1(q_1) + \phi_5(q_5) - \phi_2(q_2) &= 0 \\ \phi_2(q_2) - \phi_6(q_6) - \phi_3(q_3) &= 0 \\ h_6 - h_5 + \phi_3(q_3) - \phi_7(q_7) - \phi_4(q_4) &= 0 \end{aligned}\right\}$$

El conjunto de ecuaciones (2.8) se puede expresar en forma matricial de la siguiente manera:

$$M_{31}A_{10}\hat{h} + M_{31}\phi(q) = 0 \tag{2.9}$$

donde M_{31} es una matriz de incidencias entre mallas y líneas, de tamaño $(m-n)\times m$, donde el elemento (i, j) es igual a μ_{ij}.

En la red de la figura 2.3 las matrices de incidencias M_{31} y A_{10} serían:

$$M_{31} = \begin{bmatrix} 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 \end{bmatrix}, \qquad A_{10} = \begin{bmatrix} -1 & 0 \\ -1 & 0 \\ -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

2.4. Simulación en periodo extendido

Una simulación en período extendido no es más que una sucesión de simulaciones en régimen permanente enlazadas entre sí, en las que los resultados de cada simulación para un instante determinado, se utilizan como condiciones iniciales de contorno en la simulación para el instante siguiente.

Este tipo de modelo de simulación hidráulica calcula las alturas piezométricas en los nudos y los caudales en las líneas, dados los niveles iniciales en los embalses y depósitos, y la sucesión en el tiempo de las demandas aplicadas en los nudos. De un instante al siguiente se actualizan los niveles en los depósitos conforme a los caudales calculados que entran o salen de los mismos, y las demandas en los nudos y niveles en los embalses conforme a sus curvas de modulación.

El proceso se describe en el algoritmo 2.1, donde para obtener las alturas y caudales en un determinado instante se resuelve el problema en régimen permanente mediante las ecuaciones presentadas en el apartado 2.3. Los métodos para resolver este problema se presentan en el siguiente apartado.

Algoritmo 2.1 Simulación hidráulica en periodo extendido

```
leer datos de la red desde un fichero de entrada
inicialización (t = 0, ...)
mientras t < t_final hacer
    calcular demandas y aplicar controles en válvulas y bombas
    resolver problema hidráulico estático, obteniendo q, h
    escribir resultados del tiempo t en un fichero
    avanzar t y  actualizar niveles de depósitos
fin mientras
```

La primera tarea llevada a cabo por el algoritmo consiste en leer los datos de la red de un fichero de entrada, que contiene toda la información relativa a los componentes del sistema (nudos, tuberías, bombas, válvulas, depósitos y embalses), con sus correspondientes propiedades (longitudes y rugosidades de tuberías, curvas características de bombas y válvulas, etc.), y las demandas a aplicar a los nudos. También puede incluir una serie de reglas de control, las cuales indican la forma en que deben actuar los elementos de regulación (válvulas y bombas) de la red.

A continuación, en el proceso de inicialización se llevan a cabo tareas que dependen del método elegido para resolver el problema estático, que pueden comprender determinar la estructura de dispersidad de la matriz correspondiente a los sistemas lineales a resolver, realizando una reordenación y descomposición simbólica de la matriz. En algunos métodos esta inicialización también incluye encontrar un conjunto de mallas independientes sobre la red. Obsérvese que estas tareas se hacen una sola vez, en vez de repetirse para cada sistema lineal a resolver, ya que sólo dependen de la conexión entre los nudos de la red hidráulica, y ésta es siempre la misma.

Tras la fase de inicialización se pasa al bucle que lleva a cabo la simulación del sistema en periodo extendido. En él, el proceso de análisis hidráulico estático se repite para sucesivos instantes de tiempo hasta que se completa el periodo de simulación.

El paso de tiempo es proporcionado por el usuario, siendo frecuentemente de 1 hora, aunque se debe acortar automáticamente cuando ocurre algún evento que lo hace necesario (llenado o vaciado de un depósito o activación de una regla de control).

Tras realizar el análisis hidráulico estático, la solución obtenida es guardada en un fichero, y a continuación se determina cuál debe ser el siguiente paso de tiempo, actualizándose en consecuencia el tiempo actual t, y el nivel de agua de los depósitos. El proceso se repite hasta completar el periodo de simulación indicado por el usuario.

2.5. Métodos de solución del problema hidráulico estático

En la simulación hidráulica mediante periodo extendido, la parte de mayor complejidad es resolver el problema de equilibrio en régimen permanente, que satisface las ecuaciones presentadas en el apartado 2.3.

Para la solución de ese problema estático o en régimen permanente, se han propuesto diversos métodos. Ya en 1936, Hardy Cross [18] formuló dos métodos para la resolución del problema, que debido sobre todo a su simplicidad consiguieron amplia aceptación, estando considerados como el punto de inicio del análisis de redes. Los métodos fueron ampliamente usados en la resolución manual del problema, así como en los primeros programas para el cálculo por ordenador. El más conocido de ellos es el *método de mallas de Cross* (*loop Cross method*), también llamado *método de ajuste de malla única* (*single path adjustment*), y basado en las ecuaciones de mallas (2.8).

El desarrollo posterior de nuevos métodos estuvo motivado por la disponibilidad de computadores digitales que permitieron el uso de técnicas matemáticas más sofisticadas, como el método de Newton-Raphson. El interés por el desarrollo de nuevos algoritmos fue debido a la escasa fiabilidad de los métodos existentes cuando se enfrentaban a algunos problemas mal condicionados. De entre los distintos métodos propuestos, cabe destacar los siguientes, todos ellos basados en el método de Newton-Raphson para la resolución de sistemas de ecuaciones no lineales:

1. **Newton-Raphson por nudos**. Este método fue propuesto por [50]. El método se basa en la aplicación del método de Newton-Raphson sobre las ecuaciones nodales (2.4). Para ello, se obtiene para cada línea una expresión de su caudal en función de su pérdida de altura, lo que se hace por medio de (2.1) para las tuberías y por medio de (2.2) para las bombas.

   Estas expresiones se introducen en (2.4), con lo que se obtiene un sistema no lineal de n ecuaciones con n incógnitas (el vector de alturas piezométricas $\tilde{h}$), que se resuelve mediante Newton-Raphson. A continuación, los caudales q se obtienen a partir de las alturas piezométricas.

2. **Formulación nodal de Shamir y Howard** [63]. La publicación de este método se convirtió en un clásico de obligada referencia para trabajos futuros. Al igual que en el caso anterior, se utiliza la formulación nodal, aunque la

característica a destacar de este método es el hecho de que permite combinar incógnitas de diferentes tipos, tanto alturas piezométricas y caudales, como consumos nodales, diámetros de tuberías o sus rugosidades. Presenta además una metodología para realizar un análisis de sensibilidad y determinar el efecto de los cambios en una determinada variable.

3. **Newton-Raphson por mallas** (*looped Newton-Raphson* o *simultaneous path adjustment method*). Fue propuesto por Epp y Fowler [25], y parte, al igual que el método de *single path adjustment* de Cross, de las $m - n$ ecuaciones de conservación de energía en las mallas de la red (2.8), pero en este caso se resuelven simultáneamente.

   Además, la consideración de las ecuaciones de conservación de masa en los nudos (2.4) permite reducir el vector de caudales de líneas a un vector de caudales asociados a cada una de las mallas, con lo que pasa a haber $m - n$ incógnitas (los caudales de mallas). Sobre el sistema obtenido de esta manera se aplica el método de Newton-Raphson.

   Este método será explicado con más detalle en la sección 2.5.2.

4. **Método del gradiente global de Todini** (GGA) [68]. Este método se basa en plantear el sistema de $m+n$ ecuaciones e incógnitas proporcionado por (2.3) y (2.4), y en aplicar al mismo el método de Newton-Raphson. Al igual que en el método de Newton-Raphson por nudos, no considera las ecuaciones de conservación de energía en las mallas de la red, cuyo cumplimiento es automático al imponer una altura única por nudo.

   Este es el método utilizado por el *software* Epanet. Tiene un papel destacado en esta tesis, pues es la base de distintos algoritmos paralelos tanto sobre memoria distribuida como sobre memoria compartida. Por esta razón, el método se describe con mayor detalle en la sección 2.5.1.

5. **Método de teoría lineal basado en gradientes** (*gradient-based linear theory method*). Este método fue desarrollado por Wood [71], sobre la base del método original de teoría lineal de Wood y Charles [72]. Básicamente consiste en considerar las ecuaciones de conservación de energía en las mallas (2.8), junto con las de conservación de masa en los nudos (2.4), con lo que se obtiene un sistema de m ecuaciones con m incógnitas (los caudales de líneas), que se resuelve mediante Newton-Raphson.

Las características de las matrices correspondientes a los sistemas lineales en cada uno de los métodos pueden verse en la tabla 2.1. En el caso del método de Gradiente Global, el sistema planteado inicialmente tiene una matriz de tamaño $(m + n) \times (m + n)$, pero este sistema se desacopla, reduciéndose a uno de tamaño $n \times n$, como veremos en el apartado 2.5.1.

Todini y Rossman ofrecen en [69] un marco unificado de los distintos métodos conocidos para el problema hidráulico estático, derivándolos mediante transformaciones lineales y no lineales a partir de las ecuaciones que expresan la conservación

Tabla 2.1: Características de la matriz del sistema de ecuaciones lineales para distintos métodos.

Método	Tamaño de la matriz	Simetría
Newton-Raphson por nudos	$n \times n$	Sí
Shamir y Howard	$n \times n$	No
Newton-Raphson por mallas	$(m-n) \times (m-n)$	Sí
Gradiente global	$n \times n$	Sí
Teoría lineal	$m \times m$	No

de la masa y energía, es decir, ecuaciones (2.6) y (2.5). Se indica que los métodos que utilizan la formulación nodal (los dos primeros métodos presentados anteriormente), tienen peores propiedades de convergencia que el resto, como se había discutido anteriormente en otros trabajos [63, 71].

Además, [69] muestra que las formulaciones del método de Newton-Raphson por mallas y del método de teoría lineal basado en gradientes se pueden obtener a partir de la formulación correspondiente al método del gradiente global, sin más que aplicar simples transformaciones lineales. De esta manera, aunque el tamaño y propiedades de los sistemas lineales a resolver en cada iteración son distintos dependiendo del método utilizado, los tres métodos son equivalentes en términos de propiedades de convergencia. Es decir, utilizando aritmética exacta, las iteraciones producidas por los tres métodos serán las mismas, siempre que se parta de la misma solución inicial balanceada. El requisito de que la solución inicial esté balanceada, o sea que verifique (2.4), solo es necesario para poder aplicar el método de Newton-Raphson por mallas.

El tamaño, así como la simetría o no, de las matrices correspondientes a los sistemas lineales a resolver, varían dependiendo del método [67, 69], tal como puede verse en la tabla 2.1. Las redes de abastecimiento de agua están por lo general poco malladas, es decir, el número de mallas, dado por $m - n$, es generalmente mucho menor que m o n. Por tanto, el método de Newton-Raphson por mallas es el que presenta un menor tamaño de matriz. Por el contrario, el mayor tamaño de matriz corresponde al método de teoría lineal, que además tiene el inconveniente de utilizar una matriz no simétrica.

Otro aspecto a tener en cuenta es que tanto el método de Newton-Raphson por mallas como el de teoría lineal requieren la definición previa de un conjunto de mallas independientes sobre la red. Como se discutirá en el capítulo 4, existen muchos posibles conjuntos de mallas independientes, pero la elección de uno u otro afecta considerablemente a la dispersidad de la matriz correspondiente a los sistemas lineales a resolver, y consecuentemente, a las prestaciones del método. Además, ambos métodos requieren una solución inicial de caudales que satisfaga la continuidad de flujo (2.4).

El trabajo desarrollado en esta tesis se centra en los métodos de gradiente global y Newton-Raphson por mallas. Por este motivo, en los siguientes apartados se

introducen más a fondo ambos métodos.

2.5.1. Método del gradiente global de Todini (GGA)

El estudio del problema de flujo en régimen permanente se puede enfocar como la minimización de la potencia disipada en la red, sujeta a la restricción de continuidad de flujo. Bajo esta aproximación, Todini y Pilati [66, 67] mostraron que las condiciones necesarias para la operación en régimen permanente de una red de distribución de agua son las ecuaciones (2.5) y (2.6) enunciadas en el apartado 2.3.

Ambas ecuaciones forman un sistema de ecuaciones no lineales que se puede expresar en la forma $F(x) = 0$, siendo

$$x = \begin{bmatrix} q \\ \tilde{h} \end{bmatrix},$$

y resolver mediante el método de Newton-Raphson. De esta forma, partiendo de una aproximación x^k de la solución, la nueva aproximación x^{k+1} vendrá dada por la solución del sistema lineal

$$F'(x^k)\Delta x = -F(x^k), \tag{2.10}$$

siendo $x^{k+1} = x^k + \Delta x$, y $F'(x^k)$ la matriz *Jacobiana* de $F(x)$ en el punto x^k.

En el caso que nos ocupa, el sistema lineal (2.10) toma la forma

$$\begin{bmatrix} D & A_{12} \\ A_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ \Delta\tilde{h} \end{bmatrix} = \begin{bmatrix} -\phi(q^k) - A_{12}\tilde{h}^k - A_{10}\hat{h} \\ -A_{12}^T q^k + c \end{bmatrix}, \tag{2.11}$$

donde $\Delta q = q^{k+1} - q^k$, $\Delta\tilde{h} = \tilde{h}^{k+1} - \tilde{h}^k$, mientras que D es una matriz diagonal de dimensiones $m \times m$, de manera que el elemento d_{ii}, o simplemente d_i, corresponde a la derivada de la pérdida de carga ϕ_i de la línea i, respecto del caudal que circula por la misma. Por tanto, de acuerdo con las expresiones (2.1) y (2.2), tenemos que si la línea i es una tubería:

$$d_i = \frac{\partial\phi_i(q_i)}{\partial q_i} = \beta r_i \left|q_i\right|^{\beta-1}$$

y si la línea i es una bomba:

$$d_i = \frac{\partial\phi_i(q_i)}{\partial q_i} = \gamma_i r_i \omega_i^{2-\gamma_i} q_i^{\gamma_i - 1}$$

Reescribiendo (2.11) tenemos:

$$D\Delta q + A_{12}\Delta\tilde{h} = -\phi(q^k) - A_{12}\tilde{h}^k - A_{10}\hat{h} \tag{2.12}$$

$$A_{12}^T\Delta q = -A_{12}^T q^k + c \tag{2.13}$$

y despejando Δq de (2.12) resulta:

$$\Delta q = -D^{-1}\left(\phi(q^k) + A_{12}\tilde{h}^{k+1} + A_{10}\hat{h}\right), \tag{2.14}$$

expresión que podemos sustituir en (2.13), de manera que

$$A_{12}^T D^{-1} \left(\phi(q^k) + A_{12}\tilde{h}^{k+1} + A_{10}\hat{h} \right) = A_{12}^T q^k - c,$$

de donde se obtiene el vector $\tilde{h}^{k+1}$ mediante la resolución del siguiente sistema lineal:

$$\left(A_{12}^T D^{-1} A_{12} \right) \tilde{h}^{k+1} = -A_{12}^T D^{-1} \left(\phi(q^k) + A_{10}\hat{h} \right) + A_{12}^T q^k - c. \tag{2.15}$$

Finalmente q^{k+1} se obtiene de (2.14), en particular

$$q^{k+1} = q^k - D^{-1} \left(\phi(q^k) + A_{12}\tilde{h}^{k+1} + A_{10}\hat{h} \right). \tag{2.16}$$

Las expresiones (2.15) y (2.16) corresponden a la iteración del método del gradiente global (GGA). Obsérvese que el sistema lineal planteado inicialmente en (2.11) es de $m + n$ ecuaciones e incógnitas, aunque se reduce posteriormente a las expresiones (2.15) y (2.16), la primera de las cuales supone la resolución de un sistema lineal de menor tamaño (matriz de coeficientes, $A_{12}^T D^{-1} A_{12}$, de dimensiones $n \times n$, simétrica y definida positiva).

El patrón de ceros y no-ceros de la matriz $A = A_{12}^T D^{-1} A_{12}$ está estrechamente ligado a las conexiones existentes entre los nudos de la red hidráulica. En particular, es fácil ver que un elemento $a_{i,j}$ de la matriz es diferente de cero sólo si existe una línea que une los nudos i y j. Obsérvese que si reordenamos los nudos de la red se alterará el patrón de ceros de la matriz, hecho que puede afectar enormemente la eficiencia de la resolución del sistema de ecuaciones lineales.

Algoritmo 2.2 Simulación de un paso de tiempo mediante GGA

mientras no convergencia **hacer**
  calcular matriz y vector del sistema lineal (2.15)
  resolver sistema lineal anterior, obteniendo h^{k+1}
  obtener nuevos caudales q^{k+1} mediante ec. (2.16)
  actualizar estado de válvulas, bombas y tuberías
fin mientras

El método GGA se describe mediante el algoritmo 2.2. Puede verse que tras calcular el vector de caudales q^{k+1} en cada iteración, se comprueba si se han producido cambios en el estado de las válvulas, bombas e incluso tuberías, realizando las actualizaciones de estado necesarias. Este proceso se repite hasta que se logre la convergencia, o bien se exceda un número máximo de iteraciones.

Implementación del método GGA en Epanet

En este apartado se comentan algunos aspectos de la implementación del método GGA en el paquete Epanet, ya que este es el punto de partida para las implementaciones paralelas descritas en los capítulos 5 y 6 de esta tesis.

Epanet utiliza el método GGA para resolver el problema estático dentro de un proceso de simulación en periodo extendido como el presentado en el algoritmo 2.1. Utiliza la descomposición de Cholesky para la resolución de los sistemas lineales (2.15), utilizando almacenamiento disperso para la matriz del sistema, y llevando a cabo al inicio de la simulación una reordenación y descomposición simbólica de dicha matriz. La reordenación se realiza mediante el algoritmo de *Mínimo Grado* [26], un algoritmo de reordenación muy utilizado para todo tipo de sistemas lineales dispersos.

2.5.2. Método de mallas

El método de Newton-Raphson por mallas, propuesto en en [25], puede considerarse según se indica en diferentes trabajos [24, 2, 1], como un método de *espacio nulo*, dado que trabaja con vectores de caudales expresados en el espacio nulo de la matriz A_{12}^T. Por el contrario, el método GGA es un método de *espacio rango*, pues trabaja con vectores de caudales expresados en el espacio rango de la matriz A_{12}^T.

Como se ha comentado previamente en la sección 2.5, el método de Newton-Raphson por mallas considera el conjunto de $m-n$ ecuaciones de (2.8), que expresan la conservación de energía a lo largo de cada una de las mallas (y pseudo-mallas) de la red.

Por otra parte, el vector de caudales q debe satisfacer las n ecuaciones de conservación de masa (2.4), de manera que tenemos un conjunto de m ecuaciones con m incógnitas (los elementos del vector de caudales q).

Sin embargo, el sistema puede reducirse a $m-n$ ecuaciones si se tiene en cuenta que, dado un vector inicial de caudales q^0 que satisface la ecuación (2.4), cualquier otro vector que satisfaga la misma ecuación puede obtenerse considerando una corrección de caudal $\hat{q}_j$ para cada malla independiente j, de manera que el caudal de una línea será igual a su caudal inicial más las correcciones de cada una de las mallas a las que pertenece la línea, es decir:

$$q_l = q_l^0 + \sum_{j=1}^{m-n} \mu_{jl}\hat{q}_j \qquad l = 1, 2 \dots m \tag{2.17}$$

o bien en forma matricial:

$$q = q^0 + M_{31}^T \hat{q} \tag{2.18}$$

Este conjunto de ecuaciones se puede combinar con las ecuaciones de conservación de energía en las mallas (2.8), obteniendo el siguiente sistema no lineal de $m-n$ ecuaciones y el mismo número de incógnitas (las correcciones de caudal de las mallas $\hat{q}_j$):

$$z_i + \sum_{l=1}^{m} \mu_{il}\, \phi_l(q_l^0 + \sum_{j=1}^{m-n} \mu_{jl}\hat{q}_j) = 0, \qquad i = 1, 2 \dots m-n \tag{2.19}$$

donde $z_i = 0$ si la malla i es real, y $z_i = \hat{h}_{i_2} - \hat{h}_{i_1}$ si es una pseudo-malla que va del depósito i_1 al i_2 . Este sistema se resuelve a continuación mediante el método de

Newton-Raphson, lo que conduce a una secuencia de sistemas lineales de la forma:

$$z_i + \sum_{l=1}^{m} \left(\mu_{il}\phi_l^k + \mu_{il} d_l \sum_{j=1}^{m-n} \mu_{jl}\Delta\hat{q}_j \right) = 0, \qquad i = 1, 2 \ldots m-n \tag{2.20}$$

donde $\phi_l^k = \phi_l(q_l^k)$ es la pérdida de carga en la línea l en la iteración actual k, d_l es la derivada de $\phi_l(q_l)$ en la misma iteración, y $\Delta\hat{q}_j$ es el incremento de la corrección de caudal de la malla j.

Una vez obtenidas las correcciones de caudal de las mallas $\Delta\hat{q}_j$, los nuevos caudales de líneas se calculan mediante la expresión:

$$q_l^{k+1} = q_l^k + \sum_{j=1}^{m-n} \mu_{jl}\Delta\hat{q}_j \tag{2.21}$$

El método de mallas consiste en repetir sucesivamente la iteración dada por las ecuaciones (2.20) y (2.21). Expresándolo de forma matricial, la ecuación (2.20) sería:

$$M_{31}A_{10}\hat{h} + M_{31}\phi^k + M_{31}DM_{31}^T\Delta\hat{q} = 0$$

o bien:

$$(M_{31}DM_{31}^T)\Delta\hat{q} = -M_{31}(\phi^k + A_{10}\hat{h}) \tag{2.22}$$

mientras que la iteración (2.21) sería:

$$q^{k+1} = q^k + M_{31}^T\Delta\hat{q} \tag{2.23}$$

Obsérvese que en la formulación del método de mallas dada por (2.22) y (2.23) no aparecen las alturas piezométricas de los nudos de consumo $\tilde{h}$. Estas se pueden obtener a partir de las pérdidas de carga ϕ^k, haciendo un recorrido de la red que parta de los depósitos/embalses, cuya altura piezométrica $\hat{h}$ es conocida.

A modo de ejemplo, podemos considerar la red de la figura 2.3, con las mallas L1 y L2, y la pseudo-malla L3. El sistema de ecuaciones no lineales (2.19) es en este caso:

$$\left.\begin{aligned} \phi_1(q_1^0 + \hat{q}_1) + \phi_5(q_5^0 + \hat{q}_1) - \phi_2(q_2^0 - \hat{q}_1 + \hat{q}_2) = 0 \\ \phi_2(q_2^0 - \hat{q}_1 + \hat{q}_2) - \phi_6(q_5^0 - \hat{q}_2) - \phi_3(q_3^0 - \hat{q}_2 + \hat{q}_3) = 0 \\ h_6 - h_5 + \phi_3(q_3^0 - \hat{q}_2 + \hat{q}_3) - \phi_7(q_7^0 - \hat{q}_3) - \phi_4(q_4^0 - \hat{q}_3) = 0 \end{aligned}\right\}$$

mientras que las ecuaciones lineales (2.20) correspondientes al método de Newton-Raphson son:

$$\left.\begin{aligned} \phi_1^k + d_1\Delta\hat{q}_1 + \phi_5^k + d_5\Delta\hat{q}_1 - \phi_2^k + d_2(\Delta\hat{q}_1 - \Delta\hat{q}_2) = 0 \\ \phi_2^k + d_2(-\Delta\hat{q}_1 + \Delta\hat{q}_2) - \phi_6^k + d_6\Delta\hat{q}_2 - \phi_3^k + d_3(\Delta\hat{q}_2 - \Delta\hat{q}_3) = 0 \\ h_6 - h_5 + \phi_3^k + d_3(-\Delta\hat{q}_2 + \Delta\hat{q}_3) - \phi_7^k + d_7\Delta\hat{q}_3 - \phi_4^k + d_4\Delta\hat{q}_3 = 0 \end{aligned}\right\}$$

o bien, escrito en forma matricial:

$$\begin{bmatrix} d_1+d_2+d_5 & -d_2 & 0 \\ -d_2 & d_2+d_3+d_6 & -d_3 \\ 0 & -d_3 & d_3+d_4+d_7 \end{bmatrix} \begin{bmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \end{bmatrix} = \begin{bmatrix} -\phi_1^k - \phi_5^k + \phi_2^k \\ -\phi_2^k + \phi_6^k + \phi_3^k \\ -\phi_3^k + \phi_7^k + \phi_4^k - h_6 + h_5 \end{bmatrix} \quad (2.24)$$

donde la matriz del sistema es simétrica y definida positiva.

Algoritmo 2.3 Simulación de un paso de tiempo mediante el método de mallas

```
obtener un vector de caudales balanceado
mientras no convergencia hacer
  calcular matriz y vector del sistema lineal (2.22)
  resolver sistema lineal anterior, obteniendo Δq̂
  obtener nuevos caudales q^{k+1} mediante ec. (2.23)
  obtener nuevas alturas piezométricas h̃^{k+1}
  actualizar estado de válvulas, bombas y tuberías
fin mientras
```

El proceso correspondiente al método de mallas se describe en el algoritmo 2.3. Obsérvese que antes de comenzar el proceso iterativo es necesario obtener un vector de caudales balanceado, es decir, que cumpla la ecuación (2.6). Otro aspecto a tener en cuenta es que el algoritmo actualiza las alturas piezométricas en cada iteración. Esto es necesario únicamente para poder actualizar correctamente el estado de elementos que dependan de la presión o altura piezométrica de algún nudo, como pueden ser válvulas reductoras/sostenedoras de presión (VRP/VSP), u otros elementos gobernados por alguna regla de control.

Como se ha comentado en el apartado 2.5, la principal ventaja del método de mallas en la simulación de redes de distribución de agua es el hecho de que trabaja con una matriz de tamaño $(m-n) \times (m-n)$, que en la mayoría de los casos es mucho menor que la matriz $n \times n$ del método GGA. Sin embargo, eso no significa necesariamente que el sistema lineal se resuelva de manera más rápida, ya que dependerá principalmente del número de elementos distintos de cero en la matriz, lo cual está fuertemente relacionado con la manera en que se definan las mallas de la red, como se explica en el capítulo 4.

2.5.3. Equivalencia entre métodos

En [68, 69] se muestra que el método GGA, el de teoría lineal basado en gradientes y el método de mallas producen la misma secuencia de iteraciones de Newton (si empiezan de la misma aproximación inicial, y suponiendo aritmética exacta), dado que en los tres casos la iteración de Newton corresponde a la resolución del sistema

lineal (2.11):

$$\begin{bmatrix} D & A_{12} \\ A_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ \Delta \tilde{h} \end{bmatrix} = \begin{bmatrix} -\phi^k - A_{12}\tilde{h}^k - A_{10}\hat{h} \\ -A_{12}^T q^k + c \end{bmatrix}$$

Como hemos visto en el apartado (2.5.1), este sistema lineal se puede resolver despejando Δq en el primer bloque de ecuaciones y sustituyendo en el segundo bloque (mediante el complemento de Schur), lo que nos lleva al método GGA.

Otra posibilidad es premultiplicar ambos lados del sistema de la siguiente manera:

$$\begin{bmatrix} M_{31} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} D & A_{12} \\ A_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ h \end{bmatrix} = \begin{bmatrix} M_{31} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} -\phi^k - A_{12}\tilde{h}^k - A_{10}\hat{h} \\ -A_{12}^T q^k + c \end{bmatrix} \tag{2.25}$$

donde M_{31} es la matriz de incidencias mallas-líneas correspondiente a la elección de un determinado conjunto de mallas independientes sobre la red. Operando tenemos:

$$\begin{bmatrix} M_{31}D & M_{31}A_{12} \\ A_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ h \end{bmatrix} = \begin{bmatrix} M_{31}(-\phi^k - A_{12}\tilde{h}^k - A_{10}\hat{h}) \\ -A_{12}^T q^k + c \end{bmatrix} \tag{2.26}$$

Es sencillo verificar que $M_{31}A_{12} = 0$, con lo que el sistema anterior queda:

$$\begin{bmatrix} M_{31}D \\ A_{12}^T \end{bmatrix} \Delta q = \begin{bmatrix} M_{31}(-\phi^k - A_{10}\hat{h}) \\ -A_{12}^T q^k + c \end{bmatrix} \tag{2.27}$$

que corresponde a la iteración del método de teoría lineal basado en gradientes.

Por último, si partimos de una aproximación q^k que satisface el equilibrio de masas en los nudos de consumo, dado por (2.6), se verifica que el vector q^{k+1} resultante de (2.27) también satisface dicho equilibrio y, teniendo en cuenta (2.18), Δq se puede expresar como $\Delta q = M_{31}^T \Delta \hat{q}$, siendo $\Delta \hat{q}$ un vector de correcciones de caudal de mallas. Por tanto, el sistema anterior se puede transformar en:

$$\begin{bmatrix} M_{31}DM_{31}^T \\ A_{12}^T M_{31}^T \end{bmatrix} \Delta \hat{q} = \begin{bmatrix} M_{31}(-\phi^k - A_{10}\hat{h}) \\ 0 \end{bmatrix} \tag{2.28}$$

que es un sistema sobredeterminado, pero donde el segundo bloque de ecuaciones se verifica necesariamente, dado que $M_{31}A_{12} = 0$. Por tanto tenemos:

$$M_{31}DM_{31}^T \Delta \hat{q} = -M_{31}(\phi^k + A_{10}\hat{h})$$

que no es otra cosa que la iteración del método de mallas (2.22).

2.6. Simulación de la calidad del agua

Tras haber analizado la simulación hidráulica, en lo referente a caudales y presiones, a partir de este apartado pasamos a ocuparnos de la simulación de la calidad del agua potable.

Dicha calidad se ve afectada durante su recorrido desde la planta de tratamiento hasta el punto de consumo, de manera que, por ejemplo, la concentración de desinfectantes disminuye durante el tiempo que el agua permanece en la red, como consecuencia de reacciones químicas, con el consiguiente riesgo sanitario si no se garantiza una concentración mínima.

El objetivo de los modelos de calidad es determinar la evolución de los parámetros de calidad del agua desde la planta de tratamiento hasta el grifo del consumidor. Los modelos de calidad necesitan los resultados proporcionados por los modelos hidráulicos, en concreto, los caudales circulantes por las líneas de la red.

Los modelos de calidad se dividen en estáticos y dinámicos. El campo de aplicación de los modelos estáticos es bastante restringido, dado que son válidos únicamente en el supuesto de que las condiciones de funcionamiento de la red se mantengan constantes durante un periodo prolongado, y para sustancias no reactivas.

En esta tesis el interés se centra en los modelos dinámicos de calidad, los cuales se pueden clasificar espacialmente en *Eulerianos* y *Lagrangianos*, y temporalmente en conducidos por el tiempo (*time-driven*) y conducidos por eventos (*event-driven*) [60]. Los modelos Eulerianos dividen la red en una serie de volúmenes fijos y registran los cambios en los parámetros de calidad de estos volúmenes a medida que el agua fluye a través de ellos. Los modelos Lagrangianos, por su parte, dividen el agua en una serie de paquetes o volúmenes de tamaño variable, y siguen los cambios producidos en los mismos a medida que viajan a través de la red. Los modelos conducidos por el tiempo actualizan el estado de la red a intervalos fijos de tiempo. Los modelos conducidos por eventos llevan a cabo la actualización sólo cuando se produce un evento o cambio, como, por ejemplo, cuando un volumen de agua llega al final de la tubería y se mezcla con el agua procedente de otras tuberías.

2.7. Ecuaciones de equilibrio en la simulación de la calidad

El transporte de las sustancias a través de las tuberías está causado por el flujo del agua bajo la acción del gradiente de presiones (véase [60]). La difusión longitudinal es relativamente pequeña y suele despreciarse. Así pues, la conservación de masa para una sustancia reactiva en una línea l se formula a través de la siguiente ecuación:

$$\frac{\partial c_l(x,t)}{\partial t} = -u_l \frac{\partial c_l(x,t)}{\partial x} + R(c_l(x,t)), \tag{2.29}$$

donde $c_l(x,t)$ es la concentración de la sustancia considerada en la línea l, el punto x y el instante de tiempo t, u_l es la velocidad del flujo en la línea l, y $R(c_l)$ es la velocidad de reacción. Normalmente la reacción se modeliza por medio de una cinética de primer orden:

$$R(c_l) = \alpha c_l \tag{2.30}$$

Asumiendo que las sustancias que llegan a un nudo de consumo se mezclan completa e instantáneamente, se puede formular un conjunto adicional de ecuaciones

que expresan la conservación de masa en los nudos de consumo:

$$c_l(0,t) = \frac{\sum\limits_{j \in I_k} q_j c_j(L_j,t) + q_s c_s(t)}{\sum\limits_{j \in I_k} q_j + q_s}, \tag{2.31}$$

donde l es una línea cuyo flujo sale del nudo k; I_k es el conjunto de líneas cuyo flujo entra en el nudo k; L_j es la longitud de la línea j; q_s es el caudal de una fuente externa conectada al nudo k, en caso de que la haya; y c_s es la concentración del agua proveniente de la fuente externa.

Por último, se asume igualmente que el agua en los depósitos se mezcla completa e instantáneamente, con lo que la conservación de masa en los depósitos se expresa de la siguiente manera:

$$\frac{\partial(V_T(t)c_T(t))}{\partial t} = \sum_{i \in I_T} q_i c_i(L_i,t) - \sum_{j \in O_T} q_j c_T(t) + R(c_T(t)), \tag{2.32}$$

donde $V_T(t)$ es el volumen de agua que contiene el depósito T en el instante t; $c_T(t)$ es la concentración de la sustancia de interés en el depósito en el instante t; I_T es el conjunto de líneas que aportan agua al depósito y O_T es el conjunto de líneas que sustraen agua del mismo. La hipótesis de que la mezcla en los depósitos es completa e instantánea puede suavizarse a través de la utilización de modelos de mezcla.

Aplicando sobre la red entera el conjunto de ecuaciones (2.29) con las condiciones de frontera (2.31) y (2.32), se obtiene un conjunto de ecuaciones diferenciales-algebraicas cuya resolución analítica resulta intratable excepto para las redes más simples. Por lo tanto su resolución se lleva a cabo mediante métodos numéricos.

Los modelos de calidad del agua sirven también para determinar parámetros como el tiempo de permanencia del agua en la red, o el porcentaje de agua procedente de una determinada fuente de suministro. El tiempo de permanencia se puede modelizar sin más que interpretar la variable c de las ecuaciones anteriores como dicho tiempo, con lo que $R(c) = 1$. Para determinar el porcentaje de agua que llega a un nudo desde otro nudo de suministro, basta con interpretar la variable c como el factor de contribución de la fuente de referencia, no debiendo considerar en este caso término de reacción alguno. En esta situación el valor de la variable c en el punto de suministro de referencia será 100 %, actuando como condición de contorno.

2.8. Esquema general de la simulación de la calidad del agua

En los distintos métodos que se presentan en el siguiente apartado para la simulación de la calidad en redes de distribución de agua, se asume que se ha determinado previamente, por medio de una simulación hidráulica, las direcciones y velocidades del agua en cada tubería de la red. Se considera que las condiciones hidráulicas varían a lo largo del tiempo de acuerdo con una *simulación en periodo extendido*,

en la cual el tiempo se divide en una serie de pasos, no necesariamente de la misma duración, de manera que las variables hidráulicas permanecen constantes en cada uno de ellos. A estos pasos les llamaremos *intervalos hidráulicos* o *pasos de tiempo hidráulicos*, para diferenciarlos de los que aparecen en la simulación de calidad.

En esta situación, la simulación de calidad completa se llevará a cabo mediante simulaciones más cortas correspondientes a cada uno de los intervalos hidráulicos. Al inicio de un nuevo paso de tiempo hidráulico será necesario realizar algún tipo de ajuste, dependiente del método de calidad elegido, para acomodar los posibles cambios en sentidos y velocidades del flujo del agua.

El proceso es el que se muestra en el algoritmo 2.4, donde el bucle exterior es el correspondiente a los intervalos hidráulicos, mientras que el interior corresponde a los pasos de tiempo de calidad (de duración τ). El proceso requiere conocer las concentraciones iniciales $c_l(x, 0)$ a lo largo de las tuberías, si bien puede proporcionarse en su lugar las concentraciones iniciales c_j en los nudos de la red, obteniendo a partir de éstas las concentraciones en las tuberías mediante interpolación lineal, tal y como se detalla en [61].

Puede verse que al comenzar un nuevo paso hidráulico se leen las condiciones correspondientes, mientras que al acabar el mismo los resultados se escriben en fichero. Aparece también en el algoritmo 2.4 el ajuste de la simulación de calidad a las nuevas condiciones hidráulicas.

Algoritmo 2.4 Simulación de calidad completa.

```
t = 0
leer concentraciones iniciales
mientras t < t_fin hacer
    Leer condiciones hidráulicas y duración del intervalo hidráulico τ_h
    Ajustar simulación de calidad a las nuevas condiciones hidráulicas
    t_sig = t + τ_h
    mientras t < t_sig hacer
        Simulación de calidad de un paso
        t = mín(t + τ, t_sig)
    fin mientras
    Escribir resultados de la simulación
fin mientras
```

2.9. Métodos existentes para la simulación de la calidad del agua

A continuación se presentan cuatro métodos distintos para la simulación de la calidad del agua, haciendo hincapié en el *método de volúmenes discretos* (DVEM), que es el método en el que se centrará el trabajo de la tesis en cuanto a simulación de la calidad. El DVEM es el método utilizado por la versión 1.1e de Epanet, mientras

que el *método Lagrangiano conducido por el tiempo*, que también será descrito aquí, es el utilizado por la versión 2 del mismo programa.

Puede verse en [60] una comparación de los cuatro métodos aquí expuestos, aportando un contraste de los resultados con ellos obtenidos respecto a soluciones analíticas en redes simples, así como respecto a resultados experimentales. Se comparan así mismo los métodos desde el punto de vista del tiempo de ejecución y requerimientos de memoria. Los resultados de este estudio no muestran diferencias importantes en cuanto a la precisión de los resultados obtenidos con los distintos métodos, siendo todos ellos adecuados desde ese punto de vista. En cuanto al tiempo de CPU y requerimientos de memoria, los métodos Lagrangianos se muestran más eficientes para modelizar sustancias químicas. En cambio, para modelizar la edad del agua, los métodos Eulerianos son más eficientes desde el punto de vista de la memoria, si bien el método Lagrangiano conducido por tiempo (TDM) es el más rápido, mientras que el método Lagrangiano conducido por eventos (EDM) resulta ser poco eficiente.

2.9.1. Método de Volúmenes Discretos (*Discrete Volume Element Method*, DVEM)

El DVEM [61] es un método Euleriano, en el cual cada tubería l se divide en una serie de elementos de igual volumen v_l. Considerando un solo paso de tiempo de la simulación hidráulica, el avance del tiempo de la simulación de calidad se realiza mediante una sucesión de pasos de duración τ, en cada uno de los cuales se simula la reacción de la sustancia química en cada elemento de volumen de la tubería y, a continuación, su transporte al correspondiente elemento situado aguas abajo. El agua del elemento situado en el extremo aguas abajo de una tubería se transporta al nudo al cual está conectado, donde se produce una mezcla con las contribuciones aportadas por otras tuberías que desembocan en el mismo nudo. Igualmente, el elemento situado en el extremo aguas arriba de la tubería recibe agua, previamente mezclada, del nudo correspondiente.

El número de elementos de una tubería, denotado por η_l, debe elegirse de manera que cada elemento resultante tenga un volumen (aproximadamente) igual al del agua que sale (o entra) del elemento en el paso de tiempo τ. De esta manera, en cada paso de tiempo el agua de un elemento se reemplaza por completo por la del elemento situado aguas arriba. Teniendo en cuenta que el volumen mencionado es $v_l \approx q_l\tau$, se tiene que

$$\eta_l = \left\lfloor \frac{V_l}{q_l\tau} \right\rfloor, \qquad v_l = \frac{V_l}{\eta_l}, \tag{2.33}$$

donde V_l es el volumen total de la línea l y $\lfloor a \rfloor$ representa el mayor entero menor o igual que a. Además, el paso de tiempo τ debe ser menor que el *tiempo de permanencia* del agua en cualquier línea (tiempo necesario para que el agua de la línea sea completamente reemplazada por agua nueva), dado por V_l/q_l. Es decir:

$$\tau = \min_l \left(\frac{V_l}{q_l} \right). \tag{2.34}$$

Una vez que todas las líneas de la red han sido divididas en elementos de volumen, y se ha determinado la distribución inicial de masa en ellos, la propagación de masa a través de la red se realiza en cuatro fases, como se ilustra en la figura 2.4:

1. *Reacción cinética*: Se modeliza la reacción sufrida por la sustancia de interés de cada elemento de volumen durante el intervalo τ. Para una reacción de primer orden se tiene, de acuerdo con (2.30)

$$m'_{l,k} = m_{l,k} e^{\alpha\tau}, \qquad 1 \leq l \leq nl, 1 \leq k \leq \eta_l, \tag{2.35}$$

   donde $m_{l,k}$ es la masa original de la sustancia de interés en el elemento de volumen k de la línea l, y $m'_{l,k}$ es la masa resultante tras la reacción.

2. *Transporte a los nudos y mezcla*: El agua del último elemento de cada línea es transportada al nudo aguas abajo, donde se produce una mezcla y se calcula la concentración resultante:

$$\left.\begin{array}{c} M_j = \sum\limits_{l \in I_j} m'_{l,\eta_l} \\ V_j = \sum\limits_{l \in I_j} q_l \tau \\ c_j = \frac{M_j}{V_j} \end{array}\right\} \qquad 1 \leq j \leq n. \tag{2.36}$$

3. *Transporte a lo largo de las tuberías*: En cada tubería se produce un desplazamiento aguas abajo de la sustancia a lo largo de los elementos de volumen, de manera que

$$m'_{l,k+1} = m'_{l,k}, \qquad 1 \leq l \leq m,\ 1 \leq k \leq \eta_l - 1. \tag{2.37}$$

4. *Transporte fuera del nudo*: La masa de cada nudo sale al primer elemento de volumen de las tuberías que reciben agua del nudo

$$m'_{l,1} = c_{l_u} q_l \tau, \qquad 1 \leq l \leq m, \tag{2.38}$$

   donde l_u denota el índice del nudo situado inmediatamente aguas arriba de la línea l.

