Sylvan Elhay, Angus R. Simpson, Jochen Deuerlein, Bradley Alexander and Wil H. A. Schilders

Reformulated co-tree flows method competitive with the global gradient algorithm for solving water distribution system equations

Journal of Water Resources Planning and Management, 2014; OnlinePubl(12):04014040-1-04014040-10

http://dx.doi.org/10.1061/(ASCE)WR.1943-5452.0000431

Reformulated Co-tree flows method competitive with the Global Gradient Algorithm for solving the water distribution system equations

by

Elhay, S., Simpson, A.R., Deuerlein, J., Alexander, B. and Schilders, W.H.A.

Journal of Water Resources Planning and Management

A Reformulated Co-tree Flows Method competitive with the Global Gradient Algorithm for solving the water distribution system equations.

Sylvan Elhay[1] Angus R. Simpson[2] Jochen Deuerlein[3] Bradley Alexander[4]
Wil H.A. Schilders[5]

December 28, 2013

Abstract

Many different methods have been devised to solve the non-linear systems of equations which model water distribution networks. Probably the most popular is the Global Gradient Algorithm (GGA) of Todini and Pilati. In the face of the GGA's success, alternative methods have not aroused much interest. One example is the Co-tree method which requires some cumbersome steps in its implementation.

In this paper a Reformulated Co-Trees Method (RCTM) is presented that simplifies the procedure by manipulating the incidence matrix into trapezoidal form: a lower triangular block at the top representing a spanning tree and rectangular block below it representing the corresponding co-tree. This reordering leads to significant efficiencies which make the RCTM competitive with the GGA in certain settings.

The new method has some similarities to the Loop Flows Corrections formulation and it is is shown, by application to a set of eight case study networks with between 932 and 19,647 pipes and between 848 and 17971 nodes, to be between 15% and 82% faster than the GGA in a setting,

[1] Visiting Research Fellow, School of Computer Science, University of Adelaide, South Australia, 5005.

[2] Professor, School of Civil, Environmental and Mining Engineering, University of Adelaide, South Australia, 5005.

[3] Senior Researcher, 3S Consult GmbH, Karlsruhe, Germany & Adjunct Senior Lecturer, School of Civil, Environmental and Mining Engineering, University of Adelaide, South Australia, 5005.

[4] Lecturer, School of Computer Science, University of Adelaide, South Australia, 5005.

[5] Professor, Department of Mathematics and Computer Science, TU Eindhoven, Eindhoven, The Netherlands.

such as optimization using evolutionary algorithms, where the methods are applied hundreds of thousands, or even millions, of times to networks with the same topology. It is shown that the key matrix for the RCTM can require as little as 7% of the storage requirements of the corresponding matrix for the GGA. This can allow for the solution of larger problems by the RCTM than might be possible for the GGA in the same computing environment.

Unlike some alternatives to the GGA, the following features, make the RCTM attractive: (i) it does not require a set of initial flows which satisfy continuity, (ii) there is no need to identify independent loops or the loops incidence matrix, (iii) a spanning tree and co-tree can be found from the incidence matrix without the addition of virtual loops, particularly when multiple reservoirs are present, (iv) it does not require the addition of a ground node and pseudo-loops for each demand node and does not require the determination of cut-sets.

In contrast to the GGA, the RCTM does not require special techniques to handle zero flow problems which can occur when the head loss is modeled by the Hazen-Williams formula (a sufficient condition is given).

The paper also (i) reports a comparison of the sparsity of the key RCTM and GGA matrices for the case study networks, (ii) shows mathematically why the RCTM and GGA always take the same number of iterations and produce precisely the same iterates, (iii) establishes that the Loop Flows Corrections and the Nullspace methods (previously shown by Nielsen to be equivalent) are actually identical to the RCTM.

INTRODUCTION

A quarter of a century ago Todini & Pilati (1988) introduced the Global Gradient Algorithm (GGA) for solving water distribution system (WDS) equations. Almost twenty years later, Todini (2006) summarized the popularity of the GGA in comparison to other available approaches when he wrote "…the practical success of the Global Gradient algorithm as programmed in EPANET 2 (Rossman 2000) leaves no doubts that the easiness of the approach that does not require neither a topological analysis aimed at determining the appropriate independent loops nor the need for an initial balanced solution, make it the most appropriate fast convergent and robust tool for pipe network analysis."

The speed with which the GGA executes the Newton iterations has probably contributed most to the method's popularity. The GGA determines the solution of a non-linear system of dimension $n_p + n_j$, where n_p is the number of pipes and n_j is the number of nodes at which the heads are unknown, by a two stage iteration in which the linear solver deals with a matrix of dimension only n_j. This, together with the fact that the matrix to be inverted is sparse and symmetric, leads to a very fast algorithm.

The two points made by Todini about the need for the analysis to find loops and an initial, balanced solution were aimed at the Simultaneous Loop Flows Corrections method of Epp & Fowler (1970). That method requires the addition of virtual loops when multiple reservoirs are present (a process improved by the techniques in the recent paper by Creaco & Franchini (2013)) and some tools from graph theory to determine an appropriate set of independent loops. It also requires an initial solution which satisfies continuity or mass balance to start the iterative process which determines the steady-state solution. However, Todini's comments refer to parts of the process that are done before iteration begins and, while they may be cumbersome, are only done once.

In a very nice paper Nielsen (1989) showed, among other things, that the Simultaneous Loop Flows Corrections method, itself a development of the sequential Loop Flows Corrections method of Hardy Cross (1936), is in fact what is called a nullspace method (Benzi, Golub & Liesen 2005). Before that, Smith (1982) applied a tree and co-tree method to what is now referred to as the Linear Theory Method (Nielsen 1989). In Smith's method (i) network loops need to be found, (ii) a super-sink or ground node needs to be added if there is more than one fixed-head node and (iii) pseudo links connecting the fixed head nodes to the ground node need to be added.

A few years after Nielsen, Rahal (1995) published the Co-Trees Method (CTM). In the CTM, the network graph must be transformed into its associated circulating graph, where the network has a unique source and each flow is circulating in a pipe from one node to another. This circulating graph is formed by adding pseudo-links from each demand node to the main source. A pseudo-link which joins each secondary source to the main source is then added. These pseudo-links are required to have certain capacities determined by network parameters. Then a spanning tree must be determined and the so-called circuit matrix is determined. From it is found a global matrix associated with certain cut-sets. The basic equations for the method are then solved using Newton's method. A set of arbitrarily chosen co-tree flows is required to start the method. During the CTM solution process, it is necessary to (i) find the associated chain of branches closing a loop for each co-tree chord, and (ii) compute pseudo-link head losses.

It is shown in the present paper, that the CTM of Rahal, which has startup requirements similar to those of Smith's method, is one and the same as the Simultaneous Loop Flows Corrections method. The CTM requires the solution at each iterative step of a system of linear equations of dimension $n_c = n_p - n_j$, the number of co-tree flows. The number n_c is frequently much smaller than n_j. As for the case of the GGA, the matrix to be inverted in the CTM is symmetric but perhaps because it has been thought to be dense, or because of the two criticisms made by Todini, the CTM has not found favour and has not been used in practice.

Nielsen (1989) also suggested permuting the rows of the unknown–head node–arc incidence matrix to make its top n_j-square block invertible. Twenty years later Schilders (2009), while considering some candidates as preconditioners to be used in conjugate gradient solvers for systems similar to WDSs, suggested using row and column permutations of the unknown–head node–arc incidence matrix to transform it to trapezoidal form, a form in which the top $n_j \times n_j$ block is lower triangular. Now, the top n_j-square block of such a transformed matrix defines the unknown–head node–arc incidence matrix for a spanning tree of the graph of the network and the bottom $n_c \times n_j$ block of the trapezoidal form defines the unknown–head node–arc incidence matrix for the corresponding co-tree of the graph of the network (Diestel 2010). In the present paper, a new straightforward matrix reduction technique is introduced which, when applied to the unknown–head node–arc incidence matrix of the co-tree, leads to a reformulation of the Co-Trees Method. Efficiently implemented, The Reformulated Co-Trees Method (RCTM) leads to an algorithm that, in many cases, is faster in execution time and requires less computer memory than the GGA in settings where many networks with the same topology are to be solved.

The RCTM has the following attractive features:

(a) it requires neither the use of tools from graph theory to identify independent loops nor does it require the addition of virtual loops,

(b) like the method of Rahal (1995), it does not require an initial solution which satisfies continuity,

(c) it exhibits greater robustness in the face of zero flows than the GGA which fails catastrophically because of the singularity of its key matrix (this failure may be mitigated by the application of regularization (Elhay & Simpson 2011)).

For the eight case study networks studied here, the method (as it would be applied in a genetic or Evolutionary Algorithm (EA) optimization)

(a) has computation time that is between 87% and 55% that of the GGA, and

(b) has memory requirements that are much smaller than those of the GGA for some networks.

Item (a) is established by the application of the RCTM to a set of eight case study networks, the largest of which has nearly 20,000 pipes.

Item (b) is established by showing that the storage requirements for the RCTM, although larger than that of the GGA for some of the case study networks, is as little as 7% of the GGA requirement on other of the case study networks. Thus, in some cases, much larger problems can be solved by RCTM than the GGA for the same amount of computer memory.

