STEADY-STATE BEHAVIOR OF LARGE WATER DISTRIBUTION SYSTEMS: THE ALGEBRAIC MULTIGRID METHOD FOR THE FAST SOLUTION OF THE LINEAR STEP

A. C. Zecchin[1], P. Thum[2], A. R. Simpson[3], and C. Tischendorf[4]

ABSTRACT

The Newton-based global gradient algorithm (GGA) (also known as the Todini and Pilati method) is a widely used method for computing the steady-state solution of the hydraulic variables within a water distribution system (WDS). The Newton-based computation involves solving a linear system of equations arising from the Jacobian of the WDS equations. This step is the most computationally expensive process within the GGA, particularly for large networks involving up to $O(10^5)$ variables. An increasingly popular solver for large linear systems of the M-matrix class is the algebraic multigrid (AMG) method, a hierarchical-based method that uses a sequence of smaller dimensional systems to approximate the original system. This paper studies the application of AMG to the steady-state solution of WDSs through its incorporation as the linear solver within the GGA. The form of the Jacobian within the GGA is proved to be an M-matrix (under specific criteria on the pipe resistance functions), and thus able to be solved using AMG. A new interpretation of the Jacobian from the GGA is derived enabling physically based interpretations of AMG's automatically created hierarchy. Finally, extensive numerical studies are undertaken where it is seen that AMG outperforms the sparse Cholesky method with node reordering (the solver used in

[1]School of Civil, Environmental and Mining Engineering, The University of Adelaide, Australia. E-mail: azecchin@civeng.adelaide.edu.au

[2]Mathematical Institute, The University of Cologne, Germany.

[3]School of Civil, Environmental and Mining Engineering, The University of Adelaide, Australia.

[4]Mathematical Institute, The University of Cologne, Germany.

EPANET2), incomplete LU factorization (ILU) and PARDISO, which are standard iterative and direct sparse linear solvers.

Keywords: water distribution systems, global gradient algorithm, steady-state, algebraic multigrid methods.

INTRODUCTION

The steady-state solution of the hydraulic state variables within a water distribution system (WDS) involves the solution of a system of nonlinear equations. Many different formulations of these equations exist utilizing either the link flows, the nodal heads, the loop flows, or combinations thereof, as the primary variables. A popular method used to solve the WDS equations is the Newton-based global gradient algorithm (GGA) (also known as the Todini and Pilati method) (Todini and Pilati 1988; Todini 2011). Given the nonlinearity of the system of equations, the Newton-based computation of the solution involves an iterative two-step process. The first step (termed the *inner step*) involves computing the state variable update, which requires the solution of a linear system derived from the Jacobian of the WDS equations. The second step (termed the *outer step*) involves updating estimates of the state variables. The inner step is typically the most computationally expensive process within the GGA. For large systems of a practical size, the size of the Jacobian can be on the order of 10^5, making the use of efficient linear solvers important for the inner step. The computational cost of the steady-state solution of large networks becomes particularly critical for computations involving repeated network evaluations, such as extended period simulations, or network design involving iterative optimization methods.

An increasingly popular solver for large linear systems of the M-matrix class is the algebraic multigrid (AMG) method (Stüben 2001a). This method uses a hierarchical approach to solve the linear systems. Within this hierarchical approach, a sequence of lower dimensional systems are constructed that, in some sense, approximate the original system. The solutions of these lower dimensional systems are used to refine an approximate solution to the original system, where only the smallest system requires a direct solution. In this way, AMG provides an extremely computationally efficient approach to large systems. Typical applications for AMG are the numerical solution of elliptic partial differential equations involving large computational grids, which can be found in ground water simulation, oil reservoir simulation or fluid dynamics (Stüben et al. 2003; Stüben 2001b).This paper studies the application of AMG to the solution of the linear system that arises in the inner step of the GGA.

The structure of the paper is as follows. First, a brief background of WDS solution methods is given, the network equations are formulated, and the GGA is presented. Second, an overview of AMG is outlined. Third, issues pertaining to the application of AMG to the GGA are explored. In particular, the conditions under which the Jacobian in the GGA is an M-matrix, and hence suitable for solving using AMG, are demonstrated. Also within this section a new derivation for the Jacobian is presented which facilitates a physical network based interpretation of the AMG operations of coarse variable selection, and the construction of its hierarchy. Fourth, a detailed numerical study is presented where two variants of AMG are compared to the EPANET2 solver sparse Cholesky method with nodes reordering (SC+NR) (Rossman 2000), incomplete LU factorization (ILU) preconditioned conjugate gradient method, a popular sparse linear solver, and PARDISO, a fast and robust sparse direct linear solver. Finally, the conclusions are given.

THE STEADY-STATE SOLUTION OF WATER DISTRIBUTION SYSTEMS

Brief history of solution methods

Since Cross' seminal work (Cross 1936) on the solution of looped pipe networks through successive iterative corrections, many different solution methods have been proposed of which notable methods are: the first application of Newtons method to the solution of the pressure head form of the network equations (Martin and Peters 1963); the content minimisation model (Collins et al. 1978); the preconditioned Newton-Raphson method (Nielson 1989); and the famous global gradient method (Todini and Pilati 1988), which exploits a matrix block decomposition of the Newton-Raphson method. Notable recent work has focused on fundamental extensions to the steady-state network problem through the incorporation of head driven demand (Wu et al. 2009); nonlinear programming methods for reliably modelling control devices (Deuerlein et al. 2009); the inclusion of the exact analytic form of the Jacobian (by including the derivative of the friction factor) to improve convergence (Simpson and Elhay 2011); and the development of a regularization technique to enable the application of the global gradient method to networks containing small or near zero flows (Elhay and Simpson 2011). The current paper considers the utilisation of AMG, within the context of the global gradient method, for the fast solution of the Newton update step as this step is, typically, the most computationally expensive step within the solution process.

The WDS network equations

A WDS is a network of pipeline elements interconnected at nodes. Within this work, only nodes of the form of junctions and reservoirs are used. Consider a network comprised of n_p pipes, n_j variable-head nodes (junctions) and n_r fixed-head nodes (reservoirs). Given that the pipelines contain fully pressurised flow, and the losses within junctions are taken as negligible, there are fundamentally three types of equations that govern the steady-state behavior of the hydraulic variables (pressure and flow) of a WDS. The first type of equation, the headloss equation, describes the steady-state pressure along a pipe as a function of the flow through the pipe. That is, for a flow of Q_j in pipe j, the headloss $\Delta h_j = h_{uj} - h_{dj}$ (where h_{uj} is the upstream head, and h_{dj} is the downstream head) is given by

$$h_{uj} - h_{dj} = \mathcal{R}_j(Q) = r_j \, |Q_j| \, Q_j \tag{1}$$

where $\mathcal{R}_j$ is the hydraulic resistance function, and $r_j = r_j(Q_j)$ is the resistance coefficient which is given by $r_j = f_j(8/\pi^2 g)(L_j/D_j^5)$ where g = gravity, L_j = pipe length, D_j = pipe diameter, and f_j = Darcy-Weisbach friction factor, which is a function of the Reynolds number $\mathbb{R}_\mathrm{e} = |V|D/\nu$ (ν is the kinematic viscosity and V is the average velocity) and the relative roughness ϵ/D (ϵ is the pipe wall roughness) (Streeter et al. 1997). The functional dependence of r_j on Q_j is through the friction factor f_j.

The second network equation is associated with the lossless nature of a node, and it requires that all link ends connected to a common node share the same value of head. That is, the upstream and downstream heads of link j are related to the nodal heads at the variable-head nodes by

$$h_{uj} = h_i \text{ for all links } j \in \Lambda_{ui}, \text{ and } \ h_{dj} = h_i \text{ for all links } j \in \Lambda_{di} \tag{2}$$

where h_i is the variable head at node i, Λ_{ui} [Λ_{di}] are the set of all links with their upstream [downstream] node being the variable-head node $i = 1, \ldots, n_j$, and to the nodal heads at the fixed-head nodes by

$$h_{uj} = e_{li} \text{ for all links } j \in \Lambda^f_{ui}, \text{ and } \ h_{dj} = e_{li} \text{ for all links } j \in \Lambda^f_{di} \tag{3}$$

where e_{li} is the fixed elevation of reservoir i, Λ^f_{ui} [Λ^f_{di}] are the set of all links with their upstream [downstream] node being the reservoir $i = 1, \ldots, n_r$.

The third type of network equation is associated with the mass conservation at the variable-head nodes, where, as there is no accumulation of mass within the node the net inflow of fluid is equal to the mass outflow. That is

$$\sum_{j \in \Lambda_{di}} Q_j - \sum_{j \in \Lambda_{ui}} Q_j = d_{mi} \tag{4}$$

where d_{mi} is the nodal demand at node $i = 1, \ldots, n_j$.

From (2)-(4), it can be observed that the complete state-space for the network is the vector of nodal heads $\boldsymbol{h} = [h_1 \cdots h_{n_j}]^T$ and the vector of link flows $\boldsymbol{q} = [Q_1 \cdots Q_{n_p}]^T$. Given these state variables, a matrix representation of (1)-(4) is (Todini and Pilati 1988)

$$\begin{pmatrix} \boldsymbol{G} & -\boldsymbol{A}_1 \\ -\boldsymbol{A}_1^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{h} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_2 \boldsymbol{e}_l \\ \boldsymbol{d}_m \end{pmatrix} \tag{5}$$

where $\boldsymbol{e}_l = [e_{l1} \cdots e_{ln_r}]^T$, $\boldsymbol{d}_m = [d_{m1} \cdots d_{mn_j}]^T$, $\boldsymbol{G} = \boldsymbol{G}(\boldsymbol{q})$ is a diagonal $n_p \times n_p$ matrix function with diagonal elements $[\boldsymbol{G}]_{jj} = r_j|Q_j|$, and $\boldsymbol{A}_1$ $(n_p \times n_j)$ and $\boldsymbol{A}_2$ $(n_p \times n_f)$ are topological matrices given by

$$[\boldsymbol{A}_1]_{ji} = \begin{cases} 1 & \text{if } j \in \Lambda_{ui} \\ -1 & \text{if } j \in \Lambda_{di} \\ 0 & \text{otherwise} \end{cases}, \quad [\boldsymbol{A}_2]_{ji} = \begin{cases} 1 & \text{if } j \in \Lambda_{ui}^f \\ -1 & \text{if } j \in \Lambda_{di}^f \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $\boldsymbol{A}_1$ is associated with the connectivity of the links to the variable-head nodes, and $\boldsymbol{A}_2$ is associated with the connectivity of the links to the fixed-head nodes (note that for both matrices in (6), the first case corresponds to the upstream node of a link, the second case corresponds to the downstream node of a link, and the third case corresponds to any other node that a link is not incident to). The first matrix equation in (5) is nonlinear, of size n_p, and is associated with the link headlosses, and the second matrix equation in (5) is linear, of size n_p, and is associated with the nodal continuity.