La implementación del método se muestra más detalladamente por medio del algoritmo 2.5. Tras inicializar a cero la masa y volumen de cada nudo j, el algoritmo hace un recorrido sobre todas las líneas, aplicando en cada una de ellas las fases 1, 2 y 3 presentadas anteriormente, de acuerdo con las fórmulas (2.35)-(2.37). A continuación se hace un recorrido sobre los nudos para calcular sus concentraciones, y, por último, se recorren las líneas de nuevo para llevar a cabo la fase 4, en la cual se actualiza la masa de los elementos situados aguas arriba de todas las líneas, de acuerdo con (2.38).

Esta secuencia se repite para cada paso de tiempo τ hasta que finaliza el intervalo de tiempo hidráulico actual. El último paso será en general de una duración inferior τ', para acomodar el tiempo transcurrido a la duración del intervalo hidráulico.

Algoritmo 2.5 Un paso de tiempo del método DVEM.

Entrada: Red hidráulica (n nudos, m líneas) con líneas divididas en elementos; valores de masa $m_{l,k}$ en todos los elementos de volumen en el tiempo inicial t; paso de tiempo τ.

Salida: Valores en $t + \tau$ de la masa $m_{l,k}$ en todos los elementos de volumen, y de concentración c_j en todos los nudos.

para todo nudo de consumo j **hacer**
  $M_j = 0; \qquad V_j = 0$
fin para
para todo línea l **hacer**
  para todo elemento k de la línea l **hacer**
    $m_{l,k} = k\, m_{l,k} e^{\alpha\tau}$
  fin para
  sea j el nudo aguas abajo de la línea l
  $M_j = M_j + m_{l,\eta_l}; \qquad V_j = V_j + q_l\tau$
  para $k = \eta_l - 1$ hasta 1 **hacer**
    $m_{l,k+1} = m_{l,k}$
  fin para
fin para
para todo nudo de consumo j **hacer**
  $c_j = \frac{M_j}{V_j}$
fin para
para todo línea l **hacer**
  sea j el nudo aguas arriba de la línea l
  $m_{l,1} = c_j q_l \tau$
fin para

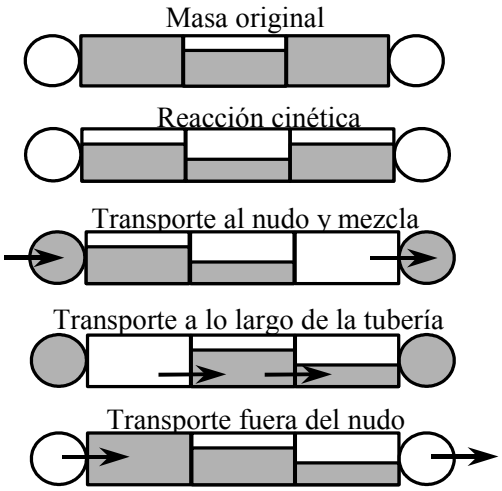

Figura 2.4: Fases del método DVEM.

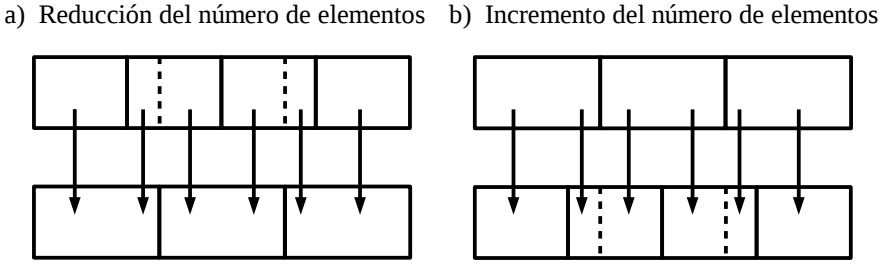

Figura 2.5: Redistribución de masas al comienzo de un paso de tiempo hidráulico.

En este caso, la masa transportada de un elemento a otro se deberá ajustar por la relación τ'/τ. El cálculo termina cuando se han cubierto todos los intervalos de tiempo.

Por otra parte, al inicio de cada paso de tiempo hidráulico se debe recalcular el paso de tiempo de calidad y la división en elementos de las líneas de la red, de acuerdo con (2.33) y (2.34). Los elementos de volumen de cada línea sufrirán en general un cambio de dimensiones debido a cambios en la velocidad y/o sentido del agua, por lo que habrá que trasladar las masas de la antigua segmentación de la línea a la nueva. En la figura 2.5 se representa este cambio distinguiéndose los dos casos posibles, según la nueva división establezca más o menos elementos que los que había. El proceso de traslación de masas es sencillo, pudiendo consultarse en [61].

Finalmente, existen algunas situaciones en las que el método presentado requiere

una cantidad excesiva de cómputo o de memoria, lo cual puede evitarse modificando el método para acomodar estos casos extremos [61]. Dichos casos son:

- Existencia de una tubería muy corta donde el agua circula a velocidad alta. El tiempo de permanencia de la tubería $V_l/q_l = L_l/u_l$ será muy pequeño, y en consecuencia también lo será τ, según (2.34). Esto provoca un elevado número de elementos de volumen en todas las tuberías, de acuerdo con (2.33), con la consiguiente demanda de memoria, al igual que un tiempo de cálculo excesivo, debido tanto al gran número de elementos a procesar, como al elevado número de pasos de tiempo a realizar. La solución pasaría por especificar un paso de tiempo mínimo $\tau_{\text{mín}}$, a costa de tolerar cierta pérdida de precisión debida a que las tuberías con tiempo de permanencia menor que $\tau_{\text{mín}}$ sufrirán un retraso en el transporte a través de ellas. Por otra parte, se puede asumir, sin pérdida de precisión, que las líneas que no son tuberías (bombas, válvulas) presentan un transporte inmediato, dado que su longitud es despreciable.
- Existencia de una tubería muy larga donde el agua circula a velocidad lenta. En este caso el número de elementos de volumen de la tubería será elevado, provocando incremento del tiempo de cálculo y de las necesidades de memoria. La solución consistiría en establecer un número máximo de elementos por tubería $\eta_{\text{máx}}$. De nuevo esto supone cierta pérdida de precisión, dado que el transporte en las tuberías que requieran un número de elementos mayor que $\eta_{\text{máx}}$ será más rápido de lo que debería ser.

2.9.2. Método Lagrangiano conducido por tiempo (*Time-Driven Method*, TDM)

Este método [46] hace un seguimiento de la concentración y tamaño de una serie de segmentos no solapados de agua que viajan a través de la red. A medida que avanza el tiempo, el tamaño del último segmento de una línea (el situado en el extremo aguas arriba) crece debido al agua que entra en él desde el nudo al que está conectado, mientras que se produce una pérdida de tamaño igual en el primer segmento de la línea debido al agua que sale del mismo. El tamaño del resto de elementos de la línea no cambia.

Para cada paso de tiempo de calidad, se determina la reacción que sufre la sustancia en cada segmento, se calcula la masa y volumen totales que entran en cada nudo, y se actualiza la posición de los segmentos. A continuación se calculan las nuevas concentraciones nodales y se crean nuevos segmentos al final de cada línea. Para evitar el crecimiento excesivo del número de segmentos, solo se crea un nuevo segmento cuando la concentración que tendría éste difiere más que una cierta tolerancia del último segmento de la línea. En caso contrario, lo que se hace es aumentar el tamaño de dicho último segmento. El proceso se repite hasta el fin del paso de tiempo hidráulico. Al principio del siguiente paso de tiempo hidráulico el único ajuste necesario es invertir el orden de los segmentos cuyo sentido de flujo cambie.

La precisión de este método depende de la elección del paso de tiempo de calidad, y de la tolerancia de concentración usada para limitar el crecimiento del número de

segmentos.

2.9.3. Método de Diferencias Finitas (*Finite Difference Method*, FDM)

El FDM es un método Euleriano que aproxima las derivadas en (2.29) por sus equivalentes en diferencias finitas a lo largo de una malla fija de puntos en el tiempo y espacio [15]. Existen diferentes posibilidades para realizar esta aproximación, entre las cuales cabe citar el esquema de Lax-Wendroff [65], el cual conduce a un algoritmo estable siempre que para toda línea l se satisfaga $0 < \alpha_l \leq 1$, donde $\alpha_l = u_l \frac{\Delta t}{\Delta x_l}$, siendo u_l la velocidad del agua en la línea l, Δx_l el espaciado entre los puntos de cálculo en la misma línea, y Δt el paso de tiempo de calidad. Adicionalmente, se debe incorporar la ecuación (2.32) correspondiente a los depósitos, para lo cual se puede usar una aproximación de diferencias progresivas en el tiempo.

El resultado final es una serie de ecuaciones algebraicas sobre la red completa, que pueden resolverse de manera explícita procediendo hacia delante en el tiempo y a lo largo de la longitud de las tuberías. Al inicio de cada paso de tiempo hidráulico, se debe recalcular el espaciado de los puntos de cálculo de cada línea de forma que α_l se aproxime a 1 tanto como sea posible. Las concentraciones en los nuevos puntos de cálculo se obtienen por medio de interpolación lineal de los puntos anteriores. La precisión de los resultados obtenidos por medio de este método depende del paso de tiempo Δt empleado.

2.9.4. Método Lagrangiano conducido por eventos (*Event-Driven Method*, EDM)

El EDM [11] es similar al TDM, excepto que la actualización del estado de la red no se lleva a cabo a intervalos de tiempo fijos, sino que se hace cada vez que ocurre un evento, entendiéndose como tal que el primer segmento de alguna línea desaparezca completamente al ser "engullido" por el nudo situado aguas abajo.

Este método debe mantener una lista ordenada de los tiempos de vida proyectados del primer segmento de cada línea. El siguiente evento que se producirá será causado por el segmento que esté en la cabeza de la lista, o sea, al que le quede un menor tiempo de vida. Cuando este evento salta, se llevan a cabo las siguientes acciones:

1. El evento se elimina de la lista y el tiempo de simulación se actualiza.
2. Se calcula la reacción que ha tenido lugar en los segmentos desde el último evento.
3. El segmento causante del evento es eliminado, y la concentración en el nudo situado aguas abajo del mismo se actualiza, al recibir el agua del segmento y mezclarse con la de los demás segmentos que aportan agua al nudo.
4. Si el cambio de concentración en el nudo anterior es mayor que una tolerancia determinada, se generan nuevos segmentos al principio de todas las líneas que reciben agua del nudo.

5. Se añade un nuevo elemento a la lista de eventos, correspondiente al segmento que ha reemplazado al segmento causante del evento. Se ajusta el tiempo de vida de todos los segmentos en la lista de eventos y se reordena la misma.

Este proceso se repite hasta el final del paso de tiempo hidráulico actual. Entonces se actualizan las posiciones de los segmentos y las concentraciones. Cuando comienza el siguiente paso de tiempo hidráulico se debe invertir el orden de los segmentos que experimenten un cambio en el sentido de su flujo. Tras ello se genera una nueva lista ordenada de eventos.

La precisión de este método depende únicamente de la tolerancia usada para limitar la generación de segmentos nuevos.

2.10. Conclusiones

En este capítulo se revisan distintos métodos para la simulación de redes de distribución de agua, tanto en lo referente a caudales y presiones (simulación hidráulica), como en el aspecto de la calidad del agua.

En el caso de la simulación hidráulica, se consideran modelos dinámicos no inerciales, y se presenta el proceso de simulación en periodo extendido como una sucesión enlazada de problemas estáticos. Se profundiza en dos de los métodos para resolver el problema hidráulico estático: el método del gradiente global (GGA), y el método de mallas, ambos de gran protagonismo en esta tesis. El método GGA es objeto de paralelización tanto en plataformas de memoria distribuida (capítulo 5), como de memoria compartida (capítulo 6). Por su parte, el método de mallas es considerado en el capítulo 4, donde se presentan contribuciones importantes para mejorar sus prestaciones, y se retoma de nuevo en el capítulo 6, donde se analiza su paralelización en sistemas de memoria compartida.

En el capítulo 5 se considera la presencia de fugas en la red, y la adaptación de las ecuaciones del modelo hidráulico para tenerlas en cuenta. Dado que las fugas sólo se consideran en ese capítulo, se ha preferido dejar el correspondiente contexto en el mismo capítulo.

En lo que respecta a la simulación de la calidad del agua, el capítulo se centra en modelos dinámicos, asumiéndose que se dispone de los valores de caudales, obtenidos mediante la simulación hidráulica correspondiente. De entre los métodos de simulación de la calidad, se profundiza especialmente en el método de volúmenes discretos (DVEM), implementado en la versión 1.1 de Epanet, y objeto de paralelización sobre sistemas de memoria distribuida en el capítulo 5 de esta tesis.

Capítulo 3

Herramientas de Computación de Altas Prestaciones

Se hace necesario en esta tesis, que toca las disciplinas de ingeniería hidráulica y computación de altas prestaciones, presentar cierto contexto sobre ambas. Tras considerar la ingeniería hidráulica, y más concretamente la simulación de redes de distribución de agua, en el capítulo anterior, entramos en este en la computación de altas prestaciones.

No se pretende entrar en profundidad en las arquitecturas paralelas ni en su programación, pero sí ofrecer el contexto suficiente para entender los capítulos 5 y 6. El lector interesado puede encontrar una gran cantidad de bibliografía sobre el tema donde ampliar información, como [3, 31, 14].

En el capítulo empezamos por hablar sobre la oportunidad y necesidad de la computación paralela o de altas prestaciones. A continuación se presentan las dos grandes categorías de computadores paralelos existentes en la actualidad: las plataformas de memoria compartida y las de memoria distribuida. Se mencionan los puntos fuertes y debilidades de cada una de ellas, y algunas ideas básicas sobre su programación. Finalmente, se describen las herramientas estándar de programación paralela OpenMP y MPI, dirigidas a computadores de memoria compartida y distribuida, respectivamente. Se trata de las herramientas con más aceptación en sus respectivas áreas, y ambas han sido utilizadas en esta tesis, como se describe en los capítulos 5 (MPI) y 6 (OpenMP).

3.1. Motivación de la computación paralela

Una característica común de prácticamente todos los computadores actuales es su capacidad de realizar múltiples operaciones simultáneamente. Esta capacidad es el

resultado de la utilización de distintas técnicas. Los procesadores *superescalares*, por ejemplo, habituales hoy en día en todo tipo de computadores, incluyendo portátiles y PCs, tienen varias unidades funcionales que pueden operar simultáneamente, de manera que se puede estar trayendo un dato de memoria, realizando una operación en coma flotante y evaluando una condición de salto todo a la vez. Por otra parte, muchos computadores actuales, o incluso *smartphones*, disponen de varios procesadores que pueden acceder a una memoria compartida, y cada vez más esos múltiples procesadores están integrados dentro de un solo chip. En otros casos, el computador tiene una arquitectura de memoria distribuida, donde cada uno de los procesadores dispone de su propia memoria, y los procesadores se comunican para realizar la transmisión de información entre ellos.

La capacidad de concurrencia, o paralelismo, de los computadores actuales ha venido motivada por la necesidad de extraer mejores prestaciones. Ciertamente, ha venido acompañada también de otros avances tecnológicos que han permitido, durante las últimas décadas, una reducción de tamaño de los componentes utilizados para construir los computadores, lo que ha hecho posible encajar cada vez más transistores en un chip. Esta reducción de tamaño ha permitido a su vez aumentar el ritmo al cual se completan instrucciones, y en consecuencia la frecuencia de reloj. Sin embargo, este enfoque tiene sus limitaciones principalmente en cuanto a consumo de energía y calor generado [14]. Además, el punto crítico es poder convertir los transistores en operaciones por segundo útiles. Aunque es posible fabricar dispositivos con un enorme número de transistores, el reto desde el punto de vista de arquitectura de computadores es poder usar esos transistores para conseguir más velocidad de computación. Para afrontar ese reto es necesario recurrir al paralelismo [31].

En este contexto, es vital que las aplicaciones *software* sean capaces de hacer un uso efectivo del paralelismo presente en el *hardware*. A pesar de avances importantes en la tecnología de compiladores, es necesario que el programador ayude mediante la descripción de la concurrencia contenida en los códigos de las aplicaciones.

3.2. Arquitecturas paralelas

Existen dos grandes categorías de computadores paralelos, que corresponden a dos maneras diferentes de organizar la interacción de los procesadores con la memoria: plataformas de *memoria compartida* (o con *espacio único de direcciones*) y plataformas de *memoria distribuida* (o de *paso de mensajes*).

Plataformas con memoria compartida. En este tipo de computadores existe un *espacio de direcciones de memoria común* a todos los procesadores. Los procesadores interaccionan mediante la modificación de datos almacenados en ese espacio de direcciones compartido.

Si bien todos los procesadores tienen garantizado el acceso a una memoria común, los tiempos de acceso a distintas posiciones de memoria pueden ser diferentes. Por ello, más que de memoria compartida se puede hablar de un espacio único de direcciones [3, 31]. Si el tiempo que le cuesta a un procesador acceder a cualquier posición

de memoria es el mismo, independientemente de la posición de que se trate, se dice que es una plataforma con *acceso a memoria uniforme*, (UMA, del inglés *uniform memory access*). Si por el contrario el tiempo de acceso a algunas posiciones es mayor que el de otras, se trata de una plataforma con *acceso a memoria no uniforme* (NUMA, de *non-uniform memory access*).

De hecho, la memoria puede estar físicamente distribuida entre los procesadores, pero lógicamente compartida. Es decir, un procesador puede acceder a la memoria de otros procesadores a través de la red, siendo ésta lo suficientemente rápida como para que, aunque el acceso a la memoria de otro procesador sea más lento que a la memoria local, el programador puede abstraerse de esa diferencia.

Puesto que la interacción entre procesadores se realiza a través de lecturas y escrituras en memoria, el tráfico entre los elementos de proceso y la memoria puede ser muy intenso. La consecuencia de esto es que los accesos a memoria y la gestión de los conflictos de acceso constituyen el cuello de botella que condiciona el rendimiento de este tipo de arquitecturas.

Durante las últimas décadas la velocidad de los procesadores ha crecido de forma mucho más rápida que la de las memorias, de manera que el problema es poder suministrar datos a los procesadores a un ritmo suficientemente rápido. Para intentar conseguirlo, en las máquinas actuales la memoria está organizada de una forma jerárquica, de acuerdo con los tiempos de acceso, teniendo normalmente varios niveles de memoria caché, de acceso muy rápido pero de pequeño tamaño, donde se replican datos de la memoria central para poder llevarlos al procesador más rápidamente. En la memoria central, además, es posible que haya direcciones cuyo acceso es más rápido que otras (arquitecturas NUMA). El objetivo de esta organización jerárquica es aprovechar al máximo el llamado *principio de localidad de referencias*, que consiste en que los programas referencian con mayor probabilidad aquellas posiciones de memoria que están próximas entre sí (localidad espacial) o que están próximas a posiciones de memoria ya referenciadas (localidad temporal).

Para obtener buenas prestaciones en este tipo de plataformas, el programador debe tener en cuenta a menudo esta jerarquía de memorias. Por ejemplo, intercambiar el orden de los bucles en un fragmento de código puede influir fuertemente en las prestaciones de un programa.

Dejando de lado los aspectos relativos a las prestaciones, la presencia de un espacio de memoria global hace que la programación en estas plataformas sea en principio más sencilla que en plataformas con memoria distribuida. Todas las interacciones de sólo-lectura son invisibles al programador, dado que se codifican de la misma manera que en un programa secuencial. Sin embargo, las interacciones de lectura/escritura son más difíciles de programar ya que se requiere la exclusión mutua para accesos concurrentes. Las herramientas para programación paralela en memoria compartida soportan por tanto diferentes mecanismos para la necesaria sincronización.

Plataformas de memoria distribuida. En este tipo de computadores paralelos (también llamados *multicomputadores*) cada procesador tiene su propio espacio exclusivo de direcciones de memoria. Las interacciones entre procesos ejecutándose en procesadores diferentes deben hacerse a través de mensajes, de ahí que también se

utilice el nombre de plataformas de paso de mensajes. Este intercambio de mensajes sirve tanto para transmitir datos como para sincronizar las acciones de los diferentes procesos.

Cada vez con mayor frecuencia, en este tipo de plataformas cada uno de los nodos de procesamiento es a su vez un multiprocesador con espacio único de direcciones.

El tiempo de ejecución de un programa paralelo en este tipo de plataformas depende principalmente de dos componentes: el correspondiente al procesamiento o cálculo propiamente, y el asociado a la comunicación de mensajes entre los procesos. Las características de la red de interconexión condicionan en gran medida este último componente. Los problemas de acceso a memoria presentes en el caso de las plataformas de memoria compartida se reducen en este caso a la memoria local de cada procesador. En cambio, es en la red de interconexión y su gestión donde aparecen ahora conflictos y dificultades. En concreto, los mensajes deben atravesar medios físicos (los enlaces de la red) para llegar a otros procesadores, para lo cual emplearán cierto tiempo. Es necesario también considerar el problema del encaminamiento de los mensajes, es decir, la elección de la ruta a seguir por los mensajes para llegar de un procesador origen a otro destino. En definitiva, las comunicaciones y su gestión juegan un papel decisivo en este tipo de plataformas, similar al que juega la gestión de memoria en el caso de las plataformas de memoria compartida.

Dado que las interacciones se llevan a cabo mediante mensajes, las herramientas de programación para este tipo de plataformas deben proporcionar operaciones de *envío* y *recepción* de un mensaje. Además, en estas operaciones se debe especificar cuál es el proceso al que se envía o del que se recibe, por lo que debe haber un mecanismo para asignar un identificador único a cada uno de los múltiples procesos de un programa paralelo. Este identificador debe ser accesible al programador por medio de alguna función. De la misma manera, la aplicación paralela necesitará también saber el número de procesos que la componen, para lo cual se utilizará otra función. Con esas cuatro operaciones básicas es posible escribir cualquier programa en una plataforma de paso de mensajes [31]. Sin embargo, las herramientas para programar en este tipo de entornos, como MPI (*Message Passing Interface*) proporcionan, además de estas operaciones, extensa funcionalidad de más alto nivel a través de diferentes funciones.

Es fácil emular una arquitectura de paso de mensajes sobre una plataforma con espacio único de direcciones. Sin embargo, hacer la emulación en el otro sentido requiere la existencia de alguna capa de *software* intermedia, dado que acceder a la memoria de otro nodo en una arquitectura de paso de mensajes requiere el envío y recepción de mensajes.

3.3. OpenMP

Existen distintas herramientas o APIs (*application programming interfaces* o interfaces de programación de aplicaciones) para la programación en memoria compartida, las cuales se basan en múltiples hilos de ejecución (o *threads*) dentro de un mismo programa. Un hilo constituye un flujo de instrucciones diferenciado, de mane-

ra que los diferentes hilos de un programa pueden ejecutarse de forma concurrente.

Algunas de estas herramientas, como Pthreads, ofrecen primitivas de bajo nivel y su uso se restringe fundamentalmente a programadores de sistemas, no de aplicaciones. Frente a esto, OpenMP ofrece construcciones (o directivas) de alto nivel que resultan adecuadas para la implementación eficiente de un amplio espectro de aplicaciones, sin exigir por parte del programador un manejo explícito de los mecanismos de manipulación de hilos. Por esta razón, OpenMP se ha convertido en el estándar *de facto* para la programación paralela en máquinas de memoria compartida.

La primera versión de OpenMP, para el lenguaje Fortran, apareció en 1997. Hoy en día OpenMP ofrece un entorno portable y estandarizado, para los lenguajes de programación C/C++ y Fortran. Se trata de una especificación definida de forma colaborativa entre partes interesadas de la industria de *hardware* y del *software*, así como instituciones académicas y gubernamentales. Las especificaciones de OpenMP surgen del ARB (*Architectural Review Board*), cuya misión es la de estandarizar APIs para la programación en memoria compartida. En `http://www.openmp.org` pueden consultarse los documentos y archivos oficiales de las especificaciones OpenMP.

Se trata de una API basada fundamentalmente en *directivas* de compilación, que el programador añade en los puntos adecuados de su aplicación. Además de las directivas de compilación, OpenMP se compone también de funciones de librería, y permite especificar diferentes opciones por medio de variables de entorno.

La programación mediante directivas de compilador facilita la *migración* de aplicaciones: el mismo código fuente de un programa OpenMP puede ser compilado deshabilitando OpenMP, con lo que las directivas son simplemente ignoradas, y se obtiene una versión secuencial del programa. Además, se puede realizar una *paralelización incremental* del código, considerando inicialmente sólo la realización en paralelo de algunas partes del programa, y añadiendo otras partes más adelante si se considera apropiado.

Las directivas de OpenMP en C/C++ se basan en las directivas de compilador `#pragma`, constando de un nombre y una serie de cláusulas.

```
#pragma omp directiva [lista de cláusulas]
```

3.3.1. Regiones paralelas

Un programa OpenMP se ejecuta de forma secuencial, con un solo hilo, hasta que encuentra una directiva `parallel`, momento en el cual se crea un grupo (*equipo*) de hilos. El hilo principal que encontró la directiva parallel se convierte en el hilo maestro, al que se le asigna el índice 0 como identificador. El resto de hilos del equipo se numeran consecutivamente hasta desde 1 hasta $p-1$, siendo p el número de hilos. El número de hilos que se crean puede especificarse mediante una cláusula de la directiva, establecerse mediante una invocación previa a la función `omp_set_num_threads`, o bien indicarse mediante una variable de entorno al lanzar el programa. Si no se especifica de ninguna de estas maneras, el número de hilos será igual al número de núcleos de proceso que tenga la máquina.

La sintaxis de la directiva `parallel` es la siguiente:

```
#pragma omp parallel [lista de cláusulas]
{
  bloque de instrucciones
}
```

A partir de la directiva `parallel`, cada uno de los hilos pasa a ejecutar las instrucciones contenidas en el bloque de código que sigue, que recibe el nombre de *región paralela*. Si este bloque de código contiene sólo una instrucción, se pueden omitir las llaves. Se trata de una aproximación SPMD (del inglés *single program multiple data*), es decir, los distintos flujos de instrucciones (en este caso hilos) ejecutan todos ellos el mismo código.

La directiva puede contener una serie de cláusulas, entre las cuales hay que destacar aquellas que especifican el *alcance* de las variables. En concreto, una variable puede ser *privada* a cada uno de los hilos del equipo, o *compartida* por todos ellos. La cláusula `private(lista de variables)` especifica que las variables de la lista son privadas o locales a cada hilo, es decir, cada hilo tiene su propia copia de la variable. Por el contrario, la cláusula `shared(lista de variables)` se utiliza para indicar que las variables son compartidas por todos los hilos. Cuando los hilos acceden a una variable compartida, todos ellos hacen referencia a la misma posición de memoria. Hay que tener cuidado al utilizar variables compartidas, para garantizar que el resultado es correcto.

Existen otras cláusulas relacionadas con el alcance de las variables. Entre ellas, una de las más útiles es la cláusula `reduction`, que se utiliza para indicar que las copias privadas de una determinada variable, correspondientes a cada uno de los hilos, se deben combinar mediante alguna operación, cuyo resultado debe asignarse a una única copia de la variable en el hilo maestro, una vez que la directiva acaba. La sintaxis de la cláusula es `reduction(operación:lista de variables)`, donde se especifica cuál es la operación que se debe aplicar para combinar las copias privadas. Además, las copias privadas se inicializan al elemento neutro de la operación correspondiente. Un ejemplo de esta sentencia sería:

```
int x=0;
#pragma omp parallel reduction(+:x)
x=x+1;
```

En este ejemplo, al llegar a la directiva `parallel` se crea un equipo de hilos, cada uno de los cuales pasa a ejecutar la instrucción que sigue a la directiva. Cada hilo dispone de una copia local de la variable `x`, inicializada a cero (elemento neutro de la suma). Esas copias se incrementan en uno, y al llegar al final de la región paralela se combinan (suman) entre sí y también con el valor que tenía la variable `x` antes de la directiva `parallel`. El resultado será en este caso igual al número de hilos del equipo, valor que se almacenará en la variable `x` justo al acabar la región paralela.

3.3.2. Directivas de reparto de trabajo

Dentro de una región paralela, donde tenemos varios hilos ejecutándose concurrentemente, es habitual que se quiera repartir el trabajo a realizar entre los hilos

disponibles. OpenMP proporciona dos directivas para especificar ese reparto de trabajo: `for` y `sections`.

Directiva for

La directiva `for`, fundamental en OpenMP, sirve para indicar que las iteraciones de un bucle deben repartirse entre los hilos del equipo actual. La forma general de la directiva `for` es:

```
#pragma omp for [lista de cláusulas]
for (...) /* instrucción for */
```

Lógicamente, la directiva `for` debe estar dentro de una región paralela definida mediante la directiva `parallel`, dado que hace falta tener un equipo de hilos entre los cuales repartir las iteraciones (también es posible combinar ambas directivas en una sola). Entre las claúsulas posibles de la directiva `for` se encuentran las ya mencionadas `private`, `shared` y `reduction`, las cuales tienen el mismo significado que en la directiva `parallel`.

Al final de la directiva `for` existe una barrera implícita, es decir, cada uno de los hilos debe esperar a que todos los demás acaben antes de continuar adelante. Esta barrera se puede eliminar utilizando la cláusula `nowait`.

Otra cláusula importante en esta directiva es `schedule`, mediante la cual se puede especificar la *planificación*, es decir, cómo queremos que se repartan las iteraciones del bucle entre los hilos. Se puede indicar uno de los siguientes cuatro tipos de planificación:

- `static`: en este tipo de planificación el espacio de iteraciones se divide en bloques o grupos de iteraciones consecutivas (en inglés *chunks*), con un tamaño fijo que se puede especificar, y los bloques se asignan a cada uno de los hilos de forma cíclica. Si no se especifica el tamaño de bloque, el espacio de iteraciones se divide de forma equitativa en tantos bloques como hilos, y se asigna un bloque a cada hilo.

- `dynamic`: al igual que en el caso `static`, el espacio de iteraciones se divide en bloques del tamaño especificado, pero en este caso la asignación de bloques a hilos se hace en tiempo de ejecución de forma dinámica, es decir, los bloques se asignan a hilos a medida que éstos quedan ociosos. Si no se especifica un tamaño de bloque, se utilizarán bloques de una sola iteración. Esta planificación puede dar lugar a un mejor equilibrio de carga entre hilos que la planificación `static`, en casos en que las distintas iteraciones del bucle tengan una duración dispar. Por otra parte, la asignación en tiempo de ejecución supone cierta sobrecarga.

- `guided`: es similar a `dynamic`, pero el tamaño de bloque va decreciendo a medida que lo hace el número de iteraciones por procesar. Se puede especificar el tamaño mínimo de bloque a utilizar.

- `runtime`: en realidad no se trata de una forma distinta de hacer la planificación, sino que indica simplemente que la planificación a utilizar (entre las tres posibles variantes anteriores), se especificará al lanzar el programa, mediante la variable de entorno `OMP_SCHEDULE`. De esta forma se pueden probar distintas planificaciones sin tener que modificar el código del programa.

Directiva sections

La directiva `sections` es útil en situaciones en que se tienen diferentes fragmentos de código que realizan tareas independientes entre sí. Mediante la directiva se especifica que esas tareas pueden realizarse concurrentemente, asignando cada tarea a un hilo diferente.

La forma general de la directiva `sections` es:

```
#pragma omp sections [lista de cláusulas]
{
   #pragma omp section
   {
      bloque de instrucciones
   }
   #pragma omp section
   {
      bloque de instrucciones
   }
   /* pueden seguir más secciones */
}
```

Al usar la directiva, cada bloque de instrucciones de una sección se asigna a un hilo, de forma que se pueden hacer varias/todas las secciones de forma concurrente. Al igual que en el caso de la directiva `for`, esta directiva debe estar contenida dentro de una directiva `parallel` (aunque puede también combinarse ambas directivas en una sola).

La lista de cláusulas incluye, entre otras, las correspondientes al alcance de variables (`private`, `shared`, `reduction`), con el significado comentado anteriormente. Como en el caso de la directiva `for`, existe una barrera implícita al final de la directiva `sections`, que se puede eliminar haciendo uso de la cláusula `nowait`.

3.4. MPI

Muchos computadores paralelos de las primeras generaciones pertenecían a la categoría de plataformas de memoria distribuida, dado que suponía una alternativa más económica que la memoria compartida. Eso hizo que los distintos fabricantes desarrollaran diferentes herramientas y librerías para el paso de mensajes, específicas para sus máquinas, e incompatibles con otras librerías y otro *hardware*. Obviamente, esto dificultaba la portabilidad de las aplicaciones, obligando a reestructurar el código cada vez que se necesitaba llevarlo a otra máquina.

El estándar MPI (*message passing interface* o interfaz de paso de mensajes) [52] surgió básicamente para solucionar este problema. Se trata de una especificación de una librería estándar de paso de mensajes que puede usarse para desarrollar aplicaciones portables en lenguaje C o Fortran. MPI se desarrolló por un grupo de investigadores tanto de las universidades como de la industria, y ha gozado de gran apoyo por parte de los fabricantes de *hardware*.

Una aplicación MPI se compone de un conjunto de procesos que se pueden comunicar entre sí por medio de llamadas a funciones de la librería. Es habitual que las aplicaciones MPI sigan un enfoque SPMD, es decir, que el código sea el mismo para todos los procesos. Sin embargo, MPI permite también un esquema MPMD (*multiple program multiple data*), donde hay procesos que corresponden a ficheros ejecutables distintos.

A diferencia de OpenMP, donde un único hilo comienza a ejecutar el programa, y sólo se crean otros hilos cuando se llega a una directiva `parallel`, aquí cada uno de los procesos empieza a ejecutar el programa desde el principio. Todos los procesos deben llamar a la función `MPI_Init` antes de llamar a cualquier otra función de MPI. Esta función realizará la inicialización del entorno. De la misma manera, todos los procesos deben llamar a la función `MPI_Finalize` cuando ya no precisen utilizar la librería, a fin de realizar tareas de limpieza del entorno.

3.4.1. Comunicadores

MPI usa el concepto de *comunicador* como un dominio de comunicación, dentro del cual un grupo de procesos puede comunicarse, sin que las comunicaciones de otros dominios distintos puedan interferir. Un comunicador tiene un grupo de procesos asociado, pero dos comunicadores pueden tener procesos en común (incluso podrían tener todos sus procesos en común). Cualquier operación de transferencia de datos debe especificar en qué comunicador debe realizarse. En general, todos los procesos de una misma aplicación pueden necesitar comunicarse entre sí, por lo que MPI define por defecto un comunicador (llamado `MPI_COMM_WORLD`), que engloba a todos los procesos.

Dentro de un comunicador, cada proceso recibe un índice identificador, que va desde 0 a $p-1$, siendo p el número de procesos del comunicador. La función `MPI_Comm_size` permite obtener el número de procesos de un comunicador, mientras que `MPI_Comm_rank` proporciona el índice que identifica al proceso que hace la llamada.

3.4.2. Envío y recepción de mensajes

La transmisión de mensajes entre dos procesos recibe el nombre de comunicación *punto a punto*. MPI proporciona dos funciones básicas para enviar y recibir un mensaje, que son `MPI_Send` y `MPI_Recv`, respectivamente.

`MPI_Send` permite el envío de una secuencia de elementos consecutivos de un mismo tipo. Para ello, los datos a ser enviados se especifican mediante la dirección de memoria donde está el primero de los elementos, el número de elementos y su tipo

de datos. De la misma manera, `MPI_Recv` considera la recepción de una secuencia de elementos consecutivos, para lo cual se debe especificar de forma análoga la dirección de memoria a partir de la cual se deben almacenar los elementos recibidos, el número de elementos *máximo* a recibir y su tipo de datos. En realidad MPI permite también el envío de elementos no consecutivos en memoria y/o de tipos diferentes en un mismo mensaje, como se explica en el apartado 3.4.4.

El proceso al cual se quiere enviar (en el caso de `MPI_Send`) o del que se quiere recibir (en `MPI_Recv`), se especifica mediante un comunicador y un índice que identifica al proceso dentro de ese comunicador. Además, cada mensaje tiene asociado un valor entero o etiqueta, que puede ser utilizada por la aplicación para diferenciar entre tipos de mensajes.

La función `MPI_Recv` recibirá un mensaje del proceso que se haya indicado y con la etiqueta que se haya especificado. Otros mensajes que puedan haber llegado pero que no correspondan a ese proceso y etiqueta se ignoran, quedándose a la espera de ser recibidos más adelante mediante otra invocación de `MPI_Recv` que encaje con ellos. También es posible indicar, en la invocación de `MPI_Recv`, que se desea recibir un mensaje de cualquier proceso y/o con cualquier etiqueta.

Tras recibir un mensaje mediante `MPI_Recv`, el proceso que lo ha hecho puede obtener información sobre él. En concreto, se puede averiguar el proceso remitente y la etiqueta del mensaje, lo que es útil si la llamada a `MPI_Recv` no ha especificado estos datos. También se puede obtener la longitud del mensaje (como se ha comentado, en la función `MPI_Recv` se especifica la longitud máxima, pero la cantidad de datos recibida puede ser menor).

3.4.3. Operaciones colectivas

MPI proporciona funciones para realizar operaciones colectivas que involucran a todos los procesos de un comunicador, cada uno de los cuales debe hacer la llamada a la función correspondiente, indicando en ella el comunicador correspondiente a la operación.

En muchas de las operaciones colectivas hay un proceso que tiene un papel destacado en la operación, y que recibe el nombre de proceso *raíz* (*root*). Como veremos, algunas operaciones implican la realización de cálculos además de comunicaciones, mientras que otras suponen sólo una sincronización entre los procesos, sin transmisión de datos.

Cualquier operación colectiva se podría sustituir por distintas operaciones punto a punto entre los procesos del comunicador. Sin embargo, las operaciones colectivas ofrecen una visión de más alto nivel, simplificando el desarrollo de aplicaciones al facilitar al programador operaciones comúnmente utilizadas en una gran variedad de algoritmos paralelos. Además, una misma operación colectiva puede estar implementada de manera diferente dependiendo de las características de la máquina paralela en particular (como la topología de la red de interconexión), con objeto de obtener prestaciones óptimas.

A continuación se describen las principales operaciones colectivas proporcionadas por MPI.

Barrera (`MPI_Barrier`): un proceso que realiza esta operación continúa adelante sólo cuando todos los demás procesos del comunicador han realizado también la operación. Es útil por ejemplo cuando se quiere obtener el tiempo empleado en cierta parte del programa. En ese caso, se realizará una medición de tiempo al inicio y al final del fragmento de código de interés, pero convendrá estar seguros de que todos los procesos han llegado al punto en cuestión (inicio o fin), para lo cual se hará uso de esta operación.

Difusión uno a todos (`MPI_Broadcast`): en esta operación uno de los procesos del comunicador (el llamado proceso raíz de la operación) envía a todos los demás procesos del comunicador una misma secuencia de datos. Inicialmente, sólo el proceso raíz tiene la secuencia de datos a comunicar. Al finalizar, cada proceso tiene su copia de los datos.

Reducción todos a uno (`MPI_Reduction`): se trata de la operación dual de la difusión uno a todos. En este caso cada proceso del comunicador parte de una secuencia de datos de longigud n. Los datos de todos los procesos se combinan mediante una operación asociativa, produciendo como resultado una secuencia de datos, también de longitud n, que se almacena únicamente en el proceso raíz. La reducción se puede usar para obtener la suma, producto, mínimo, máximo (entre otras operaciones) de un conjunto de números. El elemento en la posición i de la secuencia de datos resultante se obtiene mediante la combinación de los elementos en la posición i de las secuencias locales de todos los procesos. Puede verse la similitud que existe entre la operación de reducción de MPI (y de paso de mensajes en general) y la cláusula `reduction` de OpenMP.

Reducción a todos (`MPI_Allreduction`): es igual que la operación de reducción todos a uno, excepto que en este caso el resultado se obtiene en todos los procesos del comunicador. Por este motivo, en esta operación no hay ningún proceso raíz.

Reparto (`MPI_Scatter`, `MPI_Scatterv`): se trata de una operación parecida a la difusión uno a todos, con la diferencia de que en este caso el proceso raíz envía un mensaje *diferente* a cada uno de los procesos. Es decir, es una comunicación uno a todos personalizada. En la función `MPI_Scatter` el número de elementos a enviar a cada proceso es el mismo, mientras que la variante `MPI_Scatterv` permite que la longitud de cada mensaje sea diferente.

Recogida (`MPI_Gather`, `MPI_Gatherv`): constituye la operación dual del reparto. Cada proceso del comunicador dispone de una secuencia de elementos que se transmiten al proceso raíz, donde se concatenan formando una secuencia final, cuya longitud es la suma de las longitudes de las secuencias locales. En la función `MPI_Gather` todos los procesos aportan secuencias de la misma longitud, mientras que en `MPI_Scatterv` la longitud puede ser diferente para cada proceso.

Recogida en todos (`MPI_Allscatter`, `MPI_Allscatterv`): esta operación es muy similar a la recogida descrita anteriormente, con la diferencia de que en este

caso todos los procesos obtienen la secuencia resultante. No existe por tanto un proceso raíz.

Comunicación todos a todos (`MPI_Alltoall`, `MPI_Alltoallv`): en este caso cada proceso del comunicador envía un mensaje distinto a cada uno de los demás procesos. Dicho de otra manera, se trata de múltiples operaciones de reparto uno a todos, una por cada proceso, o bien múltiples operaciones de recogida todos a uno, una por cada proceso. No existe un proceso raíz en este caso. En la variante `MPI_Alltoall` de la operación, la cantidad de datos transmitida entre cualquier par de procesos es la misma, mientras que `MPI_Alltoallv` permite que esa cantidad sea distinta para cada par de procesos.

3.4.4. Tipos de datos derivados y empaquetamiento

MPI considera que los datos que se transmiten en un mensaje tienen la forma de una secuencia de elementos consecutivos del mismo tipo. Sin embargo, proporciona dos mecanismos mediante los cuales se puede transmitir un mensaje que contenga elementos no consecutivos en memoria y/o de tipos diferentes. Estos mecanismos son los *tipos de datos derivados*, por una parte, y el *empaquetamiento* de datos, por otra.

De forma simplificada, un tipo de datos derivado permite especificar una secuencia de datos compuesta por distintos bloques de elementos, de manera que cada bloque está compuesto por elementos consecutivos de un tipo determinado. Para cada bloque se puede especificar su tipo, su tamaño y su posición relativa en memoria respecto a una posición inicial. Por ejemplo, un tipo de datos puede describir una secuencia compuesta por 2 números enteros que comienzan en la posición relativa 0, seguidos de 1000 números reales de tipo `double` dispuestos a partir de la posición relativa 400.