The RCTM and the CTM both require the solution of symmetric matrix systems with dimension n_c. This observation raises the interesting question, not addressed in the present paper, of what differences the densities and distributions of the non-zeros in the key matrices of those methods would have on the solution times when compared with those of the RCTM. The main interest here, though, is a comparison of the GGA and RCTM methods.

Todini (2006) considered the convergence properties of variations of the GGA and the Simultaneous Loop Flow Corrections method numerically and theoretically. Most of the methods considered in that paper are derived as linear transformations of the GGA. It is shown empirically there that all the flow-based algorithms require the same number of Newton iteration steps to reach exactly the same result when applied to certain example problems. In another more recent development, Todini & Rossman (2013) have drawn together a unified framework for various algorithms that solve the equations for water distribution systems and re-examined their convergence properties.

In the present paper, the mathematical reason for the fact that the simultaneous loops method and the GGA*always*, not only take the same number of iterations to converge from the same starting value, but produce exactly the same iterates, is explained by deriving the two methods directly from the same basic Newton iteration for the steady-state heads and flows that solve the energy and continuity equations. It is shown that the Loop Flows Corrections and the Nullspace methods (previously shown by Nielsen to be equivalent) are actually mathematically equivalent to the RCTM. Thus, in this paper, the three equivalent methods will be referred to as RCTM except where they need to be distinguished.

The results presented in this paper raise the question of which of the RCTM and GGA methods should be chosen in any particular case. A discussion of this question follows the comparison of the two methods later in the paper.

The rest of this paper is structured as follows. Some definitions and notation are introduced in the next section. The section following gives the derivation of the method, with some illustrative examples interspersed. An algorithmic description of the RCTM is then given, followed by a discussion of the relation of the RCTM to other methods. The numerical experiments which support the claims about the speed and storage requirements of the method are then presented and they are followed by a discussion on choosing which of the methods is most appropriate in a particular case. The conclusions section is followed by some appendices which contain material that is necessary for the full understanding of the paper but have been moved so as to not disrupt the flow of the exposition.

DEFINITIONS AND NOTATION

Consider a water distribution network of n_p pipes, $n_j (< n_p)$ junctions or nodes and n_f fixed–head nodes. Suppose Q_j is the unknown flow for the j-th pipe, p_j, which has area of cross section A_j, length L_j, diameter D_j, and resistance factor r_j. All the pipes in the system are assumed to have the same head loss exponent, n, which is either $n = 2$ for Darcy–Weisbach head loss model or $n = 1.852$ for the Hazen–Williams head loss model. Let H_i denote the unknown head at the i-th node, v_i.

Let $\boldsymbol{q} = (Q_1, Q_2, \dots, Q_{n_p})^T$ denote the vector of unknown flows, $\boldsymbol{h} = (H_1, H_2, \dots, H_{n_j})^T$ denote the vector of unknown heads, $\boldsymbol{r} = (r_1, r_2, \dots, r_{n_p})^T$ denote the vector of resistance factors for the pipes, $\boldsymbol{d} = (d_1, d_2, \dots, d_{n_j})^T$ denote the vector of nodal demands, and $\boldsymbol{u}$ denote the vector of dimension n_f of fixed-head elevations.

The relation between the heads at two ends, node v_i and node v_k, of a pipe p_j and the flow in the pipe is defined by $H_i - H_k = r_j Q_j |Q_j|^{n-1}$. Define (i) the square, diagonal matrix $\boldsymbol{G}$ (Todini & Pilati 1988) which has elements $[\boldsymbol{G}]_{jj} = r_j |Q_j|^{n-1}, \ j = 1, 2, \dots, n_p$, (ii) $\boldsymbol{F}$ a diagonal $n_p \times n_p$ matrix in which each diagonal element is the derivative with respect to Q of the element in the corresponding row of the vector $\boldsymbol{G}\boldsymbol{q}$, (iii) the full column–rank, unknown–head node–arc incidence matrix $\boldsymbol{A}_1$ of dimension $n_p \times n_j$, and (iv) the fixed–head, node–arc incidence matrix, $\boldsymbol{A}_2$, of dimension $n_p \times n_f$.

The steady-state flows and heads in the system are the solutions of the energy and continuity equations:

$$\boldsymbol{f}(\boldsymbol{q},\boldsymbol{h}) = \begin{pmatrix} \boldsymbol{G}(\boldsymbol{q}) & -\boldsymbol{A}_1 \\ -\boldsymbol{A}_1^T & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{h} \end{pmatrix} - \begin{pmatrix} \boldsymbol{A}_2\boldsymbol{u} \\ \boldsymbol{d} \end{pmatrix} = \boldsymbol{o}. \tag{1}$$

Denote by $\boldsymbol{J}$ the Jacobian of $\boldsymbol{f}$

$$\boldsymbol{J}(\boldsymbol{q},\boldsymbol{h}) = \begin{pmatrix} \boldsymbol{F}(\boldsymbol{q}) & -\boldsymbol{A}_1 \\ -\boldsymbol{A}_1^T & \boldsymbol{O} \end{pmatrix}.$$

The Newton iteration for (1) proceeds by taking given starting values $\boldsymbol{q}^{(0)}$, $\boldsymbol{h}^{(0)}$ and repeatedly computing, for $m = 0, 1, 2, \ldots$, the iterates $\boldsymbol{q}^{(m+1)}$ and $\boldsymbol{h}^{(m+1)}$ from

$$\begin{pmatrix} \boldsymbol{F}(\boldsymbol{q}^{(m)}) & -\boldsymbol{A}_1 \\ -\boldsymbol{A}_1^T & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{q}^{(m+1)} \\ \boldsymbol{h}^{(m+1)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}(\boldsymbol{q}^{(m)}) - \boldsymbol{G}(\boldsymbol{q}^{(m)}) & \boldsymbol{o} \\ \boldsymbol{o}^T & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{q}^{(m)} \\ \boldsymbol{h}^{(m)} \end{pmatrix} + \begin{pmatrix} \boldsymbol{A}_2\boldsymbol{u} \\ \boldsymbol{d} \end{pmatrix} \tag{2}$$

until, if the iteration converges, the difference between successive iterates is sufficiently small. The block equations of (2) are, omitting for simplicity the dependency of both $\boldsymbol{F}$ and $\boldsymbol{G}$ on m and $\boldsymbol{q}$ since there is no ambiguity

$$\boldsymbol{F}\boldsymbol{q}^{(m+1)} - \boldsymbol{A}_1\boldsymbol{h}^{(m+1)} = (\boldsymbol{F} - \boldsymbol{G})\boldsymbol{q}^{(m)} + \boldsymbol{A}_2\boldsymbol{u}, \tag{3}$$

$$-\boldsymbol{A}_1^T\boldsymbol{q}^{(m+1)} = \boldsymbol{d}. \tag{4}$$

DERIVATION OF THE REFORMULATED CO-TREES METHOD

The Schilders (2009) permutations are now applied to the $\boldsymbol{A}_1$ matrix, a step which is essential to the derivation of the RCTM. To begin, the spanning tree and co-tree are defined mathematically. Suppose Y is a graph. A spanning tree, S, of Y is a subset of the edges of Y that spans every node in Y and which is also a tree (Diestel 2010). The co-tree of Y is made up of all the edges in Y which are not in S.

A method is now derived that begins by manipulating the incidence matrix $\boldsymbol{A}_1$ to find matrices, $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$, which are the unknown–head node-arc incidence matrices of, respectively, a spanning tree of the network's graph and the corresponding co-tree of the network's graph. From these two matrices a reduction of the $\boldsymbol{A}_1$ matrix is derived which leads to a solution of (2) by solving a co-tree reformulation of the problem. The method leads to an algorithm during each iterate of which (i) the co-tree flows are found and (ii) from them the spanning tree flows are found.

Recall that $n_c = n_p - n_j$ denotes the dimension of the co–tree in the graph of the network. The integer n_c is also approximately the number of loops in the system. For any unknown–head node-arc incidence matrix $\boldsymbol{A}_1$ there exist (Schilders 2009) two (orthogonal) permutation matrices $\boldsymbol{P} \in \mathbb{R}^{n_p \times n_p}$ and $\boldsymbol{R} \in \mathbb{R}^{n_j \times n_j}$ and corresponding $\boldsymbol{T}_1 \in \mathbb{R}^{n_j \times n_j}$, lower triangular and $\boldsymbol{T}_2 \in \mathbb{R}^{n_c \times n_j}$ which are such that

$$\boldsymbol{P}\boldsymbol{A}_1\boldsymbol{R} = \begin{pmatrix} \boldsymbol{T}_1 \\ \\ \boldsymbol{T}_2 \end{pmatrix} \overset{\text{def}}{=} \boldsymbol{T}. \tag{5}$$

A simple proof that the matrix $\boldsymbol{A}_1$ has full rank and an algorithm for the determination of the permutations $\boldsymbol{P}$ and $\boldsymbol{R}$ can be found in the Appendix. It is important to note in passing that $\boldsymbol{T}_1$ is invertible because it is a lower triangular matrix with non–zero diagonal elements.

