WATER RESOURCES RESEARCH, VOL. 49, 2783–2800, doi:10.1002/wrcr.20250, 2013

Quantification of the upstream-to-downstream influence in the Muskingum method and implications for speedup in parallel computations of river flow

Cédric H. David,[1,2] Zong-Liang Yang,[1] and James S. Famiglietti[2,3]

Received 24 July 2012; revised 8 April 2013; accepted 11 April 2013; published 28 May 2013.

[1] The mathematical formulation of the Muskingum method, like that of many numerical schemes used for river routing, requires that all upstream river reaches be updated prior to updating the flow rate of any given reach. Due to this topological constraint, such numerical schemes have traditionally been solved in an upstream-to-downstream manner which imposes inherent limitations on the speedup that can be achieved in a parallel computing environment because each computing core has to wait for completion of all cores addressing upstream subbasins prior to starting its own subbasin. The research presented in this paper quantifies the exact influence among river reaches during the update step of the Muskingum method and shows that the influence decreases with increasing distance between two reaches until it becomes too small to be accounted for by floating-point arithmetic. A formal definition of the minimal distance from which the relative influence becomes numerically inexistent—the radius of influence—is presented. Based on this distance, expressed as a number of river reaches, a new estimate of the maximum theoretical speedup that can be achieved by the Muskingum method or by similar numerical schemes is presented and implies large potential gains in computing time when domains are much larger than the radius of influence. An application to the approximately 180,000 river reaches of the Upper Mississippi River Basin at a 15 min time step over 2004 shows a radius of influence on the order of 150 river reaches. The speedup obtained for this application is much higher than previously thought possible but also much lower than could be attained, suggesting that further investigations are necessary.

Citation: David, C. H., Z.-L. Yang, and J. S. Famiglietti (2013), Quantification of the upstream-to-downstream influence in the Muskingum method and implications for speedup in parallel computations of river flow, *Water Resour. Res.*, *49*, 2783–2800, doi:10.1002/wrcr.20250.

1. Introduction

[2] Over the past decade, parallel computers have become increasingly accessible to scientists, and their use for studying water resources is progressively becoming common. Generally, parallel computing is used to address problems larger than could otherwise be tackled in which case scalability is of concern; or it is used to solve problems of constant size but in less time in which case speedup is studied. Many aspects of the terrestrial water cycle—such as the surface and subsurface water balances—have already benefitted from parallel computing, and the reader is referred to *Vivoni et al*. [2011] or *David et al*. [2011b] for a more detailed general overview of parallel computing in terrestrial hydrology. Few studies, however, focus on the use of parallel computing for river flow modeling and on related specific speedup issues as done in this paper. One way to leverage parallel computing for river routing is to assign disconnected river basins to different computing cores, therefore requiring no time-consuming intercore communication, but this ensues in load imbalance, and performance becomes limited by the largest river basins [e.g., *Larson et al*., 2007]. Hence, the study of parallel computing methods applied to river modeling within large basins is of interest. The main challenge of such an endeavor is that rivers flow mostly from upstream to downstream; therefore, one may have to account for network connectivity and direction of flow when assigning tasks to different computing cores, depending on the type of numerical scheme used for river routing. These schemes generally update a given river reach based on a combination of some or all of the terms corresponding to the same reach at previous time step, the upstream reaches at previous time step, and the upstream reaches at update time step. The simplest numerical schemes—referred to as type 1 here—update a given river reach based on the same reach at previous time step

[1]Department of Geological Sciences, Jackson School of Geosciences, University of Texas at Austin, Austin, Texas, USA.
[2]University of California Center for Hydrologic Modeling, University of California, Irvine, California, USA.
[3]Department of Earth System Science, University of California, Irvine, California, USA.

Corresponding author: C. H. David, University of California Center for Hydrologic Modeling, 240F Rowland Hall, Irvine, CA 92697-4690, USA. (chdavid@uci.edu)

0043-1397/13/10.1002/wrcr.20250

and on the upstream reaches at previous time step and hence allow computations to be updated all at once [e.g., *Larson et al*., 2007; *von Bloh et al*., 2010]. In contrast, more accurate numerical schemes—referred to as type 2 here—update a given reach based on the same reach at previous time step and on upstream reaches at both the previous and the updated time step and hence require that computations be carried from upstream to downstream in order [e.g., *Li et al*., 2011]. Both type 1 and type 2 numerical schemes necessitate information concerning those river reaches that are located directly upstream of a given reach, but at the *previous* time step for type 1, and at the *previous and update* time steps for type 2. The differentiation made here therefore only reflects on if the terms corresponding to the upstream reaches at update time step are used (type 2) or are not (type 1). The maximum theoretical speedup of type 1 schemes is ideal, i.e., the computing time decreases by a factor exactly equal to the number of computing cores. In the case of type 2 schemes, however, the estimates of maximum theoretical speedup have traditionally accounted for the necessity to wait for prior upstream updates and are therefore much lower that the speedup of type 1 schemes. *Larson et al*. [2007] used a type 1 scheme on disconnected basins which resulted in load imbalance. *von Bloh et al*. [2010] also used a type 1 scheme but split computations of each separate basin into a few processing cores to balance computing loads and accounted for necessary communications at subbasin boundaries. *Li et al*. [2011] used a type 2 scheme in which the subbasin that each processing core addresses changes at each time step in an upstream-to-downstream sequence. In that case, careful algorithms for choosing the ordering of computations [e.g., *Li et al*., 2010; *Veitzer and Gupta*, 2001] can help maximize parallel performance, but speedup is traditionally assumed to be limited by river network topology. *David et al*. [2011b] also used a type 2 scheme but virtually avoided sequential upstream-to-downstream computations through the use of an iterative solver on multiple computing cores and showed better speedup results than was predicted by existing theory. *David et al*. [2011b] proposed that such speedup may be explained by flow waves not being fast enough to propagate through the entire domain within any given time step but failed to provide a proof.

[3] The purpose of this paper is to quantify the exact upstream-to-downstream influence among river reaches during the update between two consecutive time steps of the Muskingum method. We put the magnitude of this influence in the perspective of precision in computer operations and present a new estimate of the maximum theoretical speedup that can be obtained when performing river routing computations that follow a numerical scheme in which the update of any given river reach requires prior update of all directly upstream reaches.

[4] This paper first provides a theoretical approach, followed by a small theoretical example. A practical application focusing on the Upper Mississippi River Basin over the year 2004 at a 15 min time step and using approximately 180,000 river reaches (Figure 1) is then presented. Finally, practical results, discussion, and conclusions are given.

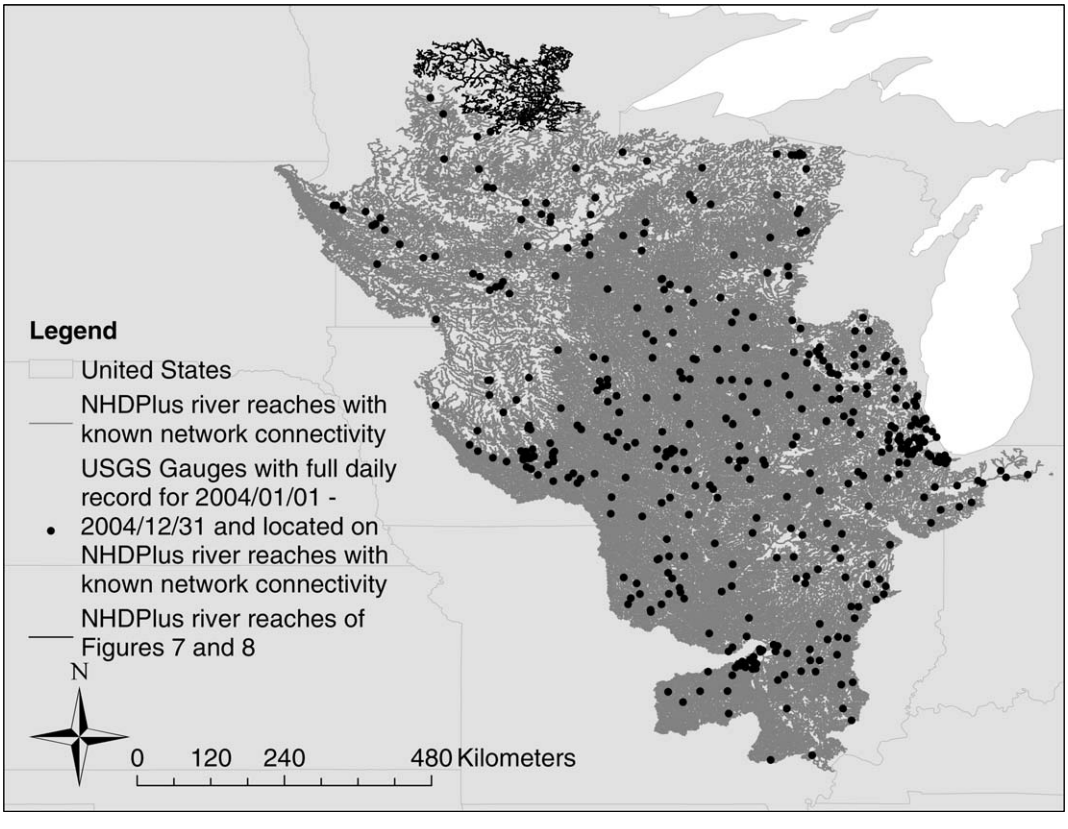

Figure 1. The Upper Mississippi River Basin is the domain of this study. All NHDPlus river reaches and USGS gauges used are shown, as well as the specific river reaches used in Figures 7 and 8.

2. Theoretical Approach

2.1. Muskingum Method as a Linear System of Equations

[5] The Muskingum method [*McCarthy*, 1938] is a river routing scheme in which flow waves propagate in a river reach j during the computing time step Δt as a function of a temporal parameter k_j and of a dimensionless parameter x_j following equation (1).

$$\begin{aligned} Q_j(t+\Delta t) = C_{1_j} \cdot \left(Q_j^{\text{up}}(t+\Delta t) + Q_j^e(t)\right) \\ + C_{2_j} \cdot \left(Q_j^{\text{up}}(t) + Q_j^e(t)\right) + C_{3_j} \cdot Q_j(t), \end{aligned} \tag{1}$$

where t is time and Δt is strictly positive. The variable Q_j^{up} is the flow entering reach j and coming from upstream river reaches. The formulation of the Muskingum method given in equation (1) includes the inflow of water Q_j^e coming from outside of the river network into a given reach j (e.g., lateral inflow from runoff and river/aquifer exchanges) and added upstream. Such inclusion of Q_j^e in the Muskingum method as un upstream contribution is in effect identical to approaches using time-averaged lateral inflow [*National Environment Research Council*, 1975; *Ponce*, 1986] or time-varying lateral inflow [*O'Donnell*, 1985; *Orlandini and Rosso*, 1998]. These previous approaches only differ by the time at which Q_j^e is specified in equation (1), i.e., $Q_j^e(t)$ as done here, $Q_j^e(t+\Delta t)$ as in *Orlandini and Rosso* [1998], or both as in *O'Donnell* [1985]. In this study, Q_j^e is computed by a land surface model and is assumed to be slowly time-varying; the impact of the time at which Q_j^e is specified is hence small. The variable Q_j is the flow exiting reach j and therefore represents the flow at the node downstream of a given river reach j, although it is loosely referred to here as the flow at reach j in order to simplify the wording. The parameters C_{1_j}, C_{2_j}, and C_{3_j} are constant parameters computed using equation (2).

$$\begin{aligned} C_{1_j} &= \frac{\Delta t/2 - k_j \cdot x_j}{k_j \cdot (1-x_j) + \Delta t/2}, \quad C_{2_j} = \frac{\Delta t/2 + k_j \cdot x_j}{k_j \cdot (1-x_j) + \Delta t/2}, \\ C_{3_j} &= \frac{k_j \cdot (1-x_j) - \Delta t/2}{k_j \cdot (1-x_j) + \Delta t/2}. \end{aligned} \tag{2}$$

[6] One should note that the sum $C_{1_j} + C_{2_j} + C_{3_j}$ always leads to a value of one. With the Muskingum method, updating the flow in any given river reach necessitates access to the updated flow in river reaches that are located directly upstream as can be seen from the first element of the right-hand side in equation (1). For this reason, solving the Muskingum method using computers has traditionally been done sequentially from upstream to downstream on one unique computing core.