Por otra parte, MPI permite empaquetar múltiples secuencias de elementos consecutivos, cada una con su propio tipo y longitud, en un único *buffer*, que luego se puede transmitir mediante un mensaje. Al recibir el mensaje, el *buffer* correspondiente se desempaqueta, colocando cada secuencia o bloque de elementos en su lugar.

3.5. Análisis de prestaciones

El principal objetivo de la computación paralela es aumentar las prestaciones. Por tanto, es importante estudiar las prestaciones de algoritmos paralelos, con objeto de poder comparar distintos algoritmos, comparar la versión secuencial con la paralela, o estudiar la influencia del número de procesadores y el tamaño del problema en las prestaciones obtenidas.

Tiempo de ejecución paralelo

Lógicamente el *tiempo de ejecución* es un parámetro fundamental. El tiempo de ejecución paralelo es el tiempo que transcurre desde que comienza un algoritmo paralelo hasta que el último procesador termina la ejecución. El tiempo de ejecución paralelo, que denotaremos como $t(n, p)$, no sólo depende del tamaño del problema n sino del número de procesadores utilizados p. El tiempo de ejecución del algoritmo secuencial se representará como $t(n)$.

Los algoritmos paralelos llevan asociada una sobrecarga debida a distintas causas. Además de realizar los cálculos esenciales (los que se realizarían también en un algoritmo secuencial que resolviera el mismo problema), un algoritmo paralelo puede consumir tiempo también en la comunicación o interacción entre procesador, en ocio o espera de algún procesador, o bien en cálculos extra, no realizados por el algoritmo secuencial.

De estas distintas causas de sobrecarga, la correspondiente a la interacción o comunicación de datos entre los distintos procesadores es a menudo la fuente más significativa de sobrecarga [31]. En el caso de plataformas de memoria distribuida, el tiempo necesario para comunicar un mensaje entre dos procesadores es la suma del tiempo para preparar la transferencia del mensaje más el tiempo para que el mensaje atraviese la red hasta el destino. Se suele expresar entonces el tiempo de comunicación de un mensaje como:

$$t_c = t_s + t_w m$$

donde t_s es el tiempo de establecimiento (o preparación de la comunicación), t_w es el tiempo de transferencia de un dato/palabra, y m es la longitud del mensaje a comunicar.

La expresión anterior para t_c implica que para minimizar el tiempo de comunicaciones de un algoritmo, interesa agrupar la información a comunicar en mensajes grandes, en vez de transmitir muchos mensajes pequeños, cada uno con su tiempo de establecimiento asociado.

En el caso de plataformas de memoria compartida, los costes de interacción entre los procesadores son mucho más difíciles de modelizar [31].

Speedup y eficiencia

Cuando se evalúa un algoritmo paralelo, a menudo interesa saber cuánta ganancia de velocidad se consigue respecto al algoritmo secuencial. El *speedup*, representado como S o $S(n, p)$, expresa esa ganancia, en términos relativos. En concreto, se define como el cociente entre el tiempo de ejecución del algoritmo secuencial entre el tiempo de ejecución paralelo, es decir:

$$S(n, p) = \frac{t(n)}{t(n, p)}$$

Normalmente el tiempo secuencial que se utiliza corresponde al algoritmo secuencial más rápido, dado que de esa forma se mide el beneficio real de la paralelización.

En teoría el *speedup* nunca puede superar el número de procesadores, p. En la práctica, se observa a veces un *speedup* mayor que p (fenómeno conocido como *speedup superlineal*). Esto puede deberse a características del *hardware* que ponen a la implementación secuencial en desventaja. Por ejemplo, los datos con los que trabaja un algoritmo pueden no caber en la caché de un sólo procesador, pero en el algoritmo paralelo cada procesador trabajará sólo con una parte de esos datos, que tal vez cabe en la caché.

La eficiencia, E o $E(n, p)$ expresa la fracción o porcentaje de tiempo durante el cual un procesador está haciendo cálculo útil del algoritmo. Se define como el cociente entre el *speedup* y el número de procesadores p:

$$E(n, p) = \frac{S(n, p)}{p}$$

En un caso ideal, el *speedup* sería p y por tanto la eficiencia sería 1. En la práctica el *speedup* es menor que p y la eficiencia es menor que 1.

3.6. Conclusiones

En este capítulo se han presentado conceptos básicos de la computación de altas prestaciones, y se han revisado las dos herramientas para la programación paralela utilizadas en esta tesis.

La razón de ser de la computación paralela es obtener mejores prestaciones en la resolución de una amplia gama de problemas. Casi todos los computadores actuales son, en mayor o menor medida, paralelos, pero pueden clasificarse las máquinas paralelas en dos grandes categorías, según sean de memoria compartida, con un único espacio de direcciones de memoria común a todos los procesadores, o de memoria distribuida, donde cada procesador tiene su propia memoria, no accesible por los demás.

Existen distintas herramientas para la programación de máquinas paralelas, entre las cuales las que gozan de mayor aceptación son OpenMP, para memoria compartida, y MPI, para memoria distribuida. Se trata de especificaciones de las que existen múltiples implementaciones. OpenMP utiliza un modelo de proceso multi-hilo, y se basa en directivas de compilador. MPI por su parte se basa en la ejecución concurrente de distintos procesos, que realizan operaciones de comunicación por medio de llamadas a funciones de una librería. Como se ha comentado, estas dos herramientas se han usado en esta tesis, como se describirá en los capítulos 5 (MPI) y 6 (OpenMP).

Finalmente, se presentan conceptos relacionados con la evaluación de prestaciones de algoritmos paralelos, que también serán de utilidad en los capítulos mencionados.

Capítulo 4

Aportaciones al método de mallas de simulación hidráulica

Este capítulo se centra en el desarrollo de mejoras para superar algunas de las dificultades existentes en la aplicación del método de mallas para la simulación hidráulica.

Básicamente se han desarrollado dos aportaciones. La primera de ellas consiste en el desarrollo de métodos para encontrar un conjunto adecuado de mallas independientes sobre la red. Existen muchos posibles conjuntos de mallas independientes, pero la elección de uno u otro afecta considerablemente a la dispersidad de la matriz correspondiente a los sistemas lineales a resolver, y consecuentemente, a las prestaciones del método. Algunos algoritmos encuentran un conjunto óptimo de mallas, pero a costa de emplear un tiempo de cálculo muy elevado. En este capítulo se presentarán métodos que son rápidos y al mismo tiempo consiguen obtener un conjunto de mallas que está bastante cerca del óptimo, en términos de la dispersidad de la matriz correspondiente.

La segunda aportación corresponde a la simulación de válvulas. La consideración de este tipo de elementos en la simulación mediante el método de mallas presenta algunas complicaciones. Una de ellas es el hecho de que cuando una válvula se cierra, se altera la topología de la red, pudiendo desaparecer algunas de las mallas de la misma. Ello obligaría a tener que redefinir el conjunto de mallas independientes y, lo que es peor, a tener que cambiar la correspondiente estructura de dispersidad de la matriz de los sistemas lineales, rehaciendo también la descomposición simbólica de la misma. La segunda dificultad existente se presenta en las válvulas reductoras/sostenedoras de presión (VRP/VSP), que llevan a introducir nuevas mallas o caminos entre el nudo de control de la válvula y algún nudo de altura fija, y además introducen no simetrías en la matriz del sistema lineal. En este capítulo se presentarán métodos para el tratamiento de válvulas que evitan redefinir el

conjunto de mallas independientes. Además, respecto a la no simetría introducida por las VRP/VSP, se consideran dos aproximaciones: la primera consiste en separar la parte no simétrica de la matriz, de manera que se pueda usar la descomposición de Cholesky para la parte simétrica; la segunda supone aproximar las ecuaciones correspondientes a la iteración de Newton para conseguir un sistema simétrico, de manera análoga a lo que realiza Epanet.

4.1. Algoritmo básico de simulación

En el capítulo 2 se presentaron algoritmos correspondientes a la simulación hidráulica. La simulación mediante el método de mallas se basa en los algoritmos 2.1 y 2.3 de aquel capítulo, el primero de los cuales describe el proceso de simulación hidráulica en periodo extendido como una secuencia encadenada de problemas hidráulicos estáticos, mientras que el segundo corresponde al método de mallas para la solución de cada problema estático. El proceso completo resultante se presenta en el algoritmo 4.1.

Algoritmo 4.1 Algoritmo de simulación hidráulica mediante el método de mallas.

```
1: leer datos de la red desde un fichero de entrada
2: selecionar mallas, hacer reordenación y descomp. simbólica
3: mientras t < t_final hacer
4:    establecer demandas, controles en válvulas y bombas
5:    obtener un vector de caudales balanceado
6:    mientras no convergencia hacer
7:       actualizar el sistema lineal
8:       resolver sistema lineal, obteniendo caudales correctores de mallas (solver
         lineal)
9:       obtener nuevos caudales de líneas
10:      obtener nuevas alturas piezométricas
11:      actualizar estado de válvulas, bombas y tuberías
12:   fin mientras
13:   paso al siguiente instante de tiempo
14: fin mientras
```

En esta tesis se ha hecho una implementación completa del método, partiendo del código de Epanet (en su versión 2.00.12), y realizando las modificaciones correspondientes a cada una de las tareas del algoritmo.

Las aportaciones presentadas en este capítulo afectan, por una parte, al proceso de selección de mallas, y por otra a la construcción inicial, actualización y resolución del sistema sistema lineal.

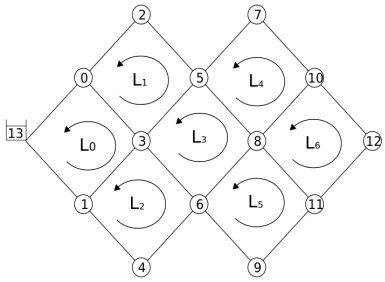

Figura 4.1: Red de ejemplo, con un conjunto de mallas independientes.

4.2. Escogiendo un conjunto de mallas

Como se ha mencionado, la elección del conjunto de mallas de una red puede tener un gran impacto en la cantidad de elementos no nulos de la matriz del sistema, y por tanto en la eficiencia del método de mallas. Más concretamente, el sistema de ecuaciones (2.20), correspondiente al método de mallas:

$$z_i + \sum_{l=1}^{m} \left(\mu_{il}\phi_l^k + \mu_{il} d_l \sum_{j=1}^{m-n} \mu_{jl} \Delta \hat{q}_j \right) = 0, \qquad i = 1, 2 \ldots m-n$$

supone que cada elemento a_{ij} de la matriz del sistema viene dado por:

$$a_{ij} = \sum_{l=1}^{m} \sum_{j=1}^{m-n} \mu_{il}\mu_{jl} d_l$$

es decir, a_{ij} es igual a la suma de $\pm d_l$ de aquellas líneas l que son comunes a las mallas i, j. Si ambas mallas no tienen líneas en común, el elemento será cero.

Esto significa que el número de elementos no nulos de la matriz del sistema viene determinado por el número de parejas de mallas que tienen alguna línea en común, por lo que es deseable que el solapamiento entre las mallas del conjunto elegido sea lo menor posible.

Considérese por ejemplo la red de la figura 4.1, con $n = 13$ uniones, $m = 20$ líneas, y un número de mallas independientes de $m - n = 7$. Si el conjunto de mallas seleccionadas fuera el que se muestra en la figura, el patrón de dispersidad de la matriz correspondiente sería el que se puede ver en la figura 4.2 (sólo se muestra la parte triangular superior, dado que la matriz es simétrica). Sin embargo, otros posibles conjuntos de mallas conducirían a una matriz completamente densa, como veremos a continuación.

Un método usado comúnmente para seleccionar el conjunto de mallas independientes empieza con la obtención de un árbol de expansión de la red. Una vez formado éste, cualquier otra línea que se añada al árbol produce la aparición de una malla, que se conoce como un *ciclo/malla fundamental*. Para obtener esa malla, se parte

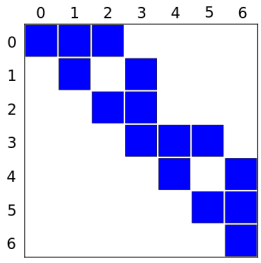

Figura 4.2: Patrón de dispersidad de la matriz correspondiente a las mallas definidas en la red de ejemplo.

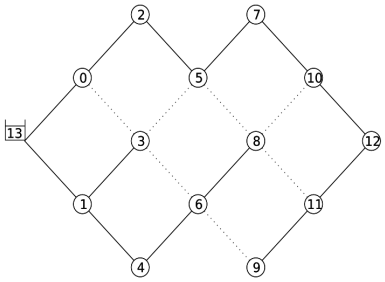

Figura 4.3: Un posible árbol de expansión para la red de ejemplo.

de cada uno de los extremos de la línea añadida, recorriendo el árbol hacia la raíz, hasta que los dos caminos se juntan. Por ejemplo, las líneas de trazo continuo de la figura 4.3 corresponden a un árbol de expansión. Se se añade al árbol la línea 5-8, la malla formada viene dada por la propia línea 5-8, junto con los caminos 5-2-0-13 y 8-6-4-1-13. El conjunto de mallas fundamentales del árbol de expansión constituye un conjunto de mallas independientes. Diversos artículos utilizan esta aproximación [24, 2, 10, 70], a la que nos referiremos en esta tesis como el método **m1** para la selección del conjunto de mallas.

Aunque este método es bastante simple, presenta la desventaja de que generalmente produce una matriz que no es especialmente dispersa. En nuestro ejemplo, cada malla resultante del árbol de expansión tiene al menos una línea en común con cada una de las demás mallas, por lo que la matriz producida es completamente densa. Si bien se pueden encontrar otros árboles de expansión que sean más favorables, no existe ninguno que produzca el conjunto de mallas presentado en la figura 4.1. Esto se puede apreciar si se tiene en cuenta que, para obtener la malla central L_3, el árbol de expansión debe contener exactamente 3 de sus 4 líneas. Sin pérdida de generalidad, podemos suponer que la línea que falta es la que conecta los nudos 5 y 8. Si es así, es fácil ver que el árbol de expansión no puede tener más de dos líneas de la malla L_4, lo que significa que L_4 no es una malla fundamental del árbol de

expansión.

En un contexto más general de teoría de grafos, [44] y [43] estudian el problema de encontar una *Base de Ciclos Mínima* (BCM) de un grafo. En esos trabajos, el conjunto de todos los posibles ciclos de un grafo se ve como un espacio vectorial, y una *base de ciclos* se define como un conjunto de ciclos que forman una base de ese espacio vectorial. Una base de ciclos mínima de un grafo ponderado se define entonces como una base de ciclos cuya suma, para todas las mallas, de la suma de pesos de las líneas de la malla, es mínimo. En nuestro caso, dado que trabajamos con grafos no ponderados, una base de cíclos mínima sería una donde la suma del número de líneas de cada malla es mínima.

En el caso de las mallas mostradas en la figura 4.1, cada malla consta de 4 líneas, por lo que la suma de líneas es 28. Si consideramos las mallas fundamentales resultantes de la figura 4.3, la suma resulta ser 56. Obsérvese que las líneas que son comunes a dos o más mallas se cuentan varias veces, lo que implica que una base de ciclos mínima tendrá poco solapamiento entre mallas, y producirá por lo tanto una matriz bastante dispersa.

Teniendo en cuenta lo anterior, en esta tesis se ha desarrollado una versión simplificada del algoritmo presentado en [44] para el cálculo de una base de cíclos mínima. Aunque los resultados, que se encuentran en el apartado 4.8, son muy buenos en términos de la dispersidad de la matriz, el problema se encuentra en el alto coste computacional del algoritmo, en términos tanto de tiempo de ejecución como de memoria. En esta tesis nos referiremos a este método como **m2**. Una aproximación similar se emplea en [16, 17], usando un algoritmo basado en [20], y también se describe el mismo problema del alto coste computacional.

Con objeto de superar los problemas de los dos métodos mencionados, en esta tesis se presentan dos aproximaciones diferentes para la definición de un conjunto de mallas independientes.

El primero de los métodos propuestos (al que llamaremos **m3**), comienza construyendo un árbol de expansión y obteniendo las mallas fundamentales, de la misma forma que se hace en **m1**. A continuación se procede a simplificar las mallas mediante combinaciones de las mismas. Más concretamente, al combinar dos mallas el resultado es una malla que contiene las líneas que están en alguna de las dos mallas originales, pero no en ambas. Por ejemplo, en la figura 4.3, la malla 5-2-0-13-1-3-5 podría ser simplificada combinándola con la malla 0-13-1-3-0, produciendo como resultado la malla 5-2-0-3-5. Antes de combinar las mallas, sin embargo, se ordenan de acuerdo con su profundidad en el árbol, de menos profundidad a más profundidad.

El proceso de simplificación se describe en el algoritmo 4.2, donde cada malla l_i se intenta reducir mediante su combinación únicamente con las mallas anteriores $(l_1 \dots l_{i-1})$. Nótese que el orden en el cual se consideran las mallas $l_1 \dots l_{i-1}$ para su posible combinación con la malla l_i es importante. A este respecto, el algoritmo 4.2 sigue una estrategia voraz, en la cual los primeros candidatos considerados son aquellos que producirían una malla más corta si se combinaran con l_i. Más concretamente, se construye una lista de las mallas entre $l_1 \dots l_{i-1}$ que solapan con la malla l_i, y se ordena la lista ascendentemente de acuerdo con la longitud de la malla resultante de la combinación con l_i. A continuación cada una de las mallas en la

Algoritmo 4.2 Algoritmo de simplificación de mallas para el método m3.

Entrada: L, lista de mallas
Salida: L', lista de mallas simplificadas

```
L' ← ∅
para todo malla l_i ∈ L hacer
  P ← ∅
  para todo malla l_j ∈ L, donde j < i hacer
    si l_j solapa con l_i entonces
      n_j ← longitud de la malla resultante de combinar l_i con l_j
      insertar (l_j, n_j) en P, ordenado ascendentemente por n_j
    fin si
  fin para
  l'_i ← l_i
  para todo par (l_j, n_j) ∈ P hacer
    c ← malla resultante de combinar l'_i con l_j
    si longitud de c < longitud de l'_i entonces
      l'_i ← c
    fin si
  fin para
  insertar l'_i en L'
fin para
```

lista se combina con l_i, descartando el resultado de una combinación si no reduce la longitud de la malla. Finalmente, la nueva malla reducida l'_i se inserta en el nuevo conjunto de mallas L'.

El segundo método propuesto (que llamaremos **m4**) se describe en el algoritmo 4.3. Básicamente, realiza un recorrido en anchura del grafo de la red G, partiendo de un nudo dado u. Durante este recorrido se va construyendo un grafo G' que contiene las líneas y nudos de la red que ya han sido visitados. Cuando en el recorrido aparece una nueva línea $(i, j) \notin G'$, que conecta el nudo actual i con otro nodo j ya visitado, se añade una nueva malla al conjunto de mallas L. Esa nueva malla estará compuesta por la línea (i, j) y el camino más corto en G' entre los nudos i y j, donde "camino más corto" significa un camino con mínimo número de líneas. Nótese que la línea (i, j) se añade a continuación a G', por lo que puede ser usada para las siguientes mallas.

A menudo en la literatura, cada malla del conjunto de mallas independientes se identifica mediante una *cuerda* correspondiente, es decir, mediante una línea que no pertenece a un árbol de expansión definido previamente, de manera que el caudal corrector de la malla es igual al caudal de esa línea. Sin embargo, eso sólo se puede hacer si se usa el método **m1** de definición de mallas, es decir, un método que imponga la restricción de que el conjunto de mallas debe corresponder a las mallas fundamentales de un árbol de expansión. Por el contrario, los métodos **m2-m4** no imponen esa restricción, por lo que pueden encontrar un mejor conjunto de mallas, produciendo una matriz más dispersa, como se ilustra en el ejemplo de la figura 4.1.

Algoritmo 4.3 Algoritmo del método m4 de definición de mallas

Entrada: G, grafo de la red; u, nudo inicial para la exploración.
Salida: L, lista de mallas

```
L ← ∅
G' ← ∅
S ← {u}
mientras S ≠ ∅ hacer
  i ← extraer primer elemento de S
  para todo nudo j tal que la línea (i, j) ∈ G hacer
    si j ∉ G' entonces
      añadir nudo j y línea (i, j) al grafo G'
      insertar j como el último elemento de S
    si no si línea (i, j) ∉ G' entonces
      p ← camino más corto de i a j en G'
      c ← {líneas de p} ∪ {(i, j)}
      insertar c en L
      añadir línea (i, j) a G'
    fin si
  fin para
fin mientras
```

4.3. Enfoque para modelizar las válvulas de control

Las siguientes dos secciones tratan de elementos hidráulicos que pueden cambiar de estado, tales como válvulas de retención, válvulas de control del caudal (VCQ), válvulas reductoras de presión (VRP) y válvulas sostenedoras de presión (VSP). Estos elementos pueden estar en diferentes estados dependiendo de condiciones hidráulicas que son desconocidas a priori, dado que deberían obtenerse mediante simulación. Sin embargo, para la simulación es necesario conocer el estado de los elementos. Esto representa un importante reto en la simulación de redes de distribución de agua.

Epanet [59] utiliza un método en el cual los estados de las válvulas se establecen al principio del proceso iterativo, y se comprueban durante el mismo, ajustándose de acuerdo con reglas heurísticas si es necesario. No hay garantía de que este método sea capaz de encontrar el estado correcto de las válvulas en todos los casos (véase por ejemplo [64]), pero funciona bien en la práctica y es un método ampliamente aceptado en la comunidad de simulación hidráulica. Existen otros métodos más rigurosos en los cuales el problema se formula como la minimización de funciones de *contenido* o *co-contenido* ("content" o "co-content" en inglés) sujetas a restricciones de desigualdad [22, 21, 23, 55]. Estos métodos sortean las dificultades de un método heurístico, aunque son más complejos y pueden por tanto necesitar mayor tiempo de cálculo.

En este capítulo se asume que el estado operacional de las válvulas de la red se va a determinar mediante un método similar al implementado en Epanet. Sin embargo,

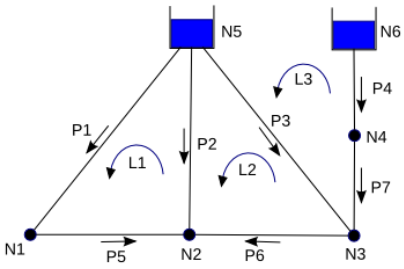

Figura 4.4: Red sencilla.

incluso en ese contexto, la presencia de válvulas de control afecta a la formulación del método de mallas presentada en el capítulo 2. Más concretamente, [39] usa un enfoque para incluir VRP en el cual el conjunto de mallas independientes cambia en función del estado de las válvulas. Otros autores, como [10], proponen una redefinición parcial del conjunto de mallas, mediante la modificación de un árbol de expansión que es la base para la determinación de las mallas. El problema de estos enfoques es la necesidad de redefinir el conjunto de mallas, lo que implica a su vez introducir cambios en el patrón de dispersidad de la matriz del sistema. Esto es importante porque los sistemas lineales que surgen en análisis de redes de distribución de agua se resuelven generalmente mediante un método directo, realizándose una descomposición simbólica al principio de la simulación para determinar el patrón de dispersidad de la matriz factorizada. Si la estructura de la matriz cambia, se tendría que repetir, o al menos actualizar, la descomposición simbólica, lo que provocaría un aumento del tiempo de cálculo.

En este capítulo se presentan métodos para tratar las válvulas de control sin tener que cambiar el conjunto de mallas independientes cuando hay un cambio de estado de una válvula.

4.4. Modelización de líneas temporalmente cerradas

Existen distintos tipos de elementos de control, como válvulas y bombas, que pueden cerrarse ante determinadas condiciones de simulación. Cuando esto ocurre, la topología de la red cambia, lo que podría hacer necesario redefinir el conjunto de mallas independientes.

Por ejemplo, consideremos la red de la figura 4.4, con 4 uniones (N1-N4), 2 depósitos (N5-N6), 7 tuberías (P1-P7), y las mallas independientes L1, L2 y L3. Supongamos que la tubería 2 está equipada con una válvula de retención, y que en un determinado momento la válvula se cierra. Con ello las mallas L1 y L2 dejan de serlo, y deberían ser sustituidas por una malla resultante de la combinación de ambas, que comprendería las tuberías P1-P5-P6-P3.

En esta tesis se proponen dos formulaciones alternativas para modelizar el cierre de líneas, que evitan tener que redefinir el conjunto de mallas independientes. Además, se mostrará la relación que existen entre ambas formulaciones.

4.4.1. Modelización como tuberías de gran resistencia

La primera alternativa para modelizar una válvula o bomba cerrada, consiste en sustituirla por una tubería de gran resistencia.

Continuando con el ejemplo de la figura 4.4, el sistema de ecuaciones lineales a resolver en cada iteración del método de Newton-Raphson, utilizando la formulación del método de mallas dada por la ec. (2.22), es:

$$\begin{bmatrix} d_1+d_2+d_5 & -d_2 & 0 \\ -d_2 & d_2+d_3+d_6 & -d_3 \\ 0 & -d_3 & d_3+d_4+d_7 \end{bmatrix} \begin{bmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \end{bmatrix} = \begin{bmatrix} -\phi_1^k+\phi_2^k-\phi_5^k \\ -\phi_2^k+\phi_3^k+\phi_6^k \\ -\phi_3^k+\phi_4^k+\phi_7^k-h_6+h_5 \end{bmatrix} \tag{4.1}$$

donde el significado de los distintos símbolos ha sido presentado en el capítulo 2.

La tubería cerrada P2 se modelizaría simplemente mediante un valor grande para d_2 (por ejemplo $\alpha = 10^8$), que correspondería a una tubería con gran resistencia con una pérdida de carga dada por $\phi_2^k = \alpha q_2^k$. Es decir:

$$\begin{bmatrix} d_1+d_5+\alpha & -\alpha & 0 \\ -\alpha & d_3+d_6+\alpha & -d_3 \\ 0 & -d_3 & d_3+d_4+d_7 \end{bmatrix} \begin{bmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \end{bmatrix} = \begin{bmatrix} -\phi_1^k+\phi_2^k-\phi_5^k \\ -\phi_2^k+\phi_3^k+\phi_6^k \\ -\phi_3^k+\phi_4^k+\phi_7^k-h_6+h_5 \end{bmatrix} \tag{4.2}$$

Se puede argumentar que esta forma de proceder introduce números muy grandes en la matriz, lo que provocaría que el sistema de ecuaciones esté mal condicionado, y por tanto cabría esperar errores de redondeo importantes.

Se ha comprobado, sin embargo, que este enfoque funciona bien en la práctica. Además, el problema de la aparición de números grandes en la matriz es similar al que sucede en Epanet con tuberías que tienen una resistencia muy pequeña y/o un caudal muy pequeño.

4.4.2. Modelización mediante ecuación adicional

Proponemos otro enfoque para tratar con una línea temporalmente cerrada. De nuevo nos referimos al ejemplo de la figura 4.4, donde la tubería 2 ha sido cerrada, de manera que la pérdida de carga entre los extremos de la misma no está relacionada con el caudal a través de la misma (que es cero). En ese caso no debería usarse

d_2, sino introducir ϕ_2 como una nueva variable. Introducimos además una nueva ecuación, que establece que el caudal a través de la tubería 2 es cero, es decir:

$$q_2^k - \Delta\hat{q}_1 + \Delta\hat{q}_2 = 0 \tag{4.3}$$

donde q_2^k es la aproximación, al inicio de la iteración k, del caudal de la línea 2.

Teniendo esto en cuenta, el sistema original se transforma de la siguiente manera:

$$\begin{bmatrix} d_1+d_5 & 0 & 0 & -1 \\ 0 & d_3+d_6 & -d_3 & 1 \\ 0 & -d_3 & d_3+d_4+d_7 & 0 \\ -1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \\ \phi_2 \end{bmatrix} = \begin{bmatrix} -\phi_1^k - \phi_5^k \\ \phi_3^k + \phi_6^k \\ -\phi_3^k + \phi_4^k + \phi_7^k - h_6 + h_5 \\ -q_2^k \end{bmatrix} \tag{4.4}$$

A continuación generalizamos la metodología propuesta. Si se considera que puede haber un número cualquiera de líneas cerradas, la ecuación (4.4) pasa a tener la forma:

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta\hat{q} \\ \hat{\phi} \end{bmatrix} = \begin{bmatrix} b \\ \hat{b} \end{bmatrix} \tag{4.5}$$

donde G_{11} es la misma matriz que la del sistema original, cambiando únicamente el valor de los coeficientes afectados por las líneas cerradas (como si esos elementos hubieran sido reemplazados por tuberías de resistencia nula). En particular, no se añaden nuevos elementos no nulos a la matriz. G_{12} es una matriz de incidencias, cuyos elementos pueden valer 0 o ± 1; $\hat{\phi}$ es el vector de pérdidas de carga para las líneas cerradas, y $\hat{b}$ son los elementos añadidos al vector de términos independientes de la ecuación.

Esta formulación se deriva también en [22] siguiendo un enfoque diferente, en el cual la simulación hidráulica en régimen permanente se trata como la minimización de la función de contenido con restricciones de desigualdad de caudales, y las pérdidas de carga de los líneas cerradas se interpretan como multiplicadores de Lagrange. En esta tesis se llega a la expresión (4.5) sin la necesidad de considerar un problema de optimización, que puede ser más complicado computacionalmente. En su lugar, usamos un esquema iterativo de Newton-Raphson, junto con un método para encontrar el estado correcto (cerrado o no) de los elementos de control (como puede ser el método heurístico de Epanet).

Queda considerar cómo resolver el sistema lineal 4.5. Es fácil ver que la matriz de este sistema es indefinida, por lo que no se puede calcular su descomposición de Cholesky. Se podría utilizar una descomposición del tipo LDL^T, o bien recurrir a una solución por partes, usando la descomposición de Schur, como se comenta a continuación.

El sistema (4.5) se puede dividir en dos ecuaciones:

$$G_{11}\Delta\hat{q} + G_{12}\hat{\phi} = b \tag{4.6}$$

$$G_{12}^T \Delta\hat{q} = \hat{b} \tag{4.7}$$

Si tenemos que $G_{11} = LL^T$ es la descomposición de Cholesky de G_{11}, se puede aislar $\Delta\hat{q}$ en (4.6) y sustituir en (4.7), con lo que se obtiene:

$$\Delta\hat{q} = L^{-T}L^{-1}(b - G_{12}\hat{\phi}) \tag{4.8}$$

$$\hat{\phi} = (G_{12}^T L^{-T} L^{-1} G_{12})^{-1}(G_{12}^T L^{-T} L^{-1} b - \hat{b}) \tag{4.9}$$

Si calculamos los sistemas triangulares $z = L^{-1}b$ y $E = L^{-1}G_{12}$, tenemos que:

$$\hat{\phi} = (E^T E)^{-1}(E^T z - \hat{b}) \tag{4.10}$$

que se puede resolver mediante la descomposición de Cholesky, tras calcular la matriz $(E^T E)$ y el vector $(E^T z - \hat{b})$. Hay que tener en cuenta que la dimensión de la matriz $(E^T E)$ es igual al número de líneas cerradas, que será normalmente muy pequeño.

Por su parte, $\Delta\hat{q}$ se obtiene mediante:

$$\Delta\hat{q} = L^{-T}L^{-1}b - L^{-T}L^{-1}G_{12}\hat{\phi} = L^{-T}(z - E\hat{\phi}) \tag{4.11}$$

4.4.3. Conexión entre ambas modelizaciones

Se muestra aquí que la primera de las dos modelizaciones presentadas puede ser obtenida a partir de una aproximación sobre la segunda.

Para mostrarlo, recurrimos al sistema 4.4, correspondiente a la red de la figura 4.4. En este sistema introducimos una pequeña perturbación, que consiste en sustituir el cero del elemento (4, 4) de la matriz por $-1/\alpha$, donde α es un escalar grande (10^{-8}), de tal manera que obtenemos:

$$\begin{bmatrix} d_1+d_5 & 0 & 0 & -1 \\ 0 & d_3+d_6 & -d_3 & 1 \\ 0 & -d_3 & d_3+d_4+d_7 & 0 \\ -1 & 1 & 0 & -1/\alpha \end{bmatrix} \begin{bmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \\ \phi_2 \end{bmatrix} = \begin{bmatrix} -\phi_1^k - \phi_5^k \\ \phi_3^k + \phi_6^k \\ -\phi_3^k + \phi_4^k + \phi_7^k - h_6 + h_5 \\ -q_2^k \end{bmatrix} \tag{4.12}$$

De la última ecuación tenemos que:

$$\phi_2 = -\alpha(\Delta\hat{q}_1 - \Delta\hat{q}_2 - q_2^k) = -\alpha\Delta\hat{q}_1 + \alpha\Delta\hat{q}_2 + \alpha q_2^k$$

y sustituyendo en las tres primeras ecuaciones:

$$\begin{bmatrix} d_1+d_5+\alpha & -\alpha & 0 \\ -\alpha & d_3+d_6+\alpha & -d_3 \\ 0 & -d_3 & d_3+d_4+d_7 \end{bmatrix} \begin{bmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \end{bmatrix} = \begin{bmatrix} -\phi_1^k - \phi_5^k + \alpha q_2^k \\ \phi_3^k + \phi_6^k - \alpha q_2^k \\ -\phi_3^k + \phi_4^k + \phi_7^k - h_6 + h_5 \end{bmatrix} \tag{4.13}$$

que coincide con la ecuación (4.2) correspondiente a la modelización de la línea cerrada como una tubería de gran resistencia.

Pasando al caso general, el sistema (4.5) se perturba de la siguiente manera:

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & -\frac{1}{\alpha}I \end{bmatrix} \begin{bmatrix} \Delta\hat{q} \\ \hat{\phi} \end{bmatrix} = \begin{bmatrix} b \\ \hat{b} \end{bmatrix} \tag{4.14}$$

De donde tenemos que:

$$\hat{\phi} = -\alpha\hat{b} + \alpha G_{12}^T \Delta\hat{q}$$

y sustituyendo:

$$\left(G_{11} + \alpha G_{12} G_{12}^T\right) \Delta\hat{q} = b + \alpha G_{12}\hat{b} \tag{4.15}$$

4.5. Modelización de válvulas de control del caudal

Las válvulas de control del caudal (VCQ) tratan de mantener el caudal a través de una válvula en un determinado valor, evitando que se sobrepase. Una válvula activa de este tipo puede manejarse de manera muy similar a una línea cerrada, como se ha mostrado arriba, sin más que cambiar el cero de la ecuación (4.3) por el valor de consigna de la VCQ. Esto da lugar a un sistema con la misma estructura que (4.5), que puede resolverse tal como se propone en la sección 4.4.2, o bien tratarse como una tubería de gran resistencia de acuerdo con 4.4.3.

4.6. Modelización de válvulas reguladoras de presión

En esta sección consideramos la inclusión en la simulación de dos tipos de válvulas reguladoras de presión: válvulas reductoras de presión (VRP) y válvulas sostenedoras de presión (VSP). En primer lugar tratamos las VRP.

Una VRP se utiliza para reducir la presión del punto de salida de la válvula hasta un valor establecido como consigna. La válvula puede estar en tres estados diferentes: i) si la altura piezométrica del punto de entrada es demasiado baja para proporcionar la presión de salida deseada, la válvula se abre por completo; ii) si las alturas piezométricas de los extremos de la válvula producirían un caudal inverso, la válvula se cierra; iii) en otro caso la válvula está activa y la presión de salida es igual al valor de la consigna. El primer caso corresponde a una tubería normal, mientras que el segundo se trataría como se ha descrito en la sección 4.4. En esta sección nos ocupamos del tercer caso.

En [39], una VRP activa se modeliza (en el contexto del método de mallas) considerando un camino independiente (o pseudo-malla) que va del nudo aguas abajo de la VRP a un nudo de altura fija. Sobre dicho camino se impone una ecuación de equilibrio de energía, que sustituye la ecuación de energía de una malla que pase por la VRP. Si hay más de una malla que pasa por la VRP, todas las demás deben redefinirse para que no pasen por ella. Además, las incógnitas consideradas

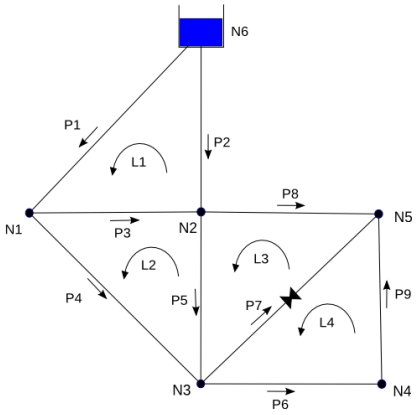

Figura 4.5: Red con una VRP.

son los caudales correctores asociados a las mallas originales, como si la VRP fuera una tubería normal. El procedimiento produce un sistema lineal con una matriz no simétrica.

Aquí se presentan dos maneras distintas de modelizar las VRP. La primera de ellas supone ampliar el sistema lineal de la iteración de Newton con una parte no simétrica. A diferencia de lo que se hace en Epanet, no se introducen aproximaciones adicionales para conseguir que la matriz sea simétrica, por lo que el proceso iterativo es más exacto y puede reducirse el número de iteraciones. La desventaja es que la resolución del sistema lineal es más compleja. La segunda forma de tratar las VRP supone trasladar al método de mallas la aproximación simétrica que realiza Epanet en el método GGA. En este caso la secuencia de iteraciones es la misma que la de Epanet.

4.6.1. Modelización mediante ampliación no simétrica del sistema

Al igual que en [39], en el método propuesto aquí se considera un camino entre el nudo aguas abajo de la VRP y un nudo de altura fija. Sin embargo, la ecuación de equilibrio de energía de dicho camino se añade a las demás, sin eliminar ninguna otra, y la pérdida de carga de la VRP pasa a ser una nueva incógnita. Las ventajas son que no hay necesidad de redefinir las mallas, y que la parte no simétrica del sistema está aislada, de manera que hay una parte en la que sí se puede aprovechar la simetría.

Consideremos la red mostrada en la figura 4.5. Inicialmente, si la válvula fuera

una tubería normal, el sistema de ecuaciones lineales en una iteración k sería:

$$\begin{bmatrix} d_1+d_2+d_3 & -d_3 & 0 & 0 \\ -d_3 & d_3+d_4+d_5 & -d_5 & 0 \\ 0 & -d_5 & d_5+d_7+d_8 & -d_7 \\ 0 & 0 & -d_7 & d_6+d_7+d_9 \end{bmatrix} \begin{bmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \\ \Delta\hat{q}_4 \end{bmatrix} = \begin{bmatrix} -\phi_1^k+\phi_2^k-\phi_3^k \\ \phi_3^k-\phi_4^k+\phi_5^k \\ -\phi_5^k-\phi_7^k+\phi_8^k \\ -\phi_6^k+\phi_7^k-\phi_9^k \end{bmatrix} \tag{4.16}$$

Si la VRP de la línea 7 está activa, la relación entre el caudal que circula a través de ella y la pérdida de carga, dada por d_7, es desconocida. Sin embargo, podemos eliminarla, e introducir en su lugar la propia pérdida de carga (ϕ_7) como una incógnita.

Por otra parte, la altura piezométrica en el nudo aguas abajo de la válvula h_5 es conocida, siendo igual a la suma de la elevación del nudo más la consigna de presión de la VRP. Además, la pérdida de carga entre el depósito y el nudo 5 debe ser igual a la suma de pérdidas de carga de las líneas de un camino que va del depósito a dicho nudo, por ejemplo el camino formado por las tuberías 2 y 8, es decir:

$$\phi_2 + \phi_8 = h_6 - h_5$$

Aproximando las funciones no lineales de los caudales, ϕ_2 y ϕ_8, por medio de los dos primeros términos de la serie de Taylor, tenemos:

$$-h_6 + h_5 + \phi_2^k - d_2\Delta\hat{q}_1 + \phi_8^k - d_8\Delta\hat{q}_3 = 0 \tag{4.17}$$

Teniendo esto en cuenta, obtenemos:

$$\begin{bmatrix} d_1+d_2+d_3 & -d_3 & 0 & 0 & 0 \\ -d_3 & d_3+d_4+d_5 & -d_5 & 0 & 0 \\ 0 & -d_5 & d_5+d_8 & 0 & 1 \\ 0 & 0 & 0 & d_6+d_9 & -1 \\ -d_2 & 0 & -d_8 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \\ \Delta\hat{q}_4 \\ \phi_7 \end{bmatrix} = - \begin{bmatrix} \phi_1^k-\phi_2^k+\phi_3^k \\ -\phi_3^k+\phi_4^k-\phi_5^k \\ \phi_5^k-\phi_8^k \\ \phi_6^k+\phi_9^k \\ -h_6+h_5+\phi_2^k+\phi_8^k \end{bmatrix} \tag{4.18}$$

En un caso general, con un número cualquiera de VRP, el sistema anterior presenta la forma:

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} \Delta\hat{q} \\ \hat{\phi} \end{bmatrix} = \begin{bmatrix} b \\ \hat{b} \end{bmatrix} \tag{4.19}$$

o de manera equivalente:

$$G_{11}\Delta\hat{q} + G_{12}\hat{\phi} = b \tag{4.20}$$

$$G_{21}\Delta\hat{q} + G_{22}\hat{\phi} = \hat{b} \tag{4.21}$$

donde, de forma similar al caso de las líneas cerradas (y VCQ), G_{11} es la misma matriz que en el sistema original, cambiando únicamente los coeficientes afectados por las VRP (como si las válvulas se hubieran sustituido por tuberías de resistencia nula). No se añaden nuevos elementos no nulos a la matriz. G_{12} y G_{22} son matrices de incidencia, cuyos elementos pueden tomar sólo los valores 0 y ± 1. Concretamente, G_{12} indica las válvulas contenidas en cada malla, y G_{22} indica las válvulas contenidas en cada uno de los caminos asociados a las VRP. En el ejemplo anterior, el camino del nudo 5 al nudo 6 no contiene ninguna válvula, por lo que el único elemento de G_{22} es cero. Finalmente, G_{21} contiene las derivadas de las pérdidas de carga de las tuberías que forman los caminos asociados a las VRP.