Example 1 Consider the network shown in Figure 1. It has $n_p = 6$ pipes, $n_j = 4$ nodes at which the head is unknown, and $n_f = 1$ reservoir. The co–tree is comprised of $n_c = n_p - n_j = 2$ pipes. Note that if the pipe and node characteristics for this network are symmetric, pipe p_3 will have zero flow at steady-state. As will be seen, this does not cause a failure of the method, unlike the GGA on the same network if the head loss is modeled by the Hazen-Williams formula (Elhay & Simpson 2011).

The unknown–head node-arc incidence matrix $\boldsymbol{A}_1$ for the network in Figure 1 and the matrices $\boldsymbol{T}_1, \boldsymbol{T}_2$ on the right–hand–side of (5) which result from taking the rows in the order $s = (6, 2, 3, 4, 5, 1)$

and its columns in the order $t = (1, 3, 2, 4)$ are

$$\boldsymbol{A}_1 = \begin{pmatrix} 1 & \text{-1} & 0 & 0 \\ 1 & 0 & \text{-1} & 0 \\ 0 & 1 & \text{-1} & 0 \\ 0 & 1 & 0 & \text{-1} \\ 0 & 0 & 1 & \text{-1} \\ \text{-1} & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{T}_1 = \begin{pmatrix} \text{-1} & 0 & 0 & 0 \\ 1 & \text{-1} & 0 & 0 \\ 0 & \text{-1} & 1 & 0 \\ 0 & 0 & 1 & \text{-1} \end{pmatrix} \text{ and } \boldsymbol{T}_2 = \begin{pmatrix} 0 & 1 & 0 & \text{-1} \\ 1 & 0 & \text{-1} & 0 \end{pmatrix},$$

and the lower triangular shape of $\boldsymbol{T}_1$ is now evident. The spanning tree for this particular choice of permutations (shown in Figure 1 as dark lines) is thus made up of pipes p_6, p_2, p_3, p_4 and the co-tree is made up of pipes p_5,and p_1. The permutation matrix $\boldsymbol{P}$, for this example, is an $n_p \times n_p = 6 \times 6$ identity with its rows taken in the order s and the permutation matrix $\boldsymbol{R}$ is a $n_j \times n_j = 4 \times 4$ identity with its columns taken in the order t. □

By successively subtracting appropriate multiples of rows $n_j, n_j - 1, \ldots, 2, 1$ of $\boldsymbol{T}_1$ from rows $1, 2, \ldots, n_c$ of $\boldsymbol{T}_2$ it is possible to zero the whole of $\boldsymbol{T}_2$. This process is similar to Gaussian elimination and it produces a lower triangular matrix $\boldsymbol{L} \in \mathbb{R}^{n_p \times n_p}$ which is such that

$$\boldsymbol{L}\boldsymbol{P}\boldsymbol{A}_1\boldsymbol{R} = \boldsymbol{L} \begin{pmatrix} \boldsymbol{T}_1 \\ \boldsymbol{T}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{T}_1 \\ \boldsymbol{O} \end{pmatrix}. \tag{6}$$

Therefore

$$\boldsymbol{A}_1 = \boldsymbol{P}^T \boldsymbol{L}^{-1} \begin{pmatrix} \boldsymbol{T}_1 \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{R}^T. \tag{7}$$

Let $\boldsymbol{I}_{n_j}$ and $\boldsymbol{I}_{n_c}$, respectively, denote identity matrices of dimension n_j and n_c. From its construction it follows that $\boldsymbol{L}$ can be blocked

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{I}_{n_j} & \boldsymbol{O} \\ \boldsymbol{L}_{21} & \boldsymbol{I}_{n_c} \end{pmatrix} \text{ and so } \boldsymbol{L}^{-1} = \begin{pmatrix} \boldsymbol{I}_{n_j} & \boldsymbol{O} \\ -\boldsymbol{L}_{21} & \boldsymbol{I}_{n_c} \end{pmatrix}, \tag{8}$$

as is easily verified by forming the product $\boldsymbol{L}\boldsymbol{L}^{-1}$. An algorithm which produces $\boldsymbol{L}$ is given in the Appendix. In fact, the matrix $\boldsymbol{L}_{21}$ forms a part of a basis for the nullspace of the permuted node-arc incidence matrix $\boldsymbol{P}\boldsymbol{A}_1\boldsymbol{R}$. It represents one particular set of fundamental loops with the distinct property that every loop has at least one link that is not included in any other loop i.e. such that each co-tree link is only in one loop.

Example 2 Consider the matrices found in Example 1. Multiplying the matrix $\boldsymbol{T}$ of (5) on the left by $\boldsymbol{L}^{(1)}$ has the effect of subtracting the last row of $\boldsymbol{T}_1$ from $\boldsymbol{T}_2$, thereby zeroing both rows of the last column of $\boldsymbol{T}_2$ (shown in bold).

$$\boldsymbol{L}^{(1)}\boldsymbol{T} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \text{-1} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \text{-1} & 0 & 0 & 0 \\ 1 & \text{-1} & 0 & 0 \\ 0 & \text{-1} & 1 & 0 \\ 0 & 0 & 1 & \text{-1} \\ 0 & 1 & 0 & \text{-1} \\ 1 & 0 & \text{-1} & 0 \end{pmatrix} = \begin{pmatrix} \text{-1} & 0 & 0 & 0 \\ 1 & \text{-1} & 0 & 0 \\ 0 & \text{-1} & 1 & 0 \\ 0 & 0 & 1 & \text{-1} \\ 0 & 1 & \text{-1} & \mathbf{0} \\ 1 & 0 & \text{-1} & \mathbf{0} \end{pmatrix}.$$

Similarly, multiplying $\boldsymbol{T}$ on the left by $\boldsymbol{L}^{(2)}$ has the effect of zeroing both rows in the last two columns of $\boldsymbol{T}_2$ (shown in bold):

$$\boldsymbol{L}^{(2)}\boldsymbol{T} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \text{-1} & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \text{-1} & 0 & 0 & 0 \\ 1 & \text{-1} & 0 & 0 \\ 0 & \text{-1} & 1 & 0 \\ 0 & 0 & 1 & \text{-1} \\ 0 & 1 & 0 & \text{-1} \\ 1 & 0 & \text{-1} & 0 \end{pmatrix} = \begin{pmatrix} \text{-1} & 0 & 0 & 0 \\ 1 & \text{-1} & 0 & 0 \\ 0 & \text{-1} & 1 & 0 \\ 0 & 0 & 1 & \text{-1} \\ 0 & 0 & \mathbf{0} & \mathbf{0} \\ 1 & \text{-1} & \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

Finally, multiplying the matrix $\boldsymbol{T}$ on the left by $\boldsymbol{L}^{(3)}$ has the effect of zeroing both rows in all four columns of $\boldsymbol{T}_2$ (shown in bold):

$$\boldsymbol{L}^{(3)}\boldsymbol{T} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{-1} & 1 & 0 \\ \mathbf{0} & \mathbf{-1} & \mathbf{1} & \mathbf{0} & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{T}_1 \\ \boldsymbol{O} \end{pmatrix}.$$

Thus, $\boldsymbol{L}^{(3)}$ is the matrix $\boldsymbol{L}$ of (8) and $\boldsymbol{L}_{21}$ is the $n_p - n_j \times n_j = 2 \times 4$ bottom-left block of $\boldsymbol{L}$ (shown in bold):

$$\boldsymbol{L}_{21} = \begin{pmatrix} 0 & 0 & 1 & -1 \\ 0 & -1 & 1 & 0 \end{pmatrix}.$$

The block structure of $\boldsymbol{L}$ indicated in (8) is now evident. □

It is now possible, using the representation of $\boldsymbol{A}_1$ given in (7) to derive a solution of the Newton equations (3) and (4) which advances by finding, at each iteration, the flows in the co-tree and then the flows in the spanning tree.

Substituting (7) into the first block equation of the Newton method, (3), gives

$$\boldsymbol{F}\boldsymbol{q}^{(m+1)} - \boldsymbol{P}^T\boldsymbol{L}^{-1}\begin{pmatrix} \boldsymbol{T}_1 \\ \boldsymbol{O} \end{pmatrix}\boldsymbol{R}^T\boldsymbol{h}^{(m+1)} = (\boldsymbol{F} - \boldsymbol{G})\boldsymbol{q}^{(m)} + \boldsymbol{A}_2\boldsymbol{u}$$

and left–multiplying this relation by $\boldsymbol{L}\boldsymbol{P}$ and noting that the product $\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{I}$ gives

$$\boldsymbol{L}\boldsymbol{P}\boldsymbol{F}\boldsymbol{P}^T\boldsymbol{P}\boldsymbol{q}^{(m+1)} - \begin{pmatrix} \boldsymbol{T}_1 \\ \boldsymbol{O} \end{pmatrix}\boldsymbol{R}^T\boldsymbol{h}^{(m+1)} = \boldsymbol{L}\boldsymbol{P}(\boldsymbol{F} - \boldsymbol{G})\boldsymbol{P}^T\boldsymbol{P}\boldsymbol{q}^{(m)} + \begin{pmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \end{pmatrix}, \tag{9}$$

where the last term on the right has been denoted by $\boldsymbol{L}\boldsymbol{P}\boldsymbol{A}_2\boldsymbol{u} = (\begin{matrix} \boldsymbol{a}_1^T & \boldsymbol{a}_2^T \end{matrix})^T$. Denoting

$$\widehat{\boldsymbol{q}}^{(m+1)} = \boldsymbol{P}\boldsymbol{q}^{(m+1)}, \widehat{\boldsymbol{q}}^{(m)} = \boldsymbol{P}\boldsymbol{q}^{(m)}, \widehat{\boldsymbol{h}}^{(m+1)} = \boldsymbol{R}^T\boldsymbol{h}^{(m+1)}, \widehat{\boldsymbol{F}} = \boldsymbol{P}\boldsymbol{F}\boldsymbol{P}^T, \text{ and } \widehat{\boldsymbol{G}} = \boldsymbol{P}\boldsymbol{G}\boldsymbol{P}^T \tag{10}$$

allows (9) to be rewritten as

$$\boldsymbol{L}\widehat{\boldsymbol{F}}\widehat{\boldsymbol{q}}^{(m+1)} - \begin{pmatrix} \boldsymbol{T}_1 \\ \boldsymbol{O} \end{pmatrix}\widehat{\boldsymbol{h}}^{(m+1)} = \boldsymbol{L}(\widehat{\boldsymbol{F}} - \widehat{\boldsymbol{G}})\widehat{\boldsymbol{q}}^{(m)} + \begin{pmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \end{pmatrix}. \tag{11}$$

