Testing linear solvers for global gradient algorithm

Orazio Giustolisi and Naser Moosavian

ABSTRACT

Steady-state Water Distribution Network models compute pipe flows and nodal heads for assumed nodal demands, pipe hydraulic resistances, etc. The nonlinear mathematical problem is based on energy and mass conservation laws which is solved by using global linearization techniques, such as global gradient algorithm (GGA). The matrix of coefficients of the linear system inside GGA belongs to the class of sparse, symmetric and positive definite. Therefore a fast solver for the linear system is important in order to achieve the computational efficiency, especially when multiple runs are required. This work aims at testing three main strategies for the solution of linear systems inside GGA. The tests are performed on eight real networks by sampling nodal demands, considering the pressure-driven and demand-driven modelling to evaluate the robustness of solvers. The results show that there exists a robust specialized direct method which is superior to all the other alternatives. Furthermore, it is found that the number of times the linear system is solved inside the GGA does not depend on the specific solver, if a small regularization to the linear problem is applied, and that pressure-driven modelling requires a greater number which depends on the size and topology of the network and not only on the level of pressure deficiency.

Key words | algebraic multi-grid method, direct solvers, iterative solvers, network analysis, pipe networks, water distribution systems

Orazio Giustolisi
Department of Civil Engineering and Architecture,
Technical University of Bari,
Bari,
Italy

Naser Moosavian (corresponding author)
Civil Engineering Department,
University of Torbat-e-Heydarieh,
Torbat-e-Heydarieh,
Iran
E-mail: *naser.moosavian@yahoo.com*

NOTATION

$\mathbf{A}$	coefficient matrix
$\mathbf{A}_{nn}$	coefficient matrix in GGA
$\mathbf{A}_{pp}$	diagonal matrix in GGA
$\mathbf{A}_{np}$, $\mathbf{A}_{pn}$, $\mathbf{A}_{p0}$	topological incidence sub-matrices in GGA
b	vector of known terms
$\mathbf{d}_n$	column vector of demands
$\mathbf{D}$	diagonal matrix
$\mathbf{D}_{nn}$	diagonal matrix of derivatives of the pressure-demand with respect to $\mathbf{H}_n$
$\mathbf{D}_{pp}$	diagonal matrix of derivative of head losses with respect to $\mathbf{Q}_p$
$\mathbf{F}_n$	temporary matrix used in GGA
$\mathbf{H}_0$	column vector of known nodal heads
$\mathbf{H}_n$	column vector of unknown nodal heads
iter	counter for iterations
$\mathbf{L}$	lower triangular matrix
$\mathbf{M}$	pre-conditioner of the iterative method
n_n	number of nodes with unknown heads
n_p	number of pipes
n_0	number of nodes with known heads
$\mathbf{P}$	permutation matrix of a symmetric ordering strategy
$\mathbf{Q}$	orthogonal matrix
$\mathbf{Q}_p$	column vector of unknown pipe flow rates
$\mathbf{R}$	upper triangular matrix
$\mathbf{U}$	upper triangular matrix
x	vector of unknowns

INTRODUCTION

In recent years information technologies have made more data available and consequently the water industry is interested in utilizing those records to inform timely

doi: 10.2166/hydro.2014.136

operational decisions (for security, leakage management, energy management, etc.). This in turn requires the hydraulic steady-state simulation of larger and larger network systems with increased detailed information about their layout, such as, for example, connection to properties and minor subsystems. Numerous model runs in real-time are required to find near optimal management solutions for networks consisting of thousands of nodes and pipes and this fact asks for computational efficiency. The improvement of the computational efficiency can be achieved by: (i) improved mathematics, for example, new/more robust numerical techniques to manage the solution of the system equations; (ii) better technology, for example, parallelization of existing codes for multi-cores processing and/or Graphic Processor Units (GPU) (Crous *et al.* 2011; Wu & Lee 2011; Guidolin *et al.* 2012); and (iii) innovative engineering, i.e. techniques for simplifying the topological representation of pipe networks while preserving the accuracy of the analysis as for example in Giustolisi (2010) and Giustolisi *et al.* (2012).

Simulation models for hydraulic system networks are based on the solution of the nonlinear system of equations related to energy and mass balance principles. The most widely used algorithm is the global gradient algorithm (GGA) (Todini & Pilati 1988; Todini & Rossman 2013). It is implemented in EPANET (Rossman 2000) and involves the iterative solution of a sparse symmetric system of linear equations.

The implementation of a robust and computationally efficient linear solver is a relevant issue considering that the analysis of hydraulic systems is generally repeated many times, such as for optimization purposes, extended period simulations, real-time decisions, etc.

Here, we are here interested in analyzing the numerical methods, see point (i). For a very large linear system of equations, several methods and their implementation in solvers exist. They can be divided into three classes: (a) direct solvers; (b) iterative solvers; and (c) the most recent multi-grid solvers. Direct methods are highly efficient but require a large memory space. Iterative methods require less memory but need a scalable pre-conditioner to remain competitive. Multi-grid methods are often efficient and scalable, but are sensitive to the condition number (Wesseling 2004) and their convergence and accuracy depend on it. Therefore, the most suitable solver depends on several parameters. One of the most important is the mathematical problem size (e.g. in the case of Water Distribution Network (WDN) modelling using GGA the number of nodes of the hydraulic system) which significantly influences the numerical complexity of the linear problem inside GGA.

The computing architecture, i.e. point (ii), can also influence the selection of the solver. For example, direct methods, as implemented today (e.g. CHOLMOD factorization routines used in Matlab package, available at www.cise.ufl.edu/research/sparse/cholmod), can already support multi-thread computing (Davis & Hager 1999, 2001, 2005).

Here, we are interested in the issue of understanding which method is the most robust and effective for the specific case of steady-state WDN modelling using GGA independently on the increase of velocity of the linear system solution due to the technology such as multi-thread processing or computing by GPU.

Therefore, an existing Matlab code used inside the WDNetXL system (www.hydroinformatics.it) for steady-state analysis has been slightly changed, in order to option the call to different linear solvers and to store the related total CPU time, without using parallel computing or multi-thread facility of the Matlab environment. It is worth noting that the most computational expensive operations in Matlab are performed by C built-in codes, i.e. Matlab works with internally highly optimized C functions for computational expensive tasks. Furthermore, the sparse matrix representation is always the same, inside Matlab and linear solvers, and a 32 bit environment was used.

For the purpose of testing solvers, eight real networks were used and the classical demand-driven analysis (DDA) and the pressure-driven analysis (PDA) were considered. PDA entails customer-demand variation with nodal pressure (Gupta & Bhave 1996) or more in general several demand components depending on the nodal pressure (Giustolisi & Walski 2012). In fact, in PDA demands can fail to values close to zero and the solution of the linear system inside GGA can become critical (high condition number) independently on the use of a regularization method (Piller 1995; Giustolisi 2010), for example using a minimum flow threshold equal to $10^{-6}/R$ m^3/s, as in this work, where R is the pipe hydraulic resistance.

Moreover, we study the number of iterations (i.e. the number of times the linear system inside GGA is solved)

of PDA vs. DDA which significantly influences the actual computational burden during multiple model runs for any technical purpose. We found that the number of iterations of the steady-state analysis is not only related to the level of pressure deficiency (i.e. number of nodes in pressure deficient condition and level of actual pressure) but also to the size of the network and to the type of topology.

The structure of the work is as follows. A brief description of steady-state WDN modelling is initially reported, DDA and PDA are formulated in a single modelling framework, and the system of linear equations being solved inside GGA is presented. Successively, three main strategies of linear system solvers – (i) direct methods, (ii) iterative methods and (iii) algebraic multi-grid method – are outlined. Then, the numerical study is presented; it is performed using eight real networks where nodal demands are sampled in order to evaluate robustness and effectiveness of each method.

Finally, the discussion of results demonstrates that direct methods (and in particular a specialized one) are robust and effective for the specific complexity of the linear system inside GGA and actual size of WDN models, without considering the possibility to reduce the problem size as for example using Enhanced GGA (Giustolisi *et al.* 2012). Furthermore, it was proved that larger looped networks generally do not require an increased number of iterations of the steady-state analysis passing from DDA to PDA because the multiple paths that can be activated in PDA conditions, varying the boundary conditions scenarios, makes the mathematical problem less complex. Thus, the only unquestionable source of growth of computational burden in PDA is the need of computing demands through pressure-demand relationships (Giustolisi & Walski 2012).