Operando de forma similar a la sección 4.4.2, tenemos:

$$\Delta\hat{q} = G_{11}^{-1}b - G_{11}^{-1}G_{12}\hat{\phi} \tag{4.22}$$

$$\hat{\phi} = (G_{21}G_{11}^{-1}G_{12} - G_{22})^{-1}(G_{21}G_{11}^{-1}b - \hat{b}) \tag{4.23}$$

Si calculamos los sistemas triangulares $z = G_{11}^{-1}b$ y $E = G_{11}^{-1}G_{12}$, tenemos que:

$$\hat{\phi} = (G_{21}E - G_{22})^{-1}(G_{21}z - \hat{b}) \tag{4.24}$$

que es un sistema lineal de pequeño tamaño (igual al número de VRP activas), cuya solución se puede obtener mediante descomposición LU con pivotación, tras calcular la matriz $(G_{21}E - G_{22})$ y el vector $(G_{21}z - \hat{b})$. A continuación, $\Delta\hat{q}$ se obtiene como:

$$\Delta\hat{q} = G_{11}^{-1}b - G_{11}^{-1}G_{12}\hat{\phi} = z - E\hat{\phi} \tag{4.25}$$

Resumiendo, las VRP activas se pueden tratar sin redefinir las mallas de la red, y por tanto sin cambiar el patrón de dispersidad de la matriz del sistema, mediante un procedimiento similar al presentado en 4.4.2 para líneas cerradas. La diferencia principal consiste en que en el caso de las VRP el pequeño sistema de ecuaciones que se introduce, con la matriz $(G_{21}G_{11}^{-1}G_{12} - G_{22})$, no es simétrico. Sin embargo, la matriz G_{11} sigue siendo simétrica y puede factorizarse mediante la descomposición de Cholesky.

Queda sin embargo comprobar que la matriz del pequeño sistema adicional, $(G_{21}G_{11}^{-1}G_{12} - G_{22})$, conocido como complemento de Schur, es en efecto invertible. Pasamos a discutirlo a continuación.

Invertibilidad del complemento de Schur.

Vamos a mostrar que la inversa de la matriz $(G_{21}G_{11}^{-1}G_{12} - G_{22})$ existe si las siguientes condiciones se cumplen: (i) la matriz del sistema (4.19) es invertible, y (ii) la matriz A es invertible. La primera condición es la misma que se necesita también

para el GGA, por ejemplo en Epanet. La segunda condición se deriva del hecho de que G_{11} es la matriz de la iteración de Newton-Raphson para una red como la original salvo por las VRP, que son sustituidas por tuberías de resistencia nula, y por tanto es una matriz definida positiva [68].

Asumimos pues que las dos condiciones se cumplen. Es decir, la matriz del sistema (4.19):

$$G = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \tag{4.26}$$

es invertible, y también lo es la matriz G_{11}. A continuación construimos la matriz L:

$$L = \begin{bmatrix} I_{11} & 0 \\ -G_{21}G_{11}^{-1} & I_{22} \end{bmatrix} \tag{4.27}$$

donde I_{11} e I_{22} son matrices identidad con las mismas dimensiones que G_{11} y G_{22} respectivamente. Tenemos que:

$$LG = \begin{bmatrix} G_{11} & G_{12} \\ 0 & G_{22} - G_{21}G_{11}^{-1}G_{12} \end{bmatrix} \tag{4.28}$$

Consecuentemente,

$$\det(G) = \det(LG) = \det(G_{11})\det(G_{22} - G_{21}G_{11}^{-1}G_{12}) \tag{4.29}$$

Dado que $\det(G) \neq 0$ y $\det(A_{11}) \neq 0$, se deriva que $\det(G_{22} - G_{21}G_{11}^{-1}G_{12}) \neq 0$.

4.6.2. Modelización mediante aproximación simétrica

En el apartado anterior se ha visto que la consideración de las VRP introduce una modificación no simétrica en la matriz del sistema. Sin embargo, Epanet es capaz de trabajar con una matriz simétrica incluso en presencia de VRP. ¿Cómo lo hace? ¿es posible trasladar esa formulación al método de mallas?

Esas preguntas han sido abordadas en esta tesis. Concretamente, tras estudiar la formulación de las VRP en Epanet, se ha visto que efectivamente es posible trasladar esa formulación al método de mallas, tal como se presenta en este apartado.

Eso no quiere decir que esta forma de tratar las VRP sea preferible a la del apartado anterior, dado que el precio a pagar por conservar la simetría es la introducción de una aproximación adicional, que puede llevar a que el proceso de Newton-Raphson necesite más iteraciones para la convergencia. Es importante destacar, sin embargo, que la secuencia de iteraciones producidas es la misma que la resultante de Epanet, salvo pequeños errores de redondeo.

Como se expuso en el apartado 2.5.3, en [68, 69] se muestra que el método GGA (Epanet), el de teoría lineal basado en gradientes y el método de mallas producen la misma secuencia de iteraciones de Newton (si empiezan de la misma aproximación inicial, y suponiendo aritmética exacta), dado que en los tres casos la iteración de Newton se puede obtener a partir del siguiente sistema lineal:

$$\begin{bmatrix} D & A_{12} \\ A_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ h \end{bmatrix} = \begin{bmatrix} -\phi^k - A_{10}\hat{h} \\ -A_{12}^T q^k + c \end{bmatrix} \tag{4.30}$$

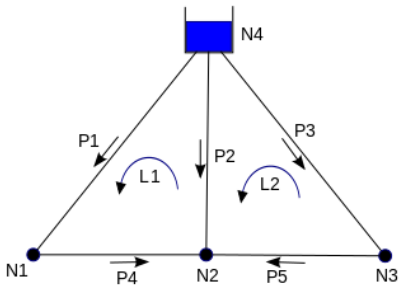

Figura 4.6: Una red muy simple.

Por ejemplo, considerando una red muy sencilla como la mostrada en la figura 4.6, el sistema sería:

$$\left[\begin{array}{ccccc|ccc} d_1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & d_2 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & d_3 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & d_4 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & d_5 & 0 & 1 & -1 \\ \hline 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \Delta q_1 \\ \Delta q_2 \\ \Delta q_3 \\ \Delta q_4 \\ \Delta q_5 \\ \hline h_1 \\ h_2 \\ h_3 \end{array}\right] = \left[\begin{array}{c} -\phi_1^k + \hat{h}_4 \\ -\phi_2^k + \hat{h}_4 \\ -\phi_3^k + \hat{h}_4 \\ -\phi_4^k \\ -\phi_5^k \\ \hline -q_1^k + q_4^k + c_1 \\ -q_2^k - q_4^k - q_5^k + c_2 \\ -q_3^k + q_5^k + c_3 \end{array}\right] \tag{4.31}$$

Ese sistema lineal se puede resolver de maneras distintas, dando lugar a los métodos de GGA, teoría lineal basado en gradientes y mallas.

Aproximación para las VRP en Epanet.

Consideremos a continuación el caso en que una de las líneas es una VRP. Por ejemplo, si la línea 4 fuera una VRP, la ecuación anterior cambiaría de la siguiente manera (se muestran en negrita los elementos que han cambiado):

$$\left[\begin{array}{ccccc|ccc} d_1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & d_2 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & d_3 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \mathbf{0} & 0 & \mathbf{0} & 1 & 0 \\ 0 & 0 & 0 & 0 & d_5 & 0 & 1 & -1 \\ \hline 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \Delta q_1 \\ \Delta q_2 \\ \Delta q_3 \\ \Delta q_4 \\ \Delta q_5 \\ \hline h_1 \\ h_2 \\ h_3 \end{array}\right] = \left[\begin{array}{c} -\phi_1^k + \hat{h}_4 \\ -\phi_2^k + \hat{h}_4 \\ -\phi_3^k + \hat{h}_4 \\ \mathbf{s_4} \\ -\phi_5^k \\ \hline -q_1^k + q_4^k + c_1 \\ -q_2^k - q_4^k - q_5^k + c_2 \\ -q_3^k + q_5^k + c_3 \end{array}\right] \tag{4.32}$$

donde s_4 es la altura piezométrica fijada para el nudo de control de la válvula (o sea, la elevación del nudo más la consigna de presión de la válvula), de forma que la cuarta

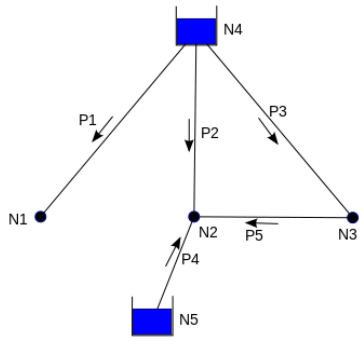

Figura 4.7: Modelización de VRP en el método GGA

ecuación del sistema expresa simplemente el hecho de que la altura piezométrica de dicho nudo viene dada por la consigna de la válvula.

Obsérvese que el sistema ha dejado de ser simétrico. Para conservar la simetría, Epanet introduce una aproximación que, aunque se realiza sobre un sistema diferente (el del método GGA), se puede trasladar al sistema anterior. En concreto, la aproximación consistiría en modificar la ecuación relativa al balance de masas del nudo aguas arriba de la VRP (ecuación 6 en el sistema anterior), despreciando el incremento de caudal de la propia VRP. Es decir:

$$\left[\begin{array}{ccccc|ccc} d_1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & d_2 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & d_3 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \mathbf{0} & 0 & \mathbf{0} & 1 & 0 \\ 0 & 0 & 0 & 0 & d_5 & 0 & 1 & -1 \\ \hline 1 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \Delta q_1 \\ \Delta q_2 \\ \Delta q_3 \\ \Delta q_4 \\ \Delta q_5 \\ \hline h_1 \\ h_2 \\ h_3 \end{array}\right] = \left[\begin{array}{c} -\phi_1^k + \hat{h}_4 \\ -\phi_2^k + \hat{h}_4 \\ -\phi_3^k + \hat{h}_4 \\ \mathbf{s_4} \\ -\phi_5^k \\ \hline -q_1^k + q_4^k + c_1 \\ -q_2^k - q_4^k - q_5^k + c_2 \\ -q_3^k + q_5^k + c_3 \end{array}\right] \tag{4.33}$$

El de arriba es un sistema casi equivalente al que usa Epanet (para que lo fuera habría que modificar el elemento (4, 4) de la matriz, colocando un valor pequeño en vez de un cero).

Se puede interpretar que el sistema planteado corresponde a una red como la de la figura 4.7, donde se ha añadido un depósito ficticio conectado al nudo de control de la VRP mediante la tubería 4, que se supone que tiene resistencia cero. Además, la demanda del nudo 1 se va modificando en cada iteración k, de forma que $c_1^k = q_4^k + c_1$.

En general, el sistema se puede escribir como:

$$\left[\begin{array}{cc} \hat{D} & \hat{A}_{12} \\ \hat{A}_{12}^T & 0 \end{array}\right] \left[\begin{array}{c} \Delta q \\ h \end{array}\right] = \left[\begin{array}{c} -\hat{\phi}^k - A_{10}\hat{h} \\ -A_{12}^T q^k + c \end{array}\right] \tag{4.34}$$

donde $\hat{A}$ y $\hat{A}_{12}$ se obtienen a partir de A y A_{12}, respectivamente, introduciendo ceros

en los elementos afectados por la VRP, y $\hat{\phi}^k$ se obtiene a partir de ϕ^k, modificado para tener en cuenta la consigna de la válvula, tal como se ilustra en el ejemplo anterior.

Es importante tener en cuenta que, debido al cero introducido en $\hat{A}_{12}^T$, el nuevo vector de caudales no satisface, en general, el equilibrio de masas. En particular, el equilibrio se cumple en todos los nudos excepto en el nudo aguas arriba de la VRP (el nudo 1 en el ejemplo).

Derivación del método de mallas sobre el sistema modificado.

A continuación derivaremos la iteración del método de mallas a partir del sistema modificado (4.34). Lo haremos primero de manera general, y luego lo ilustraremos con el caso de la red de ejemplo. Empezamos premultiplicando el sistema (4.34):

$$\begin{bmatrix} M_{31} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{D} & \hat{A}_{12} \\ \hat{A}_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ h \end{bmatrix} = \begin{bmatrix} M_{31} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} -\hat{\phi}^k - A_{10}\hat{h} \\ -A_{12}^T q^k + c \end{bmatrix} \tag{4.35}$$

donde M_{31} es una matriz de mallas para la red original. Al multiplicar obtenemos:

$$\begin{bmatrix} M_{31}\hat{D} & M_{31}\hat{A}_{12} \\ \hat{A}_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ h \end{bmatrix} = \begin{bmatrix} M_{31}(-\hat{\phi}^k - A_{10}\hat{h}) \\ -A_{12}^T q^k + c \end{bmatrix} \tag{4.36}$$

Al introducir una VRP, el producto $M_{31}\hat{A}_{12}$ tendrá una columna no nula z correspondiente al nudo aguas arriba de la VRP (nudo i), lo que nos lleva al sistema:

$$\begin{bmatrix} M_{31}\hat{D} & z \\ \hat{A}_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ h_i \end{bmatrix} = \begin{bmatrix} M_{31}(-\hat{\phi}^k - A_{10}\hat{h}) \\ -A_{12}^T q^k + c \end{bmatrix} \tag{4.37}$$

que es un sistema indeterminado, dado que hay una incógnita más que ecuaciones. Hace falta una ecuación más, que corresponde a la pérdida de carga que se produce en un camino que vaya de un nudo de altura fija al nudo de control de la VRP. Eso supone usar una matriz de bucles modificada, $\hat{M}_{31}$, formada añadiendo a M_{31} una nueva fila correspondiente al camino mencionado. Es decir:

$$\begin{bmatrix} \hat{M}_{31}\hat{D} & \hat{z} \\ \hat{A}_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ h_i \end{bmatrix} = \begin{bmatrix} \hat{M}_{31}(-\hat{\phi}^k - A_{10}\hat{h}) \\ -A_{12}^T q^k + c \end{bmatrix} \tag{4.38}$$

donde $\hat{z}$ es la columna i de $\hat{M}_{31}\hat{A}_{12}$.

Se puede demostrar que si Δq cumple la ecuación anterior, se puede escribir de la forma $\Delta q = \hat{M}_{31}^T \Delta\hat{q}$, donde $\Delta\hat{q}$ es un vector de correcciones de mallas que incorpora un elemento más, correspondiente al camino de la VRP.

Sustituyendo obtenemos:

$$\begin{bmatrix} \hat{M}_{31}\hat{D}\hat{M}_{31}^T & \hat{z} \\ \hat{A}_{12}^T\hat{M}_{31}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta\hat{q} \\ h_i \end{bmatrix} = \begin{bmatrix} \hat{M}_{31}(-\hat{\phi}^k - A_{10}\hat{h}) \\ -A_{12}^T q^k + c \end{bmatrix} \tag{4.39}$$

que se puede transformar en:

$$\begin{bmatrix} \hat{M}_{31}\hat{D}\hat{M}_{31}^T & \hat{z} \\ \hat{z}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta\hat{q} \\ h_i \end{bmatrix} = \begin{bmatrix} \hat{M}_{31}(-\hat{\phi}^k - A_{10}\hat{h}) \\ -w^T q^k + c_i \end{bmatrix} \tag{4.40}$$

donde w es la columna i de la matriz A_{12}.

El nuevo vector de caudales se obtiene como:

$$q^{k+1} = q^k + \hat{M}_{31}^T \Delta\hat{q} \tag{4.41}$$

Se puede además introducir una simplificación adicional, que consiste en sustituir el cero de la matriz por un pequeño valor:

$$\begin{bmatrix} \hat{M}_{31}\hat{D}\hat{M}_{31}^T & \hat{z} \\ \hat{z}^T & -1/\alpha \end{bmatrix} \begin{bmatrix} \Delta\hat{q} \\ h_i \end{bmatrix} = \begin{bmatrix} \hat{M}_{31}(-\hat{\phi}^k - A_{10}\hat{h}) \\ -w^T q^k + c_i \end{bmatrix} \tag{4.42}$$

donde α es un valor grande (ej. 10^8). Tenemos que:

$$h_i = \alpha\hat{z}^T\Delta\hat{q} - \alpha(-w^T q^k + c_i) \tag{4.43}$$

y sustituyendo:

$$(\hat{M}_{31}\hat{D}\hat{M}_{31}^T + \alpha\hat{z}\hat{z}^T)\Delta\hat{q} = \hat{M}_{31}(-\hat{\phi}^k - A_{10}\hat{h}) + \alpha z(-w^T q^k + c_i) \tag{4.44}$$

En el caso de varias VRP, el sistema sería:

$$(\hat{M}_{31}\hat{D}\hat{M}_{31}^T + \alpha\hat{Z}\hat{Z}^T)\Delta\hat{q} = \hat{M}_{31}(-\hat{\phi}^k - A_{10}\hat{h}) + \alpha\hat{Z}(-W^T q^k + \hat{c}) \tag{4.45}$$

Ejemplo

En la red de ejemplo de la figura 4.6, donde la línea 4 es una VRP activa, tenemos que $\hat{A}_{12}$ será igual a la matriz A_{12} de la red de la figura 4.7:

$$\hat{A}_{12} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

mientras que la matriz de mallas M_{31} es:

$$M_{31} = \begin{bmatrix} 1 & -1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & -1 \end{bmatrix} \tag{4.46}$$

con lo que la primera columna del producto $M_{31}\hat{A}_{12}$ es no nula, y es igual a $z = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$.

Para formar $\hat{M}_{31}$, añadimos a M_{31} una fila correspondiente a un camino desde un nudo de altura fija al nudo de control de la VRP. Consideramos para ello la red

modificada según la figura 4.7, cogiendo un camino del depósito real (N4) al ficticio (N5). Ese camino podría ser el formado por las líneas 2 y 4, con lo que tendríamos:

$$\hat{M}_{31} = \begin{bmatrix} 1 & -1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & -1 \\ 0 & 1 & 0 & -1 & 0 \end{bmatrix} \tag{4.47}$$

La iteración de Newton (4.40) sería en este caso:

$$\left[\begin{array}{ccc|c} d_1+d_2 & -d_2 & -d_2 & 1 \\ -d_2 & d_2+d_3+d_5 & d_2 & 0 \\ -d_2 & d_2 & d_2 & 0 \\ \hline 1 & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \\ \hline h_1 \end{array}\right] = \left[\begin{array}{c} -\phi_1^k + \phi_2^k + s_4 \\ -\phi_2^k + \phi_3^k + \phi_5^k \\ -\phi_2^k + \hat{h}_4 - s_4 \\ \hline -q_1^k + q_4^k + c_1 \end{array}\right] \tag{4.48}$$

y se podría aproximar mediante:

$$\left[\begin{array}{ccc|c} d_1+d_2 & -d_2 & -d_2 & 1 \\ -d_2 & d_2+d_3+d_5 & d_2 & 0 \\ -d_2 & d_2 & d_2 & 0 \\ \hline 1 & 0 & 0 & -1/\alpha \end{array}\right] \left[\begin{array}{c} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \\ \hline h_1 \end{array}\right] = \left[\begin{array}{c} -\phi_1^k + \phi_2^k + s_4 \\ -\phi_2^k + \phi_3^k + \phi_5^k \\ -\phi_2^k + \hat{h}_4 - s_4 \\ \hline -q_1^k + q_4^k + c_1 \end{array}\right] \tag{4.49}$$

de manera que:

$$h_i = \alpha\Delta\hat{q}_1 - \alpha(-q_1^k + q_4^k + c_1)$$

y sustituyendo:

$$\begin{bmatrix} d_1+d_2+\alpha & -d_2 & -d_2 \\ -d_2 & d_2+d_3+d_5 & d_2 \\ -d_2 & d_2 & d_2 \end{bmatrix} \begin{bmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \end{bmatrix} = \begin{bmatrix} -\phi_1^k + \phi_2^k + s_4 + \alpha(-q_1^k + q_4^k + c_1) \\ -\phi_2^k + \phi_3^k + \phi_5^k \\ -\phi_2^k + \hat{h}_4 - s_4 \end{bmatrix} \tag{4.50}$$

Este sistema se puede interpretar como una red como la de la figura 4.8, donde, además del depósito ficticio N5 correspondiente a la VRP, se ha añadido otro, N6, conectado al nudo 1 mediante una tubería P6 de gran resistencia (α). La altura del depósito N6 sería $\alpha(q_1^k - q_4^k - c_1)$, ajustándose en cada iteración para buscar el balance de masas en el nudo 1.

Cambio de estado de una VRP

Consideramos ahora el caso en que tras cierta iteración de Newton, una VRP inicialmente activa deja de estarlo. En principio la siguiente iteración vendría dada por la iteración estándar del método de mallas, que recordemos es:

$$M_{31}DM_{31}^T\Delta\hat{q} = M_{31}(-\phi^k - A_{10}\hat{h}) \tag{4.51}$$

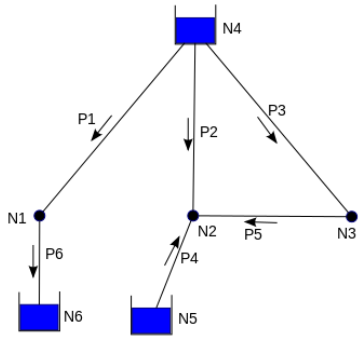

Figura 4.8: Modelización de VRP en el método de mallas

Pero hay un problema: mientras la VRP está activa el vector de caudales q no satisface en general el equilibrio de masas, lo cual es necesario para poder aplicar la iteración anterior.

Así pues, la primera iteración después de que una VRP deje de estar activa deberá ser diferente. Como la VRP no está activa, se comporta como una tubería normal. Premultiplicando (4.30):

$$\begin{bmatrix} \hat{M}_{31} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} D & A_{12} \\ A_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ h \end{bmatrix} = \begin{bmatrix} \hat{M}_{31} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} -\phi^k - A_{10}\hat{h} \\ -A_{12}^T q^k + c \end{bmatrix} \tag{4.52}$$

obtenemos:

$$\begin{bmatrix} \hat{M}_{31}D & \hat{M}_{31}A_{12} \\ A_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ h \end{bmatrix} = \begin{bmatrix} \hat{M}_{31}(-\phi^k - A_{10}\hat{h}) \\ -A_{12}^T q^k + c \end{bmatrix} \tag{4.53}$$

Obsérvese que el segundo bloque de ecuaciones, dado por

$$A_{12}^T \Delta q = -A_{12}^T q^k + c$$

garantiza que el vector de caudales resultante cumplirá el equilibrio de masas.

El producto $\hat{M}_{31}A_{12}$ no es cero, sino que tendrá una columna no nula correspondiente al nudo aguas arriba de la VRP (nudo i por ejemplo), por tanto, procediendo de la misma manera que hemos hecho anteriormente, llegamos a:

$$\begin{bmatrix} \hat{M}_{31}D\hat{M}_{31}^T & \hat{z} \\ \hat{z}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta \hat{q} \\ h_i \end{bmatrix} = \begin{bmatrix} \hat{M}_{31}(-\phi^k - A_{10}\hat{h}) \\ -w^T q^k + c_i \end{bmatrix} \tag{4.54}$$

donde w es la columna i de la matriz A_{12}, y $\hat{z}$ es la columna i de $\hat{M}_{31}A_{12}$.

El nuevo vector de caudales, obtenido como

$$q^{k+1} = q^k + \hat{M}_{31}^T \Delta \hat{q}$$

sí cumple el equilibrio de masas, por lo que la siguiente iteración ya puede realizarse mediante (4.51).

Por último, se puede introducir también la simplificación adicional mencionada anteriormente.

Ejemplo

En nuestro ejemplo, la iteración explicada anteriormente tendría la forma:

$$\left[\begin{array}{ccc|c} d_1+d_2+d_4 & -d_2 & -d_2-d_4 & 0 \\ -d_2 & d_2+d_3+d_5 & d_2 & 0 \\ -d_2-d_4 & d_2 & d_2+d_4 & 1 \\ \hline 0 & 0 & 1 & 0 \end{array}\right] \left[\begin{array}{c} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \\ \hline h_1 \end{array}\right] = \left[\begin{array}{c} -\phi_1^k+\phi_2^k-\phi_4^k \\ -\phi_2^k+\phi_3^k+\phi_5^k \\ -\phi_2^k+\phi_4^k+\hat{h}_4 \\ \hline -q_1^k+q_4^k+c_1 \end{array}\right] \tag{4.55}$$

O bien, introduciendo la simplificación adicional:

$$\left[\begin{array}{ccc} d_1+d_2+d_4 & -d_2 & -d_2-d_4 \\ -d_2 & d_2+d_3+d_5 & d_2 \\ -d_2-d_4 & d_2 & d_2+d_4+\alpha \end{array}\right] \left[\begin{array}{c} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \end{array}\right] = \left[\begin{array}{c} -\phi_1^k+\phi_2^k-\phi_4^k \\ -\phi_2^k+\phi_3^k+\phi_5^k \\ -\phi_2^k+\phi_4^k+\hat{h}_4+\alpha(-q_1^k+q_4^k+c_1) \end{array}\right] \tag{4.56}$$

4.6.3. Válvulas sostenedoras de presión (VSP).

Las VSP son muy similares a las VRP. En concreto, una VSP trata de mantener la presión del nudo de entrada a un valor determinado. Los dos métodos descritos anteriormente para las VRP, en los apartados 4.6.1 y 4.6.2, son igualmente válidos para las VSP, sin más que intercambiar el papel de los nudos aguas arriba y aguas abajo.

4.7. Elección de un vector inicial de caudales

Una de las dificultades encontradas en la simulación mediante el método de mallas ha sido la elección de un vector inicial de caudales, es decir, un vector q^0 que satisfaga la ecuación de equilibrio de masas. Se ha comprobado que la elección del vector tiene una influencia considerable en el número de iteraciones necesarias.

Para determinar q^0, construimos un árbol de expansión de la red, e imponemos un determinado caudal a las cuerdas (líneas que no pertenecen al árbol). El caudal de las líneas del árbol se puede calcular entonces recorriendo éste desde las hojas hasta la raíz, e imponiendo la ecuación de equilibrio de masas en cada nudo de la red.

Se pueden usar diferentes árboles de expansión, así como diferentes caudales para las cuerdas. En las pruebas realizadas, los mejores resultados se han obtenido usando

Tabla 4.1: Redes de test consideradas

Red	nudos	líneas	D/E/B/V	VP	Duración/paso	Fórmula
red 1	97	119	2/3/2/0	0	24h/1h	H-W
red 2	1893	2467	2/0/0/2	1	0h/-	D-W
red 3	12527	14831	2/2/4/5	1	26h 55min/5min	H-W
red 4	26653	29046	26/0/0/0	0	48h/20min	H-W
red 5	4240	4649	4/0/0/6	5	96h/5min	H-W
red 6	25816	29345	3/0/0/51	10	24h/1h	D-W

un árbol de expansión de mínima resistencia (es decir, un árbol de expansión donde la suma de resistencias R de todas las líneas del árbol es mínima), y asignando a cada cuerda un caudal correspondiente a una velocidad de 1 m/s (este es caudal inicial utilizado por Epanet para todas las líneas). El árbol de expansión de mínima resistencia se obtiene mediante una implementación del algoritmo de Prim [56].

4.8. Resultados

En esta sección presentamos resultados que comparan los métodos de GGA y de mallas, teniendo en cuenta diferentes aspectos. Consideramos las redes hidráulicas que se muestran en la tabla 4.1. La red 1 es la red de ejemplo 3 de Epanet [58]. La red 2 puede descargarse del *Centre for Water Systems* de la Universidad de Exeter[1]. La red 3 es la propuesta en [54] como red 2. Las redes 4, 5 y 6 son modelos de redes reales de distintas ciudades de España. En la tabla 4.1 pueden consultarse los datos principales de las redes, incluyendo el número de depósitos (D), embalses (E), bombas (B), válvulas (V), el número válvulas reguladoras de presión (VP, que incluye tanto VRPs como VSPs), el periodo de simulación considerado (duración), el paso de tiempo para la simulación hidráulica y la fórmula usada para las pérdidas de carga de las tuberías según se describe en el apartado 2.2 (H-W para Hazen-Williams y D-W para Darcy-Weisbach).

Para mostrar resultados realistas de tiempos de cómputo se deben considerar implementaciones eficientes de los métodos. Por esta razón, hemos usado Epanet (versión 2.00.12) como implementación altamente eficiente, escrita en lenguaje C, del método GGA, mientras que para el método de mallas hemos usado una implementación, también en C, realizada como una versión modificada de Epanet. Por supuesto, se han utilizado las mismas opciones de optimización en la compilación de ambos códigos. El código para la resolución de sistemas lineales mediante la descomposición de Cholesky se ha tomado de Epanet, y es exactamente el mismo para ambos métodos. Para las implementaciones de los métodos de selección de mallas se ha utilizado código Python, aunque el método con mejores prestaciones (**m4**, como veremos) se ha implementado también en lenguaje C y se ha integrado en la

[1] http://emps.exeter.ac.uk/media/universityofexeter/emps/research/cws/downloads/exnet.inp

implementación del *solver* de mallas. Este enfoque difiere de otros trabajos, como [16, 17, 24], donde se usan implementaciones en Matlab de los métodos.

Los tiempos de cómputo a medir en esta sección son en general muy pequeños, pudiendo verse afectados de forma importante por factores aleatorios no controlados, lo que dificulta la obtención de medidas fiables. Por ello, las tareas objeto de medición se repiten varias veces para conseguir un tiempo acumulado suficiente, obteniendo el tiempo medio correspondiente. Además, la serie de repeticiones se lleva a cabo tres veces, extrayendo el tiempo mínimo. Los tiempos medidos corresponden a tiempos reales (es decir, no tiempos de CPU), que se han obtenido en el caso de código C mediante la función de OpenMP `omp_get_wtime()`, y en el caso del código Python mediante la función `time()` del módulo del mismo nombre de la librería estándar de Python 2.7. Los tiempos se expresan en segundos excepto donde se indica lo contrario.

Respecto a las máquinas utilizadas, las pruebas de simulación de redes han sido realizadas sobre un computador con dos procesadores *multicore* Intel® Xeon® E5-2697 v3 a 2,60 GHz y 128 Gbytes de RAM (aunque los programas sólo hacen uso de un *core* de uno de los procesadores). Se realizaron pruebas con los compiladores de GNU (`gcc`) e Intel (`icc`), y distintas opciones de optimización. Como resultado de ello, se eligió el segundo compilador al observarse que los tiempos de ejecución eran considerablemente mejores, y la opción de optimización del compilador `-O2` (no había prácticamente diferencia entre `-O2` y `-O3`). Los tiempos presentados están en segundos.

Otras pruebas en las que se muestra el tiempo para la obtención del conjunto de mallas independientes (ver tabla 4.3) han sido realizadas sobre un computador de menores prestaciones, dotado de un procesador Intel® Core™2 Duo E8400 a 3 GHz con 2 cores, y una memoria de 4 Gbytes. Esto se debe a que algunos de los códigos para selección de mallas utilizan librerías para Python que no estaban instaladas en la máquina donde se hicieron las simulaciones.

4.8.1. Selección del conjunto de mallas

En primer lugar se presentan en la tabla 4.2 resultados que evalúan los métodos de selección de mallas presentados en el apartado 4.2, desde el punto de vista de la dispersidad de la matriz resultante. Las columnas bajo el título **n** comparan el tamaño de la matriz para GGA y el método de mallas. Se puede ver que para redes normales como las consideradas aquí, la matriz producida por el método de mallas es mucho más pequeña que la del método GGA. Las columnas bajo el título **nnz(A)** muestran el número de elementos no nulos de las matrices, para el método GGA y para cada uno de los métodos de selección de mallas presentados anteriormente. Como era de esperar, los mejores resultados se consiguen con **m2**, aunque el alto coste computacional del método, en términos del tiempo de ejecución y memoria, hacen imposible su uso para redes de tamaño medio o grande (redes 3, 4 y 6 de las analizadas). Entre los otros métodos de selección de mallas, **m4** es el mejor, consiguiendo una reducción considerable del número de elementos no nulos respecto GGA.

Figura 4.9: Esquema de la red 4

Tabla 4.2: Número de elementos no nulos en la matriz del sistema (A) y su factor de Cholesky (L). Comparación de GGA frente al método de mallas con diferentes algoritmos de selección de mallas.

	n		**nnz(A)**					**nnz(L)**	
Red	**GGA**	**mallas**	**GGA**	**m1**	**m2**	**m3**	**m4**	**GGA**	**m4**
red 1	92	27	206	128	84	93	90	274	94
red 2	1891	577	4306	4895	1664	1825	1695	5958	1935
red 3	12 523	2309	26 839	15 934	-	6736	6439	37493	7408
red 4	26 627	2419	55 607	52 833	-	10 875	9608	81501	13 822
red 5	4236	418	8861	5562	1528	1736	1574	12680	1968
red 6	25 813	3542	55 122	93 651	-	16 882	14 749	90783	23 307

Las columnas bajo el título **nnz(L)** muestran el número de elementos no nulos del factor de Cholesky de la matriz del sistema lineal. El número de elementos no nulos de la matriz factorizada, que se determina al principio de la simulación tras la reordenación y descomposición simbólica de la matriz, es un buen indicador del tiempo de cálculo necesario para resolver el sistema lineal. El número de elementos no nulos de la matriz factorizada es mayor que el de la matriz original, y depende del método de reordenación utilizado. En nuestro caso se usa el método de mínimo grado [26], cuya implementación se ha cogido de Epanet. Puede verse que hay una reducción considerable en el número de elementos no nulos para **m4** respecto a GGA.

Comparando los resultados de esta tesis con los de otros trabajos previos, en [24] se utilizaba un método de co-árbol reformulado (*reformulated co-tree method*) que se aplicaba sobre la red 2, produciendo un incremento de 2 % en el número de elementos no nulos de la matriz factorizada respecto GGA. Los resultados de esta tesis son claramente mejores, con una reducción del 68 % en el número de elementos no nulos respecto GGA.

En [16, 17] se presentan resultados para un algoritmo similar a **m2**, aplicándolo sobre dos conjuntos de redes: un conjunto de redes compuestas por mallas rectangulares y otro donde las redes están formadas por redes hexagonales. El artículo analiza la dispersidad de las matrices y el tiempo de computación empleado. En esta tesis se ha aplicado el método de selección de mallas **m4** a esas mismas redes y se han obtenido exactamente los mismos resultados en términos de dispersidad de la matriz, lo que indica que el método **m4** es capaz de obtener los resultados óptimos para esas redes, con la ventaja de ser un algoritmo muy rápido, como se verá a continuación.

A continuación consideramos, por medio de la tabla 4.3, el tiempo de cálculo necesario para obtener el conjunto de mallas. Las columnas m1-p a m4-p corresponden a las implementaciones de los algoritmos **m1**-**m4** en lenguaje Python. La columna m4adj-c corresponde a la implementación de **m4** en lenguaje C, e incluye también el tiempo para construir la información de adyacencias entre mallas (es decir, averiguar para cada malla con qué otras mallas solapa), que se necesita para determinar la estructura de la matriz. Como se ha explicado más arriba, el método **m2** no se ha podido ejecutar para las redes 3, 4 y 6, y tardó una gran cantidad de tiempo para las redes 2 y 5.

Los tiempos del código Python se han obtenido mediante el comando "`%timeit`" del intérprete `ipython`[2], el cual obtiene el tiempo mínimo entre varias repeticiones de la ejecución (por defecto y también en nuestro caso se usan 3 repeticiones). Además, si la tarea a medir tarda muy poco, cada repetición consiste en un bucle que ejecuta la tarea un número de veces adecuado (determinado automáticamente por el comando "`%timeit`"), midiéndose el tiempo del bucle completo y obteniéndose a continuación el tiempo por iteración. Con esto se trata de evitar que los tiempos medidos se vean excesivamente afectados por la sobrecarga debida a la propia medición de tiempos, así como por otros procesos ejecutándose en la máquina. Para la medición de tiempos en lenguaje C se ha seguido una aproximación similar.

[2] `ipython.org`

Tabla 4.3: Tiempo (segs) empleado en seleccionar el conjunto de mallas.

	m1-p	**m2-p**	**m3-p**	**m4-p**	**m4adj-c**
red 1	0,00107	0,311	0,00290	0,00194	0,000076
red 2	0,0232	1565,0	0,0912	0,0430	0,00219
red 3	0,156	–	0,514	0,287	0,0229
red 4	0,330	–	1,88	0,630	0,0597
red 5	0,0518	7320,0	0,193	0,0927	0,00376
red 6	0,342	–	2,84	0,644	0,0661

Tabla 4.4: Media y desviación estándar de las mediciones de tiempo para la red 1.

	m1-p	**m2-p**	**m3-p**	**m4-p**	**m4adj-c**
media	0,00107	0,311	0,00290	0, 00194	0,000076
desv. est.	0,000006	0,0011	0,000013	0, 0000045	0, 00000063

Se ha comprobado que los tiempos obtenidos de esta manera presentan una variación suficientemente pequeña, como se puede comprobar en la tabla 4.4, obtenida repitiendo 10 veces la medición de tiempos para la red 1.

Volviendo a la tabla 4.3, puede verse que existe enorme diferencia de tiempo entre el método **m2** y el resto de métodos, lo que se debe a que **m2** resuelve el problema de obtener una *base de ciclos mínima*, que como se ha comentado en el apartado 4.2 es de una gran complejidad computacional. A cambio de ello, el método **m2** es el que proporciona una mayor dispersidad, como se ha visto en la tabla 4.2, aunque el tiempo empleado le resta utilidad a efectos prácticos. El método presentado en [17] obtiene, al igual que **m2**, una base de ciclos mínima, y aunque es algo más rápido que este último, sigue siendo extraordinariamente lento comparado con el resto de métodos de la tabla 4.3.

Respecto al resto de métodos, puede verse que **m1** es el más rápido, seguido de **m4**, y finalmente **m3**. La tabla 4.3 muestra claramente que obtener un conjunto de mallas mediante el método **m4** es extremadamente rápido. Eso implica que el método de mallas puede ser competitivo incluso en casos donde se quiere realizar una única simulación, como contraposición a casos en que se busca realizar múltiples simulaciones de redes con una topología común. Otros enfoques, tales como [16] y [24], se centraban principalmente en este segundo caso.

4.8.2. Simulación de las redes

La tabla 4.5 compara el tiempo de simulacion total de los dos métodos, junto con el número de pasos de tiempo y el número total de iteraciones (la suma de las iteraciones de todos los pasos). Puede verse que el método de mallas es más rápido

Tabla 4.5: Comparativa de tiempo total (en segundos), speedup (S), número de pasos de tiempo e iteraciones.

	método	**red 1**	**red 2**	**red 3**	**red 4**	**red 5**	**red 6**
t_tot	GGA	0,0013	0,0048	0,841	1,010	0,803	0,983
t_tot	mallas	0,0011	0,0058	0,772	0,740	0,812	0,811
S	-	1,12	0,83	1,09	1,36	0,99	1,21
pasos t	ambos	27	1	328	145	1153	26
iters	GGA	86	5	415	199	1396	153
iters	mallas	89	5	417	199	1424	126

que GGA en 4 de las 6 redes examinadas, y es prácticamente igual de rápido en otra de ellas.

El número de iteraciones realizadas es muy similar en ambos métodos. Las diferencias que se observan en dicho número se deben a dos factores: por un lado, la solución inicial utilizada es diferente, dado que en el caso del método de mallas dicha solución debe estar balanceada; por otro lado, el tratamiento de las VRP en el método de mallas se hace mediante la aproximación no simétrica descrita en el apartado 4.6.1, más exacta que la aproximación simétrica utilizada en Epanet, lo que puede reducir el número de iteraciones (como ocurre en la red 6).

A continuación se analiza, mediante la tabla 4.6, el tiempo de distintas tareas de la simulación. Estas tareas corresponden, en el método de mallas, al algoritmo 4.1 presentado al principio del capítulo. La mayoría de las tareas tienen su correspondencia en el método GGA, aunque algunas de ellas no son necesarias en este último método. La tabla 4.6 muestra sólo el tiempo de aquellas tareas que son diferentes en los dos métodos analizados aquí. Éstas son:

Inicialización (inic): corresponde a la linea 2 del algoritmo 4.1. Además de crear las estructuras matriciales, realizando la reordenación y descomposición simbólica, en el método de mallas supone previamente seleccionar el conjunto de mallas. En cualquier caso, el tamaño y estructura de las matrices de ambos métodos es distinto, como se explica en el capítulo 2. Esta tarea es más rápida en el método de mallas para las 4 redes más grandes, pese a tener que hacer la selección de las mallas. Esto se explica por el menor tamaño de las matrices.

Obtener un vector de caudales balanceado (blc): corresponde a la línea 5, y sólo es necesario hacerla en el método de mallas.

Actualización del sistema (act): se realiza en la línea 7, y supone la actualización de los coeficientes del sistema lineal. Puede verse que en ambos métodos esta tarea consume una gran cantidad de tiempo respecto del tiempo total, siendo ligeramente más rápida en el caso del método de mallas.

Tabla 4.6: Detalle de tiempos (en segundos) y *speedup* (S) de distintas tareas de la simulación.

	método	**red 1**	**red 2**	**red 3**	**red 4**	**red 5**	**red 6**
t_inic	GGA	0,00007	0,00264	0,063	0,271	0,009	0,263
t_inic	mallas	0,00017	0,00359	0,026	0,069	0,006	0,079
S_inic	-	0,41	0,74	2,40	3,94	1,53	3,33
t_blc	GGA	-	-	-	-	-	-
t_blc	mallas	0,00003	0,00004	0,089	0,085	0,095	0,016
t_act	GGA	0,00059	0,00122	0,327	0,296	0,338	0,379
t_act	mallas	0,00050	0,00119	0,273	0,253	0,291	0,297
S_act	-	1,17	1,03	1,20	1,17	1,16	1,28
t_lin	GGA	0,00041	0,00051	0,278	0,305	0,275	0,276
t_lin	mallas	0,00016	0,00027	0,083	0,056	0,120	0,275
S_lin	-	2,48	1,90	3,34	5,45	2,29	1,01
t_caud	GGA	0,00006	0,00010	0,046	0,040	0,049	0,034
t_caud	mallas	0,00009	0,00023	0,101	0,099	0,100	0,064
S_caud	-	0,65	0,42	0,46	0,40	0,49	0,52
t_alt	GGA	-	-	-	-	-	-
t_alt	mallas	0,00003	0,00012	0,076	0,082	0,072	0,051

Solver lineal (lin): resolución del sistema lineal (línea 8). La implementación del método de mallas utiliza la variante no simétrica para la modelización de VRPs, lo que implica que si la red tiene ese tipo de elementos, el sistema lineal tendrá una estructura como la de la ecuación (4.19), resolviéndose de acuerdo con las expresiones (4.24) y (4.25). El número de válvulas reguladoras de presión de cada red se muestra en la columna VP de la tabla 4.1.