Now, let $\widehat{\boldsymbol{F}}$, $\widehat{\boldsymbol{G}}$ and $\widehat{\boldsymbol{q}}^{(m)}$ be blocked conformally, recalling that $\widehat{\boldsymbol{F}}$ and $\widehat{\boldsymbol{G}}$ are both diagonal, as

$$\widehat{\boldsymbol{F}} = \begin{array}{c c} & \begin{array}{cc} n_j & n_c \end{array} \\ \begin{array}{c} n_j \\ \\ n_c \end{array} & \begin{pmatrix} \widehat{\boldsymbol{F}}_1 & \\ & \widehat{\boldsymbol{F}}_2 \end{pmatrix} \end{array}, \quad \widehat{\boldsymbol{G}} = \begin{array}{c c} & \begin{array}{cc} n_j & n_c \end{array} \\ \begin{array}{c} n_j \\ \\ n_c \end{array} & \begin{pmatrix} \widehat{\boldsymbol{G}}_1 & \\ & \widehat{\boldsymbol{G}}_2 \end{pmatrix} \end{array}, \quad \widehat{\boldsymbol{q}}^{(m)} = \begin{array}{c c} \begin{pmatrix} \widehat{\boldsymbol{q}}_1^{(m)} \\ \widehat{\boldsymbol{q}}_2^{(m)} \end{pmatrix} & \begin{array}{c} n_j \\ \\ n_c \end{array} \end{array}.$$

Then, $\widehat{\boldsymbol{q}}_1^{(m)}$ is a vector of the flows in the spanning tree of the network's graph at the m-th iteration and $\widehat{\boldsymbol{q}}_2^{(m)}$ is a vector of the flows in the co-tree of the network's graph at the same iteration. With this notation (11) can be rewritten as

$$\begin{pmatrix} \boldsymbol{I}_{n_j} & \boldsymbol{O} \\ \boldsymbol{L}_{21} & \boldsymbol{I}_{n_c} \end{pmatrix} \begin{pmatrix} \widehat{\boldsymbol{F}}_1 & \\ & \widehat{\boldsymbol{F}}_2 \end{pmatrix} \begin{pmatrix} \widehat{\boldsymbol{q}}_1^{(m+1)} \\ \widehat{\boldsymbol{q}}_2^{(m+1)} \end{pmatrix} - \begin{pmatrix} \boldsymbol{T}_1 \widehat{\boldsymbol{h}}^{(m+1)} \\ \boldsymbol{O} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{n_j} & \boldsymbol{O} \\ \boldsymbol{L}_{21} & \boldsymbol{I}_{n_c} \end{pmatrix} \begin{pmatrix} \widehat{\boldsymbol{F}}_1 - \widehat{\boldsymbol{G}}_1 & \\ & \widehat{\boldsymbol{F}}_2 - \widehat{\boldsymbol{G}}_2 \end{pmatrix} \begin{pmatrix} \widehat{\boldsymbol{q}}_1^{(m)} \\ \widehat{\boldsymbol{q}}_2^{(m)} \end{pmatrix} + \begin{pmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \end{pmatrix}$$

which expands to

$$\begin{pmatrix} \widehat{\boldsymbol{F}}_1 \widehat{\boldsymbol{q}}_1^{(m+1)} \\ \boldsymbol{L}_{21} \widehat{\boldsymbol{F}}_1 \widehat{\boldsymbol{q}}_1^{(m+1)} + \widehat{\boldsymbol{F}}_2 \widehat{\boldsymbol{q}}_2^{(m+1)} \end{pmatrix} - \begin{pmatrix} \boldsymbol{T}_1 \widehat{\boldsymbol{h}}^{(m+1)} \\ \boldsymbol{O} \end{pmatrix} = \begin{pmatrix} \left(\widehat{\boldsymbol{F}}_1 - \widehat{\boldsymbol{G}}_1 \right) \widehat{\boldsymbol{q}}_1^{(m)} \\ \boldsymbol{L}_{21} \left(\widehat{\boldsymbol{F}}_1 - \widehat{\boldsymbol{G}}_1 \right) \widehat{\boldsymbol{q}}_1^{(m)} + \left(\widehat{\boldsymbol{F}}_2 - \widehat{\boldsymbol{G}}_2 \right) \widehat{\boldsymbol{q}}_2^{(m)} \end{pmatrix} + \begin{pmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \end{pmatrix}.$$

The first block equation of the Newton method, (3), is itself now in two blocks: the first is

$$\widehat{\boldsymbol{F}}_1 \widehat{\boldsymbol{q}}_1^{(m+1)} - \boldsymbol{T}_1 \widehat{\boldsymbol{h}}^{(m+1)} = \left(\widehat{\boldsymbol{F}}_1 - \widehat{\boldsymbol{G}}_1 \right) \widehat{\boldsymbol{q}}_1^{(m)} + \boldsymbol{a}_1 \tag{12}$$

and the second is

$$\boldsymbol{L}_{21} \widehat{\boldsymbol{F}}_1 \widehat{\boldsymbol{q}}_1^{(m+1)} + \widehat{\boldsymbol{F}}_2 \widehat{\boldsymbol{q}}_2^{(m+1)} = \boldsymbol{L}_{21} \left(\widehat{\boldsymbol{F}}_1 - \widehat{\boldsymbol{G}}_1 \right) \widehat{\boldsymbol{q}}_1^{(m)} + \left(\widehat{\boldsymbol{F}}_2 - \widehat{\boldsymbol{G}}_2 \right) \widehat{\boldsymbol{q}}_2^{(m)} + \boldsymbol{a}_2. \tag{13}$$

Similarly, the second block equation of the Newton method, (4), (which is also just the continuity equation) can be written, in view of (7), as

$$-\boldsymbol{A}_1^T \boldsymbol{q}^{(m+1)} = \left(\boldsymbol{P}^T \boldsymbol{L}^{-1} \begin{pmatrix} \boldsymbol{T}_1 \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{R}^T \right)^T \boldsymbol{q}^{(m+1)} = \boldsymbol{R} \begin{pmatrix} \boldsymbol{T}_1^T & \boldsymbol{O} \end{pmatrix} \boldsymbol{L}^{-T} \boldsymbol{P} \boldsymbol{q}^{(m+1)} = -\boldsymbol{d}.$$

Multiplying this relation on the left by $\boldsymbol{R}^T$ (which is the inverse of $\boldsymbol{R}$ by virtue of orthogonality) gives, denoting $\widehat{\boldsymbol{d}} = \boldsymbol{R}^T \boldsymbol{d}$ and substituting for $\boldsymbol{L}$ using (8),

$$\begin{pmatrix} \boldsymbol{T}_1^T & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{n_j} & -\boldsymbol{L}_{21}^T \\ \boldsymbol{O} & \boldsymbol{I}_{n_c} \end{pmatrix} \begin{pmatrix} \widehat{\boldsymbol{q}}_1^{(m+1)} \\ \widehat{\boldsymbol{q}}_2^{(m+1)} \end{pmatrix} = -\widehat{\boldsymbol{d}}.$$

On multiplication, this expands to

$$\boldsymbol{T}_1^T \left(\widehat{\boldsymbol{q}}_1^{(m+1)} - \boldsymbol{L}_{21}^T \widehat{\boldsymbol{q}}_2^{(m+1)} \right) = -\widehat{\boldsymbol{d}}. \tag{14}$$

Multiplying (14) on the left by $\boldsymbol{T}_1^{-T}$ and rearranging gives the important constraint between $\widehat{\boldsymbol{q}}_1^{(m+1)}$ and $\widehat{\boldsymbol{q}}_2^{(m+1)}$,

$$\boxed{\widehat{\boldsymbol{q}}_1^{(m+1)} = \boldsymbol{L}_{21}^T \widehat{\boldsymbol{q}}_2^{(m+1)} - \boldsymbol{T}_1^{-T} \widehat{\boldsymbol{d}}.} \tag{15}$$

