Sparse Null Space Algorithms for Hydraulic Analysis of Large-Scale Water Supply Networks

Edo Abraham, M.ASCE[1]; and Ivan Stoianov, M.ASCE[2]

Abstract: In this article, a comprehensive review of existing methods is presented and computationally efficient sparse null space algorithms are proposed for the hydraulic analysis of water distribution networks. The linear systems at each iteration of the Newton method for nonlinear equations are solved using a null space algorithm. The sparsity structure of these linear equations, which arises from the sparse network connectivity, is exploited to reduce computations. A significant fraction of the total flops in the Newton method are spent in computing pipe head losses and matrix-matrix multiplications involving flows. Because most flows converge after a few iterations, a novel partial update of head losses and matrix products is used to further reduce computational complexity. Convergence analyses are also presented for the partial-update formulas. A new heuristic for reducing the number of pressure head computations of a null space method is proposed. These savings enable fast near-real-time control of large-scale water networks. It is often observed that the linear equations that arise in solving the hydraulic equations become ill-conditioned due to hydraulic solutions with very small and zero flows. The condition numbers of the Newton equations are bounded using a regularization technique with insignificant computational overheads. The convergence properties of all proposed algorithms are analyzed by posing them as an inexact-Newton method. Small-scale to large-scale models of operational water networks are used to evaluate the proposed algorithms. **DOI: 10.1061/(ASCE)HY.1943-7900.0001089.**

Author keywords: Water distribution networks; Hydraulic analysis; Sparse solver; Null space algorithm; Sparse LU; Partial updates; Graph theory; Sparse Cholesky; Inexact Newton method.

Introduction

Water distribution networks (WDNs) are large-scale systems the management of which faces an increasingly complex and challenging future due to aging infrastructure and rapid population growth. Emerging research in smart water distribution systems encompasses various topics in modeling, optimal control, estimation, model identification, and design optimization for large-scale hydraulic systems. An extensive overview of operational, technical, and economical challenges facing water utilities, and a collection of current research problems, can be found in Sensus (2012) and Brunone et al. (2013), respectively, and the references therein. Advances in sensor, control, and information technologies have also enabled the solution of some of these operational problems in near real time and for progressively larger networks. In all these, hydraulic analysis is required; a set of nonlinear equations governing pipe flows and nodal pressures across the network are solved to simulate the water distribution system behavior.

This article is concerned with the most common formulation of the hydraulic analysis problem, demand-driven hydraulic analysis (Guidolin et al. 2013), which poses the flow continuity and energy conservation laws for a pipe network as a set of nonlinear equations of the flows and unknown pressure heads for given nodal demands. The Newton method for solving nonlinear equations was exploited by Todini and Pilati (1988) to pose an iterative scheme, often called the global gradient algorithm (GGA) or the Todini and Pilati method. Collins et al. (1978) and, subsequently, Todini and Pilati (1988) have reformulated the hydraulic equations into an equivalent optimization problem to show the existence of a unique solution for network models containing only elements with convex loss functions. In Berghout and Kuczera (1997), conditions are given for the existence of unique solutions even when locally controlled elements like valves and pumps are included in the network. In addition, Berghout and Kuczera (1997) demonstrate the superiority of GGA (Todini and Pilati 1988) compared to linear programming techniques.

Previous work in literature has coupled conventional hydraulic simulation tools like *EPANET version 2.0* (Rossman 2000) with heuristic optimization schemes (e.g., genetic algorithms) to solve nonreal-time network design problems (Savic and Walters 1997; Nicolini and Zovatto 2009). A mathematical optimization approach has also been used to pose performance constraints and the hydraulic equations explicitly within the optimization (Eck and Mevissen 2013). The same nonlinear equations are employed as constraints in optimization problems for the dynamic reconfigurability of network topologies (Wright et al. 2014). The network hydraulic analysis problem is therefore more than a simulation problem. A prerequisite for efficiently solving all these water network design or near-real-time system management problems is the formulation of computationally efficient and robust hydraulic solvers.

The various approaches proposed by recent literature to improve computational efficiency of hydraulic solvers are reviewed here. As the size of networks tackled by water utilities become larger, some have considered the reduction of the mathematical problem through a smaller topological representation of the original network model; it has been standard practice for water utilities to skeletonize networks so each node abstracts an entire area or multiple points of consumption (Bhave 1988; Jankovic-Nisic and Chan 2013).

[1]Research Associate, Dept. of Civil and Environmental Engineering, Imperial College London, London SW7 2BU, U.K. E-mail: edo.abraham04@imperial.ac.uk

[2]Senior Lecturer in Water Systems Engineering, Dept. of Civil and Environmental Engineering, Imperial College London, London SW7 2BU, U.K. (corresponding author). E-mail: ivan.stoianov@imperial.ac.uk

Note. This manuscript was submitted on August 1, 2014; approved on August 3, 2015; published online on October 15, 2015. Discussion period open until March 15, 2016; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Hydraulic Engineering***, © ASCE, ISSN 0733-9429.

Recently, appropriate lumping of nodal demands along a pipe without compromising accuracy is proposed in (Giustolisi et al. 2011). In the cases where multiple simultaneous simulations of networks are required, parallelizing at the level of the analysis software using clusters of computers, multiple core CPUs, or graphics processing unit (GPUs) has been shown to be promising (Mair et al. 2014; Crous et al. 2012).

A finer-grain parallelization at the programming code level has also been proposed. At each iteration of the GGA, a linear system of equations is solved to determine the Newton step. The linear system of equations is formed by computing the network nonlinear equations and their Jacobian. Guidolin et al. (2013) analyze data parallelizing high-performance computing techniques to accelerate pipe head-loss computations at each linear solve of a GGA iteration. In addition to being highly data parallel (i.e., for each pipe, a head-loss computation is dependent only on the flow and roughness characters of the same pipe), the head-loss computations are shown by Guidolin et al. (2013), through simulations, to take approximately 20–40% of computational time justifying their candidacy for parallelism within the CSWNet library (Guidolin et al. 2010). Their CPU implementation is shown to speed up computations within the range of 9–26%, whereas their GPU implementations show speedups of 9–19% only for sufficiently large networks; this is due to the overhead incurred in moving data from the CPU main memory to the on-chip memory of GPUs. On the other hand, although a significant percentage of computational time is used by the linear solver, the sequential data access by the linear algebra operations makes it less suitable for parallelism (Guidolin et al. 2013).

Algebraic multigrid (AMG) methods, which were designed for solving partial differential equations with fine meshes, have also recently been considered for solving very large size (up to $\mathcal{O}(10^6)$) hydraulic problems (Zecchin et al. 2012). AMG methods solve a hierarchy of smaller coarse problems, of which direct solutions are used to refine approximate solutions of the overall linear system. Zecchin et al. (2012) also consider AMG preconditioners for the iterative preconditioned conjugate gradient (PCG) method. AMG is shown to outperform, in CPU time, the incomplete Cholesky preconditioner for the conjugate gradient method for larger problems with smaller error tolerances. However, the examples used in Zecchin et al. (2012) are randomly generated dense rectangular network models; for real water networks, which are very sparsely connected, experiments in Giustolisi and Moosavian (2014) show that AMG is much inferior to sparse direct solvers.

This approach exploits the structure of the linear problem of the Newton iterations for more efficient and robust alternatives. The GGA linear solve step was formulated in Todini and Pilati (1988) as a two-stage scheme; via block matrix substitutions, the heads are first solved for, followed by the flows. This requires the inversion of a diagonal matrix that becomes badly conditioned when some flows are very small or zero. Hydraulic analysis software substitutes an arbitrary threshold for zero flows to guarantee this matrix inversion (Todini and Pilati 1988; Rossman 2000; Todini and Rossman 2012). For systems with zero flows, Elhay and Simpson (2011) propose a systematic regularization method to restrict the condition number of the linear system using only marginally more computations. This approach reduces the loss of accuracy or convergence caused by inverting a badly conditioned Jacobian.

The Newton method for hydraulic analysis has a Jacobian with what is called a saddle point structure. In the numerical optimization literature, null space algorithms for saddle point problems have been studied extensively, often called *reduced Hessian* methods; see the review paper Benzi et al. (2005) and the references therein. Null space algorithms, as opposed to the range space method of GGA (Todini and Pilati 1988), have also been applied for hydraulic analysis of water and gas pipe networks (Nielsen 1989; Rahal 1995; Elhay et al. 2014; Todini and Rossman 2012). The three references (Nielsen 1989; Rahal 1995; Elhay et al. 2014) differ in the way they generate the null space bases. Nielsen (1989) proposes the use of matrix row and column permutations to compute null space bases, whereas Rahal (1995) uses graph theoretic tools to solve what he called the cotree formulation of the problem. In addition to showing the equivalence of the two methods in Nielsen (1989) and Rahal (1995), Elhay et al. (2014) propose a simple matrix reduction-based approach for null basis generation in their reformulation, which they call the reformulated cotree flows method (RCTM). In addition to mathematically showing that the GGA and null space methods always take the same number of Newton iterations, Elhay et al. (2014) also demonstrate, through case studies, the superiority of RCTM in both memory requirements and computational speed compared to GGA.

For a WDN with n_p number of pipes (or links) and n_n unknown-head nodes, the number $n_l = n_p - n_n$, which is the number of cotree flows or loops (Elhay et al. 2014), is often much smaller than n_n. At each iteration, whereas the GGA method solves a linear problem of size n_n, a null space method solves a possibly much smaller problem of size n_l but with the same symmetric positive definiteness properties. Therefore, for a sparsely networked system, significant computational savings can be made. Moreover, unlike the GGA scheme, null space algorithms do not involve matrix inversion. Thus, they may not require processes to deal with zero flows provided some mild conditions are satisfied by the system matrices (Benzi et al. 2005; Elhay et al. 2014).

In this article, an efficient null space algorithm-based Newton method for hydraulic analysis is developed. From studies (see section "Partial Update Methods for the Null Space Algorithm"), and in line with the results in Guidolin et al. (2013), it is apparent that more than 60% of computational time of the Newton method is spent in computing head losses, matrix-matrix multiplications, and linear solves. Having shown that most flows converge after a few iterations, the authors propose a novel partial update scheme to reduce the number of computations in calculating head losses and matrix-matrix multiplications involving flows. The proposed scheme is also formulated as an inexact Newton method to guarantee convergence properties. In addition, case studies are used to demonstrate that the method results in significant computational savings.

It is also noted that the algorithm allows for and can benefit from all the software level and small-grain parallelizations discussed in Mair et al. (2014), Crous et al. (2012), and Guidolin et al. (2013). The regularization of Elhay and Simpson (2011) is also treated within the same mathematical framework to derive conditions under which it stays an inexact Newton method, hence guaranteeing convergence. Because the flow update equations of the null space algorithm do not depend on pressure evaluations, a novel scheme for reducing the number of pressure head computations is proposed for further computational savings.