[7] Adapting equation (1) using a matrix notation, *David et al*. [2011b] showed that the Muskingum method can be adapted to equation (3), which can be solved in parallel by splitting the vectors and matrices on multiple computing cores and allowing for intercore communication.

$$\begin{aligned} (\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}) \cdot \mathbf{Q}(t+\Delta t) = \mathbf{C_1} \cdot \mathbf{Q^e}(t) + \mathbf{C_2} \cdot [\mathbf{N} \cdot \mathbf{Q}(t) + \mathbf{Q^e}(t)] \\ + \mathbf{C_3} \cdot \mathbf{Q}(t), \end{aligned} \tag{3}$$

where the bold notation is used for vectors and square matrices which are all of size m, the total number of river reaches in a river network. The matrix $\mathbf{I}$ is the identity matrix; $\mathbf{C_1}$, $\mathbf{C_2}$, and $\mathbf{C_3}$ are diagonal matrices in which diagonal elements are C_{1_j}, C_{2_j}, and C_{3_j}, respectively; $\mathbf{N}$ is the network matrix in which—assuming a maximum of one downstream river reach being allowed for each river reach—a value of one is used at row i and column j if reach j flows into reach i and zero is used elsewhere. The vector $\mathbf{Q}$ (respectively, $\mathbf{Q^e}$) is made of the corresponding elements Q_j (respectively, Q_j^e). The partial temporal uniformity of Q_j^e simplifies the river routing model formulation, limits the quantity of input data, facilitates the coupling with land surface models, and is a valid assumption if such information is made available less often that the river routing time step as in this study. One should note that equation (1) allows for solving "all river reaches, one time step after another" or for solving "all time steps, one river reach after another," whereas equation (3) is limited to the former. By combining all elements on the right-hand side of equation (3) into a single vector $\mathbf{b}$, the following equation is obtained:

$$(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}) \cdot \mathbf{Q}(t+\Delta t) = \mathbf{b}(t). \tag{4}$$

[8] Solving for the unknown $\mathbf{Q}(t+\Delta t)$ in equation (4) can be done by providing the highly sparse matrix $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ and the right-hand side $\mathbf{b}(t)$ to a linear system solver. Equations (3) and (4) are valid regardless of the ordering of $\mathbf{N}$. However, $\mathbf{N}$ can be made strictly lower triangular—i.e., all elements on and above the diagonal are zero—by ordering the river reaches in an upstream-to-downstream manner which will be assumed in the following. More information on the derivation of this matrix-based Muskingum method can be found in *David et al*. [2011b].

2.2. Inversion of This Linear System of Equations

[9] The matrix $\mathbf{C_1}$ being a diagonal matrix, and $\mathbf{N}$ being a strictly lower triangular matrix, the product $\mathbf{C_1} \cdot \mathbf{N}$ is strictly lower triangular (see Appendix A) and $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ is a lower unit triangular matrix (i.e., a lower triangular matrix for which all diagonal elements have a value of one). The determinant of a triangular matrix is the product of its diagonal elements; therefore, $\det(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}) = 1$ which is not null, and hence, the inverse of $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ exists and is unique. This is not surprising because the Muskingum method can be solved explicitly from upstream to downstream, as done traditionally.

[10] The characteristic polynomial $P_{\mathbf{I}-\mathbf{C_1}\cdot\mathbf{N}}(y)$ of $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ is defined by

$$P_{\mathbf{I}-\mathbf{C_1}\cdot\mathbf{N}}(y) = \det(y \cdot \mathbf{I} - (\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N})). \tag{5}$$

[11] Accounting for $\mathbf{C_1} \cdot \mathbf{N}$ being strictly lower triangular and for the determinant of a triangular matrix being the product of its diagonal elements, equation (5) can be simplified as

$$\begin{aligned} P_{\mathbf{I}-\mathbf{C_1}\cdot\mathbf{N}}(y) &= \det((y-1) \cdot \mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}) = \det((y-1) \cdot \mathbf{I}) \\ &= (y-1)^m. \end{aligned} \tag{6}$$

[12] Following the Caley-Hamilton theorem, $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ satisfies its own characteristic polynomial:

$$P_{\mathbf{I}-\mathbf{C_1}\cdot\mathbf{N}}(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}) = \mathbf{0}, \tag{7}$$

where $\mathbf{0}$ is the null matrix. Substituting equation (7) into equation (6) leads to

$$P_{\mathbf{I}-\mathbf{C_1}\cdot\mathbf{N}}(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}) = (-1)^m \cdot (\mathbf{C_1} \cdot \mathbf{N})^m = \mathbf{0} \Longleftrightarrow (\mathbf{C_1} \cdot \mathbf{N})^m = \mathbf{0}. \tag{8}$$

[13] Taking equation (8) into account, one can then show that

$$\left(\mathbf{I} + \sum_{p=1}^{p=m-1} (\mathbf{C_1} \cdot \mathbf{N})^p\right) \cdot (\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}) = \mathbf{I}, \tag{9}$$

where p is an integer. The unique inverse of $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ has therefore been determined. This inverse can be seen as the Muskingum operator $\mathbf{M}$, which like $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$, is a unit lower triangular matrix:

$$\mathbf{M} = (\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N})^{-1} = \left(\mathbf{I} + \sum_{p=1}^{p=m-1} (\mathbf{C_1} \cdot \mathbf{N})^p\right). \tag{10}$$

[14] Finally, the linear system of equation (4) can be inverted as shown in equation (11).

$$\mathbf{Q}(t + \Delta t) = \mathbf{M} \cdot \mathbf{b}(t). \tag{11}$$

[15] Despite being presented here for time-invariant Muskingum parameters, one should expect similar results if $\mathbf{C_1}$ varied with time in which case the linear system matrix $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ and the Muskingum operator $\mathbf{M}$ should be updated at every time step, although such is beyond the scope of this study.

[16] Albeit being lower triangular, $\mathbf{M}$ is much denser than its highly sparse inverse $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$. Therefore, for computer performance purposes, a river routing scheme should solve equation (4) using a linear system solver rather than solving equation (11) directly. However, despite its high associated computational cost, $\mathbf{M}$ can be used to quantify the exact relative influence among river reaches and is therefore useful in the context of this study.

2.3. Concerning the Precision That Can Be Expected From the Muskingum Method

[17] The Muskingum method—whether in its traditional form of equation (1) or in the matrix-based form of equation (4)—is a linear system of equations in which the matrix $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ and the right-hand side $\mathbf{b}(t)$ are given and the unknown $\mathbf{Q}(t + \Delta t)$ is to be determined. The properties of this linear system are properties of the linear equations to be solved (i.e., the physical problem being considered) and not of the mathematics used to solve them. One such property is the relationship linking small relative variations of $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ and of $\mathbf{b}(t)$ to those of $\mathbf{Q}(t + \Delta t)$. The corresponding inherent stability (or instability) of the linear system is generally referred to as its condition (or ill condition) and measured by a quantity called the condition number κ defined by

$$\kappa(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}) = \|\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}\| \cdot \|(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N})^{-1}\|, \tag{12}$$

where $\|\|$ is a norm. The paper by *Turing* [1948] and the book by *Wilkinson* [1963] have both been helpful to the authors in understanding the inception of the condition number, and the reader is referred to these for additional information. However, we found that the formulation of the relationship linking small relative variations of $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$, $\mathbf{b}(t)$ and $\mathbf{Q}(t + \Delta t)$ to the condition number is more clearly presented in *Higham* [1990] than in these earlier studies and is hence better suited here. The formulation of *Higham* [1990] is

$$\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} \le \varepsilon, \frac{\|\Delta(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N})\|}{\|\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}\|} \le \varepsilon, \quad \frac{\|\Delta\mathbf{Q}\|}{\|\mathbf{Q}\|} \le \frac{2 \cdot \varepsilon \cdot \kappa(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N})}{1 - \varepsilon \cdot \kappa(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N})}, \tag{13}$$

where ε is a small real number, and the same norm type is used in equations (12) and (13) for both vectors and matrices. The numerical value of the condition number therefore depends on the linear system being considered as well as on the norm being used. In the case of the 2-norm, equation (12) can be expressed as

$$\kappa_{\|\|_2}(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}) = \frac{\sigma_{\max}}{\sigma_{\min}}, \tag{14}$$

where $\sigma_{\min}$ and $\sigma_{\max}$ are, respectively, the minimum and maximum singular values of $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$. The determination of the exact singular values for $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ is nontrivial and outside of the scope of this study. However, a well-established scientific library is used in section 5 to estimate the singular values and hence the condition number, therefore providing an estimate of the bounds relating relative variations of $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$, $\mathbf{b}(t)$, and $\mathbf{Q}(t + \Delta t)$.

[18] Variations of $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ and $\mathbf{b}(t)$ are to be expected among various computer models. For example, the value of $\mathbf{b}(t)$ is prone to small relative variations depending on the ordering of the floating-point operations used for its computation. Computing $\mathbf{b}(t)$ using $\mathbf{b}(t) = \mathbf{C_1} \cdot \mathbf{Q^e}(t) + \mathbf{C_2} \cdot [\mathbf{N} \cdot \mathbf{Q}(t) + \mathbf{Q^e}(t)] + \mathbf{C_3} \cdot \mathbf{Q}(t)$ or using $\mathbf{b}(t) = [\mathbf{C_1} + \mathbf{C_2}] \cdot \mathbf{Q^e}(t) + \mathbf{C_2} \cdot \mathbf{N} \cdot \mathbf{Q}(t) + \mathbf{C_3} \cdot \mathbf{Q}(t)$ would not lead to the exact same results in a floating-point environment (see section 2.7), despite these two expressions being mathematically equivalent. Therefore, the exact floating-point value of $\mathbf{b}(t)$ will differ from one computer model to another. Such differences are also bound to exist for $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$. The resulting relative differences in the values of $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ and $\mathbf{b}(t)$ will generate relative differences in $\mathbf{Q}(t + \Delta t)$ of a magnitude linked to the condition number of the Muskingum method. Consequently, one should not hope to achieve better accuracy in the computation of $\mathbf{Q}(t + \Delta t)$ than that allowed by equation (13). Such differences are not only expected, they are also unavoidable.

[19] In addition to these precision issues which are inherent to the physical problem represented by the Muskingum method, one should also expect small variations in the

values computed for $\mathbf{Q}(t+\Delta t)$ from the various mathematical ways of solving the linear system, i.e., directly with equation (1) or equation (11); or using a linear system solver with equation (4). Such matters are also discussed in general terms in *Turing* [1948] and *Wilkinson* [1963]. However, variations in $\mathbf{Q}(t+\Delta t)$ resulting from the mathematics of resolution that are of the same order of magnitude as those due to small variations of the linear system matrix or of the right-hand side can therefore be deemed acceptable.

2.4. Calculation of the Muskingum Operator on Computers

[20] In river networks with m river reaches, computing the elements of $\mathbf{M}$ based on equation (10) necessitates on the order of $(m+1)\cdot m/2$ multiplications of matrices of size m. In large river networks—e.g., m on the order of 100,000—equation (10) would therefore be highly computationally demanding, and an alternate method is preferable. The matrix $\mathbf{I}-\mathbf{C_1}\cdot\mathbf{N}$ being the inverse of $\mathbf{M}$, their product is the unit matrix, which can be expended in the following equality:

$$\mathbf{M}=\mathbf{I}+(\mathbf{C_1}\cdot\mathbf{N})\cdot\mathbf{M}. \tag{15}$$

[21] By definition of sums and products of matrices, applying equation (15) for each element $[\mathbf{M}](i,j)$ of $\mathbf{M}$ gives

$$\forall(i,j)\in[1,m]^2,\quad [\mathbf{M}](i,j)=\delta(i,j)+\sum_{q=1}^{q=m}([\mathbf{C_1}\cdot\mathbf{N}](i,q)\cdot[\mathbf{M}](q,j)), \tag{16}$$

where $\delta(i,j)$ is the Kronecker function which has a value of one if i and j are equal, and zero otherwise. The quantity q is an integer representing all river reaches, and $[\mathbf{C_1}\cdot\mathbf{N}](i,q)$ is the element of $\mathbf{C_1}\cdot\mathbf{N}$ located at row i and column q. Accounting for a maximum of one downstream river reach per given reach and for related consequences on the properties of $\mathbf{N}$, $\mathbf{C_1}\cdot\mathbf{N}$, and their powers, equation (16) leads to (detailed derivation given in Appendix A):

$$\begin{aligned}&\forall(i,j)\in[1,m]^2,\\ &\quad\forall p\in\mathbb{N}^+,[\mathbf{N}^p](i,j)=0\iff[\mathbf{M}](i,j)=\delta(i,j)\\ &\quad\exists p\in\mathbb{N}^+,[\mathbf{N}^p](i,j)\neq0\iff\exists!r\in[1,m],[\mathbf{M}](i,j)=\delta(i,j)\\ &\qquad+C_{1_i}\cdot[\mathbf{N}](i,r)\cdot[\mathbf{M}](r,j),\end{aligned} \tag{17}$$

where $\mathbb{N}^+$ represents all positive integers, $\exists$ is the mathematical symbol for existence, ! is used for uniqueness, and $\iff$ is for material equivalence. The integer quantity r is therefore the unique river reach (if it exists) that is located directly upstream of reach i and that is also located p reaches downstream of river reach j. Physically, equation (17) means that if a river reach i is located directly downstream of a reach r which is itself somewhere downstream of a reach j, then one can compute $[\mathbf{M}](i,j)$ as a function of $[\mathbf{M}](r,j)$. Also, $[\mathbf{M}](i,j)$ is null if reach i is not anywhere downstream of reach j which makes sense because flow only goes downstream in the Muskingum method. One can therefore travel downstream through the river network to determine the elements of $\mathbf{M}$. Equation (17) is much more computationally manageable than equation (10) and is used herein to determine the elements of $\mathbf{M}$.

