1. Introduction

Regarding computer modeling of water distribution systems, very sophisticated methods of hydraulic analysis have been developed to deal with complex issues related to the design, operation and management of distribution systems. Nonetheless, there is still a need to put effort into improving the computational efficiency and stability of the hydraulic simulation. This is due to the opportunities given by the development in computer science and the increasing demand for computing power from more sophisticated methods of hydraulic analysis have been developed and coded over the last 56 years (Ormsbee, 2006). Most notably, the EPANET software (Rossman, 2000), which is the most widely used water network simulation package, has realized extremely speedy and stable hydraulic simulation through the application of Todini's approach and sophisticated code design. Recently, the upcoming EPANET 3.0 is expected to improve the computation efficiency even further by using modified node reordering techniques.

To speedup the simulation, the Newton–Raphson method is applied as an extension of the Hardy Cross method (Todini and Pilati, 1988). By simultaneously considering all loops or nodes in the whole system (Featherstone and Nalluri, 1988; Bhave, 2003; Sitzenfrei et al., 2013; Scholten et al., 2013), the reduction of the computational load allows for comprehensive hydraulic simulation-based analyses.

Thus far, two major groups of approaches are commonly used for steady state hydraulic analysis (Todini and Rossman, 2013). The early local approaches (Cross, 1936), which solve one equation at a time, and the more recent simultaneous equation approaches including: the simultaneous loop method (Epp and Fowler, 1970; the simultaneous node method (Martin and Peters, 1963; Shamir and Howard, 1968); the simultaneous pipe method (Wood and Charles, 1972); the hybrid approach (Hamam and Brameller, 1971; Osiadcz, 1987) and the Global Gradient Algorithm (Todini and Pilati, 1988; Salgado et al., 1988). These methods have been developed and coded over the last 56 years (Ormsbee, 2006). Most notably, the EPANET software (Rossman, 2000), which is the most widely used water network simulation package, has realized extremely speedy and stable hydraulic simulation through the application of Todini’s approach and sophisticated code design. Recently, the upcoming EPANET 3.0 is expected to improve the computation efficiency even further by using modified node reordering techniques.

To speedup the simulation, the Newton–Raphson method is applied as an extension of the Hardy Cross method (Todini and Pilati, 1988). By simultaneously considering all loops or nodes in the whole system (Featherstone and Nalluri, 1988; Bhave, 2003; Sitzenfrei et al., 2013; Scholten et al., 2013).
Crous, 2009; Crous et al., 2012), the method converges faster than any method that corrects either the flow or the head at a single component in the system (Featherstone and Nalluri, 1988; Bhave, 2003; Crous, 2009; Crous et al., 2012). The Newton-based linear theory method simplifies the solution scheme by converting the system of equations into a set of linear equations, i.e. linearization of the non-linear equations governing the conservation of mass (Featherstone and Nalluri, 1988). The Gradient Algorithm (GA) constructs a linear system with its coefficient matrix being symmetrical and positive definite (Stietjtes type). Subsequently, the system can be efficiently solved for both the unknown heads and unknown pipe flows simultaneously using a variety of methods, such as the Conjugate Gradient Method, the Incomplete Cholesky Factorization, or the Modified Conjugate Gradient Method (Todini and Pilati, 1988). The Two Point Linear Method (Van Zyl et al., 2008) reduces computational cost by trading off accuracy and convergence speed. Furthermore, as a flow range is assigned to each pipe, the accuracy of the solution can be checked for each pipe before conducting another full iteration. Zecchin et al. (2012) showed that for each pipe, the accuracy of the solution can be checked for each pipe before conducting another full iteration. Zecchin et al. (2012) proved the efficiency of using the algebraic multigrid (AMG) method, which is a defect-correction approach that iteratively corrects an approximate solution of the original system using solutions from a sequence of constructed lower dimensional systems.

Alonso et al. (2000) implemented parallel computing in hydraulic simulations based on the multifrontal Cholesky method. Crous (2009), Crous et al. (2012) examined the feasibility of using graphical processing units (GPU) with the Conjugate Gradient Method. Thus far, however, these studies do not indicate that application of parallel processing to hydraulic network solvers would formulate different systems of linear equations (e.g. vertices and/or edges) allows the graph to be partitioned into subgraphs with at most a constant fraction of the number of components. At each level of recursion, the components of the graph are numbered in such a way that the separator components are ordered after the components in the partitions (Karypis and Kumar, 1998a,b). Based on the reordering strategy above, all the entries are hydraulic method-specific and therefore large water distribution models (Crous et al. 2012).

This study aims at making a substantial modification on the computation scheme of the hydraulic simulation in order to reduce the computational complexity, and consequently to increase simulation efficiency. Accordingly, the Schur complement domain decomposition method (Toselli and Widlund, 2005) is applied. The algorithm is based on subdivision of the domain into smaller sub-domains and reordering of the nodes within the sub-domains (Alekseandrov and Samuel, 2010). Thus, the solution of a large system of linear equations (the domain) is converted to the solution of a series of smaller systems (the sub-domains). Consequently, a nearly linear computation complexity can be reached. In terms of the water distribution systems, the domain is the whole system and the sub-domains are subsystems specified after decomposition. As it is not uncommon that a distribution system is divided into smaller metered subsystems to improve water audit (Farley, 2001; Thornton et al., 2008), the computation scheme could therefore be formed based on the subsystems’ layout. Hence, in the context of water distribution systems, this method has a close tie to practice instead of being an abstract mathematical concept.