A step-by-step derivation of the null space algorithm from the hydraulic equations is first presented, and then various computational tools for generating sparse null bases and sparse factorizations are discussed. State-of-the-art solvers from the *SuiteSparse version 4.1.2* library (Davis 2004) are used. It is also the case that the most expensive tasks of the scheme need not be recomputed more than once. In the scheme, the values that stay constant over different steady-state simulations are computed only once. In addition to the rational exponent pipe resistance computations in Hazen-Williams models, the matrix whose columns span the null space of the network topology and the Cholesky factors for the head equations are two more examples. Within each hydraulic

simulation, resistance coefficients and head losses that do not change with further Newton iterations are also not recomputed; for example, for both Darcy-Weisbach and Hazen-Williams models, a significant fraction of pipes have flows that do not change with Newton iterations.

The remainder of this article is organized as follows. In the next section, the hydraulic analysis problem and traditional solution methods are discussed, with comprehensive literature review where relevant. New proofs are derived that show the convergence properties of the Newton method for hydraulic analysis. The third section examines the structure of the Newton linear systems and then discusses relevant null space algorithms. Sparse null basis computation tools are also developed and implemented. In the fourth section, novel methods for making the null space algorithm faster are proposed. The use of partial update sets and a related new method for reducing head computations are described. Mathematical proofs are derived to show the Newton method stays convergent with the introduced modifications. Finally, a numerical study with further results is presented using a number of operational network examples in the case study section, followed by conclusions.

For a vector $v \in \mathbb{R}^n$, the usual p-norms are defined as $\|v\|_p := (\sum_{i=1}^{n} |v_i|^p)^{1/p}$, $p = 1, 2$ and $\|v\|_p = \max_i |v_i|$ if $p = \infty$. For a matrix A, $\|A\|_p = \max_{\|x\|=1} (\|Ax\|_p)/(\|x\|_p)$, where $\|Ax\|_p$, $\|x\|_p$ are the corresponding vector norms. The variable P^T denotes the transpose of the matrix P. The condition number for an invertible matrix X is denoted by $\kappa(X)_p := \|X\|_p \|X^{-1}\|_p$.

Network Analysis Problem

Nonlinear Flow and Energy Equations: Solution via Newton Iterations

In demand-driven hydraulic analysis, the demand is assumed known, as opposed to pressure-driven simulations in which demands are written as functions of pressure (Giustolisi et al. 2008) to be solved for. Once a WDN is defined by its connectivity, and the characteristic of its pipes and the demands at each node, a steady-state solution of the system is computed by solving the flow conservation and energy loss equations for a given demand. The objective is to compute the unknown flows in each pipe and the pressures at the demand nodes. Let pipe p_j have flow q_j going from node i to node k, and with pressure heads h_i and h_k at nodes i and k, respectively. The head loss across the pipe can then be represented as

$$h_i - h_k = r_j |q_j|^{n-1} q_j \tag{1}$$

where r_j, the resistance coefficient of the pipe, can be modeled as either independent of the flow or implicitly dependent on flow q_j and given as $r_j = \alpha L_j/(C_j^n D_j^m)$. The variables L_j, D_j, and C_j denote the length, diameter, and roughness coefficient of pipe j, respectively. The triplet α, n, and m depends on the energy loss model used; Hazen-Williams (HW: $r_j = 10.670 L_j/(C_j^{1.852} D_j^{4.871})$) and Darcy-Weisbach (DW) are two commonly used loss formulas (Elhay and Simpson 2011). In DW models, the dependence of the resistance coefficient on flow is implicit; see the formulas in Simpson and Elhay (2010) [Eqs. (1) and (2)].

Given is a network with n_p links connecting $n_n (< n_p)$ unknown head nodes, and n_0 known head nodes. The vector of unknown flows and pressure heads is defined as $q = [q_1, \ldots, q_{n_p}]^T$ and $h = [h_1, \ldots, h_{n_n}]^T$. With head-loss equations defined for each pipe and the fixed heads and demands for each node taken into account, the set of nonlinear equations that define the steady state flow conditions are given by the matrix equation [Todini and Pilati 1988, Eq. (1)]

$$f(q,h) := \begin{pmatrix} A_{11}(q) & A_{12} \\ A_{12}^T & 0 \end{pmatrix} \begin{pmatrix} q \\ h \end{pmatrix} + \begin{pmatrix} A_{10} h_0 \\ -d \end{pmatrix} = 0 \tag{2}$$

where the variables $h_0 \in \mathbb{R}^{n_0}$ and $d \in \mathbb{R}^{n_n}$ represent the known heads (e.g., at a reservoir or tank) and demands consumed at nodes, respectively. The elements of the matrix $A_{12} \in \mathbb{R}^{n_p \times n_n}$, which is the incidence matrix relating the n_p links with the n_n unknown head nodes, are defined as

$$A_{12}(j,i) = \begin{cases} -1, & \text{if pipe } j \text{ leaves node } i \\ 1, & \text{if pipe } j \text{ enters node } i \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where inflows into a node are considered positive and outflows negative. The incidence matrix A_{10} for the fixed-head nodes is defined using the same convention as (3). Demands d consumed at nodes are represented by positive values. The square matrix $A_{11} \in \mathbb{R}^{n_p \times n_p}$ is a diagonal matrix with the elements

$$A_{11}(j,j) = r_j |q_j|^{n_j - 1}, \quad j = 1, \ldots, n_p \tag{4}$$

representing part of the loss formula in Eq. (1).

Most nonlinear equations and unconstrained optimization problems are solved using Newton's method (Nocedal and Wright 2006; Dennis and Schnabel 1996). The same Newton method has been applied to solve the hydraulic analysis problem, for the first time in Todini and Pilati (1988), and has been extensively used for the same purpose since then. By considering the Jacobian of $f(q,h)$ with respect to the unknown $x := [qh]^T$, and using the head-loss model for the ith link (4), it can be shown (Appendix I) that the Newton iteration for the solution of Eq. (2) is

$$\begin{aligned} \nabla f(x^k)(x^{k+1} - x^k) &= -f(x^k) \\ \begin{bmatrix} NA_{11}(q^k) & A_{12} \\ A_{12}^T & 0 \end{bmatrix} \begin{bmatrix} dq \\ dh \end{bmatrix} &= -\begin{bmatrix} A_{11}(q^k) & A_{12} \\ A_{12}^T & 0 \end{bmatrix} \begin{bmatrix} q^k \\ h^k \end{bmatrix} \\ &\quad + \begin{bmatrix} -A_{10} h_0 \\ d \end{bmatrix} \end{aligned} \tag{5}$$

where $\begin{bmatrix} dq \\ dh \end{bmatrix} = \begin{bmatrix} q^{k+1} - q^k \\ h^{k+1} - h^k \end{bmatrix}$ and $N = \text{diag}(n_j)$, $j = 1, \ldots, n_p$

Here, it is first demonstrated why the Newton method works well for hydraulic analysis by investigating its convergence properties. This proof allows guarantee of the convergence properties of the null space algorithms proposed in the subsequent section by posing them as inexact Newton methods.

Lemma 1 (convergence of Newton method): Let $x^* := [q^* \quad h^*]^T \in D$, with open convex set D, be a nondegenerate solution of Eq. (2), i.e., the Jacobian $\nabla f(x^*)$ is not singular, and let $\{x^k\}$ be the sequence of states generated by the Newton iteration (5). For $x^k \in D$ sufficiently near x^*, the Newton sequence exists [i.e., $\nabla f(x^i)$ is nonsingular for all $i > k$] and has local superlinear convergence.

Proof: As shown in Appendix I, $f(\cdot)$ is continuously differentiable in $\mathbb{R}^{n_p + n_n}$. If x^* is assumed nondegenerate, the proof is a standard result and is relegated to Nocedal and Wright (2006) (Thm.11.2).

Remark 1: It is also shown in Appendix I that the Jacobian of the loss functions is Lipschitz either when a Darcy-Weisbach equation is used (for laminar flows) or the solution does not have zero flows. In this case, the Newton algorithm will have local quadratic convergence by Nocedal and Wright (2006), Thm.11.2.

Almost all of the computational cost of the Newton method is incurred in the repeated solving of the linear system (5) to find the Newton step. This linear system is, however, sparse and with a special structure. Therefore, the rest of this article investigates the structure of this problem and proposes novel tailored efficient solvers for it.

Schur Complement Reduction with Nonsingular A_{11}

An interesting property of the Newton equations (5) is that they have what is called a *saddle point structure* (Benzi et al. 2005); if a 2×2 block structure is considered, the A_{11} block is symmetric positive definite or semidefinite, $A_{21} = A_{12}^T \in \mathbb{R}^{n_n \times n_p}$, $n_p \geq n_n$, and $A_{22} = 0$. The same class of problems arises in many partial differential equation (PDE)-constrained optimization problems with various boundary conditions (Pearson et al. 2012). Due to the indefiniteness and often poor conditioning, saddle point systems are challenging to solve efficiently and accurately.

The two main solution algorithms for saddle point systems segregate this $n_p + n_n$ size linear equation into two smaller (or reduced) systems (Benzi et al. 2005): one for each of q and h. Methods based on the Schur complement or its approximations (Benzi et al. 2005; Pearson and Wathen 2012) are well-studied and a primary candidate. The GGA method uses the Schur complement reduction of the larger saddle point matrix in Eq. (5) to decouple the q and h equations; for this reason, from here on GGA is also called a Schur scheme/method, a name often used in the numerical analysis and optimization literature (Benzi et al. 2005). On the other hand, as will be shown in the next section, null space algorithms reduce and decouple the saddle point system in Eq. (5) by using projections onto the null basis of the A_{12} matrix. Considering the block partitions of Eq. (5), and with the assumption that A_{11} is invertible

$$\begin{bmatrix} b_1^k \\ b_2^k \end{bmatrix} = \begin{bmatrix} -(A_{11}^k q^k + A_{12} h^k + A_{10} h_0) \\ -A_{12}^T q^k + d \end{bmatrix} \tag{6}$$

Then, by block elimination (Boyd and Vandenberghe 2004, App.C.4; Benzi et al. 2005, Section 5), the equivalent but smaller linear equations can be derived:

$$\begin{aligned} &A_{12}^T (N A_{11}^k)^{-1} A_{12} dh = A_{12}^T (N A_{11}^k)^{-1} b_1^k - b_2^k \\ &dq = (N A_{11}^k)^{-1} (b_1^k - A_{12} dh) \end{aligned} \tag{7}$$

This reduction is called a Schur complement reduction because the matrix $A_{12}^T (N A_{11}^k)^{-1} A_{12}$ is the negative Schur complement of larger matrix in (5). Here, for invertible A_{11}^k, this Schur complement reduction involves only simple element-wise inversions of the diagonal matrices A_{11}^k and N. With a few more algebraic manipulations, (7) can be simplified to the following equations [Todini and Pilati 1988, Eqs. (18) and (19)] by simply substituting for b_1^k and b_2^k:

$$\begin{aligned} A_{12}^T (N A_{11}^k)^{-1} A_{12} h^{k+1} = &-A_{12}^T N^{-1} [q^k + (A_{11}^k)^{-1} A_{10} h_0] \\ &- (d - A_{12}^T q^k) \end{aligned} \tag{8}$$

$$q^{k+1} = (I - N^{-1}) q^k - (N A_{11}^k)^{-1} (A_{12} h^{k+1} + A_{10} h_0) \tag{9}$$

Therefore, given an initial guess (q^k, h^k), solving (5) can be accomplished by first solving (8) and the flows q^{k+1} are then computed by substituting for h^{k+1} in (9). In Todini and Pilati (1988), this reformulation of the original linear problem (5) is called "the nodal version of the Newton method"; this, simply because a linear problem is solved only for the node heads in Eq. (8).