Although the tests on linear solvers inside GGA are somehow dependent on the programming environment, on the steady-state WDN model implementation and even on the use of a 32 bit programming environment, it was found that a LDL specialized routine, which is a variant of Cholesky factorization, is much more effective than the other solvers. This fact is due to its more compact code which takes some computational advantages from the restriction to real matrices (Davis 2005).

STEADY-STATE WDN MODELLING

Steady-state analysis of a hydraulic network composed of n_p pipes with unknown flow rates, n_n nodes with unknown heads (internal nodes) and n_0 nodes with known heads can be performed by solving the following non-linear system based on energy and mass balance conservation equations,

$$\begin{aligned} &\mathbf{A}_{pp}\mathbf{Q}_p + \mathbf{A}_{pn}\mathbf{H}_n = -\mathbf{A}_{p0}\mathbf{H}_0 \\ &\mathbf{A}_{np}\mathbf{Q}_p - \mathbf{d}_n(\mathbf{H}) = \mathbf{0}_n \end{aligned} \tag{1}$$

where $\mathbf{Q}_p = [n_p,1]$ column vector of unknown pipe flow rates; $\mathbf{H}_n = [n_n,1]$ column vector of unknown nodal heads; $\mathbf{H}_0 = [n_0,1]$ column vector of known nodal heads; $\mathbf{d}_n = [n_n,1]$ column vector of demands which may depend on head status of the system (head/pressure driven analysis); $\mathbf{A}_{pn} = \mathbf{A}_{np}^T$ and $\mathbf{A}_{p0}$ = topological incidence sub-matrices of size $[n_p,n_n]$ and $[n_p,n_0]$, respectively; $\mathbf{A}_{pp}\mathbf{Q}_p = [n_p,1]$ column vector of pipe head losses due to the pipe hydraulic resistance and any device (e.g. valves and pumps) installed on pipes. The conventional positive signs for the topological matrices can be assumed as in Todini (2003) or Giustolisi *et al.* (2008).

Once some boundary conditions are assumed, such as hydraulic resistance in pipes; nodal demands; tank levels; status of control valves; pump curves; etc., the solution of the nonlinear system in (1) provides the hydraulic system state ($\mathbf{Q}_p$; $\mathbf{H}_n$). The general GGA solution of system (1) can be obtained by iteratively solving the following equations (Todini 2003; Piller *et al.* 2003; Cheung *et al.* 2005; Giustolisi *et al.* 2008; Wu *et al.* 2009, Giustolisi & Walski 2012):

$$\begin{aligned} &\mathbf{F}_n^{iter} = \left(\mathbf{A}_{np}\mathbf{Q}_p^{iter} - \mathbf{C}_n\right) - \mathbf{A}_{np}\left(\mathbf{D}_{pp}^{iter}\right)^{-1}\left(\mathbf{A}_{pp}^{iter}\mathbf{Q}_p^{iter} + \mathbf{A}_{p0}\mathbf{H}_0\right) \\ &\mathbf{H}_n^{iter+1} = \left[\mathbf{A}_{np}\left(\mathbf{D}_{pp}^{iter}\right)^{-1}\mathbf{A}_{pn} + \mathbf{D}_{nn}^{iter}\right]^{-1}\mathbf{F}_n^{iter} \\ &\mathbf{Q}_p^{iter+1} = \mathbf{Q}_p^{iter} - \left(\mathbf{D}_{pp}^{iter}\right)^{-1}\left(\mathbf{A}_{pp}^{iter}\mathbf{Q}_p^{iter} + \mathbf{A}_{p0}\mathbf{H}_0 + \mathbf{A}_{pn}\mathbf{H}_n^{iter+1}\right) \\ &\mathbf{C}_n = \mathbf{d}_n \text{ and } \mathbf{D}_{nn}^{iter} = \mathbf{0}_{nn} \text{ for demand-driven analysis} \\ &\mathbf{C}_n = (\mathbf{d}_n(\mathbf{H}))^{iter} - \mathbf{D}_{nn}^{iter}\mathbf{H}_n^{iter} \text{ for pressure-driven analysis} \end{aligned} \tag{2}$$

where *iter* = counter for iterations; $\mathbf{D}_{pp}$ = diagonal matrix whose elements are the derivatives of the head loss functions with respect to $\mathbf{Q}_p$ and $\mathbf{D}_{nn}$ = diagonal matrix whose elements are the derivatives of the pressure-demand relationships with respect to pressures in the network. A key point of the algorithm in (2), as well as for other algorithms, is the iterative solution of the linear system of equations,

$$\mathbf{H}_n^{iter+1} = \left[\mathbf{A}_{np}\left(\mathbf{D}_{pp}^{iter}\right)^{-1}\mathbf{A}_{pn} + \mathbf{D}_{nn}^{iter}\right]^{-1}\mathbf{F}_n^{iter} = [\mathbf{A}_{nn}]^{-1}\mathbf{F}_n^{iter} \tag{3}$$

where in the specific case of GGA $\mathbf{A}_{nn}$ = sparse square symmetric positive definite matrix whose order equals the number of internal nodes n_n. It follows that hydraulic solvers based on GGA needs to perform the solution of a symmetric sparse linear system in an efficient and robust way because that solution drives the performance (in terms of convergence and robustness issues) of the GGA (Giustolisi *et al.* 2011). EPANET software package (Rossman 2000) uses inside GGA a direct method for the system in (3) with classic LL Cholesky factorization for sparse symmetric positive definite matrices together with a bandwidth minimization ordering strategy as in George & Liu (1981).

It is worth noting here that we refer to steady-state analysis without considering devices controlling pressure/flow or pressure controlled variable speed pumps or check valves. In fact, such devices alter the solution of the system in (1) because extra equations needs to be implemented in order to model them, furthermore the prediction of the status of those devices needs to be performed.

SOLUTION OF LINEAR SYSTEMS

The solution of systems of linear equations of the form $\mathbf{A}x = b$ ($\mathbf{A}$ is the coefficient matrix, b is the vector of known terms and x is the vector of unknowns, nodal head in WDNs) can be performed using two main strategies named direct and iterative methods (Saad 1996). Those strategies are independent of the specific properties of the matrix $\mathbf{A}$. Furthermore, the algebraic multi-grid strategy, using both direct and iterative methods, has been recently proposed in different fields, for example, in order to integrate shallow water equations using implicit methods (Spitaleri & Corinaldesi 1997).

Direct methods

Direct methods are traditionally used and a number of strategies are available. The first and most popular is the Gaussian elimination and that with partial pivoting is the most widely used algorithm for solving linear systems because of its stability and better time complexity. It can be seen as an LU factorization meaning that $\mathbf{A} = \mathbf{LU}$ ($\mathbf{L}$ = lower triangular matrix and $\mathbf{U}$ = upper triangular matrix) where $\mathbf{L}$ is the factor of the elimination phase and $\mathbf{U}$ of the back-substitution phase. The solution of the linear system is given by:

$$(\mathbf{LU})x = b \quad \Rightarrow \quad \mathbf{L}(\mathbf{U}x) = b \quad \Rightarrow \quad \mathbf{L}y = b \quad \Rightarrow \quad \mathbf{U}x = y \tag{4}$$

therefore after factorization two triangular systems are solved instead of the original one.

Other well-known general methods are the QR ($\mathbf{A} = \mathbf{QR}$, $\mathbf{Q}$ = orthogonal matrix and $\mathbf{R}$ = upper triangular matrix) factorization using Householder transformations or Givens rotations. When $\mathbf{A}$ is symmetric positive definite $\mathbf{U} = \mathbf{L}^T$ and $\mathbf{A} = \mathbf{LL}^T$ that is the well-known LL Cholesky factorization or its variant $\mathbf{A} = \mathbf{LDL}^T$ named LDL factorization.