2.5. Magnitude of Elements in the Muskingum Operator

[22] By mathematical induction, equation (17) leads to

$$\forall(i,j)\in[1,m]^2,\quad [\mathbf{M}](i,j)=\delta(i,j)+C_{1_i}\prod_{l\in\Omega(i,j)}C_{1_l}, \tag{18}$$

where $\Omega(i,j)$ represents the ensemble of river reaches l that are both strictly upstream of river reach i and strictly downstream of river reach j. The computation of $\mathbf{M}$ from equation (18) involves the product of elements C_{1_j}, and their magnitude is therefore of importance. According to *Cunge* [1969], the Muskingum method is stable for $x_j\in[0,0.5]$, regardless of the strictly positive value of Δt, which associated to equation (2) allows to derive the following inequalities:

$$\begin{aligned}&\forall j\in[1,m],\forall k_j>0,\forall x_j\in[0,0.5]\\ &C_{1j}=\frac{\Delta t/2-k_j\cdot x_j}{k_j\cdot(1-x_j)+\Delta t/2}=1-\frac{k_j}{k_j\cdot(1-x_j)+\Delta t/2}<1\\ &C_{1j}=\frac{\frac{\Delta t}{2}-k_j\cdot x_j}{k_j\cdot(1-x_j)+\Delta t/2}=-1+\frac{\Delta t+k_j\cdot(1-2x_j)}{k_j\cdot(1-x_j)+\Delta t/2}>-1,\end{aligned} \tag{19}$$

which can also be expressed as

$$\begin{aligned}&\forall j\in[1,m],\forall k_j>0,\forall x_j\in[0,0.5]\\ &-1<C_{1j}<1.\end{aligned} \tag{20}$$

[23] Therefore, being the product of elements C_{1_j}, the subdiagonal elements of $[\mathbf{M}](i,j)$ are all bounded by the interval $]-1,1[$, i.e., the strict interval between -1 and 1 that excludes both -1 and 1. The parameter C_{1_j} approaches unity for flow waves of infinite celerity, i.e., infinitesimally small values of k_j with regard to the modeling time step Δt, but such is usually avoided by picking a time step smaller than k_j. Additionally, each element $[\mathbf{M}](i,j)$ becomes increasingly closer to zero as a reach i downstream of a given reach j becomes increasingly distant from it.

2.6. Quantification of Upstream-to-Downstream Influence in the Muskingum Method

[24] By definition of the matrix-vector product, equation (11) can be expressed as

$$\forall(i,j)\in[1,m]^2,\quad Q_i(t+\Delta t)=[\mathbf{M}\cdot\mathbf{b}(t)](i)=\sum_{j=1}^{j=m}[\mathbf{M}](i,j)\cdot b_j(t). \tag{21}$$

[25] The vector $\mathbf{b}(t)$ is by definition a combination of the inflow $\mathbf{N}\cdot\mathbf{Q}(t)$ from upstream, of the inflow $\mathbf{Q}^{\mathbf{e}}(t)$ from outside of the river network, and of the outflow $\mathbf{Q}(t)$; all defined at the previous time step t. Therefore, equation (21) shows that the contribution from each reach j at the previous time t to the updated flow $Q_i(t+\Delta t)$ at reach i can be

seen in the product $[\mathbf{M}](i,j) \cdot b_j(t)$. Additionally, one can show (detailed derivation given in Appendix B) that the contribution from each reach j at the updated time $t + \Delta t$ to the updated flow $Q_i(t + \Delta t)$ at a different reach i can be seen in the product $[\mathbf{M}](i,j) \cdot Q_j(t + \Delta t)$. Given that the magnitude of the elements in the Muskingum operator decreases with distance between river reaches, one can therefore wonder how far does the update need be accounted for.

2.7. Radii of Influence

[26] Given ε the bound for the relative error when a real number is rounded to its closest floating-point number (i.e., the machine epsilon), the precision that can be expected from the floating-point addition $\oplus$ of two floating-point numbers y and z is [*Goldberg*, 1991]

$$\begin{aligned} &\forall (y,z) \in \mathbf{F}^2, \\ &\quad (y+z) \cdot (1-2\varepsilon) \leq y \oplus z \leq (y+z) \cdot (1+2\varepsilon), \end{aligned} \tag{22}$$

where $\mathbf{F}$ designates all floating-point numbers. The approximation of floating-point additions can also be expressed as

$$\begin{aligned} &\forall (y,z) \in \mathbf{F}^2, \\ &\quad |z| \leq \frac{\varepsilon}{2} \cdot |y| \iff y \oplus z = y. \end{aligned} \tag{23}$$

[27] Applying equation (23) to the contribution from each reach j to each reach i during the update step, one can conclude that if $|[\mathbf{M}](i,j) \cdot Q_j(t + \Delta t)|$ is always smaller than $0.5 \cdot \varepsilon \cdot |Q_i(t + \Delta t)|$, then from a floating-point arithmetic perspective, the river reach j has never an influence on the update of river reach i. Therefore, the river reaches i and j could be computed independently, and such would be true for both equations (11) and (4). Since the elements $[\mathbf{M}](i,j)$ become increasingly closer to zero as the distance between reaches i and j increases, the independence of reaches i and j is bound to happen starting at a large-enough distance which will be here referred to as the radius of influence. A radius of downstream influence R_j^{down} can be defined as

$$\begin{aligned} &\forall j \in [1,m], \forall t \\ &\quad \exists r \in \mathbb{N}^+ \cap \left[0, R_j^{\text{down}}\right], [\mathbf{N}^r](i,j) = 1 \iff \forall i \in [1,m], \\ &\quad |[\mathbf{M}](i,j) \cdot Q_j(t + \Delta t)| \geq \frac{\varepsilon}{2} \cdot |Q_i(t + \Delta t)|. \end{aligned} \tag{24}$$

[28] The radius of downstream influence R_j^{down} is defined here as an integer for each river reach, but one could also think of other definitions where R_j^{down} varies not only in space but also in time. Physically, this means that the influence that any river reach further away than R_j^{down} downstream from reach j is smaller than can be accounted for by floating-point arithmetic. Similarly, one can define a radius of upstream influence:

$$\begin{aligned} &\forall i \in [1,m], \forall t \\ &\quad \exists r \in \mathbb{N}^+ \cap [0, R_i^{\text{up}}], [\mathbf{N}^r](i,j) = 1 \iff \forall j \in [1,m], \\ &\quad |[\mathbf{M}](i,j) \cdot Q_j(t + \Delta t)| \geq \frac{\varepsilon}{2} \cdot |Q_i(t + \Delta t)|. \end{aligned} \tag{25}$$

[29] R_j^{down} therefore allows looking downstream at what river reaches are influenced *by* a given reach, and R_i^{up} allows looking upstream at what river reaches have influence *on* a given reach. One should note that the values of both $[\mathbf{M}](i,j)$ and $Q_j(t + \Delta t)$ need be accounted for in computing the radii of influence since instantaneous values of $Q_j(t + \Delta t)$ can be of greater magnitude than that of $Q_i(t + \Delta t)$—e.g., when a flow wave is propagating—even if such is not true for their respective temporal averages.

2.8. Implications for Parallel Computing

[30] Most studies looking at parallel speedup of river flow computations using a simple upstream-to-downstream river routing scheme of type 2 as done here assume that the update of any river reach cannot be done prior to the update of all of its upstream river reaches [e.g., *Li et al.*, 2011]. Under such an assumption, the maximum theoretical speedup $S_1(N)$ that can be reached when assigning N subbasins to N different computing cores while keeping loads balanced is

$$S_1(N) = \frac{N}{N_b}, \tag{26}$$

where N_b is the largest number of subbasins (i.e., computing cores) crossed—out of all possible paths going from upstream to downstream—in the entire river network.

[31] Based on the work presented in this paper, using subbasins that are longer than the radii of influence and accounting for the corresponding independence of river reaches, one can therefore compute each subbasin entirely in two iterations and guarantee that the downstream-most elements are computed exactly in the first iteration while others are still inexact, and be assured that a second iteration is sufficient to update all other elements of each subbasin. Such an approach can allow reaching the following maximum theoretical speedup:

$$S_2(N) = \frac{N}{2}. \tag{27}$$

[32] One should note here that equation (27) assumes that each core computes all its corresponding elements twice—as done in this study—which is not necessary, and therefore, $S_2(N)$ could be higher. Alternatively, one could update all elements in a first iteration, again knowing that only the downstream-most river reaches are exactly computed, and limit the second iteration to those reaches that were inexact, i.e., those located within a distance of $R^{\text{down}} - 1$ downstream of the connections between subbasins. The integer quantity R^{down} can be picked, for example, as the maximum value of all R_j^{down}. In doing so, one could expect the following ideal maximum theoretical speedup:

$$S_3(N) = \frac{N}{1 + (R^{down} - 1) \cdot N/m}. \tag{28}$$

[33] In the case where the radius of influence or the number of subbasins are much smaller than the total number of river reaches in the entire domain, i.e., $\left(R^{\text{down}} - 1\right) \cdot N/m << 1$, S_3 becomes close to ideal. As a consequence, if the modeling domain allows for more than

two consecutive subbasins that are large enough with regard to the radii of influence, the maximum theoretical speedup of equations (27) and (28) can be much higher than that of equation (26).