2. The Schur complement domain decomposition method

The hydraulic simulation could be formulated to the iterative solution of a sequence of systems of linear equations (Ax = b). For different solution methods (e.g. the simultaneous loop method; the Global Gradient Algorithm, etc.), different systems (Ax = b) are specified (Boulos et al., 2006; Ormsbee, 2006). In this study, the Global Gradient Algorithm (GGA) is selected to formulate the equations (for details refer to the appendix), since it is currently the most efficient solution method. However, the proposed decomposition method is general applicable to all solution methods of water distribution simulation.

To implement the Schur complement domain decomposition method, the whole distribution system is first divided into a number of subsystems. Correspondingly, the system of linear equations (Ax = b) is decomposed into a set of matrix equations with reduced dimensionality. Thus, the problem is converted from solving a huge dimensional system into solving sequentially a series of low dimensional subsystems (Kran, 1963; Toshechcva and Shishkin, 2008).

2.1. System decomposition: nested dissection partitioning

System decomposition is the prerequisite for using the domain decomposition method. In this regard, graph partitioning algorithms can be used for distribution system decomposition based on mapping the system into an undirected graph G = (V, E) in which the vertices V represent consumers, sources, and tanks - the edges E the connecting pipes, pumps, and valves (Perlman and Ostfeld, 2011). Corresponding to the hydraulic simulation scheme, the vertices represent rows and columns of the system of linear equations, and an edge represents a nonzero entry in the coefficient matrix (A) representing the system.

In this study, the graph partitioning package METIS is invoked to nested dissection partitioning (Karypis and Kumar, 1998a,b). Operated with the reduced-size graph, the partitioning algorithms in METIS are extremely fast compared to traditional partitioning algorithms that compute a partition directly on the original graph. Extensive testing has also shown that the partitions provided by METIS are consistently better than those produced by spectral partitioning algorithms (Karypis and Kumar, 1998a,b; Miettinen et al., 2006). Nested dissection (George, 1973) is an approach that recursively splits a graph into almost equally-sized (balanced numbers of components) subgraphs using separators. The removal of small subsets of components in the graph (e.g. vertices and/or edges) allows the graph to be partitioned into subgraphs with at most a constant fraction of the number of components. At each level of recursion, the components of the graph are numbered in such a way that the separator components are ordered after the components in the partitions (Karypis and Kumar, 1998a,b). Based on the reordering strategy above, the original system of linear equations is transformed into a linear, bordered, block-diagonal (Eq. (1)) (Kocin, 1963). For the application to water distribution systems, a simple example is given in Figs. 1 and 2. As shown in Fig. 1, the network is divided into two subsystems using the specified separators. Selecting both a pipe and a node as separators is a result of using the GGA that solves the pipe flows and nodal heads simultaneously. By such a division, the sub matrix equation will have the same form as the whole matrix equation of the GGA (Appendix). Consequently, the accuracy and stability of the solution is guaranteed. After the decomposition, the two subsystems are nearly equally-sized with 6 components (in subsystem 1) and 5 components (in subsystem 2), respectively. Based on the decomposition, the coefficient matrix (A) is reordered by numbering the components in subsystems first and then the components in separators (Fig. 2).

2.2. The Schur complement domain decomposition algorithm

This section introduces how the Schur complement domain decomposition method is applied to solve the linear, bordered, block-diagonal model (Eq. (1)).

\[ A_{11} x_1 + A_{12} x_2 = b_1, \]
\[ A_{21} x_1 + A_{22} x_2 = b_2. \]

In Eq. (1), \( A_{ij} \) is the coefficient matrix of subsystems and \( N \) is the total number of subsystems. \( A_{ij} \) represents the coefficient of coupling between subsystem \( i \) and the separators, in which the nonzero entries correspond to edges connecting them. \( A_{i0} = A_{0i}^T \) \( A_{0} \) and \( X_0 \) and \( b_0 \) are unknowns for subsystem \( i \) and the separators respectively. \( b_i \) and \( b_0 \) are right hand side terms for subsystem \( i \) and the separators respectively. In each matrix or vector introduced above, all the entries are hydraulic method-specific since different hydraulic methods would formulate different systems of linear equations (\( Ax = b \)) (Boulos et al., 2006; Ormsbee, 2006). For this study, the Global Gradient Algorithm (GGA) is applied to build the equations (See as the Appendix).

From Eq. (1), it yields,
\[ x_i = A_{i0}^T x_0 + A_{i0}^T x_0 - b_0. \]

\[ \sum_{i=1}^{N} A_{i0} x_0 + A_{i0} x_0 - b_0. \]

Fig. 1. A water distribution system graph partitioned using nested dissection.
Fig. 2. The original coefficient matrix of the Global Gradient Algorithm and the coefficient matrix reordered and partitioned with nested dissection. (A) The original matrix (B) The reordered and partitioned matrix (Gray blocks represent coupling between separators and subsystems, and the colored entries in gray blocks represent separator-subsystem edges or vertices in the graph). (C) Matrix visualization corresponding to Fig. 1.

