Document downloaded from:

http://hdl.handle.net/10251/61056

This paper must be cited as:

Alvarruiz Bermejo, F.; Martínez Alzamora, F.; Vidal Maciá, AM. (2015). Improving the efficiency of the loop method for the simulation of water distribution networks. Journal of Water Resources Planning and Management. 141(10):1-10. doi:10.1061/(ASCE)WR.1943-5452.0000539.

The final publication is available at

http://dx.doi.org/10.1061/(ASCE)WR.1943-5452.0000539

Copyright American Society of Civil Engineers

Additional Information

Improving the Efficiency of the Loop Method for the Simulation of Water Distribution Systems

F. Alvarruiz [1], F. Martínez-Alzamora [2], A. M. Vidal [3]

ABSTRACT

Efficiency of hydraulic solvers for the simulation of flows and pressures in water distribution systems (WDS) is very important, especially in the context of optimization and risk analysis problems, where the hydraulic simulation has to be repeated many times. Among the methods used for hydraulic solvers, the most prominent nowadays is the global gradient algorithm (GGA), based on a hybrid node-loop formulation and used by the software package Epanet. Earlier, another method based just on loop flow equations was proposed, which presents the advantage that it leads to a system matrix which is in most cases much smaller than in the GGA method, but has also some disadvantages, mainly a less sparse system matrix, and the fact that introducing some types of valves requires the redefinition of the set of network loops initially defined.

The contribution of this paper is to present solutions for overcoming the mentioned disadvantages of the method based on loop flow equations. In particular, efficient procedures are shown for selecting the network loops so as to achieve a highly sparse matrix, and methods are presented to incorporate check valves and automatic control valves, while avoiding the need to redefine the loops initially selected.

[1]Dept. Sistemas Informáticos y Computación, Universitat Politècnica de València. 46022. Valencia (Spain)

[2]Ph.D. Research Institute of Water and Environmental Engineering (IIAMA). Universitat Politècnica de Valencia. 46022. Valencia (Spain)

[3]Ph.D. Dept. Sistemas Informáticos y Computación, Universitat Politècnica de València. 46022. Valencia (Spain)

INTRODUCTION

Hydraulic solvers for the simulation of flows and pressures in water distribution systems (WDS) are used extensively to solve a large number of problems. Among them, optimization (e.g. design or model calibration), and risk analysis problems, usually require the simulation to be repeated many times, with variations in the input data, before a solution is reached. In those contexts, the computational performance of the hydraulic solver is of the utmost importance (Guidolin et al. 2013).

Since the 1960s, a number of different methods for hydraulic solvers have been proposed, among which we should mention the method presented in (Martin and Peters 1963) as the first one to use a Newton-Raphson approach, applying it to a formulation based on nodal equations, with heads as unknowns. Later, (Epp and Fowler 1970) proposed a method using a formulation based on loop equations, with loop corrective flows as unknowns. Another method proposed, known as the *global gradient algorithm* (GGA) (Todini and Pilati 1988), which solves simultaneously for pipe flows and nodal heads, was adopted by Epanet, a public domain WDS modeling software package developed by the *US Environmental Protection Agency* (EPA) (Rossman 1999), (Rossman 2000). GGA is probably the most popular method used for the simulation of WDS, and Epanet is still considered nowadays a reference software package in this field.

Different papers have compared the GGA method and the loop method of (Epp and Fowler 1970). From the point of view of convergence, (Todini and Pilati 1988) showed that both of them are equivalent. As they put it, one can project the results obtained in the problem space of GGA (pipe flows and nodal heads) into the problem space of the loop method (loop corrective flows) by simple linear algebra manipulations. Thus, the sequence of iterations of both methods is the same, if they start from the same initial values. This is also pointed out in (Todini 2008) and (Elhay et al. 2014). Therefore, as (Todini 2008) states, when comparing the computing time required by both methods, the key issues are the dimension of the space on which the problem is solved and the symmetry and the sparsity of the resulting system matrix.

In this context, the loop method presents the advantage that the size of the linear systems to be solved is considerably smaller. As disadvantages, the matrix of those linear systems is generally less sparse, and the introduction of valves and closed pipes presents difficulties, because it changes the set of loops over which the method is to be applied.

Recently, there have been different publications considering the loop method. (Creaco and Franchini 2014) propose an automatic procedure to find the basis of "minimum loops", producing a matrix with maximum sparsity for the linear systems. The main drawback of that method is its excessive computation time, which is reported to be up to 3 hours for a network of 5,100 pipes. (Arsene et al. 2012) consider the need to redefine the loops when there is a status change in a controlling element such as a valve or a pump. They propose a partial redefinition of the loop set by modifying a spanning tree that is the base for the loop definition. (Elhay et al. 2014) present a reformulated co-tree flows method (RCTM) , which is similar to the loop method and also produces the same sequence of iterations. They provide results on a number of case study networks, where their method is reported to be between 15% to 82% faster than GGA.

This paper presents some novel contributions in the context of the loop method for analysis of WDS, with the aim of improving its computational efficiency. These contributions are: (i) a fast method for selecting the network loops, that achieves a highly sparse matrix, and (ii) treatment of check valves and automatic control valves in a way that avoids the need to redefine the loops initially selected.

In the next section, we provide the necessary background on the loop method for the simulation of WDS. Then, we consider the choice of a set of independent loops, presenting two novel methods. The next three sections describe the approach for modeling control valves, considering the cases of flow regulating and pressure regulating devices. After considering the choice of an initial flow vector for the loop method, we present results for the proposed methods. Finally, conclusions and future work are presented.

THE LOOP METHOD FOR WDS SIMULATION

The loop method was formulated by Epp and Fowler in (Epp and Fowler 1970). The method considers the set of energy-conservation equations, that state that the sum of energy losses around any network loop must be zero. In particular, if a set of l independent loops or cycles is found for a network of p pipes, the following equations hold:

$$\sum_{j=1}^{p} \delta_{ij} h_j(q_j) = 0, \qquad i = 1, 2 \ldots l \tag{1}$$

where the notation δ_{ij} is used to express which pipes form each loop: δ_{ij} is 0 if the pipe j is not included in loop i, and ± 1 otherwise, the sign accounting for the two possible orientations of the pipe in the loop. $h_j(q_j)$ is the energy loss in the pipe j due to friction, expressed as a function of the flow q_j. There are different formulas that can be used to compute that loss, e.g. the Hazen-Williams formula, which is (using international system units):

$$h_j(q_j) = \frac{10.674\ L}{C^{1.852} D^{4.871}} q_j |q_j|^{0.852} = R\ q_j |q_j|^{0.852} \tag{2}$$

where C is a roughness coefficient, and D and L are the pipe diameter and length, respectively. Other formulas are used for hydraulic elements such as pumps or throttle control valves.

Additionally, the flows q_j must satisfy the mass conservation equations, stating that the sum of all flows entering/leaving any junction must be zero, i.e., for a network of n junctions:

$$\sum_{j=1}^{p} \gamma_{ij} q_j - c_i = 0, \qquad i = 1, 2 \ldots n \tag{3}$$

where c_i is the flow consumed in the junction i, and γ_{ij} is $+1$ (-1) if the pipe j ends (starts) at node i, and 0 otherwise.