En esta tarea el método de mallas destaca claramente, con un speedup que llega a ser de 5,45 para la red 4. La única excepción es la red 6, donde el tiempo es prácticamente el mismo para ambos métodos, lo que cabe atribuir al número de VRPs de la red (10), que hacen que la matriz tenga una parte no simétrica de tamaño considerable.

Obtener nuevos caudales (caud): consiste en actualizar los caudales (línea 9), de acuerdo con las correcciones de caudal de las mallas. Este proceso de actualización es más costoso en el método de mallas que en GGA.

Obtener nuevas alturas (alt): en el método de mallas se deben obtener las nuevas alturas piezométricas a partir de los nuevos caudales (línea 10). En el método GGA esto no es necesario, ya que las alturas se han obtenido como solución del sistema linea.

En resumen, el método de mallas destaca en la resolución del sistema lineal. Por contra, los puntos débiles son la obtención de nuevos caudales y la necesidad de realizar algunas tareas adicionales que no se hacen en el método GGA.

4.9. Conclusiones

Este capítulo presenta aportaciones importantes al método de mallas, que lo convierten en un método competitivo frente al método GGA.

En primer lugar, se aborda el problema de la selección de un conjunto de mallas que tengan el menor solapamiento posible, y por tanto produzcan un sistema lineal con una matriz de gran dispersidad. Si bien existen algoritmos que encuentran un conjunto óptimo de mallas, el tiempo de cálculo necesario es prohibitivo para redes de tamaño medio o grande. Frente a esto, en este capítulo se presentan dos algoritmos (**m3** y **m4**) que son capaces de proporcionar un conjunto de mallas cercano al óptimo, con un tiempo de cálculo muy inferior. En especial, el método **m4** es el que proporciona mejores resultados.

En segundo lugar, se considera la problemática de la modelización de válvulas. Por un lado, se presentan formulaciones para considerar líneas temporalmente cerradas y VCQ. Se consideran dos formulaciones diferentes, mostrándose la relación entre ambas. Frente a otros métodos existentes, los presentados en esta tesis no precisan modificar el conjunto de mallas seleccionado, y por tanto mantienen la estructura de la matriz del sistema lineal.

Por otro lado, se presentan formulaciones para la modelización de VRPs y VSPs, lo que supone una mayor dificultad por la pérdida de simetría de la matriz del

sistema. Se desarrollan dos formulaciones para la modelización de VRPs/VSPs, las cuales tienen en común el hecho de que evitan modificar el conjunto de mallas seleccionado. La primera formulación presentada añade una parte no simétrica a la matriz básica simétrica, resolviendo el sistema por bloques. Esta separación de la matriz en partes hace posible el aprovechamiento de la propiedad de simetría de una de las partes. Además, el sistema planteado corresponde exactamente a la iteración de Newton-Raphson, a diferencia de la formulación usada en Epanet, que introduce algunas aproximaciones para conseguir que el sistema sea simétrico. Eso significa que el número de iteraciones puede reducirse.

La segunda formulación para la modelización de VRPs/VSPs consigue llevar al método de mallas las aproximaciones que realiza Epanet sobre el sistema lineal para conseguir que la matriz mantenga la simetría en presencia de este tipo de válvulas. La formulación desarrollada produce las mismas iteraciones (salvo errores de redondeo) que la usada por Epanet. Frente a otros métodos existentes para formular las VRPs/VSPs en el método de mallas, la ventaja clara es la simetría de la matriz. Si bien eso se consigue a costa de sacrificar exactitud en el sistema lineal, se consigue el mismo nivel de exactitud que Epanet.

Teniendo en cuenta las contribuciones presentadas en el capítulo, se ha desarrollado una implementación completa del método de mallas, basada en el código de Epanet. Esto ha permitido poder evaluar las propuestas, comprobando su efectividad y su corrección.

Los resultados presentados permiten comprobar que los métodos de selección de mallas producen una reducción muy importante en el número de elementos no nulos de la matriz, respecto al método GGA. Además, se comprueba que el tiempo de cómputo necesario es muy pequeño.

Respecto a los tiempos de simulación, se presentan resultados con seis redes de distribución. En cuatro de ellas el método de mallas es más rápido que GGA, con ganancias de velocidad entre 9 % y 36 %. En otra de ellas los tiempos de ambos métodos son prácticamente iguales, y sólo una de las redes se simula más rápidamente mediante GGA.

El análisis de tiempos de las distintas tareas de la simulación deja ver que la resolución del sistema lineal es la parte que más se beneficia del método de mallas, con *speedups* que, exceptuando una de las redes, van desde 1,9 hasta 5,45. Esto es una consecuencia directa del buen comportamiento del algoritmo de selección de mallas **m4**, que produce una matriz con un alto grado de dispersidad.

La ganancia de prestaciones conseguida es especialmente importante en contextos como el diseño de redes mediante un proceso de optimización, que pueden requerir la simulación de miles o millones de redes ligeramente diferentes.

Capítulo 5

Algoritmos paralelos en memoria distribuida para simulación de redes y minimización de fugas

En este capítulo se describe en primer lugar la implementación de algoritmos paralelos en memoria distribuida para la simulación de redes de distribución de agua, comprendiendo tanto la simulación hidráulica (de caudales y presiones) como la de calidad del agua. En segundo lugar, se describe un algoritmo paralelo para la minimización de fugas.

Se ha tomado como código de partida el del software Epanet, en concreto su versión 1.1e. Esto determina los algoritmos secuenciales correspondientes a los dos tipos de simulación considerados, que son el método de gradiente global (GGA) para la simulación hidráulica, y el método de elementos de volumen discreto (DVEM) para la de calidad del agua. La elección de la versión 1.1e para este capítulo se debe a que ésta era la última existente en el momento en que se inició el trabajo correspondiente a esta parte. En los demás capítulos de la tesis se considera la versión más reciente de Epanet (2.00.12).

Para la implementación de los algoritmos paralelos se ha utilizado la herramienta MPI [52] (ver apartado 3.4).

5.1. Aproximación de paralelización de la simulación hidráulica

En el capítulo 2 se presentaron algoritmos correspondientes a la simulación hidráulica. La simulación llevada a cabo en el software Epanet se basa en los al-

goritmos 2.1 y 2.2 de aquel capítulo, el primero de los cuales describe el proceso de simulación hidráulica en periodo extendido como una secuencia encadenada de problemas hidráulicos estáticos, mientras que el segundo corresponde al método GGA para la solución de cada problema estático. El proceso completo resultante implementado en Epanet se presenta en el algoritmo 5.1.

Algoritmo 5.1 Simulación hidráulica en periodo extendido mediante GGA

```
1: leer datos de la red desde un fichero de entrada
2: reordenar nudos, realizar descomposición simbólica
3: mientras t < t_final hacer
4:    calcular demandas y aplicar controles en válvulas y bombas
5:    mientras no convergencia hacer
6:       actualizar matriz y vector del sistema lineal (2.15)
7:       resolver sistema lineal anterior, obteniendo h^{k+1}
8:       obtener nuevos caudales q^{k+1} mediante ec. (2.16)
9:       actualizar estado de válvulas, bombas y tuberías
10:   fin mientras
11:   escribir resultados del tiempo t en un fichero
12:   avance de t, actualizar niveles de depósitos
13: fin mientras
```

De las distintas operaciones de la simulación hidráulica que aparecen en este algoritmo, la que supone una mayor dificultad desde el punto de vista de su paralelización es la solución del sistema lineal (línea 7), estando también entre las tareas que más tiempo consumen. Se trata de un sistema de ecuaciones lineales con una matriz de coeficientes dispersa, simétrica y definida positiva, propiedades que deben ser aprovechadas en toda implementación eficiente.

Por lo tanto, a la hora de construir un algoritmo paralelo para la simulación hidráulica, y en concreto para el problema estático, el punto principal es la paralelización adecuada del proceso de resolución del sistema lineal. El resto de operaciones a realizar en la resolución del problema estático deben acomodarse a la distribución de datos que resulte más adecuada para la resolución del sistema lineal.

La resolución de sistemas lineales dispersos de gran dimensión puede abordarse por medio de métodos directos o bien mediante métodos iterativos. Los métodos directos tienen la ventaja de su generalidad y robustez. Los métodos iterativos pueden llegar a ser más eficientes que los directos, dependiendo de la naturaleza del problema y las particularidades del sistema a resolver, aunque requieren usualmente el uso de técnicas de precondicionado, cuyo coste puede llegar a superar al de los métodos directos.

Epanet utiliza métodos directos, y concretamente la descomposición de Cholesky. Algunos autores han intentado aplicar métodos iterativos para los sistemas lineales que surgen en la simulación hidráulica [73, 29, 34], pero han obtenido escaso éxito en términos de prestaciones. Por esta razón, en esta tesis nos centramos en los métodos directos y, más concretamente en métodos basados en la descomposición de Cholesky.

En este contexto, hay que resaltar que la factorización de una matriz densa

puede realizarse de forma eficiente en computadores paralelos de memoria distribuida, pero el problema de factorización de matrices dispersas en paralelo reviste una mayor dificultad. Sin embargo, se han producido avances que han permitido desarrollar algoritmos paralelos altamente escalables para la factorización de Cholesky de matrices dispersas, basados en *algoritmos multifrontales*. En [35, 36] se muestran algoritmos paralelos, para arquitecturas de malla e hipercubo, tan escalables como los algoritmos para matrices densas, en una amplia clase de matrices dispersas, mejorando el estado del arte existente hasta el momento. Estos algoritmos han sido implementados en la librería PSPASES [40].

En esta tesis se aborda la aplicación de los algoritmos de [35, 36] al análisis hidráulico estático, es decir, a la resolución del sistema lineal (2.15). Esto se hace por medio de la librería PSPASES.

5.2. Algoritmos multifrontales paralelos

El problema consiste en resolver el sistema $Ax = b$, donde A es una matriz dispersa, simétrica y definida positiva. La factorización de Cholesky se compone de cuatro fases consecutivas: reordenación para minimización de llenado, factorización simbólica, factorización numérica y resolución de sistemas triangulares.

Durante la fase de *reordenación* se calcula una matriz de permutación P para que el llenado de la matriz PAP^T, esto es, el incremento del número de elementos no nulos producido al realizar su factorización, sea pequeño. Durante la fase de *factorización simbólica* se determina la estructura de dispersidad del factor triangular de Cholesky L, lo cual permite incrementar las prestaciones de la fase de factorización numérica. Las operaciones necesarias para calcular los valores de L que satisfacen la igualdad $PAP^T = LL^T$ se realizan durante la fase de *factorización numérica*. Por último, las solución de $Ax = b$ se calcula mediante la resolución de dos *sistemas triangulares*, $Ly = b'$ seguido de $L^T x' = y$, donde $b' = Pb$. El primero de los sistemas se resuelve mediante el algoritmo de eliminación progresiva, mientras que en el segundo se utiliza el algoritmo de sustitución regresiva. La solución final, x, se obtiene como $x = P^T x'$.

En el caso de la simulación hidráulica, las etapas de reordenación y factorización simbólica sólo es necesario hacerlas una vez, como parte del proceso de inicialización previo a la simulación. Esto es debido a que el patrón de dispersidad de la matriz no cambia entre los diferentes pasos de tiempo, ya que depende de la estructura de la red, que es invariante en el tiempo.

Básicamente, los algoritmos de las distintas fases están dirigidos por el *árbol de eliminación* de la matriz A [47]. El árbol de eliminación de una matriz A, simétrica de orden n, con factor de Cholesky L, es un árbol con n nudos $\{1, 2, \ldots, n\}$, tal que el nudo p es el padre del nudo j si se cumple que

$$p = \text{mín}\left\{i \mid i > j, l_{i,j} \neq 0\right\},$$

es decir, p es el índice del primer elemento no nulo en la columna j de L. Llamando $T[x]$ al subárbol cuya raíz es el nudo x, si $y \in T[x]$ se dice que y es un *descendiente*

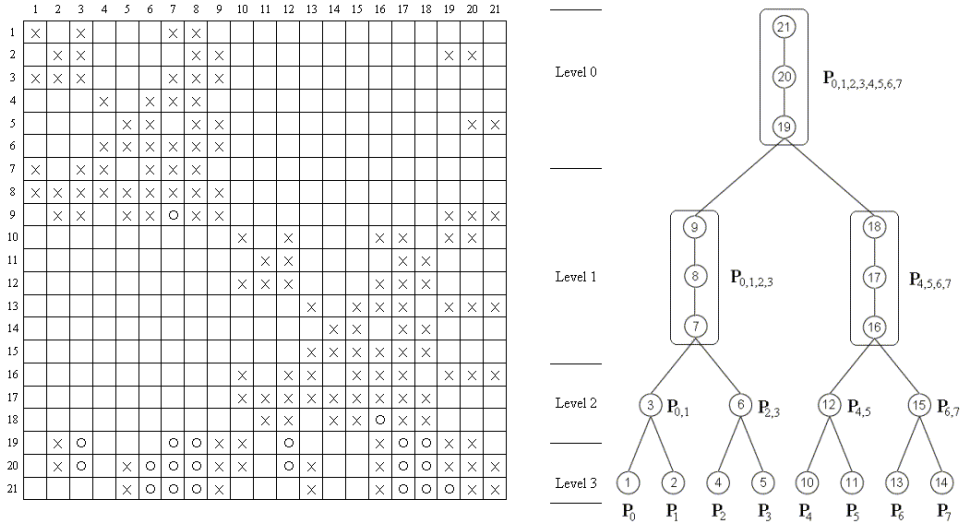

Figura 5.1: Árbol de eliminación de una matriz simétrica dispersa

de x y x es un *ascendente* de y.

La importancia del árbol de eliminación estriba en que indica las dependencias entre la eliminación de las distintas variables del sistema lineal, o sea, entre el cálculo de las distintas columnas de L. En particular, [47] muestra que el conjunto de columnas que modifican la columna j de L es un subconjunto de $T[j]$. De esta manera, el árbol de eliminación define un orden parcial en la eliminación de las variables del sistema (o el cálculo de las columnas de L), de manera que todas las variables correspondientes a los hijos de un nudo deben eliminarse antes de eliminar la variable del nudo padre. Puesto que el orden definido es parcial, en general existirán muchos órdenes diferentes consistentes con el árbol de eliminación, o dicho de otra manera, hay nudos que se pueden procesar concurrentemente, abriendo la puerta a la computación paralela.

La figura 5.1 muestra un ejemplo de una matriz simétrica y su correspondiente árbol de eliminación. Un conjunto de nodos consecutivos del árbol tales que cada uno tiene un solo hijo se llama un *supernodo* (por ejemplo, los nodos 7, 8 y 9, o bien 16, 17 y 18, de la figura 5.1). Agrupando los nodos de esta manera se forma un árbol de eliminación supernodal, que es un árbol binario con $\lceil \log p \rceil$ niveles, donde p es el número de procesos participantes.

Para factorizar la matriz dispersa en paralelo, se asignan partes de árbol de eliminación a los procesos usando una estrategia de asignación de subárbol a subcubo. En concreto, el supernodo superior se asigna a todos los procesos. Los dos subárboles de la raíz se asignan a subcubos de $p/2$ procesos cada uno. Cada subárbol se particiona de nuevo de forma recursiva usando la misma estrategia. De esta mane-

ra, los p subárboles del nivel $\log p$ se asignan a procesos individuales. La figura 5.1 muestra el árbol de eliminación de una matrix simétrica dispersa, con la estrategia de distribución de subárbol a subcubo sobre 8 procesos (P_0 a P_7). Los elementos no nulos de la matriz original se representan median el símbolo "$\times$", y los elementos no nulos producidos por el llenado se representan mediante "$\circ$".

5.2.1. Reordenación de la matriz

Para obtener una reordenación de la matriz de coeficientes que reduzca su llenado se usa una versión secuencial del algoritmo de *Disección Anidada Multinivel* (DAM) [12, 37]. La estrategia de reordenación es un factor clave que condiciona la eficiencia tanto en implementaciones secuenciales como paralelas. En el primer caso, el objetivo es minimizar los requisitos de memoria y el número de operaciones a realizar durante la factorización de la matriz de coeficientes. En el caso paralelo, debe tenerse en cuenta además que el grado de paralelismo durante la factorización debe ser alto.

El algoritmo de mínimo grado (MG), usado en Epanet, ha demostrado producir muy buenas reordenaciones en algoritmos secuenciales. Sin embargo, las reordenaciones basadas en disección anidada proporcionan mayor concurrencia y mejor equilibrio de carga que MG. Además, el algoritmo DAM es normalmente más rápido que el de MG para matrices de gran tamaño y puede ser paralelizado de forma efectiva, mientras que el de MG es de naturaleza secuencial. De hecho, en las pruebas realizadas en esta tesis para la simulación de redes de distribución de agua, se ha constatado que el algoritmo DAM no sólo es preferible al de MG en implementaciones paralelas, sino también en el caso secuencial, especialmente cuando se tienen matrices de gran tamaño. El algoritmo de DAM reduce el número de elementos no nulos en la matriz factorizada, acelerando las fases de factorización y resolución de sistemas triangulares.

El algoritmo de DAM se basa en la técnica de particionado de grafos mediante *bisección multinivel* [12]. Básicamente, la bisección multinivel implica la progresiva reducción de un grafo hasta unos pocos cientos de vértices, seguida del particionado de este grafo de pequeño tamaño, y finalmente la proyección de las particiones de vuelta al grafo original, mediante su refinamiento gradual. En la sección de la simulación de la calidad del agua se proporcionan más detalles al respecto.

5.2.2. Factorización simbólica

En esta segunda etapa se calcula la estructura de dispersidad del factor triangular de Cholesky L. Eso se lleva a cabo mediante un algoritmo paralelo en el que el árbol de eliminación supernodal se distribuye entre los procesos usando un esquema de subárbol a subcubo, y la submatriz de coeficientes de cada supernodo se distribuye usando una estrategia cíclica por bloques basada en máscaras de bits. La estructura de L se obtiene de abajo a arriba, calculando primero la estructura de dispersión de los nodos hoja y enviándola hacia arriba en el árbol de eliminación a los procesos que almacenan el supernodo del nivel inmediatamente superior. Estos procesos calculan la estructura de elementos no nulos de su supernodo y la combinan con la estructura

recibida de sus nodos hijo. Además, la estructura de dos subárboles sin solapamiento entre ellos puede ser calculada concurrentemente.

5.2.3. Factorización numérica

En la tercera etapa de la resolución del sistema lineal se lleva a cabo la factorización numérica en paralelo, utilizando un algoritmo multifrontal [37, 47], calculando con ello los valores de L.

Existen dos niveles de paralelismo en esta fase. En primer lugar, el procesamiento de las matrices frontales asociadas a dos subárboles que no se solapan entre sí puede ser tratado como dos tareas completamente independientes. En segundo lugar, dado que la cantidad de cálculo implicado en el ensamblaje y eliminación parcial de una matriz frontal puede ser sustancial, se debe considerar una subdivisión adicional de la tarea en subtareas más pequeñas.

De nuevo el árbol de eliminación supernodal se distribuye entre los procesos siguiendo la misma estrategia subárbol a subcubo mencionada anteriormente, y la matriz frontal asociada a cada supernodo se asigna a cada proceso usando el mismo esquema cíclico por bloques basado en máscaras de bits. Con esta distribución, las operaciones de *extensión-adición* del algoritmo se calculan en paralelo, y cada proceso intercambia aproximadamente la mitad de sus datos con el vecino del otro subcubo. La operación de factorización paralela en cada supernodo se basa en la descomposición de Cholesky por columnas para matrices densas.

5.2.4. Resolución de sistemas triangulares

Finalmente, se deben resolver dos sistemas triangulares, concretamente $Ly = b'$ en primer lugar, y a continuación $L^T x' = y$, donde $b' = Pb$, y la solución final es $x = P^T x'$. Para ello se usan versiones paralelas de los algoritmos de eliminación progresiva y sustitución regresiva, respectivamente. Estas versiones paralelas de nuevo están dirigidas por el árbol de eliminación supernodal, utilizando para L la misma correspondencia subárbol a subcubo y la distribución cíclica por bloques en los supernodos que ha sido mencionada en las dos fases de factorización. Mientras que en la eliminación progresiva el cálculo avanza de abajo a arriba, la sustitución regresiva procede al contrario, desde el nudo superior hasta las hojas. En ambos sistemas triangulares se usan algoritmos densos *pipeline* de dos dimensiones.

5.3. Implementación paralela de la simulación hidráulica

En el apartado anterior hemos tratado las técnicas utilizadas por los algoritmos paralelos para la resolución de sistemas de ecuaciones lineales dispersos mediante una aproximación multifrontal. Una de las implementaciones de estos algoritmos que consigue mejores prestaciones es la librería PSPASES [40], razón por la cual esta fue la herramienta elegida en esta tesis en el contexto de la simulación hidráulica.

En este apartado consideramos la implementación paralela de la simulación hidráulica en su conjunto, la cual sigue en líneas generales los mismos pasos de la implementación secuencial mostrada en el algoritmo 5.1.

Empezamos este apartado comentando la distribución de los datos utilizada en la aplicación, para pasar a continuación a describir la paralelización de cada una de las fases del algoritmo 5.1.

Distribución de datos: nudos y líneas

Todos los procesos que colaboran en la simulación hidráulica disponen de los datos de entrada de todos los elementos de la red: nudos, líneas, depósitos, bombas, válvulas, reglas de control, patrones de tiempo y curvas. Sin embargo, cada proceso trabajará sólo con algunos de los nudos y líneas, produciendo los resultados correspondientes a los mismos.

La asignación de los nudos y líneas a los procesos viene determinada por la distribución de la matriz. Aunque internamente la librería PSPASES utiliza una distribución subárbol a subcubo, desde el punto de vista de la aplicación se utiliza una distribución por bloques de filas. Esto hace que la distribución de los nudos de consumo entre los procesos se realice también por bloques, de manera que el número de nudos de consumo que recibe cada proceso es aproximadamente n/p (siendo n el número de nudos y p el número de procesos).

La distribución de las líneas entre los procesos se hace de manera que un proceso se ocupe de todas las líneas conectadas directamente con alguno de sus nudos (tanto nudos de consumo como depósitos). Nótese que si los dos extremos de una línea están en procesos diferentes, la línea será tratada por ambos procesos, lo que supone cierta repetición de cálculos, pero evita comunicaciones adicionales en cada iteración del método GGA.

A continuación presentamos los detalles de implementación de las distintas fases de la simulación.

Lectura y difusión inicial de los datos

Antes de empezar la simulación de la red, se lleva a cabo la lectura del fichero de entrada y la difusión de los datos contenidos en él a todos los procesos (línea 1 del algoritmo 5.1). El proceso P_0 es el encargado de leer el fichero de entrada correspondiente a la red hidráulica que se quiere analizar, y difundir su contenido al resto de procesos. La información difundida comprende diversos aspectos: estructura topológica de la red, características de cada línea de la misma (tipo de línea, diámetro, longitud, coeficientes de pérdida, etc.), características particulares de las bombas, depósitos y válvulas, reglas de operación de la red, patrones temporales, curvas, así como opciones de simulación (paso de tiempo hidráulico, duración de la simulación, precisión requerida, etc.). Al finalizar esta fase, esta información queda replicada en todos los procesos, para evitar comunicaciones adicionales durante la simulación.

Los datos de los elementos de la red están contenidos en arrays de estructuras, para cuya difusión se hace uso de las capacidades de MPI para definición de *tipos de datos derivados* (véase apartado 3.4.4). Por ejemplo, la información de todos los nudos de la red se almacena en el array `Node`, cuyos elementos son estructuras de tipo `Snode`. Para la comunicación de este array, en primer lugar se define un tipo de datos MPI correspondiente a `Snode`, y a continuación se envía/recibe el array como si se tratara de un array de elementos de cualquier tipo simple.

Hay que tener en cuenta que algunas estructuras contienen referencias (punteros) a datos almacenados fuera de la estructura, lo que requiere un tratamiento especial, dado que lo que se quiere enviar no es la referencia, sino los datos apuntados por la misma. Para la comunicación de esta información se hace uso del *empaquetamiento de datos* de MPI. Por ejemplo, la estructura `Snode` contiene una referencia a una estructura `Ssource`, que almacena los datos de la fuente de calidad asociada al nudo, o es una referencia vacía en caso de que el nudo no tenga una fuente asociada. En el ejemplo citado, se empaquetan en un *buffer* los datos de las fuentes de calidad de todos los nudos de la red, junto con el nudo asociado a cada fuente, y a continuación se envía el contenido del buffer a todos los procesos. Cuando un proceso recibe el envío, desempaqueta la información de cada fuente y de su nudo correspondiente, colocando en el nudo la referencia a la fuente.

Reordenación de nodos y descomposición simbólica

Una vez se ha difundido esta información, se genera en paralelo la estructura de la matriz de coeficientes del sistema, siguiendo una distribución por bloques de filas con almacenamiento en fila comprimida (*Compressed Sparse Row*). Este proceso corresponde a la línea 2 del algoritmo 5.1. Nótese que cada fila/columna de la matriz corresponde a un nudo de la red, y cada elemento no nulo corresponde a una línea de la misma.

Conviene apuntar que la generación de la estructura de la matriz se hace previamente a las fases de reordenación y descomposición simbólica, lo que puede parecer extraño, puesto que estas fases alteran la estructura de la matriz. La razón es que la librería PSPASES mantiene dos representaciones distintas de la matriz:

- Una representación externa, manejada por la aplicación, la cual no se ve afectada por la reordenación aplicada, ni por la descomposición simbólica. En esta representación la distribución es por bloques de filas.

- Una representación interna, manejada por la librería, sobre la cual se aplica la reordenación y la descomposición simbólica. En esta representación, se utiliza la distribución subárbol a subcubo/submalla mencionada en el apartado 5.2.

Tras construir la estructura de la matriz, se efectúa la reordenación de los nudos, mediante el algoritmo de Disección Anidada Multinivel implementado en la librería PSPASES y descrito en el apartado 5.2.1. A continuación, se lleva a cabo la descomposición simbólica de la matriz, de acuerdo con lo comentado en el apartado 5.2.2.

Actualización de demandas y aplicación de controles

Esta tarea corresponde a la línea 4 del algoritmo 5.1. Cada nudo de consumo puede tener varios *patrones de demanda* asociados, que imponen una modulación temporal sobre un valor base. Para la actualización de la demanda en paralelo, cada proceso realiza el cálculo de la nueva demanda sobre sus nudos locales. Hay que tener en cuenta que los patrones de demanda están replicados en todos los procesos.

De la misma manera, cada proceso se encarga de actualizar los cambios impuestos en cada instante de tiempo sobre los estados y consignas de sus válvulas y bombas locales.

No son necesarias comunicaciones en esta fase.

Actualización del sistema lineal

La actualización del sistema de ecuaciones lineales en cada iteración (línea 6 del algoritmo 5.1) implica el cálculo de dos valores, p_k y y_k, para cada línea, y el ensamblaje de los mismos en la matriz y el vector del sistema [58]. Estos valores están relacionados con la pérdida de carga que se produce en una línea, expresada como función del caudal que circula por ella.

Ese proceso de actualización se analiza con detalle en el capítulo 6, donde se presenta un algoritmo secuencial (algoritmo 6.3), y varios algoritmos paralelos sobre OpenMP. En este apartado se pretende sólo dar una idea de la aproximación de paralelización de esta tarea sobre plataformas de memoria distribuida.

En dicha paralelización, cada proceso se encarga de actualizar el bloque de filas (tanto de la matriz A como del vector de términos independientes b) correspondiente a sus nudos locales. La matriz está almacenada en el formato requerido por la librería PSPASES, en concreto en fila comprimida (CSR) y sin tener en cuenta la simetría (es decir, se mantiene tanto la parte triangular inferior como la superior).

Para actualizar su parte del sistema, cada proceso necesita los caudales de todas las líneas conectadas directamente con alguno de los nudos locales del proceso. Estas líneas son justamente las que le corresponden al proceso (líneas locales) de acuerdo con la distribución de líneas comentada anteriormente. Eso permite realizar la actualización del sistema lineal sin necesidad de comunicaciones.

El algoritmo 5.2 hace una descripción más formal del proceso, aunque para su comprensión puede ser conveniente leer primero la sección 6.3.1. Hay que tener en cuenta que el algoritmo debe ser ejecutado por cada uno de los procesos que intervienen en la simulación, de manera que cada uno de ellos hace el trabajo correspondiente a sus datos locales (líneas y nudos locales).

El bloque de líneas 6-11 del algoritmo 5.2 realiza la actualización de los elementos no diagonales correspondientes a una línea k. Se han obviado detalles sobre el almacenamiento disperso (CSR) de la matriz. En realidad, el acceso a los elementos no diagonales a_{ij}, a_{ji} de una línea k se haría por medio de una tabla que obtendría las posiciones de los elementos no nulos correspondientes de la línea. La actualización de los elementos diagonales y elementos del vector b correspondientes a los extremos de la línea k se lleva a cabo en las líneas 12-19. De nuevo el acceso al elemento a_{ii} se

Algoritmo 5.2 Algoritmo paralelo sobre memoria distribuida para actualización del sistema lineal en GGA

Entrada: Datos de líneas locales: coefs. de resistencia $(r, \rho, \cdots)$, caudales (q). Demandas de nudos locales (c). Almacenamiento disperso para la matriz A (parte local).

Salida: Matriz A, vector b

1: /* *Algoritmo a ejecutar en cada proceso* */
2: inicializar bloque local de A, b, a cero
3: **para todo** línea local k **hacer**
4: calcular p_k, y_k (coeficientes relacionados con pérdida de carga)
5: sean i, j los nudos inicial y final, resp. de la línea k
6: **si** i, j son ambos nudos de consumo **entonces**
7: **si** nudo i es local **entonces**
8: $a_{ij} \leftarrow a_{ij} - p_k$
9: **fin si**
10: ... idem para nudo j
11: **fin si**
12: **si** el nudo i es local **entonces**
13: **si** el nudo i es un nudo de consumo **entonces**
14: $a_{ii} \leftarrow a_{ii} + p_k, \qquad b_i \leftarrow b_i - q_k + y_k$
15: **si no si** el nudo j es local **entonces**
16: $b_j \leftarrow b_j + p_k h_i$
17: **fin si**
18: **fin si**
19: ... se repite el bloque **si** anterior, ahora para el nudo j
20: **fin para**
21: **para todo** nudo local i **hacer**
22: $b_i \leftarrow b_i - c_i$
23: **fin para**

realizaría por medio de una tabla o array que proporcionaría la posición del elemento a partir del índice de línea.

Resolución del sistema lineal

Una vez se han actualizado los coeficientes del sistema lineal (2.15), su resolución se lleva a cabo mediante las fases de factorización numérica y resolución de sistemas triangulares, que se efectúan mediante los algoritmos multifrontales de PSPASES como se ha descrito más arriba. Esto corresponde a la línea 7 del algoritmo 5.1.

El vector h^{k+1} solución del sistema se obtiene distribuido por bloques de filas, al igual que la matriz y el vector parte derecha. Sin embargo, tras resolver el sistema cada proceso comunica su parte del vector a los demás procesos (mediante una operación `MPI_Allgatherv`) de forma que al acabar esta tarea todos los procesos acaban disponiendo del vector h^{k+1} completo.

Actualización de caudales

El siguiente paso en el método GGA consiste en actualizar los caudales a partir de las nuevas alturas piezométricas (línea 8 del algoritmo 5.1). Esto se hace mediante la aplicación del siguiente incremento sobre cada una de las líneas:

$$\Delta q_k = -y_k + p_k(h_i - h_j), \qquad 1 \leq k \leq m \tag{5.1}$$

donde i, j son los nudos en los extremos de la línea k.

Además, se realiza el cálculo del siguiente cociente, que corresponde al cambio relativo de los caudales en la iteración actual, y se utiliza para la comprobación de convergencia del método.

$$\frac{\sum_{k=1}^{m} |\Delta q_k|}{\sum_{k=1}^{m} |q_k|} \tag{5.2}$$

En la actualización de los caudales de acuerdo con la expresión (5.1), cada proceso actualiza los caudales de las líneas que le corresponden, para lo cual necesita las alturas piezométricas de los nudos extremos de las mismas. Esto no supone ningún problema ya que tras la resolución del sistema lineal todos los procesos disponen de las alturas piezométricas de todos los nudos de consumo.

Respecto al cálculo del cociente de la ec. (5.2), cada proceso calcula una parte de los sumatorios del numerador y denominador, combinándose las partes de los distintos procesos por medio de una operación de reducción (`MPI_Allreduce`) para obtener el resultado final en todos los procesos. Todos los procesos deben conocer el resultado para saber cuándo se alcanza la convergencia. Hay que tener en cuenta que una misma línea puede ser compartida por dos procesos distintos (los propietarios de sus nudos extremos), por lo que al realizar la suma sólo uno de los dos (por ejemplo, el propietario del nudo inicial) debe tener en cuenta la aportación de la línea.

Actualización del estado de válvulas y bombas

En esta fase (línea 9 del algoritmo 5.1) se comprueba si se debe cambiar el estado de alguna línea de la red, teniendo en cuenta los nuevos valores obtenidos de caudales y alturas piezométricas. Esto incluye posibles cambios en el estado de válvulas reductoras/sostenedoras de presión (VRP/VSP), válvulas de control de caudal (VCQ), válvulas antirretorno, bombas, líneas conectadas a depósitos llenos/vacíos, o bien líneas afectadas por alguna regla de control que dependa de la altura piezométrica de algún nudo.

Todos los procesos deben conocer si hay algún cambio de estado de alguna línea, ya que eso afecta a la comprobación de terminación del método iterativo GGA. Esto se hace mediante una operación de reducción a todos (`MPI_Allreduce`).

Conviene aclarar que en la implementación del método GGA realizada en Epanet, los cambios de estado de válvulas y bombas no suponen un cambio de la estructura de la matriz del sistema lineal. Por el contrario, sólo afectan a los valores de los elementos del sistema lineal.

Avance hasta el siguiente instante de tiempo

Para pasar de un instante de tiempo al siguiente (línea 12 del algoritmo 5.1) hay que realizar operaciones como escritura de resultados en fichero, cálculo de la duración del paso de tiempo en curso, y actualización de los niveles de depósitos. La duración del paso de tiempo vendrá dada por el siguiente evento que obligue a considerar un nuevo paso de simulación, como refleja la siguiente fórmula:

$$\Delta t = \text{mín}(\Delta t_H, t_{\text{dem}} - t, t_{\text{inf}} - t, t_{\text{dep}} - t, t_{\text{cont}} - t)$$

donde Δt es la duración del paso de tiempo, t es el instante de tiempo actual, Δt_H es la duración (máxima) del paso de tiempo hidráulico, t_{dem} es el tiempo de la siguiente actualización de las demandas, t_{inf} el tiempo de la siguiente escritura de informe, t_{dep} el tiempo hasta el llenado/vaciado de algún depósito y t_{cont} el tiempo hasta la activación de alguna regla de control.

Respecto a las comunicaciones de datos, la de escritura de resultados supone la comunicación de información tanto de nudos como de líneas, desde los distintos procesos al proceso 0. Además, para el cálculo de la duración del paso de tiempo cada proceso calcula un valor local y a continuación se obtiene el valor final mediante una operación de reducción a todos (`MPI_Allreduce`).

5.4. Simulación paralela de la calidad del agua

En el capítulo 2 se presentaron distintos métodos de simulación de la calidad del agua, y en especial el Método de Volúmenes Discretos (DVEM), usado en Epanet 1.1. En este capítulo consideramos la paralelización sobre memoria distribuida de este método.

Como se comentó en el capítulo 2, en la simulación de calidad cada paso de tiempo hidráulico se divide en pasos más pequeños. Dado que estos pequeños pasos

deben llevarse a cabo de manera secuencial, la idea básica de paralelización de la simulación consiste en dividir la red de distribución en diferentes partes, una para cada proceso de la aplicación. De esta manera, cada paso del DVEM puede realizarse en paralelo. Como el autor de la tesis describe en [5], la paralelización se basa en dos tareas: el particionado inicial de la red, por una parte, y el algoritmo paralelo para cada paso de tiempo del DVEM, por otra. Estas dos tareas se discuten en los siguientes apartados.

5.4.1. Particionado de la red

Una red de abastecimiento de agua puede verse como un grafo, en el que los vértices son los nudos de la red, y los arcos son las líneas de la misma (tuberías, bombas y válvulas). Resulta adecuado por lo tanto utilizar algoritmos de particionado de grafos para dividir la red en distintas partes, de manera que éstas tengan un número similar de nudos, y el número de líneas que conecten partes distintas sea mínimo (como consecuencia de ello, se reducirá la comunicación entre procesos). Este particionado inicial de la red juega un papel esencial en la minimización de las comunicaciones y el balance de la carga computacional.

Los algoritmos multinivel [12, 38] son una de las técnicas más efectivas de particionado de grafos, razón por la cual se han utilizado en esta tesis, tanto en lo relativo a la simulación hidráulica (para la ordenación de la matriz de coeficientes del sistema lineal), como en este caso en el contexto de la simulación de la calidad del agua. En concreto, en este segundo caso se utiliza el método de *Bisección Recursiva Multinivel* [41, 42], por medio del cual se puede llevar a cabo un buen particionado de un grafo de forma rápida. Dado que el particionado de la red se realiza una sola vez y no es una tarea que consuma mucho tiempo (menos de un cuarto de segundo para las redes con las que se ha trabajado) se ha aplicado una versión secuencial de este algoritmo.

Este algoritmo, como otros algoritmos de particionado multinivel, siguen un proceso de tres fases como se muestra en la figura 5.2. En la primera fase se reduce progresivamente (atravesando distintos niveles de detalle) el tamaño del grafo mediante la fusión de vértices y arcos, obteniéndose finalmente un grafo con algunos cientos de vértices. En la segunda fase se realiza una bisección del grafo reducido, obteniendo dos subgrafos con un número mínimo de arcos que los conectan entre sí y un número similar de vértices en cada subgrafo. Finalmente la última fase consiste en proyectar esta partición sobre el grafo original, a través de los distintos niveles existentes, mediante un proceso de refinamientos sucesivos. En cada refinamiento hay más grados de libertad, que se usan para mejorar la calidad de la partición mediante el movimiento de elementos de una parte a la otra.

El proceso completo produce de manera rápida una buena partición del grafo. El particionado obtenido de esta manera establece la distribuición de los nudos entre los procesos, quedando sin embargo por definir la correspondiente distribución de las líneas. Lógicamente, si los dos nudos extremos de una línea pertenecen a un mismo proceso, la línea será asignada a dicho proceso. En cambio, si cada nudo extremo pertenece a un proceso distinto (la línea queda dividida por la frontera

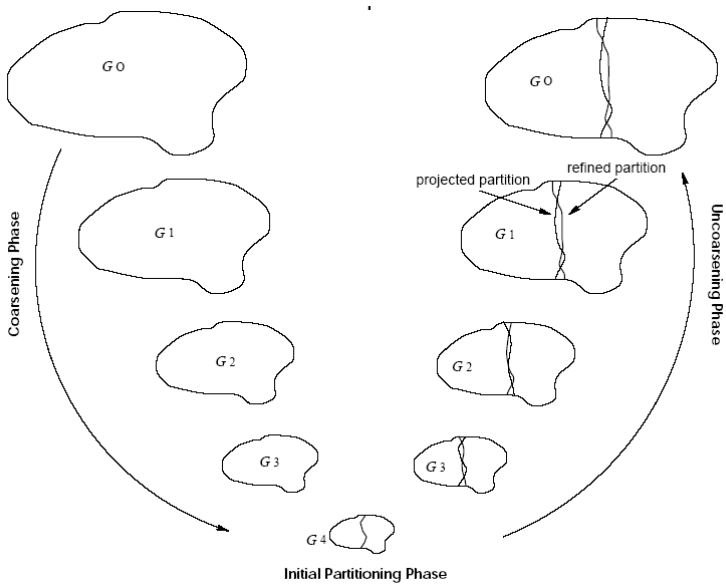

Figura 5.2: Fases de la bisección multinivel

entre dos partes), será asignada arbitrariamente a uno de los dos. En realidad, eso supone trasladar las fronteras entre partes para que dividan los nudos de la red en vez de dividir las líneas (véase la figura 5.3). De esta forma, cada línea será tratada por un solo proceso, aunque a cambio serán necesarias comunicaciones al realizar el tratamiento de los *nudos compartidos* (aquellos atravesados por una frontera).

5.4.2. Paralelización del paso de tiempo del DVEM

Para la aplicación del método DVEM en paralelo, en cada paso de tiempo se llevarán a cabo las cuatro fases del método explicadas en el capítulo 2, junto con una fase adicional para la comunicación de masa y volumen de los nudos compartidos, tal como se ilustra en la figura 5.4.

El algoritmo 5.3 presenta con mayor detalle el proceso a realizar en cada paso de tiempo de la simulación de calidad. Como en todos los algoritmos paralelos de este capítulo, el algoritmo debe ser ejecutado por cada uno de los procesos que intervienen en la simulación. Las fases 1 (líneas 4-6), 2 (líneas 6-7), 3 (líneas 8-10) y 5 (líneas 23-26) son básicamente iguales que en el caso secuencial, teniendo en cuenta que cada proceso aplica las fases sólo sobre su parte local de la red.

La fase 4, innecesaria en el caso secuencial, corresponde a las líneas 12-19, donde se realiza la suma de masas y volúmenes de los nudos compartidos, mediante la comunicación entre todos los procesos y la suma de las contribuciones de cada uno de ellos. Para realizar esta tarea cada proceso forma dos vectores, M' y V', en los que se almacena la contribucion del proceso a la masa y volumen, respectivamente, de

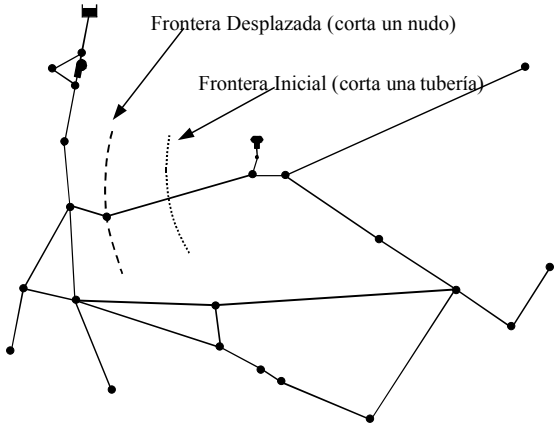

Figura 5.3: Desplazamiento de las fronteras de la partición.