Eliminating \( \mathbf{X}_i \) by combining Eqs. (2) and (3), it produces

\[
\left( \mathbf{A}_{i,R,B} - \sum_{i=1}^{N} \mathbf{A}_{i,R} \mathbf{A}_{i,B} \mathbf{A}_{i,B}^{-1} \mathbf{A}_{i,R} \right) \mathbf{X}_i = \mathbf{b}_i - \sum_{i=1}^{N} \mathbf{A}_{i,R} \mathbf{A}_{i,B} \mathbf{b}_i
\]

(4)

Let

\[
\mathbf{A}_{i,R,B} \mathbf{X}_i = \mathbf{b}_i
\]

(7)

where \( \mathbf{A}_{i,R,B} \) is termed as the Schur complement of \( \mathbf{A} \) (Toselli and Widlund, 2005).

According to Eq. (7), the unknowns of separators (\( \mathbf{X}_i \)) could be solved first, and then be substituted in Eq. (2) to obtain:

\[
\mathbf{A}_{i,R} \mathbf{X}_i - \mathbf{b}_i = - \mathbf{A}_{i,R} \mathbf{X}_i
\]

(8)

Using this method, the solution of a large scale distribution system is simplified to the solution of a series of much smaller subsystems. Therefore, the computation complexity is reduced. As the computational scheme originates from mathematical deduction, theoretically there is no loss of accuracy in computation results. In the course of practical implementation, however, numerical errors are unavoidable. For instance, the matrix factorization required by solving the linear system does cause small numerical errors. Moreover, the factorization errors from the GGA and the decomposition method may not be equivalent, as the former comes from factoring the whole coefficient matrix at one time while the latter is a combination of errors from factorizing all decomposed matrices. Subsequently, the solution of a water distribution simulation is not exactly identical when applying the decomposition method but the differences are marginal. Overall, the domain decomposition method is formulated to solve the standard formulation of matrix equations (e.g. the GGA method) in a more efficient way. Nonetheless, the non-linear convergence is still driven by the original computation scheme (e.g. the GGA method).

3. Results and discussion

Based on two dialectical examples and a real-world case study, the methodology is tested regarding both accuracy and efficiency. The system of linear equations is formulated according to the GGA in order to facilitate a comparison of results between this method and a standard GGA solution as implemented in EPANET.

3.1. Dialectical example — demand-driven formulation

The same network as shown in Fig. 1 is used as the dialectical example in order to avoid that the system of equations becomes too large to be explicitly presented. This case study applies the developed approach to the “demand driven” problem, which is the classical problem of WDS analysis, namely checking whether or not all the available heads at the demand nodes are higher than a fixed minimum value, for a given WDS design. Using this example, the decomposition method is checked by comparing the calculation results with the GGA. As can be seen from Fig. 3, the head of node 1 is 35.35 m, elevations for all nodes are assumed to be 0. Besides, length and roughness of each pipe is set to 1000 m and 100 respectively. For headloss calculation, the Hazen–Williams Formula \( h = 10.67Q^1.852L/\left(C^{1.852}d^{4.87}\right) \) is used, where \( Q \) is the flow rate in m³/s; \( d \) the pipe diameter in meters; and \( l \) the pipe length in meters. Minor losses are not considered in this example. More information is available in Table 1.

3.1.1. Using the Global Gradient Algorithm

After initialization (Table 1), the matrix equation \((\mathbf{D} \cdot \mathbf{A} \times \Delta \mathbf{X} = -\mathbf{F})\) could be written as:

\[
\begin{array}{cccccccccc}
P1 & 10.394 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ P2 & 0 & 10.394 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ P3 & 0 & 0 & 16.074 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ P4 & 0 & 0 & 0 & 74.905 & 0 & 0 & 0 & 0 & 0 & 1 \\ P5 & 0 & 0 & 0 & 0 & 107.219 & 0 & 0 & 0 & 0 & 1 \\ P6 & 0 & 0 & 0 & 0 & 0 & 160.022 & 0 & 0 & 0 & 1 \\ P7 & 0 & 0 & 0 & 0 & 0 & 0 & 107.219 & 0 & 0 & 1 \\ N2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ N3 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ N4 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ N5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ N6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ \end{array}
\]

\[
\begin{array}{cccc}
\Delta Q_1 & 0.61 \\ \Delta Q_2 & -0.39 \\ \Delta Q_3 & -0.13 \\ \Delta Q_4 & 0.88 \\ \Delta Q_5 & 3.11 \\ \Delta Q_6 & 1.91 \\ \Delta Q_7 & 1.89 \\ \Delta H_2 & 0.00 \\ \Delta H_3 & 0.00 \\ \Delta H_4 & 0.00 \\ \Delta H_5 & 0.00 \\ \Delta H_6 & 0.00 \\ \end{array}
\]

Please cite this article in press as: Diao, K., et al., Speedup of water distribution simulation by domain decomposition, Environmental Modelling & Software (2013), http://dx.doi.org/10.1016/j.envsoft.2013.09.025
Notice that the first column in Eq. (9) indicates the order of nodes and pipes. Numbers with prefix “P” refers to pipe ID, “N” refers to node ID. The calculation results are provided in Table 1.