The Global Gradient Algorithm

By applying a standard Newton's method approach to solving (5), Todini and Pilati (1988) derived the following sequence of iterates for solving the link flows and nodal heads

$$\boldsymbol{h}^{[m+1]} = \boldsymbol{V}^{-1}\left[\boldsymbol{A}_1^T\boldsymbol{F}^{-1}\left(\left(\boldsymbol{G}-\boldsymbol{F}\right)\boldsymbol{q}^{[m]} - \boldsymbol{A}_2\boldsymbol{e}_l\right) - \boldsymbol{d}_m\right], \tag{7}$$

$$\boldsymbol{q}^{[m+1]} = \boldsymbol{q}^{[m]} + \boldsymbol{F}^{-1}\boldsymbol{A}_1\boldsymbol{h}^{[m+1]} - \boldsymbol{F}^{-1}\left(\boldsymbol{G}\boldsymbol{q}^{[m]} - \boldsymbol{A}_2\boldsymbol{e}_l\right) \tag{8}$$

which requires an arbitrary initial point $\boldsymbol{q}^{[0]} = \boldsymbol{q}_0$ to commence the iterative solution process, where matrix functions $\boldsymbol{G}$, $\boldsymbol{F}$ $(n_p \times n_p)$, and $\boldsymbol{V}$ $(n_j \times n_j)$ are evaluated at $\boldsymbol{q} = \boldsymbol{q}^{[m]}$ with

$$\boldsymbol{F} = \mathrm{diag}\left[\frac{\mathrm{d}\mathcal{R}_1}{\mathrm{d}Q_1} \cdots \frac{\mathrm{d}\mathcal{R}_{n_p}}{\mathrm{d}Q_{n_p}}\right], \tag{9}$$

$$\boldsymbol{V} = \boldsymbol{A}_1^T\boldsymbol{F}^{-1}\boldsymbol{A}_1. \tag{10}$$

The major component of the computational effort required for the GGA is associated with the inversion of the negative Jacobian Schur complement $\boldsymbol{V}$ in (7). This is due to the fact that $\boldsymbol{G}$ and $\boldsymbol{F}$ are diagonal matrices meaning that all other operations in (7) are simply matrix-vector multiplications. The matrix $\boldsymbol{V}$ is $n_j \times n_j$ and for some practical applications n_j can be as large as $n_j \approx 10^5$ implying large computational times for computing (7). Rather than actually computing $\boldsymbol{V}^{-1}$, the approach adopted by most linear solvers for large systems is to solve directly for $\boldsymbol{h}$ in

$$\boldsymbol{V}\boldsymbol{h} = \boldsymbol{b} \tag{11}$$

where $\boldsymbol{b}$ is the term in the square brackets on the right hand side of (7). The solution of (11) is referred to as the *inner step* of the GGA, as it arises from the linear inner step within the original Newton process. The efficient solution of (11) is the focus of this paper.

THE ALGEBRAIC MULTIGRID METHOD

In many current applications there is an increasing demand for more efficient methods to solve large sparse and unstructured linear systems of equations. For linear systems of problem sizes relevant in practice, classical one-level iterative methods (*e.g.* Gauss-Seidel) have reached their limits. Fortunately, state-of-the-art hierarchical algorithms, such as AMG, allow an efficient solution of even larger problems.

The idea of hierarchical algorithms is to accelerate the convergence of the iterative solution of large sparse linear systems by creating a hierarchical process. From the original system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ defined by the pair $(\boldsymbol{A}, \boldsymbol{b})$, a sequence of lower dimensional (or coarser) systems $(\boldsymbol{A}_1, \boldsymbol{b}_1), \ldots, (\boldsymbol{A}_N, \boldsymbol{b}_N)$ are constructed, and are used to iteratively approximate the solution for the original system. In this process, AMG only directly solves the coarsest level system (the lowest dimensional level N system) and the solutions to the finer level systems are incrementally approximated from this solution.

The motivation for such methods arise from the inability of classical one-level iterative solvers to efficiently reduce the approximation error from iteration to iteration. Classical one-level iterative solvers often experience a slow convergence as they cannot handle all error frequencies effectively (Trottenberg et al. 2001). To be more specific, the high frequency components within the errors are dealt with much faster than the low frequency components, so by introducing a hierarchy of coarser levels (also termed multi-grids), each error frequency can be handled efficiently at an appropriate level.

In comparison to standard methods, AMG requires only $O(N)$ computational time to solve the discretized system up to given precision, where N denotes the dimensionality of the system (Trottenberg et al. 2001). AMG can be implemented as a *plug-in* solver, provided that the underlying matrix satisfies certain properties. Theoretically, AMG is applicable to M-matrices only, but in practice AMG works for many positive definite matrices (Stüben 2001b). Although the development of AMG goes back to the early eighties (Stüben 1983; Brandt et al. 1984), it still provides one of the most efficient, and notable robust, algebraic methods to solve elliptic problems (Stüben 2001a).

AMG can be seen as a defect-correction method. Broadly speaking, given the problem of computing $\boldsymbol{x}$ from $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, AMG starts with an approximate solution $\tilde{\boldsymbol{x}}$ and constructs a sequence of lower dimensional (coarser) systems to correct the approximation. The coarser levels are defined by the triples $(\boldsymbol{A}_1, \boldsymbol{b}_1, \tilde{\boldsymbol{x}}_1), \ldots, (\boldsymbol{A}_N, \boldsymbol{b}_N, \tilde{\boldsymbol{x}}_N)$ where $\boldsymbol{A}_l$ and $\boldsymbol{b}_l$ are constructed through the process of *restriction*, and $\tilde{\boldsymbol{x}}_l$ is an approximate solution to $\boldsymbol{A}_l \Delta\boldsymbol{x}_l = \boldsymbol{b}_l$. Each consecutive system $(l + 1)$ is associated with the error residual on the previous level l. At the coarsest level N, $\boldsymbol{x}_N$ is solved directly from $\boldsymbol{A}_N\boldsymbol{x}_N = \boldsymbol{b}_N$. This solution is then used to compute the sequence of corrections $\Delta\boldsymbol{x}_{N-1}, \ldots, \Delta\boldsymbol{x}_1$ from the coarse level $N - 1$ to the finest level 1 through the process of *interpolation*, where the objective is to achieve a reduction in the error residual, namely $||\boldsymbol{A}_l\boldsymbol{x}_l - \boldsymbol{b}_l|| < ||\boldsymbol{A}_l\tilde{\boldsymbol{x}}_l - \boldsymbol{b}_l||$ where $\boldsymbol{x}_l = \tilde{\boldsymbol{x}}_l + \Delta\boldsymbol{x}_l$. The computational advantage of this process is that only the coarsest system requires a direct solution, and the operations at all the other levels are simply matrix-vector multiplications.

This process is referred to as a V-cycle as the restriction phase follows the downward arc reducing the dimensionality of the problem, and the interpolation phase follows the upward arc using the information gained at the coarser level systems to refine the solution for the higher dimensional systems. A number of V-cycles may be performed within any AMG solution process until an adequately small residual $||\boldsymbol{Ax} - \boldsymbol{b}||$ is achieved. This process is discussed in greater depth below, but firstly some important preliminary concepts are outlined.

Preliminary Concepts

Coarse variable selection

The construction of each coarser level system within AMG involves the selection of the variables to be carried over. In other words, in each step l, the variable set $\mathcal{N}$ is partitioned into the coarse variables $\mathcal{N}_c$ and the fine variables $\mathcal{N}_f$. This process is termed C/F-splitting (Stüben 2001b). For the next step (step $l+1$), the coarse level set $\mathcal{N}_c$ becomes the new variable set $\mathcal{N}$, and the process is continued.

The selection of the $\mathcal{N}_c$ variables from the variable set $\mathcal{N}$ is based entirely on the terms of the system matrix within the current step $\boldsymbol{A}_l$. The splitting process utilizes strong negative couplings (n-couplings) between variables, where variable i is considered strongly n-coupled to variable j if the term $-a_{i,j}$ is large with respect to the other terms within the $i-$th row. The reason why strong n-couplings are important is that they indicate high correlations between variables within the solution process.

The objective in the C/F-splitting (to obtain $\mathcal{N}_c$) is to select a *minimally sized* set of variables that is *maximally n-coupled* to all the variables in $\mathcal{N}_f$. A variable set $\mathcal{N}_c$ defined as such can be understood as the smallest set of variables that is most representative of the entire variable set $\mathcal{N}$. There exist many different algorithms to undertake the C/F-splitting, and the interested reader is referred to (Stüben 2001b).

The restriction and interpolation operators

The operations of restriction and interpolation are used to transfer information between consecutive levels within the AMG hierarchy. These operators are dependent on the C/F-splitting at each level. Consider the consecutive systems $(\boldsymbol{A}_l, \boldsymbol{b}_l, \tilde{\boldsymbol{x}}_l)$ and $(\boldsymbol{A}_{l+1}, \boldsymbol{b}_{l+1}, \tilde{\boldsymbol{x}}_{l+1})$ as defined previously. The interactions between these levels consist of (i) the construction of the coarser $l+1$-th level system from the restriction of the finer l-th level system (the downward arc of the V-cycle), and (ii) the improvement of the l-th level solution from the interpolation of the $l+1$-th level solution (the upward arc of the V-cycle). To explain this further, it is more instructive to consider interpolation. Given an initial approximate solution to the l-th level system $\tilde{\boldsymbol{x}}_l$, an improvement to this approximation is obtained from the l-th level as $\tilde{\boldsymbol{x}}_l \leftarrow \tilde{\boldsymbol{x}}_l + \Delta\boldsymbol{x}_l$ where

$$\Delta\boldsymbol{x}_l = \boldsymbol{P}_l\tilde{\boldsymbol{x}}_{l+1} \tag{12}$$

where $\tilde{\boldsymbol{x}}_{l+1}$ is the solution to the $(l+1)$-th level system and $\boldsymbol{P}_l$ is the $n \times n_c$ interpolation matrix where n is the number of l-th level variables $\mathcal{N}$ and n_c is the number of coarse level variables $\mathcal{N}_c$ (*i.e.* the $(l+1)$-th level variables). So from (12) it is seen that the interpolation operator serves to interpolate the correction to the higher dimensional (finer level) solution from the lower dimensional (coarser level) solution. Ordering $\Delta\boldsymbol{x}_l$ so that the n_c coarser level variables are at the top of the vector, $\boldsymbol{P}_l$ can be partitioned as

$$\boldsymbol{P}_l = \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{W} \end{pmatrix} \tag{13}$$

where $\boldsymbol{I}$ is the $n_c \times n_c$ identity matrix, and $\boldsymbol{W}$ is a $n_f \times n_c$ weighting matrix possessing the interpolation coefficients used to compute the n_f finer level variables $\mathcal{N}_f$ form the coarser level variables $\mathcal{N}_c$. Different methods exist for constructing the weighting matrix $\boldsymbol{W}$, and the interested reader is referred to (Stüben 2001b).