Eq. (15) can also be derived from Eq. (13a) of Nielsen (1989). Substituting for $\widehat{\boldsymbol{q}}_1^{(m+1)}$ in (13) with (15) and denoting

$$\boldsymbol{V} = \boldsymbol{L}_{21} \widehat{\boldsymbol{F}}_1 \boldsymbol{L}_{21}^T + \widehat{\boldsymbol{F}}_2, \tag{16}$$

means that (13) can be written, after rearrangement, as

$$\boxed{\boldsymbol{V} \widehat{\boldsymbol{q}}_2^{(m+1)} = \boldsymbol{L}_{21} \left(\widehat{\boldsymbol{F}}_1 - \widehat{\boldsymbol{G}}_1 \right) \widehat{\boldsymbol{q}}_1^{(m)} + \left(\widehat{\boldsymbol{F}}_2 - \widehat{\boldsymbol{G}}_2 \right) \widehat{\boldsymbol{q}}_2^{(m)} + \boldsymbol{a}_2 + \boldsymbol{L}_{21} \widehat{\boldsymbol{F}}_1 \boldsymbol{T}_1^{-T} \widehat{\boldsymbol{d}}.} \tag{17}$$

Strictly speaking $\boldsymbol{V}$ should be denoted with a superscript, $\boldsymbol{V}^{(m)}$, because of its dependence on m, the iteration number: it is different for each m, as are the matrices $\widehat{\boldsymbol{F}}_1$, $\widehat{\boldsymbol{F}}_2$, $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$. However, once again the superscripts will not be shown in the interests of clarity.

Now, rearranging (12) gives

$$\boxed{\boldsymbol{T}_1 \widehat{\boldsymbol{h}}^{(m+1)} = \widehat{\boldsymbol{F}}_1 \widehat{\boldsymbol{q}}_1^{(m+1)} - \left(\widehat{\boldsymbol{F}}_1 - \widehat{\boldsymbol{G}}_1 \right) \widehat{\boldsymbol{q}}_1^{(m)} - \boldsymbol{a}_1.} \tag{18}$$

Together (17), (15) and (18) form the basis of an iterative scheme for solving (2) provided $\boldsymbol{V}$ is invertible.

The iterative scheme consists of repeatedly executing steps (b)(i) and (b)(ii), below, until a suitable stopping test, based on the difference between successive iterates, is satisfied. The scheme only requires an initial set, $\widehat{\boldsymbol{q}}_2^{(0)}$, of co-tree flows (which can be chosen arbitrarily). When the iterates are sufficiently close for the stopping test to be satisfied, the heads are found by solving (18) for $\widehat{\boldsymbol{h}}_1^{(m+1)}$ using a forward substitution. The required solution flows and heads are then found by inverting the permutation $\boldsymbol{P}$ in (10), that took $\boldsymbol{q}^{(m+1)}$ to $\widehat{\boldsymbol{q}}^{(m+1)}$ and applying it: $\boldsymbol{q}^{(m+1)} = \boldsymbol{P}^T \widehat{\boldsymbol{q}}^{(m+1)}$. Similarly, the solution heads can be found as $\boldsymbol{h}^{(m+1)} = \boldsymbol{R}\widehat{\boldsymbol{h}}^{(m+1)}$.

Of course, the permutation matrices, $\boldsymbol{P}$ and $\boldsymbol{R}$, would not be formed explicitly in the practical algorithm and all the permutations would be done by indirect addressing via permutation vectors. They are shown in matrix form only to simplify the exposition. We note also that all the terms which do not depend on the flows or heads, such as the term $\boldsymbol{T}_1^{-T}\widehat{\boldsymbol{d}}$ in (15) or $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, can be pre-computed to improve the efficiency of the algorithm implementation. A robust implementation of the method would also compute the residuals of the system (1) at completion to reject as unsatisfactory any solution for which the residual is large (see Simpson & Elhay (2011) for details).

THE RCTM ALGORITHM

The algorithm can be summarized as:

Input

A set of initial co-tree flows $\widehat{\boldsymbol{q}}_2^{(0)}$, the permutations $\boldsymbol{P}$ and $\boldsymbol{R}$ of (5), and the matrix $\boldsymbol{L}_{21}$ of (8).

Algorithm

```
(a) Compute $\widehat{\boldsymbol{q}}_1^{(0)}$ using (15).

(b) for $m = 1, 2, \ldots$, until the stopping test is satisfied do

        (i) Solve (17) for the co-tree flows, $\widehat{\boldsymbol{q}}_2{}^{(m+1)}$.

        (ii) Use (15) to get the corresponding spanning tree flows, $\widehat{\boldsymbol{q}}_1^{(m+1)}$ which satisfy continuity.

    end $m$-loop
```

(c) Solve (18) for $\widehat{\boldsymbol{h}}_1^{(m+1)}$.

(d) Get the solution flows from $\boldsymbol{q}^{(m+1)} = \boldsymbol{P}^T\widehat{\boldsymbol{q}}^{(m+1)}$ and the solution heads from $\boldsymbol{h}^{(m+1)} = \boldsymbol{R}\widehat{\boldsymbol{h}}^{(m+1)}$.

End

Note that relation (15) can be viewed as a constraint which exists between the flows in the pipes which make up the spanning tree and the flows in the co–tree. For any given set of co-tree flows (15) specifies the unique set of spanning tree flows which ensure that all the flows in the network satisfy continuity.

THE RELATION OF RCTM TO OTHER METHODS

In the Appendix it is shown that the RCTM encapsulated by (15), (17) and (18) is, in fact, a nullspace method (Benzi et al. 2005, 32) in the following sense: it finds one of the infinite number of sets of flows which satisfy the continuity equation (4) and then uses the Newton method to find, from within the left nullspace of the $\boldsymbol{A}_1$ matrix, the correction to those flows which ensures that they also satisfy the energy equation (3). Thus, it is one and the same as the Simultaneous Loop Flows Corrections method of Epp & Fowler (1970). Since the RCTM also finds the co-tree flows at each step (in (17)) it is also necessarily equivalent to the Co-tree method of Rahal.

It is also shown in the Appendix that the matrix $\boldsymbol{V}$ of (16) can also be written as $\boldsymbol{V} = \boldsymbol{Z}^T\widehat{\boldsymbol{F}}\boldsymbol{Z}$, $\boldsymbol{Z} \in \mathbb{R}^{n_p \times n_c}$ a full column-rank matrix whose columns span the left nullspace of $\boldsymbol{T}$. Importantly, $\boldsymbol{V}$ may be invertible even though $\widehat{\boldsymbol{F}}$ is not. In fact, $\boldsymbol{Z}^T\widehat{\boldsymbol{F}}\boldsymbol{Z}$ will be invertible as long as $\widehat{\boldsymbol{F}}\boldsymbol{Z}$ has full rank (See the Appendix for details). The invertibility of $\boldsymbol{V}$ even though $\boldsymbol{F}$ may be singular is in stark contrast to the GGA which fails catastrophically if zero flows cause the matrix $\widehat{\boldsymbol{F}}$ to be singular (Elhay & Simpson 2011). Thus, using the method described here may obviate the need to take special measures to handle zero flows. A regularization method such as the one presented in Elhay & Simpson

(2011) may be required if the condition of $\widehat{\boldsymbol{F}}$ is too large or even to handle cases where the Jacobian is singular since even then the solution to the system may still be obtainable (see e.g. Elhay & Simpson (2013) for details).

The RCTM described above uses the Newton method to find the flows and heads in the WDS. Observe that the steady-state heads and flows of (1) satisfy the fixed point equation which forms the basis of the Newton iteration:

$$\begin{pmatrix} \boldsymbol{F} & -\boldsymbol{A}_1 \\ -\boldsymbol{A}_1^T & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{h} \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}-\boldsymbol{G} & \boldsymbol{o} \\ \boldsymbol{o}^T & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{h} \end{pmatrix} + \begin{pmatrix} \boldsymbol{A}_2\boldsymbol{u} \\ \boldsymbol{d} \end{pmatrix}. \tag{19}$$

Both the GGA and the RCTM iterations use this form and so they produce precisely the same flow iterates as each other (see the Appendix for the derivation of the GGA from the same equations). The GGA produces heads and flows at each iteration while the RCTM iterates only on the flows because (15) and (17) do not involve the heads. Even so, the two methods produce exactly the same flow iterates as each other. In the RCTM, the heads need only be found once after the steady-state flows are determined.

In a setting where the heads are required to be used for the stopping test, the system in (18) would need to be solved at every step. The extra work involved will be minor, however, because the forward substitution of a sparse, triangular system such as the one in (18) can be done very rapidly.

Comparison of the RCTM and GGA methods

In this section it is shown that, if the solutions are required to many problems with the same topology (such as in EA optimization), the efficiently implemented RCTM may provide a significant reduction over the GGA in execution time and/or computer memory.