The matrix $X := A_{12}^T (N A_{11}^k)^{-1} A_{12}$ is symmetric positive definite (SPD) and has the same sparsity structure as $A_{12}^T A_{12}$; positive definite, because A_{11} is assumed invertible and A_{12} has full column rank (Elhay and Simpson 2011; Elhay et al. 2014). In addition, for real water networks, the average degree of a node in the most dense networks is sub 3, as shown in the examples in the case study section and the case studies in Giustolisi and Moosavian (2014) and references therein, i.e., the number of nonzero elements in A_{12} is $\lessapprox 3n_n$. Therefore, the linear problem that is solved in (8) is smaller (of size n_n as opposed to $n_p + n_n$), highly sparse, and SPD.

Sparsity is of paramount importance because it allows reduction of the number of flops (floating point operations, where a flop refers to one addition, subtraction, multiplication, or division operation between two floating point numbers (Boyd and Vandenberghe 2004, Appx.C) required in solving (7). For example, by avoiding multiplications and additions with zero, calculating matrix-matrix and matrix-vector multiplications, matrix factorizations, and forward and back substitutions can be accelerated (Boyd and Vandenberghe 2004, Appx.C). In addition to significant reduction in flops of operations, exploiting sparsity also allows the efficient use of memory through clever sparse storage formats (Davis 2006). For example, whereas sparse matrix operations on a water network model with $\mathcal{O}(10^5)$ nodes are possible on a current workstation computer, using full matrices would result in storing and manipulating matrices with memory requirements of $\mathcal{O}(10^{15})$ bytes—an infeasible prospect on a state-of-the-art workstation in 2014.

Both direct and iterative methods for SPD systems are used to solve (8) (Todini and Pilati 1988; Giustolisi and Moosavian 2014). In Todini and Pilati (1988), the iterative preconditioned conjugate gradient method, with a zero-fill incomplete Cholesky factorization for a preconditioner, was proposed and used. However, it has been noted in recent literature (Giustolisi and Moosavian 2014, and references therein) that the condition number and, therefore, the effectiveness of iterative methods degrade as the problem size increases. For GGA, the case study in Giustolisi and Moosavian (2014) has shown that sparse direct solvers are superior to iterative methods for problems of varying size. Because the linear systems solved by both the Schur and null space algorithms are sparse SPD, the sparse Cholesky factorization and its variants are preferred for solving these equations.

Regularization Scheme for Zero Flows

When the flow in the ith link is zero or very small, so is the ith diagonal element of A_{11}, causing numerical ill-conditioning and possible positive semidefiniteness of X. To avoid this, Todini and Pilati (1988) proposed replacing a diagonal element of A_{11} that is smaller than an arbitrary small positive number δ by the bound δ; this is implemented in hydraulic solvers like *EPANET version 2.0*. By setting this small tolerance δ whenever the value of an iterate q^k is too small, zero flow cases are never allowed for in any link.

This use of arbitrary small numbers can introduce significant losses of accuracy in the solution, which cannot be made arbitrarily small in a numerically robust way. Recent works in the literature suggest alternatives; for example, Gorev et al. (2012) propose the use of a linear loss model to approximate the Hazen-Williams loss formulas when a flow is below some threshold. In Elhay and Simpson (2011), a Jacobian regularization that systematically guarantees to avoid the ill-conditioning of the head-loss matrices A_{11} is described. The Newton system of linear Eq. (5) can be rewritten as

$$\begin{bmatrix} F^k & A_{12} \\ A_{12}^T & 0 \end{bmatrix} \begin{bmatrix} q^{k+1} \\ h^{k+1} \end{bmatrix} = \begin{bmatrix} F^k - G^k & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} q^k \\ h^k \end{bmatrix} + \begin{bmatrix} -A_{10}h_0 \\ d \end{bmatrix} \tag{10}$$

where $G^k = A_{11}(q^k)$ and $F^k = NG^k$.

Similarly to Todini and Pilati (1988), the authors of Elhay and Simpson (2011) suggest changing the very small or zero elements on the diagonals of F^k that would otherwise cause ill-conditioning. At the kth iteration, for small flow q_i^k of the ith link, it is assumed that $t_i q_i^{k+1} \approx t_i q_i^k$ for some $t_i > 0$, resulting in the approximation

$$\begin{bmatrix} F^k + T^k & A_{12} \\ A_{21} & 0 \end{bmatrix} \begin{bmatrix} q^{k+1} \\ h^{k+1} \end{bmatrix} = \begin{bmatrix} F^k - G^k + T^k & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} q^k \\ h^k \end{bmatrix} + \begin{bmatrix} -A_{10}h_0 \\ d \end{bmatrix} \tag{11}$$

where $T^k = \text{diag}(t_j)$, $j = 1, \ldots, n_p$

Let $\tilde{F}^k := F^k + T^k$. With $\tilde{F}^k$ invertible, it can be shown using block eliminations that the Newton iterations solve the decoupled linear systems that follow:

$$\underbrace{A_{12}^T(\tilde{F}^k)^{-1}A_{12}}_{Y^k} h^{k+1} = A_{21}(\tilde{F}^k)^{-1}\{(\tilde{F}^k - G^k)q^k - A_{10}h_0\} - d \tag{12}$$

$$q^{k+1} = (\tilde{F}^k)^{-1}\{(\tilde{F}^k - G^k)q^k - A_{10}h_0 - A_{12}h^{k+1}\} \tag{13}$$

Because $\tilde{F}^k$ is diagonal and invertible, it is straightforward to derive the bound on the two-norm condition number of Y^k, $\kappa_2[A_{12}^T(\tilde{F}^k)^{-1}A_{12}] \le \kappa_2(\tilde{F}^k)\kappa(A_{12})^2$ shown in Elhay and Simpson (2011), using the triangle inequality for the matrix norm. Therefore, by having $\kappa_2(\tilde{F}^k)$ reduced, $\kappa_2(Y^k)$ can be reduced. Because the error in the numerical solution of Eq. (12) is proportional to $\kappa_2(Y^k)$ (Higham 1996), the work in Elhay and Simpson (2011) suggests a systematic way to choose T so that the condition number of the regularized matrix $F^k + T^k$ is bounded above by some given number $\bar{\kappa}$, i.e., $\kappa_2(F^k + T^k) \le \bar{\kappa}$. This, instead of adding an arbitrary constant as in Todini and Pilati (1988) or linearizing the loss model at a flow threshold as in Gorev et al. (2012).

For the diagonal matrix with only positive elements $F^k + T^k$, the condition number is the ratio of the maximum diagonal element to the minimum diagonal element:

$$\kappa_2(F^k + T^k) = \frac{\max_j(F_{jj} + T_{jj})}{\min_j(F_{jj} + T_{jj})}$$

If $\bar{\kappa} \ge 1$ denotes the upper bound for the condition number of the regularized matrix, the following results:

$$\frac{\max_j(F_{jj}^k)}{F_{jj}^k + t_j^k} \le \bar{\kappa}, \qquad j = 1, \ldots, n_p \tag{14}$$

Therefore, choosing

$$t_j^k = \max\left[\frac{\lambda_{\max}(F_{ii}^k)}{\bar{\kappa}} - F_{jj}^k, 0\right]$$

will guarantee that $\kappa_2(\tilde{F}^k) \le \bar{\kappa}$.

At each iteration k, finding the maximum diagonal element of F^k, which has complexity of $\mathcal{O}(n_p)$ comparison operations, and performing n_p divisions have, together, a computational cost that is only marginal compared to solving the set of regularized linear systems. However, it must be noted that the regularized process is no longer the same Newton method. As a result, the conditions that guarantee the convergence of this regularized problem need to be carefully defined. In section "Partial Update Methods for the Null Space Algorithm," mathematical conditions are derived for this regularization to still be a Newton iteration with convergence properties. In the next section, null space algorithms are introduced for solving the Newton linear systems and various computational aspects are discussed. A comparison with the Schur method will also be made using a real WDN model.

Null Space Algorithms for Hydraulic Analysis

The Schur method of Eq. (7) is also called a range space method because the flows are solved for in the range space of A_{12}^T or *nodal analysis* because the Schur-complement $A_{12}^T F^{-1} A_{12}$ is used to solve for nodal heads. In addition to its higher computational cost, the main disadvantage is the requirement that the A_{11} block be nonsingular.

Compared to Schur methods, as will be outlined in the following section, null space algorithms are advantageous for solving problems in which $n_p - n_n$ is small and in which the A_{12} matrix does not change with the Newton iteration. In addition to hydraulic analysis (Nielsen 1989; Rahal 1995; Elhay et al. 2014), null space algorithms have been exploited in optimization, electrical circuit analysis, computational structural mechanics, and unsteady fluid dynamics applications in which problems have this saddle point structure; see Benzi et al. (2005), Section 6, for a long list of literature on such applications. Moreover, there is no requirement for A_{11} to be nonsingular; even when A_{11} is singular the condition that $\ker(A_{11}) \cap \ker(A_{12}^T) = \{0\}$ is sufficient to guarantee a unique solution by a null space method. If the regularization scheme of the last section is used, this condition will always be valid.

Assuming that A_{12} has full column rank, which is shown to be true for the WDN model in (Elhay et al. 2014), and $\ker(A_{11}) \cap \ker(A_{12}^T) = \{0\}$, a much smaller problem can be solved at each iteration using null space methods. Let the columns of a nonzero matrix $Z \in \mathbb{R}^{n_n \times n_l}$, $n_l = n_p - n_n$, span the null space of A_{12}^T, i.e., $A_{12}^T Z = 0$; q^{k+1} in Eq. (10) can be decomposed as

$$q^{k+1} = x^* + Zv \tag{15}$$

where x^* is one of an infinite number of solutions for $A_{12}^T x = d$ (e.g., a least-squares solution if $d \ne 0$ would suffice) and v is unknown. Substituting for q^{k+1} in the first block row of Eq. (10) and premultiplying by Z^T gives

$$\begin{aligned} &F^k(x^* + Zv) + A_{12}h^{k+1} = (F^k - G^k)q^k - A_{10}h_0 \\ &\Rightarrow Z^T F^k Z v = Z^T[(F^k - G^k)q^k - A_{10}h_0 - F^k x^*] \end{aligned} \tag{16}$$

because $Z^T A_{12} = 0$ (Nielsen 1989; Rahal 1995; Elhay et al. 2014; Todini and Rossman 2012).