Equations (1) and (3) are a set of $l + n$ equations in p unknowns (the flows q_j). If the network has only one tank/reservoir, the number of independent loops that can be formed is $l = p - n$, as in the network of figure 1, with 3 junctions (N1-N3), one tank (N4), 5 pipes (P1-P5) and 2 loops (L1-L2). In the general case of n_t tanks, where $n_t \geq 1$, $n_t - 1$ fictitious loops are formed connecting the tanks, and again $l = p - n$ (see figure 5). Thus, the system of equations given by (1) and (3) always has p equations and unknowns.

However, the system can be reduced to l equations if we take into account that, given an initial vector of flows q^0 satisfying equation (3), any other vector that satisfies the same equation can be obtained by considering a flow correction $\hat{q}_k$ for each independent loop k, and adding the correction to the initial flow of the pipes forming the loop, i.e.:

$$q_j = q_j^0 + \sum_{k=1}^{l} \delta_{kj} \hat{q}_k \qquad j = 1, 2 \ldots p \tag{4}$$

Thus, equations (1) and (3) can be combined in the following way:

$$\sum_{j=1}^{p} \delta_{ij} \, h_j(q_j^0 + \sum_{k=1}^{l} \delta_{kj} \hat{q}_k) = 0, \qquad i = 1, 2 \ldots l \tag{5}$$

which is a non-linear system of l equations in l unknowns (the loop flow corrections $\hat{q}_k$). The system is then solved by means of the Newton-Raphson method, which leads to a sequence of linear systems of the form:

$$\sum_{j=1}^{p} \left(\delta_{ij} h_j^k + \delta_{ij} d_j \sum_{k=1}^{l} \delta_{kj} \Delta\hat{q}_k \right) = 0, \qquad i = 1, 2 \ldots l \tag{6}$$

where h_j^k is the head loss across pipe j at the current iteration k, d_j is the derivative of $h_j(q_j)$ at the same iteration, and $\Delta\hat{q}_k$ is the increment of the flow correction for loop k. According to (2):

$$d_j = 1.852 \, R \, |q_j|^{0.852} \tag{7}$$

As an example, let us consider the network of figure 1, with the two loops denoted by L1 and L2. The nonlinear system of equations is:

$$\begin{aligned} h_1(q_1^0+\hat{q}_1)+h_4(q_4^0+\hat{q}_1)-h_2(q_2^0-\hat{q}_1+\hat{q}_2) &= 0 \\ h_2(q_2^0-\hat{q}_1+\hat{q}_2)-h_5(q_5^0-\hat{q}_2)-h_3(q_3^0-\hat{q}_2) &= 0 \end{aligned} \tag{8}$$

The linear equations corresponding to the Newton-Raphson method are:

$$\begin{aligned} h_1^k+d_1\Delta\hat{q}_1+h_4^k+d_4\Delta\hat{q}_1-h_2^k+d_2(\Delta\hat{q}_1-\Delta\hat{q}_2) &= 0 \\ h_2^k+d_2(-\Delta\hat{q}_1+\Delta\hat{q}_2)-h_5^k+d_5\Delta\hat{q}_2-h_3^k+d_3\Delta\hat{q}_2 &= 0 \end{aligned} \tag{9}$$

or, written in matrix form:

$$\begin{pmatrix} d_1+d_2+d_4 & -d_2 \\ -d_2 & d_2+d_3+d_5 \end{pmatrix} \begin{pmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \end{pmatrix} = -\begin{pmatrix} h_1^k+h_4^k-h_2^k \\ h_2^k-h_5^k-h_3^k \end{pmatrix} \tag{10}$$

where the system matrix is symmetric positive definite.

An advantage of the loop method for water distribution systems is that it works with a matrix of size $l \times l$, which is in most cases much smaller than the matrix for the gradient method, which is $n \times n$. However, this does not necessarily mean that the linear system can be solved faster, since it will largely depend on the number of nonzero coefficients in the matrix, which is strongly related to the way the network loops are defined, as explained next.

CHOOSING THE LOOPS

As we can see in the example above, each diagonal element of the matrix is the sum of d_i for the pipes i in a loop. Likewise, an off-diagonal coefficient is the sum of $\pm d_i$ for those pipes i which are common to two different loops ($-d_2$ in the example).

It is important to note that the choice of the set of independent loops for a network can greatly influence the amount of non-zero off-diagonal elements in our matrix, and thus the efficiency of the method. In particular, it is desirable to choose loops that are short and with minimum overlapping among them.

Let us consider, for instance, the network in figure 2, with $n = 13$ junctions, $p = 20$ pipes, and a number of independent loops of $p - n = 7$. If the loops were selected as shown in the figure, the sparsity pattern of the resulting matrix would be as presented in figure 3 (only the upper triangular part is shown, since the matrix is symmetric). Other possible choices of loops may result in a completely dense matrix, as we will shortly see.

A commonly used method for selecting the set of independent loops starts by obtaining a spanning tree of the network (Travers 1967), (Arsene et al. 2012). Once it has been formed, adding any other pipe to the tree results in a loop, which is known as a *fundamental cycle* or fundamental loop. To obtain the loop, we go from each end of the added pipe following the tree towards the root, until the two paths join. As an example, solid lines in figure 4 correspond to a spanning tree. If pipe 5-8 is added to the tree, the loop formed is given by the pipe 5-8 itself, together with the paths 5-2-0-13 and 8-6-4-1-13. The set of fundamental loops for the spanning tree constitute a set of independent loops. This method will be referred to as **m1** in this paper.

Although this method is simple, it presents the disadvantage that it generally produces a matrix that is not very sparse. In our example, each loop resulting from the spanning tree has at least one pipe in common with every other loop, therefore the matrix produced is completely dense. While other spanning trees can be found that are more favorable, no spanning tree can produce the set of loops presented in figure 2.

From a graph theory perspective, (Kavitha et al. 2004) and (Kavitha et al. 2009) study the problem of finding a *Minimum Cycle Basis* (MCB) of a graph. In those papers, the set of all possible cycles in a graph is seen as a vector space, and a cycle basis is defined as a set of cycles forming a basis of that vector space. A minimum cycle basis of a weighted graph is then defined as a cycle basis such that the sum over all cycles of the edge weights is minimum. Our case corresponds to an unweighted graph, where a minimum cycle basis is one in which the sum of the number of edges of each cycle is minimum.

In the case of the loops shown in figure 2, each loop has 4 edges (pipes), so the sum is 28.

If we take the fundamental cycles resulting from figure 4, the sum will be 56. Pipes that are common to two or more loops will be counted more than once, which implies that a minimum cycle basis will have little overlap between the loops, and will consequently produce a fairly sparse matrix.

Taking into account the good properties of MCB, we have implemented a simplified version of the algorithm presented in (Kavitha et al. 2004) for its computation. Although the results are very good in terms of matrix sparsity, as presented in section 8, the problem encountered is the high computational cost of the algorithm, both in terms of execution time and memory. This method is referred to in this paper as **m2**. A similar approach was taken in (Creaco and Franchini 2014), using an algorithm based on (De Pina 1995), and the same problem of high computational cost is reported.