The matrix $\mathbf{A}$ can be square or rectangular and dense or sparse. Sparse means that the matrix has few non-zero entries although 'few' is not well defined in scientific literature but in water system network analysis this mathematical condition for the coefficient matrix of the linear system in (3) always occurs (Giustolisi *et al.* 2011). For sparse matrix, there are effective ways to store the matrix $\mathbf{A}$ and the related factors (e.g. $\mathbf{L}$ and $\mathbf{U}$) in memory to avoid storing zeros of such a matrix. When only the nonzero entries are stored, there are some consequences for the factorization algorithm. For example the position of non-zeros is not known in advance, e.g. in $\mathbf{L}$ and $\mathbf{U}$ factors. Consequently a critical issue in linear solvers for sparse problems is 'ordering'. Ordering means permuting the rows and columns of the matrix $\mathbf{A}$ so that the fill-in in the $\mathbf{L}$ and $\mathbf{U}$ factors is reduced to a minimum. A fill-in is defined as non-zeros appearing in

either of the matrices **L** or **U**, while the element in the corresponding position in **A** is a zero. Fill-in has obvious consequences to memory because the factorization algorithm could create dense **L** and **U** factors that can exhaust available memory although the matrix **A** is sparse. A good ordering algorithm yields a low fill-in. Finding the ordering that gives minimal fill-in is a NP (nondeterministic polynomial time) complete problem (i.e. a polynomial time algorithm to solve the problem is not known). The solution for a sparse linear system needs: an analysis of fill-in reduction by matrix permutation (ordering); a numerical factorization and the phase solution of the problem. The ordering phase exploits the matrix structure which is sometimes referred to as the symbolic factorization step and (optionally) determines a pivot sequence and data structures for efficient factorization. A good pivot sequence significantly reduces both memory requirements and the number of floating-point operations required. The numerical factorization phase uses the pivot sequence to factorize the matrix (some strategies scale the matrix prior to the factorization). The solution phase performs forward elimination followed by back-substitution using the stored factors. The solution phase may include iterative refinement. Of the different phases in a serial implementation, the numerical factorization is usually the most time-consuming, while the solution phase is generally significantly faster (Giustolisi *et al.* 2011).

In the specific case of the system in (3) with a symmetric matrix $\mathbf{A}_{nn}$ using, for example, the Cholesky LDL:

$$\begin{aligned}&\left(\mathbf{P}_{nn}\mathbf{A}_{nn}^{iter}\mathbf{P}_{nn}^{T}\right)\left(\mathbf{P}_{nn}\mathbf{H}_{n}^{iter+1}\right)=\mathbf{P}_{nn}\mathbf{F}_{n}^{iter+1}\qquad\text{and}\\&\mathbf{P}_{nn}\mathbf{A}_{nn}^{iter}\mathbf{P}_{nn}^{T}=\mathbf{L}_{nn}\mathbf{D}_{nn}\mathbf{L}_{nn}^{T}\mathbf{L}_{nn}y=\mathbf{P}_{nn}\mathbf{F}_{n}^{iter+1}\Rightarrow\mathbf{D}_{nn}z=y\\&\Rightarrow\mathbf{L}_{nn}^{T}\left(\mathbf{P}_{nn}\mathbf{H}_{n}^{iter+1}\right)=z\end{aligned}\tag{5}$$

where $\mathbf{P}^T$ = permutation matrix of a symmetric ordering strategy (i.e. the same ordering is applied to rows and columns); **L** = lower triangular matrix with unit diagonal and **D** = diagonal matrix. It follows that the **L** factor in Equation (5) should have a low fill-in because of ordering by **P**. $\mathbf{P}_{nn}\mathbf{H}_n$ and $\mathbf{P}_{nn}\mathbf{F}_n$ are the ordered vectors of solution and known term, respectively, while a proper indexing using the diagonal elements of **P** easily gives the **H** solution in the original order. Therefore, the use of Cholesky LDL decomposition instead of the classical Cholesky LL is related to its applicability for positive indefinite matrices because some square roots in LL factorization are avoided, achieving a faster factorization also.

It is important to consider that increasing the size of the mathematical problem the matrix, $\mathbf{A}_{nn}$ of steady-state WDN models might become closer and closer to semi-definite. Furthermore, in PDA some network sections might have flow rates close to zero making $\mathbf{A}_{nn}$ indefinite because $\mathbf{D}_{nn}$ is not invertible (Giustolisi *et al.* 2011). This makes the linear problem ill-conditioned, therefore the accuracy and robustness of direct solvers decrease because of the effect of the round off errors due to the finite precision of the computing environment and their propagation due to the increased number of arithmetic operations to compute the solution.

Iterative methods

Any system of equations can be solved by direct methods. Unfortunately, the triangular factors of sparse matrices are not sparse, so the computational burden of these methods can be quite high, increasing the problem size. Furthermore, the required model accuracy of the technical problem, which is also related to the errors in the domain and boundary conditions representation, is usually much lower than the accuracy of the direct solvers. Therefore, sometimes there is no reason to try solving the linear system with much higher accuracy and this leaves an opening for iterative methods especially for large size domains (i.e. networks), because the increase of the computational burden of iterative solvers are less sensitive to problem size and considering the advances in parallel computing.

Differently from the direct methods where the matrix **A** is decomposed into its factors, in iterative methods, the first approximate solution is somehow guessed and equations to systematically improve it are then used. For this purpose it is assumed $\mathbf{A}=\mathbf{M}-\mathbf{N}$, consequently:

$$\begin{aligned}&\mathbf{A}x=b\quad\Leftrightarrow\quad(\mathbf{M}-\mathbf{N})x=b\quad\Leftrightarrow\\&\mathbf{M}^{-1}(\mathbf{M}-\mathbf{N})x=\mathbf{M}^{-1}b\end{aligned}\tag{6}$$

where **M** = pre-conditioner of the iterative method. For the effectiveness of the method, **M** should be easily inverted (i.e. from a practical point of view it should be diagonal,

tri-diagonal, triangular, block tri-diagonal or triangular, etc.) and a good approximation to $\mathbf{A}$ so that $\mathbf{N}$ is small in some sense. Equation (6) opportunely rewritten provides:

$$(\mathbf{I}-\mathbf{M}^{-1}\mathbf{N})x=\mathbf{M}^{-1}b \quad \Leftrightarrow \quad \mathbf{I}x=\mathbf{M}^{-1}\mathbf{N}x+\mathbf{M}^{-1}b \quad \Leftrightarrow \quad x_{k+1}=\mathbf{M}^{-1}\mathbf{N}x_k+\mathbf{M}^{-1}b \tag{7}$$

The third of Equation (7) is the general formulation of iterative methods to solve a linear system of equations. It is easy to demonstrate that if x_k approaches to x_{k+1} it converges to the original solution of the linear system, i.e. the first of Equation (7) which is a pre-conditioned $\mathbf{A}x=b$. It is worth noting that for iterative methods to be effective, solving the system (7) must be computationally not expensive and the method must converge rapidly if a good initial guess of x is given. Furthermore, if for example $\mathbf{M}=\mathbf{A}=\mathbf{LU}$, the iteration form in (7) becomes $\mathbf{x}=(\mathbf{LU})^{-1}b$ being $\mathbf{N}=\mathbf{0}$, then the direct method arises.

As presented in a previous section, the incomplete LL zero fill-in (ILL) is usually used as a pre-conditioner for the Preconditioned Conjugate Gradient (PCG) iterative solver. As reported above, in order to explain the use of LDL factorization instead of LL, it was found in Giustolisi *et al.* (2011) that, for the same reason the ILUTP factorization (Saad 1996) (staying for incomplete LU factorization with Threshold and Pivoting) instead of ILL is effective. Other pre-conditioners have been found unstable and ineffectual with respect to convergence and robustness performance in steady-state WDN modelling (Giustolisi *et al.* 2011).

It is noted that increasing the problem (network) size, i.e. the condition number of the linear problem, the convergence rate, for a given accuracy, of the iterative solvers generally decreases.

Algebraic multi-grid methods

The accuracy and robustness of direct solvers decreases for large size problems and the computational burden increases more than linearly. On the other hand, finding and/or computing a good pre-conditioner for iterative solvers can be computationally more expensive than using a direct solver and it results in the lack of robustness of the iterative methods. Furthermore, the convergence rate of iterative solvers is influenced by the condition number of the system matrix.

The basic idea of multi-grid method MG is then to combine results obtained on different scales to overcome the drawback of direct solvers, applying results from one scale to eliminate certain error components of the approximation of the solution on another scale.

The general framework of MG is composed of three phases:

1. Smoothing: reducing high frequency errors, for example using a few iterations of the Gauss–Seidel method.
2. Restriction: down sampling the residual error to a coarser grid.
3. Interpolation or Prolongation: interpolating a correction computed on a coarser grid.

Algebraic multi-grid (AMG) method is a subcategory of MG which is best developed for symmetric, positive (semi-) definite problems. Various recent research activities aim to apply AMG to systems of partial differential equations such as Navier–Stokes equations or structural mechanics problems, and recently in WDN modelling (Zecchin *et al.* 2012). Therefore, AMG is guaranteed to converge for $\mathbf{A}_{nn}$ (a sparse square symmetric positive definite matrix) of GGA.