In contrast to the interpolation operator, the restriction operator $\boldsymbol{R}_l$ is an $n_c \times n$ matrix that is used to construct the coarser level terms $\boldsymbol{A}_{l+1}$ and $\boldsymbol{b}_{l+1}$ from the finer l-level terms. Typically, Galerkin's principle is used, that is $\boldsymbol{R}_l = \boldsymbol{P}_l^T$. The nature of this construction is discussed later, but it is summarized in the AMG algorithm in Figure 1.

Smoothing

Geometrically speaking, an error $\boldsymbol{e} = \boldsymbol{x} - \tilde{\boldsymbol{x}}$ can be displayed as a linear combination of different (error) frequencies. The nature of classical one level iterative solvers like Gauss Seidel or Jacobi is that the high error frequencies vanish after a few iterations where as the low error frequencies need many iterations to vanish. AMG accelerates this process of broadband error reduction through its hierarchy of coarser level systems. That is, through the restriction process, the low frequency errors at level l become the high frequency errors at the coarser level $l+1$. Therefore, applying a few iterations of a one level iterative solver at each level resolves a broad band of error frequencies. This process of applying a few iterations of classical one level iterative solvers in the context of AMG methods is called *smoothing*, due to the fact that the high error frequencies are *smoothed* out.

Mathematically, given the system $(\boldsymbol{A}, \boldsymbol{b}, \tilde{\boldsymbol{x}})$ as defined above, the smoothing of a candidate solution $\tilde{\boldsymbol{x}}$ is given by

$$\overline{\boldsymbol{x}} = \boldsymbol{S}_1 \boldsymbol{A} \tilde{\boldsymbol{x}} + \boldsymbol{S}_2 \boldsymbol{b} \tag{14}$$

where $\overline{\boldsymbol{x}}$ is the smoothed candidate solution, and $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are real matrices, whose form depends on the type of the underlying one level iterative solver implemented (Saad 2003).

The AMG Algorithm

The details of the AMG algorithm as outlined in Figure 1 are now discussed. The input to the algorithm is the triple $(\boldsymbol{A}, \boldsymbol{b}, \tilde{\boldsymbol{x}}_{\text{initial}})$, where the objective is to determine a new approximation $\tilde{\boldsymbol{x}}$ such that $\|\boldsymbol{b} - \boldsymbol{A}\tilde{\boldsymbol{x}}_{\text{final}}\| < \epsilon$ where ϵ is the desired accuracy. The first phase (Figure 1, lines 2-7) is the setup phase, which involves the construction of all operators at each level. Based on the current matrix $\boldsymbol{A}_l$ and the C/F-splitting, the smoother, restriction, and interpolation operators are constructed (Figure 1, lines 3 and 4). The coarser stage $\boldsymbol{A}_{l+1}$ is then constructed according to the equation on line 5 in Figure 1. These steps are then repeated until the dimension of $\boldsymbol{A}_l$ is sufficiently small so as to solve the system directly (Figure 1, line 17). Typically, the coarsest level matrix lies in the range $O(10^2)$ to $O(10^3)$

variables.

After setting up all the required components, the solution phase starts (Figure 1, lines 9-23) where a new approximation $\boldsymbol{x}_1^k$ to the system $\boldsymbol{A}\boldsymbol{x}_1^k = \boldsymbol{b}$ is computed within each k-th V-cycle (Figure 1, lines 11-22). The V-cycles are applied until the desired accuracy is reached. The first loop in the solution phase (Figure 1, lines 11-16) represents the downward arc of an AMG V-cycle. In this loop, smoothing is applied to the candidate solution (Figure 1, line 12), after which the new defect $\tilde{\boldsymbol{b}}_l$ is computed (line 13). The smooth defect is restricted to the coarser grid (Figure 1, line 14). On the coarser grid, the correction equation is to be solved with a zero first guess (Figure 1, line 15). This process is iterated until the coarsest level $l = N$ is reached.

On the coarsest level the exact solution to the correction equation is computed via a direct solver (Figure 1, line 17).

Once the coarsest system is solved, the second loop (the interpolation phase) is performed (Figure 1, lines 19-23) which represents the upward arc of the V-cycle. The finer level correction is interpolated from the solution at the coarser level (Figure 1, line 19), which is used to update the candidate solution (line 20), which is then smoothed to remove the high frequency error components (Figure 1, line 21). This process is continued up until the finest level $l = 1$ is reached.

The entire solution phase loop (Figure 1, lines 9-23) is continued until an approximation $\tilde{\boldsymbol{x}}_l^k$ fulfils $\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_1^k\| < \epsilon$. Once the termination criteria on line 9 is reached, the final k-th cycle approximation is returned as $\tilde{\boldsymbol{x}}_{\text{final}}$ (Figure 1, line 24).

Accelerators for the AMG process

A well known approach to accelerate AMG (as well as one-level iterative solvers) is to use them as preconditioners for Krylov methods such as the conjugate gradient (CG) method (see e.g. Saad (2003)). Krylov methods are well known to accelerate iterative solvers in the case that the convergence of the solver is impeded due to eigenvalues that are not clustered within the circle including the majority of the systems eigenvalues. The resulting convergence of the overall preconditioned Krylov method is usually far better than the convergence of the stand-alone accelerators, multigrid methods or one-level iterative solvers.

THE APPLICATION OF AMG TO THE WDS SOLUTION

As discussed previously, the focus of this paper is the employment of AMG for the fast solution of the inner step (11) within the GGA. The importance of the fast solution of this step is that it represents the majority of the computational expense in the GGA. From a theoretical point of view, AMG is only guaranteed to converge for Stieltjes (a sub-class of M-matrices) matrices. Therefore, before employing AMG to solve (11), it is demonstrated that $\boldsymbol{V}$ is Stieltjes matrix.

This section discusses the issues associated with applying AMG to the solution of (11). First it is demonstrated that the $\boldsymbol{V}$ matrix from (10) is a Stieltjes matrix, and this is shown to hold for all cases where the friction factor models are consistent with the Colebrook-White formula. Second, a physical interpretation of the hierarchical AMG process is presented, and examples given.

Suitability of AMG for the Global Gradient Algorithm

Many of the theorems pertaining to the effectiveness of AMG have been so far proven only for systems involving Stieltjes matrices (Stüben 2001a) (however, in practice, this restriction can generally be relaxed to systems involving positive definite matrices). Todini and Pilati (1988) asserted that the $\boldsymbol{V}$ from (10) is a Stieltjes matrix, which is a symmetric sub-class of the M-matrix class of matrices. The implication of this is that AMG is ideally suited solving systems involving matrices of the form of $\boldsymbol{V}$. This statement was first proved by Piller (1995). An alternative theorem is offered below, where the conditions under which it holds are made explicitly dependent on the friction factor f and Reynolds number $\mathbb{R}_\mathrm{e}$.

Theorem 1: *The matrix* $\boldsymbol{V}$ *as defined in* (10) *is a Stieltjes matrix under the condition that the friction factor* f *and Reynolds number* $\mathbb{R}_\mathrm{e}$ *satisfy the inequality*

$$\frac{f}{\mathbb{R}_\mathrm{e}} + \frac{1}{2}\frac{\mathrm{d}f}{\mathrm{d}\mathbb{R}_\mathrm{e}} > 0 \tag{15}$$

for every pipe within the network.

A proof for this theorem is given in Appendix I. The condition (15) can be seen to hold for the case of laminar flow (where $f = 64/\mathbb{R}_\mathrm{e}$) and the case of fully rough turbulent flow (where $\mathrm{d}f/\mathrm{d}\mathbb{R}_\mathrm{e} = 0$). However, given the myriad formulate for the transitional and turbulent regions, this condition cannot be demonstrated to uniformly hold, but must be considered on a model-by-model basis. An important friction factor model is the Colebrook-White formula which is widely considered as the defining formula of f for transitional and turbulent flows with $\mathbb{R}_\mathrm{e} \geq 4000$, and is given by the implicit equation

$$\frac{1}{\sqrt{f}} = -2\log_{10}\theta, \quad \theta = \frac{\epsilon}{3.7D} + \frac{2.51}{\mathbb{R}_\mathrm{e}\sqrt{f}}. \tag{16}$$

The following theorem demonstrates that (16) satisfies condition (15).

Theorem 2: *For* $\mathbb{R}_\mathrm{e} \geq 4000$*, the Colebrook-White formula* (16) *for calculating* f *satisfies the Stieltjes condition* (15)*.*

A proof of this theorem is given in Appendix II. The importance of the Colebrook-White formula satisfying (15) is that most explicit models for computing f, within the transitional and turbulent region, are approximated from (16). Therefore, if they approximate (16) with sufficient accuracy, they too will satisfy the requirement guaranteeing that the $\boldsymbol{V}$ matrix will be of a Stieltjes type. An important example is the Swamee-Jain formula for the Darcy-Weisbach friction factor (Swamee and Jain 1976). It is not included here, but it can be demonstrated that the Swamee-Jain formula satisfies (15).

In conclusion, given Theorems 1 and 2, $\boldsymbol{V}$ from (10) is a Stieltjes matrix, which is consistent with the findings of (Piller 1995). Hence, AMG is guaranteed to converge if applied to (11).

Physical interpretation of the AMG process

An interpretation of the AMG process, based on the physical meaning of the $\boldsymbol{V}$ matrix, is outlined below. Firstly a reinterpretation of the $\boldsymbol{V}$ matrix is given, where it is seen to be a first order approximation to the networks admittance matrix. This is followed by a discussion of the AMG operations of coarsening and restriction where an example is given.

Interpretation of the $\boldsymbol{V}$ matrix

Within the GGA, the $\boldsymbol{V}$ matrix arises as the negative of the Schur complement to the Jacobian of the full nonlinear system of network equations (5) (Simpson and Elhay 2011; Elhay and Simpson 2011). However, this matrix can also be derived by an alternative means, which provides a physically based interpretation of the matrix. This derivation is outlined below.