Trying to overcome the problems of the two mentioned methods, this paper presents two different approaches for the definition of the set of independent loops.

The first method proposed (which will be called **m3**) starts by constructing a spanning tree and obtaining the fundamental loops as in **m1**. Then the loops are simplified by combining them. When we combine two loops, the result is a loop that contains the pipes which are in either one of the two original loops, but not in both of them. For instance, in figure 4, the loop 5-2-0-13-1-3-5 could be simplified by combining it with the loop 0-13-1-3-0, resulting in the loop 5-2-0-3-5. Before combining the loops, however, they are sorted according to its depth in the tree, from less depth to more depth.

The simplification process is described in algorithm 1, in which each loop l_i is tried to be reduced by combining it only with the previous loops ($l_1 \dots l_{i-1}$). Note that the order in which to consider the loops $l_1 \dots l_{i-1}$ for their possible combination with the loop l_i is important. Here, algorithm 1 follows a greedy approach, in which the first candidates considered are those that would produce a shorter loop if combined with l_i. In particular, a list P of those loops, among $l_1 \dots l_{i-1}$, that overlap with the loop l_i is built, and it is sorted ascendingly by the length of the loop resulting from the combination with l_i. Then each of the loops in the

Algorithm 1 Loop simplification process for method m3

```
Input: $L$, list of loops
Output: $L'$, list of simplified loops
  $L' \leftarrow \emptyset$
  for all loop $l_i$ in $L$ do
    $P \leftarrow \emptyset$
    for all loop $l_j$ in $L$, where $j < i$ do
      if $l_j$ overlaps with $l_i$ then
        $n_j \leftarrow$ length of the loop resulting from the combination of $l_i$ and $l_j$
        Insert $(l_j, n_j)$ in $P$, sorted ascendingly by $n_j$
      end if
    end for
    $l'_i \leftarrow l_i$
    for all pair $(l_j, n_j)$ in $P$ do
      $c \leftarrow$ loop resulting from the combination of $l'_i$ and $l_j$
      if length of $c <$ length of $l'_i$ then
        $l'_i \leftarrow c$
      end if
    end for
    Insert $l'_i$ in $L'$
  end for
```

list is combined with l_i, the result of a combination being discarded if it fails to reduce the length of the loop. Finally, the new reduced loop l'_i is inserted in the new set of loops L'.

The second method proposed (which will be called **m4**) is described in algorithm 2. Basically, it performs a breadth-first exploration of the network graph G, starting from a given node u. During this exploration, a graph G' is built containing the edges and nodes of the network that have already been visited. Whenever a new edge $(i, j) \notin G'$ is encountered that connects the current node i with a node j already visited, a new loop is added to the set of loops L. That new loop will consist of the edge (i, j) and the shortest path in G' between nodes i and j, where "shortest path" means a path with minimum number of pipes. Note that the edge (i, j) is then added to G', and thus can also be used for the following loops to be found.

In the literature, very often each loop in the independent set is identified by a corresponding *chord* pipe (i.e. a pipe that is not in an initially defined spanning tree), and the loop flow correction is equal to the flow through that pipe. Note however that this can only

Algorithm 2 Loop definition method m4

```
Input: $G$, network graph; $u$, initial node for exploration.
Output: $L$, list of loops
  $L \leftarrow \emptyset$
  $G' \leftarrow \emptyset$
  $S \leftarrow \{u\}$
  while $S \neq \emptyset$ do
    $i \leftarrow$ pop first element of $S$
    for all node $j$ where edge $(i, j) \in G$ do
      if $j \notin G'$ then
        Add node $j$ and edge $(i, j)$ to graph $G'$
        Insert $j$ as the last element of $S$
      else if edge $(i, j) \notin G'$ then
        $p \leftarrow$ shortest path from $i$ to $j$ in $G'$
        $c \leftarrow \{$edges of $p\} \cup \{(i, j)\}$
        Insert $c$ into $L$
        Add edge $(i, j)$ to $G'$
      end if
    end for
  end while
```

be done if the method **m1** is used, because the method imposes the constraint that the set of loops must be the set of fundamental loops of a spanning tree. Methods **m2-m4** do not impose that constraint, and as a result of that they can find a better set of loops, producing a more sparse matrix, as shown in the network of figure 2.

APPROACH FOR MODELING CONTROL VALVES

The next two sections deal with hydraulic elements that can change their status, such as check-valves, flow control valves (FCV), pressure reducing valves (PRV) and pressure sustaining valves (PSV). These elements can be in different status depending on hydraulic conditions which are not known a priori, presenting an important challenge for the simulation.

Epanet (Rossman 1999) uses a method in which the status of the valves is assumed at the beginning of the iterative process, checked between the iterations and if necessary adjusted by specific heuristics. There is no guarantee that this method will be able to find the correct valve status in all cases, see e.g. (Simpson 1999), but it works well in practice and is a widely accepted method in the hydraulic modeling community. There are other more rigorous approaches in which the problem is formulated as the minimization of the *content* or *co-content* functions subject to inequality constraints (Deuerlein et al. 2009a), (Deuerlein et al. 2005), (Deuerlein et al. 2009b), (Piller and van Zyl 2014). These methods overcome the difficulties found in a heuristic method, although they are more complex and can therefore require more computing time.

This paper assumes that a method similar the one implemented in Epanet is going to be used to determine the operational status of the valves in the network. Even in that context, the presence of control valves affects the formulation of the loop method given by (6). (Jeppson 1976) uses an approach to include PRV in which the set of independent loops changes depending on the status of the valves. Other authors, such as (Arsene et al. 2012), propose a partial redefinition of the loop set by modifying a spanning tree that is the base for the loop definition. The problem with these approaches is the need to redefine the loop set, which implies introducing changes in the sparsity structure of the system matrix. This is important because the linear systems arising in water distribution system analysis are normally solved by means of a direct method, and a symbolic decomposition is done at the beginning of the simulation, to determine the sparsity structure of the factorized matrix. If the structure of the matrix changes, the symbolic decomposition would have to be done again, or at least updated, resulting in increased computing time.

This paper presents a method to cope with control valves that avoids changing the set of independent loops when a valve changes its status.

MODELING FLOW REGULATING DEVICES

Check valves are used to ensure that the flow through a pipe is always in the desired direction, preventing reverse flow by closing the pipe. This can be a difficulty for the loop simulation method, because the topology of the network changes, which might require a redefinition of the set of independent loops.

For example, let us consider the network in figure 5, with 4 junctions (N1-N4), 2 tanks (N5-N6), 7 pipes (P1-P7), and the independent loops L1, L2 and L3. The system of linear equations to be solved in each iteration of the Newton-Raphson method, using the loop formulation, is:

$$\begin{pmatrix} d_1+d_2+d_5 & -d_2 & 0 \\ -d_2 & d_2+d_3+d_6 & -d_3 \\ 0 & -d_3 & d_3+d_4+d_7 \end{pmatrix} \begin{pmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \end{pmatrix} = - \begin{pmatrix} h_1^k - h_2^k + h_5^k \\ h_2^k - h_3^k - h_6^k \\ h_3^k - h_4^k - h_7^k + \hat{H}_6 - \hat{H}_5 \end{pmatrix} \tag{11}$$

where $\hat{H}_5$ and $\hat{H}_6$ are the head values at the tanks, which are assumed to be known, and the rest of the symbols are as defined in section 2.