The RCTM and the GGA were each applied to a set of eight case study networks with between 932 and 19,651 pipes and between 848 and 17,977 nodes and neither pumps nor valves. The computation times of the two methods were compared. The GGA code used is an efficient implementation of the method described in Simpson & Elhay (2011) and the RCTM code is an efficient implementation of the method described in the previous sections of the present paper. Both methods used Matlab (Mathworks 2008) sparse arithmetic. The most computationally intensive parts of the two methods are (i) the computation of the friction factors (required for $\boldsymbol{F}$, $\boldsymbol{G}$, $\widehat{\boldsymbol{F}}_{1,2}$ and $\widehat{\boldsymbol{G}}_{1,2}$ where the head loss is modeled by the Darcy-Weisbach formula), (ii) the determination of the permutations $\boldsymbol{P}$ and $\boldsymbol{R}$ and the matrix $\boldsymbol{L}_{21}$, and (iii) the solution of the linear systems with $\boldsymbol{V}$ and $\boldsymbol{W}$. The rows and columns of these key matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ were permuted (just once at the start of each problem) by the Approximate Minimum Degree reordering of Amestoy, Davis & Duff (2004). Special C++ codes were devised for items (i) and (ii) while the built-in sparse matrix solver in Matlab was used for item (iii). The very efficient built-in Matlab solver (\) was used for the two linear systems because, for sparse arguments, there is no Cholesky factoring function in Matlab and there are no built-in functions for forward and back-substitution. Since the same linear solver was applied to both cases in the comparison, this represents a fair test.

The basic details of the networks considered are given in Table 1 and more detail about them may be found in Simpson, Elhay & Alexander (2012). The use of an EA in the design of a network may require the determination of the steady-state solutions for hundreds of thousands, or even millions, of cases in which all the networks have exactly the same topology but each case has a different set of parameters (such as, for example, pipe diameters). It is one purpose of this paper to establish that the RCTM, in applications where the solutions are required to many problems with the same topology, can be significantly faster than the GGA method and/or can require much less computer memory.

The speed advantage of the RCTM stems from the fact that (i) the permutations $\boldsymbol{P}$ and $\boldsymbol{R}$ in (5), and (ii) the matrix $\boldsymbol{L}_{21}$ of (14)–(17) need to be determined only once at the start of the design process because all the generations of the EA use the same $\boldsymbol{P}$, $\boldsymbol{R}$ and $\boldsymbol{L}_{21}$ since the topology remains unchanged. Thus, in all the timings that follow the times taken to determine $\boldsymbol{P}$, $\boldsymbol{R}$ and $\boldsymbol{L}_{21}$ have been excluded from the analysis. It would be necessary, in a one-off calculation using the RCTM, to allow the extra time to compute these (which accounts for between 10% and 55% of the total time of the RCTM on the case studies presented here).

In those cases where the RCTM has a memory advantage, it derives from the fact that the key matrix which has to be solved at each iteration of the method has fewer non–zero elements.

Timings

Columns 2–5 of Table 1 show the number, n_p, of pipes, n_j, of nodes, n_f, of reservoirs, and, n_c, the dimension of the co–tree. The next column shows the ratio, as a percentage, of the number of co-tree pipes divided by the number of nodes in the network (this number, sometimes called the loop ratio, can have an important bearing on the sparsity of matrix factors involved in the solution of the linear system. See Piller (1995) for details). The next two columns show the number of non–zero elements in the matrices $\boldsymbol{V}$ of (16) and $\boldsymbol{W}$ of (25) and the last column in Table 1 shows the ratio of these numbers as a percentage. Of course, the number of non–zeros in $\boldsymbol{V}$ and $\boldsymbol{W}$ for a particular case is a function of the spanning tree chosen for that case and different spanning trees could lead to quite different matrix sparsities.

Each method was applied 15 times to each of the case study networks on a single processor PC (the calculations were repeated because small variations in the way the operating system runs background processes lead to variations in the time taken for the same program to run on the same data). The means of the times for solution, τ_{RCTM}, τ_{GGA} and the means of their ratios τ_{RCTM}/τ_{GGA} were computed. The standard errors of the ratios were also computed and from these the 95% confidence intervals for the ratios of the mean times were derived. The mean times for solution, the mean ratios of times for solution, and the 95% confidence intervals, $\boldsymbol{I}_{95}$, are shown in columns 2–5 of Table 2. In Table 3 are listed, for illustration, the actual solution times for the RCTM and GGA methods and their ratios for the 15 computer runs on Network N_1.

From Table 2 it is evident that the GGA takes between 15% and 82% more time to run than the RCTM on the case studies. Clearly, the case in which the RCTM runs in 55% of the time of the GGA, on a computation that takes a week justifies the investment required to develop the RCTM program code.

Although it is not the only factor, one important factor which influences the ratio of the computation times for the two methods is the number of non–zeros in the matrices $\boldsymbol{V}$, and $\boldsymbol{W}$: the more non–zeros there are in a matrix, the more computation time will be required to solve a system with that matrix if all other thing are equal. But the distribution of the non–zeros within the matrix also plays a determining role and this probably explains the deviation from direct proportionality between the number of non-zeros and the timings observed in the result reported here.

It is worth noting that the number of non-zeros in each of the matrices $\boldsymbol{V}$ and $\boldsymbol{W}$, for the networks tested here, is an excellent predictor of the number of non-zeros in its Cholesky factor: the ratio of the number of non-zeros in $\boldsymbol{V}$ to the number of non-zeros in its Cholesky factor has an average, for the networks tested here, of about 1.8 with standard deviation of .090, so the proportionality between them is very close to constant. Similarly, there is an almost 1:1 relationship between the number of non-zeros in the matrix $\boldsymbol{V}$ or $\boldsymbol{W}$ and the time taken to solve the linear system efficiently.

Memory requirements

These numbers shown in columns 7–9 of Table 1 are important because only the non–zero elements of a matrix are stored in sparse matrix implementations of matrix solvers such as those used here. Thus, the RCTM would require only 7% the storage locations of the GGA for the solution of N_5. Of course, in some cases, such as N_4 and N_8 the RCTM has similar or higher storage requirements than the GGA. So, it may be possible to solve some problems using the RCTM on a particular machine while it is impossible to solve those same problems on the same machine using the GGA because of the GGA's memory requirements.

A strategy to decide which of the methods to use in a particular case is outlined next.

Choosing between the RCTM and GGA

The choice between the two methods presented here can be made on any or all of three considerations: (i) the speed of computation, (ii) the storage requirements or (iii) the presence of zero flows.

In a setting such as network design with an EA it may well be worth investing some effort to decide which method is preferable. The matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ can be calculated and the number of non–zero elements quickly determined. Prohibitive memory requirements for one method but not the other might decide the question. If computer memory is not an issue, then the same problem could be solved once by each method and the faster method chosen for the EA solution.

If the computation times and storage requirements are comparable then the decision might be based on the occurrence of zero flows and the possibility that they might occur in some of the variations that arise during the optimization. Any significant difference between the two methods, if such exists for the problem in question, will help decide the choice.

CONCLUSIONS

A reformulation which improves the Co-Trees Method, the RCTM, is introduced. The method operates by permuting the rows and columns of the incidence matrix to transform it into trapezoidal form: a lower triangular block at the top representing a spanning tree and rectangular block below it representing the corresponding co-tree. This reordering leads to significant efficiencies which make the RCTM competitive with the GGA in certain settings.

The improved method is shown, by application to a set of eight case study networks with between 932 and 19,647 pipes and between 848 and 17971 nodes, to take between 55% and 87% of the time taken by the GGA to solve the same problems in a setting (e.g. evolutionary algorithms) where the methods are applied hundreds of thousands, or even millions, of times to networks with the same topology. It is shown that the key matrix for the RCTM can require as little as 7% of the storage requirements of the corresponding matrix for the GGA. This can allow for the solution of larger problems by the RCTM than might be possible for the GGA in the same computing environment.

Unlike some alternatives to the GGA, several features, aside from the faster execution time mentioned above, make the RCTM attractive: (i) it does not require a set of initial flows which satisfy continuity, (ii) there is no need to identify independent loops or the loops incidence matrix, (iii) a spanning tree and co-tree can be found simply from the incidence matrix $\boldsymbol{A}_1$ without the addition of virtual loops, particularly when multiple reservoirs are present, (iv) the RCTM does not require the addition of a ground node and pseudo-loops or the determination of cut-sets, and (v) exhibits greater robustness in the face of zero flows than the GGA which fails catastrophically because of the singularity of its key matrices.

This paper also (i) reports a comparison of the sparsity of the key RCTM and GGA matrices for the case study networks, (ii) shows mathematically why the RCTM and GGA always take the same number of iterations and produce precisely the same iterates, and (iii) establishes that the RCTM, the Loop Flow Corrections Method and the Nullspace Method are one and the same.

It would be interesting to know if the RCTM applied to pressure-driven simulations can deliver similar savings in execution time and/or memory requirements over the GGA applied to those problems.

ACKNOWLEDGMENTS

The authors gratefully thank Prof. Caren Tischendorf for helpful discussions and advice with the manuscript. The authors also thank the anonymous reviewers for their helpful and insightful comments.

References

Amestoy, P. R., Davis, T. A. & Duff, I. (2004), ‘Algorithm 837: AMD, an approximate minimum degree ordering algorithm’, *ACM Trans. Math. Softw.* **30**(3), 381–388.

Benzi, M., Golub, G. & Liesen, J. (2005), ‘Numerical solution of saddle point problems’, *Acta. Num.* pp. 1–137.

Creaco, E. & Franchini, M. (2013), ‘Comparison of newton-raphson global and loop algorithms for water distribution network resolution’, *J. Hydraul. Eng.* . Posted ahead of print. DOI: 10.1061/(ASCE)HY.1943-7900.0000825.