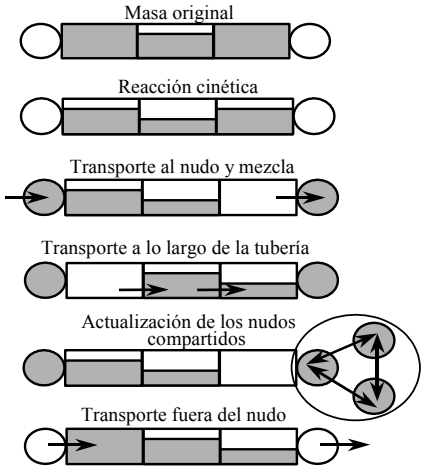

Figura 5.4: Fases del algoritmo paralelo DVM, sobre una tubería con un nudo local y un nudo compartido.

cada uno de los nudos compartidos (entre cualesquiera procesos) de la red. Una vez formados, se realiza la comunicación y suma de los vectores entre todos los procesos, lo que correspondería a la operación `MPI_Allreduce` del sistema de paso de mensajes MPI. De esta manera todos los procesos obtienen la masa y volumen final de todos los nudos compartidos, tras lo cual actualizan los valores correspondientes de sus nodos.

Algoritmo 5.3 Un paso de tiempo del método DVM paralelo.

Entrada: valores de masa $m_{l,k}$ en elementos de volumen de líneas locales en el tiempo inicial t; paso de tiempo τ.

Salida: Valores en $t+\tau$ de la masa $m_{l,k}$ en los elementos de volumen locales, y de concentración c_j en todos los nudos de trabajo locales.

1: /* *Algoritmo a ejecutar en cada proceso* */
2: $M_j = V_j = 0 \qquad \forall$ nudo de trabajo j
3: **para todo** línea local l **hacer**
4: **para todo** elemento k de la línea l **hacer**
5: $m_{l,k} = k\, m_{l,k} e^{\alpha\tau}$
6: **fin para**
7: $M_j = M_j + m_{l,\eta_l}; \quad V_j = V_j + q_l\tau; \quad$ donde j es el nudo aguas abajo de la línea l
8: **para** $k = \eta_l - 1$ hasta 1 **hacer**
9: $m_{l,k+1} = m_{l,k}$
10: **fin para**
11: **fin para**
12: $M'_j = V'_j = 0 \qquad \forall$ nudo compartido j en la red entera
13: **para todo** nudo local compartido j **hacer**
14: $M'_{j'} = M_j; \quad V'_{j'} = V_j; \quad$ donde j' es el índice en M' del nudo local j
15: **fin para**
16: actualizar vectores M', V', con contribuciones de todos los procesos (`allreduce`)
17: **para todo** nudo local compartido j **hacer**
18: $M_j = M'_{j'}; \quad V_j = V'_{j'}; \quad$ donde j' es el índice en M' del nudo local j
19: **fin para**
20: **para todo** nudo de trabajo j **hacer**
21: $c_j = \frac{M_j}{V_j}$
22: **fin para**
23: **para todo** línea local l **hacer**
24: sea j el nudo aguas arriba de la línea l
25: $m_{l,1} = c_j q_l \tau$
26: **fin para**

Por otra parte, al inicio de cada paso de tiempo hidráulico hay que hacer algunas tareas adicionales. En primer lugar, se debe calcular el paso de tiempo para esa porción de la simulación de calidad, lo cual se hace mediante la expresión (2.34),

que implica el cálculo del tiempo de permanencia de todas las tuberías de la red, para a continuación obtener el mínimo. Para hacer esto en paralelo, cada proceso calculará el mínimo local, y a continuación se comunicarán estos mínimos locales entre todos los procesos para obtener el mínimo global (de nuevo mediante una operación `MPI_Allreduce`).

En segundo lugar, se debe rehacer la división en elementos de todas las líneas, de acuerdo con el nuevo paso de tiempo, y trasladar las masas/concentraciones de la antigua segmentación de la líneas a la nueva (véase figura 2.5). Esto se puede hacer sin necesidad de comunicaciones, dado que cada tubería pertenece a un sólo proceso.

5.5. Solapamiento de la simulación hidráulica y la de calidad

Este sección presenta un enfoque paralelo complementario al presentado en las secciones anteriores, y basado en solapar la simulación hidráulica con la de calidad, de manera que se realicen a la vez.

Como se ha mencionado anteriormente, para poder realizar una simulación de la calidad del agua es necesario realizar previamente una simulación hidráulica o de la "cantidad" de agua, que determine cuáles son los caudales que circulan por las tuberías de la red. En Epanet las simulaciones hidráulica y de calidad del agua se realizan por separado, es decir, primero la simulación hidráulica se lleva a cabo por completo y se escriben sus resultados en un fichero. A continuación se realiza la simulación de calidad, leyendo los resultados hidráulicos de ese fichero.

Esto se ilustra mediante la figura 5.5, donde ambas simulaciones se dividen en varias celdas, cada una de las cuales corresponde a un paso de tiempo hidráulico. Tras la resolución de un paso de tiempo de la simulación hidráulica, los resultados que serán necesarios para la simulación de calidad (caudales en las tuberías) son almacenados en un fichero. Más tarde, estos resultados son leídos antes de resolver un paso de tiempo de la simulación de calidad. La subdivisión de cada paso de tiempo hidráulico en pasos más pequeños de calidad no se considera en esta sección, por lo que el término "paso de tiempo" se usa aquí con el significado de "paso de tiempo hidráulico".

Como alternativa al modo de proceder ilustrado en la figura 5.5, aquí se considera una aproximación basada en solapar ambas simulaciones. En concreto, se tendría un proceso a cargo de la simulación hidráulica y otro a cargo de la simulación de calidad. Tan pronto como el primer proceso completa la simulación hidráulica de un paso de tiempo, se envían los resultados al otro proceso. Entonces, mientras el primer proceso procede con la simulación hidráulica del siguiente paso de tiempo, el segundo proceso puede llevar a cabo la simulación de calidad del primer paso de tiempo (véase la figura 5.6). De esta manera, ambas simulaciones se realizan simultáneamente, con la excepción de la simulación hidráulica del primer paso de tiempo y la simulación de calidad del último paso. El resultado es una importante reducción del tiempo de ejecución, como puede apreciarse comparando la figura 5.5 con la figura 5.6.

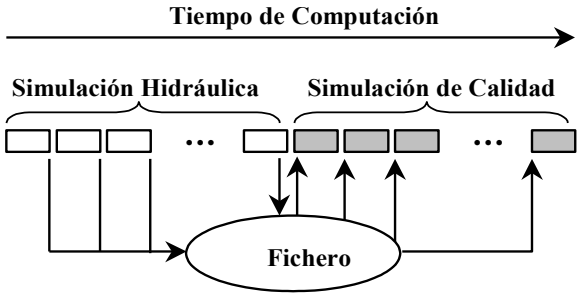

Figura 5.5: Simulaciones hidráulica y de calidad en Epanet.

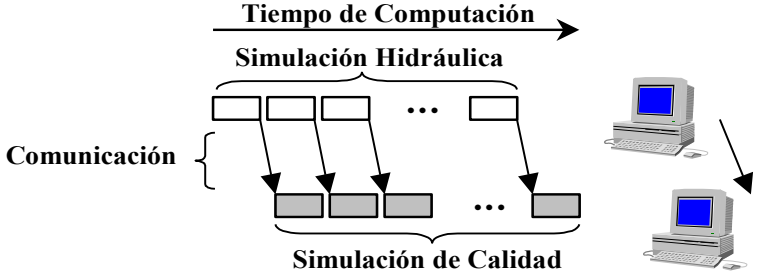

Figura 5.6: Solapamiento de las simulaciones hidráulica y de calidad.

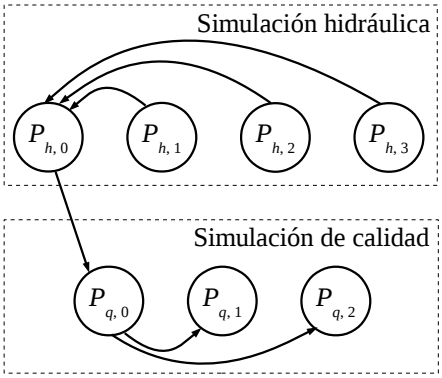

Figura 5.7: Solapamiento entre simulaciones paralelas.

Por otra parte, es importante destacar el hecho de que esta aproximación de aplicación de computación paralela es complementaria con la expuesta en los apartados anteriores. Es decir, ambas aproximaciones se pueden combinar, de manera que existan dos grupos de procesos, uno de ellos trabajando en la simulación hidráulica y el otro en la simulación de calidad. Esto se ilustra en la figura 5.7, que refleja la manera en que tienen lugar las comunicaciones entre ambos grupos de procesos: cuando el "grupo hidráulico" termina un paso de tiempo, los procesos envían los resultados al maestro de su grupo, de la misma manera que harían en una simulación no solapada. Dichos resultados son a continuación enviados por el maestro del grupo hidráulico al maestro del "grupo de calidad", quien finalmente los distribuye entre los procesos de su grupo, de la misma manera que lo haría en una simulación no solapada.

Una ventaja añadida de este enfoque es que no es necesario escribir un fichero temporal con los resultados de la simulación hidráulica, de manera que se ahorra el tiempo de lectura/escritura de resultados. Las operaciones de comunicación entre procesos sólo consumirán parte de ese tiempo.

5.6. Simulación de fugas

La minimización de las fugas requiere en primer lugar la utilización de un método para cuantificarlas, por medio de la consideración de las mismas en la simulación hidráulica. Epanet, en su versión 2.0, incluye la posibilidad de asignar a los nudos un consumo dependiente de la presión de acuerdo con $q = Kp^{\alpha}$, siendo K y α valores proporcionados por el usuario, y p la presión del nudo. De esta manera es posible incluir el tratamiento de fugas en la simulación hidráulica.

El problema de la minimización de fugas, por medio del control operacional de las presiones, queda fuera de las tareas consideradas por Epanet, por lo que el trabajo que se realizará en la tesis comprenderá el desarrollo de algoritmos tanto secuenciales como paralelos para la minimización de fugas.

Caracterización de las fugas

Si consideramos que las fugas permanentes al exterior se producen a través de defectos en la red, la primera hipótesis que se plantea para caracterizar las fugas nos lleva a la ecuación del flujo de un fluido a través de un orificio

$$q = K\ (p - p_o)^{\gamma}\,, \tag{5.3}$$

donde q es el caudal que fluye a través del orificio, p, p_o representan la presión en la parte interior y exterior del orificio, respectivamente, y K es el denominado coeficiente de descarga, que depende del tamaño y forma del orificio. En cuanto al exponente γ, éste toma valores próximos a 0,5 o un poco mayores según las experiencias realizadas en laboratorio.

Para la aplicación de la fórmula anterior a las fugas de redes hidráulicas, podemos asumir que $p_o = 0$, por tratarse de fugas al exterior donde reina la presión atmosférica. En cuanto al valor del exponente γ, Goodwin realizó en [30], por encargo del National Water Council de Inglaterra, una serie de experiencias en periodo nocturno sobre sectores bien instrumentados y con poco consumo en dicho periodo, a fin de medir los caudales entrantes al mismo a medida que se elevaba la presión de alimentación, e identificar así el valor real del exponente γ para las fugas en las redes. Los resultados proporcionaron un valor medio de $\gamma = 1{,}18$, bastante mayor que el de un orificio, lo que puede justificarse por la deformación que experimenta el tamaño de los defectos con el incremento de la presión. Trabajos como [28] y [51] tienen en cuenta lo anterior para la caracterización de las fugas.

En consecuencia, asumiremos que las el caudal de fugas l_j en un nodo j viene dado por la expresión siguiente:

$$l_j = K_j p_j^{1,18} = K_j\,(h_j - z_j)^{1,18}\,, \tag{5.4}$$

donde p_j corresponde a la presión en ese nudo, que es igual a la diferencia entre la altura piezométrica h_j y la elevación z_j, mientras que K_j es un coeficiente de fugas a ser determinado para cada nudo, que se supone constante durante largos periodos de tiempo.

Análisis estático con fugas

La incorporación de las fugas a la simulación hidráulica supone que los consumos en los nudos, hasta ahora considerados como datos de partida, pasan a ser el resultado de la suma de dos componentes: las demandas consumidas por los usuarios, datos de partida; y los caudales de fugas, a priori desconocidos y de los que sólo sabemos su dependencia con las presiones, y por tanto con las alturas piezométricas h, de acuerdo con (5.4). Es decir

$$c_j = \bar{c}_j + l_j, \tag{5.5}$$

donde $\bar{c}_j$ representa la demanda en el nudo j, o sea, el caudal consumido por los usuarios de la red abastecidos por el nudo.

De esta manera, las ecuaciones de equilibrio hidráulico, presentadas en el capítulo 2, se pueden modificar para acomodar la consideración de fugas de la siguiente manera:

$$\begin{aligned} &\phi_k(q_k) - h_{k_1} + h_{k_2} = 0, \quad k = 1, 2, \ldots, m \\ &\textstyle\sum_{k=1}^{m} \delta_{ki} q_k - \bar{c}_i - l_j = 0, \quad i = 1, 2, \ldots, n \\ &l_i - K_i (h_i - z_i)^{1,18} = 0, \quad i = 1, 2, \ldots, n \end{aligned} \tag{5.6}$$

La resolución del sistema de ecuaciones no lineales (5.6) puede realizarse mediante la adaptación a la nueva formulación de alguno de los métodos disponibles para el análisis hidráulico estático, discutidos en el apartado 2.5. Así, [28] introduce la consideración de las fugas en el método de Newton-Raphson por nudos, mientras que [62] considera una versión linealizada de (5.4) para introducir el tratamiento de fugas en el método del Gradiente. En ambos casos la complejidad adicional introducida por la consideración de las fugas es pequeña. La versión 2 de Epanet también permite tener en cuenta la existencia de fugas en la simulación hidráulica mediante elementos llamados *emisores* (*emitters* en inglés), que modelizan un consumo dependiente de la presión.

Otra posibilidad sencilla para la simulación de las fugas es resolver varias veces el problema de equilibrio hidráulico sin considerar la variación de las fugas con la presión, ajustando cada vez el valor de las fugas de acuerdo con las presiones obtenidas, hasta llegar a la convergencia [4].

Finalmente, queda por considerar cómo obtener los valores de K_j. Esto se puede llevar a cabo mediante un método descrito en [51], que tiene en cuenta el porcentaje de fugas totales observado en toda la red para áreas de medición de distrito, bajo condiciones de operación bien definidas.

5.7. Minimización de fugas

El problema de minimización de fugas considerado consiste en encontrar los valores de consigna óptimos de una serie de *Válvulas Reductoras de Presión* (VRP) con el objetivo de reducir las fugas tanto como sea posible, satisfaciendo al mismo tiempo ciertas presiones mínimas de servicio en algunos (o todos) los nudos. Las VRP permiten disminuir la presión de funcionamiento de un determinado sector de una red, lo cual a su vez produce una disminución de las fugas, dada la relación entre fugas y presión vista en el apartado anterior.

El problema de minimización de fugas sobre el periodo de simulación completo se enfoca desde la minimización de los caudales de fuga en régimen permanente en una serie de instantes de tiempo puntuales, de acuerdo con un paso de tiempo prefijado. Se asume que en cada instante los valores de consigna de las VRP pueden ser diferentes, y que existe independencia entre la minimización de las fugas en cada uno de los instantes. Esta suposición de independencia será cierta siempre que el área controlada por las válvulas no contenga depósitos, y sea suministrada a través

de una o más VRP desde otras zonas de la red que operan a una presión mayor, lo cual suele darse en la práctica.

El problema de la minimización de fugas correspondiente a un instante de tiempo se puede formular de la siguiente manera:

$$\begin{aligned} &\text{minimizar } f(x) = \sum_{j=1}^{n} l_j \\ &\text{sujeto a: } \quad F(x,u) = 0 \\ &\qquad\qquad g_j(x) = p_j^{\text{mín}} - p_j \leq 0, \qquad \forall j \in I_c, \end{aligned} \tag{5.7}$$

donde $x = \left(q^T, l^T, h^T\right)^T$ es el vector de variables hidráulicas del sistema, que incluye el vector de fugas l, y u es el vector de consignas de las válvulas, de n_v componentes. La restricción $F(x,u) = 0$ expresa las ecuaciones de equilibrio del análisis hidráulico estático, dadas por el sistema de ecuaciones (5.6), mientras que la segunda restricción impone presiones mínimas, $p_j^{\text{mín}}$, sobre un determinado conjunto de nodos denotado por I_c.

Dado el sistema de ecuaciones no lineales $F(x,u) = 0$, podemos denotar mediante $x(u)$ la solución al sistema correspondiente al vector de consignas u, solución que se supone existe y es única. Podemos entonces definir $\tilde{f}(u) = f(x(u))$, $\tilde{g}_j(u) = g_j(x(u))$, y reformular el problema (5.7) en términos de las variables u

$$\begin{aligned} &\text{minimizar } \tilde{f}(u) = \sum_{j=1}^{n} l_j \\ &\text{sujeto a } \quad \tilde{g}_j(u) = p_j^{\text{mín}} - p_j \leq 0, \qquad \forall j \in I_c, \end{aligned} \tag{5.8}$$

donde, como puede verse, las ecuaciones correspondientes al equilibrio hidráulico no aparecen explícitamente, sino de forma implícita a través de las funciones $\tilde{f}(u)$, $\tilde{g}_j(u)$. En particular, dado un valor de u, la obtención de los valores de $\tilde{f}(u)$, $\tilde{g}_j(u)$ requiere la resolución del problema de análisis hidráulico estático dado por el sistema de ecuaciones (5.6).

El problema de optimización planteado se puede resolver por muchos métodos diferentes de programación no lineal. Para poder aplicarlos es necesario en primer lugar poder evaluar las funciones $\tilde{f}(u)$ y $\tilde{g}_j(u)$, de acuerdo con lo que se indica en el párrafo anterior. En segundo lugar, muchos de los métodos de programación no lineal precisan también la evaluación de los gradientes de las funciones anteriores, $\nabla f(x)$ y $\nabla g_j(x)$. Estos gradientes pueden calcularse mediante diferencias finitas.

5.7.1. Aplicación de programación secuencial cuadrática

Esta tesis plantea el uso del método de *Programación Secuencial Cuadrática* (SQP, del inglés *Sequential Quadratic Programming*) para resolver el problema de minimización presentado anteriormente. La programación secuencial cuadrática es una generalización del método de Newton para optimización con restricciones, que se basa en encontrar un paso a partir del punto actual por medio de la minimización de un modelo cuadrático del problema. Varios paquetes software, incluyendo NPSOL, NLPQL, OPSYC, OPTIMA, MATLAB y CFSQP están basados en este enfoque.

En nuestro caso, la aplicación del método de SQP se lleva a cabo por medio de la implementación de propósito general disponible en la librería CFSQP (*C Code for Feasible Sequential Quadratic Programming*) [45]. CFSQP implementa un algoritmo bastante complejo y sofisticado, capaz de resolver un gran número de problemas de optimización con diferentes tipos de restricciones. Sin embargo, obviaremos aquí los detalles y todos aquellos aspectos que tengan que ver con problemas más generales que aquellos a los que nos enfrentamos en la minimización de fugas. Así pues, consideraremos el problema de minimización

$$\begin{array}{cc} \text{minimizar} & f(x) \\ \text{sujeto a} & g(x) \leq 0 \end{array}$$

donde $x \in \Re^n$, $g : \Re^n \to \Re^p$, con $g(x) = [g_1(x),\ g_2(x), \cdots , g_p(x)]^T$.

Se parte de una aproximación a la solución x_k , y una aproximación H_k a la matriz *Hessiana* del Lagrangiano, la iteración estándar del método SQP consiste en calcular en primer lugar una dirección de búsqueda d^0 mediante la resolución del programa cuadrático

$$\begin{array}{cc} \text{minimizar} & \frac{1}{2} {d^0}^t H_k d^0 + {d^0}^t \nabla f(x_k) \\ \text{sujeto a} & g_j(x_k) + {d^0}^t \nabla g_j(x_k) \leq 0, \qquad 1 \leq j \leq p \end{array}$$

y a continuación obtener $x_{k+1} = x_k + \alpha_k d^0$, donde α_k , denominado tamaño de paso, se obtiene de forma que x_{k+1} minimice una determinada *función de mérito* o función de coste modificada (*merit function*) en la dirección d^0.

Este método presenta entre otras, dos dificultades principales. En primer lugar, aunque las restricciones originales sean consistentes, las aproximaciones lineales que aparecen en los sucesivos problemas cuadráticos pueden fácilmente llegar a ser inconsistentes. En segundo lugar, la selección de una función de mérito adecuada es un tema complicado.

El método CFSQP evita estas dificultades al requerir que cada aproximación x_k sea factible (esto es, se cumpla que $g(x_k) \leq 0$), lo cual elimina posibles inconsistencias en las restricciones linealizadas, a la vez que convierte la propia función objetivo en una función de mérito apropiada. Sin embargo, la factibilidad de x_k no garantiza la factibilidad de la dirección estándar de SQP d^0. Incluso si d^0 fuera factible, la búsqueda del mínimo en dicha dirección podría no permitir tomar un paso $\alpha_k = 1$ en las proximidades de una solución, lo cual es un requisito para garantizar un ritmo de convergencia superlineal.

Para superar estas dificultades, se lleva a cabo una desviación de d^0, es decir, se sustituye d^0 por una combinación $d = (1 - \rho)d^0 + \rho d^1$ de d^0 y una dirección factible de descenso d^1 esencialmente arbitraria. En segundo lugar, se curva la dirección de búsqueda, lo que significa que la búsqueda de x_{k+1} se llevará a cabo a lo largo de un arco de la forma $x + td + t^2\tilde{d}$. El propósito de $\tilde{d}$ es permitir tomar un paso de uno cerca de una solución.

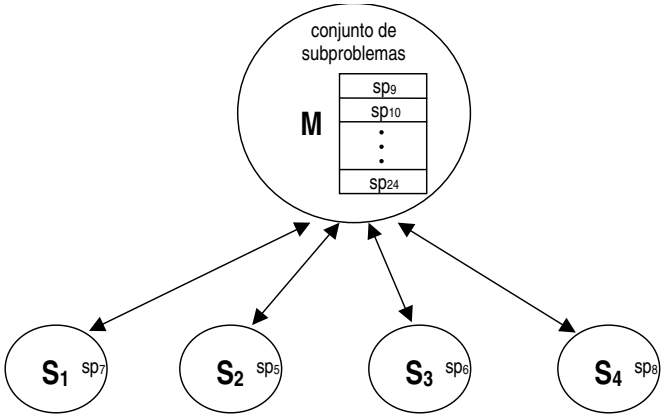

Figura 5.8: Esquema maestro-esclavo con asignación dinámica de subproblemas.

5.8. Algoritmo paralelo de minimización de fugas

Para aplicar computación paralela al proceso de optimización, la idea básica es aprovechar el hecho de que los subproblemas correspondientes a cada paso de tiempo son independientes entre sí. Por lo tanto cada subproblema puede ser asignado a un proceso diferente, lo que conduce a un algoritmo paralelo de grano grueso simple a la vez que eficiente.

El problema es muy adecuado para ser tratado mediante un paradigma *maestro-esclavo*. El maestro es un proceso cuya única tarea es distribuir los subproblemas asociados a instantes de tiempo entre los diferentes esclavos. Éstos están continuamente esperando recibir un subproblema a fin de procesarlo y devolver los resultados al maestro.

El funcionamiento se ilustra en la figura 5.8. El maestro (**M**) posee un conjunto de subproblemas (**sp**) que deben ser resueltos; por ejemplo si el paso de tiempo es 1 hora y la duración de la simulación es 24 horas, habría 25 subproblemas, correspondientes a los instantes de tiempo 0-24. El maestro asigna inicialmente un subproblema a cada esclavo. Tan pronto como un esclavo (**S**) completa la solución de su subproblema, envía un mensaje al maestro informando del valor de la solución. Cuando el maestro recibe el mensaje, selecciona un nuevo subproblema del conjunto y se lo da al proceso esclavo. Este esquema de asignación dinámica de subproblemas permite un buen balance de carga, sin requerir una implementación sofisticada. El algoritmo 5.4 muestra el esquema del código correspondiente al proceso maestro.

El algoritmo implica comunicaciones punto-a-punto siempre entre el maestro y cualquiera de los esclavos. No se producen comunicaciones entre dos esclavos. Por otra parte, el volumen de datos transmitidos es muy reducido. En particular, se hace uso de dos tipos de mensajes:

1. Mensajes del maestro a un esclavo, para encargar a éste la resolución de un subproblema. Estos mensajes contienen únicamente un dato de tipo entero que

Algoritmo 5.4 Algoritmo del proceso maestro para minimización de fugas.

Entrada: N_{proc}, numero de procesos esclavos; N_{sub}, número de subproblemas
Salida: consignas óptimas de las VRP

```
construir conjunto de subproblemas, C
para p = 0, 1, ··· mín(N_proc, N_sub) − 1 hacer
  extraer un subproblema sp del conjunto C
  asignar subproblema sp al proceso p
fin para
N_rec = 0 /* número de soluciones recibidas */
mientras N_rec < N_sub hacer
  esperar una solución, sol de cualquier proceso p
  recibir la solución sol del proceso p
  si C ≠ ∅ entonces
    extraer un subproblema sp de C
    enviar subproblema sp al proceso p
  fin si
fin mientras
```

indica el instante de tiempo correspondiente al subproblema a resolver.

2. Mensajes de un esclavo al maestro, para devolver la información relativa a la solución de un subproblema. Esa información consiste en las consignas de las VRP, junto con información adicional sobre el funcionamiento del método de optimización, número de iteraciones necesarias, etc. Las consignas se representan mediante números en coma flotante, siendo el número de consignas reducido (normalmente menos de 10).

5.8.1. Implementación de la aplicación de minimización de fugas

Consideramos en este apartado los aspectos principales de implementación de la aplicación completa para la minimización de fugas, en cuyo marco está situado el algoritmo de minimización de fugas que se ha presentado en los párrafos anteriores.

Los procesos de la aplicación tienen distintos roles. Como se ha comentado, uno de los procesos es el maestro y los demás son los esclavos o trabajadores. Además, uno de los procesos (al que llamaremos *lector-escritor*) asumirá las tareas de leer los datos de la red y escribir los resultados de salida. El proceso maestro se podría ejecutar en el mismo procesador que alguno de los procesos esclavos (por ejemplo el proceso *lector-escritor*), dado que el maestro estará ocioso la mayor parte del tiempo, esperando mensajes de los esclavos.

Un caso especial es aquel en el que el programa paralelo se ejecuta sobre un solo procesador. En tales circunstancias, el proceso maestro ya no tiene sentido, y sólo se utilizará un proceso que solucionará de forma secuencial cada uno de los subproblemas de optimización. Por lo tanto, el programa paralelo se ejecuta de forma

eficiente sobre una sola máquina, pudiendo utilizarse como programa secuencial de referencia para la obtención de índices de prestaciones del programa paralelo.

La ejecución de la aplicación de minimización de fugas puede dividirse en las siguientes fases:

1. Inicialización: en primer lugar, uno de los procesos se encarga de leer los datos de entrada y distribuirlos entre el resto de procesos.

2. Minimización de fugas: a continuación, se lleva a cabo la minimización de fugas mediante el algoritmo paralelo descrito en el apartado 5.8.

3. Simulación final: por último, se realiza una simulación utilizando los valores óptimos de las consignas de las VRP, calculados en la fase anterior. Los resultados de la simulación se escriben en disco.

A continuación se describen estas fases, presentando los distintos roles adoptados por los procesos que componen el programa paralelo.

Inicialización

Los datos que son leidos en esta fase comprenden:

- Datos necesarios para la simulación hidráulica mediante Epanet, es decir, todos aquellos datos que describen la topología de la red, las características de todos sus elementos, y datos de la simulación a realizar (perido de simulación, paso de tiempo, etc.)

- Datos para establecer el valor de los coeficientes de fugas.

- Datos relativos al problema de minimización de fugas a resolver. En particular, a) definición de las VRP mediante las cuales se desea llevar a cabo la minimización, y b) definición de las restricciones de presión que deben cumplirse.

Todos estos datos son leidos por el proceso *lector-escritor*, quien los difunde al resto de esclavos. Sólo algunos de estos datos son enviados al proceso maestro, dado que este es un mero coordinador que no realiza computaciones hidráulicas, y por lo tanto no necesita muchos de los datos. Eso explica además que la lectura la realice un esclavo, en vez del maestro.

Minimización de fugas

Esta es la parte central del programa paralelo, que consume la mayor parte del tiempo de ejecución. Es también la parte en la cual se saca provecho de la computación paralela, como se ha ha descrito en el apartado 5.8.

Tabla 5.1: Redes hidráulicas para evaluar las simulaciones hidráulica y de calidad

	tuberías	**nudos**	**depósitos**
Red 1	4901	2501	1
Red 2	19 801	10 001	1
Red 3	34 516	32 404	4

Tabla 5.2: Redes hidráulicas para evaluar la minimización de fugas

	tuberías	**nudos**	**depósitos**	**VRP**
Red 4	1345	1241	1	3
Red 5	126	66	4	2

Simulación final

Las consignas óptimas de las VRP son recogidas por el maestro durante la fase anterior. Una vez ha finalizado esta, se lleva a cabo una simulación final, utilizando los valores óptimos para las consignas de las válvulas. Dado que el proceso maestro es un simple coordinador que no es capaz de realizar tareas de simulación, las valores de las consignas son transmitidos al proceso *lector-escritor*. Este proceso realiza la simulación hidráulica final y escribe los resultados de la misma en un fichero.

5.9. Resultados obtenidos

En este apartado consideramos dos conjuntos de redes para evaluar los algoritmos presentados en este capítulo.

En primer lugar, se utilizan las redes de la tabla 5.1 para evaluar las prestaciones de los algoritmos paralelos para la simulación hidráulica y de calidad del agua. Desde el punto de vista hidráulico, las redes son sencillas, siendo abastecidas por la acción de la gravedad desde uno o más depósitos. La modulación de las demandas se lleva a cabo mediante cinco patrones previamente establecidos, haciendo corresponder a cada nudo de la red uno de ellos.

En segundo lugar, la evaluación de la minimización de fugas se lleva a cabo mediante las redes de la tabla 5.2.

Los resultados que se muestran en este capítulo fueron obtenidos en un cluster de PCs con sistema operativo Windows NT e interconexión mediante Fast-Ethernet. Existen plataformas mucho más potentes que las utilizadas en este capítulo, sin embargo se trata aquí de evaluar las ventajas del paralelismo. Más que el tiempo de ejecución en términos absolutos, lo importante es la ganancia de velocidad de los algoritmos paralelos respecto a los secuenciales.

Tabla 5.3: *Speedup* (S) de la simulación hidráulica paralela (con p procesadores) respecto a Epanet y respecto al código paralelo en 1 procesador (par-1p).

	S Epanet			**S par-1p**	
p	1	2	4	2	4
Red 1	1,10	0,91	0,84	0,83	0,76
Red 2	1,89	3,80	4,53	2,01	2,39
Red 3	1,47	0,71	0,87	0,48	0,59

5.9.1. Resultados de la simulación hidráulica

En la tabla 5.3 se presenta la ganancia de velocidad o *speedup* de la simulación hidráulica paralela para las redes 1, 2 y 3, tanto con respecto a Epanet como con respecto al propio código paralelo con un solo procesador. Hay que decir que la librería PSPASES utilizada impone la restricción de que el número de procesadores p sea una potencia de 2, debiendo cumplirse además que $p \geq 2$. Debido a ello, cuando el código paralelo se ejecuta sobre un solo procesador se hace uso del *solver* lineal de Epanet, aunque se combina con el algoritmo de reordenación DAM en vez de MG (para lo cual se utiliza la librería METIS de particionado de grafos [42]).

Se puede ver que la simulación paralela en un procesador es más rápida que Epanet, lo que se debe precisamente al diferente algoritmo de ordenación utilizado. Por otra parte, se puede apreciar que si bien el algoritmo paralelo muestra buenas prestaciones para la red 2, este no es el caso con las redes 1 y 3. Esto se debe al hecho de que la resolución de los sistemas lineales correspondientes no contiene la suficiente complejidad computacional. En el caso de la red 1, el reducido tamaño de la red (2500 nudos) es el causante de la falta de complejidad. Respecto a la red 3, aunque la red es grande (32 404 nudos), el escaso número de líneas (34 516) con respecto al número de nudos produce una matriz con muy pocos elementos no nulos, lo que significa la realización de pocas operaciones en coma flotante durante la resolución del sistema. De esta manera, aunque la red 3 es más grande que la red 2, el tiempo de simulación (tanto con Epanet como con el programa paralelo) para la red 3 es menor que el tiempo para la red 2.

Resulta algo extraño que el tiempo de simulación de la red 3 aumenta considerablemente al pasar de 1 a 2 procesadores (speedup de 0,48), y sin embargo se reduce ligeramente al pasar de 2 a 4 (speedup de 0,59). Sin embargo, hay que tener en cuenta que en el caso de un procesador se utiliza en realidad el *solver* de Epanet, mientras que para más de un procesador se usa PSPASES. Los resultados parecen indicar que para sistemas altamente dispersos como el de la red 3, la aproximación multifrontal empleada por PSPASES introduce una sobrecarga muy elevada respecto al *solver* lineal más sencillo empleado por Epanet. Aunque al pasar de 2 a 4 el tiempo se reduce ligeramente, esta mejora no compensa en absoluto la sobrecarga que supone la estrategia multifrontal.

Como conclusión, podemos decir que las prestaciones del algoritmo de simula-

Tabla 5.4: *Speedup* (S) de la simulación de calidad en paralelo (con p procesadores) respecto a Epanet y respecto al código paralelo en 1 procesador (par-1p).

	S Epanet				**S par-1p**		
p	1	2	3	4	2	3	4
Red 1	1,14	1,89	2,70	3,07	1,66	2,36	2,69
Red 2	1,08	1,81	1,96	2,37	1,68	1,82	2,20
Red 3	1,04	1,74	1,85	2,17	1,67	1,77	2,08

Tabla 5.5: *Speedup* (S) de la simulación completa con solapamiento (con p procesadores) respecto a Epanet y respecto al código paralelo en 1 procesador (par-1p)

	S Epanet			**S par-1p**		
p	2	3	4	2	3	4
Red 1	1,39	2,25	3,15	1,23	1,98	2,77
Red 2	2,22	3,50	4,57	1,41	2,24	2,91
Red 3	1,73	2,65	2,82	1,50	2,29	2,44

ción hidráulica, en términos de tiempo de ejecución, dependen fuertemente de la complejidad de los sistemas de ecuaciones correspondientes a la red considerada. Esto no es sorprendente si se tiene en cuenta que el enfoque de paralelización usado para el proceso de simulación está conducido principalmente por la resolución de los sistemas lineales. Por otra parte, las buenas prestaciones de la versión paralela sobre un solo procesador constituyen también un resultado interesante.

5.9.2. Resultados de la simulación de la calidad

En la tabla 5.4 se muestra el *speedup* del algoritmo paralelo de simulación de la calidad del agua, tanto respecto a Epanet como respecto al propio algoritmo paralelo ejecutado sobre un procesador, para las redes de la tabla 5.1. Se puede ver que el algoritmo paralelo produce una reducción importante del tiempo respecto a Epanet, llegando a un *speedup* superior a 3 con 4 procesadores para la red 1.

5.9.3. Resultados de solapamiento de simulaciones

La tabla 5.5 muestra la ganancia de velocidad de la simulación completa (hidráulica + calidad) usando el enfoque de solapar los dos tipos de simulación. Se muestra el speedup tanto respecto a Epanet como respecto al código paralelo sobre un solo procesador. Para el solapamiento se han usado 2, 3 y 4 procesadores, de manera que uno de ellos se encarga de la simulación hidráulica y el resto realiza la simulación de calidad.

En este caso el *speedup* respecto Epanet llega a 4,57 (con 4 procesadores en la red 2), lo que significa que este enfoque de paralelización es la mejor opción si se quieren realizar los dos tipos de simulación de redes. El *speedup* respecto a Epanet para la red 2 es mayor que el número de procesadores, lo que se debe a que la versión paralela de la simulación hidráulica en un procesador es considerablemente más rápida que Epanet para la red 2 (1,89 veces más rápida según la tabla 5.3). Esto a su vez es debido, como se ha comentado, a la utilización de un algoritmo diferente de reordenación de la matriz (DAM en vez de MG).

5.9.4. Resultados de la minimización de fugas

El procedimiento de reducción de fugas descrito anteriormente ha sido aplicado a las redes de test 4 y 5 descritas en la tabla 5.2. La red 5 consta de cuatro depósitos, de manera que el nivel tres de ellos permanece fijo para mantener la distribución del caudal inyectado, mientras que el nivel del último varía durante la simulación. Dos VRP interactuando controlan las presiones, y por tanto las fugas que se producen.

La red 4 se controla mediante tres VRP que pueden configurarse para una operación combinada o independiente (cada una sobre un sector aislado). Se han analizado ambas posibilidades, resultando la segunda opción preferible desde el punto de vista de la reducción de fugas, por lo que consideramos la configuración correspondiente a operación independiente.

Cada una de las VRP de la red 4 se coloca en el punto de suministro de un sector de la red. Inicialmente se considera que las VRP están completamente abiertas (es decir, no producen ninguna pérdida de carga) y se asume que el volumen de las fugas supone un 12 % del volumen total inyectado en la red durante un periodo de 24 horas. Este porcentaje inicial se fija arbitrariamente con objeto de obtener valores estimados de los coeficientes de fugas de los nudos de la red, para poder mostrar las posibilidades ofrecidas por el método de reducción de fugas. Partiendo de esta situación, se realiza la simulación correspondiente, obteniendo la distribución espacial de fugas mostrada en la figura 5.9 (a las 0:00 horas).

A continuación, se obtienen las consignas óptimas de las VRP para cada hora, por medio del método de minimización de fugas paralelo descrito anteriormente. El resultado de la simulación con las consignas obtenidas de esta manera muestra que las fugas se reducen a un 4,71 %. La figura 5.10 presenta la nueva distribución de fugas a las 0:00 horas.

La tabla 5.6 muestra los porcentajes de fugas obtenidos mediante la aplicación del algoritmo de minimización de fugas, para las redes 4 y 5, suponiendo un porcentaje inicial de fugas de 12 %.

Finalmente, en la tabla 5.7 se analiza el tiempo empleado en la resolución del problema de minimización de fugas para las redes 4 y 5. El algoritmo paralelo se ejecuta con diferente número de procesadores (de 1 a 5). En este caso no se puede comparar con Epanet puesto que Epanet no realiza este tipo de minimización. Los resultados muestran un buen *speedup*, siendo éste de 5,0 y 5,37 para las redes 4 y 5 respectivamente, con 5 procesadores.

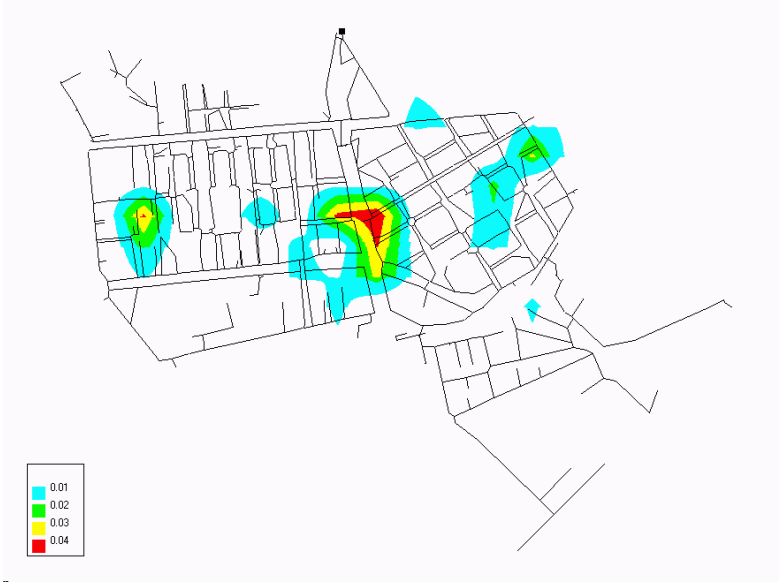

Figura 5.9: Distribución inicial de fugas en la red 4 a las 0:00 horas

Figura 5.10: Distribución final de fugas en la red 4 a las 0:00 horas

Tabla 5.6: Reducción de los porcentajes de fugas de las redes 4 y 5

	% fugas inicial	**%fugas final**
Red 4	12 %	4,71 %
Red 5	12 %	8,82 %

Tabla 5.7: Tiempo (segs) y *speedup* para minimización de fugas con distinto número de procesadores (p)

	p	1	2	3	4	5
tiempo	**Red 4**	382,64	191,03	128,40	99,80	76,48
	Red 5	11,49	5,66	3,93	2,76	2,14
Speedup	**Red 4**	1,00	2,00	2,98	3,83	5,00
	Red 5	1,00	2,03	2,92	4,16	5,37

5.10. Conclusiones

Se han presentado en este capítulo algoritmos paralelos en memoria distribuida para la simulación hidráulica, la simulación de la calidad y la minimización de fugas de redes de distribución de agua.

Los resultados obtenidos dejan ver unas prestaciones desiguales. Si bien pueden considerarse satisfactorias las obtenidas por los algoritmos paralelos de calidad del agua y de minimización de fugas, no ocurre lo mismo con la simulación hidráulica.

Cabe preguntarse si las prestaciones de la simulación hidráulica podrían mejorar al considerar su paralelización sobre plataformas de memoria compartida, las cuales han ganado protagonismo en los últimos años frente a la opción de memoria distribuida. Esa vía es la que se explora en el capítulo 6, donde se considera la paralelización no sólo del método GGA usado en Epanet, sino también del método de mallas, usando en este caso una implementación que incorpora las aportaciones al método presentadas en el capítulo 4.

Capítulo 6

Simulación paralela en sistemas multicore

En este capítulo se considera la paralelización, sobre máquinas de memoria compartida y mediante la herramienta OpenMP, del proceso de simulación de caudales y presiones de redes de distribución de agua. Se realiza la paralelización de la simulación tanto mediante el método del gradiente (GGA), como mediante el método de mallas. En el primer caso se parte de la implementación secuencial correspondiente a Epanet (versión 2.00.12), mientras que en el segundo caso el punto de partida es la implementación del método de mallas desarrollada en esta tesis, que incorpora las aportaciones descritas en el capítulo 4.