The heads are then calculated by solving

$$A_{12}^T A_{12} h^{k+1} = A_{12}^T\{(F^k - G^k)q^k - A_{10}h_0 - F^k q^{k+1}\} \tag{17}$$

A null space algorithm-based Newton method (10) first solves for x^* such that $A_{12}^T x^* = d$, and then iteratively solves (16) and (17) in sequence until convergence is achieved.

This method boasts a number of computationally attractive properties. First, where the null space dimension n_l is small, the linear system in (16) is much smaller than the Schur method equations (12). Moreover, the matrices $Z^T F^k Z$ can be shown to be SPD. Even when F^k is singular, the condition $\ker(F^k) \cap \ker A_{12}^T = \{0\}$

is sufficient to show positive definiteness. The matrix coefficient of (17), $A_{12}^T A_{12}$, is similarly SPD—see the appendix of Elhay et al. (2014) for proof that A_{12} has full rank, and positive definiteness follows. It is also the case that $A_{12}^T A_{12}$ does not change with Newton iteration—it is a constant for a given network topology. This means the factorization of this matrix needs to be done only once. For (sparse) linear solvers, because the matrix factorization stage is the most computationally demanding stage (Boyd and Vandenberghe 2004, Appx.C), the single factorization results in large computational savings.

Finally, because F^k is diagonal, the null space problem will be sparse if Z is sparse. It is also desirable that the condition number of Z be low because the condition number of $Z^T F^k Z$ in Eq. (16) is bounded by its square. Depending on the method of choice for computing Z, a number of null space methods can be adopted; Algorithm 1 shows the Newton method tailored to a null space solver for hydraulic equations.

In this paper, the authors are concerned with demand-driven analysis in which the demand d is constant, resulting in the saddle point structure of the Newton equations in Eq. (5). In leakage analysis, pressure-driven models are used in which the demand depends on the nodal pressures, i.e., $d := d(h)$, see Giustolisi and Walski (2011), Eqs. (1)–(3). Because the derivative of the continuity equation with respect to pressure is nonzero in pressure-driven simulations, the A_{22} block of the matrix in Eq. (5) becomes nonzero and the demand would change as pressure changes with each Newton iteration k, $d := d^k(h^k)$. In such cases, the standard saddle point structure is lost and the applicability of the null space algorithms is limited. A more detailed discussion of various nonlinear models for pressure-dependent demand and leakage components can be found in Giustolisi and Walski (2011).

Computing Null Space Basis Z

For large-scale systems, if Z is sparse, $Z^T F^k Z$ can be explicitly formed for solution with direct or iterative methods. Depending on the structure of A_{12}, a number of graph-based methods can be employed to compute a sparse null basis Z. For example, Kirchhoff's classical method seeks to find the null spaces by finding a spanning tree of the network using the incidence matrix A_{12} and then constructing loops using the respective cotree (Benzi et al. 2005). The resulting basis is called an edge-loop matrix or solenoidal basis depending on the application. This process, also employed in Rahal (1995), requires no floating-point arithmetic to form Z but with no guarantees on its sparsity. In fact, the sparsity of Z will depend on the particular spanning tree used; the tree for which the sum of the number of edges in the fundamental loops is minimized results in the sparsest basis Z. Finding such a tree is, however, an non-deterministic polynomial-time hard (NP-hard) problem (Benzi et al. 2005). Nonetheless, practical heuristics exist for solving this problem approximately.

Algorithm 1. Exact Newton Method with Null Space Algorithm

Preprocessing: Compute all constants in the algorithm
(1) Compute null space basis Z
(2) Factorize $A_{12}^T A_{12}$ (i.e., sparse Cholesky to get L such that $LL^T = A_{12}^T A_{12}$)
(3) Compute x from $A_{12}^T x = d$ [e.g., a least-squares solution with LSQR (Davis 2004)]
Input: δ_N, $k_{\max}$, (x, L, Z)
Algorithm:
1: set $k = 0$, and compute $G^0, F^0, \|f(q^0, h^0)\|_\infty$
2: while $\|f(q^k, h^k)\|_\infty > \delta_N$ AND $k \le k_{\max}$ do
3: $F^k = \text{Regularize}(F^k)$
4: $\underbrace{Z^T F^k Z}_{X^k} = \sum_{i=1}^{n_p} f_i^k z_i z_i^T$
5: Solve $X^k v^k = b^k$
6: $q^{k+1} = x + Z v^k$
7: Recompute G^k, F^k
8: Solve $LL^T h^{k+1} = b(q^{k+1})$ by back substitutions
9: Set k to $k + 1$
10: Compute the residual error $\|f(q^k, h^k)\|_\infty$
11: end while

Unlike in Rahal (1995), the methods of Nielsen (1989) and Elhay et al. (2014) do not consider virtual loops, spanning trees, and cotrees—a purely algebraic approach is taken in forming the null bases. Because the incidence matrix $A_{12} \in \mathbb{R}^{n_p \times n_n}$ has full column rank, it follows that there always exist a row permutation matrix P and a column permutation matrix Q such that

$$PA_{12}Q = \begin{bmatrix} L_1^T \\ L_2^T \end{bmatrix} =: L \tag{18}$$

where $L_1^T \in \mathbb{R}^{n_n \times n_n}$ is invertible, and $L_2^T \in \mathbb{R}^{(n_p - n_n) \times n_n}$. It is then straightforward to show that the matrix

$$Z = P^T \begin{bmatrix} -L_1^{-1} L_2 \\ I \end{bmatrix} \tag{19}$$

is a null basis for A_{12}^T, i.e., $A_{12}^T Z = 0$ (Benzi et al. 2005).

In Nielsen (1989), no assumptions are made on the factorization (18) but that L_1 be invertible and $Q = I_{n_n}$.

In Elhay et al. (2014), it is noted that all WDNs have at least one fixed head node (e.g., a reservoir or tank) connected to an unknown head node. For such a link connecting the fixed head node to the unknown head node, the corresponding row of the A_{12} matrix will have only one nonzero element. This nonzero element is used as an initial pivot in interchanging rows and columns. The permutations are repeated n_n times to find row and column permutations P and Q, respectively, resulting in a lower triangular L_1^T. A Gaussian substitution is then used to form the null basis (19). In practice, this method results in a very sparse and well-conditioned null basis from the sparse matrices L_1 and L_2.

If a triangular structure is considered for (18) similarly to Elhay et al. (2014), a sparse LU factorization can be used to compute Z. Let

$$PA_{12}Q = LU \tag{20}$$

where $L^T = [\, L_1 \quad L_2 \,]$; $L_1^T \in \mathbb{R}^{n_n \times n_n}$ is lower triangular with a unit diagonal; $L_2 \in \mathbb{R}^{n_n \times (n_p - n_n)}$; $U \in \mathbb{R}^{n_n \times n_n}$ upper triangular (Benzi et al. 2005); and Z is as in Eq. (19).

The fill-reducing ordering and factorization methods of *SuiteSparse* (Davis 2004, 2006; Davis and Duff 1997), which are part of *MATLAB version 8.2. 0.701*'s sparse LU functions, are used to compute the LU factors. The resulting null basis is found to be sparser but of the same order of sparsity as the result of the method of Elhay et al. (2014). The two methods also produce similarly well-conditioned null bases.

The best conditioned null space can be computed using a QR factorization. Every full rank $A_{12} \in \mathbb{C}^{n_p \times n_n}$, $n_p \ge n_n$ has a full QR factorization:

$$A = [\, Q_1 \quad Q_2 \,] \begin{bmatrix} R \\ 0 \end{bmatrix}$$

where $Q = [\, Q_1 \quad Q_2 \,] \in \mathbb{C}^{n_p \times n_p}$ is unitary and $R \in \mathbb{C}^{n_n \times n_n}$ is upper triangular. Moreover, the factorization $A = Q_1 R$, with $R_{ii} > 0$ is

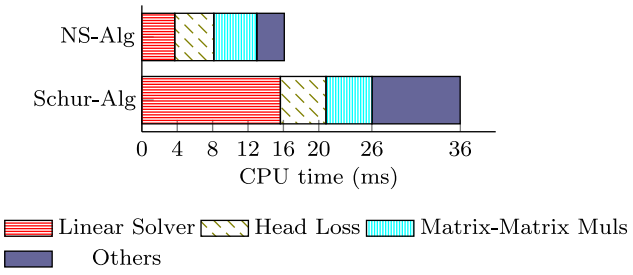

Fig. 1. Average CPU times (in ms) for null space and Schur algorithms; BWFLnet network

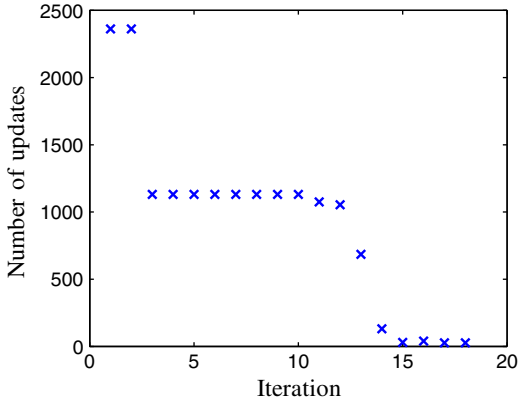

Fig. 2. Number of pipe flows that change by more than $\delta_N \times 1e^{-6}$ against iteration number, $\delta_N = 1e^{-6}$ is Newton tolerance

the unique Cholesky factor of $A_{12}^T A_{12}$ (Golub and Van Loan 1996, Section 5.2.6). Because the columns of Q_2 span $\ker(A_{12})$, $Z = Q_2$ such that $\kappa_2(Z'Z) = 1$.

In principle, the QR factorization also gives the Cholesky factor of $A_{12}^T A_{12}$ and so seems attractive. However, even with the sparsest QR factorizations, the bases are much more dense than those from an LU factorization. For example, for the BWFLnet network, which has 2,303 nodes and 2,362 links (see the relevant subsequent section for details of networks used and the algorithm implementations in *MATLAB version 8.2. 0.701*), the number of nonzeros in Z from a sparse LU, the method of Elhay et al. (2014), and *SuiteSparseQR* are 2,247, 2,278, and 33,041, respectively. The corresponding values of $\kappa_2(Z'Z)$ for the three methods are 291, 193, and 1, respectively. Because the QR basis is at least an order denser than the LU null basis, and the fact that the Cholesky factor of $A_{12}^T A_{12}$ needs to be computed only once anyway, the use of a QR-based basis is not proposed. Instead, the use of a sparse LU or the method of Elhay et al. (2014) is proposed for the computation of well-conditioned and sparse null bases.