3.1.2. Using the Schur complement domain decomposition method

By partitioning the network graph into two subsystems using pipe 4 and node 4 as separators, the matrix equation (Eq. (9)) is reordered into a bordered, diagonal system (Eq. (10)).

\[
P1 = \begin{bmatrix} 10.394 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T,
\]

\[
P2 = \begin{bmatrix} 0 & 10.394 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T,
\]

\[
P3 = \begin{bmatrix} 0 & 0 & 16.074 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T,
\]

\[
N2 = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T,
\]

\[
N3 = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T,
\]

\[
P5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 107.219 & 0 & 0 & 0 & 0 \end{bmatrix}^T,
\]

\[
P6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 107.219 & 1 & -1 & 0 \end{bmatrix}^T,
\]

\[
N5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^T,
\]

\[
N6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T,
\]

\[
P4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T,
\]

\[
N4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T.
\]

According to Eq. (1), the matrix equation (Eq. (10)) is decomposed into:

\[
F_1 = \begin{bmatrix} [dE_1] \\
[dq_1] \end{bmatrix} = \begin{bmatrix} 0.61 & -0.39 & -0.13 & 0 & 0 \end{bmatrix}^T,
\]

\[
F_2 = \begin{bmatrix} [dE_2] \\
[dq_2] \end{bmatrix} = \begin{bmatrix} -3.11 & 1.91 & 1.89 & 0 & 0 \end{bmatrix}^T,
\]

\[
F_B = \begin{bmatrix} [dE_B] \\
[dq_B] \end{bmatrix} = \begin{bmatrix} 0.88 & 0 \end{bmatrix}^T.
\]

\[
DA_{(1,1)} = \begin{bmatrix} \begin{bmatrix} D_{11} & A_{12} \\
A_{21} & D_{22} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 10.394 & 0 & 0 & 1 & 0 \\
0 & 10.394 & 0 & 0 & 1 \\
0 & 0 & 16.074 & -1 & 1 \\
1 & 0 & -1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 \end{bmatrix},
\]

\[
DA_{(2,2)} = \begin{bmatrix} \begin{bmatrix} D_{11} & A_{12} \\
A_{21} & D_{22} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 107.219 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 160.022 & 0 \end{bmatrix},
\]

\[
DA_{(B,B)} = \begin{bmatrix} \begin{bmatrix} D_{11} & A_{12} \\
A_{21} & D_{22} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 74.905 & 1 \\
0 & 1 \end{bmatrix},
\]

From Eqs. (5) and (6), we obtain:

\[
DA_{(B,B)} = DA_{(B,B)} - A_{(B,1)}^{-1}A_{(1,1)}A_{(1,B)} - A_{(B,2)}^{-1}A_{(2,2)}A_{(2,B)} = \begin{bmatrix} 82.417 & 1 \\
0 & 0 \end{bmatrix},
\]

\[
F_B = F_B - DA_{(B,1)}^{-1}F_1 - DA_{(B,2)}^{-1}F_2 = \begin{bmatrix} -0.731 \end{bmatrix}.
\]

Then, unknowns of the separators could be acquired from solving Eq. (7):

\[
\Delta X_B = -DA_{(B,B)}^{-1}F_B = \begin{bmatrix} 0 & -0.731 \end{bmatrix}.
\]

With boundary conditions defined by \(\Delta X_B\), the unknowns of each subsystem could therefore be computed separately and simultaneously using Eq. (8):

\[
\Delta X_1 = -DA_{(1,1)}^{-1}F_1 - DA_{(1,1)}^{-1}A_{(1,1)}\Delta X_B = \begin{bmatrix} -0.0232 & 0.0232 & -0.0232 & -0.3688 & 0.1488 \end{bmatrix}^T.
\]

\[
\Delta X_2 = -DA_{(2,2)}^{-1}F_2 - DA_{(2,2)}^{-1}A_{(2,2)}\Delta X_B = \begin{bmatrix} 0.0184 & -0.0184 & 0.4013 & 0.3138 \end{bmatrix}^T.
\]

At this moment, all submatrices are updated according to new nodal heads and pipe flows to start the next iteration. The whole process is recorded in Table 2.

3.1.3. Results comparisons

Results comparisons between the two methods are summarized in Tables 3 and 4. It can be seen that the domain decomposition scheme reaches nearly the same result as the GGA does. The maximum differences between the solutions are 0.04 m for nodal heads and 0.4 L/s for pipe flows, respectively. The maximum relative error is 0.53%. Besides, applying the same numerical stopping criteria based on accuracy both methods require the same number of iterations, since the systems of linear equations, which are solved differently, remains unchanged.
3.2. Dialectical example 2 — head-driven formulation

When the head at a node is not sufficiently high to allow delivering the demand, the actual outflow discharge must be reduced accordingly and may even become null. In such cases, the original demand driven formulation must be adapted accordingly. This new formulation is known as the "head driven" formulation (Salgado et al., 1993; Todini, 2006; Giustolisi et al., 2012) and leads

![Fig. 3. The dialectical example network.](image)