3. Theoretical Example

[34] Figure 2 shows an example river network made out of 12 river reaches. The corresponding network and linear system matrices are given by equations (29) and (30), respectively, in sparse format (only the nonzero values are shown here and saved by the computer program):

$$\mathbf{N} = \begin{bmatrix} & & & & & & & & & & & \\ 1 & 1 & & & & & & & & & & \\ & & 1 & & & & & & & & & \\ & & & 1 & 1 & & & & & & & \\ & & & & & 1 & & & & & & \\ & & & & & & 1 & 1 & & & & \\ & & & & & & & & 1 & & & \\ & & & & & & & & & 1 & 1 & \end{bmatrix} \tag{29}$$

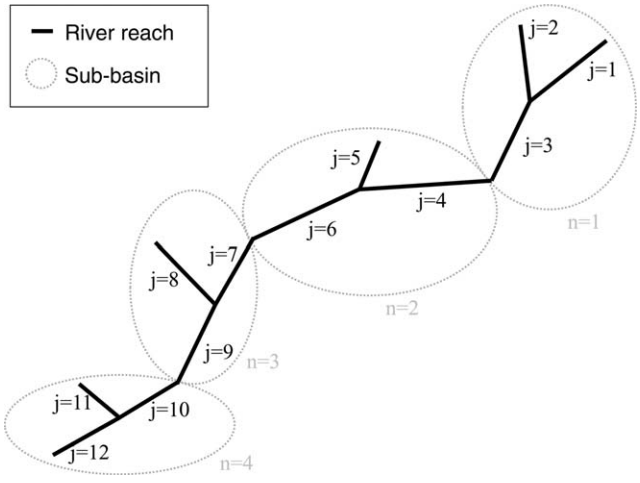

Figure 2. A river network made out of 12 river reaches and 7 nodes split into four subbasins and used as a theoretical example.

$$\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N} = \begin{bmatrix} 1 & & & & & & & & & & & \\ & 1 & & & & & & & & & & \\ -C_{1_3} & -C_{1_3} & 1 & & & & & & & & & \\ & & -C_{1_4} & 1 & & & & & & & & \\ & & & & 1 & & & & & & & \\ & & & & -C_{1_6} & -C_{1_6} & 1 & & & & & \\ & & & & & & -C_{1_7} & 1 & & & & \\ & & & & & & & -C_{1_9} & -C_{1_9} & 1 & & \\ & & & & & & & & & -C_{1_{10}} & 1 & \\ & & & & & & & & & & 1 & \\ & & & & & & & & & -C_{1_{12}} & -C_{1_{12}} & 1 \end{bmatrix}. \tag{30}$$

[35] The Muskingum operator being much denser, only the first column is shown here, for clarity:

$$[\mathbf{M}](i,1) = \begin{bmatrix} 1 \\ \\ C_{1_3} \\ C_{1_3} \cdot C_{1_4} \\ \\ C_{1_3} \cdot C_{1_4} \cdot C_{1_6} \\ C_{1_3} \cdot C_{1_4} \cdot C_{1_6} \cdot C_{1_7} \\ \\ C_{1_3} \cdot C_{1_4} \cdot C_{1_6} \cdot C_{1_7} \cdot C_{1_9} \\ C_{1_3} \cdot C_{1_4} \cdot C_{1_6} \cdot C_{1_7} \cdot C_{1_9} \cdot C_{1_{10}} \\ \\ C_{1_3} \cdot C_{1_4} \cdot C_{1_6} \cdot C_{1_7} \cdot C_{1_9} \cdot C_{1_{10}} \cdot C_{1_{12}} \end{bmatrix}. \tag{31}$$

[36] One can see that the last element of the first column is the product of seven elements C_{1_j} and is likely to be of small magnitude given the bounds of C_{1_j} themselves.

[37] Applying the traditional Muskingum method to river reach $j = 7$ and substituting into the same equation applied to $j = 9$ lead to

$$\begin{aligned} Q_9(t+\Delta t) &= b_9(t) + C_{1_9} \cdot Q_8(t+\Delta t) + C_{1_9} \cdot Q_7(t+\Delta t) \\ &= b_9(t) + C_{1_9} \cdot Q_8(t+\Delta t) + C_{1_9} \\ &\quad \cdot [b_7(t) + C_{1_7} \cdot Q_6(t+\Delta t)] \\ &= b_9(t) + C_{1_9} \cdot b_7(t) + C_{1_9} \cdot Q_8(t+\Delta t) + C_{1_9} \cdot C_{1_7} \\ &\quad \cdot Q_6(t+\Delta t). \end{aligned} \tag{32}$$

[38] One should note here that if $Q_6(t+\Delta t)$ were to be substituted by its value as a function of $b_6(t)$, the same

multiplying scalar $[\mathbf{M}](9,6) = [(\mathbf{C_1} \cdot \mathbf{N})^2](9,6) = C_{1_9} \cdot C_{1_7}$ would appear before $b_6(t)$ as it does before $Q_6(t + \Delta t)$ in equation (32); as was previously demonstrated in section 2.6. Assuming a radius of downstream influence of two for all river reaches, any given river reach only has accountable influence on the updated flow rates for itself and for the immediate downstream reach. Consequently, the value of $C_{1_9} \cdot C_{1_7} \cdot Q_6(t + \Delta t)$ can always be neglected with regard to the value of $Q_9(t + \Delta t)$. Therefore, one could compute an estimate $\tilde{Q}_9$ of Q_9 using

$$\tilde{Q}_9(t + \Delta t) = b_9(t) + C_{1_9} \cdot b_7(t) + C_{1_9} \cdot Q_8(t + \Delta t). \quad (33)$$

[39] Such an estimate would actually be exactly correct from a floating-point arithmetic perspective, and therefore, one would not need to wait for $Q_6(t + \Delta t)$ to be computed in order to compute $Q_9(t + \Delta t)$. Hence, if subbasins are constructed larger than the radius of downstream influence, then the downstream-most element—at the least—can be computed exactly without accounting for the inflow from upstream subbasins, and new computing algorithms can be developed.

[40] Figure 3 shows three ways of solving the Muskingum method if splitting the entire river network in the four subbasins of same size that are shown in Figure 2, and Figure 4 shows the corresponding time sequences. In both of Figures 3 and 4, method (a) corresponds to the traditional Muskingum method, method (b) corresponds to an iterative linear system solver, and method (c) is raised as a potential new way to solve the Muskingum method.

[41] The maximum theoretical speedup for the traditional method of Figures 3a and 4a is $S_1(4) = 4/4 = 1.00$, i.e., no speedup.

[42] Another way of solving the Muskingum method is to start computing everywhere at the same time and to allow for two iterations as shown in Figures 3b and 4b. Using core 3 as an example, a first iteration gives

$$\begin{cases} \tilde{Q}_7(t + \Delta t) = b_7(t) + 0 \\ \tilde{Q}_8(t + \Delta t) = b_8(t) \\ \tilde{Q}_9(t + \Delta t) = b_9(t) + C_{1_9} \cdot \tilde{Q}_7(t + \Delta t) + C_{1_9} \cdot \tilde{Q}_8(t + \Delta t) \end{cases}, \quad (34)$$

where $\tilde{Q}_j(t + \Delta t)$ is used for a temporary update of $Q_j(t + \Delta t)$. Notice that $Q_6(t + \Delta t) = 0$ is used in equation (34) for the computation of $Q_7(t + \Delta t)$ because its real value has not yet been computed by core 1. A second iteration gives

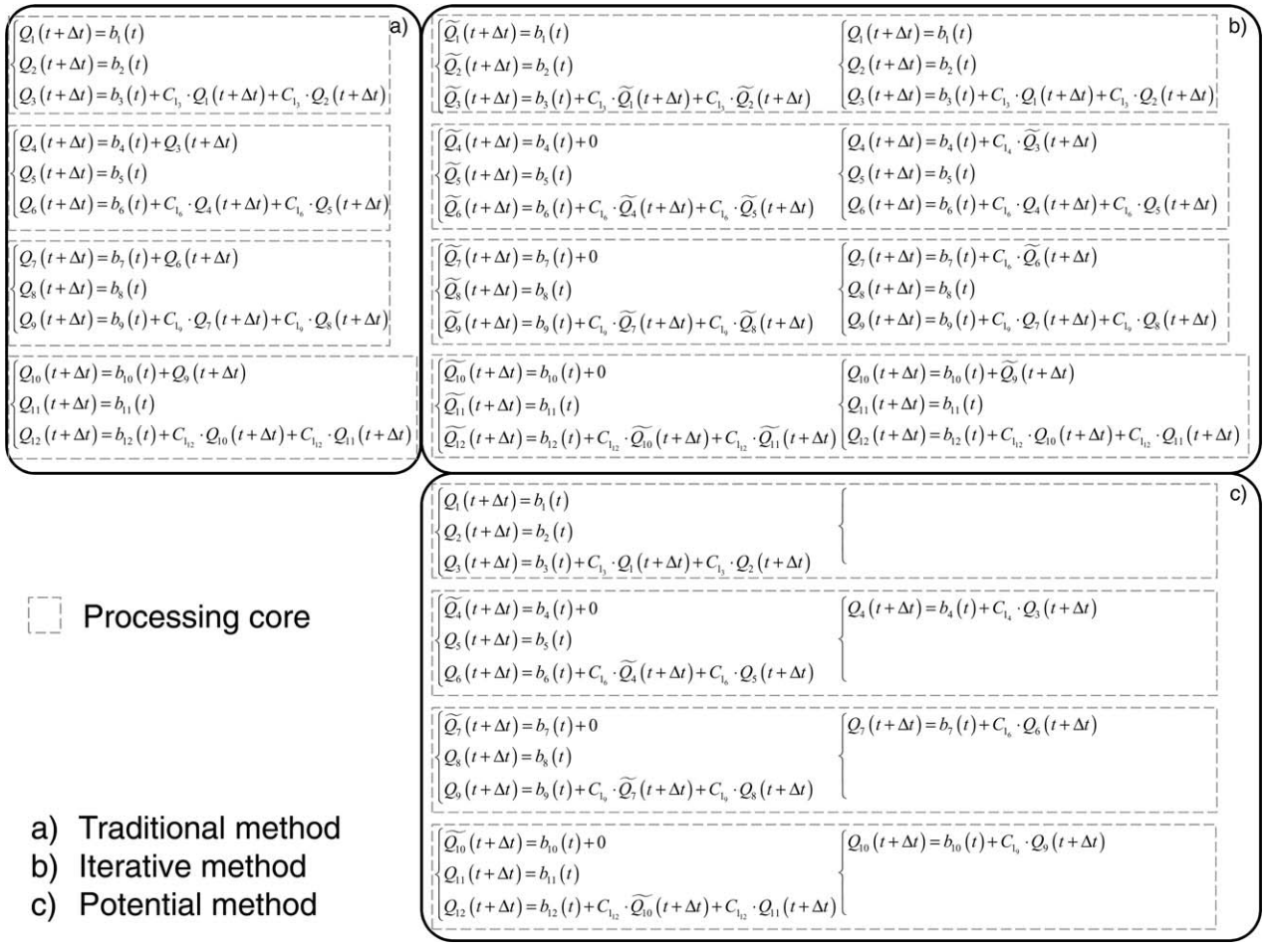

Figure 3. Three ways of solving the Muskingum method. Each processing core addresses the corresponding subbasin in the river network of Figure 2. Methods (a) and (b) are used in this paper, and (c) is not.

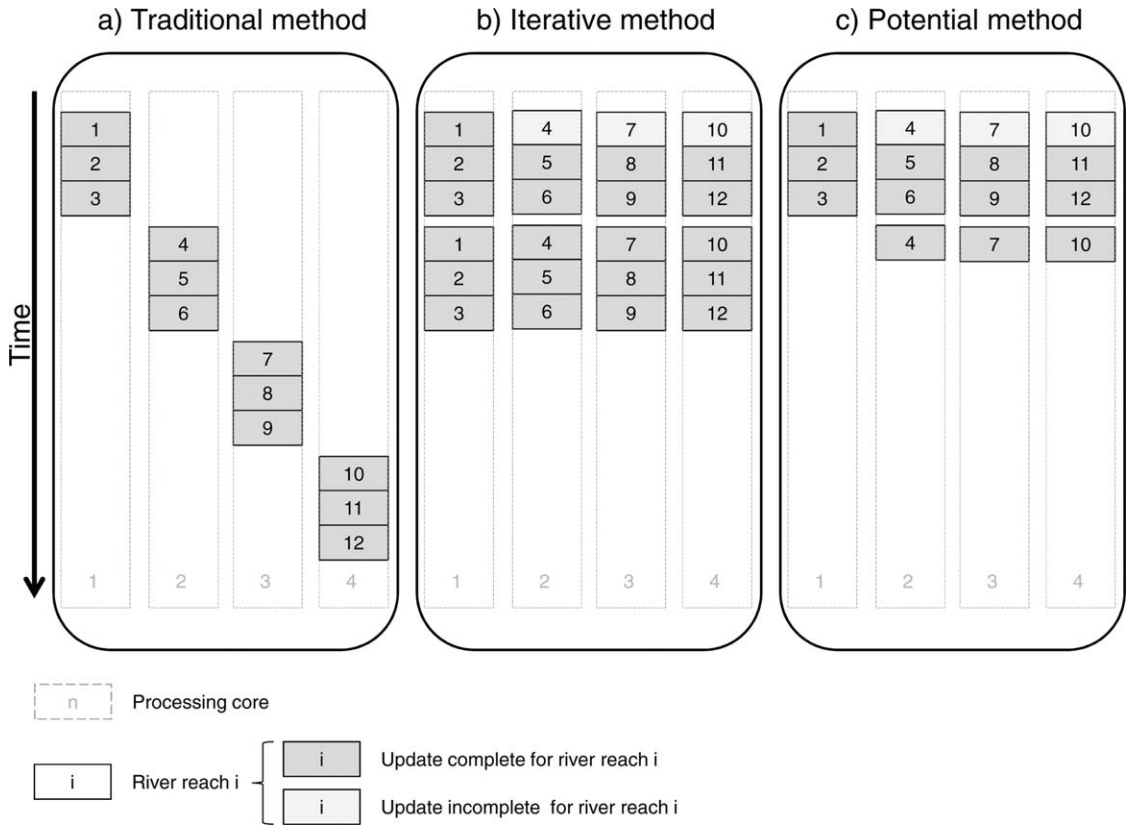

Figure 4. Time sequence corresponding to the three ways of solving the Muskingum method presented in Figure 3. Each processing core addresses the corresponding subbasin in the river network of Figure 2. Methods (a) and (b) are used in this paper, and (c) is not.

$$\begin{cases} Q_7(t+\Delta t) = b_7(t) + C_{1_6} \cdot \tilde{Q}_6(t+\Delta t) \\ Q_8(t+\Delta t) = b_8(t) \\ Q_9(t+\Delta t) = b_9(t) + C_{1_9} \cdot Q_7(t+\Delta t) + C_{1_9} \cdot Q_8(t+\Delta t) \end{cases} . \tag{35}$$

[43] The values of $\tilde{Q}_9(t+\Delta t)$ and of $Q_9(t+\Delta t)$ are exactly the same in equations (34) and (35), because with a radius of downstream influence of two, $C_{1_9} \cdot C_{1_7} \cdot Q_6(t+\Delta t)$ is too small to be accounted for by computer operations. This method allows obtaining a maximum theoretical speedup of $S_2(4) = 4/2 = 2.00$.