For any pipe, the hydraulic *admittance* function is defined as the nonlinear map $\mathcal{Y} = \mathcal{R}^{-1}$ where $\mathcal{R}$ is the resistance function defined in (1) and the inverse refers the inverse map. Being the inverse to the hydraulic resistance, this map defines the steady-state flow rate through a pipe that is admitted from a given pressure difference across a pipe, that is $Q = \mathcal{Y}(\Delta h)$. For a network, the vector of link pressure drops is given by $\boldsymbol{A}_1\boldsymbol{h} + \boldsymbol{A}_2\boldsymbol{e}_l$, yielding the following expression for the network link flow rates

$$\boldsymbol{q} = \boldsymbol{\mathcal{Y}}\left(\boldsymbol{A}_1\boldsymbol{h} + \boldsymbol{A}_2\boldsymbol{e}_l\right)$$

where $\boldsymbol{\mathcal{Y}} = \text{diag}\left[\mathcal{Y}_1 \cdots \mathcal{Y}_{n_l}\right]$ (note that each $\mathcal{Y}_j$ is a nonlinear function of the headloss across the pipe, that is $\mathcal{Y}_j = \mathcal{Y}_j(h_i - h_k)$ where h_i and h_k are the upstream and downstream nodes of link j, respectively). Applying the network nodal mass conservation law (4), the nodal demands are obtained as

$$-\boldsymbol{d}_m\left(\boldsymbol{h}\right) = \boldsymbol{A}_1^T\boldsymbol{\mathcal{Y}}\left(\boldsymbol{A}_1\boldsymbol{h} + \boldsymbol{A}_2\boldsymbol{e}_l\right) \tag{17}$$

where the dependence of the nodal demands $\boldsymbol{d}_m$ on the nodal pressures is made explicit for the purposes of the following discussion. The map $\boldsymbol{A}_1^T\boldsymbol{\mathcal{Y}}$ in (17) holds the interpretation as the nonlinear network hydraulic admittance map as it maps from the network nodal pressures to the network nodal demands. Taking a Taylor series approximation of (17) about $\boldsymbol{h}_0$ yields

$$-\boldsymbol{d}_m\left(\boldsymbol{h}_0+\Delta\boldsymbol{h}\right)=\boldsymbol{A}_1^T\boldsymbol{\mathcal{Y}}+\boldsymbol{A}_1^T\boldsymbol{\mathcal{Y}}^{(1)}\boldsymbol{A}_1\Delta\boldsymbol{h}+\frac{1}{2}\boldsymbol{A}_1^T\boldsymbol{\mathcal{Y}}^{(2)}\left[\boldsymbol{A}_1\Delta\boldsymbol{h}\right]\circ\left[\boldsymbol{A}_1\Delta\boldsymbol{h}\right]+\cdots \tag{18}$$

where $\boldsymbol{\mathcal{Y}}^{(n)} = \operatorname{diag}\left[\mathcal{Y}_1^{(n)}\cdots\mathcal{Y}_{n_l}^{(n)}\right]$ is a diagonal matrix of the n-th derivatives of the admittance functions, and $\circ$ denotes the Hadamard product, where all $\boldsymbol{\mathcal{Y}}$, $\boldsymbol{\mathcal{Y}}^{(1)}$, and $\boldsymbol{\mathcal{Y}}^{(2)}$ are evaluated at $\boldsymbol{A}_1\boldsymbol{h}_0+\boldsymbol{A}_2\boldsymbol{e}_l$. Under the assumption of nonzero first order derivatives, the following reciprocity principal holds

$$\mathcal{Y}_j^{(1)}=\frac{\mathrm{d}\mathcal{Y}_j}{\mathrm{d}\Delta h}=\left(\frac{\mathrm{d}\mathcal{R}_j}{\mathrm{d}Q}\right)^{-1}=F_{jj}^{-1}$$

which means that $\boldsymbol{F}^{-1}=\boldsymbol{\mathcal{Y}}^{(1)}$, which, in comparison to $\boldsymbol{V}$ from (10) in the GGA, leads to the recognition that

$$\boldsymbol{V}=\boldsymbol{A}_1^T\boldsymbol{\mathcal{Y}}^{(1)}\boldsymbol{A}_1, \tag{19}$$

that is, $\boldsymbol{V}$ is actually the first order term of a Taylor series expansion (18). The physical interpretation of this is that the matrix $\boldsymbol{V}$ is, in fact, a first order approximation to the network hydraulic admittance map. That is, the (i, j) element of $\boldsymbol{V}$ is an admittance scaling coefficient indicating the contribution that the pressure at node j makes to the demand at node i. This issue was explored also in Piller (1995).

Network admittance matrices of a similar form to (19) are found in many other engineering disciplines for other systems, examples of which are node-based descriptions of electrical circuit dynamics (Chen 1983), and Laplace-domain representations of transient-state fluid line networks (Zecchin et al. 2009). Indeed, the connection between these networks is their adherence to the Kirchoff network laws that govern the interactions at the nodal points, and relate the link-based relationships to properties held by the wider network.

Interpretation of the coarse variable selection process

A cycle of AMG involves partitioning the variable set into coarse and fine level variables. Coarse level variables are carried over into the construction of the restricted system. Fine level variables only exist in the non-restricted system. Partitioning of the nodal set $\mathcal{N}$ is dependent on the relative value of the elements of $\boldsymbol{V}$. The approximate aim of this is to determine a minimal subset of nodes $\mathcal{N}_c \subset \mathcal{N}$ such that they are maximally connected to the remaining finer level variables $\mathcal{N}_f = \mathcal{N}/\mathcal{N}_c$. In this way, the coarse variables $\mathcal{N}_c$ are in some sense the smallest set of variables with which to interpolate the set of finer level variables $\mathcal{N}_f$.

Within the AMG framework, variable k is considered strongly connected (or n-coupled) to variable i if the (i, j) entry within the matrix $\boldsymbol{V}$ is of a relatively large magnitude. Given the interpretation of $\boldsymbol{V}$ as outlined in the previous section, k is strongly connected to i if the derivative of the admittance value $\mathcal{Y}_j^{(1)}$ is large for link j connecting node i to node k. High values of $\mathcal{Y}_j^{(1)}$ correspond to pipes for which a small change in the pressure difference $h_i - h_k$ induces a large change in the flow rate. Such pipes possess small frictional energy losses (*i.e.* large diameter, small roughness, low flow pipes). Therefore, two nodes i and k are considered strongly connected if the headloss between them is small, that is, if the nodal heads at either end of the pipe are close in value.

An illustrative example of the selected coarse level nodes for a small 35-pipe/20-node network is depicted in Figure 2(a), where the coarse level nodes are indicated by larger bold circles (the network parametric details are given in Zecchin (2010)). For this example, the *standard coarsening* algorithm was used, for which the interested reader is referred to (Stüben 2001a). Of the 19 variable head nodes within the network, the C/F splitting resulted in the five coarse level nodes $\mathcal{N}_c = \{2, 5, 8, 11, 15, 20\}$. From this diagram, it is clear that all the 14 fine level nodes are connected to at least one coarse level node, and that no new coarse level node can be defined without introducing connections between coarse level nodes.

Interpretation of the restriction/interpolation process

The dual restriction and interpolation processes are the backbone of AMG. Restriction provides a way of constructing a smaller dimensional system $\boldsymbol{V}_1$ that is, in some sense, approximately representative of the original higher dimensional system $\boldsymbol{V}$. Interpolation provides a way of mapping from the solution of the restricted (smaller dimensional) system to a correction for a candidate solution of the higher dimensional system. This section explores the structure of a single step restricted system $\boldsymbol{V}_1$ and its topological relationship to the original system $\boldsymbol{V}$.

Consider a network with node set $\mathcal{N}$ and link set Λ. Partitioning $\mathcal{N}$ into coarse level nodes $\mathcal{N}_c$ and fine level nodes $\mathcal{N}_f$ leads to a corresponding partitioning of Λ into three disjoint subsets: Λ_{cc} the set of links connecting the coarse level nodes; Λ_{cf} the set of links connecting the coarse level to the fine level nodes; and Λ_{ff} the set of links connecting the fine level nodes (an example of this, discussed later, is depicted in Figure 2). Ordering the links according to these sets, the incidence matrix can be partitioned as

$$\boldsymbol{A}_1 = \begin{pmatrix} \boldsymbol{A}_{cc} & \mathbf{0} \\ \boldsymbol{A}_{cf} & \boldsymbol{A}_{fc} \\ \mathbf{0} & \boldsymbol{A}_{ff} \end{pmatrix},$$

where $\boldsymbol{A}_{cc}$ is the incidence matrix associated with the coarse level nodes and the Λ_{cc} links, $\boldsymbol{A}_{cf}$ is associated with the coarse level nodes and the Λ_{cf} links, $\boldsymbol{A}_{fc}$ is associated with the fine level nodes and the Λ_{cf} links, and $\boldsymbol{A}_{ff}$ is associated with the fine level nodes and the Λ_{ff} links. Similarly, the $\boldsymbol{\mathcal{Y}}$ matrix can be partitioned as

$$\boldsymbol{\mathcal{Y}} = \begin{pmatrix} \boldsymbol{\mathcal{Y}}_{cc} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\mathcal{Y}}_{cf} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\mathcal{Y}}_{ff} \end{pmatrix}$$

where $\boldsymbol{\mathcal{Y}}_{cc}$, $\boldsymbol{\mathcal{Y}}_{cf}$, and $\boldsymbol{\mathcal{Y}}_{ff}$ are the matrices of link admittance functions for the Λ_{cc}, Λ_{cf}, and

Λ_{ff} links respectively. Multiplying these block matrix representations leads to the following expression of $\boldsymbol{V}$

$$\boldsymbol{V} = \begin{pmatrix} \boldsymbol{V}_{cc} & \boldsymbol{V}_{cf} \\ \boldsymbol{V}_{fc} & \boldsymbol{V}_{ff} \end{pmatrix}$$

where $\boldsymbol{V}_{xy}$ holds the interpretation of the first order admittance map from the $\mathcal{N}_y$ nodal pressures to the $\mathcal{N}_x$ nodal demands, where these matrices are given by $\boldsymbol{V}_{cc} = \boldsymbol{A}_{cc}^T \boldsymbol{\mathcal{Y}}_{cc}^{(1)} \boldsymbol{A}_{cc} + \boldsymbol{A}_{cf}^T \boldsymbol{\mathcal{Y}}_{cf}^{(1)} \boldsymbol{A}_{cf}$, $\boldsymbol{V}_{cf} = \boldsymbol{A}_{cf}^T \boldsymbol{\mathcal{Y}}_{cf}^{(1)} \boldsymbol{A}_{fc}$, $\boldsymbol{V}_{ff} = \boldsymbol{A}_{ff}^T \boldsymbol{\mathcal{Y}}_{ff}^{(1)} \boldsymbol{A}_{ff} + \boldsymbol{A}_{fc}^T \boldsymbol{\mathcal{Y}}_{cf}^{(1)} \boldsymbol{A}_{fc}$, and $\boldsymbol{V}_{fc} = \boldsymbol{V}_{cf}^T$.

For such an ordering of the coarse and fine level variables, the restriction operator can be partitioned as $\boldsymbol{R} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{W}^T \end{pmatrix}$ where $\boldsymbol{I}$ is an $n_c \times n_c$ identity matrix, and $\boldsymbol{W}$ is a $n_c \times n_f$ matrix of the interpolating weights. The single step restricted system, given by $\boldsymbol{V}_1 = \boldsymbol{R}\boldsymbol{V}\boldsymbol{R}^T$, can be expressed as $\boldsymbol{V}_1 = \boldsymbol{V}_{cc} + \boldsymbol{V}_{(1)} + \boldsymbol{V}_{(2)}$ where

$$\boldsymbol{V}_{(1)} = \boldsymbol{W}^T \boldsymbol{V}_{fc} + \boldsymbol{V}_{cf} \boldsymbol{W}, \quad \boldsymbol{V}_{(2)} = \boldsymbol{W}^T \boldsymbol{V}_{ff} \boldsymbol{W} \tag{20}$$

The matrix $\boldsymbol{V}_{(1)}$ introduces terms in $\boldsymbol{V}_1$ that are associated with coarse level variables that are coincidently connected to the same finer level nodes. That is element (i, j) in $\boldsymbol{V}_{(1)}$ possesses a nonzero term if coarse level nodes i and j are connected to the same fine level nodes. The matrix $\boldsymbol{V}_{(2)}$ introduces terms in $\boldsymbol{V}_1$ that are associated with connections between finer level nodes. That is, element (i, j) in $\boldsymbol{V}_{(2)}$ possesses a nonzero term if any of the fine level nodes connected to coarse level node i are connected to any of the fine level nodes connected to coarse level node j.