Table 1

<table>
<thead>
<tr>
<th>Iteration</th>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td>N1</td>
<td>N2</td>
<td>N3</td>
<td>N4</td>
<td>N5</td>
<td>N6</td>
<td>N7</td>
<td>N8</td>
</tr>
<tr>
<td>d (mm)</td>
<td>600</td>
<td>600</td>
<td>500</td>
<td>400</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>Q (m³/s)</td>
<td>0.17</td>
<td>0.17</td>
<td>0.10</td>
<td>0.17</td>
<td>0.05</td>
<td>0.08</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>nK/Q</td>
<td>10.394</td>
<td>10.394</td>
<td>16.074</td>
<td>74.905</td>
<td>107.219</td>
<td>107.219</td>
<td>160.022</td>
<td>107.219</td>
</tr>
<tr>
<td>H (m)</td>
<td>35.35</td>
<td>35.00</td>
<td>34.00</td>
<td>28.00</td>
<td>22.00</td>
<td>22.00</td>
<td>23.00</td>
<td>21.00</td>
</tr>
<tr>
<td>h (m)</td>
<td>0.35</td>
<td>1.35</td>
<td>1.00</td>
<td>6.00</td>
<td>6.00</td>
<td>6.00</td>
<td>5.00</td>
<td>1.00</td>
</tr>
<tr>
<td>-dE</td>
<td>-0.61</td>
<td>-0.39</td>
<td>0.13</td>
<td>-0.88</td>
<td>3.11</td>
<td>-1.91</td>
<td>-1.89</td>
<td>-0.00</td>
</tr>
<tr>
<td>-dq</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>ΔQ (m³/s)</td>
<td>-0.0036</td>
<td>0.0236</td>
<td>-0.0236</td>
<td>0.0000</td>
<td>0.0185</td>
<td>-0.0185</td>
<td>-0.0185</td>
<td>0.0000</td>
</tr>
<tr>
<td>ΔH (m)</td>
<td>0.0000</td>
<td>-0.3647</td>
<td>0.1447</td>
<td>-0.7753</td>
<td>0.3961</td>
<td>0.3076</td>
<td>0.3076</td>
<td>0.3076</td>
</tr>
</tbody>
</table>

Note: The Hazen-Williams Formula used in this example is \( h = \frac{10.67Q^{1.852}L^{1.852}d^{4.87}}{C^{1.852}d^{4.87}} \), where Q is in m³/s; d is in meters; L is the pipe length in meters.

- Prefix "P" refers to pipe ID.
- Prefix "N" refers to node id.
- d = diameter.
- K = resistance coefficient from the Hazen–Williams Formula.
- Q = flow rate.
- n = flow exponent from the Hazen–Williams Formula, n = 1.852.
- H = nodal head.
- h = headloss (\( H_{ni} - H_{nj} \), Ni, Nj is node ID).
- -dE = pipe balance error.
- -dq = nodal balance error.
- ΔQ = pipe flow correction.
- ΔH = nodal head correction.

Please cite this article in press as: Diao, K., et al., Speedup of water distribution simulation by domain decomposition, Environmental Modelling & Software (2013), http://dx.doi.org/10.1016/j.envsoft.2013.09.025
to matrices with matrix $A_{22}$ (see as the Appendix) being non null (including non-zero elements) as it was in the original GGA. Again, the use of variable speed pumps (typically needed in simulation) leads to a non-null $A_{22}$ matrix (Todini et al., 2007). Subsequently, this case study tests the applicability of the domain decomposition method to the head-driven formulation, in which non-null $A_{22}$ is involved.

The same network as the case study one example is analyzed. The only difference, however, is that the demands at node 4, 5, and 6 are assumed to be head-driven. The head-flow relationship is given by Eq. (A.1), and the corresponding $A_{22}, D_{22}$ are defined by Eqs. (A.2) and (A.3) respectively. Specifically, $Z_i = 0(m), H_4 = 32(m), \text{ and } H_5 = H_6 = 30(m)$. As for the GGA method, the system of equations (Eq. (9)) is then changed to:

\[
\begin{align*}
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta Q_1 \\
6 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \Delta Q_2 \\
5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \Delta Q_3 \\
4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \Delta Q_4 \\
3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta Q_5 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \Delta Q_6 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta Q_7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta Q_8 \\
\end{align*}
\]

After, Eq. (13) could be decomposed into a series of sub-matrices and vectors, following the same way indicated in the case study one. Assigned by the same initial values ($Q$ and $H$ in the HIPS library (Gaidamour and Hénon, 2008) is selected as the linear solver. Table 5 lists the computation time corresponding to different levels of system decomposition.
The method is evaluated for both efficiency and accuracy using several typical performance metrics (Jakeman et al., 2006; Bennett et al., 2013). In terms of efficiency, the decomposition method could accelerate the hydraulic analysis by up to 8.57 times (when the No. of subsystems = 7, see as Table 5). The speedup mainly attributes to the dimensionality reduction mentioned before. Yet, as the test is based on using a stand-alone library HIPS instead of directly modifying the EPANET engine accordingly, it cannot be assured that whole amount of speed-up stems from the advantage of the decomposition method. As computational speed increase is rather difficult to assess precisely, the intention of this study is to demonstrate that the domain decomposition algorithm performs faster on a particular setting of machine, compiler and code implementation. Although the exact amount of speed increase might be different on a different setting, however, speedup is

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Comparison on nodal head results between GGA and DGGA.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node ID</td>
<td>Iterations</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Demand-Driven GGA</strong></td>
<td>35.35</td>
</tr>
<tr>
<td></td>
<td>35.35</td>
</tr>
<tr>
<td><strong>Head-Driven GGA</strong></td>
<td>35.35</td>
</tr>
<tr>
<td></td>
<td>35.35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Comparison on pipe flow results between GGM and DDGM.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node ID</td>
<td>Iterations</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td><strong>Demand-Driven GGA</strong></td>
<td>145.2</td>
</tr>
<tr>
<td></td>
<td>145.6</td>
</tr>
<tr>
<td><strong>Head-Driven GGA</strong></td>
<td>140.0</td>
</tr>
<tr>
<td></td>
<td>140.0</td>
</tr>
</tbody>
</table>