The AMG method combines the effect of a smoother and a coarse grid correction. The smoother is fixed and generally based on a simple iterative method such as the Gauss–Seidel method. The coarse grid correction consists of computing an approximate solution to the residual equation on a coarser grid. This solution is then transferred back to the actual grid by means of an appropriate prolongation. This coarse grid correction is entirely defined once the prolongation is known, that is, once an appropriate prolongation matrix has been set up by applying a so-called coarsening algorithm to the system matrix.

A famous approach to accelerate AMG is to use them as pre-conditioners for Krylov methods such as the PCG method (Saad 1996).

Here we use Aggregation Based Algebraic Multi-grid (AGMG) which considers coarsening by aggregation of the unknown nodal heads and leads to prolongation matrices with at most one non-zero entry per row, which are much sparser than the ones obtained by the classical AMG

approach (Notay 2010), therefore AGMG should be effective for the linear system inside GGA.

Furthermore, AGMG is used as pre-conditioner of a specialized conjugate gradient method, named flexible, if the matrix is symmetric positive definite in order to enhance the stability of the method in the presence of variable pre-conditioning (Notay 2010). For the details of this algorithm, readers can refer to Notay (2006, 2010).

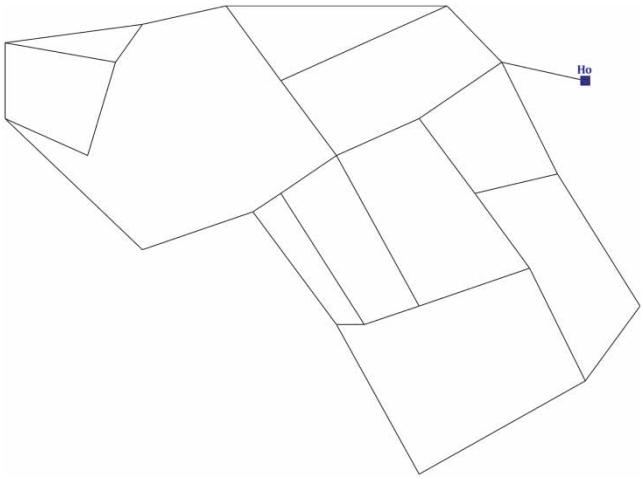

Figure 1 | Apulian network.

CASE STUDY

We will use the eight real networks reported from Figure 1–8:

1. Apulian, 24 nodes, 34 pipes, one reservoir, average number of pipes incident the nodes equal to 2.84 (Giustolisi *et al.* 2008).
2. C-Town, 396 nodes, 444 pipes, eight reservoirs/tanks and average number of pipes incident the nodes equal to 2.24 (Ostfeld *et al.* 2011).
3. Campania, 799 nodes, 847 pipes, 11 reservoirs/tanks and average number of pipes incident the nodes equal to 2.12.
4. Richmond, 872 nodes, 957 pipes, seven reservoirs/tanks and average number of pipes incident the nodes equal to 2.18 (Van Zyl *et al.* 2004).
5. WRC, 1,786 nodes, 1,995 pipes, four reservoirs/tanks and average number of pipes incident the nodes equal to 2.24 (Lippai 2005).
6. Exnet, 1,894 nodes, 2,467 pipes, two reservoirs/tanks and average number of pipes incident the nodes equal to 2.6 (Farmani *et al.* 2000).
7. BWSN, 12,527 nodes, 14,831 pipes, three reservoirs/tanks and average number of pipes incident the nodes equal to 2.36 (Ostfeld *et al.* 2008).

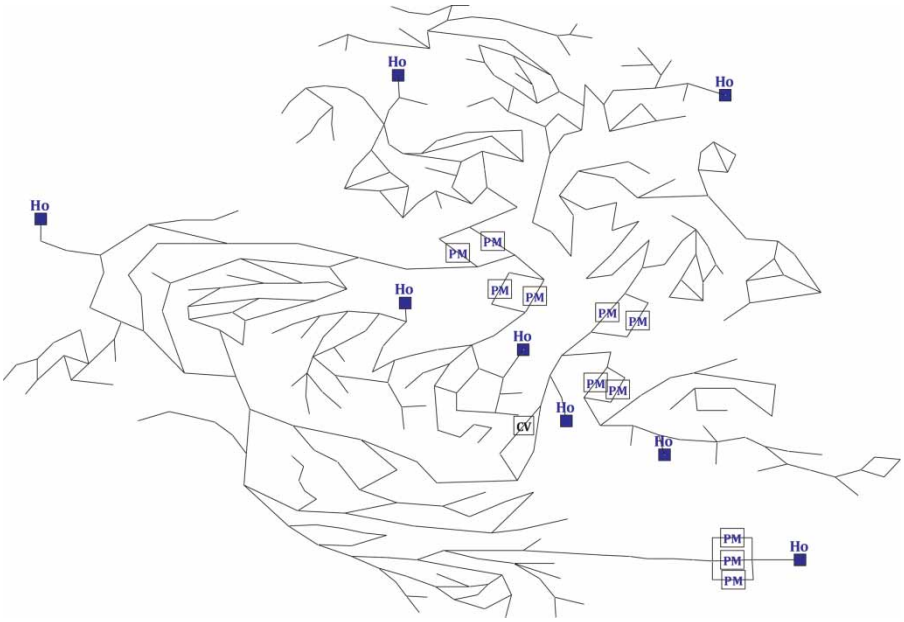

Figure 2 | C-TOWN network.

Figure 3 | Campania network.

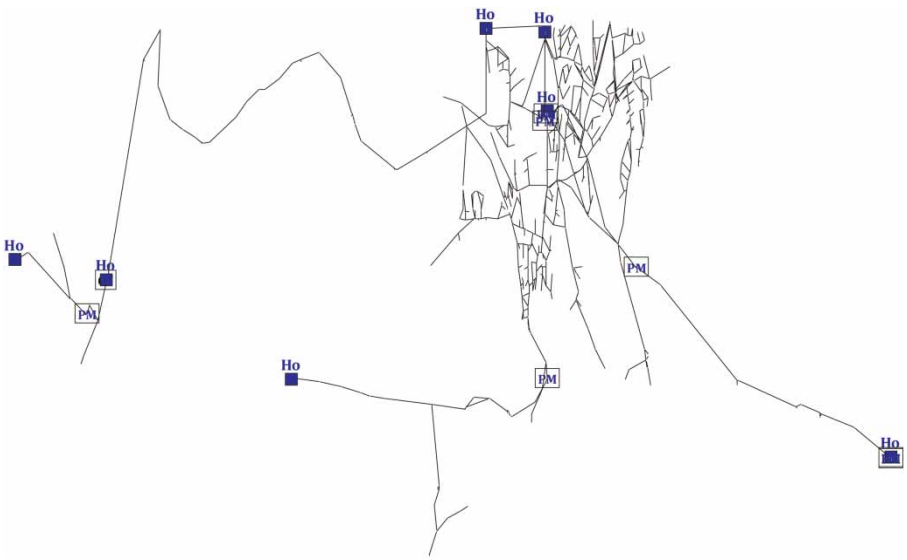

Figure 4 | Richmond network.

8. Big-Town, 26,967 nodes, 32,331 pipes, 28 reservoirs/tanks and average number of pipes incident the nodes equal to 2.4. Big-Town is a commercial sensitive network and thus it is not publicly available.