To explain this further, expanding out the terms in (20) leads to the realization that the interpolation matrix $\boldsymbol{W}$ appears as a post multiplier to the incidence matrices, that is it features as $\boldsymbol{A}_{fy}\boldsymbol{W}$ where $y = f$ or c. The post multiplication of the incidence matrix by $\boldsymbol{W}$ acts to compresses the columns of $\boldsymbol{A}_{fy}$ from the n_f columns (each associated with a fine level variable) to n_c columns (each associated with a coarse level variable). In this way, the column associated with the i-th fine level variable is distributed amongst the n_c coarse level variable columns according to the interpolation weights from the interpolation matrix. As a result, the columns of $\boldsymbol{A}_{fy}\boldsymbol{W}$ are a weighted sum of the columns of $\boldsymbol{A}_{fy}$, that is

$$[\boldsymbol{A}_{fy}\boldsymbol{W}]_{\bullet,i} = \sum_{k=1}^{n_f} w_{k,i} \, [\boldsymbol{A}_{fy}]_{\bullet,k} \, ,$$

where $w_{k,i}$ is the interpolation weight of the i-th coarse level variable for the k-th fine level variable.

From this representation of the restricted system, it is seen that the restriction operation serves to create an admittance matrix for a new network that is actually the superposition of three separate networks: (i) the links from the original network connecting the coarse level nodes; (ii) links involving connections between coarse level nodes coincidently connected to the same finer level nodes; and (iii) links based on connections between fine level nodes.

An example of the link sets Λ_{cc}, Λ_{cf} and Λ_{ff} and the associated networks is given in Figure 2. The links in Λ_{cf} are depicted as bold lines and the links in Λ_{ff} as dashed lines. For this network, an interesting outcome is that $\Lambda_{cc} = \emptyset$, meaning that $\boldsymbol{V}_{cc}$ contains only the diagonal terms corresponding to the Λ_{cf} links. This outcome results from the coarse node selection, where the objective of this process is to partition the node set $\mathcal{N}$ into disjoint sets $\mathcal{N}_c$ and $\mathcal{N}_f$, where $\mathcal{N}_c$ is a the minimum set that is maximally connected to $\mathcal{N}_f$. A byproduct of this process is that the nodes in $\mathcal{N}_c$ are typically not connected to each other.

Given the coarse node partitioning in Figure 2(a), the topology of the coarse level networks associated with the admittance-type matrices $\boldsymbol{V}_{(1)}$ and $\boldsymbol{V}_{(2)}$ are given in Figures 2(b) and 2(c), respectively, where the new links from Λ_{cf} and Λ_{ff} connecting the $\mathcal{N}_c$ variables are listed on the links. The resultant coarse level network associated with $\boldsymbol{V}_1$ is simply the superposition of these two networks.

NUMERICAL STUDY

An extensive series of numerical experiments was undertaken in order to test the utility of AMG for the fast solution of the linear inner step of the GGA (*i.e.* the solution of (11)).

These experiments involved the comparison of the computational time required by two AMG variants against ILU (a standard iterative solver for sparse linear systems of large size), PARDISO (a fast and robust direct sparse linear solver) and the direct Cholesky solver from EPANET2. For these experiments, the performance of these algorithms in solving systems with $\boldsymbol{V}$ matrices from 10 Newton iterations of 10,000 randomly generated networks were analysed. The network sizes ranged from 10^3 up to $10^{5.75}$ nodes, and $10^{3.5}$ to $10^{6.4}$ links.

Preliminaries

Linear solver algorithms

For the purposes of comparison, two AMG variants where tested, namely (i) standard AMG, and (ii) AMG preconditioned conjugate gradient (AMG+CG). These variants were compared to the EPANET2 solver SC+NR, another standard sparse linear solver ILU preconditioned conjugate gradient method (ILU+CG), and PARDISO, all of which are outlined below.

1. *AMG.* The variable-based algebraic multi-grid (VAMG) variant (Stüben 2001b) was used in the numerical experiments with *standard* coarsening and interpolation operators, Gauss-Seidel relaxation for smoothing, and a sparse Gauss-Seidel solver to solve the coarsest level system.
2. *AMG+CG.* This method involved the VAMG, as described above, as a preconditioner for the CG method.
3. *SC+NR.* This method is adopted by the commonly used hydraulic simulation software EPANET2 (Rossman 2000). It involves a node reordering process coupled with a sparse Cholesky solver. The routines from EPANET2 were directly imported and included as a dynamically linked library to our software.
4. *ILU+CG.* This method involved the use of ILU as a preconditioner for the CG method. ILU is the incomplete LU factorization method for iteratively solving linear systems, where a detailed description can be found in Saad (2003).

5. *PARDISO*. A widely recommended fast and reliable direct solver (Gould et al. 2007), PARDISO adopts a combination of Level 3 BLAS supernode techniques, with a LU, LDL or a LL^T factorization (Schenk et al. 1999). PARDISO is included in the Intel Math Kernal Library.

Algorithms AMG, AMG+CG, and ILU+CG are contained in the linear solver library SAMG provided by The Fraunhofer Institute for Algorithms and Scientific Computing (SCAI), Germany.

Outline of Experiments

For the experiments, 10,000 networks were analyzed, involving 100 different network configurations at 100 different network sizes with between $O(10^{2.7})$ to $O(10^{5.75})$ nodes. The networks were randomly generated within a rectangular grid pattern consisting of $n_x \times n_y$ nodes, where 1% of nodes were randomly selected to be reservoirs. The grids were constructed by first randomly selecting n_x, and then calculating n_y based on achieving an overall network size. The network parameters of each reservoir and pipe were independently sampled from uniform distributions as follows: reservoir elevations $\sim \mathcal{U}[120, 140]$ m; nodal demands $\sim \mathcal{U}[0, 10]$ L/s; pipe lengths $\sim \mathcal{U}[100, 1100]$ m; pipe diameters $\sim \mathcal{U}[100, 300]$ mm (where $x \sim \mathcal{U}[a, b]$ symbolizes a random variable x uniformly distributed on the interval $[a, b]$). All pipe roughness heights were set to 0.3 mm.

The convergence condition for the different iterative linear solvers was based on the l_2 norm of the residual $||\boldsymbol{b} - \boldsymbol{V}\boldsymbol{h}||_2$. The numerical experiments were conducted for two different tolerance values of this norm, namely $||\boldsymbol{b} - \boldsymbol{V}\boldsymbol{h}||_2 = 10^{-2}$ simulating a low accuracy convergence criteria, and $||\boldsymbol{b} - \boldsymbol{V}\boldsymbol{h}||_2 = 10^{-6}$ simulating a higher accuracy convergence criteria (typical of many applications).

For each of the 10^4 networks, exactly 10 Newton iterations were performed, meaning a total of 10^5 different $\boldsymbol{V}$ matrices were tested. As the analysis within this paper is focused on the *inner* linear iterations involving equations of the form (11), it was not necessary to reach convergence with the *outer* Newton iterations. Due to the additional overhead associated with the node reordering routines in SC+NR as adopted from EPANET2, this procedure was only performed once within the first Newton iteration, and the reordering structure was retained and reused for the consequent Newton iterations for each network. Consequently, the results presented for SC+NR distribute the total setup time of the reordering routines equally over the 10 iterations. The numerical experiments were performed on a 64-bit 2.6 GHz Linux machine, where the `procstat` routine was used to determine the CPU time for each computation.

Results and Discussion

The results of the numerical experiments are summarized in Figures 3, 4, and 5. Figure 3 presents statistics of the computational times, where the subfigure rows (1) and (2) correspond to the experiments for the tolerances 10^{-2} and 10^{-6}, respectively, and the subfigures (a, 1 and 2), (b, 1 and 2), (c, 1 and 2), (d) and (e) correspond to the algorithms AMG, AMG+CG, ILU+CG, PARDISO and SC+NR respectively. Within these plots, the upper and lower dotted lines correspond to the maximum and minimum computational times required within a network group of the same node size (within each group, 1000 different $\boldsymbol{V}$ were used), and the bolded line corresponds to the mean computational time. For the smaller network sizes, the individual computational times were occasionally too small to be measured by the `procstat` routine, hence the minimum is not observed here. In a similar organization to Figure 3, Figure 4 gives the median of the number of cycles of each of the iterative algorithms, where a cycle is defined as a single iteration of the algorithm.

Figure 5 gives a direct comparison between the average computational times for the AMG variants, ILU+CG, PARDISO and SC+NR in both logarithmic and linear computational time. The averaged computational times for each algorithm are computed by first averaging the computational times for each set of networks with equivalent nodal sizes, and secondly applying an 11-point smoother to the averages of the network nodal groupings smoother to the resultant data series.

As observed in Figure 3, the general trend for AMG, AMG+CG, and PARDISO is that the mean is relatively close to both the minimum and maximum times, indicating that the computational times were in a relatively small band about the mean (with the exception of isolated cases for smaller networks for the pure AMG). ILU+CG demonstrated a greater variability in the computational times than most other algorithms. Similarly, SC+NR also exhibited a larger variability in computational times, with a significant skewness towards the longer times. It is clear from Figure 3 that the computational time for SC+NR increased at a significantly greater rate than all other algorithms for an increasing network size. As such, the simulations for SC+NR were only undertaken up to a network size of $10^{4.3}$ nodes. In considering the iterative solvers, decreasing the tolerance from 10^{-2} to 10^{-6} resulted in a significant increase in the computational time for ILU+CG in comparison to AMG and AMG+CG.

To further explore the performance of the iterative solvers, the number of computational cycles used by each iterative solver is given in Figure 4. This figure demonstrates the linear complexity of the AMG variants investigated. Clearly, the number of iterations does not rise considerably despite the increasing sizes of the linear systems. However, the AMG (without CG acceleration) demonstrated a slightly less stable behaviour than the AMG+CG, since the cycle number fluctuated for different linear systems. In contrast, the ILU+CG method, demonstrated a typical performance for these kind of problems, namely, the number of iterations was heavily dependent on the size of the linear system. By implication, ILU+CG's behavior strongly suggests not only a dependence on the network size but also a dependence on the specific matrix entries, due to the high bandwidth between maximum and minimum computational times. That is, ILU+CG demonstrated an unpredictable and unstable behavior. SC+NR's computational times also show a high bandwidth between maximum and minimum computational times which is caused by different sparsity patterns within the $\boldsymbol{V}$ matrix.