Note: The Hazen-Williams Formula used in this example is $h = \frac{1}{10.67Q^{0.852}L}{C^{1.852}d^{0.487}}$, where Q is in $m^3/s$; d is in meters; L is the pipe length in meters.

a Prefix “P” refers to pipe ID.
b Prefix “N” refers to node ID.
c $d$ – diameter.
d $K$ – resistance coefficient from the Hazen–Williams Formula.
e $Q$ – flow rate.
f $n$ – flow exponent from the Hazen–Williams Formula, $n = 1.852$.
g $H$ – nodal head.
h $h$ – headloss ($H_{Hi} - H_{Nj}, Ni, Nj$ is node ID).
i $\Delta e$ – pipe balance error.
j $\Delta q$ – nodal balance error.
k $\Delta Q$ – pipe flow correction.
l $\Delta H$ – nodal head correction.

The method is evaluated for both efficiency and accuracy using several typical performance metrics (Jakeman et al., 2006; Bennett et al., 2013). In terms of efficiency, the decomposition method could accelerate the hydraulic analysis by up to 8.57 times (when the No. of subsystems = 7, see as Table 5). The speedup mainly attributes to the dimensionality reduction mentioned before. Yet, as the test is based on using a stand-alone library HIPS instead of directly modifying the EPANET engine accordingly, it cannot be assured that whole amount of speed-up stems from the advantage of the decomposition method. As computational speed increase is rather difficult to assess precisely, the intention of this study is to demonstrate that the domain decomposition algorithm performs faster on a particular setting of machine, compiler and code implementation. Although the exact amount of speed increase might be different on a different setting, however, speedup is...
the case study, the proper level of decomposition is to create subsystems in which the average numbers of vertices account for at least 1.90% (1207/63616) of the total number of vertices. Dividing the system into a few subsystems (e.g. 2) may lead to tremendous size differences among subsystems (Standard Deviation > 30,000) and therefore imbalanced computational costs for solving each of all subsystems. Contrarily, to solve a vast number of subsystems sequentially, also triggers additional computational expense, although the sizes of subsystems tend to become more homogeneously distributed (e.g. Standard Deviation = 57.8 when the No. of subsystems = 233) along with further decomposition. Besides, the number of separators may get too large, and hence the Schur complement has to be solved in comparatively low efficiency.

The results of this test have shown that the Schur complement domain decomposition approach can improve the efficiency of hydraulic simulation without losing accuracy, in spite of the additional cost due to the division into subsystems. Table 6 concludes pros and the limitations of the approach.

### 4. Conclusion

This paper introduced a systematic approach that could reduce the computation complexity of hydraulic analysis substantially. Instead of solving the whole system at once (like in the traditional solution methods), the Schur complement domain decomposition method divides the system into smaller subsystems which could then be solved sequentially. Exemplified by case studies, it was shown that:

1) The approach is reliable as the final calculation results are almost identical to the conventional method, for both demand-driven and head-driven models.

2) By solving the standard formulation of matrix equations (e.g. the GGA method) in a more efficient way, considerable speedup is achieved by applying the method, i.e. app. 4–8 times faster.

3) For a specific water distribution system, an optimal level of domain decomposition can be defined, as any further

---

**Table 5**

<table>
<thead>
<tr>
<th>No. of subsystems</th>
<th>Average sizes of subsystems</th>
<th>Standard deviation of subsystems’ sizes</th>
<th>Computation time (ms)</th>
<th>Time required for partitioning (ms)</th>
<th>Sizes of the separators (X_a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>31809</td>
<td>31798.0</td>
<td>2583</td>
<td>7.97</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>21218</td>
<td>14839.5</td>
<td>2417</td>
<td>8.07</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>9099</td>
<td>2982.7</td>
<td>1236</td>
<td>8.80</td>
<td>29</td>
</tr>
<tr>
<td>14</td>
<td>4554</td>
<td>1293.8</td>
<td>1536</td>
<td>9.53</td>
<td>59</td>
</tr>
<tr>
<td>24</td>
<td>2660</td>
<td>703.6</td>
<td>1470</td>
<td>10.88</td>
<td>112</td>
</tr>
<tr>
<td>53</td>
<td>1207</td>
<td>323.2</td>
<td>1262</td>
<td>12.58</td>
<td>215</td>
</tr>
<tr>
<td>104</td>
<td>616</td>
<td>156.2</td>
<td>1634</td>
<td>13.37</td>
<td>379</td>
</tr>
<tr>
<td>233</td>
<td>276</td>
<td>57.8</td>
<td>2054</td>
<td>14.25</td>
<td>745</td>
</tr>
<tr>
<td>EPANET</td>
<td></td>
<td></td>
<td>10593</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please cite this article in press as: Diao, K., et al., Speedup of water distribution simulation by domain decomposition, Environmental Modelling & Software (2013). http://dx.doi.org/10.1016/j.envsoft.2013.09.025
Acknowledgments

A.1 The Global Gradient Algorithm (GGA)

fl

combining individual energy equations for each pipe with the
neously (Boulos et al. 2006; Ormsbee, 2006). Following the
could be maintained as well.