The numerical tests were performed by assuming 100 different networks in terms of demand variations which were randomly varied using a beta-decreasing function using as maximum demand eight times the original

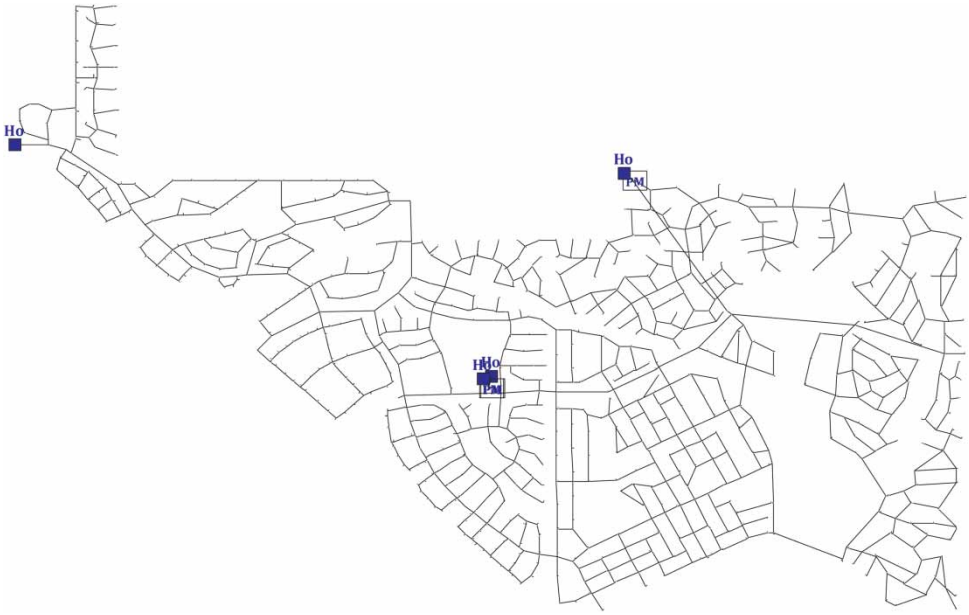

Figure 5 | WRC network.

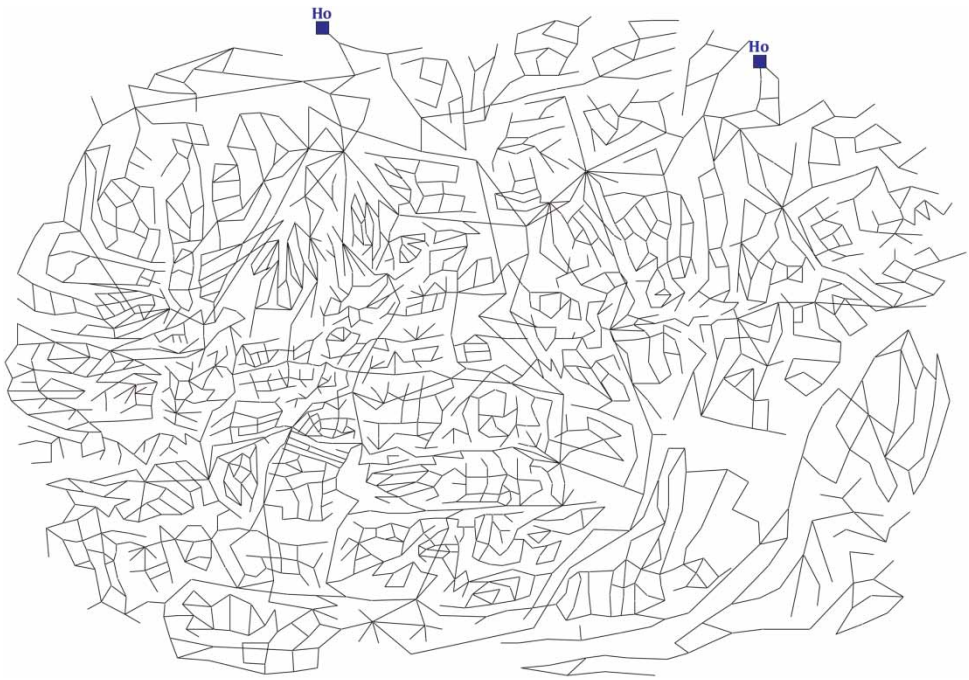

Figure 6 | Exnet network.

demand (Giustolisi *et al.* 2009), i.e. 100 steady-state analyses were run.

We will test some direct and iterative methods for linear solvers applied to GGA. As ordering methods, the approximate minimum degree permutation is here used (Giustolisi *et al.* 2011).

As reported by Giustolisi *et al.* (2011), EPANET uses a bandwidth minimization method, while it was found that

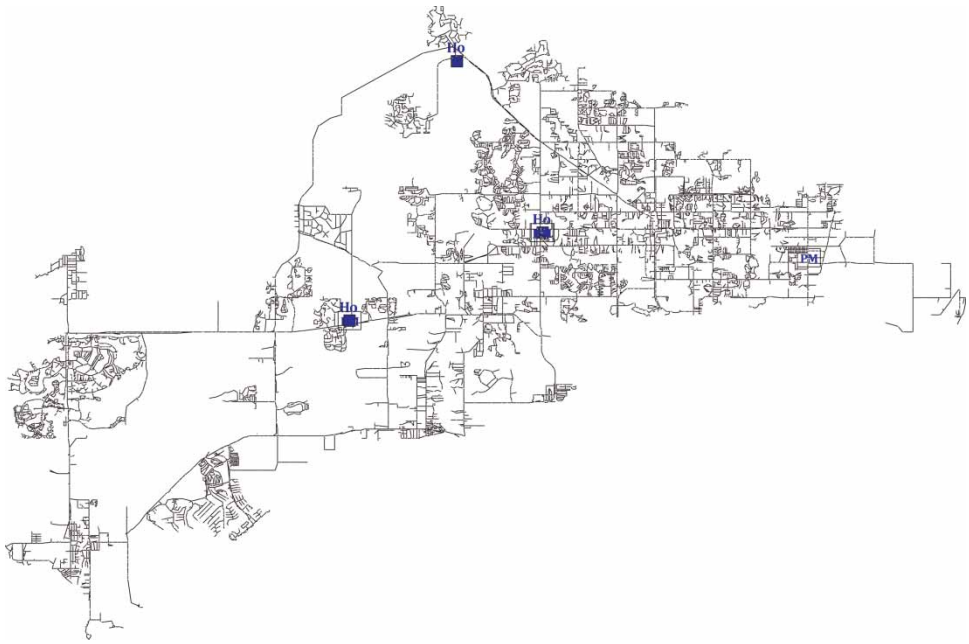

Figure 7 | BWSN network.

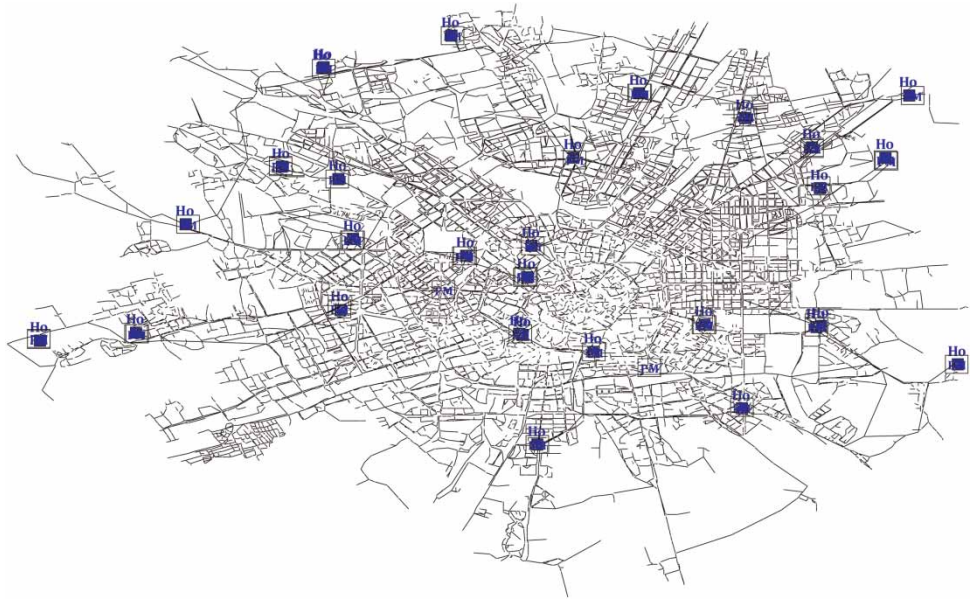

Figure 8 | Big-Town network.

approximate minimum degree permutation is more effective in GGA. In fact, minimum degree minimizes the number of fill entries introduced at each step of sparse Cholesky factorization and in theory is suboptimal but in practice often wins for medium-size problems, while bandwidth minimization try to keep all non-zero close to the diagonal and in theory and practice often wins for 'long, thin' problems.

The factorization used for direct methods are the standard Cholesky LL and LDL. Furthermore, a specialized version of the LDL for real matrices of coefficients is also used (Davis 2005) and the Matlab backslash function.