Cross, H. (1936), ‘Analysis of flow in networks of conduits or conductors’, *Bulletin, The University of Illinois Eng. Experiment Station* **286**.

Diestel, R. (2010), *Graph Theory*, Vol. 173 of *Graduate texts in mathematics*, fourth edn, Spriger-Verlag, Heidelberg.

Elhay, S. & Simpson, A. (2011), ‘Dealing with zero flows in solving the non–linear equations for water distribution systems’, *Journal of Hydraulic Engineering* **137**(10), 1216–1224. doi:10.1061/(ASCE)HY.1943-7900.0000411. ISSN: 0733-9429.

Elhay, S. & Simpson, A. (2013), ‘Closure on ”dealing with zero flows in solving the non-linear equations for water distribution systems”’, *Journal of Hydraulic Engineering* **139**(5), 560–562. DOI: 10.1061/(ASCE)HY.1943-7900.0000696.

Epp, R. & Fowler, A. (1970), 'Efficient code for steady-state flows in networks', *Journal of the Hydraulics Division, Proceedings of the ASCE* **96**(HY1), 43–56.

Mathworks, T. (2008), *Using Matlab*, The Mathworks Inc., Natick, MA.

Nielsen, H. (1989), 'Methods for analyzing pipe networks', *J. Hydraul. Eng.* **115**(2), 139–157.

Piller, O. (1995), Modeling the behavior of a network - Hydraulic analysis and a sampling procedure for estimating the parameters., PhD thesis, University of Bordeaux, France. Ph.D. Thesis, in French.

Rahal, H. (1995), 'A co–tree formulation for steady state in water distribution networks', *Advances in Engineering Software* **22**(3), 169–178.

Rossman, L. (2000), *EPANET 2 Users Manual*, Water Supply and Water Resources Division, National Risk Management Research Laboratory, Cincinnati, OH45268.

Schilders, W. (2009), 'Solution of indefinite linear systems using an LQ decomposition for the linear constraints', *Linear Algebra Appl.* **431**, 381–395.

Simpson, A. & Elhay, S. (2011), 'The Jacobian for solving water distribution system equations with the Darcy-Weisbach head loss model', *Journal of Hydraulic Engineering* **137**(6), 696–700. doi:10.1061/(ASCE)HY.1943-7900.0000341. ISSN: 0733-9429.

Simpson, A., Elhay, S. & Alexander, B. (2012), 'A forestcore partitioning algorithm for speeding up the analysis of water distribution systems', *J. Water Resour. Plann. Manage.* . Accepted and posted online Nov. 21, 2012. DOI: 10.1061/(ASCE)WR.1943-5452.0000336.

Smith, A. (1982), 'Compact formulation of pipe network problems', *Can. J. Civ. Eng.* **9**, 611–623.

Todini, E. (2006), On the convergence properties of the different pipe network algorithms, *in* 'Proceedings of the 8th Annual Water Distribution Systems Analysis Symposium', Cincinnati, Ohio, USA, pp. 1–16.

Todini, E. & Pilati, S. (1988), *A gradient algorithm for the analysis of pipe networks.*, John Wiley and Sons, London, pp. 1–20. B. Coulbeck and O. Chun-Hou (eds).

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

APPENDICES

THE UNKNOWN–HEAD NODE–ARC INCIDENCE MATRIX HAS FULL RANK

Consider a fully connected network with at least one reservoir. Let $\boldsymbol{A}_1 \in \mathbb{R}^{n_p \times n_j}$, defined by

$$[\boldsymbol{A}_1]_{ji} = \begin{cases} -1 & \text{if the flow in pipe } j \text{ enters the unknown–head node } i, \\ 0 & \text{if pipe } j \text{ does not connect to the unknown–head node } i, \\ 1 & \text{if the flow in pipe } j \text{ leaves the unknown–head node } i \end{cases} \tag{20}$$

be the unknown–head node–arc incidence matrix for this network. It has one row for each pipe and one column for each node at which the head is unknown.

This matrix always has full (column) rank. To see this, observe that there is always at least one pipe which is connected to only one node at which the head is unknown: it could be a pipe connected to a reservoir or a leaf node at an extremity of the network. This means that the row in $\boldsymbol{A}_1$ corresponding to this pipe has exactly one non–zero element. Permute the rows and columns of $\boldsymbol{A}_1$ in such a way as to place that element (always a ± 1) in the top left-hand (the $(1, 1)$) position. Now consider the submatrix of $\boldsymbol{A}_1$ formed by excluding the first row and the first column. This $(n_p - 1) \times (n_j - 1)$ submatrix also has at least one pipe connected to only one node at which the head is unknown – the pipe which was connected to the node just removed is one such. The row for this pipe has exactly one non–zero element. So it is possible to repeat the process of permuting rows and columns to place this element in the $(1, 1)$ position of the (reduced dimension) submatrix and then consider the $(n_p - 2) \times (n_j - 2)$ submatrix formed by excluding first row and column of this submatrix. In fact, this process can be performed a total of n_j times at which point the top $n_j \times n_j$ square of the row and column permuted $\boldsymbol{A}_1$ matrix is a lower triangle with all diagonal elements ± 1. This establishes that $\boldsymbol{A}_1$ has full column rank.

The top $n_j \times n_j$ square of the permuted $\boldsymbol{A}_1$ represents a spanning tree of the network and the bottom $n_p - n_j$ rows, represent the co–tree, the pipes not in the spanning tree which make up the network loops.

THE ALGORITHM TO COMPUTE THE MATRIX $\boldsymbol{L}$

A Matlab implementation of the algorithm to determine the matrix $\boldsymbol{L}$ of (6), which zeros the matrix $\boldsymbol{T}_2$ by a process similar to Gaussian elimination, is presented in this section. It is assumed that the $\boldsymbol{P}$ and $\boldsymbol{R}$ of (5) have been applied to $\boldsymbol{A}_1$ to produce the matrices $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$.

```
% algorithm to find L from T1, T2
L=eye(np,np);
for j=nj:-1:1
  for k=1:npj
    m=T2(k,j)/T1(j,j); L(k+nj,j)=-m; T2(k,j)=0;
    for i=1:(j-1)
      T2(k,i)=T2(k,i)-m*T1(j,i);
    end %i
  end %k
end %j
```

THE INVERTIBILITY OF $\boldsymbol{Z}^T\boldsymbol{F}\boldsymbol{Z}$

Suppose that $\boldsymbol{Z} \in \mathbb{R}^{n_p \times n_c}$, $n_p > n_j$ has full rank and that $\boldsymbol{F} = \text{diag}\left\{f_1, f_2, \ldots, f_{n_p}\right\}$ is non–negative definite. Under what conditions is $\boldsymbol{Z}^T\boldsymbol{F}\boldsymbol{Z}$ invertible?

Denoting

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{z}_1^T \\ \boldsymbol{z}_2^T \\ \vdots \\ \boldsymbol{z}_{n_p}^T \end{pmatrix} \text{ so } \boldsymbol{Z}^T = \begin{pmatrix} \boldsymbol{z}_1 & \boldsymbol{z}_2 & \ldots & \boldsymbol{z}_{n_p} \end{pmatrix}$$

gives that $\boldsymbol{W} = \boldsymbol{Z}^T\boldsymbol{F}\boldsymbol{Z} = \sum_{k=1}^{n_p} f_k \boldsymbol{z}_k \boldsymbol{z}_k^T$. If more than n_j of the diagonal elements of $\boldsymbol{F}$ vanish then $\boldsymbol{W}$ is certainly singular because it is not then possible to find n_c linearly independent terms $f_k \boldsymbol{z}_k \boldsymbol{z}_k^T$.

Since $\boldsymbol{Z}$ has full rank it follows that, if $\boldsymbol{F}\boldsymbol{Z}$ has full rank then $\boldsymbol{W}$ is invertible. Another way of saying this is: Let $\widehat{\boldsymbol{Z}}$ be the matrix formed by deleting the rows $\boldsymbol{z}_k^T$ for which $f_k = 0$. If $\widehat{\boldsymbol{Z}}$ has full rank then $\boldsymbol{W}$ is invertible.