Fig. 1 shows a comparison of the computational cost of the Schur and null space Newton algorithms using the example network BWFLnet. In addition, similar to the analysis done for the Schur (or GGA) method in Guidolin et al. (2013), Fig. 1 shows an analysis of the main computational blocks of both the Schur and null space algorithms. The contribution of each block to the total computational time is shown. It is apparent that the matrix-matrix multiplications for the linear solves, the linear solves, and the head-loss computations together constitute over 75% of the computational time. The Others block includes matrix-vector multiplications, residual error norm computations, Jacobian regularizations, and diagonal matrix inversions in the case of the Schur method, which can add up to a significant portion of the total CPU time. The linear solve time for the null space Newton method is much shorter because the linear system that changes at each Newton iteration, i.e., (16) and line 4 of Algorithm 1, is much smaller than that of the Schur method in Eq. (8). Moreover, the bigger of the two linear systems of the null space algorithm, (17) and line 8 of Algorithm 1, requires only a single factorization.

The computational cost of forming the matrix-matrix multiplications $Z^T F^k Z$ is smaller than that of $A_{12}^T F^k A_{12}$ for the Schur methods because Z has far fewer columns than A_{12}. Nonetheless, this cost is significant even for a null space method.

It is noted in Guidolin et al. (2013) that the Hazen-Williams head-loss computations take significant computational time; this is shown here to be the case in both the Schur and null space methods. In the next section, a novel partial update scheme is proposed to reduce the computational cost associated with head-loss computations and matrix-matrix multiplications. The partial update scheme will be defined and its convergence analysis presented.

Based on the partial updates, a stopping criteria heuristic for the null space method is also proposed, which will reduce the number of head computations in Eq. (17) or line 8 of Algorithm 1.

Partial Update Methods for Null Space Algorithm

Algorithm Derivation

One interesting aspect of the Newton method for solving WDNs is an appropriate choice of convergence criteria. From a number of simulation results, in Elhay and Simpson (2011) and the discussions on it (Kovalenko and Prokhorov 2013), it is noted that the flow continuity residual ($\|A_{12}^T q - d\|$) converges much faster than the energy residual ($\|A_{11}(q)q + A_{12}h + A_{10}h_0\|$). As a result, the two articles propose that a solution should not be accepted if only the flow continuity residual has converged—the residual of the whole nonlinear equations (2) should be used. For the null space algorithm, this led to the investigation of the convergence of flows. Because the flow continuity residual converges in a few iterations, so do the flows in most of the pipes. For the example network BWFLNet, Fig. 2 shows the number of flows that have not converged, in this case, the ones that change by more than $1e^{-12}$ m^3/s at each iteration. Therefore, it is proposed that computational savings can be made by updating the head losses (G^k and F^k), and corresponding elements of the matrices in (16), only for flows that have not converged in this sense.

The concept of a partial update set is introduced to start. Let the residual error tolerance for the Newton iterations be δ_N and let $0 < \varepsilon < 1$ be a small number. The (partial) update set is defined at the kth iteration as

$$\mathcal{U}^k(\varepsilon) := \{i \in \{1, \ldots, n_p\} : |s_i^k| := |q_i^{k+1} - q_i^k| \geq \varepsilon \delta_N\} \qquad (21)$$

where s_i^k is the Newton step in the flow of the ith pipe or link at iteration k.

At each Newton iteration, rather than recomputing all the frictional head losses and coefficients across the network of links $i = 1, \ldots, n_p$, the partial update formula uses the index set defined in Eq. (21) to update over the smaller set:

$$G_{ii}^{k+1} = r_i |q_i^{k+1}|^{n_i - 1}, \qquad \text{for all } i \in \mathcal{U}^k(\varepsilon) \qquad (22)$$

For DW head-loss models, the resistance coefficient is a function of flow, i.e., $r_i := r_i(q_i^{k+1})$, and is also updated in Eq. (22) for links in the update index set $\mathcal{U}^k(\varepsilon)$. Moreover, rather than

computing $X^k := Z^T F^k Z$ on line 4 of Algorithm 1, the following update formula is proposed:

$$X^k = X^{k-1} + \sum_{i \in \mathcal{U}^k} (f_i^k - f_i^{k-1}) z_i z_i^T \tag{23}$$

where z_i is the ith column of Z^T.

Convergence Analysis: Partial Updates As an Inexact Newton Method

Algorithm 1 solves, at each iteration k, the linear equation $\nabla f(x^k)(x^{k+1} - x^k) = -f(x^k)$ in (5). In exact Newton methods, these linear equations are solved exactly to sufficiently small tolerances using direct or iterative solvers; for example, $\|\nabla f(x^k)(x^{k+1} - x^k) + f(x^k)\| \le e_{\text{tol}} \|f(x^k)\|$, where e_{tol} can be as low as machine precision depending on the condition number of the problem. Solving these linear systems to high accuracy is the bottleneck of the Newton method (Nocedal and Wright 2006, Section 11.1).

Because this is often computationally expensive, inexact Newton methods solve the Newton equation only approximately to find a step $s^k = x^{k+1} - x^k$ that satisfies the milder condition

$$\|\nabla f(x^k) s^k + f(x^k)\| \le \eta^k \|f(x^k)\|, \quad \text{for some} \quad \eta^k \in [0, \eta] \tag{24}$$

where $\eta \in [0, 1)$ (Dembo et al. 1982). The parameter $\{\eta^k\}$ is referred to as the forcing sequence and determines how fast the Newton iterations converge. By using a sequence that gets smaller with iterations, the linear equations are solved with progressively smaller relative error as the iterates get closer to the solution. Iterative linear solvers are especially suited for this because, unlike direct methods, they allow early termination. With this condition only and the standard assumptions, Dembo et al. (1982) showed that local q-linear convergence can be achieved and that $\eta^k \to 0$ guarantees q-superlinear convergence locally (Kelley 2003, Thm.1.3) (Nocedal and Wright 2006, Thm.11.3).

In this article, the concern is not the use of inexact Newton methods with iterative solvers. Rather, the aim is to prove that the convergence of the Newton method is preserved under the partial Jacobian and function update of (22) and (23), respectively. This is shown through the update algorithm as an inexact Newton method.

Proposition 1 (partial-updates inexact Newton method): Assume the Newton method of Algorithm 1 with error tolerance δ_N is coupled with the partial update formulas for the head losses as in (22). Then, with the mild assumption that flows that have converged do not move away from the solution, there always exists $\varepsilon > 0$, and update set $\mathcal{U}^k(\varepsilon)$, such that the partial-update Newton scheme is an inexact Newton method, guaranteeing at least q-linear local convergence.

Proof: See Appendix II.

On Stopping Criteria for the Null Space Algorithm

From the null space formulation in Algorithm 1, the flow iterations are independent of the head values. By using the convergence of the flow conservation equations as stopping criteria, it may appear that the head needs to be solved for only once. However, the flow conservation equation often converges to within machine precision many iterations before the energy residual becomes small. In fact, this is satisfied from the second iterate because all feasible solutions of the Newton step should satisfy the continuity equation; see the second block row of (5). Therefore, the convergence of the flow continuity equation on its own should not be used as stopping criteria for the Newton iterations; this is also shown in Elhay and Simpson (2011). It is necessary to compute the head at each iteration to determine convergence using either the residual of the entire nonlinear equation (2), or convergence of nodal head differences as proposed by Elhay and Simpson (2011) and Kovalenko and Prokhorov (2013).

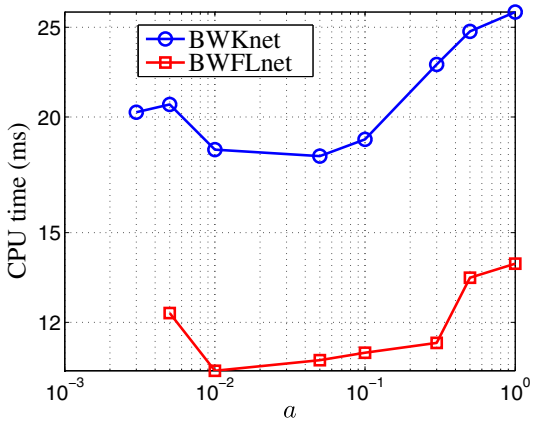

Fig. 3. CPU time against head calculation trigger a

Because computing the heads at each iteration and the associated norm of the residual together have significant computational cost, new stopping criteria are proposed that will allow fewer head computations in line 8 of Algorithm 1. When the null space algorithm with the efficient partial update formulas is considered, a large proportion of the flows will have converged, i.e., the update set $\mathcal{U}$ is small, when the Newton error tolerance is reached. Therefore, the overhead in computing the pressure heads and associated error norms can be reduced by computing them only when the fraction of unconverged flows is small.

Traditionally, in open-source software like *EPANET version 2.0*, a pragmatic convergence test is applied based on the sum of all link flow changes as a proportion of the sum of all the link flow rates (Rossman 2000). The cardinality of the index set $|\mathcal{U}^k|$ is directly related to the change in all flows. The number of flows in the partial update set quantifies flows whose change is greater than $\varepsilon \times \delta_N$; as such it relates the relative change in flows to the required Newton tolerance in an automatic way. A bound on the number of elements in the update set, $|\mathcal{U}^k| < a \times n_p$, is proposed as a flag for the onset of pressure head computations. Fig. 3 shows the computational times achieved for values $a = 0.1$–50%, each averaged over 1,000 simulations with the example network BWFLnet. As can be seen from Fig. 3, a smaller value for a means fewer head and error computations. However, if a is too small, it will lead to having many more Newton iterations than necessary because residual errors are not checked early enough, and so increase computations. The figure shows a similar profile for the example network BWKnet; see the subsequent section for details of networks used. From these and various other simulations in the case study, using $a = 0.01$–$0.1 \sim n_l/n_p$, i.e., $|\mathcal{U}^k| < n_l$, seems to give a good compromise. For example, it is shown in Fig. 3 that this range of values for a reduces computational cost by over 20%.

Fig. 4 compares the null space algorithm as described in Algorithm 1, called NSM1 here, with its modified versions with the proposed partial update scheme only (NSM2), and one with both a partial update scheme and a-parameterized head-loss computations (NSM3). As expected, in going from NSM1 to NSM2, the partial updates reduce the CPU times for head-loss

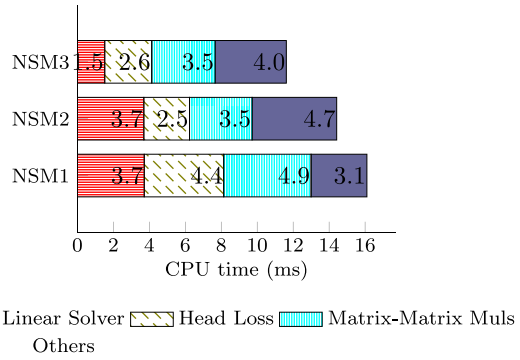

Fig. 4. Average CPU times (in ms) for various null space algorithms; NSM1, NSM2, and NSM3 refer to Algorithm 1, Algorithm 1 with the partial updates only, and Algorithm 1 with both the partial updates and a-parameterized head computations, respectively; example network BWFLnet is used with $a = 0.05$

and matrix-matrix computations, incurring a bit more overhead in the process of indexing the partial update set (21) but with an overall CPU time reduction for the algorithm. Comparing NSM2 to NSM3, Fig. 4 clearly demonstrates the impact of solving fewer head equations in reducing overall CPU time for solving linear systems in Algorithm 1; here solve time for linear systems is reduced by approximately 60%. The overheads in the Others are also reduced; a significant reason is the reduction in convergence checks and associated error norm computations.