La motivación de este capítulo es buscar una paralelización más eficiente del proceso de simulación hidráulica, que como se vio en el capítulo anterior no obtiene buenas prestaciones al paralelizarse sobre plataformas de memoria distribuida. No se considera en este capítulo la simulación de la calidad del agua ni la minimización de fugas, dado que la paralelización de esos procesos en memoria distribuida sí que resulta satisfactoria.

Las plataformas de memoria compartida han ganado relevancia y aceptación en los últimos años. Los sistemas *multicore* presentes hoy en día están en todo tipo de máquinas, tanto servidores como equipos portátiles e incluso *smartphones*. Por tanto, es interesante el desarrollo de un código paralelo en memoria compartida para la simulación hidráulica, dado que podrá ser utilizado en una gran cantidad de computadores actuales.

El autor de la tesis describe en [6, 7] el trabajo desarrollado en el contexto de este capítulo.

6.1. Trabajos previos

En este apartado revisamos algunos trabajos que consideran la aplicación de computación de altas prestaciones sobre memoria compartida para la simulación de

redes de distribución de agua.

En [33] se analizan las distintas tareas que componen la simulación hidráulica en periodo extendido mediante el método GGA (algoritmo 5.1), tomando para ello la implementación del método correspondiente a la librería CWSNet [32]. Se señala que la tarea de actualización del sistema lineal (línea 6 del algoritmo) es la que más tiempo consume, debido al cálculo de coeficientes relacionados con las pérdidas de carga de las líneas (como veremos en la sección 6.3).

Teniendo esto en cuenta, el trabajo [33] aplica tanto instrucciones vectoriales (SIMD) como procesadores gráficos (GPU) para acelerar la simulación mediante el método GGA. Las instrucciones SIMD ejecutan la misma operación sobre múltiples datos simultáneamente. Estas operaciones están disponibles en casi todas las familias de procesadores actuales, a través del uso de registros e instrucciones especiales. En el artículo mencionado, las instrucciones SIMD se aplican fundamentalmente a la actualización del sistema lineal, obteniéndose una reducción del tiempo de ejecución entre 11 % y 28 % respecto a CWSNet (el código secuencial de partida), aunque la reducción respecto a Epanet es algo menor, llegando hasta un 19,5 %. Respecto a las GPUs, el artículo considera su utilización también para la actualización de los coeficientes del sistema lineal, consiguiendo reducir el tiempo de cálculo entre 9 % y 19 % con respecto a CWSNet para redes suficientemente grandes, aunque la mejora se queda en sólo 3,7 % al comparar con Epanet. Los resultados se obtienen con una GPU NVIDIA GeForce GTX 285 (240 cores).

En [34] se pretende acelerar el código de Epanet mediante la utilización de GPUs, aunque en este caso el uso de GPUs se centra en la resolución del sistema lineal, utilizando gradiente conjugado precondicionado con Jacobi. La ventaja de usar el método de gradiente conjugado para resolver el sistema lineal, de cara a la utilización de GPUs, es que utiliza operaciones (productos matriz-vector, actualización de vectores y productos escalares) que han sido ampliamente investigadas e implementadas para matrices dispersas en GPUs, con buenos resultados. Sin embargo, los resultados del trabajo no son buenos, puesto que el método no converge para algunas de las redes estudiadas, de lo que se deduce que el método de gradiente conjugado precondicionado con Jacobi no es suficientemente estable para resolver los sistemas lineales producidos por Epanet en redes de tamaño medio o grande.

De manera similar, [19] evalúa la utilización de GPUs para la resolución de sistemas lineales de características similares a las que aparecen en el contexto de la simulación hidráulica. El trabajo considera el método de Gradiente Conjugado sin precondicionado, comparando las prestaciones obtenidas en una GPU respecto a una CPU. Aunque se utiliza almacenamiento denso para las matrices, se analiza las implicaciones que los resultados obtenidos tendrían si se trabajara sobre matrices dispersas, concluyéndose que la utilización de GPUs para resolver el sistema lineal sólo reduciría el tiempo en casos de redes excepcionalmente grandes.

El trabajo [13] considera la reducción del tiempo de simulación de redes hidráulicas utilizando procesadores *multicore* modernos, centrándose en la sustitución del *solver* lineal de Epanet (que cuenta ya con 30 años de antigüedad) por otros códigos actuales capaces de aprovechar la potencia de múltiples *cores*. En concreto, el artículo considera siete *solvers* lineales modernos (algunos basados en métodos directos y

otros en métodos iterativos, seis de ellos orientados a CPUs y uno a GPUs), y concluye que ninguno de ellos es capaz de funcionar más rápido que el *solver* original de Epanet para las redes reales consideradas. Sólo dos de los *solvers* fueron capaces de superar al de Epanet, aunque únicamente con redes artificiales de tamaño medio o grande. El trabajo concluye por tanto que el *solver* lineal de Epanet sigue siendo hoy en día el más rápido para las redes hidráulicas encontradas en la práctica.

En definitiva, se puede decir que los esfuerzos por mejorar las prestaciones de la simulación hidráulica mediante computación de altas prestaciones sobre memoria compartida han obtenido un éxito limitado, con una reducción del tiempo de ejecución con respecto a Epanet por debajo del 20 %. Frente a esto, en este capítulo mostramos una aproximación basada en OpenMP que conduce a resultados considerablemente mejores que los de trabajos relacionados existentes. Por otra parte, la utilización de instrucciones SIMD y la paralelización mediante OpenMP no son excluyentes, por lo que la aproximación mostrada en este capítulo se podría por ejemplo combinar con la presentada en [33].

6.2. Bloques computacionales

Antes de llevar a cabo la paralelización de la simulación hidráulica, se debe analizar las tareas o bloques computacionales que la componen, determinando cuáles son los que contribuyen en mayor medida al tiempo de ejecución y son asimismo susceptibles de paralelización.

6.2.1. Identificación de tareas

Como se ha mencionado anteriormente, en [33] se realiza un análisis de los bloques computacionales de la simulación hidráulica mediante el método GGA, utilizando como código de referencia el de la librería CWSNet [32]. Aquí llevamos a cabo un análisis similar al de [33], pero considerando el código de Epanet (versión 2.00.12) en vez de CWSNet, lo que lleva a que la división en tareas sea ligeramente diferente, así como la contribución de cada tarea al tiempo de ejecución total.

En algoritmo 6.1 puede verse la división en tareas considerada para la simulación hidráulica mediante el método GGA. En él se pueden identificar los siguientes bloques computacionales principales (en negrita y entre paréntesis):

Inicialización: aquí se crean las estructuras de datos necesarias para el almacenamiento de la matriz y vector del sistema lineal. Esto se hace tras reordenar los nudos para minimizar el llenado de la matriz y llevar a cabo la factorización simbólica de la misma. Sólo se hace una vez al principio de la simulación.

Demandas: en esta tarea se calcula la demanda de cada uno de los nudos, que se obtiene como la suma de varios componentes, cada uno con una variación en el tiempo descrita mediante un patrón de demanda. La tarea comprende también la aplicación de acciones de control sobre válvulas y bombas.

Algoritmo 6.1 Algoritmo GGA de simulación hidráulica.

```
leer datos de la red desde un fichero de entrada
realizar reordenación, descomp. simbólica (inicialización)
/* Simulación en periodo extendido */
mientras t < t_final hacer
  establecer demandas, controles en válvulas y bombas (demandas)
  /* Análisis hidráulico estático mediante método de Todini */
  mientras no convergencia hacer
    actualizar el sistema lineal (actualización sistema)
    resolver sistema lineal, obteniendo alturas piezométricas (solver lineal)
    obtener nuevos caudales (caudales)
    actualizar estado de válvulas, bombas y tuberías (estado)
  fin mientras
  paso al siguiente instante de tiempo
fin mientras
```

Actualización del sistema: corresponde al cálculo de coeficientes relacionados con las fórmulas de pérdida de carga en las líneas, y la actualización correspondiente de la matriz y el vector del sistema lineal. Existen diferentes variantes de las fórmulas de pérdida de carga: Hazen-Williams, Darcy-Weisbach y Chezy-Manning (ver apartado 2.2). En estas fórmulas se calculan potencias y (en el caso de Darcy-Weisbach) logaritmos, que son operaciones computacionalmente costosas.

Solver lineal: para resolver el sistema lineal se utiliza la descomposición de Cholesky para matrices dispersas [27], aplicado sobre la matriz previamente reordenada. La solución del sistema lineal corresponde a la nueva aproximación de las alturas piezométricas en los nudos.

Caudales: consiste en actualizar los caudales de acuerdo con las nuevas alturas piezométricas, por medio de la fórmula 2.16.

Estado: se trata de actualizar el estado de las válvulas teniendo en cuenta las condiciones de operación de la red. Estas condiciones pueden suponer que una válvula o bomba que se cierre (por ejemplo para evitar flujo en dirección contraria a la de operación de la válvula), o bien que una válvula se abra completamente. El cambio de estado de una válvula o bomba supone en general un régimen de operación distinto de la red, por lo que afecta también a la convergencia del algoritmo.

En el caso del método de mallas, el proceso de simulación simulación hidráulica corresponde al algoritmo 6.2. Puede verse que muchas de las tareas son análogas a las del algoritmo GGA. En concreto:

Inicialización: en esta fase se debe obtener un conjunto de mallas independientes de la red. Como se ha explicado en el capítulo 4, el conjunto elegido tiene

Algoritmo 6.2 Algoritmo de simulación hidráulica mediante el método de mallas.

```
leer datos de la red desde un fichero de entrada
selecionar mallas, hacer reordenación y descomp. simbólica (inicialización)
/* Simulación en periodo extendido */
mientras t < t_final hacer
  establecer demandas, controles en válvulas y bombas (demandas)
  obtener un vector de caudales balanceado (balance)
  /* Análisis hidráulico estático mediante método de mallas */
  mientras no convergencia hacer
    actualizar el sistema lineal (actualización sistema)
    resolver sistema lineal, obteniendo caudales correctores de mallas (solver lineal)
    obtener nuevos caudales de líneas (caudales)
    obtener nuevas alturas piezométricas (alturas)
    actualizar estado de válvulas, bombas y tuberías (estado)
  fin mientras
  paso al siguiente instante de tiempo
fin mientras
```

una influencia muy grande en las prestaciones de la resolución de los sistemas lineales del método. Una vez hecho esto, se realiza la reordenación y descomposición simbólica de la matriz del sistema, de la misma manera que se hace en el caso de GGA. La diferencia es que las características de la matriz en cuanto a tamaño o número de elementos no nulos son distintas.

Balance, alturas: ambas tareas suponen recorrer un árbol de nudos de la red (árbol de expansión del grafo formado por la red). En el caso del balance de caudales, el recorrido es desde las hojas a la raíz, mientras que para actualizar las alturas piezométricas el recorrido es en sentido contrario.

Actualización sistema: esta tareas es muy similar a la análoga del método GGA. De nuevo entran en juego coeficientes relacionados con las fórmulas de pérdida de carga en las líneas, que suponen la realización de operaciones computacionalmente costosas.

Solver lineal: el sistema a resolver tiene en principio una matriz dispersa, simétrica y definida positiva. Sin embargo, el tratamiento de las VRP se realiza mediante la aproximación no simétrica descrita en el apartado 4.6.1, lo que significa que la matriz tiene la siguiente estructura:

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix}$$

donde además del bloque G_{11}, que es una submatriz dispersa, simétrica, y definida positiva, tenemos una parte no simétrica formada por el resto de bloques, que se almacenan además de forma densa.

Tabla 6.1: Batería de redes de test.

Red	nudos	líneas	D/E/B/V	VP	Duración/paso	Fórmula
red 1	12527	14831	2/2/4/5	1	26h 55min/5min	H-W
red 2	26653	29046	26/0/0/0	0	48h/20min	H-W
red 3	4240	4649	4/0/0/6	5	96h/5min	H-W
red 4	25816	29345	3/0/0/51	10	24h/1h	D-W

Caudales: esta tarea supone acumular sobre cada línea las correcciones de caudal de cada una de las mallas en las que está presente la línea.

Demandas, estado: estas tareas se realizan de la misma manera que en el método GGA.

6.2.2. Contribución de las tareas al tiempo de ejecución

Con el propósito de determinar la contribución de cada una de las tareas anteriores al tiempo total de ejecución de la simulación, se han seleccionado las redes 3-6 del conjunto de redes utilizadas en el capítulo 4, descartando las redes 1 y 2, por ser demasiado pequeñas para beneficiarse del paralelismo. La tabla 6.1 muestra de nuevo los detalles de las redes, que han sido renumeradas como redes 1 a 4. Se incluyen el número de depósitos (D), embalses (E), bombas (B) y válvulas (V), el número válvulas reguladoras de presión (VP, que incluye tanto VRPs como VSPs), el periodo de simulación considerado (duración), el paso de tiempo para la simulación hidráulica y la fórmula usada para las pérdidas de carga de las tuberías según se describe en el apartado 2.2 (H-W para Hazen-Williams y D-W para Darcy-Weisbach).

La tabla 6.2 presenta el tiempo de ejecución (en segundos) para la simulación de las redes, y la contribución a ese tiempo de cada una de las tareas, para ambos métodos. Las simulaciones se han realizado usando un sólo núcleo de CPU, con la máquina y el compilador descritos en la sección 6.5. Se puede ver que el método GGA es el más rápido para las redes 1 y 3, mientras que el método de mallas lo es para las redes 2 y 4. Cabe señalar que los tiempos de ejecución del método de mallas son algo superiores a los presentados en el capítulo 4, pese a utilizarse la misma máquina, debido a que, como se comenta en la sección 6.5, la paralelización del método llevada a cabo en este capítulo dio como resultado una mejora también del código secuencial, que se tuvo en cuenta para los resultados del capítulo 4.

Con respecto a las diferentes tareas, la actualización del sistema lineal destaca como la que más tiempo consume, representando entre un 29 % y 42 % del tiempo de ejecución para GGA, y entre un 40 % y 42 % para el método de mallas. El *solver* lineal también tiene una contribución importante, entre 27 % y 34 %, en el caso de GGA, mientras que tiene menos importancia en el método de mallas, entre 9 % y 13 % con la excepción de la red 4 (31,5 %). El menor peso de la resolución del sistema en el método de mallas se debe al menor tamaño de las matrices, aunque en la red

Tabla 6.2: Contribución de cada tarea al tiempo de ejecución secuencial.

	GGA				método de mallas			
	red1	**red2**	**red3**	**red4**	**red1**	**red2**	**red3**	**red4**
inicialización	7,6 %	27,0 %	1,2 %	27,0 %	3,6 %	9,3 %	0,9 %	10,3 %
demandas	8,4 %	5,3 %	9,1 %	1,0 %	8,2 %	6,3 %	8,2 %	1,1 %
balance					10,3 %	9,9 %	10,6 %	1,8 %
act. sist.	38,7 %	29,2 %	42,0 %	38,3 %	42,3 %	41,0 %	42,6 %	40,5 %
solver lineal	32,9 %	29,7 %	34,0 %	27,8 %	9,8 %	8,1 %	13,4 %	31,5 %
caudales	5,8 %	4,3 %	6,4 %	3,7 %	10,8 %	10,8 %	10,4 %	6,8 %
alturas					8,8 %	9,6 %	8,1 %	5,8 %
estado	4,3 %	2,6 %	4,7 %	1,6 %	4,1 %	3,0 %	4,0 %	1,6 %
Tiempo total	0,834	1,001	0,796	0,976	0,864	0,849	0,902	0,880

4 este menor tamaño se ve compensado por la presencia de 10 VRP, que hacen que la matriz tenga una parte no simétrica de tamaño considerable.

El análisis realizado en [33], correspondiente al método GGA, coincide en señalar la actualización del sistema lineal, y más concretamente el cálculo de los coeficientes de pérdidas de carga, como la tarea más costosa de la simulación hidráulica, con entre un 25 % y 42 % del tiempo de cálculo total. En dicho análisis el peso de la resolución del sistema lineal resulta algo menor de lo observado en esta tesis, estando entre 16 % y 20 %.

Las tareas consideradas para su paralelización en este capítulo se muestran en negrita en la tabla 6.2 (*demandas*, *actualización del sistema* y *caudales*). Esta selección se ha realizado teniendo en cuenta tanto el beneficio potencial que es de esperar de su paralelización, como la dificultad de dicha paralelización.

Inicialmente se consideró también la realización en paralelo de la resolución del sistema lineal, de forma similar a lo realizado en el capítulo 3. La paralelización se basó en una aproximación multifrontal mediante árboles de eliminación, seleccionando el *solver* `HLS_MA87`, de la librería matemática HSL[1], como una implementación paralela adecuada con OpenMP. Sin embargo, los resultados fueron decepcionantes, siendo la versión paralela más lenta que la secuencial. Esto confirma lo expuesto en [13], que concluye que el *solver* lineal de Epanet es el más rápido para las redes reales que se encuentran en la práctica, y también en [33], donde se señala que el *speedup* que se puede obtener depende del tamaño y dispersidad de la matriz, y que en este caso la ganancia obtenida podría ser nula o mínima, teniendo en cuenta el tamaño relativamente pequeño de los sistemas lineales producidos por el método GGA.

Respecto a la posible paralelización de otras tareas, el peso de la tarea de *inicialización* depende del periodo de simulación considerado, siendo pequeño en el caso de simulaciones largas. Las tareas de *balance* de caudales y actualización de *alturas* suponen recorrer un árbol de nudos de la red. Esto se hace de forma muy eficiente

[1] http://www.hsl.rl.ac.uk/

Tabla 6.3: *Speedup* máximo con la paralelización propuesta. *max S(p)* indica el *speedup* máximo con p núcleos.

	GGA				método de mallas			
	red1	**red2**	**red3**	**red4**	**red1**	**red2**	**red3**	**red4**
max S(2)	1,36	1,24	1,40	1,27	1,44	1,41	1,44	1,32
max S(4)	1,66	1,41	1,76	1,48	1,85	1,78	1,85	1,57
max S(8)	1,86	1,51	2,01	1,60	2,16	2,04	2,15	1,74
max S(16)	1,98	1,57	2,17	1,68	2,35	2,20	2,35	1,83
max S(32)	2,05	1,60	2,26	1,71	2,46	2,29	2,46	1,88

Tabla 6.4: Eficiencia máxima con la paralelización propuesta. *max E(p)* indica la eficiencia máxima con p núcleos.

	GGA				método de mallas			
	red1	**red2**	**red3**	**red4**	**red1**	**red2**	**red3**	**red4**
max E(2)	0,68	0,62	0,70	0,64	0,72	0,70	0,72	0,66
max E(4)	0,41	0,35	0,44	0,37	0,46	0,44	0,46	0,39
max E(8)	0,23	0,19	0,25	0,20	0,27	0,25	0,27	0,22
max E(16)	0,12	0,10	0,14	0,10	0,15	0,14	0,15	0,11
max E(32)	0,06	0,05	0,07	0,05	0,08	0,07	0,08	0,06

en el código secuencial, debido a que el orden en el que recorren los nudos ha sido precalculado, lo que significa que una versión paralela podría producir una ganancia mínima o incluso inexistente, excepto en el caso de redes muy grandes.

Para finalizar esta sección, se presenta en la tabla 6.3 el *speedup* máximo que se puede alcanzar con la paralelización propuesta. Estos valores han sido calculados de acuerdo con la ley de Amdahl [9], teniendo en cuenta los porcentajes de tiempo correspondientes a las partes secuenciales y paralelas. Por su parte, la tabla 6.4 muestra los valores de eficiencia correspondientes. Como puede observarse, la eficiencia baja mucho al aumentar el número de procesadores, debido a las tareas que se realizan de forma secuencial. Se trata por tanto de una aproximación de paralelización útil para sistemas con un número de procesadores (núcleos) medio o bajo, que es lo habitual en sistemas *multicore*.

6.3. Tareas a paralelizar

En este apartado se estudia el código secuencial que es objeto de paralelización en ambos métodos.

6.3.1. Actualización del sistema lineal en el método GGA

Revisamos aquí la tarea de actualización del sistema lineal, teniendo en cuenta tanto las pérdidas de carga, como los valores de caudales, demandas y niveles de depósitos.

La actualización del sistema de ecuaciones lineales en cada iteración implica el cálculo de dos valores, p_k y y_k, para cada línea, y el ensamblaje de los mismos en la matriz y el vector del sistema [58]. Estos valores están relacionados con la pérdida de carga que se produce en una línea, expresada como función del caudal que circula por ella. Recuérdese que en una tubería entre los nudos i y j, la pérdida de carga viene dada por la ecuación (2.1):

$$h_i - h_j = q_k(r_k|q_k|^{\beta-1} + \rho_k|q_k|)$$

donde k es el índice de línea de la tubería, r_k y ρ_k son coeficientes de resistencia de la tubería, y β es un exponente constante. En el caso de bombas, la pérdida de carga (ganancia de carga en negativo), corresponde a la ecuación (2.2):

$$h_i - h_j = -\omega_k^2(h_{0k} - r_k(q_k/\omega_k)^{\gamma_k})$$

donde h_{0k} es la altura de corte para la bomba, ω_k es la velocidad relativa, y r_k y γ_k son los coeficientes de la curva característica.

A partir de las expresiones anteriores se obtiene el valor p_k de una línea k, que corresponde a la inversa de la derivada con respecto del caudal de las pérdidas en la línea, así como y_k, que es el producto de p_k por la pérdida de carga. Es decir, en el caso de una tubería, tenemos que:

$$p_k = \frac{1}{\beta r_k|q_k|^{\beta-1} + 2\rho_k|q_k|}, \quad y_k = p_k q_k(r_k|q_k|^{\beta-1} + \rho_k|q_k|) \tag{6.1}$$

mientras que en una bomba las expresiones correspondientes son:

$$p_k = \frac{1}{\gamma_k r_k \omega_k^{2-\gamma_k} q_k^{\gamma_k-1}}, \quad y_k = -p_k\omega_k^2(h_{0k} - r_k(q_k/\omega_k)^{\gamma_k}) \tag{6.2}$$

Una vez obtenidos los valores anteriores, se deben ensamblar en el sistema. En concreto, en el método del gradiente se resuelve un sistema lineal $Ax = b$, donde los elementos de la matriz Jacobiana A son:

$$a_{ii} = \sum_{k \in I_i} p_k$$

$$a_{ij} = -\sum_{k \in I_{i,j}} p_k, \quad j \neq i$$

donde I_i denota el conjunto de líneas conectadas al nudo i, y $I_{i,j}$ corresponde al conjunto de líneas que tienen el nudo i en un extremo y el nudo j en el otro. En la mayoría de los casos el conjunto $I_{i,j}$ será, o bien un conjunto vacío, si no hay ninguna

línea que una los nudos, o bien un conjunto con un solo elemento. Sin embargo, no es raro el caso en que el conjunto está formado por varios elementos, correspondientes a múltiples líneas con los mismos nudos extremos (líneas paralelas).

Por su parte, cada elemento del vector b de términos independientes consiste en el desequilibrio del caudal neto en el nudo más un factor de corrección de caudal:

$$b_i = \left(\sum_{k \in I_i^+} q_k - \sum_{k \in I_i^-} q_k - c_i \right) - \sum_{k \in I_i^+} y_k + \sum_{k \in I_i^-} y_k + \sum_{k \in I_i^f} p_k h_j$$

donde I_i^+, I_i^- representan el conjunto de líneas que entran y salen, respectivamente, del nudo i, I_i^f es el conjunto de líneas que tienen el nudo i en un extremo y un depósito/embalse en el otro, y j hace referencia al otro extremo de la línea k (el extremo distinto del nudo i).

De acuerdo con lo anterior, la actualización del sistema lineal se puede llevar a cabo mediante el proceso que se describe en el algoritmo 6.3, que corresponde básicamente al código que utiliza Epanet [59].

El algoritmo tiene en cuenta que la matriz A es simétrica, por lo que sólo almacena la parte triangular inferior de la misma. Además, la matriz es dispersa, por lo que una implementación eficiente debe usar almacenamiento disperso. En concreto, Epanet utiliza formato CSC (*Compressed Sparse Column*). Además, se utiliza un vector (d) para los elementos de la diagonal, y otro (v) para los de la parte no diagonal. El algoritmo también considera que las filas de la matriz están reordenadas de alguna manera (ordenación que se usa para minimizar el llenado al resolver el sistema lineal).

La utilización de un formato disperso hace necesario el uso de estructuras de datos que proporcionen un acceso eficiente a los elementos de la matriz, permitiendo obtener rápidamente el elemento que corresponde a un nudo dado o a una línea determinada. Para ello, Epanet utiliza un vector que proporciona, para cada línea, la posición del correspondiente elemento $a_{i,j}$ en la matriz. El elemento que corresponde a un nudo (situado en la diagonal) se obtiene de forma directa, dado que la diagonal se mantiene en un vector aparte.

6.3.2. Actualización del sistema lineal en el método de mallas

El código revisado en este apartado corresponde a la tarea de actualización del sistema lineal, teniendo en cuenta tanto las pérdidas de carga, como los valores de los niveles de depósitos/embalses.

Al igual que en el caso del método GGA, en el método de mallas el sistema se construye ensamblando en el mismo dos valores para cada línea, a los que llamaremos p'_k, y'_k. El primero de ellos corresponde a la derivada con respecto del caudal de las pérdidas en la línea, mientras que el segundo es la propia pérdida de carga. Por tanto, se obtienen mediante las expresiones:

$$p'_k = \beta r_k |q_k|^{\beta-1} + 2\rho_k |q_k|, \quad y'_k = q_k (r_k |q_k|^{\beta-1} + \rho_k |q_k|) \tag{6.3}$$

en el caso de tuberías, o bien

$$p'_k = \gamma_k r_k \omega_k^{2-\gamma_k} q_k^{\gamma_k - 1}, \quad y'_k = -\omega_k^2 (h_{0k} - r_k (q_k/\omega_k)^{\gamma_k}) \tag{6.4}$$

Algoritmo 6.3 Actualización del sistema lineal en el método GGA

Entrada: Parámetros de las líneas $(r, \rho, h_0, \omega, \gamma)$, vector de caudales (q), demandas (c). Nudos inicial y final de cada línea. Estructura dispersa para la matriz A, con vectores para elementos diagonales (d) y no diagonales (v). Correspondencia entre nudos y filas de A, b.

Salida: Matriz A, vector b.

inicializar A, b, a ceros
para todo línea k **hacer**
  calcular p_k, y_k mediante ecs. (6.1), (6.2)
  Sean i, j los nudos inicial y final, respectivamente, de la línea k
  si los nudos i, j son ambos uniones **entonces**
    $v_{k'} \leftarrow v_{k'} - p_k$, donde k' es el índice del elemento no nulo corresp. a la línea k
  fin si
  sean i', j' las filas de los nudos i, j resp. (si los nudos i, j son uniones)
  si el nudo i es una unión **entonces**
    $d_{i'} \leftarrow d_{i'} + p_k$
    $b_{i'} \leftarrow b_{i'} - q_k + y_k$
  si no
    $b_{j'} \leftarrow b_{j'} + p_k h_i$
  fin si
  si el nudo j es una unión **entonces**
    $d_{j'} \leftarrow d_{j'} + p_k$
    $b_{j'} \leftarrow b_{j'} + q_k - y_k$
  si no
    $b_{i'} \leftarrow b_{i'} + p_k h_j$
  fin si
fin para
para todo unión i **hacer**
  $b_{i'} \leftarrow b_{i'} - c_i$, donde i' es la fila de la unión i
fin para

en el caso de bombas.

Una vez obtenidos los valores anteriores, se deben ensamblar en el sistema. En el método de mallas los elementos de la matriz Jacobiana A son:

$$a_{ii} = \sum_{k \in I_i^m} p'_k \tag{6.5}$$

$$a_{ij} = \sum_{k \in I_{i,j}^m} \mu_{ik}\mu_{jk}p'_k, \quad j \neq i \tag{6.6}$$

donde I_i^m denota el conjunto de líneas que forman parte de la malla i, $I_{i,j}^m$ corresponde al conjunto de líneas compartidas por las mallas i y j, mientras que μ_{ik} es cero si la línea k no forma parte de la malla i, y en caso contrario toma el valor $+1$ o -1, según si la línea tiene el mismo sentido o sentido contrario, respectivamente, que la malla.

Por su parte, cada elemento del vector b de términos independientes corresponde a la pérdida de carga a lo largo de una malla, en la iteración actual:

$$b_i = -\sum_{k \in I_i^m} \mu_{ik}y'_k - \sum_{j \in I_i^{m,f}} \mu_{ij}^f h_j$$

donde, si la malla i es una *pseudo-malla* (un camino entre dos depósitos/embalses), $I_i^{m,f}$ contiene los depósitos en sus extremos, y en caso contrario es un conjunto vacío. Por su parte, μ_{ij}^f toma el valor -1 o $+1$, según el depósito j esté en el extremo inicial o final, respectivamente, de la malla i.

Por otra parte, la consideración de válvulas como VCQ, VRP, VSP y líneas temporalmente cerradas se ha hecho de acuerdo con lo descrito en el capítulo 4, utilizando la aproximación no simétrica para las VRP/VSP.

El proceso completo de actualización del sistema lineal se describe en el algoritmo 6.4. El algoritmo requiere obtener el índice de un elemento no diagonal a partir de la pareja de mallas correspondiente, lo cual puede ser poco eficiente. Una mejor manera de hacerlo sería almacenar, para cada línea, la posición de los elementos no nulos a los que contribuye la línea, junto con el signo de la contribución. Esto se puede hacer al determinar la estructura de la matriz, tal como se explica a continuación.

Al inicio de la simulación se crea la lista de adyacencias entre mallas. Cada adyacencia corresponde a una pareja de mallas que comparten una o más líneas, lo que da lugar a un elemento no nulo en la matriz. Al construir la lista de adyacencias se puede asociar a cada una el correspondiente conjunto de líneas compartidas, tal como se presenta en el algoritmo 6.5.

A continuación se realiza la fase de reordenación y descomposición simbólica, con lo que la lista de adyacencias se expande con nuevos elementos no nulos correspondientes a la factorización. Estas nuevas adyacencias no tienen líneas asociadas. Una vez terminada esta fase se crea la estructura de datos necesaria para almacenar la matriz en formato disperso, determinando la posición de cada elemento no nulo, y en ese momento se puede asociar a cada línea las posiciones de sus elementos no nulos.

Algoritmo 6.4 Actualización del sistema lineal en el método de mallas

Entrada: Parámetros de las líneas $(r, \rho, h_0, \omega, \gamma)$, vector de caudal q. Lista de mallas para cada línea, con signos correspondientes μ_{ij}. Lista de pseudo-mallas para cada depósito, con signos correspondientes μ^f_{ij}. Estructura dispersa para la matriz A, con vectores para elementos diagonales (d) y no diagonales (v). Correspondencia entre mallas y filas de A, b.

Salida: Matriz A, vector b.

inicializar A, b a ceros.
para todo línea k **hacer**
  calcular p'_k, y'_k mediante ecs. (6.3), (6.4)
  para todo malla i que contiene la línea k **hacer**
    sea i' la fila de la malla i
    $d_{i'} \leftarrow d_{i'} + p'_k, \qquad b_{i'} \leftarrow b_{i'} - \mu_{ik} y'_k$
    para todo malla $j < i$ que contiene la linea k **hacer**
      sea k' el índice del elem. no nulo corresp. a la pareja de mallas i, j
      $v_{k'} \leftarrow v_{k'} + \mu_{ik}\mu_{jk} p'_k$
    fin para
  fin para
fin para
para todo depósito/embalse j **hacer**
  para todo malla i con el nudo j como extremo **hacer**
    $b_{i'} \leftarrow b_{i'} - \mu^f_{ij} h_j$, donde i' es la fila de la malla i
  fin para
fin para

Algoritmo 6.5 Generación de las listas de adyacencias de mallas

Entrada: Lista de mallas para cada línea, con signos correspondientes μ_{ij}.
Salida: Listas de adyacencias: para cada malla, lista de mallas adyacentes, con las líneas compartidas con cada una y los signos correspondientes.

```
para todo malla i hacer
  adyacencias de malla i ← ∅
fin para
para todo línea k hacer
  para todo malla i que contiene la línea k hacer
    para todo malla j > i que contiene la línea k hacer
      sean μ_ik, μ_jk los signos de la línea k en i, j resp.
      s ← μ_ik μ_jk
      si la malla j está en la lista de ady. de la malla i entonces
        añadir la línea k, con signo s, a la adyacencia
      si no
        añadir adyacencia con malla j a la lista de ady. de la malla i
        añadir la línea k, con signo s, a la adyacencia
      fin si
    fin para
  fin para
fin para
```

Teniendo esto en cuenta, el algoritmo de actualización del sistema lineal 6.4 se transformaría en el algoritmo 6.6, más eficiente.

6.3.3. Actualización de caudales

En el método GGA, la actualización de los caudales se realiza por medio del algoritmo 6.7, que aplica a cada una de las líneas el siguiente incremento de caudal:

$$\Delta q_k = -y_k + p_k(h_i - h_j), \qquad 1 \leq k \leq m \tag{6.7}$$

donde i, j son los nudos inicial y final, respectivamente, de la línea k. Además, se calcula el siguiente cociente, que corresponde al cambio relativo de la suma de caudales en la iteración actual, y se utiliza en la comprobación de convergencia del algoritmo GGA:

$$\frac{\sum_{k=1}^{m} |\Delta q_k|}{\sum_{k=1}^{m} |q_k|} \tag{6.8}$$

Por otra parte, la actualización de los caudales en el método de mallas se realiza por medio del algoritmo 6.8, el cual aplica a cada línea un incremento de caudal igual a la suma de las correcciones de caudal de las mallas a las que pertenece la línea, es decir:

$$\Delta q_k = \sum_i \mu_{ik} \hat{q}_i, \qquad 1 \leq k \leq m \tag{6.9}$$

Algoritmo 6.6 Actualización del sistema lineal en el método de mallas

Entrada: Parámetros de las líneas $(r, \rho, h_0, \omega, \gamma)$, vector de caudal q. Lista de mallas para cada línea, con los signos corresp. μ_{ij}. Lista de pseudo-mallas para cada depósito, con signos corresp. μ^f_{ij}. Estructura dispersa para la matriz A, con vectores para elementos diagonales (d) y no diagonales (v). Correspondencia entre mallas y filas de A, b. Contribuciones de cada línea a elems. no diagonales.
Salida: Matriz A, vector b.

inicializar A, b, a ceros.
para todo línea k **hacer**
  calcular p'_k, y'_k mediante ecs. (6.3), (6.4)
  para todo malla i donde está la línea k **hacer**
    sea i' la fila de la malla i
    $d_{i'} \leftarrow d_{i'} + p'_k, \qquad b_{i'} \leftarrow b_{i'} - \mu_{ik}\, y'_k$
  fin para
  para todo elemento no diagonal i al que contribuye la línea k **hacer**
    $v_i \leftarrow v_i + s\, p'_k$, donde s es el signo de la contribución
  fin para
fin para
para todo depósito/embalse j **hacer**
  para todo malla i con el nudo j como extremo **hacer**
    $b_{i'} \leftarrow b_{i'} - \mu^f_{ij} h_j$, donde i' es la fila de la malla i
  fin para
fin para

Algoritmo 6.7 Actualización de caudales en el método GGA

Entrada: Vectores p, y, de coeficientes relacionados con las pérdidas de carga. Vectores h, q de alturas piezométricas y caudales, respectivamente.
Salida: Vector q actualizado. Variación relativa en los caudales, e.

$s \leftarrow 0, \quad s' \leftarrow 0$
para todo línea k **hacer**
  Sean i, j los nudos inicial y final, respectivamente, de la línea k
  $\Delta q_k \leftarrow -y_k + p_k(h_i - h_j)$
  $q_k \leftarrow q_k + \Delta q_k$
  $s \leftarrow s + |q_k|$
  $s' \leftarrow s' + |\Delta q_k|$
fin para
$e \leftarrow s'/s$

donde i itera sobre cada una de las mallas a las que pertenece la línea k, y $\hat{q}_i$ es la corrección de caudal asociada a la malla i. Se calcula también el cambio relativo en la suma de caudales, usando la ecuación (6.8).

Algoritmo 6.8 Actualización de caudales en el método de mallas

Entrada: Vector q, caudales actuales. Vector $\hat{q}$, caudales correctores de malla. Lista de mallas para cada línea, con signos correspondientes μ_{ij}.
Salida: Vector q actualizado. Variación relativa en los caudales, e.

$s \leftarrow 0, \quad s' \leftarrow 0$
para todo línea k **hacer**
  $\Delta q_k \leftarrow 0$
  para todo malla i en la que está la línea k **hacer**
    $\Delta q_k \leftarrow \Delta q_k + \mu_{ik}\hat{q}_i$
  fin para
  $q_k \leftarrow q_k + \Delta q_k$
  $s \leftarrow s + |q_k|$
  $s' \leftarrow s' + |\Delta q_k|$
fin para
$e \leftarrow s'/s$

6.3.4. Actualización de demandas

La actualización de las demandas se hace igual en ambos métodos. Las demandas se modelizan usando lo que Epanet llama *patrones*, que constan de un valor base y una secuencia de coeficientes multiplicadores que se aplican sobre el valor base para obtener una modulación temporal. Cada unión puede tener varios patrones de demanda asociados a él, dando como resultado la siguiente expresión de actualización de las demandas en cada iteración:

$$c_i = \sum_j \hat{c}_{i,j}\mu_{i,j,k}, \qquad 1 \leq i \leq n \tag{6.10}$$

donde c_i es la demanda de la unión i, el índice j itera sobre los patrones de demanda de la unión i, $\hat{c}_{i,j}$ es el valor base del patrón y $\mu_{i,j,k}$ es el coeficiente multiplicador del patrón para el paso de tiempo k. Además, en cada paso de tiempo se calcula también la suma de las demandas de todos los nudos de consumos.

Los patrones temporales también se usan para actualizar las alturas de embalses en el caso de que éstas varíen temporalmente, así como para la utilización de las bombas. Las actualizaciones correspondientes de estos elementos también se consideran parte de esta tarea, y se llevan a cabo mediante una fórmula similar a (6.10).

6.4. Algoritmos paralelos

6.4.1. Actualización del sistema lineal en el método GGA

La paralelización directa del bucle que recorre las líneas en el algoritmo secuencial 6.3 es problemática, dado que en general un mismo elemento de la matriz A o del vector b se verá afectado por distintas iteraciones del bucle, por lo que sería necesario utilizar directivas de sincronización para evitar condiciones de carrera. Esto puede conllevar problemas de pérdida de eficiencia.

Una aproximación muy simple consiste en paralelizar sólo el cálculo de los coeficientes p_k, y_k, dejando que el ensamblaje de la matriz y del vector se haga en secuencial. El algoritmo 6.9 presenta el proceso correspondiente, donde se usa la construcción **[en paralelo] para todo**, que correspondería a la directiva `parallel for` de OpenMP (ver apartado 3.3). Esta aproximación puede tener sentido si se tiene en cuenta que el cálculo de los coeficientes es la parte más costosa computacionalmente.

Algoritmo 6.9 Actualización del sistema lineal de GGA con cálculo de coef. en paralelo

---

Entrada: Parámetros de las líneas $(r, \rho, h_0, \omega, \gamma)$, vector de caudales (q), demandas (c). Nudos inicial y final de cada línea. Estructura dispersa para la matriz A, con vectores para elementos diagonales (d) y no diagonales (v). Correspondencia entre nudos y filas de A, b.
Salida: Matriz A, vector b
  inicializar A, b, a ceros
  [en paralelo] para todo línea k **hacer**
    calcular p_k, y_k mediante ecs. (6.1), (6.2)
  fin para
  para todo línea k **hacer**
    Sean i, j los nudos inicial y final, respectivamente, de la línea k
    actualizar $d_{i'}, d_{j'}, v_{k'}, b_{i'}, b_{j'}$ igual que en algoritmo 6.3
  fin para
  para todo unión i **hacer**
    $b_{i'} \leftarrow b_{i'} - c_i$, donde i' es la fila de la unión i
  fin para

Otra posibilidad es hacer una paralelización completa del algoritmo, manteniendo la estructura del algoritmo secuencial 6.3, y haciendo uso de directivas de sincronización de OpenMP para evitar condiciones de carrera en actualizaciones simultáneas por parte de varios hilos. La forma en que se actualizan los elementos hace que baste con utilizar la directiva `atomic`, tal como puede verse en el algoritmo 6.10. Sobre este algoritmo se podría realizar una pequeña optimización adicional para evitar que en una misma iteración pueda haber dos actualizaciones de un mismo elemento del vector b.

Por último, una tercera alternativa, en principio la más eficiente, consiste en reorganizar el código secuencial para poder llevar a cabo la paralelización sin usar

Algoritmo 6.10 Actualización paralela del sistema lineal de GGA con directivas de sincronización.

Entrada: Parámetros de las líneas $(r, \rho, h_0, \omega, \gamma)$, vector de caudales (q), demandas (c). Nudos inicial y final de cada línea. Estructura dispersa para la matriz A, con vectores para elementos diagonales (d) y no diagonales (v). Correspondencia entre nudos y filas de A, b.

Salida: Matriz A, vector b

inicializar A, b, a ceros
[en paralelo] para todo línea k **hacer**
  calcular p_k, y_k mediante ecs. (6.1), (6.2)
  Sean i, j los nudos inicial y final, respectivamente, de la línea k
  si los nudos i, j son ambos uniones **entonces**
    sea k' el índice del elemento no nulo corresp. a la línea k
    [atómico] $v_{k'} \leftarrow v_{k'} - p_k$
  fin si
  sean i', j' las filas de los nudos i, j resp. (si los nudos i, j son uniones)
  si el nudo i es una unión **entonces**
    [atómico] $d_{i'} \leftarrow d_{i'} + p_k$
    [atómico] $b_{i'} \leftarrow b_{i'} - q_k + y_k$
  si no
    [atómico] $b_{j'} \leftarrow b_{j'} + p_k h_i$
  fin si
  si el nudo j es una unión **entonces**
    [atómico] $d_{j'} \leftarrow d_{j'} + p_k$
    [atómico] $b_{j'} \leftarrow b_{j'} + q_k - y_k$
  si no
    [atómico] $b_{i'} \leftarrow b_{i'} + p_k h_j$
  fin si
fin para
[en paralelo] para todo unión i **hacer**
  [atómico] $b_{i'} \leftarrow b_{i'} - c_i$, donde i' es la fila de la unión i
fin para

directivas específicas de sincronización. La idea es agrupar las actualizaciones que se realizan sobre un mismo elemento y, una vez hecho esto, aplicar las directivas de bucles paralelos. El resultado es el algoritmo 6.11, donde las actualizaciones se organizan en cuatro bucles diferentes. El primero de ellos simplemente calcula los coeficientes p_k y y_k, pudiendo realizarse en paralelo sin problemas. El segundo bucle, que empieza en la línea 5, calcula los elementos de la diagonal de la matriz y el vector b. Cada iteración del bucle calcula un elemento diagonal diferente $d_{j'}$ y su correspondiente elemento del vector $b_{j'}$, de manera que no hay conflicto entre iteraciones al realizarlas concurrentemente.