Backslash uses the LL method when a symmetric sparse matrix of coefficients is detected, but it performs the matrix analysis at each function call. In addition, three iterative methods have been investigated: the PCG; the symmetric least square (SYMLQ); and minimum residual (MINRES). They are all specific for symmetric positive sometimes semi-definite problems and they were all pre-conditioned using the ILUTP factorization as reported in the 'Iterative methods' section. Finally the AGMG was used as pre-conditioner of the flexible conjugate gradient method (Notay 2010).

The environment for analysis was Matlab R2011a-32bit for Windows 7.0 installed on a Notebook equipped with an Intel vPro *i*7 processor. The accuracy of each steady-state analysis was set as equal to 10^{-7}, i.e. the accuracy is computed as average value of the squared errors on mass and energy balance (Giustolisi *et al.* 2008).

RESULTS AND DISCUSSION

Tables 1 and 2 report the results of different linear solvers in eight real networks for demand-driven and pressure-driven steady-state analyses, respectively. For each strategy used, in the first and second columns the size of networks and the acronyms identifying the linear solvers are reported (i.e. LL = Cholesky LL decomposition, LDL = Cholesky LDL decomposition, LDL spec. = Cholesky LDL decomposition specialized for real number matrix, AGMG = Aggregation Based Algebraic Multi-grid, PCG = preconditioned conjugate gradient method, MINRES = minimum residual method, SYMLQ = symmetric least square method, Matlab = Backslash use '\' in Matlab software).

In the third column, the average CPU time for the single steady-state analysis, in the fourth and fifth columns, regarding the solution of the linear system, the average CPU time and the percentage of CPU time required (i.e. considering the number of times the linear system inside GGA is solved) with respect steady-state analysis, respectively, are reported.

It is first to report that all the tested linear solvers show the same robustness meaning that the use for regularization of the solution, a simple flow threshold equal to 10^{-6}/R m^3/sec which is especially useful in PDA when flow can approach null values, is effective.

Tables 1 and 2 show that the ranking of solvers with respect to CPU time is LDL spec, LL, LDL, PCG, MINRES, SYMLQ, Matlab and AGMG, in DDA, while PCG becomes the third and LDL the forth, in PDA.

Tables 1 and 2 demonstrate that the LDL decomposition (specialized for sparse symmetric positive definite real matrix) has the best performance both in DDA and PDA being between two and three times faster than the LL decomposition. This is due to the fact that, as already reported, the LDL specialized has a more compact code which takes some computational advantages from the restriction to real matrices. Furthermore, the results demonstrate that the direct solvers are the most effective for steady-state WDN modelling with GGA, while among the iterative solvers the PCG with the ILUP pre-conditioner is the most effective.

The standard use in Matlab of the backslash is not advised because it is a general routine implementing some controls to determine for example the matrix type, ordering, etc., which makes it much more robust as a general purpose linear system solver but much slower than specialized solvers for sparse, symmetric and positive definite matrix.

It is worth noting that PCG increases its effectiveness in PDA because the system of linear equations is better conditioned, i.e. the matrix of coefficients tends to being strictly diagonally dominant (because matrix $\mathbf{A}_{nn}$ in Equation (3) is generally strictly diagonal dominant), which improves convergence of iterative solvers together with the problem regularization.

In general, the tests show that the problem size of steady-state modelling does not increase the computational burden of direct solvers so that iterative solvers can be better alternatives, although an effective pre-conditioner is used and the problem regularization increases their convergence rate.

AGMG shows the least effective performance because it is actually well-suited for very large linear systems of equations (million nodes) which are inherently weakly conditioned (high condition numbers). On the contrary, the large size hydraulic systems in WDN modelling are far from the mathematical concept of large size systems. It is also reported that the average number of pipes incident (average number of pipes connecting to each node) is low in real networks, see the last column in Table 3 reporting the average nodal degree of each network that is a topology indicator which could be related to the sparseness of the system matrix of the linear problem inside GGA. In fact,

Table 1 | Performance of different solvers in demand-driven modelling

Network	Method	CPU WDN model (sec)	CPU Linear solver (sec)	% Solver time	Network	Method	CPU WDN model (sec)	CPU Linear solver (sec)	% Linear solver time
24 × 34	LL	0.0022	1.14E-04	23.63	396 × 444	LL	0.0084	4.82E-04	31.09
	LDL	0.0024	1.31E-04	25.53		LDL	0.0108	7.55E-04	37.73
	LDL spec	0.0020	5.99E-05	13.95		LDL spec	0.0078	2.35E-04	16.42
	AGMG	0.0041	4.61E-04	52.23		AGMG	0.0226	2.94E-03	70.43
	PCG	0.0029	2.34E-04	38.12		PCG	0.0113	8.52E-04	40.93
	MINRES	0.0029	2.34E-04	37.76		MINRES	0.0119	9.86E-04	44.78
	SYMLQ	0.0029	2.53E-04	40.27		SYMLQ	0.0123	1.03E-03	45.55
	Matlab	0.0027	1.89E-04	32.57		Matlab	0.0140	1.32E-03	51.47
799 × 847	LL	0.0242	8.26E-04	36.86	872 × 957	LL	0.0223	9.10E-04	36.14
	LDL	0.0306	1.30E-03	45.95		LDL	0.0284	1.44E-03	44.77
	LDL spec	0.0210	3.90E-04	20.02		LDL spec	0.0194	4.24E-04	19.32
	AGMG	0.0730	5.22E-03	77.26		AGMG	0.0667	5.80E-03	76.85
	PCG	0.0307	1.31E-03	46.16		PCG	0.0278	1.40E-03	44.67
	MINRES	0.0340	1.62E-03	51.27		MINRES	0.0308	1.74E-03	49.88
	SYMLQ	0.0345	1.66E-03	51.82		SYMLQ	0.0310	1.77E-03	50.46
	Matlab	0.0417	2.33E-03	60.27		Matlab	0.0382	2.56E-03	59.36
1,786 × 1,995	LL	0.0274	1.92E-03	34.94	1,894 × 2,467	LL	0.0386	2.36E-03	45.65
	LDL	0.0333	2.83E-03	42.58		LDL	0.0464	3.36E-03	53.83
	LDL spec	0.0235	8.79E-04	18.73		LDL spec	0.0287	1.02E-03	26.59
	AGMG	0.0794	1.20E-02	75.78		AGMG	0.1198	1.31E-02	81.60
	PCG	0.0327	2.76E-03	42.20		PCG	0.0451	3.20E-03	52.82
	MINRES	0.0361	3.43E-03	47.57		MINRES	0.0504	3.93E-03	58.06
	SYMLQ	0.0363	3.47E-03	47.86		SYMLQ	0.0510	4.00E-03	58.34
	Matlab	0.0447	5.16E-03	57.70		Matlab	0.0676	6.20E-03	68.26
12,527 × 14,831	LL	0.2407	1.40E-02	40.74	26,967 × 32,331	LL	0.9354	3.70E-02	49.51
	LDL	0.3075	2.36E-02	53.72		LDL	1.2654	6.30E-02	62.43
	LDL spec	0.1826	5.99E-03	22.96		LDL spec	0.6964	1.80E-02	32.25
	AGMG	0.7406	8.55E-02	80.80		AGMG	3.3240	2.23E-01	84.16
	PCG	0.2765	1.93E-02	48.95		PCG	1.1709	5.52E-02	59.06
	MINRES	0.3158	2.47E-02	54.79		MINRES	1.3203	6.74E-02	64.01
	SYMLQ	0.3131	2.46E-02	54.99		SYMLQ	1.3285	6.80E-02	64.12
	Matlab	0.4087	3.81E-02	65.32		Matlab	1.7640	1.02E-01	72.69

LL = Cholesky LL decomposition, LDL = Cholesky LDL decomposition, LDL spec. = Cholesky LDL decomposition specialized for real number matrix, AGMG = Aggregation Based Algebraic Multi-grid, PCG = preconditioned conjugate gradient, MINRES = minimum residual, SYMLQ = symmetric least square, Matlab = Backslash use '\' in Matlab software.

AGMG works better for denser coefficient matrix not corresponding to real hydraulic systems.

The computational burden of the solution of the linear system clearly depends on its effectiveness with respect to the other parts of the code implementing steady-state analysis. Tables 1 and 2 show that computational time of WDN modelling can decrease to less than 20% using the specialized routine for LDL.