[44] Finally, the method of Figures 3c and 4c could be used to update everywhere in a first iteration knowing that the upstream-most elements in each subbasin are inexact, and finish the update of these upstream elements in a second iteration which would lead to $S_3(4) = 4/(1 + (2-1) \cdot 4/12) = 3.00$.

[45] Accounting for the relative upstream-to-downstream influence between river reaches when solving the Muskingum method can therefore double or even triple the maximum theoretical speedup for this particular example. The value of two picked here for the radius of downstream influence is smaller than the values obtained in the practical application presented later, for clarity, but the same concept applies.

4. Practical Application

[46] The Upper Mississippi River Basin is used here as a test bed for the theoretical approach presented in section 2. The enhanced version of the National Hydrography Dataset (NHDPlus) [*U.S. Environmental Protection Agency and U.S. Geological Survey*, 2010] provides all river reaches of this large network as well as the corresponding contributing catchments. Runoff data from the second phase of the North American Land Data Assimilation System (NLDAS2) [*Xia et al*., 2012a, 2012b] are used here to estimate the inflow of water from surface and subsurface into the river network. The computations of river flow using the Muskingum method are performed with the Routing Application for Parallel computatIon of Discharge (RAPID) [*David et al*., 2011b], and Muskingum parameters are optimized based on gauge observations from the U.S. Geological Survey (USGS). Our analysis focuses on days from 1 January to 31 December 2004, and all tools used are briefly described in this section.

4.1. NHDPlus Description of Upper Mississippi River Basin

[47] The Upper Mississippi River Basin is available as NHDPlus Region 07 (Figure 1) and has 191,646 river reaches of which 182,240 have known network connectivity and are used in this study. These 182,240 river reaches vary in size from 0.001 to 27.353 km (median: 1.64 km, mean: 1.87 km, standard deviation 1.64 km). The total basin area of 491,777.837 km^2 is divided into 180,787 contributing catchments, each corresponding to a unique river reach with known network connectivity. The smallest of the river reaches with known network connectivity do not

have an associated contributing catchment. The contributing catchments vary in size from 0.001 to 318.91 km^2 (median: 1.58 km^2, mean: 2.72 km^2, standard deviation 4.63 km^2).

4.2. Inflow of Surface and Subsurface Water Into NHDPlus River Reaches

[48] NLDAS2 [*Xia et al*., 2012a, 2012b] offers hourly land surface data from four land surface models from 1979 to current times and on a 1/8 degree grid. Out of the four NLDAS2 land models, only the Variable Infiltration Capacity (VIC) model originally developed by *Liang et al*. [1994], and now in its version 4.0.5 was used in this study. Three-hourly inflows of water into NHDPlus river reaches were estimated from the sum of VIC surface and subsurface runoff by means of the conceptual translation between the gridded NLDAS2 environment and the vector-based NHDPlus catchments that is described by *David et al*. [2013] and that uses catchment centroids.

4.3. Flow Observations on NHDPlus Rivers

[49] Flow observations are needed in this study for optimization of river routing parameters and for comparisons between computations and observations. Here we use the gauges of the USGS National Water Information System (NWIS) of which the exact location on the NHDPlus river network is available as part of the NHDPlus deployment. A total of 1277 NWIS gauges exist in the NHDPlus description of the Upper Mississippi River Basin but only 1251 are located on river reaches with known network connectivity. Out of these 1251 gauges, only 419 have full daily record from 1 January to 31 December 2004. Each of these 419 gauges is located on a unique NHDPlus river reach, except for four which are located on two river reaches (two stations per reach). The comparisons between computed and observed flow are done here using the flow at the outlet of reaches that are host to a gauge, and hence only the most downstream gauge was retained in case of duplicates. Therefore, daily data from 417 USGS gauges are used in this study. The selection and downloading of data were performed using HydroExcel (http://his.cuahsi.org/hydroexcel.html) and HydroGET (http://his.cuahsi.org/hydroget.html).

4.4. Routing Application for Parallel Computation of Discharge

[50] The RAPID [*David et al*., 2011b] is a river routing model. Given surface and groundwater inflow to rivers, this model can compute the flow and volume of water everywhere in river networks made out of many thousands of reaches. The design of RAPID allows it to be adapted to any river network, if given basic connectivity information as is the case in NHDPlus. Computations in RAPID can be performed using the traditional Muskingum method of equation (1) and accounting for surface/groundwater inflow or with the matrix-based Muskingum method of equation (3). RAPID also has an automated parameter estimation procedure that allows finding optimal model parameters based on available gauge measurements. This model uses the Fortran programming language along with the Portable, Extensible Toolkit for Scientific Computation (PETSc) [*Balay et al*., 1997, 2010, 2012] and the Toolkit for Advanced Optimization [*McInnes et al*., 2011] and can be run on personal computers, as well as on massively parallel supercomputers. The model code for RAPID as well as related documents and animations can be found at http://www.ucchm.org/david/rapid.htm. Calculations in RAPID are performed using double-precision floating points.

[51] As in *David et al*. [2011b, 2013], a river routing time step of $\Delta t = 900$ s is used. This time step should properly capture the flow dynamics at least for river reaches that are 0.9 km long or longer (i.e., more than two thirds of river reaches in this study) if flow wave celerities are $c = 1$ m/s or slower, as in this paper.

5. Results

5.1. Calibration of RAPID

[52] In this study, the optimization of the Muskingum parameters k_j (time) and x_j follows the simple approach of *David et al*. [2011b] using a squared-error cost function ϕ_1 computed using all 417 available gauges over 1 January to 30 December 2004. This procedure leads to the following values:

$$\begin{aligned} &\forall j \in [1, 182240] \\ &k_j = 0.3087 \cdot \frac{L_j}{c^0} \\ &x_j = 0.0652 \cdot 0.1, \end{aligned} \tag{36}$$

where L_j is the length of each river reach j, $c^0 = 1$ km/h is a constant flow wave celerity, and k_j is expressed in seconds. The value of the flow wave celerity $c = c^0/0.3087 = 0.90$ m/s obtained here for the Upper Mississippi River Basin is close to $c = 0.78$ m/s that was presented by *David et al*. [2013] for the Texas Gulf Coast Hydrologic Region; a domain of similar size but with smaller rivers. Figure 5 shows an example hydrograph obtained for the most downstream gauge of the Upper Mississippi River Basin. The more advanced optimization methods presented by *David et al*. [2011a, 2013] could be used here to improve the match between observed and computed hydrographs. However, such is beyond the goal of this paper, and the simple parameters of equation (36) will be used for the analysis in this study.

5.2. Computation of the Muskingum Operator and Quantification of the Upstream-to-Downstream Influence

[53] The values of the elements C_{1_j} for the Upper Mississippi River Basin are computed based on equations (2) and (36). Figure 6 shows the probability density function and the cumulative distribution function of C_{1_j} for the entire study area. As expected, the elements C_{1_j} are bounded by the interval $]-1, 1[$. In this particular case, no negative values are obtained for C_{1_j} because x_j is small.

[54] Equation (17) was then used to compute the Muskingum operator $\mathbf{M}$. One should note here that storing the values of the Muskingum operator in a file requires saving 182240^2 double-precision floating points, each necessitating 8 B, which represents 247 GB; i.e., the equivalent of approximately 60 years of three-hourly river flow data for the same domain. Allowing for a maximum of three potential upstream river reaches for each river reach as done in NHDPlus, the sparse linear system matrix $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ only

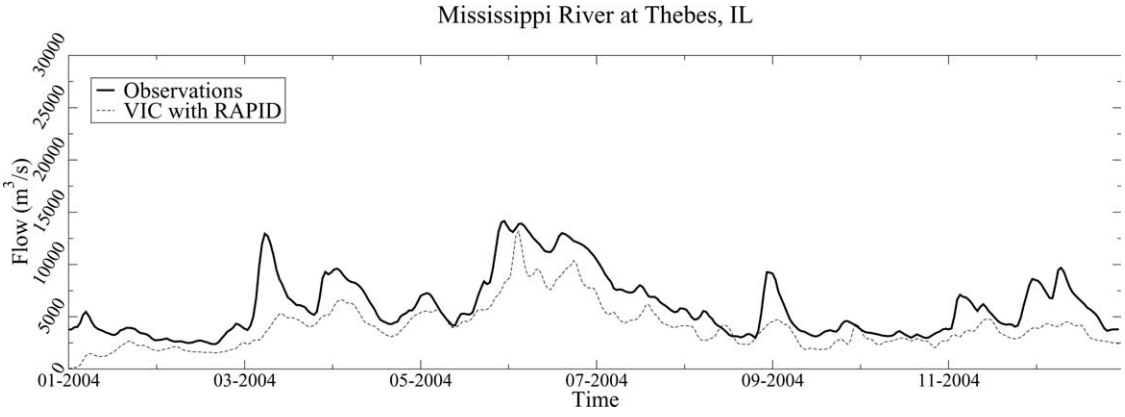

Figure 5. Comparison of observed and modeled daily hydrographs for the Mississippi River at Thebes, IL.

necessitates a maximum of $182240 \cdot (3+1)$ floats which represents only 5 MB. It is therefore much easier to use $\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}$ for river routing on computers than it is to use $\mathbf{M}$ not only because $\mathbf{M}$ is much denser and hence requires a higher number of computations, but also because of its associated higher memory needs. Because of the large size of $\mathbf{M}$, only the part corresponding to all 3916 river reaches located upstream of the Mississippi River at Brainerd, MN, is shown here for clarity of the graphics (Figure 7). As expected, the magnitude of $\mathbf{M}$ gets smaller as one moves away from the main diagonal, which confirms that the relative upstream-to-downstream influence decreases with the distance between river reaches.

5.3. Radii of Influence

[55] Figure 8 shows all river reaches located upstream of the Mississippi River at Brainerd, MN, as well as those river reaches that have accountable influence *on* and *from* the Mississippi River at Aitkin, MN. Equation (24) was used to select all river reaches for which the influence *from* Aitkin is large enough to be accounted for by a computer model, and 24 river reaches match this criterion; so $R^{\text{down}}_{\text{Aitkin}} = 24$. Equation (25) was used to select all the river reaches that have a large-enough influence *on* Aitkin, and a total of 472 river reaches match this other criterion but include many branches upstream of Aitkin and therefore provide an upper estimate for $R^{\text{up}}_{\text{Aitkin}}$. The number of river reaches for which floating-point arithmetic allows accounting for the relative influence among river reaches is therefore much smaller than the domain size in this study.

[56] Equation (24) was then used to compute the radius of downstream influence for the entire domain, and values are shown in Figure 9. Values of R^{down}_j range between 0 and 155 and have an average of 36 which is again much smaller than the domain size. The spatial repartition of R^{down}_j does not seem to particularly match any topological network structure, which can be explained by the dependence of $Q_j(t+\Delta t)$ on inflow of water from surface and subsurface. Similar values are found for the radii of influence when replacing $Q_j(t+\Delta t)$ by $b_j(t)$ in equations (24) and (25), although these results are not shown here.