Let us suppose that pipe 2 is equipped with a check valve and that the valve closes. This could be modelled by using a very high value for d_2, e.g. 10^8 (corresponding to a high resistance for the pipe), and solving the system of linear equations (11) normally. However, this approach introduces very large numbers in the matrix, causing the system of equations to be ill conditioned, which means that we should expect important round-off errors. Another approach is to eliminate the closed pipe and redefine the loop set accordingly. In this example network, loops 1 and 2 could be replaced by a single loop with pipes 1, 3, 5 and 6. This is done e.g. in (Arsene et al. 2012).

We propose another approach to cope with a closed check valve, which avoids the need to redefine the loops of the network. In particular, if check valve in pipe 2 closes, the difference in head between the two ends of the pipe is not related to the flow through it, since that flow is zero. It follows that we should not use d_2, but introduce h_2 as a new variable. We also introduce a new equation, which states that flow through pipe 2 is zero, i.e:

$$q_2^0 - \hat{q}_1 + \hat{q}_2 = 0 \tag{12}$$

Expressing this equation using the system unknowns, which are $\Delta\hat{q}$, we have:

$$q_2^0 - \hat{q}_1^k - \Delta\hat{q}_1 + \hat{q}_2^k + \Delta\hat{q}_2 = 0 \tag{13}$$

where $\hat{q}_i^k$ is the flow correction for loop i ($\hat{q}_i$), at the start of iteration k.

Taking that into account, the original system is transformed in the following way:

$$\begin{pmatrix} d_1+d_5 & 0 & 0 & -1 \\ 0 & d_3+d_6 & -d_3 & 1 \\ 0 & -d_3 & d_3+d_4+d_7 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \\ h_2 \end{pmatrix} = - \begin{pmatrix} h_1^k + h_5^k \\ -h_3^k - h_6^k \\ h_3^k - h_4^k - h_7^k + H_6 - H_5 \\ -q_2^0 + \hat{q}_1^k - \hat{q}_2^k \end{pmatrix} \tag{14}$$

We now generalize the methodology proposed. With any number of closed check valves, equation (14) presents the form:

$$\begin{pmatrix} A & C \\ C^T & 0 \end{pmatrix} \begin{pmatrix} \Delta\hat{q} \\ \hat{h} \end{pmatrix} = \begin{pmatrix} b \\ \hat{b} \end{pmatrix} \tag{15}$$

where A is the same matrix as that of the original system, only changing the value of the coefficients affected by the check valves (as if those valves had been replaced by pipes of zero resistance). In particular, no new non-zero elements are added to the matrix. C is an incidence or topological matrix, the elements of which can only have the values 0 and ±1; $\hat{h}$ is the vector of head losses for the closed check valves, and $\hat{b}$ are the elements added to the right-hand side of the equation.

This formulation was also derived in (Deuerlein et al. 2009a) following a different approach, in which the hydraulic steady-state simulation is treated as the minimization of the content function with inequality flow constraints, and the head losses of the check valves are interpreted as Lagrange multipliers. In this paper we use expression (15) without considering an optimization problem, which may be more cumbersome. Instead, we use a

Newton-Raphson iterative scheme together with a method to find out the correct status of the valves (such as the heuristic method of Epanet). Another contribution of this paper is to address the efficient solution of the system (15), which could be split in two equations:

$$A\Delta\hat{q} + C\hat{h} = b \tag{16}$$

$$C^T\Delta\hat{q} = \hat{b} \tag{17}$$

Then, using the Schur decomposition (Zhang 2005), we isolate $\Delta\hat{q}$ from (16) and substitute in (17) to get:

$$\Delta\hat{q} = A^{-1}b - A^{-1}C\hat{h} \tag{18}$$

$$\hat{h} = (C^TA^{-1}C)^{-1}(C^TA^{-1}b - \hat{b}) \tag{19}$$

Vector $\hat{h}$ can be obtained from (19), which implies: (i) solving the two linear systems $A^{-1}b$ and $A^{-1}C$, which share the coefficient matrix; (ii) multiplying the solution of those systems by the matrix C^T, and (iii) solving a linear system with the symmetric matrix $C^TA^{-1}C$. The dimension of the latter linear system is equal to the number of closed check-valves, which will usually be few. Finally, $\Delta\hat{q}$ is obtained from (18), taking advantage of the fact that $A^{-1}b$ and $A^{-1}C$ have already been solved as part of (19).

As a summary, the introduction of closed check-valves can be done without redefining the loops of the network and thus without changing the sparsity pattern of the system matrix.

Pipes can also be closed directly by means of control rules during the simulation process. The approach presented in this section is also valid for that case, which is in fact simpler, because the status of the pipe (open/closed) does not depend on the direction of the flow.

Flow control valves (FCV). These valves try to maintain the flow through the valve at a set value. They can be dealt with in a very similar way to a closed pipe, as discussed above, changing the zero in equation (12) for the set value of the FCV. This results in a system with the same structure as (15), which is solved in the same way.

MODELING PRESSURE REGULATING VALVES

In this section we consider the inclusion of two different types of valves in the simulation: pressure reducing valves and pressure sustaining valves.

Pressure reducing valve (PRV). A PRV is used to reduce the pressure of the valve outlet to a given set value. The valve can be in three different status: i) if the inlet head is too low to provide the desired outlet pressure, the valve opens fully; ii) if the heads at the valve ends would produce a negative flow, the valve closes; iii) otherwise the valve is active and the outlet pressure is equal to the set value. The first two cases correspond to a normal pipe, possibly closed, and can be dealt with as described in previous sections. In the following paragraphs we discuss the third case.

In (Jeppson 1976), an active PRV is modelled in the context of the loop method by considering an independent path (or pseudo-loop) that goes from the downstream node of the PRV to a reservoir/tank. An energy balance equation is imposed on that path, replacing the energy equation of a loop containing the PRV. Additionally, if the PRV is contained in more than one loop, the rest of the loops have to be redefined so that they do not contain the PRV. The procedure produces a linear system with a matrix that is no longer symmetric.

Here we present another way to model the PRV. Like in (Jeppson 1976), a path from the downstream node of the PRV to a reservoir/tank is considered. However, the balance equation for that path is added, without replacing another equation, and the headloss at the PRV is added as a new unknown. The advantages are that there is no need to redefine the loops, and that the non-symmetric part of the system matrix is isolated.

Let us consider the network shown in figure 6. Initially, if the valve were a normal pipe, the system of linear equations at an iteration k would be:

$$\begin{pmatrix} d_1+d_2+d_3 & -d_3 & 0 & 0 \\ -d_3 & d_3+d_4+d_5 & -d_5 & 0 \\ 0 & -d_5 & d_5+d_7+d_8 & -d_7 \\ 0 & 0 & -d_7 & d_6+d_7+d_9 \end{pmatrix} \begin{pmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \\ \Delta\hat{q}_4 \end{pmatrix} = -\begin{pmatrix} h_1^k - h_2^k + h_3^k \\ -h_3^k + h_4^k - h_5^k \\ h_5^k + h_7^k - h_8^k \\ h_6^k - h_7^k + h_9^k \end{pmatrix} \tag{20}$$

If link 7 is an active PRV, the relationship between the flow circulating through the link and the head loss, given by d_7, is unknown. However, we can eliminate it, and instead introduce the head loss itself (h_7) as an unknown.