Summarizing, this methodology is an interesting alternative to
increase the computational efficiency of hydraulic water dis-
tribution simulation. It is noteworthy that the method has a huge po-
tential to be a parallel computing scheme by allowing sub-domains
being solved simultaneously, but further studies are required to
explore the efficiency of such an implementation. This is because
currently the overhead communication is still a bottleneck for
parallel solution of water distribution systems simulation (Alonso
et al., 2000; Crous et al., 2011).

Appendix A. The Schur complement domain decomposition
scheme for the global gradient algorithm

A.1 The Global Gradient Algorithm (GGA)

The Gradient Method's computation scheme derives from
combining individual energy equations for each pipe with the
individual nodal equations for each junction node so that all un-
known pipe flows and nodal heads could be solved simulta-
neously (Boulos et al. 2006; Ormsbee, 2006). Following the
formulation of the GGA given in Todini and Pilati (1988) and
Todini (2006), the solution scheme of the method can be illust-
trated as follows:

Flowchart of the solution scheme

Set initial values for all unknowns, \( \begin{bmatrix} Q_1(m-0) \\ H_1(m-0) \end{bmatrix} \)\n
Determine the value of the objective function \( F \)

\( F = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} Q \\ H \end{bmatrix} - \begin{bmatrix} A_{10} H_0 \\ -q^* \end{bmatrix} = \begin{bmatrix} dE \\ dq^* \end{bmatrix} \to 0 \)

Define stopping criteria \( |\Delta Q| \leq \xi_1 \) and \( |\Delta H| \leq \xi_2 \)

(\( \xi_1, \xi_2 \) could be arbitrary small numbers)

while \( |\Delta Q| > \xi_1 \) or \( |\Delta H| > \xi_2 \) do

Calculate DA \( = \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix} \)

Solve \( \begin{bmatrix} Q(m-1) \\ H(m-1) \end{bmatrix} = \begin{bmatrix} Q(m) \\ H(m) \end{bmatrix} + \begin{bmatrix} \Delta Q \\ \Delta H \end{bmatrix} \)

Update F and DA with \( \begin{bmatrix} Q(m+1) \\ H(m+1) \end{bmatrix} \)

end while

In the objective function:

\[
Q = [Q_1, Q_2, \ldots, Q_{n_{pipe}}]^T \text{ - the } [n_{pipe}] \text{ column vector of unknown pipe flow rates;}
\]

\[
H = [H_1, H_2, \ldots, H_{n_{node}}]^T \text{ - the } [n_{node}] \text{ column vector of unknown nodal heads;}
\]

\[
H_0 = [H_{node_{up-1}}, H_{node_{up-2}}, \ldots, H_{node_{up+1}}]^T \text{ - the } [n_{node_{up}} - n_{node_{down}}] \text{ column vector of known nodal heads;}
\]

\( q^* \) is a \([n_{node}] \) vector whose generic element is defined as, the
actual demand, if the relevant node is not head-driven, or is equal
zero otherwise;

\( dq^* \) and \( dH \) refers to the nodal balance error and pipe balance
equation, respectively.

\( A_{12} \) — edge-to-non-source-node connectivity matrix; \( A_{10} \) —
edge-to-source-node connectivity matrix; In both connectivity
matrices, each row corresponds to a pipe, and values of \(-1,1,0\) are
assigned to the upstream node, outlet node, and unconnected node
of the pipe respectively. \( A_{11} = A_{22}^T \), \( A_{11} = \text{diag}(s_1|q_1|^{n-1}, \ldots, s_{n_{pipe}}|q_{n_{pipe}}|^{n-1}) \). The elements of matrix \( A_{22} \) are either zero, if the
relevant nodal demand is not pressure-driven, or a non-linear
function of the expression representing the pressure-flow rela-
tionship. For instance, to use the expression defined by Eq. (A.1), the
corresponding element of matrix \( A_{22} \) becomes [Eq. (A.2)]:

\[
q_{i act} = \begin{cases} 0, & H_i \leq Z_i \\ q_i(H_i-Z_i)^{1/2}, & Z_i < H_i < H_i^* \\ q_i, & H_i \geq H_i^* \end{cases} \quad (A.1)
\]

\[
A_{22}(i,i) = \begin{cases} 0, & H_i \leq Z_i \\ -q_i(H_i-Z_i)^{1/2}, & Z_i < H_i < H_i^* \\ q_i, & H_i \geq H_i^* \end{cases} \quad (A.2)
\]

\[
D_{22}(i,i) = \begin{cases} 0, & H_i \leq Z_i \\ -q_i(H_i-Z_i)^{1/2}, & Z_i < H_i < H_i^* \\ 0, & H_i \geq H_i^* \end{cases} \quad (A.3)
\]

A.2 The Schur Complement Domain Decomposition Scheme for the Global Gradient Algorithm