THE RCTM IS A NULLSPACE METHOD

From (5) it follows that $\boldsymbol{A}_1^T = \boldsymbol{R}\boldsymbol{T}^T\boldsymbol{P}$ and so the continuity equation, (4), can be written $\boldsymbol{R}\boldsymbol{T}^T\boldsymbol{P}\boldsymbol{q}^{(m+1)} = -\boldsymbol{d}$ or, after rearrangement, as $\boldsymbol{T}^T\boldsymbol{P}\boldsymbol{q}^{(m+1)} = -\boldsymbol{P}^T\boldsymbol{d}$ and this is just

$$-\boldsymbol{T}^T\widehat{\boldsymbol{q}}^{(m+1)} = \widehat{\boldsymbol{d}}. \tag{21}$$

Let $\boldsymbol{Z}$ be a matrix whose columns span the left nullspace of $\boldsymbol{T}$: i.e. such that $\boldsymbol{Z}^T\boldsymbol{T} = \boldsymbol{O}$. Suppose the vector $\tilde{\boldsymbol{q}}^{(m+1)}$ is one of the infinite number of solutions of the under-determined linear system in (21). A vector $\boldsymbol{v}^{(m+1)}$ is sought which is such that

$$\widehat{\boldsymbol{q}}^{(m+1)} = \tilde{\boldsymbol{q}}^{(m+1)} + \boldsymbol{Z}\boldsymbol{v}^{(m+1)} \tag{22}$$

also satisfies the energy equation

$$\widehat{\boldsymbol{F}}\widehat{\boldsymbol{q}}^{(m+1)} - \boldsymbol{T}\widehat{\boldsymbol{h}}^{(m+1)} = (\widehat{\boldsymbol{F}} - \widehat{\boldsymbol{G}})\widehat{\boldsymbol{q}}^{(m)} + \boldsymbol{L}^{-1}\begin{pmatrix}\boldsymbol{a}_1\\ \\ \boldsymbol{a}_2\end{pmatrix}. \tag{23}$$

Using a correction term of the form $\boldsymbol{Z}\boldsymbol{v}^{(m+1)}$ ensures that the resulting vector $\widehat{\boldsymbol{q}}^{(m+1)}$ still satisfies (21) because the added term lies in the left nullspace of $\boldsymbol{T}$. Substituting the right–hand–side of (22) into (23), multiplying on the left by $\boldsymbol{Z}^T$ and rearranging gives, noting that $\boldsymbol{Z}^T\boldsymbol{T}\widehat{\boldsymbol{h}}^{(m+1)} = \boldsymbol{o}$,

$$\boldsymbol{Z}^T\widehat{\boldsymbol{F}}\boldsymbol{Z}\boldsymbol{v}^{(m+1)} = \boldsymbol{Z}^T\left[\left(\widehat{\boldsymbol{F}} - \widehat{\boldsymbol{G}}\right)\widehat{\boldsymbol{q}}^{(m)} + \boldsymbol{L}^{-1}\begin{pmatrix}\boldsymbol{a}_1\\ \\ \boldsymbol{a}_2\end{pmatrix} - \widehat{\boldsymbol{F}}\tilde{\boldsymbol{q}}^{(m+1)}\right] \tag{24}$$

Thus, (24) is the key equation in the nullspace method. It can be seen that (17) is equivalent to (24) by choosing $\boldsymbol{Z} = (\,\boldsymbol{L}_{21} \quad \boldsymbol{I}_{nc}\,)^T$ since this matrix does indeed span the left nullspace of $\boldsymbol{P}\boldsymbol{A}_1\boldsymbol{R}$ as is easily seen from the second block equation of (6). Using this choice of $\boldsymbol{Z}$ gives, by direct evaluation, that

$$\boldsymbol{Z}^T\widehat{\boldsymbol{F}}\boldsymbol{Z} = (\,\boldsymbol{L}_{21} \quad \boldsymbol{I}_{n_c}\,)\begin{pmatrix}\widehat{\boldsymbol{F}}_1 & \\ & \widehat{\boldsymbol{F}}_2\end{pmatrix}\begin{pmatrix}\boldsymbol{L}_{21}^T\\ \\ \boldsymbol{I}_{n_c}\end{pmatrix} = \boldsymbol{L}_{21}\widehat{\boldsymbol{F}}_1\boldsymbol{L}_{21}^T + \widehat{\boldsymbol{F}}_2 = \boldsymbol{V}$$

of (16) and expansion of the right–hand–side of (24) shows that it is precisely the right–hand–side of (17). This establishes that the RCTM is indeed the nullspace method.

WHY THE GGA AND RCTM PRODUCE EXACTLY THE SAME ITERATES FOR THE SAME STARTING VALUES

The RCTM was derived to solve for the iterates in the Newton equations (3) and (4). But the RCTM and GGA methods produce exactly the same flow iterates for the same starting values (the heads at each iteration, were they to be computed in the RCTM, would also agree exactly). The proof for this rests on the fact that the GGA method can also be derived from the equations, (3) and (4), which lead to the RCTM.

Multiplying (3) on the left by $-\boldsymbol{A}_1\boldsymbol{F}^{-1}$ and rearranging gives

$$\boldsymbol{A}_1^T\boldsymbol{F}^{-1}\boldsymbol{A}_1\boldsymbol{h}^{(m+1)} = -\boldsymbol{A}_1\boldsymbol{F}^{-1}\Big((\boldsymbol{F}-\boldsymbol{G})\boldsymbol{q}^{(m)} + \boldsymbol{A}_2\boldsymbol{u}\Big) + \boldsymbol{A}_1^T\boldsymbol{q}^{(m+1)}.$$

Replacing the last term on the right–hand–side by $-\boldsymbol{d}$ using (4) gives

$$\boldsymbol{A}_1^T\boldsymbol{F}^{-1}\boldsymbol{A}_1\boldsymbol{h}^{(m+1)} = -\boldsymbol{A}_1\boldsymbol{F}^{-1}\Big((\boldsymbol{F}-\boldsymbol{G})\boldsymbol{q}^{(m)} + \boldsymbol{A}_2\boldsymbol{u}\Big) - \boldsymbol{d}$$

which is precisely the first block equation of the GGA (see Simpson & Elhay (2011) for further details). The GGA equation for the flows is unchanged from (4). The key matrix which must be inverted here is

$$\boldsymbol{W} = \boldsymbol{A}_1^T\boldsymbol{F}^{-1}\boldsymbol{A}_1. \qquad (25)$$

If zero flows cause some elements of $\boldsymbol{F}$ to vanish then the method fails catastrophically because then neither $\boldsymbol{F}^{-1}$ nor $\boldsymbol{W}$ exist.

TABLES

ID	n_p	n_j	n_f	n_c	$\frac{n_c}{n_j}\%$	$nnz(\boldsymbol{V})$	$nnz(\boldsymbol{W})$	$\frac{nnz(\boldsymbol{V})}{nnz(\boldsymbol{W})}\%$
N_1	932	848	8	84	10%	350	2682	13%
N_2	1118	1039	2	79	8%	1141	3265	35%
N_3	1975	1770	4	205	12%	2491	5706	44%
N_4	2465	1890	3	575	30%	6855	6714	102%
N_5	2509	2443	2	66	3%	534	7451	7%
N_6	8585	8392	2	193	2%	2633	25554	10%
N_7	14830	12523	7	2307	18%	31601	41147	77%
N_8	19647	17971	15	1676	9%	70942	57233	124%

Table 1: The case study networks, their characteristics, the number of non–zero elements, nnz, in the key matrices, $\boldsymbol{V}$ for RCTM and $\boldsymbol{W}$ for the GGA method, and their ratios.

ID	$\overline{\tau}_{RCTM}(s)$	$\overline{\tau}_{GGA}(s)$	mean $\frac{\tau_{GGA}}{\tau_{RCTM}}$	$\boldsymbol{I}_{95}$
N_1	0.0168	0.0307	1.82	$[1.78, 1.86]$
N_2	0.0158	0.0280	1.77	$[1.68, 1.86]$
N_3	0.0321	0.0483	1.50	$[1.45, 1.56]$
N_4	0.0404	0.0525	1.30	$[1.25, 1.34]$
N_5	0.0282	0.0489	1.74	$[1.65, 1.82]$
N_6	0.0970	0.1779	1.84	$[1.82, 1.85]$
N_7	0.2346	0.2682	1.15	$[1.11, 1.20]$
N_8	0.4668	0.5414	1.29	$[0.99, 1.58]$

Table 2: The case study networks showing the estimates of the mean times of the two methods, the mean ratio of times and the 95% confidence intervals, $\boldsymbol{I}_{95}$.

$t_{RCTM}(s)$	$t_{GGA}(s)$	$\frac{t_{GGA}}{t_{RCTM}}$
0.0196	0.0349	1.78
0.0166	0.0293	1.76
0.0166	0.0292	1.76
0.0166	0.0293	1.77
0.0166	0.0295	1.78
0.0167	0.0294	1.76
0.0171	0.0295	1.73
0.0166	0.0306	1.84
0.0166	0.0298	1.80
0.0166	0.0297	1.79
0.0166	0.0319	1.92
0.0166	0.0316	1.90
0.0166	0.0311	1.88
0.0166	0.0327	1.97
0.0166	0.0316	1.90

Table 3: The times, $t_{RCTM}(s)$ and $t_{GGA}(s)$, for each of the 15 repetitions of the RCTM and GGA methods applied to network N_1 and the ratios t_{GGA}/t_{RCTM} of the times.

FIGURES

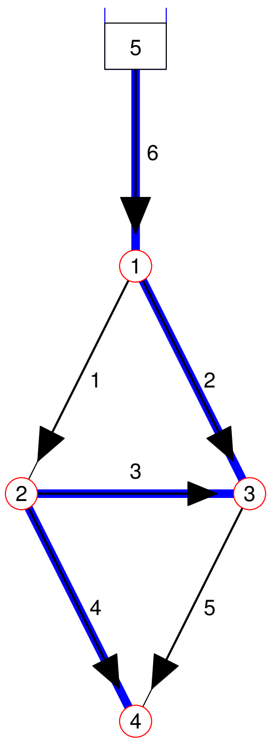

Figure 1: The network discussed in the examples with the spanning tree shown by the darker lines.

List of Figures

List of Tables