On the other hand, we know that the head at the downstream node of the valve, H_5, is equal to $e_5 + k_7$, where e_5 is the elevation of the node, and k_7 is the pressure setting for the PRV. Additionally, the head difference between the tank and node 5 must be equal to the sum of head losses along a path going from the tank to that node, e.g. the path going through pipes 2 and 8, i.e.:

$$h_2 + h_8 = H_6 - H_5$$

By approximating the nonlinear functions of the flows, h_2 and h_8, using the first two terms of the Taylor series, we get:

$$-H_6 + H_5 + h_2^k - d_2\Delta\hat{q}_1 + h_8^k - d_8\Delta\hat{q}_3 = 0 \tag{21}$$

Taking that into account, we have:

$$\begin{pmatrix} d_1+d_2+d_3 & -d_3 & 0 & 0 & 0 \\ -d_3 & d_3+d_4+d_5 & -d_5 & 0 & 0 \\ 0 & -d_5 & d_5+d_8 & 0 & 1 \\ 0 & 0 & 0 & d_6+d_9 & -1 \\ -d_2 & 0 & -d_8 & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta\hat{q}_1 \\ \Delta\hat{q}_2 \\ \Delta\hat{q}_3 \\ \Delta\hat{q}_4 \\ h_7 \end{pmatrix} = - \begin{pmatrix} h_1^k - h_2^k + h_3^k \\ -h_3^k + h_4^k - h_5^k \\ h_5^k - h_8^k \\ h_6^k + h_9^k \\ -H_6 + H_5 + h_2^k + h_8^k \end{pmatrix} \tag{22}$$

In a more general case, with any number of PRV, the previous system presents the form:

$$\begin{pmatrix} A & C \\ E & F \end{pmatrix} \begin{pmatrix} \Delta\hat{q} \\ \hat{h} \end{pmatrix} = \begin{pmatrix} b \\ \hat{b} \end{pmatrix} \tag{23}$$

or, equivalently:

$$A\Delta\hat{q} + C\hat{h} = b \tag{24}$$

$$E\Delta\hat{q} + F\hat{h} = \hat{b} \tag{25}$$

where, similarly to the case of check valves and FCV, A is the same matrix as that of the original system (20), only changing the value of the coefficients affected by the PRV (as if those valves had been replaced by pipes of zero resistance). No new non-zero elements are added to the matrix. C and F are incidence or topological matrices, the elements of which can only take the values 0 and ±1. In particular, C indicates the valves involved in each cycle, and F indicates the valves involved in each of the PRV paths. In the example above, the path from node 5 to node 6 does not contain any valve, and thus the only element of F is zero. Finally, E contains the headloss derivatives for the pipes in each of the PRV paths.

Operating in a similar way to section 5, we have:

$$\Delta\hat{q} = A^{-1}b - A^{-1}C\hat{h} \tag{26}$$

$$\hat{h} = (EA^{-1}C - F)^{-1}\left(EA^{-1}b - \hat{b}\right) \tag{27}$$

where $\hat{h}$ can be obtained from (27), which involves solving the linear systems $A^{-1}C$ and $A^{-1}b$, multiplying the results by E, and solving a small linear system with the matrix $(EA^{-1}C-F)$, with size equal to the number of active PRV. Then, $\Delta\hat{q}$ is obtained from the expression (26), where taking into account that $A^{-1}b$ and $A^{-1}C$ have already been computed.

It can be shown (see Appendix I) that the inverse of the matrix $(EA^{-1}C - F)$ exists if the following assumptions hold: (i) the system matrix of (23) is invertible, and (ii) the matrix A is invertible. The first condition is the same requirement existing also in the GGA method, e.g. in Epanet. The second condition follows from the fact that A is the matrix of the Newton-Raphson iteration (6) for the original network where active PRV have been replaced by zero-resistance pipes, and is consequently positive definite (Todini and Pilati 1988).

To sum up, active PRV can be treated without redefining the loops of the network and thus without changing the sparsity pattern of the system matrix, with a procedure which is very similar to that of check valves presented in section 5. The main difference is that in the case of PRV, the small system of equations introduced, with the matrix $(EA^{-1}C - F)$, is not symmetric. However, the matrix A is still symmetric and can be factorized using a Cholesky decomposition.

Pressure sustaining valves (PSV). These valves are very similar to PRV. In particular, a PSV tries to keep the inlet pressure at a set value. The approach described above for PRV is also valid for PSV, with the difference that we should use a path from a tank to the upstream node of the valve (instead of the downstream node).

CHOOSING AN INITIAL FLOW VECTOR

One of the difficulties found was the choice of a suitable initial flow vector for the loop method, i.e. a vector q^0 satisfying the mass balance equation (3). It was found that the choice of that vector has a considerable impact on the number of iterations performed.

In order to compute q^0, we build a spanning tree of the network and impose a given flow to each of the chord links. The flow of all the other links (the links in the tree) can be computed by going over the tree from the leaves to the root, and imposing the mass balance equation on each of the network nodes.

Different spanning trees and different flow values for the chord links can be used. We found that the best results were obtained using a minimum-resistance spanning tree, i.e. a spanning tree where the sum of resistances R of all the tree links is minimum, and assigning to each chord link a flow corresponding to a velocity of 1 m/s (this is the initial flow used by Epanet for all the links). The minimum-resistance spanning tree was obtained by means of Prim's algorithm (Prim 1957).

RESULTS

In this section, we present results that compare the GGA and loop methods, taking into account different aspects. We consider the hydraulic networks shown in table 1. **Net3** is the example network 3 of Epanet (Rossman 2000). **bwsn2m** is a modified version of the network 2 proposed in (Ostfeld et al. 2008), where parallel pipes (i.e. pipes having the same end nodes) and valves have been removed. In order to remove the valves, we focused on producing a steady-state model for the initial time step of the simulation, for which Epanet revealed that only one PSV was active, and the remaining four valves were closed. The four closed valves were removed, and the only active PSV was substituted by a pipe producing the same headloss. **urb** is a large real urban water network, the outline of which is shown in figure 7. Finally, the **exnet** network can be downloaded from the Centre for Water Systems of the University of Exeter (http://emps.exeter.ac.uk/media/universityofexeter/emps/research/cws/downloads/exnet.inp).

Realistic results of computing time should consider efficient implementations of the methods. For that reason, we use here the very efficient GGA implementation of Epanet, written in C, while for the loop method we have also used a C implementation which has been integrated with the source code of Epanet. Of course, the same optimization flags were used for compiling both codes. The code for the solution of linear systems using Cholesky factorization is taken from Epanet, and is exactly the same for both methods. This approach differs from other works such as (Creaco and Franchini 2014), (Elhay et al. 2014), where Matlab implementations of the methods are used. However, we use a simpler Matlab implementation of the loop method for the simulation of networks containing control valves, for which the computing time is not evaluated.