Case Study/Simulation Results

Four operational water network models were used to analyze the effectiveness of the proposed null space methods. The networks range in size from small to large and have varying levels of loopedness as measured by the average degree of the graph, i.e., the average number of pipes incident at the nodes. Generally, the densest of water networks are still sparse in the mathematical sense. Networks BWFLnet and BWKnet represent a typical network in a built-up (urban) area in England, United Kingdom. These models are part of an extensive experimental program on the dynamic sectorization of water supply networks carried out by the InfraSense Labs in partnership with a United Kingdom water utility (Wright et al. 2014). A very dense city network is represented by NYnet (Ostfeld et al. 2008); this is similar to the densest in the literature (Giustolisi and Moosavian 2014). The relatively smaller size network C-town (Giustolisi and Moosavian 2014), called CTnet here, is also used. The relevant topological characteristics of the networks are as follows:

- CTnet: 396 nodes and 444 pipes (average degree = 2.24, $n_l/n_p = 0.11$).
- BWFLnet: 2,303 nodes and 2,362 pipes (average degree = 2.051, $n_l/n_p = 0.025$).
- BWKnet: 4,577 nodes and 4,648 pipes (average degree = 2.055, $n_l/n_p = 0.015$).
- NYnet: 12,527 nodes and 14,831 pipes (average degree = 2.37, $n_l/n_p = 0.155$).

All computations were performed within *MATLAB version 8.2.0.701 R2013b*-64 bit for Windows 7.0 installed on a 2.4 GHz Intel Xeon(R) CPU E5-2665 0 with 16 cores. To make the CPU time profiling most accurate, the number of active CPUs used by *MATLAB* was set to one before starting profiling. This prevents spurious results from the use of multiple cores by some of the solvers used. For example, the approximate minimum degree (AMD) ordering and its variants (minimum fill, column minimum degree ordering, etc.) and graphs-based permutations used in the sparse Cholesky, LU, and QR factorizations and solves, within *MATLAB version 8.2.0.701* and *SuiteSparse*, take advantage of parallelizing work over multiple cores; these should be disabled to make a fairer comparison of the proposed algorithms. Moreover, a large number of simulations (1,000) was used to analyze each case study because small variations in task scheduling by the processor could result in variations not caused by computational complexity only. The numerical tests were performed by randomly varying the demands from the typical diurnal demand profile. As in Elhay et al. (2014), Giustolisi and Moosavian (2014), and other referenced literature, all analysis presented here does not consider control devices like pumps and check valves.

These tests were done using *MATLAB* and therefore the computational times can be reduced significantly by implementing in a lower-level programming language. The trends in the results, nonetheless, should be valid generally. Future work includes the implementation of these methods in C++.

The results of Table 1 demonstrate the trends observed in Figs. 1 and 4. The null space algorithms reduce average CPU time for all the given networks, the highest being by around 68% for BWKnet. As expected from Algorithm 1, null space methods have the biggest impact when the network is not highly looped, i.e., $n_l \ll n_p$. This is apparent in the results because the least dense networks, BWFLnet and BWKnet, have the highest reduction in CPU time. For CTnet and NYnet, the null space algorithms have a relatively smaller reduction in CPU time, up to 34% and 40%, respectively. In Creaco and Franchini (2013), it is shown that for networks with n_l/n_p very close to 1, the Schur (GGA) method performs as well as a loop-based method. However, unlike in Fig. 5 of Creaco and Franchini (2013), as shown in the case studies here and studies in Elhay et al. (2014), the most meshed of real water network models have values for n_l that are far less than n_p. For such real network models, null space algorithms do outperform a Schur algorithm as confirmed by the trends in Table 1.

The trends for the null space algorithms NSM1, NSM2, and NSM3 (defined in the previous section) also demonstrate the relative savings made using the novel partial updates and the new heuristic to avoid computing pressure head values and convergence

Table 1. Mean CPU Times for Various Hydraulic Solvers Applied to Networks of Different Size and Connectivity

			CPU times (ms)				% improvement over Schur		
Network	n_p	Degree	Schur	NSM1	NSM2	NSM3	NSM1	NSM2	NSM3
CTnet	444	2.24	8.04	6.05	6.18	5.28	24.75	23.16	34.29
BWFLnet	2,362	2.05	27.86	16.06	12.59	9.90	42.34	54.81	64.46
BWKnet	4,648	2.05	46.0	21.9	18.3	15.5	52.4	60.2	68.38
NYnet	14,831	2.37	738.6	677.7	520.6	440.8	8.25	29.5	40.3

Note: Accuracy and partial update set parameters were set to $\delta_N = 1e^{-6}$ and $\varepsilon = 1e^{-6}$, respectively.

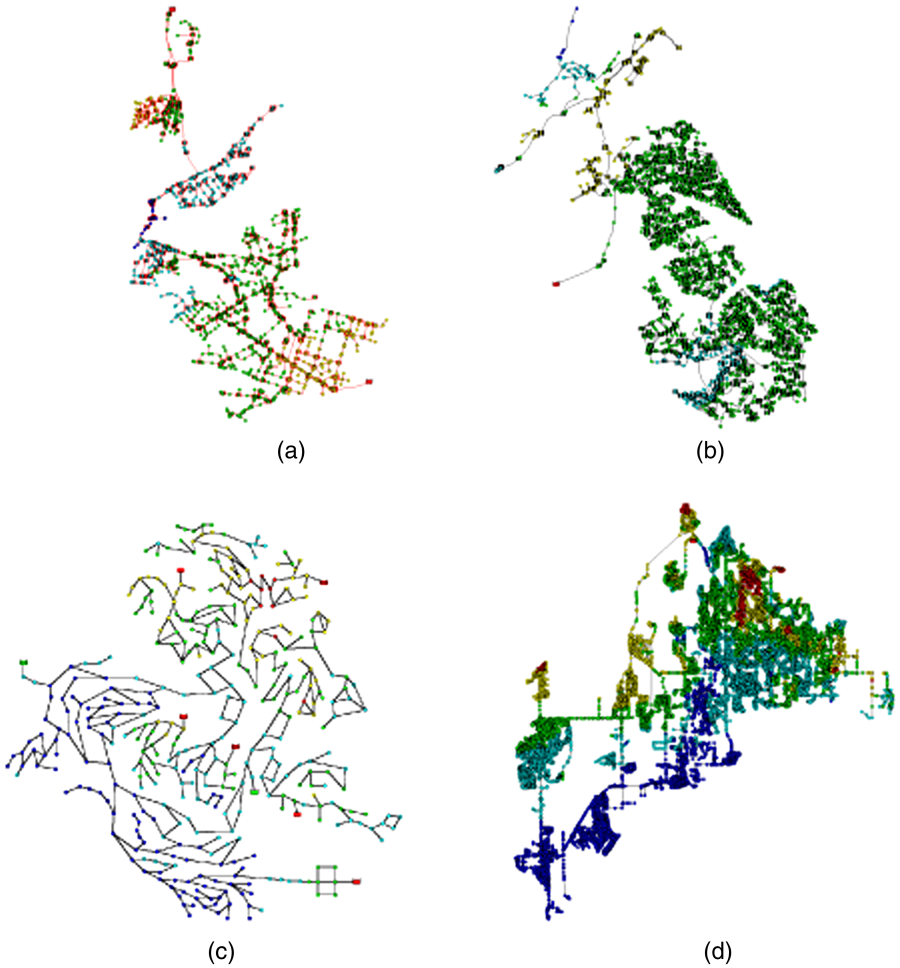

Fig. 5. Graphs of networks used in numerical study: BWFLnet, BWKnet, CTnet, NYnet

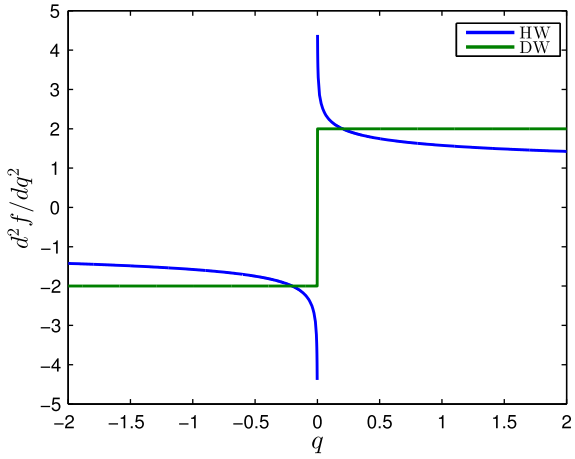

Fig. 6. Second derivative of head-loss functions when HW and DW models are used

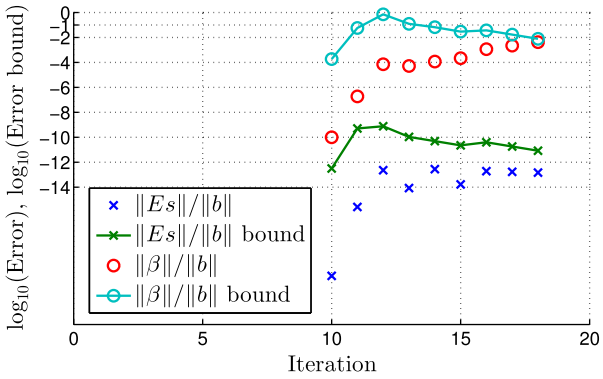

Fig. 7. Error components $\|Es\|/\|b\|$ and $\|\beta\|/\|b\|$ in (30), and their bounds derived in Eqs. (32) and (33), respectively, plotted against iteration number for the network BWFLnet; settings $\delta_N = 1e^{-6}$, $\varepsilon = 1e^{-6}$ are used

checks at each Newton iteration. It is also noted that the CPU time for networks increases by less than linearly as the model size (n_p) increases, even less so for the null space algorithms. It is noted here that the NYnet network takes significantly more CPU time for the same accuracy level, set here to $\delta_N = 1e^{-6}$, because the model is badly conditioned.