However, it is considered that the parallelization, e.g. of PCG iterative solver, or the use of multi-thread

Table 2 | Performance of different solvers in pressure-driven modelling

Network	Method	CPU WDN model (sec)	CPU Linear solver (sec)	% Solver time	Network	Method	CPU WDN model (sec)	CPU Linear solver (sec)	% Linear solver
24 × 34	LL	0.0182	1.12E-04	16.42	396 × 444	LL	0.0287	5.13E-04	29.66
	LDL	0.0190	1.24E-04	17.43		LDL	0.0318	6.99E-04	36.48
	LDL spec	0.0172	5.85E-05	9.09		LDL spec	0.0238	2.24E-04	15.67
	AGMG	0.0262	3.87E-04	38.98		AGMG	0.0660	2.78E-03	69.93
	PCG	0.0212	2.23E-04	27.76		PCG	0.0313	6.86E-04	36.40
	MINRES	0.0211	2.17E-04	27.19		MINRES	0.0329	7.79E-04	39.33
	SYMLQ	0.0216	2.39E-04	29.33		SYMLQ	0.0331	8.01E-04	40.12
	Matlab	0.0206	1.86E-04	24.02		Matlab	0.0400	1.21E-03	50.33
799 × 847	LL	0.0297	8.76E-04	30.17	872 × 957	LL	0.0262	9.67E-04	29.53
	LDL	0.0341	1.30E-03	38.92		LDL	0.0297	1.43E-03	38.48
	LDL spec	0.0249	3.88E-04	15.96		LDL spec	0.0217	4.20E-04	15.47
	AGMG	0.0733	5.15E-03	71.86		AGMG	0.0647	5.81E-03	71.83
	PCG	0.0336	1.26E-03	38.42		PCG	0.0293	1.37E-03	37.55
	MINRES	0.0365	1.55E-03	43.58		MINRES	0.0318	1.69E-03	42.58
	SYMLQ	0.0369	1.59E-03	44.04		SYMLQ	0.0321	1.73E-03	43.18
	Matlab	0.0442	2.29E-03	52.95		Matlab	0.0388	2.55E-03	52.58
1,786 × 1,995	LL	0.1094	1.91E-03	36.62	1,894 × 2,467	LL	0.1661	2.38E-03	43.11
	LDL	0.1273	2.75E-03	45.19		LDL	0.1906	3.36E-03	51.34
	LDL spec	0.0867	8.35E-04	20.17		LDL spec	0.1194	1.03E-03	24.83
	AGMG	0.3179	1.18E-02	77.56		AGMG	0.5039	1.32E-02	79.78
	PCG	0.1123	2.05E-03	38.22		PCG	0.1827	3.08E-03	49.39
	MINRES	0.1140	2.14E-03	39.30		MINRES	0.1986	3.77E-03	54.27
	SYMLQ	0.1143	2.16E-03	39.63		SYMLQ	0.2056	3.77E-03	54.61
	Matlab	0.1759	5.05E-03	60.11		Matlab	0.2795	6.11E-03	65.63
12,527 × 14,831	LL	0.2965	1.39E-02	36.03	26,967 × 32,331	LL	1.4216	3.78E-02	45.89
	LDL	0.3741	2.38E-02	48.99		LDL	1.8789	6.43E-02	59.06
	LDL spec	0.2359	6.02E-03	19.61		LDL spec	1.0793	1.83E-02	29.24
	AGMG	0.8508	8.60E-02	77.65		AGMG	4.7823	2.27E-01	82.09
	PCG	0.3372	1.91E-02	43.58		PCG	1.7151	5.52E-02	55.61
	MINRES	0.3760	2.42E-02	49.44		MINRES	1.9131	6.71E-02	60.57
	SYMLQ	0.3777	2.42E-02	49.31		SYMLQ	1.9114	6.71E-02	60.66
	Matlab	0.4849	3.83E-02	60.67		Matlab	2.4977	1.01E-01	69.71

LL = Cholesky LL decomposition, LDL = Cholesky LDL decomposition, LDL spec. = Cholesky LDL decomposition specialized for real number matrix, AGMG = Aggregation Based Algebraic Multi-grid, PCG = preconditioned conjugate gradient, MINRES = minimum residual, SYMLQ = symmetric least square, Matlab = Backslash use '\' in Matlab software.

computing, e.g. in specialized LDL direct solver, can further decrease the computational burden related to the solution of the linear system inside GGA. Nevertheless, the parallelization or the use of multi-thread computing for other parts of the steady-state model code (e.g. matrix operations, head-loss and, for PDA, pressure-relationship computations) is more advisable if the linear solver is already efficient.

Considering the LDL specialized (Cholesky LDL decomposition specialized for real number matrix), its computational burden for the smallest network (24 × 34) is 13.95 and 9.09% in DDA and PDA, respectively. The computational

Table 3 | Pressure deficiency and number of iterations in demand-driven and pressure-driven modelling

Network	Deficiency index [10%]	Deficiency index [50%]	Deficiency index [95%]	Number of Iteration DDA	Number of Iteration PDA	Number of Iteration PDA-DDA	Average nodal degree
24 × 34	4.48%	35.57%	78.87%	4.66	26.77	22.11	2.84
396 × 444	0.17%	2.78%	11.54%	5.43	16.61	11.18	2.24
799 × 847	3.13%	4.65%	43.93%	10.79	10.23	−0.56	2.12
872 × 957	0.00%	0.42%	23.87%	8.84	8	−0.84	2.18
1,786 × 1,995	0.00%	1.66%	73.56%	5	20.94	15.94	2.24
1,894 × 2,467	1.56%	21.20%	79.66%	7.45	30.53	23.08	2.6
12,527 × 14,831	0.05%	0.62%	72.86%	7	7.69	0.69	2.36
26,967 × 32,331	0.00%	0.00%	100.00%	12.58	17.3	4.72	2.4

burden of that solver generally increases with network size as well as for the other solvers, especially direct ones. For example, in the largest network (26,967 × 32,331), it is 32.25 and 29.24% for DDA and PDA, respectively. This fact is consistent with the circumstance that increasing the model size (i.e. the linear system size) the computational burden of the direct solvers inside GGA increases more than linearly.

Table 3 summarizes the results of the number of iterations, fifth and sixth columns, required by GGA in DDA and PDA for each of the eight networks. It is noted that for the tested solvers the number of iterations required by the steady-state WDN modelling do not vary because they show a comparable accuracy.

This is a consequence of the small regularization which decreases the condition number of the linear problem inside GGA. Therefore, the accuracy of any tested linear solver is greater than the required accuracy for steady-state WDN modelling. In fact, it is noted that, actually, the related mathematical nonlinear problem is not very complex or large size from a numerical perspective.

Moreover, Table 3 reports the degree of pressure deficiency for the PDA case in the second, third and fourth columns. It is the percentage of nodal demands lower than 10, 50 and 95% of the customer requests, respectively, and the seventh column reports the difference of the number of iterations between PDA and DDA. Finally the last column of Table 3 reports the average nodal degree of each network.

Table 3 shows that the increase of the number of iterations is dependent on the level of pressure deficiency as it is easy to argue in advance. However, the network 799 × 847 and 872 × 957, which are similar because of less compact due to some long pipes connecting more compact network sections, show a similar number of iterations in DDA and PDA. This is caused by the fact that a pressure deficient condition in such a network should influence much fewer long pipes and related nodal demands than the compact areas.

For a similar reason the two largest networks (12,527 × 14,831 and 26,967 × 32,331) are characterized by a comparable number of iterations between PDA and DDA. In fact, the pressure deficient concept is not scalable because large size networks are characterized by an increasing number of flow paths allowing to find the extra hydraulic capacity, to deliver a greater nodal demand, with respect to small size network as for example the Apulian (24 × 34).

Thus, the only unquestionable source of increase of computational burden in PDA is the need of computing demands through pressure-demand relationships.

CONCLUSIONS

The present work describes the application of three methods which can be used to solve the linear system of equations inside GGA. They are tested considering both demand-driven and pressure-driven steady-state analysis of WDNs. The linear system inside GGA belong to the class of sparse, symmetric and positive definite; the fast and robust solution of such linear systems is an important issue in order to achieve computational efficiency, especially with respect to large size hydraulic networks and multiple runs, e.g. for system optimization.

The tests of several solvers using eight real networks show that direct solvers are the most effective from a computational burden perspective and a specialized code for real matrices using LDL decomposition has the best performance. The computational burden of the solution of the linear system can decrease to less than 20% using direct solvers, although the percentage of the linear solver weight inside the steady-state analysis is greatly influenced by the effectiveness of the rest of the WDN model code. Additionally, we demonstrate that the number of iterations (i.e. number of times the linear system is solved) does not relate to the specific solver. In fact the accuracy at each iteration, using a small regularization involving minimum flow, is sufficient considering the required accuracy of the steady-state analysis.