With respect to the time results, the times to be measured were in general very short. For that reason, the task under consideration was repeated a sufficient number of times to get an accumulated time of about a second, and then the average time was obtained. Additionally, the whole series of repetitions was run eleven times, and the median time was obtained. Times are in seconds, except where indicated otherwise. The machine were the tests were run was equipped with an Intel Core 2 Duo CPU at 3GHz, with 4GB RAM.

We first present results in table 2 that evaluate the loop selection methods presented in the paper, considering the sparsity of the resulting linear system matrix. The columns under **n** compare the matrix size for GGA and the loop method. We can see that, for normal networks like those used here, the matrix produced by the loop method is much smaller than that of the gradient method. The columns under **nnz(A)** show the number of non-zero elements of the linear system matrices, for the case of the GGA method and for each of the loop selection methods previously presented. As expected, the best results are achieved with **m2**, although the high computational requirements of the method, in terms of execution time and memory, prevent its use for the two larger networks (bwsn2m and urb). Among the other loop selection methods, **m4** is the best, producing a considerable difference in number of non-zero elements with respect to the GGA.

The columns under **nnz(L)** show the number of non-zeros of the Cholesky factor of the linear system matrices. The amount of non-zero elements of the factorized matrix, which is determined at the beginning of the simulation as a result of a reordering and a symbolic decomposition of the matrix, is a good indicator of the computing time necessary to solve the linear system. The number of non-zero elements of the factorized matrix is higher than that of the original matrix, and depends on the reordering method used. In our case, the method is Mimimum Degree (George and Liu 1989), the implementation of which has been taken from Epanet. We can see that there is a considerable reduction in the number of non-zero elements for **m4** with respect to the GGA.

(Elhay et al. 2014) also present results for the exnet network for a reformulated co-tree method (RCTM), reporting a 2% increase in the number of non-zeros of the factorized matrix with respect to GGA. The results in this paper are clearly better, with a reduction of 68% in the number of non-zeros with respect to GGA.

(Creaco and Franchini 2014) present results for an algorithm similar to **m2**, applying it to two sets of networks: a set of networks made up of rectangular loops, and another set of networks made up of hexagonal loops. The paper then analyses the sparsity of the matrices and the computing time. We have applied our loop selection method **m4** to those same networks, and have reached exactly the same results in terms of matrix sparsity, indicating that **m4** was able to obtain optimal results for those networks, with the advantage of being a very fast procedure, as will be shown next.

Considering now the computing time that is necessary to obtain the set of loops, the results are presented in table 3. Columns m1-p to m4-p correspond to implementations of the algorithms **m1**-**m4** in Python. Column m4adj-c corresponds to the implementation of **m4** in C language, and includes also the time to build the loop-adjacency information (i.e. for each loop, which other loops it overlaps with) necessary to determine the matrix structure. As explained above, the method **m2** could not be run for networks **bwsn2m** and **urb**. It also takes more than 26 minutes for network exnet.

Table 3 shows clearly that obtaining the set of loops with the method **m4** is extremely fast. This implies that the loop method can be competitive even in cases where a single simulation is wanted, as opposed to situations where many simulations of networks with the same topology is required. Other approaches, such as (Creaco and Franchini 2014) and

(Elhay et al. 2014), assumed the second case.

Table 4 compares the number of iterations performed by the loop and GGA solvers, for the networks without control valves. The left part of the table shows the number of iterations for the initial instant of the simulation, while the right part presents the sum of iterations for all the time steps of an extended-period simulation. The difference in the number of iterations between the two methods is due to the fact that the initial solution used is different. The loop method requires in some cases some extra iterations, but the difference is very small.

We now analyze the time for the simulation of the networks. First, we consider the time per iteration of the non-linear solver. This is shown in table 5 (times are in milliseconds because they are very small), where the column "speedup" is the time for GGA divided by the corresponding time for the loop method. The table also shows the time spent on the different tasks in the iteration. In particular, *newcoeffs* is the part that sets the values of the linear system coefficients, which involves computing the derivatives of the headloss formula. The task *linsolve* corresponds to the solution of the linear system that has been formed in *newcoeffs*. Finally, *newflows* is the part that computes the new vector of flows. In the case of the GGA method, this new vector is computed from the new heads obtained by solving the linear system. In the case of the loop method, it is formed using (4). As we can see, *linsolve* is the part where the loop method shines. The speedup achieved in this part is more than 5 for the two large networks, although the weight of *newcoeffs* (around 60% in both cases) masks the real advantages of the loop method.

Table 5 does not consider the computation of the initial balanced flow vector for the loop method, nor the computation of the heads, because those tasks are done only once for each time step, instead of in every iteration. Those two aspects are taken into account in table 6, that presents the time to solve all the steady-state problems in a complete extended-period simulation. The number of time steps in the simulation was shown in table 4.

We can see that the loop method, with the definition of loops proposed in this paper, is between 11% and 20% faster than the GGA method. This performance gain is especially important in the context of network design by means of an optimization process, which may require the solution of steady-state problems for thousands or millions of slightly different networks.

Finally, we tested our approach for the treatment of control valves, by means of a simple implementation written in Matlab, which was used to simulate the network of figure 5 (with a closed check valve in pipe 2), and the network of figure 6 (with the PRV assumed to be active). In both cases, convergence was achieved with few iterations, and the results matched those obtained with Epanet.

CONCLUSIONS AND FUTURE WORK

In this paper, we present contributions for overcoming the main disadvantages of the method based on loop flow equations.

As the first of these contributions, we present efficient procedures for selecting the network loops so as to achieve a highly sparse matrix. Results on the application of those procedures to four networks, some of them coming from large real WDS, are given. Method **m2**, based on (Kavitha et al. 2004), and similar to the algorithms presented in (Creaco and Franchini 2014), produces very good results in terms of sparsity and, although it presents excessive requirements in terms of execution time and memory needed, it can be used as a reference for other methods. We propose two other novel faster methods, **m3** and **m4**, and the latter is identified as the most suitable one, producing considerably less non-zero elements than the GGA solver.

This contribution leads to important reductions in the time to solve the linear systems, with speedup of more than 5 with respect to the GGA method for two of the networks considered, and more than 2 for the other one. Considering the whole problem of extended-period simulation, the speedup achieved is between 1.11 and 1.20. In a context of network design by means of an optimization process, requiring the solution of many steady-state problems of slightly different networks, this performance gain can be very important.

We also show that the method **m4** to obtain the set of loops is extremely fast, which makes the loop method a competitive option with respect to the GGA method, even in cases where a single simulation is needed.

The second contribution of the paper is the development of methods to include check valves and automatic control valves in the model, avoiding the need to redefine the loops initially selected. Preliminary results on small networks show the correctness of the approach, since it produces output which agrees with the Epanet solver.

Finally, future work is needed in order to do a more complete test of the approach for control valves, considering more realistic networks. Also to be explored is the consideration of an optimization framework for the simulation with control valves, where the status of these elements is not obtained by means of a heuristic method, but as a result of the optimization process. Another direction of work is to try to reduce the time needed for the computation of the linear system coefficients (the task referred to as *newcoeffs* in the paper), which is shown to be the most time consuming part for both the loop and the GGA methods.