[57] One should note that the computations of both radii of influence are made here using the worst-case scenario, i.e., the maximum absolute value of $Q_j(t+\Delta t)$ and the minimum absolute value of $Q_i(t+\Delta t)$ over the entire duration of study, both evaluated at the 15 min time step. The radii of influence reported here are therefore safe overestimates since the ratio of $Q_j(t+\Delta t)$ over $Q_i(t+\Delta t)$ is always smaller—at any time during the simulation—than if computed based on their respective maximum and

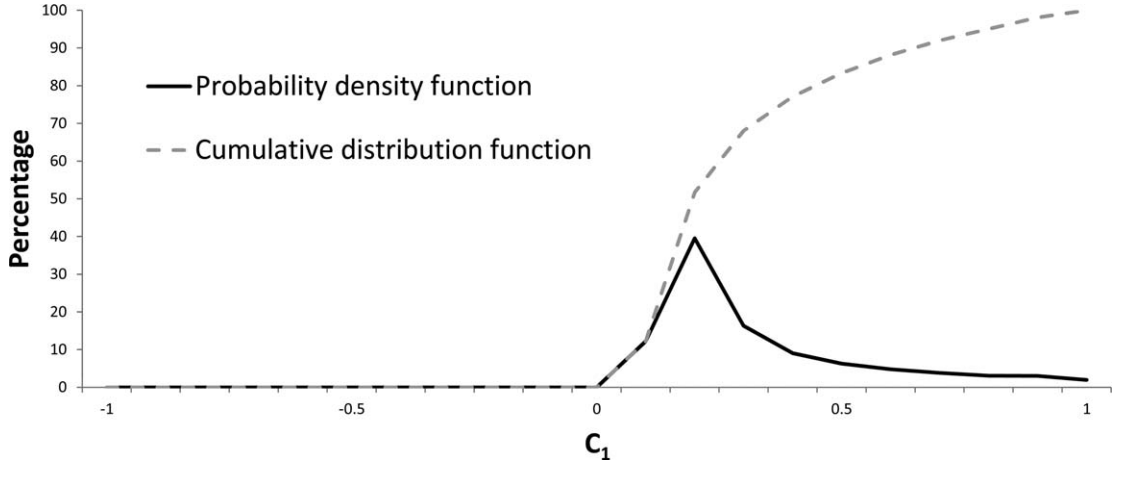

Figure 6. Probability density function and cumulative distribution function for all diagonal elements of $\mathbf{C_1}$.

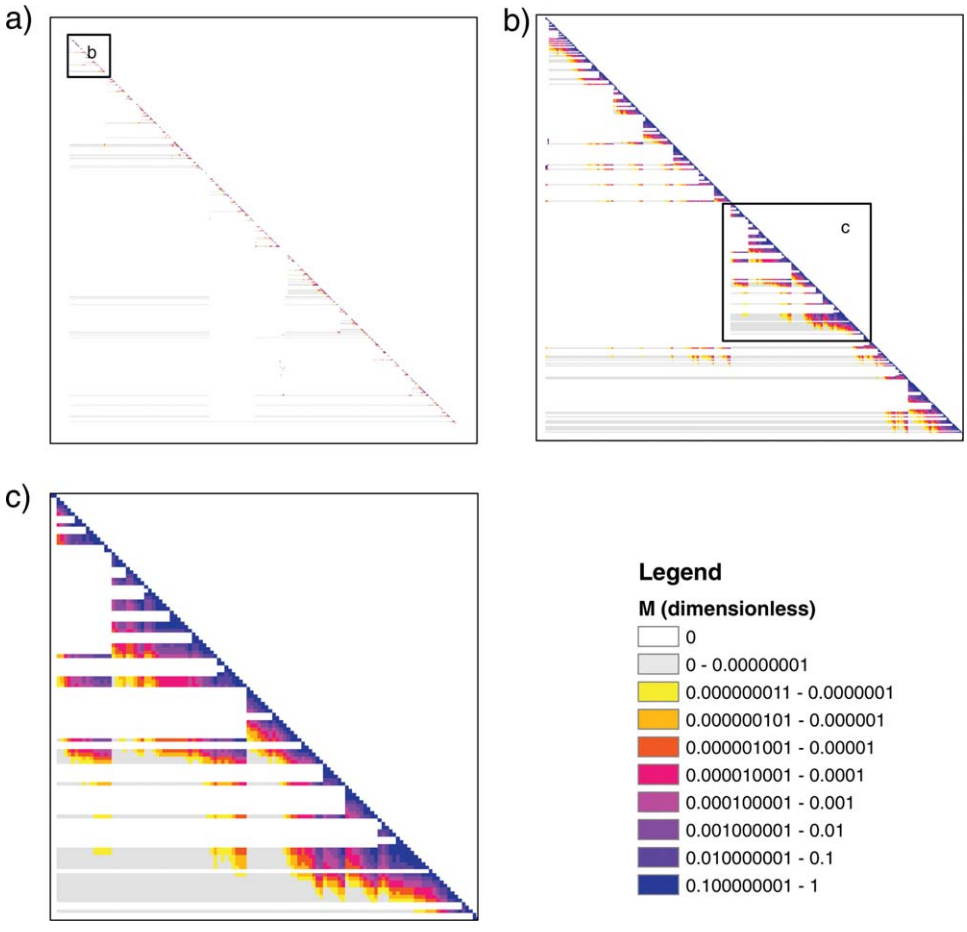

Figure 7. Shape of the Muskingum operator for all 3916 river reaches located upstream of the Mississippi River at Brainerd, MN, and shown in Figure 1.

minimum as done here. The value of $\varepsilon = 2^{-53} \approx 1.11 \times 10^{-16}$ corresponding to double-precision floating-point operations was used.

5.4. Speedup of Parallel Computations

[58] Similar to *David et al*. [2011b], RAPID was run with various computing methods all using the same input data and the same routing parameters; the corresponding computation times were compared among themselves (Figure 10). The traditional Muskingum method can be directly compared with a noniterative matrix-based Muskingum method and performed similarly; but the two methods can only be applied to a unique processing core when all river reaches of the domain are fully interconnected as is the case in the Upper Mississippi River Basin. However, an iterative matrix-based Muskingum method can be used on one or several processing cores by splitting the domain in subbasins and assigning each subbasin to a different processing core, as shown in Figure 11. The domain decomposition used here is the same as *David et al*. [2011b] and uses a native NHDPlus field informing on the relative upstream/downstream position of river reaches. Despite an initial overhead brought by the iterative matrix-based Muskingum method (Figure 10)—mainly due to the time required to compute an initial residual error in order to provide the linear system solver with a convergence criterion—the computation time quickly decreases with the number of processing cores before being limited by intercore communication and increased number of iterations at 32 cores.

[59] The experimental speedup between the iterative matrix-based Muskingum method applied on 16 cores and the traditional Muskingum method is $S(16) = 250/72 = 3.46$ which is higher than $S_1(16) = 16/13 = 1.23$ but also much lower than $S_2(16) = 16/2 = 8.00$ or than $S_3(16) = 16/(1 + (155 - 1)16/182240) = 15.79$. The real quantities $S_2(N)$ and $S_3(N)$ therefore give a new upper limit on what can be achieved for parallel computing of river flow with the Muskingum method depending on how the equations are solved. The computations of the iterative matrix-based Muskingum method converge in less than two iterations for all runs with 16 cores or less, so the domain decomposition used can be considered large enough with regard to the radii of influence. At 32 cores, an additional iteration is needed, suggesting that the domain decomposition is not any more appropriate. Detailed explanations on the differences between the methods used here can be found in *David et al*. [2011b].

5.5. Magnitude of Differences in Results Obtained Among Computing Methods

[60] *David et al*. [2011b] reported exact match—on a byte-to-byte basis—of all output files among computing

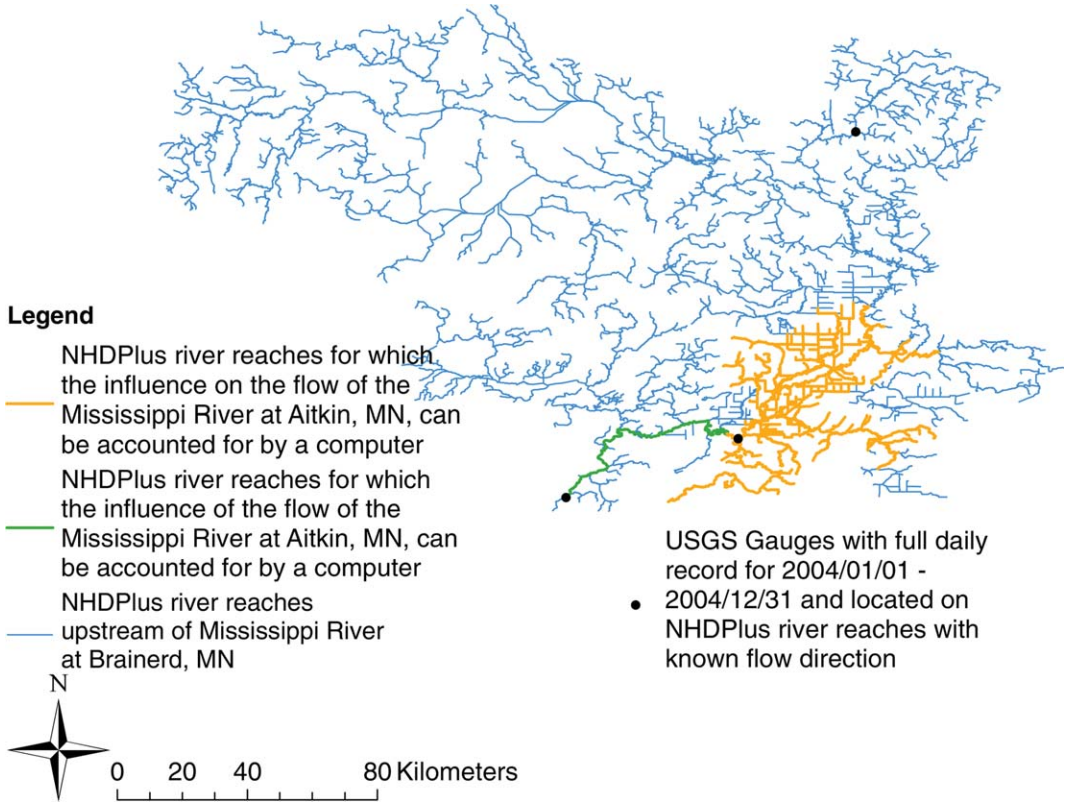

Figure 8. All 3916 river reaches located upstream of the Mississippi River at Brainerd, MN, as well as those river reaches that have accountable influence on and from the Mississippi River at Aitkin, MN.

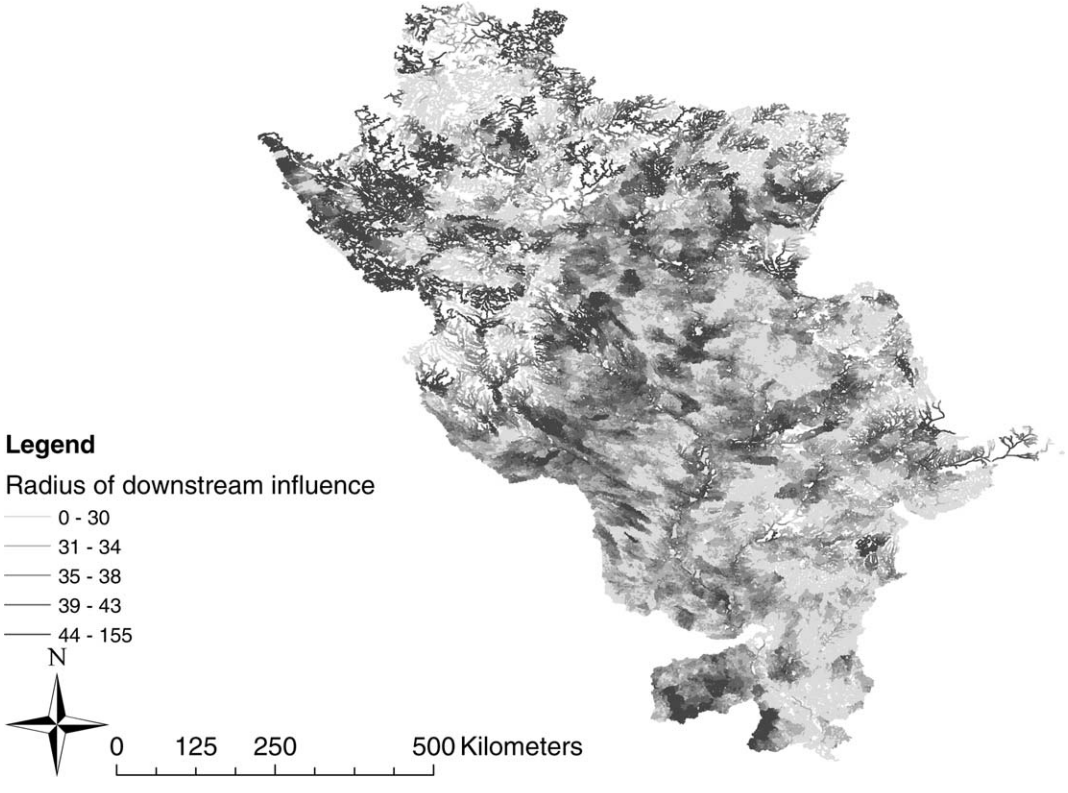

Figure 9. Radius of downstream influence.