For some smaller network sizes, AMG experienced large computational times, due to the different network properties. Specifically, in a few cases, no hierarchy was created due the fact that the system matrix was strongly diagonal dominant and small in size. If no hierarchy is created the resulting method is an ordinary Gauss-Seidel iterative solver. As no acceleration was used within the AMG, this resulted in long runtimes. This behaviour, indicates that AMG+CG is the most stable method considered as it exhibited stable and consistently low computational statistics for each network.

Considering the low accuracy convergence criteria (convergence tolerance of 10^{-2}), no significant difference was observed between the iterative algorithms average performance as observed in Figures 5(a) and (c), with ILU+CG performing just moderately faster. In comparison, PARDISO's and SC+NR's computational times were more than 4 times longer than the iterative solvers. This is to be expected as both methods are direct solvers which are not controlled by a convergence tolerance. Additionally, an important point to note is that, being a direct solver, PARDISO and SC+NR typically used between 1.5 to three times the memory of the iterative solvers. From these plots it is also clear that the computational performance of SC+NR significantly deteriorated for networks greater than 10^4 nodes in size. For the larger networks simulated by SC+NR (10^4 to $10^{4.3}$ nodes), AMG+CG was approximately 25 times faster than SC+NR, the reason being the nonlinear complexity of the reordering and fill-in of SC+NR.

To further understand the unexpected results of SC+NR, Figure 6 shows the computational statistics (minimum, mean and maximum computational times) divided into the time used by the sparse Cholesky (SC) solver, and the time used by the node reordering (NR) routines. It is observed that for network sizes with $n_j < 500$ there is no significant variation in either the SC or NR components, with both components having significantly low computational times. As n_j is increased, the SC component experiences a gradual increase in computational time. In contrast, the NR experiences a significant increase in computational time, such that at $n_j \approx 10^{3.3}$, the computational time increased over an order of magnitude, and over three orders of magnitude for $n_j \approx 10^4$. Therefore, for larger networks, the vast majority of the time required by SC+NR is attributed to the the NR routines.

For the high accuracy convergence criteria (convergence tolerance of 10^{-6}), an entirely different relative behavior of the algorithms is observed. The increased computational time cost for the increase in accuracy is large for ILU+CG (approximately a 500% increase in computational time), and for all networks ILU+CG was slower than PARDISO. Despite of the longer computational times for the iterative solvers SC+NR still is far slower than the other methods. By comparison, the increase in computational time for an increase in accuracy for AMG+CG was relatively small (only an approximately 60% increase). The computational time of AMG+CG was similar to that of PARDISO for the small networks (*i.e.* $n_j \approx 500$). However, for the larger networks, AMG+CG achieved speeds of over three times faster than PARDISO for the larger network sizes (*i.e.* $n_j > 10^{4.7}$). The increasing computational efficiency of AMG for the large network sizes is consistent with the property of AMG approaching a linear complexity for large problem sizes in comparison to non-linear complexity of the other algorithms. The AMG+CG was consistently just marginally faster than the purely AMG, hence it is not depicted in Figure 5.

CONCLUSIONS

This paper explores the application of the algebraic multigrid (AMG) method to the fast computation of the linear step within the global gradient algorithm (GGA) (also known as the Todini and Pilati method), for the solution of the steady-state behaviour of water distribution systems. The linear system in the GGA was demonstrated to be of a Stieltjes' type, meaning that it is ideal for the application of AMG. Extensive numerical studies demonstrated that, for an accurate convergence criteria, the AMG performed consistently faster than the conjugate gradient preconditioned incomplete LU factorization (a commonly used sparse linear solver) and PARDISO (a fast direct sparse linear solver). It was observed that the relative computational speed of AMG was up to three times that of the other algorithms for larger networks with more than $10^{4.7}$ nodes. Additionally, AMG was also compared to the sparse Cholesky method with nodes reordering (SC+NR), the solver adopted within EPANET2. For systems with more than 10^4 nodes, AMG was observed to be approximately 25 times faster than SC+NR (the main computational cost of SC+NR was observed to be attributed to the node reordering routines). Such computational savings have important implications for not only large networks, but for computations involving repeated network evaluations, such as extended period simulations, or network design optimization. In summary, for large networks the authors suggest the use of AMG in combination with the conjugate gradient method (termed AMG preconditioned conjugate gradient (AMG+CG)) as it combines a stable performance together with low computational times.

ACKNOWLEDGEMENTS

This research has been financially supported by the *Australia-Germany Joint Research Co-operation Scheme* co-funded by the Australian Group of Eight and the German Academic Exchange Service (DAAD).

REFERENCES

Brandt, A., McCormick, S., and Ruge, J. (1984). "Algebraic Multigrid (AMG) for Sparse Matrix Equations." *Sparsity and its Applications*, 257–284. D. Evans, Ed. Cambridge University Press.

Chen, W.-K. (1983). *Linear Networks and Systems*. Brooks/Cole Engineering Division, Monterey, Calif.

Collins, M., Cooper, L., Helgason, R., Kennington, J., and LeBlanc, L. (1978). "Solving the pipe network analysis problem using optimization techniques." *Management Science*, 24(7), 747–760.

Cross, H. (1936). "Analysis of flow in networks of conduits or conductors." *Engineering Experimental Station Bulletin*, 286, 329.

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

Elhay, S. and Simpson, A. R. (2011). "Dealing with zero flows in solving the non-linear equations for water distribution systems." *Journal of Hydraulic Engineering, ASCE*, In press.

Gould, N. I. M., Scott, J. A., and Hu, Y. (2007). "A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations." *ACM Transactions on Mathematical Software*, 33(2).

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

Nielson, H. B. (1989). "Methods for analyzing pipe networks." *Journal of Hydraulic Engineering, ASCE*, 115(2), 139–157.

Piller, O. (1995). "Modeling the behavior of a network - hydraulic analysis and sampling procedures for parameter estimation," PhD, The University of Bordeaux I.

Rossman (2000). "Epanet 2 users manual." *Report No. EPA/600/R-00/057*, National Risk Management Research Laboratory Office of Research and Development U. S. Environmental Protection Agency. Includes some description of the underlying model.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. SIAM Society for Industrial and Applied Mathematics, 2nd edition.

Schenk, O., Gärtner, K., and Fichtner, W. (1999). "Efficient sparse LU factorization with left-right looking strategy on shared memory multiprocessors." *BIT Numerical Mathematics*, 40(1).

Simpson, A. R. and Elhay, S. (2011). "Jacobian matrix for solving water distribution system equations with the darcy-weisbach head-loss model." *Journal of Hydraulic Engineering, ASCE*, 137(6).

Streeter, V. L., Wylie, E. B., and Bedford, K. W. (1997). *Fluid Mechanics*. WCB/McGraw Hill, Boston, Mass., 9th edition.

Stüben, K. (1983). "Algebraic Multigrid (AMG): Experiences and Comparisons." *Appl.*

Math. Comput., 13, 419–452.

Stüben, K. (2001a). "A Review of Algebraic Multigrid. A short report on AMG and the basic solver technology." *Journal of Computational and Applied Mathematics*, 128, 281309.

Stüben, K. (2001b). "An introduction to algebraic multigrid." *Multigrid*, U. Trottenberg, C. Oosterlee, and A. Schüller, eds., Academic Press, London, 413–532.

Stüben, K., Delaney, P., and Chmakov, S. (2003). "Algebraic multigrid (AMG) for ground water flow and oil reservoir simulation." *Groundwater Modelling Conference MODFLOW and MORE*, Colorado School of Mines, Golden, Colorado.

Swamee, P. K. and Jain, A. K. (1976). "Explicit equations for pipe-flow problems." *Journal of the Hydraulics Division (ASCE)*, 102(5).

Todini, E. (2011). "Extending the global gradient algorithm to unsteady flow extended period simulations of water distribution systems." *Journal of Hydroinformatics*, 13(2).

Todini, E. and Pilati, S. (1988). "A gradient algorithm for the analysis of pipe networks." *Computer Applications in Water Supply*, B. Coulbeck and C. H. Orr, eds. Research Studies Press, Letchworth, Hertfordshire, UK, 1–20.

Trottenberg, U., Oosterlee, C., and Schüller, A. (2001). *Multigrid*. Academic Press, London.

Wu, Z. Y., Wang, R. H., Walski, T. M., Yang, S. Y., Bowdler, D., and Baggett, C. C. (2009). "Extended global-gradient algorithm for pressure-dependent water distribution analysis." *Journal of Water Resource Planning and Management, ASCE*, 135(1), 13–22.

Zecchin, A. C. (2010). "Laplace-domain analysis of fluid line networks with applications to time-domain simulation and system parameter identification," PhD, The School of Civil, Environmental and Mining Engineering, The University of Adelaide.

Zecchin, A. C., Simpson, A. R., Lambert, M. F., White, L. B., and Vitkovsky, J. P. (2009). "Transient modeling of arbitrary pipe networks by a Laplace-domain admittance matrix." *Journal of Engineering Mechanics, ASCE*, 135(6), 538–547.

APPENDIX I. PROOF OF THEOREM 1

A Stieltjes matrix is defined as a real symmetric positive definite matrix with non-positive off-diagonal entries. The proof of the theorem requires the demonstration of $\boldsymbol{V}$ holding these properties under the assumption that (15) holds. To demonstrate this, consider the elementwise expression of $\boldsymbol{V}$

$$[\boldsymbol{V}]_{ik} = \begin{cases} \sum_{j \in \Lambda_i} F_{jj}^{-1} & \text{if } k = i \\ -F_{jj}^{-1} & \text{if link } j \text{ connects nodes } i \text{ and } k \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

where $\Lambda_i = \Lambda_{ui} \cup \Lambda_{di}$. The matrix $\boldsymbol{V}$ is clearly symmetric, and its off-diagonal entries are non-positive under the condition that all F_{jj} are positive. Additionally, this condition was also required in Piller (1995) for the proof of the positive definiteness of $\boldsymbol{V}$. Consequently, $\boldsymbol{V}$ is Stieltjes if all F_{jj} are positive. The differential chain rule applied to (1) leads to

$$F_{jj} = \frac{\mathrm{d}\mathcal{R}_j}{\mathrm{d}Q_j} = \frac{16}{\pi^2 g} \frac{L_j}{D_j^5} \frac{|Q_j|}{\mathbb{R}_\mathrm{e}} \left[\mathbb{R}_\mathrm{e} f_j + \frac{\mathbb{R}_\mathrm{e}^2}{2} \frac{\mathrm{d}f_j}{\mathrm{d}\mathbb{R}_\mathrm{e}} \right].$$

Recognising that the term outside the parenthesis is unconditionally positive, the requirement of $F_{jj} > 0$ reduces to the condition (15). $\square$

APPENDIX II. PROOF OF THEOREM 2

Theorem 2 is demonstrated to hold by determining a lower bound on $\mathrm{d}f/\mathrm{d}\mathbb{R}_\mathrm{e}$. From the Colebrook-White formula (16), the gradient of the friction factor can be determined as

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbb{R}_\mathrm{e}} = \frac{f}{\mathbb{R}_\mathrm{e}}\frac{1}{\log\theta}\left[1+\sqrt{f}\left(\frac{1}{\log 10}+\frac{\mathbb{R}_\mathrm{e}}{5.02}\frac{\epsilon/D}{3.7}\right)\right]^{-1}. \tag{22}$$

It holds that the term in the square brackets has a lower bound of 1 implying that

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbb{R}_\mathrm{e}} > \frac{f}{\mathbb{R}_\mathrm{e}}\frac{1}{\log\theta}.$$

This inequality leads to the requirement that $\log^{-1}\theta > -2$ for the satisfaction of (15), which implies the upper bound on θ of $\theta < 1/\sqrt{e}$ where e is Eulers coefficient. As $0 \leq \epsilon/D < 0.5$, this is satisfied for all $\mathbb{R}_\mathrm{e} \geq 4000$. $\square$

APPENDIX III. MANUSCRIPT FIGURES

FIG. 1. Algorithm outline for the algebraic multi-grid method for solving $Ax = b$.