Los dos últimos bucles calculan los elementos no diagonales. Hay que tener en cuenta que un coeficiente no diagonal determinado puede estar afectado por varias líneas, aunque esto ocurre raramente. Para tratar esta situación, el enfoque adoptado consiste en considerar inicialmente sólo la contribución de la primera línea para cada coeficiente no diagonal, lo que se realiza en paralelo en el tercer bucle (línea 16), y a continuación considerar las contribuciones del resto de líneas en el último bucle secuencial.

Obsérvese que en este algorimo es necesario disponer de una estructura de datos que nos permita obtener de forma eficiente la lista de líneas conectadas a cada nudo. También hará falta recorrer los elementos no nulos de la parte no diagonal, obteniendo para cada uno la primera de las líneas que contribuye a él, así como tener una lista con el resto de líneas.

6.4.2. Actualización del sistema lineal en el método de mallas

Como ocurre en el caso del método GGA, el algoritmo secuencial 6.6 de actualización del sistema resulta poco adecuado para una paralelización directa, debido a que en general un mismo elemento del sistema se verá modificado por varias iteraciones del bucle de líneas.

Por tanto, se plantea reorganizar el algoritmo, de forma que se agrupen en una sola iteración las distintas actualizaciones sobre un mismo elemento del sistema. Sobre ese algoritmo transformado se aplican las directivas de bucles paralelos, tal como puede verse en el algoritmo 6.12.

El primer bucle de dicho algoritmo corresponde, como en el caso del método GGA, al cálculo de los dos coeficientes de cada línea, y no plantea problemas para su paralelización. El siguiente bucle, que recorre adyacencias, corresponde a la actualización de los elementos no diagonales de la matriz. Obsérvese que cada elemento no nulo, $a_{i,j}$ con $j \neq i$, de la matriz, se ve afectado por todas las líneas comunes a las mallas i, j, tal como se expresa en (6.6), es decir, las líneas asociadas a la adyacencia entre las mallas. Por tanto, para agrupar las actualizaciones de un elemento hay que construir listas de adyacencias, asociando a cada adyacencia la posición del elemento no nulo $a_{i,j}$ correspondiente, así como las líneas asociadas a la adyacencia, junto con el signo con el que cada línea contribuye al elemento no nulo. Dichas listas de adyacencias pueden obtenerse mediante el algoritmo 6.5, a falta de incorporar la posición del elemento no nulo correspondiente a cada adyacencia, lo cual debe hacerse después de la fase de reordenación y descomposición simbólica de la matriz.

Algoritmo 6.11 Actualización paralela del sistema lineal de GGA sin directivas de sincronización.

Entrada: Parámetros de las líneas $(r, \rho, h_0, \omega, \gamma)$, vector de caudales (q), demandas (c). Extremos de cada línea. Líneas conectadas a cada unión. Estructura dispersa para la matriz A, con vectores para elementos diagonales (d) y no diagonales (v). Correspondencia entre nudos y filas de A, b. Líneas asociadas a cada elemento no diagonal.

Salida: Matriz A, vector b

1: inicializar A, b a ceros.
2: **[en paralelo] para todo** línea k **hacer**
3: calcular p_k, y_k mediante ecs. (6.1), (6.2)
4: **fin para**
5: **[en paralelo] para todo** unión j **hacer**
6: sea j' la fila de la unión j
7: **para todo** línea k entre el nudo j y algún otro nudo i **hacer**
8: $d_{j'} \leftarrow d_{j'} + p_k$
9: $b_{j'} \leftarrow b_{j'} + \delta_{kj}(q_k - y_k)$
10: **si** nudo i es un depósito/embalse **entonces**
11: $b_{j'} \leftarrow b_{j'} + p_k h_i$
12: **fin si**
13: **fin para**
14: $b_{j'} \leftarrow b_{j'} - c_j$
15: **fin para**
16: **[en paralelo] para todo** elemento no nulo k' de la parte no diagonal **hacer**
17: $v_{k'} \leftarrow v_{k'} - p_k$, donde k es la primera línea del elemento no nulo
18: **fin para**
19: **para todo** línea k no considerada en el bucle anterior **hacer**
20: $v_{k'} \leftarrow v_{k'} - p_k$, donde k' es el índice de elemento no diagonal de la línea k
21: **fin para**

Algoritmo 6.12 Actualización paralela del sistema lineal en el método de mallas.

Entrada: Parámetros de las líneas $(r, \rho, h_0, \omega, \gamma)$, vector de caudal q. Lista de líneas de cada malla, con signos correspondientes μ_{ij}. Depósitos/embalses en los extremos de las pseudo-mallas. Listas de adyacencias. Estructura dispersa para la matriz A, con vectores para elementos diagonales (d) y no diagonales (v). Correspondencia entre mallas y filas de A, b.

Salida: Matriz A, vector b

1: **[en paralelo] para todo** línea k **hacer**
2: calcular p'_k, y'_k mediante ecs. (6.3), (6.4)
3: **fin para**
4: **[en paralelo] para todo** adyacencia entre dos mallas **hacer**
5: sea i la posición del elemento no nulo asociado a la adyacencia
6: **para todo** línea k asociada a la adyacencia **hacer**
7: sea s el signo de la línea en la adyacencia
8: $v_i \leftarrow v_i + sp'_k$
9: **fin para**
10: **fin para**
11: **[en paralelo] para todo** malla i **hacer**
12: sea i' la fila de la malla i
13: **para todo** línea k en la malla i **hacer**
14: $d_{i'} \leftarrow d_{i'} + p'_k$
15: $b_{i'} \leftarrow b_{i'} - \mu_{ik} y'_k$
16: **fin para**
17: **si** la malla i es una pseudo-malla entre dos depósitos/embalses **entonces**
18: sean j, j', los extremos inicial y final, resp. de la pseudo-malla
19: $b_{i'} \leftarrow b_{i'} + h_j - h_{j'}$
20: **fin si**
21: **fin para**

Finalmente, el tercer bucle del algoritmo, que recorre mallas, se encarga de las actualizaciones sobre los elementos diagonales de la matriz y sobre el vector de términos independientes. Este vector se ve afectado tanto por los coeficientes y_k de las líneas, como por las alturas de los depósitos situados en los extremos de las pseudo-mallas.

Hay que decir que la paralelización de los bucles de adyacencias (línea 4) y de mallas (línea 11) del algoritmo 6.12 no se ha hecho mediante la directiva `for` de OpenMP, ya que se ha buscado hacer un reparto de iteraciones entre hilos diferente a las opciones que proporciona la cláusula `schedule` de OpenMP. En concreto, la iteración inicial y final para cada hilo se calculan para que la carga (número de operaciones de actualización sobre v, d o b) sea lo más equilibrada posible. Este cálculo se hace al principio de la simulación.

6.4.3. Actualización de caudales

La paralelización del algoritmo de actualización de caudales en el método GGA es sencilla, tal como se muestra en el algoritmo 6.13. Hay que tener en cuenta únicamente que debe aplicarse la cláusula de reducción de OpenMP (`reduction`) para acumular las sumas realizadas por los distintos hilos sobre las variables s y s'.

Algoritmo 6.13 Actualización en paralelo de caudales en el método GGA

Entrada: Vectores p, y, de coeficientes relacionados con las pérdidas de carga.
Vector h de alturas piezométricas.
Vector q, aproximación actual de caudales de línea.
Salida: Vector q actualizado.
Variación relativa en los caudales, e.

$s \leftarrow 0, \quad s' \leftarrow 0$
[en paralelo] para todo [reducción(+:s, s')] línea k **hacer**
  Sean i, j los nudos inicial y final, respectivamente, de la línea k
  $\Delta q_k \leftarrow -y_k + p_k(h_i - h_j)$
  $q_k \leftarrow q_k + \Delta q_k$
  $s \leftarrow s + |q_k|$
  $s' \leftarrow s' + |\Delta q_k|$
fin para
$e \leftarrow s'/s$

En el caso del método de mallas la paralelización de la actualización de caudales es prácticamente igual que en el método GGA, por lo que no se considera necesario incluir el algoritmo paralelo. Al igual que en el caso de GGA, se hace uso de la cláusula reduction de OpenMP.

6.4.4. Actualización de demandas

La tarea de actualización de demandas corresponde a la ecuación (6.10) en ambos métodos. Dado que cada demanda c_i se puede calcular de forma independiente, la paralelización es bastante directa. La suma total de las demandas se calcula haciendo uso de nuevo de la cláusula `reduction`. La actualización de otros elementos controlados por patrones como alturas de embalses y algunos tipos de bombas se realiza con una fórmula similar a (6.10), y también se ha paralelizado.

6.5. Resultados

Para los resultados de este capítulo se consideran las redes de la tabla 6.1, que corresponden a las redes 3 a 6 del capítulo 4. Como se ha comentado, se descartan en este caso las redes 1 y 2 de aquel capítulo, por ser demasiado pequeñas como para poder beneficiarse de la computación paralela. Se ha tomado Epanet (versión 2.00.12) como implementación secuencial para el método GGA, mientras que la implementación secuencial del método de mallas ha sido la desarrollada como parte de esta tesis, incorporando las aportaciones descritas en el capítulo 4.

Todas las simulaciones se han realizado sobre un computador con dos procesadores Intel® Xeon® E5-2697 v3 a 2,60GHz y 14 núcleos (*cores*) cada uno (28 núcleos en total). Se trata de la misma máquina utilizada para los resultados del capítulo 4, y de nuevo se utiliza el compilador de Intel (`icc`), y la opción de optimización del compilador `-O2`. Los tiempos presentados están en segundos.

La tabla 6.5 muestra el *speedup* para la simulación completa para los métodos de GGA y de mallas. Puede verse que el *speedup* obtenido no está lejos del ideal mostrado en la tabla 6.3, llegando a ser de 1,65 para GGA y 1,92 para el método de mallas, con respecto a los correspondientes códigos secuenciales. Eso supone una reducción del tiempo de cálculo de 39 % para GGA y 48 % para el método de mallas. También puede verse que el tiempo secuencial para el método de mallas se reduce con respecto a la tabla 6.2, lo que se debe al hecho de que la reorganización de código del algoritmo 6.12 condujo también a un código secuencial más rápido.

En todas las redes los mejores tiempos corresponden al método de mallas paralelo. En la tabla 6.6 se presenta la ganancia de velocidad conseguida por el método de mallas paralelo con respecto al código secuencial de Epanet. En este caso el *speedup* está entre 1,78 y 2,45, dependiendo de la red, lo que equivale a una reducción del tiempo de cálculo de entre 44 % y 59 %. Estos resultados son claramente mejores que los de otros trabajos previos analizados en la sección 6.1, donde la máxima reducción de tiempo de cómputo con respecto a Epanet era de 19,5 %.

La tabla 6.7 analiza las prestaciones de la parte de código que se ha paralelizado, presentando el *speedup* correspondiente a esta parte. Puede verse que se obtienen mejores *speedup* en caso del método de mallas.

Tabla 6.5: Tiempo secuencial **tsec** y *speedup* **S(p)** con **p** núcleos para la simulación completa.

	GGA				método de mallas			
	red 1	**red 2**	**red 3**	**red 4**	**red 1**	**red 2**	**red 3**	**red 4**
tsec	0,837	1,005	0,797	0,981	0,771	0,747	0,814	0,808
S(2)	1,21	1,14	1,17	1,20	1,31	1,30	1,31	1,24
S(4)	1,37	1,27	1,32	1,34	1,60	1,54	1,55	1,42
S(8)	1,55	1,36	1,50	1,46	1,80	1,74	1,75	1,54
S(14)	1,63	1,40	1,59	**1,52**	**1,92**	1,81	**1,82**	1,59
S(20)	**1,65**	**1,41**	**1,60**	1,52	1,89	**1,82**	1,79	**1,60**
S(28)	1,56	1,36	1,53	1,46	1,75	1,73	1,67	1,56

Tabla 6.6: *Speedup* del método de mallas paralelo con respecto Epanet.

	red 1	**red 2**	**red 3**	**red 4**
S(2)	1,42	1,75	1,28	1,51
S(4)	1,74	2,07	1,52	1,72
S(8)	1,95	2,34	1,71	1,87
S(14)	**2,08**	2,44	**1,78**	1,93
S(20)	2,05	**2,45**	1,75	**1,94**
S(28)	1,90	2,33	1,64	1,89

Tabla 6.7: Tiempo secuencial **tsec** y *speedup* **S(p)** con **p** núcleos para la parte paralelizada.

	GGA				método de mallas			
	red 1	**red 2**	**red 3**	**red 4**	**red 1**	**red 2**	**red 3**	**red 4**
tsec	0,440	0,386	0,456	0,420	0,443	0,404	0,467	0,371
S(2)	1,57	1,58	1,42	1,71	1,81	1,88	1,83	1,88
S(4)	2,41	2,58	1,98	2,79	3,27	3,35	3,10	3,38
S(8)	4,05	4,38	3,02	4,95	5,70	6,09	4,95	6,04
S(14)	5,81	6,41	3,83	7,38	8,73	9,45	6,08	8,98
S(20)	**6,84**	**7,73**	**4,14**	**8,77**	**10,37**	11,80	**6,09**	11,30
S(28)	6,20	6,91	3,92	6,91	8,88	**13,67**	4,95	**13,10**

6.6. Conclusiones

Este capítulo presenta algoritmos paralelos en memoria compartida para la simulación hidráulica mediante el método GGA y el método de mallas. La implementación utilizada para el método de mallas incorpora las contribuciones presentadas en el capítulo 4 y publicadas en [8]. La selección de las tareas a paralelizar en ambos métodos se ha realizado tras un análisis y medida de prestaciones del código secuencial. Entre estas tareas, la más importante por su influencia en el tiempo de ejecución y por su mayor complejidad de paralelización es la actualización del sistema lineal en ambos métodos.

Los resultados obtenidos muestran mejoras importantes en las prestaciones de la simulación de redes de distribución de agua respecto a Epanet, tanto para la paralelización del método GGA como especialmente para la del método de mallas. La reducción del tiempo de cálculo conseguida es considerablemente mayor que la de otros trabajos previos analizados. En concreto, se consigue una reducción de tiempo de hasta 39 % para GGA y 48 % para el método de mallas, con respecto a los correspondientes códigos secuenciales. Además, la reducción del tiempo de ejecución para el método de mallas paralelo con respecto a Epanet es mayor, estando entre 44 % y 59 %.

Se puede observar que el método de mallas se beneficia en mayor medida de la paralelización, lo que se debe en parte a que la contribución de las tareas paralelizadas al tiempo de ejecución es mayor. La paralelización del método de mallas ha conducido por otra parte a un algoritmo secuencial más rápido como consecuencia de la reestructuración del código de actualización del sistema lineal.

Teniendo en cuenta que diferentes problemas como el diseño de redes o su optimización hacen un uso intensivo de la simulación, el incremento de velocidad mostrado en este capítulo puede tener un gran impacto en el tiempo necesario para resolver esos problemas.

Capítulo 7

Conclusiones

En esta tesis se exploran básicamente tres caminos distintos para mejorar las prestaciones de los procesos de simulación de redes de distribución de agua. En primer lugar, se presentan contribuciones al método de mallas. En segundo lugar, se desarrollan algoritmos paralelos para la simulación de redes en plataformas de memoria distribuida, y por último se hace lo mismo sobre plataformas de memoria compartida.

En este capítulo se presentan las conclusiones obtenidas de la exploración de cada una de esas vías, se revisan las principales aportaciones de la tesis, y se expone el trabajo futuro que se pretende realizar.

7.1. Caminos explorados para la mejora de prestaciones

La primera vía presentada en esta tesis para la mejora de prestaciones de la simulación de redes de distribución de agua ha sido la la consideración de algunas aportaciones en el método de mallas de simulación hidráulica (capítulo 4).

En primer lugar, se presentan métodos para la obtención de un conjunto de mallas adecuado, que conducen a un sistema lineal de gran dispersidad. Los resultados obtenidos sobre distintas redes muestran una importante reducción en el número de elementos no nulos del sistema respecto al método GGA. En todos los casos analizados los resultados están cerca del óptimo. La ventaja clara respecto a otros métodos que obtienen el conjunto de mallas óptimo es la enorme reducción del tiempo de ejecución. Los resultados presentados permiten comprobar que los métodos propuestos son extremadamente rápidos.

En segundo lugar, también en el método de mallas, se desarrollan formulaciones para el tratamiento de válvulas que evitan redefinir el conjunto de mallas independientes. Se considera la modelización de líneas temporalmente cerradas y válvulas de control de caudal, por una parte, y válvulas reductoras/sostenedoras de presión (VRPs/VSPs) por otra. En este último caso se considera una aproximación que

mantiene la simetría de la matriz y otra que produce una matriz con una parte no simétrica. Los resultados con redes hidráulicas que contienen distintos tipos de válvulas muestran la capacidad de los métodos propuestos para realizar satisfactoriamente la simulación con las mismas garantías que el método GGA. Mas aún, la aproximación no simétrica para modelizar las VRPs/VSPs es más precisa que la formulación empleada por Epanet, lo que le lleva a reducir el número de iteraciones.

Las contribuciones de esta tesis al método de mallas han sido evaluadas en términos del tiempo de ejecución necesario para realizar la simulación, tomando distintas redes de test. Los resultados muestran que el método de mallas es competitivo frente al de GGA, siendo más rápido que éste en 4 de las 6 redes analizadas, y prácticamente igual de rápido en otra de ellas. El método de mallas consigue reducir especialmente el tiempo de resolución de los sistemas lineales, como consecuencia directa de los algoritmos desarrollados en la tesis para la selección del conjunto de mallas independientes.

La segunda vía para la mejora de prestaciones consiste en el desarrollo de algoritmos paralelos sobre plataformas de memoria distribuida para la simulación de redes y minimización de fugas de redes de distribución de agua (capítulo 5). Se considera aquí no sólo la simulación hidráulica (caudales y presiones), sino también la de la calidad del agua, y la minimización de fugas mediante el cálculo de las consignas óptimas de las VRP de la red. En este apartado, se presentan resultados que muestran una mejora importante de las prestaciones en la simulación de la calidad y la minimización de fugas. Sin embargo, los resultados correspondientes a la simulación hidráulica son desiguales, dependiendo mucho de las características de la red considerada, que debe tener una complejidad suficiente (alto número de líneas) para poder beneficiarse de los algoritmos paralelos desarrollados.

Finalmente, la tercera vía, que solapa con la primera y se describe en el capítulo 6, busca sortear los problemas mencionados en el párrafo anterior, y consiste en el desarrollo de algoritmos paralelos sobre memoria compartida para la simulación hidráulica, considerando no sólo el método GGA, sino también el método de mallas, con las aportaciones al mismo comentadas anteriormente. Se busca evitar el coste asociado al envío de mensajes en plataformas de memoria distribuida, teniendo en mente especialmente el uso de procesadores *multicore*, muy extendidos en la actualidad en todo tipo de entornos. Se utiliza la herramienta OpenMP, realizando una selección previa de las tareas a paralelizar, y presentando los correspondientes algoritmos desarrollados. Con esta aproximación se logra una mejora consistente de las prestaciones en la simulación hidráulica, con distintos tipos de redes, tanto con el método GGA como, especialmente, con el método de mallas.

7.2. Aportaciones originales de la tesis

A continuación señalamos las aportaciones originales de la tesis:

- En el contexto de simulación de redes mediante el método de mallas, se han desarrollado algoritmos eficientes para obtener un conjunto de mallas independientes que conduzca a un sistema lineal altamente disperso. Los algoritmos

existentes previamente para obtención de mallas independientes producen sistemas poco dispersos, o bien requieren un tiempo de cálculo muy elevado. Los presentados en la tesis muestran un muy buen balance entre tiempo de ejecución y dispersidad del sistema.

- Se han desarrollado formulaciones para llevar a cabo la modelización de líneas temporalmente cerradas y válvulas de control de caudal (VCQ) en el contexto del método de mallas. En concreto, se presentan dos formulaciones distintas y se muestra la conexión entre ambas. La ventaja frente a otras formulaciones existentes es que no se requiere cambiar el conjunto de mallas independientes cuando una línea cambia de estado, lo que proporciona mayor eficiencia.

- Se ha desarrollado una formulación para la modelización de válvulas reductoras/sostenedoras de presión (VRP/VSP) en el método de mallas, usando una aproximación no simétrica. En esta formulación, la matriz del sistema lineal correspondiente a cada iteración del método tiene una parte simétrica y otra no simétrica. La resolución del sistema se hace por bloques, lo que permite aprovechar la simetría de la parte correspondiente. A diferencia de otros métodos existentes, la formulación presentada evita redefinir el conjunto de mallas independientes, y permite aprovechar en parte la simetría, al mantener separada la parte simétrica de la que no lo es. Además, el método considera un sistema lineal más exacto que la formulación utilizada por software como Epanet, lo que puede reducir el número de iteraciones necesario para alcanzar la convergencia.

- Se ha desarrollado otra formulación para la modelización de VRS/VSP en el método de mallas, usando una aproximación simétrica. Esto ha supuesto trasladar al método de mallas la formulación usada por software como Epanet para la modelización de VRPs/VSPs. Como resultado, se obtiene un sistema lineal con una matriz simétrica y cuya solución (salvo errores de redondeo) produce una iteración equivalente a la obtenida mediante Epanet. Frente a otros métodos existentes, no sólo se evita redefinir el conjunto de mallas, sino que se trabaja con una matriz simétrica, lo que aumenta la eficiencia.

- Se han desarrollado algoritmos paralelos sobre memoria distribuida para la simulación hidráulica mediante el método GGA. Los algoritmos, que utilizan MPI, han supuesto la paralelización de las distintas tareas de la simulación, incluyendo la resolución del sistema lineal por medio de métodos multifrontales.

- Se ha desarrollado un algoritmo paralelo sobre memoria distribuida para la simulación de la calidad del agua, mediante el método DVEM. El algoritmo está basado en la partición de la red entre los procesos por medio del algoritmo de *bisección recursiva multinivel*.

- Se ha desarrollado un algoritmo paralelo para la minimización de fugas de una red mediante la obtención de las consignas óptimas de una serie de VRP. La minimización de los diferentes instantes de tiempo se hace en paralelo, y se

utiliza el método de *programación secuencial cuadrática* (SQP) para resolver el problema correspondiente a cada instante de tiempo.

- Se han desarrollado algoritmos paralelos sobre memoria compartida para la simulación hidráulica mediante el método GGA, utilizando para ello OpenMP. Otras aproximaciones existentes consideran la computación de altas prestaciones en la simulación hidráulica por medio de GPUs o instrucciones vectoriales. Frente a ellas, la aproximación presentada en esta tesis consigue un mejor *speedup*.

- Se han desarrollado algoritmos paralelos sobre memoria compartida para la simulación hidráulica mediante el método de mallas, utilizando OpenMP. Con esto se consigue un mayor speedup en la simulación hidráulica, al combinar las ventajas del método de mallas con las de la paralelización.

7.3. Publicaciones y proyectos

Publicaciones

El trabajo de esta tesis ha dado lugar a tres publicaciones en revistas indexadas en JCR (*Journal Citation Reports*):

- **F. Alvarruiz**, F. Martínez Alzamora, A. M. Vidal. Improving the efficiency of the loop method for the simulation of water distribution systems. *Journal of Water Resources Planning and Management*, Vol 141, num 10, 2015. doi:10.1061/(ASCE)WR.1943-5452.0000539.

- José M Alonso, **F. Alvarruiz**, D. Guerrero, V. Hernández, P. A. Ruiz, A. M. Vidal, F. Martínez, J. Vercher, B. Ulanicki. Parallel computing in water network analysis and leakage minimization. *Journal of Water Resources Planning and Management*, Vol 126, num 4, pp. 251–260, 2000. doi:10.1061/(ASCE)0733-9496(2000)126:4(251).

- **F. Alvarruiz**, F. Martínez Alzamora, A. M. Vidal. Improving the performance of water distribution systems simulation on multicore systems. De próxima aparición en la revista *The Journal of Supercomputing*.

Otra publicación en una revista JCR está en camino:

- **F. Alvarruiz**, F. Martínez Alzamora, A. M. Vidal. Simulation of water distribution systems with the loop method, considering valves and pressure-dependent demands. Actualmente en fase de preparación para ser enviado a *Journal of Water Resources Planning and Management*

Se han realizado también distintas publicaciones en congresos internacionales:

- **F. Alvarruiz**, F. Martínez-Alzamora, A. M. Vidal. Efficient simulation of water distribution systems using OpenMP. En *15th International Conference on Mathematical Methods in Science and Engineering*, págs. 125–129, 2015.
- J. M. Alonso, **F. Alvarruiz**, D. Guerrero, V. Hernández, P. Ruiz, A. M. Vidal. A parallel algorithm for the simulation of water quality in water supply networks. En *4th International Meeting on Vector and Parallel Processing*, 2000.
- V. Hernández, A. M. Vidal, **F. Alvarruiz**, J. M. Alonso, D. Guerrero, P. A. Ruiz, F. Martinez, J. Vercher, and B. Ulanicki. Parallel computing in water network analysis and optimization processes. En *29th Annual Water Resources Planning and Management Conference*. American Society of Civil Engineers, 1999.
- V. Hernández, A. M. Vidal, **F. Alvarruiz**, J. M. Alonso, D. Guerrero, P. Ruiz. HIPERWATER: A high performance computing demonstrator for water network analysis. En *Parallel Computing: Fundamentals and Applications (PARCO 99)*, págs. 144–151. Imperial College Press, 1999.
- V. Hernández, F. Martínez, A. M. Vidal, J. M. Alonso, **F. Alvarruiz**, D. Guerrero, P. Ruiz, J. Vercher. HIPERWATER: A high performance computing Epanet-based demonstrator for water network simulation and leakage minimisation. En *4th International Conference on Computing and Control for the Water Industry (CCWI'99)*. Research Studies Press, 1999.

Proyectos de investigación

El trabajo de esta tesis se enmarca en la participación en distintos proyectos de investigación:

- "Computación y comunicaciones de altas prestaciones y aplicaciones en ingeniería". PROMETEO FASE II 2014/003. *Generalitat Valenciana*.
- "Introducing Low-Cost HPCN Techniques in Water Network Analisis and Optimization Processes" (HIPERWATER). Proyecto europeo ESPRIT IV PST 26424. 1997-1999.
- "HIPERCOSME Technology Transfer Node". Proyecto europeo ESPRIT IV PST 24003. 1997-2000.
- "Migrable Elastic Virtual Clusters on Hybrid Cloud Infrastructures". Ministerio de Economa y Competitividad (TIN2013-44390-R). 2014.

7.4. Trabajo futuro

Como fruto de los desarrollos de esta tesis, se ha producido código que se pretende liberar de manera que pueda ser utilizado para la simulación de redes de distribución

de agua, tanto por profesionales del sector como por investigadores. En particular, se considera interesante la liberación de dos productos *software*:

- Una versión de Epanet usando OpenMP para la simulación hidráulica. Esto incluiría tanto un programa como tal (*stand-alone*) como una librería con la funcionalidad ofrecida por el *toolkit* de Epanet.
- Un *software* de simulación de redes basado en Epanet, con funcionalidad similar a éste, pero con utilización del método de mallas para la simulación hidráulica, además de utilizar también OpenMP en dicha simulación. De nuevo esto incluiría tanto un programa como una librería.

En la actualidad es frecuente que la simulación de redes de distribución de agua considere la existencia de demandas dependientes de la presión. Epanet lo hace, por ejemplo, por medio de elementos del modelo llamados emisores (*emitters*), que van asociados a nudos de consumo, sobre los cuales aplican una demanda que depende de la presión. La consideración de este tipo de demandas en el método de mallas es por tanto un desarrollo interesante a realizar en un futuro próximo.

Otra de las tareas interesantes a realizar en el futuro es la paralelización sobre máquinas de memoria compartida de la simulación de la calidad del agua, considerando el método *lagrangiano conducido por tiempo* (ver apartado 2.9), el cual se implementa en la versión actual de Epanet.

Por último, la consideración de problemas de optimización abre un campo más amplio y de gran interés, con aplicaciones muy diversas, donde el trabajo realizado en la tesis resulta valioso, tanto por la mejora de rendimiento conseguida en las tareas de simulación, como por el conocimiento adquirido sobre las formulaciones de los distintos métodos.

Bibliografía

[1] E. Abraham and I. Stoianov. Efficient preconditioned iterative methods for hydraulic simulation of large scale water distribution networks. *Procedia Engineering*, 119:623 – 632, 2015. Computing and Control for the Water Industry (CCWI2015) Sharing the best practice in water management.

[2] E. Abraham and I. Stoianov. Sparse null space algorithms for hydraulic analysis of large-scale water supply networks. *Journal of Hydraulic Engineering*, 2015.

[3] F. Almeida, D. Giménez, J. M. Mantas, and A. M. Vidal. *Introducción a la Programación Paralela*. Paraninfo, 2008.

[4] J. M. Alonso, F. Alvarruiz, D. Guerrero, V. Hernández, P. A. Ruiz, A. M. Vidal, F. Martínez, J. Vercher, and B. Ulanicki. Parallel computing in water network analysis and leakage minimization. *Journal of Water Resources Planning and Management*, 126(4):251–260, 2000.

[5] J. M. Alonso, F. Alvarruiz, D. Guerrero, V. Hernández, P. Ruiz, and A. M. Vidal. A parallel algorithm for the simulation of water quality in water supply networks. In *4th International Meeting on Vector and Parallel Processing*, 2000.

[6] F. Alvarruiz, F. M. Alzamora, and A. M. Vidal. Improving the performance of water distribution systems simulation on multicore systems. *The Journal of Supercomputing*. Próxima aparición.

[7] F. Alvarruiz, F. Martínez-Alzamora, and A. M. Vidal. Efficient simulation of water distribution systems using OpenMP. In *15th International Conference on Mathematical Methods in Science and Engineering*, pages 125–129, 2015.

[8] F. Alvarruiz, F. Martínez-Alzamora, and A. M. Vidal. Improving the efficiency of the loop method for the simulation of water distribution systems. *Journal of Water Resources Planning and Management*, 141(10):04015019, 2015.

[9] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

[10] C. Arsene, D. Al-Dabass, and J. Hartley. A study on modeling and simulation of water distribution systems based on loop corrective flows and containing controlling hydraulics elements. In *Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on*, pages 423–430, Feb 2012.

[11] P. F. Boulos, T. Altman, P. A. Jarrige, and F. Collevati. Discrete simuation approach for network water quality models. *J. Water Resources Planning and Management*, 121(1):49–60, 1995.

[12] T. Bui and C. Jones. A heuristic for reducing fill in sparse matrix factorization. In *Proc. of the 6th SIAM Conf. Parallel Processing for Scientific Computing*, pages 445–452, 1993.

[13] G. Burger, R. Sitzenfrei, M. Kleidorfer, and W. Rauch. Quest for a new solver for EPANET 2. *Journal of Water Resources Planning and Management*, 2015.

[14] B. Chapman, G. Jost, and R. Van der Pas. *Using OpenMP : portable shared memory parallel programming*. MIT Press, Cambridge, Mass., London, 2008.

[15] M. H. Chaudhry and M. R. Islam. Water quality modeling in pipe networks. In *Proc. Conf. on Improving Efficiency and Reliability in Water Distribution Systems*. UIMP, 1994.

[16] E. Creaco and M. Franchini. Comparison of Newton-Raphson global and loop algorithms for water distribution network resolution. *Journal of Hydraulic Engineering*, 140(3):313–321, 2014.

[17] E. Creaco and M. Franchini. The identification of loops in water distribution networks. *Procedia Engineering*, 119:506 – 515, 2015. Computing and Control for the Water Industry (CCWI2015) Sharing the best practice in water management.

[18] H. Cross. Analysis of flow in networks of conduits or conductors. *University of Illinois Engineering Experimental Station, Bulletin No. 286*, 1936.

[19] P. A. Crous, J. E. van Zyl, and Y. Roodt. The potential of graphical processing units to solve hydraulic network equations. *Journal of Hydroinformatics*, 14:603–612, 2012.

[20] J. De Pina. *Applications of shortest path methods*. PhD thesis, University of Amsterdam, Netherlands, 1995.

[21] J. Deuerlein, R. Cembrowicz, and S. Dempe. Hydraulic simulation of water supply networks under control. In *World Water and Environmental Resources Congress 2005*, 2005.

[22] J. Deuerlein, A. Simpson, and S. Dempe. Modeling the behavior of flow regulating devices in water distribution systems using constrained nonlinear programming. *Journal of Hydraulic Engineering*, 135(11):970–982, 2009.

[23] J. Deuerlein, A. Simpson, and E. Gross. The never ending story of modeling control-devices in hydraulic systems analysis. In *Water Distribution Systems Analysis 2008*, 2009.

[24] S. Elhay, A. Simpson, J. Deuerlein, B. Alexander, and W. Schilders. Reformulated co-tree flows method competitive with the global gradient algorithm for solving water distribution system equations. *Journal of Water Resources Planning and Management*, 140(12), 2014.

[25] R. Epp and A. G. Fowler. Efficient code for steady-state flows in networks. *Journal of the Hydraulics Division*, 96(1):43–56, 1970.

[26] A. George and J. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31(1):1–19, 1989.

[27] A. George and J. W. Liu. *Computer Solution of Large Sparse Positive Definite*. Prentice Hall Professional Technical Reference, 1981.

[28] G. A. Germanopoulos. A technical note on the inclusion of pressure dependent demand and leakage in water supply networks. *Civ. Engng. Syst.*, 2:171–179, 1985.

[29] O. Giustolisi, D. A. Savic, D. Laucelli, and L. Berardi. Testing linear solvers for WDN models. In *Computing and Control for the Water Industry: CCWI 2011*, 2011.

[30] S. J. Goodwin. The results of the experimental program on leakage and leakage control. Technical Report TR 145, Water Research Centre, 1980.

[31] A. Grama, A. Gupta, G. Karypis, and V. Kumar. *Introduction to Parallel Computing*. Pearson Education Limited, Harlow, Essex, UK, 2nd edition, 2003.

[32] M. Guidolin, P. Burovskiy, Z. Kapelan, and D. Savić. CWSNet: An object-oriented toolkit for water distribution system simulations. In *Proc. 12th Water Distribution System Analysis Symp., ASCE, Reston, VA*, 2010.

[33] M. Guidolin, Z. Kapelan, and D. Savic. Using high performance techniques to accelerate demand-driven hydraulic solvers. *Journal of Hydroinformatics*, 15(1):38–54, 2013.

[34] M. Guidolin, Z. Kapelan, D. Savic, and O. Giustolisi. High performance hydraulic simulations with epanet on graphics processing units. In *Proc. 9th Int. Conf. on Hydroinformatics*, 2010.

[35] A. Gupta, F. G. Gustavson, M. V. Joshi, G. Karypis, and V. Kumar. Design and implementation of a scalable parallel direct solver for sparse symmetric positive definite systems: Preliminary results. In *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, PPSC 1997, March 14-17, 1997, Hyatt Regency Minneapolis on Nicollet Mall Hotel, Minneapolis, Minnesota, USA*, 1997.

[36] A. Gupta, G. Karypis, and V. Kumar. Highly scalable parallel algorithms for sparse matrix factorization. *IEEE Trans. Parallel Distrib. Syst.*, 8(5):502–520, May 1997.

[37] M. T. Heath, E. G. Ng, and B. W. Peyton. Parallel algorithms for sparse linear systems. *SIAM Review*, 33(3):420–460, 1991.

[38] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. Technical Report SAND93-1301, Sandia National Laboratories, Albuquerque, NM, 1993.

[39] R. W. Jeppson. *Analysis of flow in pipe networks.* Ann Arbor Science, 1976.

[40] M. Joshi, G. Karypis, V. Kumar, A. Gupta, and F. Gustavson. PSPASES: An efficient and scalable parallel sparse direct solver. In *Proc. of the ninth SIAM Conf. on Parallel Processing for Scientific Computing*, 1999.

[41] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. on Scientific Computation*, 20(1):359–392, 1998.

[42] G. Karypis and V. Kumar. *METIS. A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 5.1.0.* University of Minnesota, Dept. of Computer Science & Engineering, 2013. `http://glaros.dtc.umn.edu/gkhome/metis/metis/download`.

[43] T. Kavitha, C. Liebchen, K. Mehlhorn, D. Michail, R. Rizzi, T. Ueckerdt, and K. A. Zweig. Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, 3(4):199–243, 2009.

[44] T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. A faster algorithm for minimum cycle basis of graphs. In *Automata, Languages and Programming*, volume 3142 of *Lecture Notes in Computer Science*, pages 846–857. Springer Berlin Heidelberg, 2004.

[45] C. T. Lawrence, J. L. Zhou, and A. L. Tits. User's guide for CFSQP version 2.5: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical Report TR-94-16r1, Institute for Systems Research, University of Maryland, 1997.

[46] C. P. Liou and J. R. Kroon. Modeling the propagation of waterborne substances in distribution networks. *J. AWWA*, 79(11):54–58, 1987.

[47] J. W. H. Liu. The multifrontal method for sparse matrix solution: Theory and practice. *SIAM Review*, 34(1):82–109, 1992.

[48] M. López-Ibáñez, D. Prasad, and B. Paechter. Parallel optimisation of pump schedules with a thread-safe variant of epanet toolkit. In *Water Distribution Systems Analysis*, 2010.

[49] M. Mair, R. Sitzenfrei, M. Kleidorfer, and W. Rauch. Performance improvement with parallel numerical model simulations in the field of urban water management. *Journal of Hydroinformatics*, 16(2):477–486, 2014.

[50] D. W. Martin and G. Peters. The application of Newton's method to network analysis by digital computer. *Journal Institution of Water Engineers and Scientists*, 17(17):115–129, 1963.

[51] F. Martínez, P. Conejos, and J. Vercher. Development of an integrated model for water distribution systems considering both distributed leakage and pressure-dependent demands. In *26th Ann. Water Resources Planning and Management Conf.* ASCE, 1999.

[52] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard. Version 3.1.* University of Tennessee, Knoxville, TN, USA, 2015. `http://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf`.

[53] M. Morley, C. Tricarico, Z. Kapelan, D. Savić, and G. D. Marinis. deEPANET: A distributed hydraulic solver architecture for accelerating optimization applications working with conditions of uncertainty. In *Proceedings 7th International Conference on Hydroinformatics*, pages 2465–2472, 2006.

[54] A. Ostfeld, J. Uber, E. Salomons, J. Berry, W. Hart, C. Phillips, J. Watson, G. Dorini, P. Jonkergouw, Z. Kapelan, F. di Pierro, S. Khu, D. Savic, D. Eliades, M. Polycarpou, S. Ghimire, B. Barkdoll, R. Gueli, J. Huang, E. McBean, W. James, A. Krause, J. Leskovec, S. Isovitsch, J. Xu, C. Guestrin, J. VanBriesen, M. Small, P. Fischbeck, A. Preis, M. Propato, O. Piller, G. Trachtman, Z. Wu, and T. Walski. The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, 134(6):556–568, 2008.

[55] O. Piller and J. van Zyl. Modeling control valves in water distribution systems using a continuous state formulation. *Journal of Hydraulic Engineering*, 140(11), 2014.

[56] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, 1957.

[57] E. Roshani and Y. Filion. Using parallel computing to increase the speed of water distribution network optimization. In *14th Water Distribution Systems Analysis Conference (WDSA 2012)*, pages 24–27. Barton, A.C.T.: Engineers Australia, 2012.

[58] A. L. Rossman. *Epanet 2 Users manual.* Water Supply and Water Resources Division, US Environment Protection Agency, 2000.

[59] L. A. Rossman. Computer models/EPANET. In L. W. Mays, editor, *Water Distribution Systems Handbook*, chapter 12. McGraw-Hill Companies, 1999.

[60] L. A. Rossman and P. F. Boulos. Numerical methods for modeling water quality in distribution systems: a comparison. *Journal of Water Resources Planning and Management*, 122(2):137–146, 1996.

[61] L. A. Rossman, P. F. Boulos, and T. Altman. Discrete volume-element method for network water-quality models. *Journal of Water Resources Planning and Management*, 119(5):505–517, 1993.

[62] R. O. Salgado. *Computer Modelling of Water Supply Distribution Networks using the Gradient Method*. PhD thesis, Newcastle University, 1988.

[63] U. Shamir and C. D. D. Howard. Water distribution systems analysis. In *Proc ASCE*, volume 94, No. HY1, pages 237–260, 1968.

[64] A. Simpson. Modeling of pressure regulating devices: The last major problem to be solved in hydraulic simulation. In *WRPMD'99*, 1999.

[65] G. D. Smith. *Numerical solution of partial differential equations: Finite difference methods*. Oxford University Press, second edition, 1978.

[66] E. Todini. Un metodo del gradiente per la verifica delle reti idrauliche. *Bolletino degli Ingegneri della Toscana*, 11:11–14, 1979.

[67] E. Todini and S. Pilati. A gradient algorithm for the analysis of pipe networks. In *Proc. Int. Conf. Computer Applications for Water Supply Distribution*, Leicester Polytechnic, UK, 1987.

[68] E. Todini and S. Pilati. A gradient algorithm for the analysis of pipe networks. In B. Coulbeck and C.-H. Orr, editors, *Computer Applications in Water Supply: Vol. 1—Systems Analysis and Simulation*, pages 1–20. Research Studies Press Ltd., 1988.

[69] E. Todini and L. A. Rossman. Unified framework for deriving simultaneous equation algorithms for water distribution networks. *Journal of Hydraulic Engineering*, 139(5):511–526, 2013.

[70] K. Travers. The mesh method in gas network analysis. *Gas Jour.*, 332:167–174, 1967.

[71] D. J. Wood. Algorithms for pipe network analysis and their reliability. Technical Report 127, Water Resources Research Institute, University of Kentucky, 1981.

[72] D. J. Wood and C. O. A. Charles. Hydraulic network analysis using linear theory. *Journal of the Hydraulics Division, ASCE*, 98(HY7):1157–1170, 1972.

[73] Z. Y. Wu and I. Lee. Lesson for parallelizing linear equation solvers and water distribution analysis. In *Eleventh International Conference on Computing and Control for the Water Industry: CCWI 2011*, 2011.