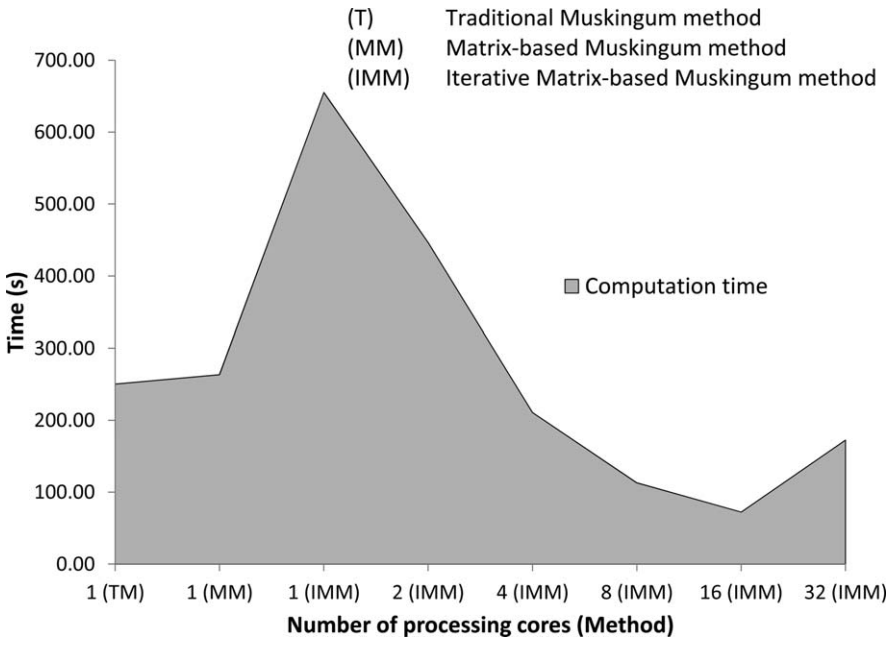

Figure 10. Computation time with various ways to solve the Muskingum method.

methods tested. However, despite computations being performed using double-precision floating-point operations, the outputs of *David et al*. [2011b] were saved using single-precision floating points. In this study, comparisons among computing methods are made using double-precision output files in order to provide a more in-depth description of differences. The various three-hourly outputs from RAPID corresponding to each computation method of Figure 10 were compared among themselves for each three-hourly average in order to quantify the differences among flow rates obtained in all methods. Results of these comparisons are presented in Table 1. The use of three-hourly outputs as opposed to computations at the 15 min routing time step is a simplification but one can argue that such high-resolution computations are generally not created by computer models anyway. With one unique computing

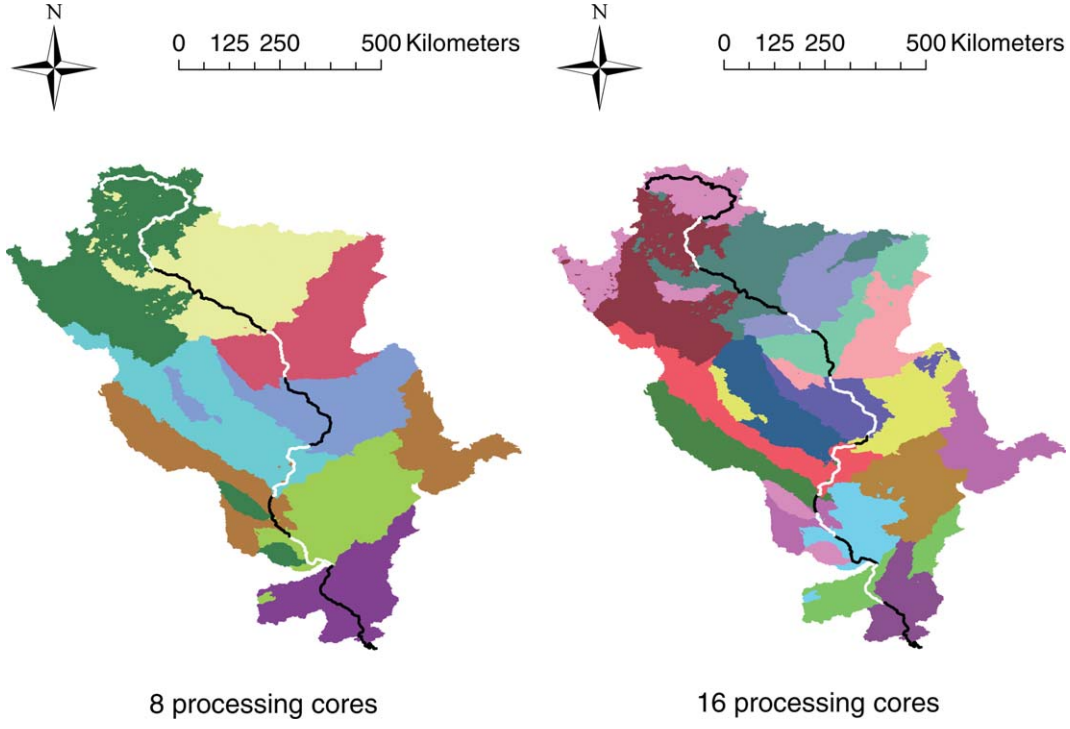

Figure 11. Domain decomposition for 8 and 16 processing cores, and longest distance path of the Upper Mississippi River Basin. Different colors correspond to different computing cores (adapted from *David et al*. [2011b]).

Table 1. Differences in Results Among Computing Methods Used

Method 1		Method 2		Maximum Value of Relative Difference in the Flow Rate Vector Using the 2-Norm Between Method 1 and Method 2 for any 3-Hourly Average During Entire Simulation	Maximum Value of Absolute Difference in Flow Rate Between Method 1 and Method 2 for Any River Reach and for Any 3-Hourly Average During Entire Simulation (m^3/s)
Method Name	Number of Processing Cores Used	Method Name	Number of Processing Cores Used		
Traditional Muskingum	1	Matrix-based Muskingum	1	1.04E−15	3.27E−11
Traditional Muskingum	1	Iterative matrix-based Muskingum	1	1.04E−15	3.27E−11
Traditional Muskingum	1	Iterative matrix-based Muskingum	2	1.14E−15	2.91E−11
Traditional Muskingum	1	Iterative matrix-based Muskingum	4	9.98E−16	2.36E−11
Traditional Muskingum	1	Iterative matrix-based Muskingum	8	1.05E−15	2.55E−11
Traditional Muskingum	1	Iterative matrix-based Muskingum	16	1.10E−15	2.73E−11
Traditional Muskingum	1	Iterative matrix-based Muskingum	32	9.58E−16	2.55E−11
Matrix-based Muskingum	1	Iterative matrix-based Muskingum	1	0.00E+00	0.00E+00
Matrix-based Muskingum	1	Iterative matrix-based Muskingum	2	1.21E−15	3.64E−11
Matrix-based Muskingum	1	Iterative matrix-based Muskingum	4	1.34E−15	3.27E−11
Matrix-based Muskingum	1	Iterative matrix-based Muskingum	8	1.15E−15	3.46E−11
Matrix-based Muskingum	1	Iterative matrix-based Muskingum	16	1.21E−15	4.00E−11
Matrix-based Muskingum	1	Iterative matrix-based Muskingum	32	1.08E−15	3.27E−11

core, differences exist between the computations of the traditional Muskingum method represented by equation (1) and those of the matrix-based Muskingum method represented by equation (4), and their relative magnitudes have a maximum of 1.04×10^{-15}. Such differences are independent of an assumption concerning the radius of influence since a unique computing core is used. These dissimilarities can therefore be attributed to two different numerical ways of solving the same linear system. Also on one unique computing core, no differences exist between the matrix-based Muskingum method and the iterative matrix-based Muskingum method, as expected. Using multiple computing cores, differences in computations of the iterative matrix-based Muskingum method exist, and their relative magnitudes have a maximum of 1.34×10^{-15}. Two iterations of the linear system solver being sufficient (for all simulations using between 2 and 16 processing cores), the corresponding subbasins can be considered large enough with regard to the radius of influence, and such differences are again independent of assumptions based on the radius of influence. These differences can therefore also be attributed to diverse numerical ways of solving the same linear system when using multiple processing cores. The reader is here again referred to *David et al*. [2011b] for further explanations on the differences between the computing methods used here. In any case, comparisons among all methods tested show that the maximum relative difference for any three-hourly average is $\max_{\forall t}(\|\Delta\mathbf{Q}\|_2/\|\mathbf{Q}\|_2) = 1.34 \times 10^{-15}$, and the maximum absolute differences for any three-hourly average and any river reach is $\max_{\forall t,\forall j}([\Delta\mathbf{Q}](j)) = 4.00 \times 10^{-11}\mathrm{m}^3/\mathrm{s}$.

[61] The precision that can be expected from the Muskingum method can be estimated by means of its condition number $\kappa(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N})$ following equation (13). In this study, the Scalable Library for Eigenvalue Problem Computations (SLEPc) [*Hernandez et al*., 2005] was used to estimate $\kappa(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N})$ with the Muskingum parameters k_j and x_j of equation (36). The results obtained by SLEPc are

$$\kappa_{\|\|_2}(\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}) = \frac{\sigma_{\max}}{\sigma_{\min}} \approx \frac{2.1156}{0.1900} \approx 11.31. \qquad (37)$$

[62] With relative errors in $\|\mathbf{b}\|_2$ and $\|\mathbf{I} - \mathbf{C_1} \cdot \mathbf{N}\|_2$ on the order of ε and using double-precision floating-point operations, one should therefore deem relative differences in flow rate computations on the order of $2 \cdot \kappa \cdot \varepsilon/(1 - \kappa \cdot \varepsilon) \approx 2 \times 11.31 \times 1.11 \times 10^{-16}/(1 - 11.31 \times 1.11 \times 10^{-16}) \approx 2.52 \times 10^{-15}$ as acceptable in this study. The maximum relative difference observed among all experiments performed in this study (see Table 1) can therefore be deemed acceptable. Additionally, the maximum absolute differences among flow rates at any river reach and any three-hourly average are on the order of $10^{-11}\mathrm{m}^3/\mathrm{s}$ and are hence much smaller than can be expected from the precision of river flow observations or attained by current modeling frameworks.

6. Discussion

[63] The study presented in this paper was motivated by results presented by *David et al*. [2011b] for which the parallel speedup was much higher than predicted by existing theory, suggesting that the speedup theory for parallel computing of river flow should be revised. Through the inversion of a large matrix, we are able to quantify the worst-case (largest) relative contribution of a given river reach on the updated flow rate of any other reach between two consecutive time steps of the Muskingum method. We show that such contribution is smaller than can be accounted for by floating-point arithmetic if the reaches are far-enough away from each other. Therefore, even if they are mathematically related—albeit by an infinitesimally small amount—these river reaches become independent from a computer modeling perspective. The minimum distance for which the relative influence between reaches becomes insignificant—the radius of influence—is defined and quantified here. It is important to note that we are *not* suggesting that river reaches located further away than the radius of downstream influence from a given river reach do not ever feel the influence of this given reach. Given enough time, water will flow from a given river reach to any of its downstream reaches; unless it is removed from the river network by evaporation, infiltration, or human withdrawals which

are beyond the purpose of the modeling addressed in this study. However, between two consecutive time steps, a given river reach has no accountable influence on far downstream elements in a floating-point environment. Therefore, in effect, flow waves are not fast enough to travel through an entire modeling domain within one time step if the domain is large enough. Such was not previously considered and explains why parallel speedup can be attained that is much higher than previously predicted. To our knowledge, the new upper limits for parallel speedup presented here have not been shown earlier. In order to achieve such speedup, one has to make sure that the subbasins assigned to different computing cores are large enough with regard to the radii of influence, and it would therefore be interesting to study network decomposition techniques such as those of *Veitzer and Gupta* [2001] or *Li et al*. [2011] but including a limitations of subbasin sizes based on the radii of influence. However, the computations of radii of influence presented here have challenges because these radii depend on the model time step, model parameters (themselves related to spatial/temporal resolution and to flow wave celerities which partly depend on landscape geometry) but also on the magnitude of water inflow into each river reach. Additionally, such computations can become very demanding as larger domains are addressed.