APPENDIX I: INVERSE OF THE SCHUR COMPLEMENT

Let A be a non-singular matrix given by:

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \tag{28}$$

where $A_{1,1}$ and $A_{2,2}$ are square matrices of sizes $p \times p$ and $q \times q$, respectively, and $A_{1,1}$ is also non-singular. We now build the matrix L:

$$L = \begin{pmatrix} I_p & 0 \\ -A_{2,1}A_{1,1}^{-1} & I_q \end{pmatrix} \tag{29}$$

where I_p and I_q are identity matrices of sizes $p \times p$ and $q \times q$, respectively. We have that:

$$LA = \begin{pmatrix} A_{1,1} & A_{1,2} \\ 0 & A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2} \end{pmatrix} \tag{30}$$

Consequently,

$$\det(A) = \det(LA) = \det(A_{1,1})\det(A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2}) \tag{31}$$

Since $\det(A) \neq 0$ and $\det(A_{1,1}) \neq 0$, it follows that $\det(A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2}) \neq 0$.

REFERENCES

Arsene, C., Al-Dabass, D., and Hartley, J. (2012). “A study on modeling and simulation of water distribution systems based on loop corrective flows and containing controlling hydraulics elements.” *Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on*, 423–430 (Feb).

Creaco, E. and Franchini, M. (2014). “Comparison of newton-raphson global and loop algorithms for water distribution network resolution.” *Journal of Hydraulic Engineering*, 140(3), 313–321.

De Pina, J. (1995). “Applications of shortest path methods.” Ph.D. thesis, University of Amsterdam, Netherlands.

Deuerlein, J., Cembrowicz, R., and Dempe, S. (2005). *Hydraulic Simulation of Water Supply Networks Under Control*. Chapter 23, 1–12.

Deuerlein, J., Simpson, A., and Dempe, S. (2009a). “Modeling the behavior of flow regulating devices in water distribution systems using constrained nonlinear programming.” *Journal of Hydraulic Engineering*, 135(11), 970–982.

Deuerlein, J., Simpson, A., and Gross, E. (2009b). *The Never Ending Story of Modeling Control-Devices in Hydraulic Systems Analysis*. Chapter 71, 1–12.

Elhay, S., Simpson, A., Deuerlein, J., Alexander, B., and Schilders, W. (2014). “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), 04014040.

Epp, R. and Fowler, A. G. (1970). "Efficient code for steady-state flows in networks." *Journal of the Hydraulics Division*, 96(1), 43–56.

George, A. and Liu, J. (1989). "The evolution of the minimum degree ordering algorithm." *SIAM Review*, 31(1), 1–19.

Guidolin, M., Kapelan, Z., and Savic, D. (2013). "Using high performance techniques to accelerate demand-driven hydraulic solvers." *Journal of Hydroinformatics*.

Jeppson, R. W. (1976). *Analysis of flow in pipe networks*.

Kavitha, T., Liebchen, C., Mehlhorn, K., Michail, D., Rizzi, R., Ueckerdt, T., and Zweig, K. A. (2009). "Cycle bases in graphs characterization, algorithms, complexity, and applications." *Computer Science Review*, 3(4), 199 – 243.

Kavitha, T., Mehlhorn, K., Michail, D., and Paluch, K. (2004). "A faster algorithm for minimum cycle basis of graphs." *Automata, Languages and Programming*, J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, eds., Vol. 3142 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 846–857.

Martin, D. W. and Peters, G. (1963). "The application of newton's method to network analysis by digital computer." *Journal Institution of Water Engineers and Scientists*, 17(17), 115–129.

Ostfeld, A., Uber, J., Salomons, E., Berry, J., Hart, W., Phillips, C., Watson, J., Dorini, G., Jonkergouw, P., Kapelan, Z., di Pierro, F., Khu, S., Savic, D., Eliades, D., Polycarpou, M., Ghimire, S., Barkdoll, B., Gueli, R., Huang, J., McBean, E., James, W., Krause, A., Leskovec, J., Isovitsch, S., Xu, J., Guestrin, C., VanBriesen, J., Small, M., Fischbeck, P., Preis, A., Propato, M., Piller, O., Trachtman, G., Wu, Z., and Walski, T. (2008). "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.

Piller, O. and van Zyl, J. (2014). "Modeling control valves in water distribution systems using a continuous state formulation." *Journal of Hydraulic Engineering*, 140(11), 04014052.

Prim, R. C. (1957). "Shortest connection networks and some generalizations." *Bell System Technical Journal*, 36(6), 1389–1401.

Rossman, A. L. (2000). *Epanet 2 Users manual*. Water Supply and Water Resources Division, US Environment Protection Agency.

Rossman, L. A. (1999). *Water Distribution Systems Handbook*. publisher, Chapter COMPUTER MODELS/EPANET.

Simpson, A. (1999). *Modeling of Pressure Regulating Devices: The Last Major Problem to be Solved in Hydraulic Simulation*. Chapter 39, 1–9.

Todini, E. (2008). "On the convergence properties of the different pipe network algorithms." *Water Distribution Systems Analysis Symposium 2006*, 1–16.

Todini, E. and Pilati, S. (1988). "Computer applications in water supply: Vol. 1—systems analysis and simulation." Research Studies Press Ltd., Taunton, UK, Chapter A Gradient Algorithm for the Analysis of Pipe Networks, 1–20.

Travers, K. (1967). "The mesh method in gas network analysis." *Gas Jour.*, 332, 167–174.

F. Zhang, ed. (2005). *The Schur Complement and Its Applications*, Vol. 4 of *Numerical Methods and Algorithms*. Springer.

List of Tables

Network	*junctions*	*pipes*	*tanks*	*pumps*	*valves*
Net3	92	117	5	2	0
bwsn2m	12523	14313	4	4	0
urb	26627	29043	26	0	0
exnet	1891	2465	2	0	2

TABLE 1. Networks considered.

	n		nnz(A)					nnz(L)	
Network	**GGA**	**loop**	**GGA**	**m1**	**m2**	**m3**	**m4**	**GGA**	**m4**
Net3	92	27	211	128	84	93	90	279	94
bwsn2m	12523	1794	26840	16765	-	5840	5527	37500	6577
urb	26627	2416	55670	52762	-	10873	9601	81564	13722
exnet	1891	576	4309	4895	1664	1825	1695	6010	1935

TABLE 2. Number of non-zero elements in the system matrix and its Cholesky factor. Comparison of GGA vs loop solver with different loop definition methods.

Network	m1-p	m2-p	m3-p	m4-p	m4adj-c
Net3	0.00049	0.27700	0.00223	0.00140	0.00006
bwsn2m	0.07360	-	0.50500	0.20600	0.01870
urb	0.16400	-	1.68000	0.47200	0.05799
exnet	0.00897	1565.0	0.05780	0.02360	0.00270

TABLE 3. Time to determine the set of loops.