Finally, the results show that the increase in the number of iterations in PDA is dependent not only on the level of pressure deficiency, but also on the size and topology of the network.

REFERENCES

Cheung, P., Van Zyl, J. E. & Reis, L. F. R. 2005 Extension of Epanet for pressure driven demand modelling in water distribution system. *Proc. of Computer and Control in Water Industry, Water Management for the 21st Century*, vol. 1. CWS, University of Exeter, UK.

Crous, P. A., Van Zyl, J. E. & Roodt, Y. 2011 The potential of graphical processing units to solve hydraulic network equations. *J. Hydroinf.* **14** (3), 603–612.

Davis, T. A. 2005 Algorithm 849: a concise sparse Cholesky factorization package. *ACM Trans. Math. Softw.* **31** (4), 587–591.

Davis, T. A. & Hager, W. W. 1999 Modifying a sparse Cholesky factorization. *SIAM J. Matrix Analy. Appl.* **20** (3), 606–627.

Davis, T. A. & Hager, W. W. 2001 Multiple-rank modifications of a sparse Cholesky factorization. *SIAM J. Matrix Analy. Appl.* **22** (4), 997–1013.

Davis, T. A. & Hager, W. W. 2005 Row modifications of a sparse Cholesky factorization. *SIAM J. Matrix Analy. Appl.* **26** (3), 621–639.

Farmani, R., Savic, D. A. & Walters, G. A. 2000 Benchmark problems for design and optimisation of water distribution systems. In: *Proc. Advances in Water Supply Management* (C. Maksimovic, D. Butler & F. A. Memon, eds). Balkema Publishers, Lisse, pp. 249–256.

George, A. & Liu, J. W. H. 1981 *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ.

Giustolisi, O. 2010 Considering actual pipe connections in WDN analysis. *J. Hydr. Eng.* **136** (11), 889–900.

Giustolisi, O. & Walski, T. 2012 Demand components in water distribution network analysis. *J. Water Res. Plan. Manage.* **138** (4), 356–367.

Giustolisi, O., Savic, D. A. & Kapelan, Z. 2008 Pressure-driven demand and leakage simulation for water distribution networks. *J. Hydr. Eng.* **134** (5), 626–635.

Giustolisi, O., Laucelli, D. & Colombo, A. 2009 Deterministic vs. stochastic design of water distribution networks. *J. Water Res. Plan. Manage.* **135** (2), 117–127.

Giustolisi, O., Savic, D. A., Laucelli, D. & Berardi, L. 2011 Testing linear solvers for WDN models. In: *Proceedings of Computer and Control in Water Industry (CCWI)* (D. A. Savic, Z. Kapelan & D. Butler, eds). University of Exeter, Exeter, vol. 3. September 5–7, pp. 817–822.

Giustolisi, O., Laucelli, D., Berardi, L. & Savić, D. A. 2012 A Computationally efficient modeling method for large size water network analysis. *J. Hydr. Eng.* **138** (4), 313–326.

Guidolin, M., Kapelan, Z. & Savic, D. 2012 Using high performance techniques to accelerate demand driven hydraulic solvers. *J. Hydroinf.* **15** (1), 38–54.

Gupta, R. & Bhave, P. R. 1996 Comparison of methods for predicting deficient-network performance. *J. Water Res. Plan. Manage.* **122** (3), 214–217.

Lippai, I. 2005 Colorado Springs utilities case study: water system calibration/optimization. *Pipelines* 1047–1057. doi: 10.1061/40800(180)84.

Notay, Y. 2006 Aggregation-based algebraic multilevel preconditioning. *SIAM J. Matrix Analy. Appl.* **27** (4), 998–1018.

Notay, Y. 2010 An aggregation-based algebraic multigrid method. *Electron. Trans. Numer. Analy.* **37**, 123–146.

Ostfeld, A., Uber, J. G., Salomons, E., Berry, J. W., Hart, W. E., Phillips, C. A., Watson, J.-P., Dorini, G., Jonkergouw, P., Kapelan, Z., Pierro, F., Khu, S.-T., Savic, D., Eliades, D., Polycarpou, M., Ghimire, S. R., Barkdoll, B. D., Gueli, R., Huang, J. J., McBean, E. A., James, W., Krause, A., Leskovec, J., Isovitsch, S., Xu, J., Guestrin, C., Van Briesen, J., Small, M., Fischbeck, P., Preis, A., Propato, M., Piller, O., Trachtman, G. B., Wu, Z. & Walski, T. 2008 The battle of the water sensor networks (BWSN): a design challenge for engineers and algorithms. *J. Water Res. Plan. Manage.* **134** (6), 556–568.

Ostfeld, A., Salomons, E., Ormsbee, L., Uber, J. G., Bros, C. M., Kalungi, P., Burd, R., Zazula-Coetzee, B., Belrain, T., Kang, D., Lansey, K., Shen, H., McBean, E., Wu, Z. Y., Walski, T., Alvisi, S., Franchini, M., Johnson, J. P., Ghimire, S. R., Barkdoll, B. D., Koppel, T., Vassilijev, A., Kim, J. H., Chung, G., Yoo, D. G., Diao, K., Zhou, Y., Li, J., Liu, Z., Chang, K., Gao, J., Qu, S., Yuan, S., Yuan, Y., Prasad, D. T., Laucelli, D., Vamyakeridou, L., Lyroudia, S., Kapelan, Z., Savic, D., Berardi, L., Barbaro, G., Giustolisi, O., Asadzadeh, M., Tolson, B. A. & McKillop, R. 2011 The battle of the water calibration networks (BWCN). *J. Water Res. Plan. Manage.* **138** (5), 523–532.

Piller, O. 1995 Modeling the Behavior of a Network – Hydraulic Analysis and a Sampling Procedure for Estimating the

Parameters. PhD Thesis, University of Bordeaux I, Bordeaux, France.

Piller, O., Brémond, B. & Poulton, M. 2003 Least action principles appropriate to pressure driven models of pipe networks. World Water and Environmental Resources Congress (EWRI03). ASCE, Philadelphia, PA, USA.

Rossman, L. A. 2000 *Epanet2 User's Manual*. US Environmental Protection Agency, Cincinnati, OH, USA.

Saad, Y. 1996 *Iterative Methods for Sparse Linear Systems*, Chapter 10, Preconditioning Techniques. PWS Publishing Company, New York.

Spitaleri, R. M. & Corinaldesi, L. 1997 A multigrid semi-implicit finite difference method for the two-dimensional shallow water equations. *Int. J. Numer. Meth. Fluids* **25** (11), 1229–1240.

Todini, E. 2003 A more realistic approach to the 'extended period simulation' of water distribution networks. *Advances in Water Supply Management*. Balkema Publishers, Lisse, The Netherlands, pp. 173–184.

Todini, E. & Pilati, S. 1988 A gradient method for the solution of looped pipe networks. *Comput. Appl. Water Supply* **1**, 1–20.

Todini, E. & Rossman, L. 2013 Unified framework for deriving simultaneous equation algorithms for water distribution networks. *J. Hydr. Eng.* **139** (5), 511–526.

Van Zyl, J. E., Savic, D. A. & Walters, G. A. 2004 Operational optimization of water distribution systems using a hybrid genetic algorithm. *J. Water Res. Plan. Manage.* **130** (3), 160–170.

Wesseling, P. 2004 *An Introduction to Multigrid Methods*. John Wiley and Sons Ltd, Chichester.

Wu, Z. Y. & Lee, I. 2011 *Lesson for Parallelizing Linear Equation Solvers and Water Distribution Analysis. In Urban Water Management: Challenges and Opportunities*. CCWI, Exeter, pp. 21–26.

Wu, Z. Y., Wang, R. H., Walski, T. M., Yang, S. Y., Bowdler, D. & Baggett, C. C. 2009 Extended global-gradient algorithm for pressure-dependent water distribution analysis. *J. Water Res. Plan. Manage.* **135** (1), 13–22.

Zecchin, A., Thum, P., Simpson, A. & Tischendorf, C. 2012 Steady-state behavior of large water distribution systems: algebraic multigrid method for the fast solution of the linear step. *J. Water Res. Plan. Manage.* **138** (6), 639–650.

First received 8 December 2013; accepted in revised form 14 March 2014. Available online 29 April 2014