Conclusion

Because water supply systems are increasingly dependent upon resilient and efficient near-real-time management, hydraulic solvers are needed with much improved computational efficiency and robustness. In this article, the application of novel efficient null space algorithms for hydraulic analysis of large-scale water distribution

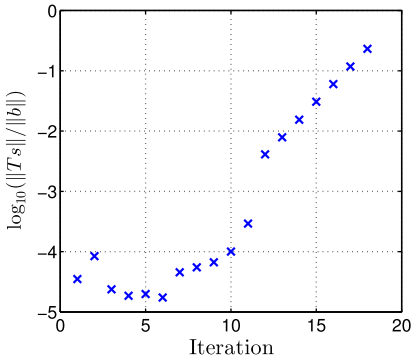

Fig. 8. Linear system error norm, $\|Ts\|/\|b\|$, due to the regularization (14), plotted against iteration number for the network BWFLnet; settings are as in Fig. 7, with condition number bound $\kappa = 1e^5$

networks is described. An extended literature review of emerging mathematical and computing advances in hydraulic analysis has been given. Because the linear system of equations inside a Newton solver for hydraulic analysis belongs to the class of sparse saddle point problems, both the structure and sparsity of the equations have been exploited to propose more efficient algorithms. New proofs have been derived to show the convergence of the various Newton methods used. The main contributions of this paper are summarized as follows.

1. A sparse null space algorithm with improved efficiencies was proposed and used to solve the Newton equations with fewer computational resources and more robustly than the equivalent Schur scheme.
2. New proofs have been derived to show the convergence of the Newton method when both Hazen-Williams and Darcy-Weisbach models are used.
3. A partial update method has been proposed for the null-space algorithm to further reduce computational complexity. This allows repeated computation of head losses to be avoided for links whose flows have already converged. The head-loss computations with rational powers take a significant fraction of total flops used in the whole iterative processes.
4. Because sparse fundamental null bases are used, the linear systems for the cotree flow calculations with a direct solver are also sparse. These linear systems are analyzed to study the convergence properties of the null space algorithm when partial updates of head losses are used. New mathematical proofs have been derived to show sufficient conditions under which the null space algorithm with partial updates stays convergent.

   A Jacobian regularization scheme from Elhay and Simpson (2011) has also been adopted to improve the condition number of network problems with zero flows. New conditions of convergence have been derived for the partial updates and Jacobian regularizations by deriving forcing functions and conditions under which the algorithm stays an inexact Newton method.
5. Based on the partial update sets for the null space algorithm, a new heuristic has also been proposed to avoid computing pressure head values and convergence checks at each Newton iteration. This has been shown to reduce computational cost.
6. Finally, case studies with real large-scale water distribution networks, of various sizes, have been used to demonstrate the novel methods for hydraulic analysis. Computational savings of up to 68% have been shown for the case study networks.

Appendix I. Differentiability of Head-Loss Formulas

For a (scalar) flow $q \in \mathbb{R}$, the Hazen-Williams and other loss equations are of the form $f(q) := r|q|^{n-1}q$, $1 \le n \le 2$. That is

$$f(q) = \begin{cases} r(-q)^{n-1}q & \text{if } q < 0, \\ r(q)^{n-1}q & \text{if } q \ge 0 \end{cases} \tag{25}$$

Here no assumptions are made on the value of n; it can take the value $n \sim 1.852$ for the Hazen-Williams case or $n = 1$, $n = 2$ for Darcy-Weisbach. By simple application of the chain rule separately for the cases $q \le 0$ and $q > 0$, the following is obtained:

$$f'(q) = \begin{cases} r(-q)^{n-1} - r(n-1)(-q)^{n-2}q = nr(-q)^{n-1} & \text{if } q < 0, \\ r(q)^{n-1}1 + r(n-1)(q)^{n-2}(q) = nr(q)^{n-1} & \text{if } q \ge 0, \end{cases}$$
$$= nr|q|^{n-1}$$

Although the derivative of the loss $f'(q)$ is continuous, it is not Lipschitz at zero—the second derivative $f''(q)$ may not be defined at zero:

$$f''(q) = \begin{cases} -n(n-1)r(-q)^{n-2}, & \text{if } q < 0, \\ n(n-1)r(q)^{n-2}, & \text{if } q > 0, \end{cases}$$
$$= \text{sign}(q)n(n-1)r|q|^{n-2}$$

For loss models with $n < 2$, the second derivative $f''(q)$ is not defined at zero, i.e., it becomes unbounded as $q = 0$ is approached from the left or right. For example, $n = 1.852$ for the Hazen-Williams loss formulas for rough flows. If the domain $D = \{q: 0 < \delta \le |q| \le \Delta < \infty\}$ is considered, $f''(q)$ is bounded and continuous in D. By the mean value theorem, it is then straightforward to show that $f'(q)$ is locally Lipschitz (i.e., Lipschitz in D) (Eriksson et al. 2004, p. 372). The continuous differentiability of the vector function $f(\cdot)$ in (2) is then implied by the elementwise continuous differentiability shown previously.

For Darcy-Weisbach models, $n = 2$ for rough flows (Mays 2000, Section 2.6) and $n = 1$ in the laminar flow range (Elhay and Simpson 2011). Because small or zero flows are laminar, if Darcy-Weisbach is used for small flows as proposed in Elhay and Simpson (2011), $f''(q)$ stays bounded (for finite flows $q < \infty$) and is continuous. Even in the case in which a rough flow model with $n = 2$ is adopted, $f''(q)$ stays bounded and is continuous almost everywhere; see Fig. 6. Both of these imply Lipschitz continuity of $f'(q)$ in the real range of flow $D = \{q: 0 \le |q| \le \Delta < \infty\}$. This way, it is possible to guarantee Lipschitz continuity of the head-loss function derivative, and so that of the Jacobian matrix in (5).

Appendix II. Proof of Proposition 1

In the exact Newton method, the linear system $\nabla f(x^k)s^k = -f(x^k)$ is solved to find the step at each iteration, $s^k := x^{k+1} - x^k$. Let $A^k := \nabla f(x^k)$ and $b^k := f(x^k)$. From here on, the superscript k is dropped in some places for convenience of notation. By Lemma 1, Algorithm 1 is a Newton method.

If Algorithm 1 is coupled with the update formulas (22) and (23), both the Jacobian A and the function evaluation b will have errors due to the unupdated head losses. Therefore, an approximate problem is solved:

$$\tilde{A}\,\tilde{s} + \tilde{b} = 0, \qquad \tilde{A} = A + E, \quad \tilde{b} = b + \beta \tag{26}$$

where E and β are the errors in the Jacobian and function evaluations, respectively, and $\tilde{s}$ is the approximate (or inexact) Newton step.

Assume this perturbed linear system is solved to sufficiently small error tolerance $e_{\text{tol}} \ll 1$, i.e., exactly (using a direct solver), such that

$$\tilde{A}\,\tilde{s} + \tilde{b} = v, \qquad \|v\| \le e_{\text{tol}}\|\tilde{b}\| \tag{27}$$

Substituting for $\tilde{A}$ and $\tilde{b}$ and rearranging, the residual for the Newton method results:

$$A\tilde{s} + b = v - E\tilde{s} - \beta \tag{28}$$

This implies

$$\frac{\|A\tilde{s} + b\|}{\|b\|} \le \frac{\|v\|}{\|b\|} + \frac{\|E\tilde{s}\|}{\|b\|} + \frac{\|\beta\|}{\|b\|} \tag{29}$$

$$\le e_{\text{tol}} + (1 + e_{\text{tol}})\frac{\|\beta\|}{\|b\|} + \frac{\|E\tilde{s}\|}{\|b\|}, \tag{30}$$

because $\|v\| \le e_{\text{tol}}\|\tilde{b}\| \le e_{\text{tol}}(\|b\| + \|\beta\|)$.

A forcing sequence is defined for the Newton method with the inexact linear solve process (28). From the relative error bound (29), the following is defined:

$$\eta^k := \frac{\|v^k\|}{\|b^k\|} + \frac{\|E^k\tilde{s}^k\|}{\|b^k\|} + \frac{\|\beta^k\|}{\|b^k\|}$$

It is shown here that the forcing term can satisfy the condition $\eta^k \in [0, \eta], \eta \in [0, 1)$ with some mild assumptions. With the tolerance for the Newton method convergence set to δ_N (i.e., $\|b^k\| \ge \delta_N$, $\forall k$), the nonupdate set $\mathcal{N}^k(\varepsilon) := \{i \in \{1, \ldots, n_p\} : |s_i^k| := |q_i^{k+1} - q_i^k| < \varepsilon\delta_N\}$ is defined. Now consider the matrix E: deriving its explicit form from (5), the following results:

$$\|E\tilde{s}\|_\infty = \|(A - \tilde{A})\tilde{s}^k\|_\infty = \max_{i \in \mathcal{N}^k} |r_i\tilde{s}_i(|q_i^k|^{n_i - 1} - |\tilde{q}_i^k|^{n_i - 1})| \tag{31}$$

where $\tilde{q}^k$ are the unupdated flows in the head-loss calculations, and q^k are the actual flow values at the kth iteration.

Assumption: The steps $s_i^k \in \mathcal{N}^k(\varepsilon)$ will stay small for k large enough; that is, flows that have converged do not move away from the solution by more than $\varepsilon\delta_N$.

With the previous assumption, the following results:

$$\frac{\|E\tilde{s}\|}{\|b\|} \le \varepsilon\delta_N \frac{\max_{i \in \mathcal{N}^k} |r_i(|q_i|^{n_i - 1} - |\tilde{q}_i|^{n_i - 1})|}{\delta_N} \tag{32}$$

If the perturbation on the function evaluations β is considered, the following results from (5):

$$\frac{\|\beta\|}{\|b\|} \le \frac{\max_{i \in \mathcal{N}^k} |r_i(|q_i|^{n_i - 1} - |\tilde{q}_i|^{n_i - 1})q_i|}{\delta_N} \tag{33}$$

Putting the two bounds in (32) and (33) together, for $e_{\text{tol}} \ll 1$, the conditions

$$\varepsilon r_i(|q_i|^{n_i - 1} - |\tilde{q}_i|^{n_i - 1})| < 1/2 - 3e_{\text{tol}}/2 \quad \text{and}$$
$$r_i|(|q_i|^{n_i - 1} - |\tilde{q}_i|^{n_i - 1})q_i| < \delta_N/2$$

are sufficient to guarantee that η^k in $[0, 1)$. Because $(|q_i|^{n_i - 1} - |\tilde{q}_i|^{n_i - 1})$ can be made arbitrarily close to zero by choosing the update parameter ε to be sufficiently small, both (32) and (33) can always be guaranteed without making any further assumptions.

Remarks

The values of $\|Es\|/\|b\|$ and $\|\beta\|/\|b\|$ in (30), and their bounds derived in (32) and (33), were plotted for the example network BWFLNet. Fig. 7 shows that $\|Es\|/\|b\|$ and $\|\beta\|/\|b\|$ are very small (here zero) until the Newton iteration gets close to the solution, where these values increase toward their bounds as $\|b\|$ becomes small near convergence. It is also the case that the bounds stay small in absolute values, implying $\eta^k \ll 1$, $\forall k$. By using the smaller update parameter ε, these bounds can be made even smaller. However, as $\varepsilon \to 0$, the partial-update inexact Newton method approaches the original Newton method.