Require: System parameters $\boldsymbol{A}$, and $\boldsymbol{b}$, and current approximation $\tilde{\boldsymbol{x}}_{\text{initial}}$

Set initial variables: $\boldsymbol{A}_1 = \boldsymbol{A}$; $\boldsymbol{b}_1 = \boldsymbol{b}$; and $\tilde{\boldsymbol{x}}_1^0 = \tilde{\boldsymbol{x}}_{\text{initial}}$, level $l = 1$
while $\dim \boldsymbol{A}_l$ is large **do** {setup-phase}
    Construct the level l smoothing operator: $\boldsymbol{S}_l = \texttt{smoother}(\boldsymbol{A}_l)$
    Construct the level l restriction $\boldsymbol{R}_l$ and interpolation $\boldsymbol{P}_l$ operators: $\boldsymbol{R}_l = \texttt{restriction}(\boldsymbol{A}_l)$; and $\boldsymbol{P}_l = \texttt{interpolation}(\boldsymbol{A}_l)$
    Set coarser level matrix: $\boldsymbol{A}_{l+1} = \boldsymbol{R}_l \boldsymbol{A}_l \boldsymbol{P}_l$
    $l \leftarrow l + 1$
end while
$N = l$; $k = 0$
while $\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_1^k\| > \epsilon$ **do** {solution-phase}
    $k \leftarrow k + 1$
    for $l = 1$ **to** $N - 1$ **do**
        Smooth candidate solution: $\tilde{\boldsymbol{x}}_l^k \leftarrow \boldsymbol{S}_l(\tilde{\boldsymbol{x}}_l^k, \boldsymbol{b}_l)$
        Compute the defect: $\tilde{\boldsymbol{b}}_l = \boldsymbol{b}_l - \boldsymbol{A}_l \boldsymbol{x}_l^k$
        Restrict the defect to determine coarser level corrections: $\boldsymbol{b}_{l+1} = \boldsymbol{R}_l \tilde{\boldsymbol{b}}_l$
        set coarse level approximation: $\tilde{\boldsymbol{x}}_{l+1}^k = 0$
    end for
    Solve the coarsest system: $\tilde{\boldsymbol{x}}_N^k = \texttt{solve}(\boldsymbol{A}_N, \boldsymbol{b}_N)$
    for $l = N - 1$ **to** 1 **do**
        Interpolate to determine finer level corrections: $\Delta \boldsymbol{x}_l^k = \boldsymbol{P}_l \tilde{\boldsymbol{x}}_{l+1}^k$
        Update finer level variable: $\tilde{\boldsymbol{x}}_l^k \leftarrow \tilde{\boldsymbol{x}}_l^k + \Delta \boldsymbol{x}_l^k$
        Smooth candidate solution: $\tilde{\boldsymbol{x}}_l^k = \boldsymbol{S}_l(\tilde{\boldsymbol{x}}_l^k, \boldsymbol{b}_l)$
    end for
end while
return Approximate solution: $\tilde{\boldsymbol{x}}_{\text{final}} = \tilde{\boldsymbol{x}}_1^k$

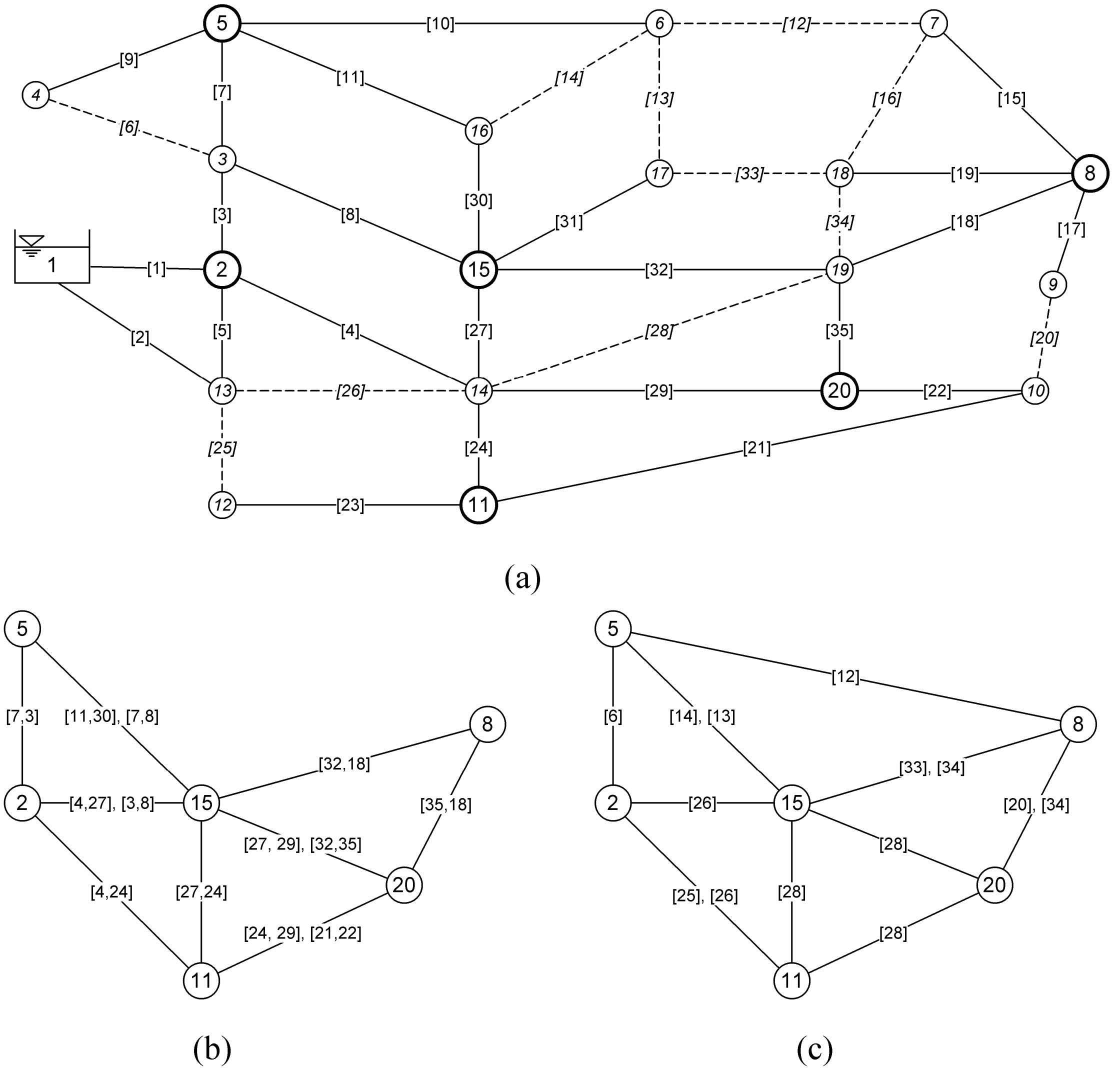

FIG. 2. Example of the topological interpretation of the AMG restriction operation. Subfigure (a) shows the original 35-pipe network where: larger nodes correspond to the coarse level nodes in node set $\mathcal{N}_c$ and the smaller nodes to the fine level nodes in node set $\mathcal{N}_f$; the bold links correspond to links within the Λ_{cf} link set; and the dashed links correspond to links within the Λ_{ff} link set. Subfigure (b) represents the coarse level network associated with matrix $\boldsymbol{V}_{(1)}$ comprised of nodes $\mathcal{N}_c$ and links Λ_{cf}. Subfigure (c) represents the coarse level network associated with matrix $\boldsymbol{V}_{(2)}$ comprised of nodes $\mathcal{N}_c$ and links Λ_{ff}.

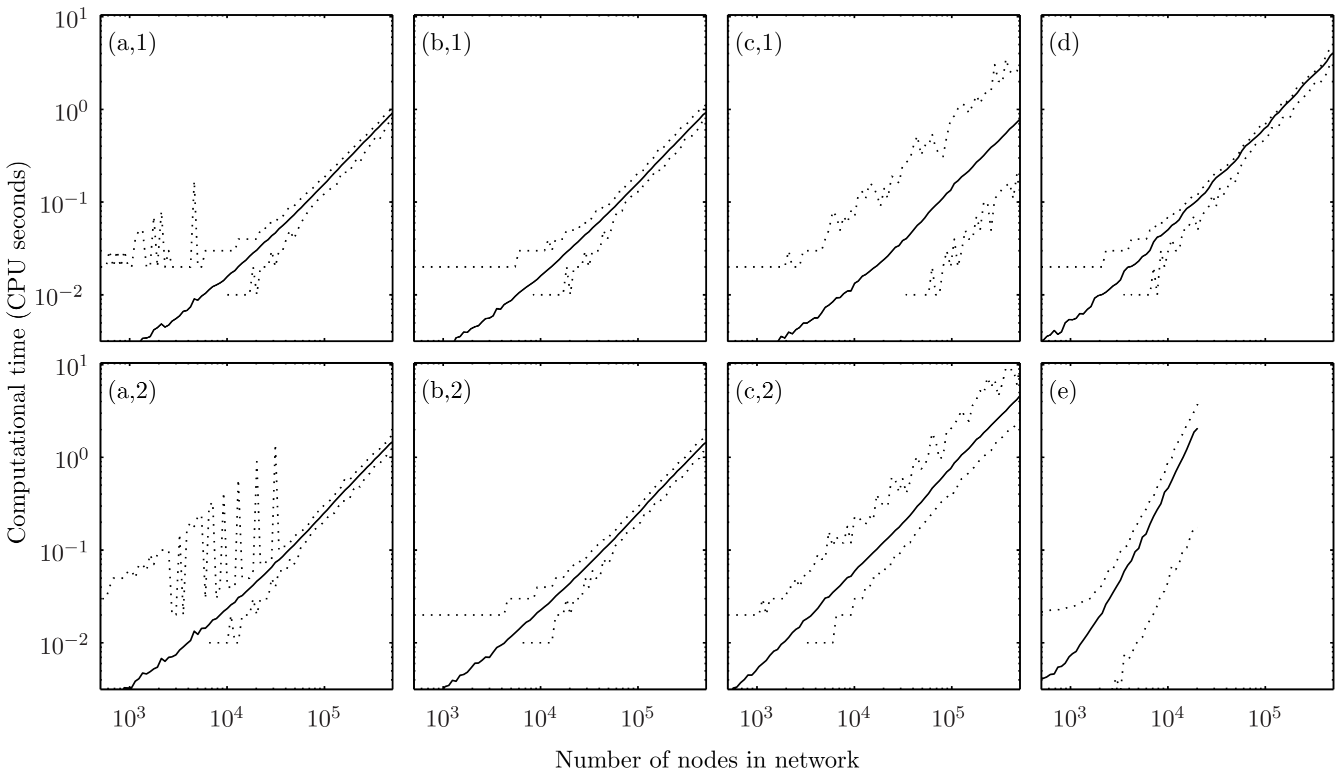

FIG. 3. Summary of computational times for numerical experiments. The lines correspond to the maximum (upper $\cdots$), the sample mean ($-$) and the minimum (lower $\cdots$) of the network nodal groupings. The plots correspond to (a) AMG, (b) AMG+CG, (c) ILU+CG, (d) PARDISO and (e) SC+NR algorithms for the tolerances (1) 10^{-2}, and (2) 10^{-6} (note that only single plots for PARDISO and SC+NR are given as these are direct solvers and not controlled by tolerance values.