	First time step		**All timesteps**		
Network	GGA	loop	Time steps	GGA	loop
Net3	5	6	27	86	89
bwsn2m	8	8	1	8	8
urb	6	7	13	30	31

TABLE 4. Number of iterations.

Net3	GGA	loop	speedup
newcoeffs	0.0164	0.0168	0.98
linsolve	0.0073	0.0034	2.16
newflows	0.0029	0.0034	0.85
Total iteration	0.0265	0.0235	1.13
bwsn2m	GGA	loop	speedup
newcoeffs	1.8963	1.8586	1.02
linsolve	0.9257	0.1720	5.38
newflows	0.2076	0.3377	0.61
Total iteration	3.0295	2.3683	1.28
urb	GGA	loop	speedup
newcoeffs	4.1298	3.9716	1.04
linsolve	2.0773	0.4016	5.17
newflows	0.9517	1.0653	0.89
Total iteration	7.1588	5.4386	1.32

TABLE 5. Execution time (in milliseconds) for a single iteration of the non-linear solver.

Network	GGA	loop	speedup
Net3	0.00221	0.00198	1.11
bwsn2m	0.02667	0.02220	1.20
urb	0.23565	0.20079	1.17

TABLE 6. Time for all steady-state problems in a simulation.

List of Figures

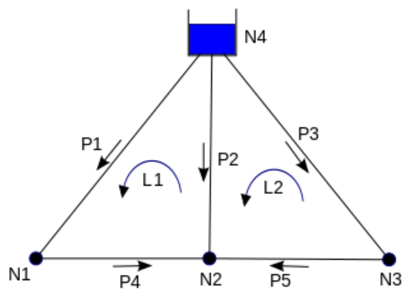

FIG. 1. A very simple network

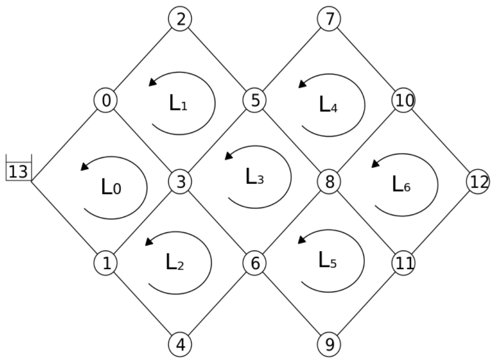

FIG. 2. Sample network, with a set of independent loops

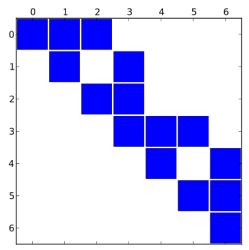

FIG. 3. Sparsity pattern for the loops defined in the sample network

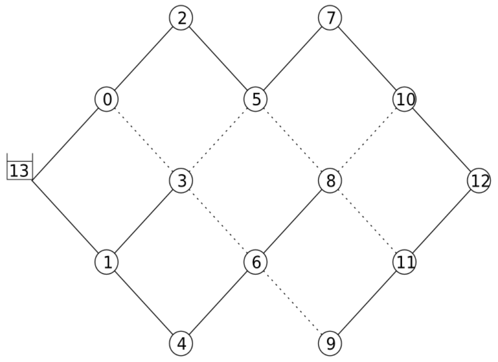

FIG. 4. A possible spanning tree for the sample network

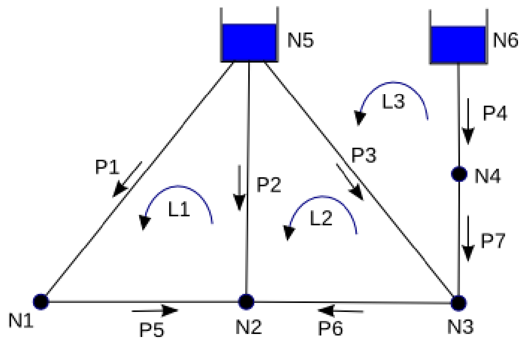

FIG. 5. A simple network

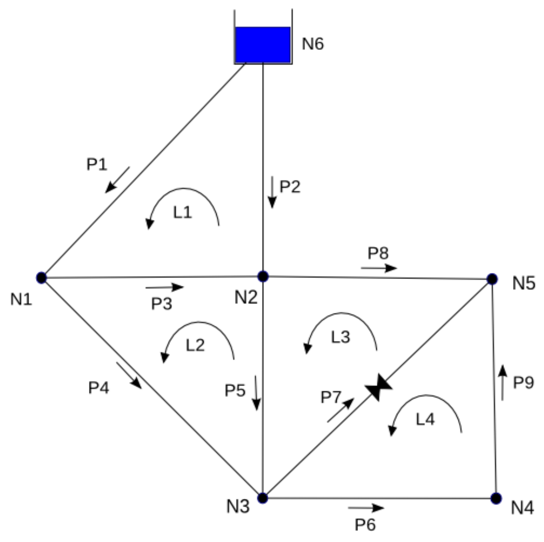

FIG. 6. Network with a PRV

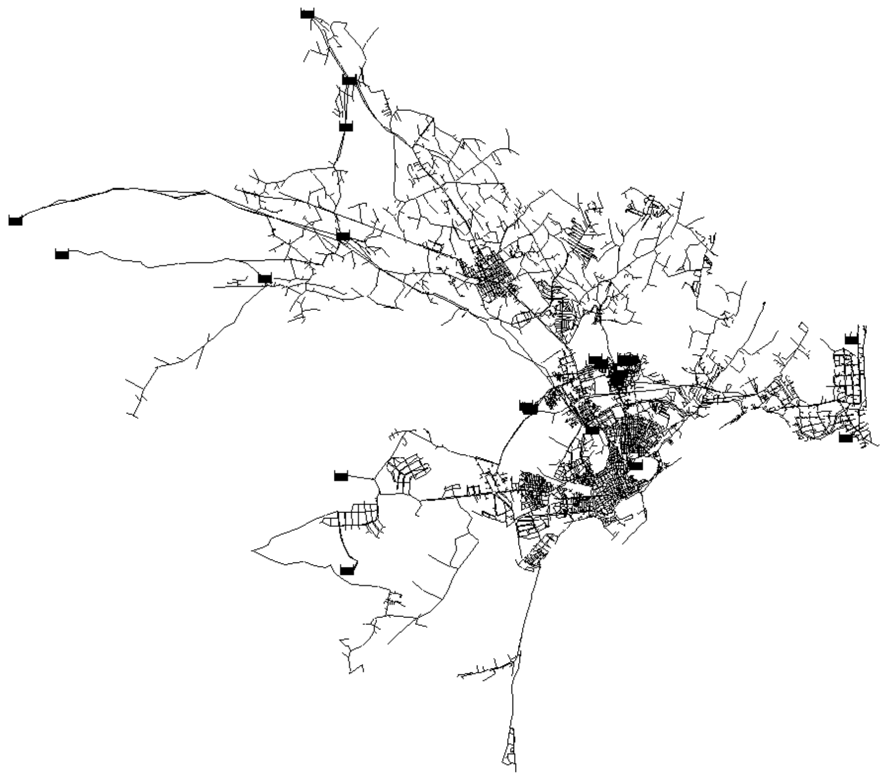

FIG. 7. Outline of network urb