Regularization As an inexact Newton method

Here the regularization scheme of Elhay and Simpson (2011) used as in (11) is considered. Similarly to the errors in the unupdated head losses, E in (26), the regularization matrix T perturbs the linear problem solved by the Newton method, i.e., $\tilde{A} = A + E + T$. Because the matrix T is determined only by the condition number constraint (14), unlike the term $\|E^k\tilde{s}^k\|/(\|b^k\|)$, the term $\|T^k\tilde{s}^k\|/(\|b^k\|)$ cannot be directly controlled to stay small using the update set parameter ε. However, simulations for multiple networks seem to indicate it stays very small far away from the solution, i.e., $\|T^k\tilde{s}^k\|/(\|b^k\|) \ll 1$, and below 1 near the solution.

If Jacobian regularization is also coupled with the partial-update formulas, with

$$\eta^k := \frac{\|v^k\|}{\|b^k\|} + \frac{\|E^k\tilde{s}^k\|}{\|b^k\|} + \frac{\|\beta^k\|}{\|b^k\|} + \frac{\|T^k\tilde{s}^k\|}{\|b^k\|}$$

simulations seem to indicate that η^k is bounded below 1 and so the method is still an inexact Newton method and guarantees at least linear convergence. For the same example as in Fig. 7, the bound on the error due to the regularization scheme is shown in Fig. 8.

Acknowledgments

This work was supported by the NEC-Imperial "Big Data Technologies for Smart Water Networks" project.

References

Benzi, M., Golub, G. H., and Liesen, J. (2005). "Numerical solution of saddle point problems." *Acta Numerica*, 14(1), 1–137.

Berghout, B. L., and Kuczera, G. (1997). "Network linear programming as pipe network hydraulic analysis tool." *J. Hydraul. Eng.*, 10.1061/(ASCE)0733-9429(1997)123:6(549), 549–559.

Bhave, P. R. (1988). "Calibrating water distribution network models." *J. Environ. Eng.*, 10.1061/(ASCE)0733-9372(1988)114:1(120), 120–136.

Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*, Cambridge University Press, Cambridge, U.K.

Brunone, B., et al., eds. (2013). "Informatics for water systems and smart cities." *12th Int. Conf. on Computing and Control for the Water Industry, CCWI2013, Informatics for Water Systems and Smart Cities*, Univ. of Perugia, Perugia, Italy.

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

Creaco, E., and Franchini, M. (2013). "Comparison of Newton-Raphson global and loop algorithms for water distribution network resolution." *J. Hydraul. Eng.*, 10.1061/(ASCE)HY.1943-7900.0000825, 313–321.

Crous, P., Van Zyl, J., and Roodt, Y. (2012). "The potential of graphical processing units to solve hydraulic network equations." *J. Hydroinf.*, 14(3), 603–612.
Davis, T. A. (2004). "Algorithm 832: Umfpack v4. 3—An unsymmetric-pattern multifrontal method." *ACM Trans. Math. Software*, 30(2), 196–199.
Davis, T. A. (2006). *Direct methods for sparse linear systems*, Vol. 2, Society for Industrial and Applied Mathematics, Philadelphia.
Davis, T. A., and Duff, I. S. (1997). "An unsymmetric-pattern multifrontal method for sparse lu factorization." *SIAM J. Matrix Anal. Appl.*, 18(1), 140–158.
Dembo, R. S., Eisenstat, S. C., and Steihaug, T. (1982). "Inexact Newton methods." *SIAM J. Numer. Anal.*, 19(2), 400–408.
Dennis, J. E., Jr., and Schnabel, R. B. (1996). *Numerical methods for unconstrained optimization and nonlinear equations*, Vol. 16, Society for Industrial and Applied Mathematics, Philadelphia.
Eck, B., and Mevissen, M. (2013). "Fast non-linear optimization for design problems on water networks." *World Environmental and Water Resources Congress 2013*, ASCE, Reston, VA, 696–705.
Elhay, S., and Simpson, A. R. (2011). "Dealing with zero flows in solving the nonlinear equations for water distribution systems." *J. Hydraul. Eng.*, 10.1061/(ASCE)HY.1943-7900.0000411, 1216–1224.
Elhay, S., Simpson, A. R., Deuerlein, J., Alexander, B., and Schilders, W. (2014). "A reformulated co-tree flows method competitive with the global gradient algorithm for solving the water distribution system equations." *J. Water Resour. Plann. Manage.*, 10.1061/(ASCE)WR.1943-5452.0000431, 04014040.
EPANET version 2.0 [Computer software]. U.S. Environmental Protection Agency, Washington, DC.
Eriksson, K., Estep, D., and Johnson, C. (2004). *Applied mathematics: Body and soul: Volume 1: Derivatives and geometry in IR3*, Vol. 1, Springer Science & Business Media, Berlin, Heidelberg.
Giustolisi, O., Laucelli, D., Berardi, L., and Savić, D. A. (2011). "Computationally efficient modeling method for large water network analysis." *J. Hydraul. Eng.*, 10.1061/(ASCE)HY.1943-7900.0000517, 313–326.
Giustolisi, O., and Moosavian, N. (2014). "Testing linear solvers for global gradient algorithm." *J. Hydroinf*, 16(5), 1178–1193.
Giustolisi, O., Savic, D., and Kapelan, Z. (2008). "Pressure-driven demand and leakage simulation for water distribution networks." *J. Hydraul. Eng.*, 10.1061/(ASCE)0733-9429(2008)134:5(626), 626–635.
Giustolisi, O., and Walski, T. (2011). "Demand components in water distribution network analysis." *J. Water Resour. Plann. Manage.*, 10.1061/(ASCE)WR.1943-5452.0000187, 356–367.
Golub, G. H., and Van Loan, C. F. (1996). *Matrix computations*, 3rd Ed., John Hopkins University Press, Baltimore, London.
Gorev, N. B., Kodzhespirov, I. F., Kovalenko, Y., Prokhorov, E., and Trapaga, G. (2012). "Method to cope with zero flows in Newton solvers for water distribution systems." *J. Hydraul. Eng.*, 10.1061/(ASCE)HY.1943-7900.0000694, 456–459.
Guidolin, M., Burovskiy, P., Kapelan, Z., and Savić, D. (2010). "Cwsnet: An object-oriented toolkit for water distribution system simulations." *WDSA 2010 Conf.*, ASCE, Reston, VA.
Guidolin, M., Kapelan, Z., and Savić, D. (2013). "Using high performance techniques to accelerate demand-driven hydraulic solvers." *J. Hydroinf.*, 15(1), 38–54.
Higham, N. J. (1996). *Accuracy and stability of numerical algorithms*, Society for Industrial and Applied Mathematics, Philadelphia.
Jankovic-Nisic, B., and Chan, A. (2013). "London strategic model." ⟨http://www.waterprojectsonline.com⟩ (May 3, 2014).
Kelley, C. T. (2003). *Solving nonlinear equations with Newton's method*, Vol. 1, Society for Industrial and Applied Mathematics, Philadelphia.
Kovalenko, Y., and Prokhorov, E. (2013). "Discussion of dealing with zero flows in solving the nonlinear equations for water distribution systems by Sylvan Elhay and Angus R. Simpson." *J. Hydraul. Eng.*, 10.1061/(ASCE)HY.1943-7900.0000568, 557–558.
Mair, M., Sitzenfrei, R., Kleidorfer, M., and Rauch, W. (2014). "Performance improvement with parallel numerical model simulations in the field of urban water management." *J. Hydroinf.*, 16(2), 477–486.
MATLAB version 8.2. 0.701 [Computer software]. MathWorks, Natick, MA.
Mays, L. W. (2000). *Water distribution systems handbook*, McGraw-Hill, New York, London.
Nicolini, M., and Zovatto, L. (2009). "Optimal location and control of pressure reducing valves in water networks." *J. Water Resour. Plann. Manage.*, 10.1061/(ASCE)0733-9496(2009)135:3(178), 178–187.
Nielsen, H. B. (1989). "Methods for analyzing pipe networks." *J. Hydraul. Eng.*, 10.1061/(ASCE)0733-9429(1989)115:2(139), 139–157.
Nocedal, J., and Wright, S. J. (2006). *Numerical optimization*, Springer, New York.
Ostfeld, A., et al. (2008). "The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms." *J. Water Resour. Plann. Manage.*, 10.1061/(ASCE)0733-9496(2008)134:6(556), 556–568.
Pearson, J. W., Stoll, M., and Wathen, A. J. (2012). "Regularization-robust preconditioners for time-dependent PDE-constrained optimization problems." *SIAM J. Matrix Anal. Appl.*, 33(4), 1126–1152.
Pearson, J. W., and Wathen, A. J. (2012). "A new approximation of the Schur complement in preconditioners for PDE-constrained optimization." *Numer. Linear Algebra Appl.*, 19(5), 816–829.
Rahal, H. (1995). "A co-tree flows formulation for steady state in water distribution networks." *Adv. Eng. Software*, 22(3), 169–178.
Rossman, L. A. (2000). *EPANET 2: Users manual*, Water Supply and Water Resources Division, Cincinnati.
Savic, D. A., and Walters, G. A. (1997). "Genetic algorithms for least-cost design of water distribution networks." *J. Water Resour. Plann. Manage.*, 10.1061/(ASCE)0733-9496(1997)123:2(67), 67–77.
Sensus. (2012). "White paper: Water 20/20, bringing smart water networks into focus." SENSUS, Raleigh, NC.
Simpson, A., and Elhay, S. (2010). "Jacobian matrix for solving water distribution system equations with the Darcy-Weisbach head-loss model." *J. Hydraul. Eng.*, 10.1061/(ASCE)HY.1943-7900.0000341, 696–700.
SuiteSparse version 4.1.2 [Computer software]. Texas A&M Univ., TX.
Todini, E., and Pilati, S. (1988). "A gradient algorithm for the analysis of pipe networks." *Computer application in water supply, Vol. I—System analysis and simulation*, B. Coulbeck and O. Choun-Hou, eds., Wiley, London, 1–20.
Todini, E., and Rossman, L. A. (2012). "Unified framework for deriving simultaneous equation algorithms for water distribution networks." *J. Hydraul. Eng.*, 10.1061/(ASCE)HY.1943-7900.0000703, 511–526.
Wright, R., Stoianov, I., Parpas, P., Henderson, K., and King, J. (2014). "Adaptive water distribution networks with dynamically reconfigurable topology." *J. Hydroinf*, 16(6), 1280–1301.
Zecchin, A. C., Thum, P., Simpson, A. R., and Tischendorf, C. (2012). "Steady-state behavior of large water distribution systems: Algebraic multigrid method for the fast solution of the linear step." *J. Water Resour. Plann. Manage.*, 10.1061/(ASCE)WR.1943-5452.0000226, 639–650.