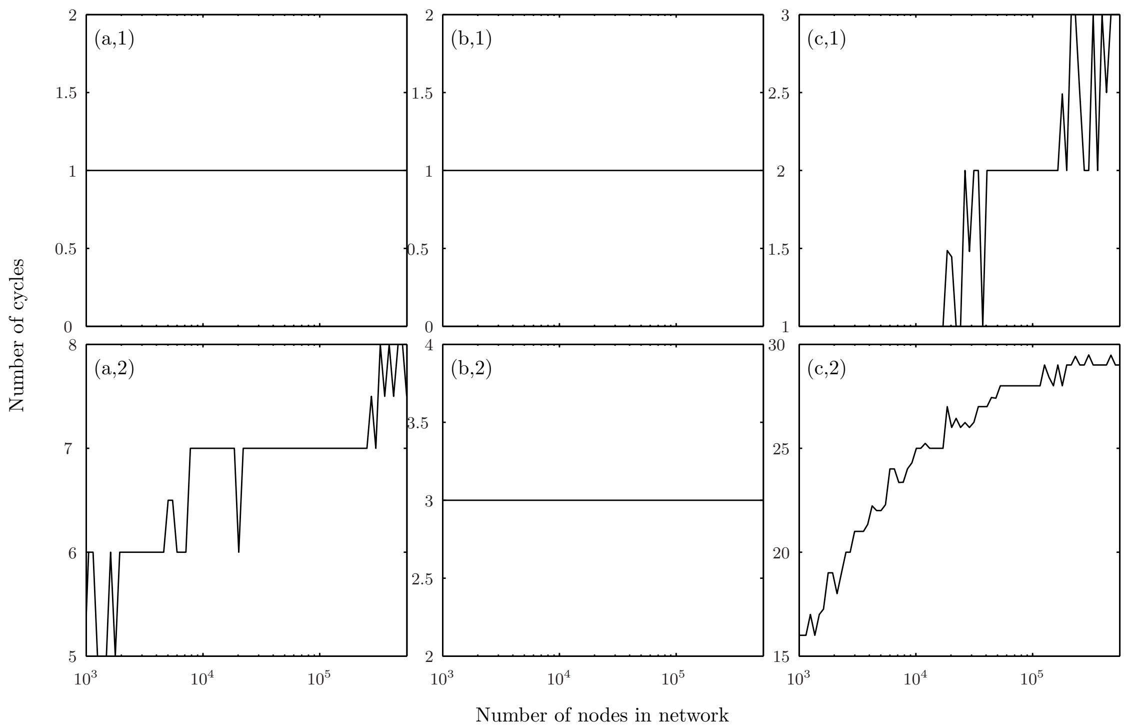

FIG. 4. Summary of computational cycles within the numerical experiments for the iterative algorithms. The lines correspond to the median of the network nodal groupings. The plots correspond to (a) AMG, (b) AMG+CG, and (c) ILU+CG algorithms for the tolerances (1) 10^{-2}, and (2) 10^{-6}. Note that the cycles presented only correspond to the AMG and ILU cycles and do not include the conjugate gradient iterations.

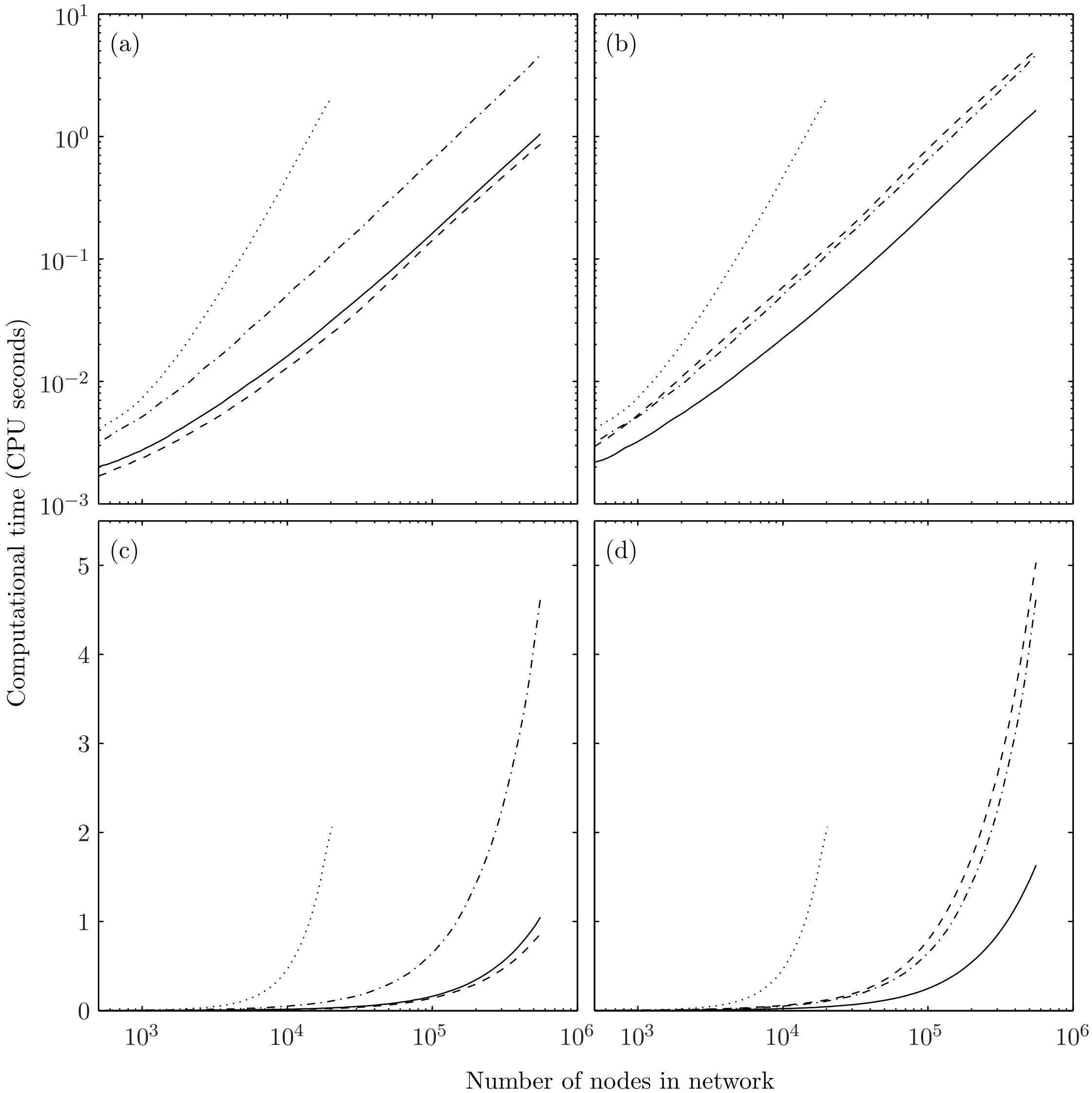

FIG. 5. Comparison of AMG+CG (–), ILU+CG (– – –), PARDISO (–·–) and SC+NR (···) in logarithmic and linear computational time (note that AMG is not depicted as its trend was indistinguishable from that of AMG+CG). The lines depict the averaged computational times, computed by averaging the computational times of the network nodal groupings and applying an 11-point smoother to the resultant data series. The plots correspond to a tolerance of 10^{-2} for plots (a) and (c), and 10^{-6} for plots (b) and (d).

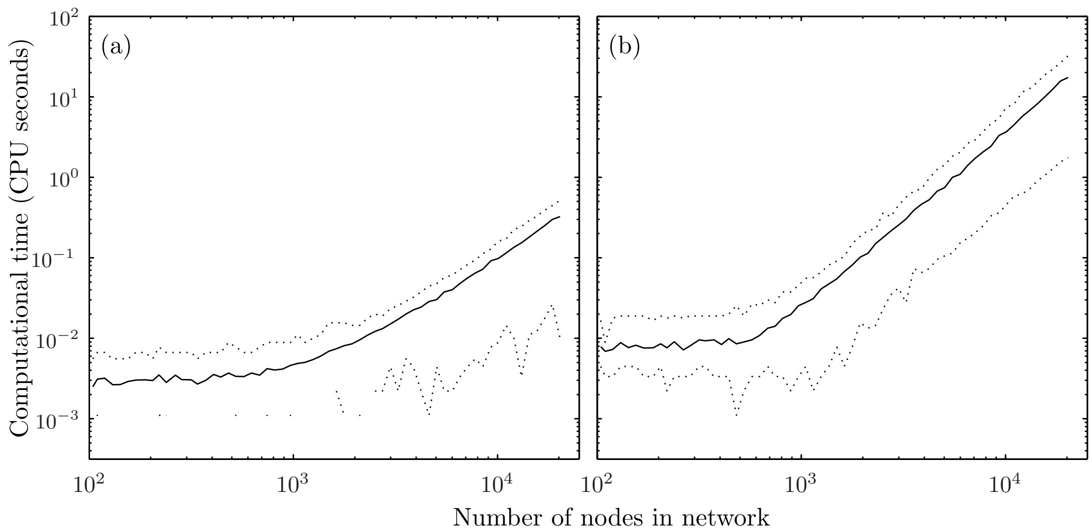

FIG. 6. Computational times for components of the SC+NR solver. The plots correspond to (a) the sparse Cholesky solver times, and (b) the node reordering times. The lines depict the minimum (lower $\cdots$), sample mean ($-$), and maximum (upper $\cdots$) computational times of the network nodal groupings.