For this case, all items in Eq. (1) are specifically defined as fol-
low: \( A_{ij} \) and \( DA_{ij} \) are the coefficient matrices of subsystem \( i \),
\( A_{1i} = \begin{bmatrix} A_{1i}^{(1)} & A_{1i}^{(2)} \\ A_{2i}^{(1)} & A_{2i}^{(2)} \end{bmatrix} \) and \( DA_{ij} = \begin{bmatrix} D_{1i}^{(1)} & D_{1i}^{(2)} \\ D_{2i}^{(1)} & D_{2i}^{(2)} \end{bmatrix} \), and \( N \) is the
number of subsystems; \( A_{ii} \) and \( DA_{ii} \) are the coefficient
matrices of all boundary networks, \( A_{ii} = \begin{bmatrix} A_{ii}^{(1)} & A_{ii}^{(2)} \\ A_{ii}^{(2)} & A_{ii}^{(2)} \end{bmatrix} \),
\( DA_{ii} = \begin{bmatrix} D_{ii}^{(1)} & D_{ii}^{(2)} \\ D_{ii}^{(2)} & D_{ii}^{(2)} \end{bmatrix} \); \( A_{ii}^{(1)} \) represents the coupling between
subsystem \( i \) and boundary networks, in which the columns

Please cite this article in press as: Diao, K., et al., Speedup of water distribution simulation by domain decomposition, Environmental Modelling & Software (2013), http://dx.doi.org/10.1016/j.envsoft.2013.09.025
correspond to boundary pipes followed by boundary nodes and the rows correspond to pipes and nodes in subsystem $i$. For nodes shared by pipes in the subsystem and boundary networks, value of $-1$ is assigned to upstream nodes of pipes in the subsystem, 1 is assigned to outlet nodes of those pipes, and 0 for the other cases. $A_{B(i)} = A_{B(i)}^T$, $X_{B(i)} = \begin{bmatrix} Q_{B(i)} \\ H_{B(i)} \end{bmatrix}$ and $X_{B} = \begin{bmatrix} Q_{B} \\ H_{B} \end{bmatrix}$ are the unknowns for subsystem $i$ and boundary networks respectively, and $\Delta X_{B(i)}$, $\Delta X_{B}$ are corresponding corrections; $F_{(i)}$ and $F_{B}$ are balance errors for nodes and pipes in subsystem $i$ and in boundary networks respectively.

$F_{(i)} = \begin{bmatrix} \frac{dE_{(i)}}{dq_{(i)}} \\ \frac{dE_{(i)}}{dq_{B(i)}} \end{bmatrix}$ $F_{B} = \begin{bmatrix} \frac{dE_{B}}{dq_{B}} \end{bmatrix}$.

The new solution scheme is then written as:

Flowchart of the solution scheme of domain decomposition method

Set initial values for all unknowns, $X_{(i)} = \begin{bmatrix} Q_{(m-0)}^{(m-0)} \\ H_{(m-0)}^{(m-0)} \end{bmatrix}$.

Reordering elements in the objective function $F$:

$F = \begin{bmatrix} F_{(1)} \\ F_{(2)} \\ \vdots \\ F_{(N)} \\ F_{B} \end{bmatrix} = \begin{bmatrix} A_{1.1} & 0 & \cdots & 0 \\ 0 & A_{2.2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_{N,N} \\ A_{B.1} & A_{B.2} & \cdots & A_{B,N} \\ A_{B,B} \end{bmatrix}$

$X_{(i)} = \begin{bmatrix} b_{1(i)} \\ b_{2(i)} \\ \vdots \\ b_{N(i)} \\ b_{B(i)} \end{bmatrix}$

$b_{(i)} = [-A_{10}^{(i)}H_{0}^{(i)} - q_{1(i)}^{(i)}]^T$ and $b_{B} = [-A_{B0}^{(B)}H_{0}^{(B)} - q_{B}^{(B)}]^T$

Reordering $DA \Delta X = -F$ into:

$DA \times \Delta X = \begin{bmatrix} DA_{1.1} & 0 & \cdots & 0 \\ 0 & DA_{2.2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & DA_{N,N} \\ A_{B.1} & A_{B.2} & \cdots & A_{B,N} \\ A_{B,B} \end{bmatrix} \begin{bmatrix} \Delta X_{1} \\ \Delta X_{2} \\ \vdots \\ \Delta X_{N} \\ \Delta X_{B(i)} \\ \Delta X_{B} \end{bmatrix} = \begin{bmatrix} F_{(1)} \\ F_{(2)} \\ \vdots \\ F_{(N)} \\ F_{B} \end{bmatrix}$

Determine the values of the objective functions $F_{(i)}$ and $F_{B}$

$F_{(i)} = A_{(i,i)}X_{(i)} - b_{(i)} \rightarrow 0$

$F_{B} = A_{B,B}X_{B} - b_{B} \rightarrow 0$

Define stopping criteria $|\Delta Q| \leq \xi_1$ and $|\Delta H| \leq \xi_2$ ($\xi_1$, $\xi_2$ could be arbitrary small numbers)

while $|\Delta Q| > \xi_1$ or $|\Delta H| > \xi_2$ do

Compute $DA_{(i,i)}$ and $DA_{B,B}$

Solve $DA_{(i,i)} \Delta X_{(i)} = DA_{(i,0)}F_{(i)}$

Solve $DA_{B,B} \Delta X_{B} = DA_{B,0}F_{B}$

Update $X_{(i)}^{(m+1)} = X_{(i)}^{(m)} + \Delta X_{(i)}$, $X_{B}^{(m+1)} = X_{B}^{(m)} + \Delta X_{B}$.

end while

References


Please cite this article in press as: Diao, K., et al., Speedup of water distribution simulation by domain decomposition, Environmental Modelling & Software (2013), http://dx.doi.org/10.1016/j.envsoft.2013.09.025