[64] Lastly, the parallel performance shown here is much lower than the new estimate of maximum parallel speedup—despite large-enough subbasins for almost all experiments—which suggests that other limitations currently exist. Such limitations deserve further investigations which could focus on current parallel computing technology, programming methods used, etc.

7. Conclusions

[65] Many river routing methods use a numerical scheme in which updating the flow rate for a given river reach depends on the updated flow rate of the reaches that are located directly upstream. By construction, such numerical schemes generate constraints on the ordering of computations and have therefore traditionally been solved by using an upstream-to-downstream approach. The maximum speedup that such an approach can attain in a parallel computing environment has previously been estimated and is based on the intuitive assumption that one has to wait for the update of all upstream river reaches to be performed prior to updating a given reach. The work presented here uses one of these numerical schemes—the Muskingum method—and quantifies the relative influence among river reaches during the update of flow rates between two consecutive time steps. This influence can be quantified exactly using linear algebra and is shown to be null from a floating-point arithmetic perspective, given that two river reaches are located far-enough away from each other with regard to a distance referred to here as the radius of influence. This rather counterintuitive finding can be explained because the relative upstream-to-downstream influence becomes increasingly smaller with distance, so small that it becomes insignificant in the addition performed by a computer. Physically, this means that flow waves are not fast enough to cross a large modeling domain within one unique time step. Therefore, one does not really have to wait for all upstream elements to be updated prior to updating a given river reach. Based on this finding a new maximum theoretical speedup for parallel computation of river flow is presented. The application of the proposed theoretical framework to the 182,240 river reaches of the Upper Mississippi River Basin over the year 2004 at a 15 min routing time step allows the estimation of the radius of influence on the order of 150 computing elements (river reaches), i.e., about three orders of magnitude smaller than the domain size which suggests large potential gains in computing times. The value found here for the radius of influence depends on the model time step, the spatial resolution of the river network, the local flow wave celerities and corresponding model parameters, and the magnitude of water inflow in our study domain, but the theoretical approach presented can be adapted to other applications. Also, despite the use of a domain decomposition where subbasins assigned to different computing cores are large enough when compared to the radii of influence, the speedup of the application presented remains far from the new upper limit developed in this study. The reasons for this less-than-optimal speedup remain to be determined, but we show an experimental speedup that is already much higher than what was previously considered an upper limit. As we address river modeling experiments of ever-increasing computing sizes, increases in the upper limits of parallel speedup such as the one presented here are likely to have valuable impacts. Finally, the work presented in this paper could be adapted to other river routing methods in which the updated flow rate depends on the prior update of upstream (and/or downstream) elements, and similar results are to be expected.

Appendix A

[66] A few properties of $(\mathbf{C_1} \cdot \mathbf{N})^p$ are used in this paper that allow simplifying the computation of $\mathbf{M}$ and are presented here. With regard to the relative sparseness of $\mathbf{N}^p$ and $(\mathbf{C_1} \cdot \mathbf{N})^p$,

$$\begin{aligned} &\forall (i,j) \in [1,m]^2, \forall p \in \mathbb{N}^+, \\ &\quad [\mathbf{N}^p](i,j) = 0 \Rightarrow [(\mathbf{C_1} \cdot \mathbf{N})^p](i,j) = 0, \end{aligned} \tag{A1}$$

where $[\mathbf{N}^p](i,j)$ is the element located at row i and column j of $\mathbf{N}^p$, and a similar notation is used for $(\mathbf{C_1} \cdot \mathbf{N})^p$. Equation (A1) is a direct consequence of $\mathbf{C_1}$ being a diagonal matrix and means that if an element of $\mathbf{N}^p$ is null, the corresponding element of $(\mathbf{C_1} \cdot \mathbf{N})^p$ is also null. Equation (A1) can also be used to prove that if $\mathbf{N}$ is a strictly lower triangular matrix, the product $\mathbf{C_1} \cdot \mathbf{N}$ is also strictly lower triangular.

[67] If each river reach is only allowed to flow into a unique downstream river reach, the sparseness of $\mathbf{N}$ is also limited by

$$\begin{aligned} &\forall j \in [1,m], \\ &\quad \exists i \in [1,m], [\mathbf{N}](i,j) \neq 0 \Rightarrow \exists! i \in [1,m], [\mathbf{N}](i,j) \neq 0. \end{aligned} \tag{A2}$$

[68] Associating the definition of the matrix product $\mathbf{N}^p \cdot \mathbf{N}$ to equation (A2) leads to

$$\begin{aligned} &\forall j \in [1,m], \forall p \in \mathbb{N}^+, \\ &\quad \exists i \in [1,m], \left[\mathbf{N}^{p+1}\right](i,j) \neq 0 \Rightarrow \exists! q \in [1,m], \\ &\quad \left[\mathbf{N}^{p+1}\right](i,j) = [\mathbf{N}^p](i,q) \cdot [\mathbf{N}](q,j). \end{aligned} \tag{A3}$$

[69] By mathematical induction, one can use equation (A3) to prove that

$$\begin{aligned}&\forall j \in [1,m], \forall p \in \mathbb{N}^+,\\ &\quad \exists i \in [1,m], [\mathbf{N}^p](i,j) \neq 0 \Rightarrow \exists! i \in [1,m], [\mathbf{N}^p](i,j) \neq 0.\end{aligned} \tag{A4}$$

[70] Therefore, if it exists, a nonnull element in a column of $\mathbf{N}^p$ is unique. In other words, the potential river reach i in which reach j flows into after p routing time steps is unique.

[71] From the definition of the matrix product $\mathbf{N} \cdot \mathbf{N}^p$, a direct consequence of equation (A4) is

$$\begin{aligned}&\forall j \in [1,m], \forall p \in \mathbb{N}^+,\\ &\quad \exists i \in [1,m], [\mathbf{N}^p](i,j) \neq 0 \Rightarrow \left[\mathbf{N}^{p+1}\right](i,j) = 0\end{aligned} \tag{A5}$$

[72] Using the definition of the matrix product $\mathbf{N}^{p-1} \cdot \mathbf{N}$ and equation (A5), a proof by contradiction allows to show that

$$\begin{aligned}&\forall j \in [1,m], \forall p \in \mathbb{N}^+,\\ &\quad \exists i \in [1,m], [\mathbf{N}^p](i,j) \neq 0 \Rightarrow \left[\mathbf{N}^{p-1}\right](i,j) = 0.\end{aligned} \tag{A6}$$

[73] By mathematical induction, one can use equations (A5) and (A6) to show that

$$\begin{aligned}&\forall j \in [1,m], \forall p \in \mathbb{N}^+,\\ &\quad \exists i \in [1,m], [\mathbf{N}^p](i,j) \neq 0 \Rightarrow \exists! p \in [1,m], [\mathbf{N}^p](i,j) \neq 0.\end{aligned} \tag{A7}$$

[74] Therefore, the number of routing time steps p necessary for a river reach j to flow into a potential reach i is unique. Combining equations (A4) and (A7) gives

$$\begin{aligned}&\forall j \in [1,m],\\ &\quad \exists (i,p) \in [1,m]^2, [\mathbf{N}^p](i,j) \neq 0 \Rightarrow \exists! (i,p) \in [1,m]^2, [\mathbf{N}^p](i,j) \neq 0.\end{aligned} \tag{A8}$$

[75] Despite the numerous steps involved in proving equation (A8), its physical meaning is intuitive: if a river reach j to flow into a reach i after p routing time steps, it does not flow into reach i after any other number of time steps, and it does not flow into any other reach in the p routing time steps. Combining equation (A1) with equation (A8) gives

$$\begin{aligned}&\forall j \in [1,m],\\ &\quad \exists (i,p) \in [1,m]^2, [(\mathbf{C_1} \cdot \mathbf{N})^p](i,j) \neq 0 \Rightarrow \exists! (i,p) \in [1,m]^2,\\ &\quad [(\mathbf{C_1} \cdot \mathbf{N})^p](i,j) \neq 0.\end{aligned} \tag{A9}$$

[76] Finally, using equation (A9) in equation (10),

$$\begin{aligned}&\forall j \in [1,m],\\ &\quad \exists i \in [1,m]\ \{j\}, [\mathbf{M}](i,j) \neq 0 \Rightarrow \exists! (i,p) \in [1,m]^2,\\ &\quad [\mathbf{M}](i,j) = [(\mathbf{C_1} \cdot \mathbf{N})^p](i,j) \neq 0\\ &\quad [\mathbf{M}](j,j) = 1.\end{aligned} \tag{A10}$$

[77] Equation (A10) allows to go from equations (16) to (17).

Appendix B

[78] Equation (4) can be reorganized as

$$\mathbf{Q}(t+\Delta t) = \mathbf{C_1} \cdot \mathbf{N} \cdot \mathbf{Q}(t+\Delta t) + \mathbf{b}(t). \tag{B1}$$

[79] Injecting equation (B1) into itself ρ times leads to

$$\mathbf{Q}(t+\Delta t) = (\mathbf{C_1} \cdot \mathbf{N})^\rho \cdot \mathbf{Q}(t+\Delta t) + \sum_{p=0}^{p=\rho-1} \{(\mathbf{C_1} \cdot \mathbf{N})^p \cdot \mathbf{b}(t)\}. \tag{B2}$$

[80] Equation (B2) allows virtually separating the contributions to the updated outflow $\mathbf{Q}(t+\Delta t)$ in two terms. The first term of the right-hand side corresponds to the contribution of those river reaches located further away than ρ reaches upstream and is expressed in the form of the updated outflow $\mathbf{Q}(t+\Delta t)$. The second term of the right-hand side corresponds to those river reaches located within a distance of ρ reaches upstream and is expressed in the form of the combination of previous flows $\mathbf{b}(t)$. One should note that starting at $\rho = m$, equation (B2) actually becomes equation (11), hence showing an alternate derivation for the inverse matrix. Applying equation (B2) at the river reach level leads to

$$\begin{aligned}Q_i(t+\Delta t) = &\sum_{j=1}^{j=m} \left\{[(\mathbf{C_1} \cdot \mathbf{N})^\rho](i,j) \cdot Q_j(t+\Delta t)\right\}\\ &+ \sum_{p=0}^{p=\rho-1} \left\{ \sum_{j=1}^{j=m} \left\{[(\mathbf{C_1} \cdot \mathbf{N})^p](i,j) \cdot b_j(t+\Delta t)\right\} \right\}.\end{aligned} \tag{B3}$$

[81] One can use equation (B3) to look at the contribution from reach j to reach i during the update time step. This contribution to $Q_i(t+\Delta t)$ comes separately from $Q_j(t+\Delta t)$ and from $b_j(t)$. However, for given river reaches i and j, equation (A9) states that if it exists, a nonnull $[(\mathbf{C_1} \cdot \mathbf{N})^\rho](i,j)$ exists only for a unique ρ being the number of river reaches separating i and j. Therefore, the contribution from reach j to reach i during the update time step cannot come from $Q_j(t+\Delta t)$ and from $b_j(t)$ at the same time. Applying equation (B3) to ρ and to $\rho+1$ allows demonstrating that the same scalar $[(\mathbf{C_1} \cdot \mathbf{N})^\rho](i,j)$ appears as a multiplier of $Q_j(t+\Delta t)$ and $b_j(t)$, respectively, when $i \neq j$. Remembering that $[(\mathbf{C_1} \cdot \mathbf{N})^\rho](i,i) = 0$ as shown in equation (A1), and accounting for equation (A10), the contribution from $Q_j(t+\Delta t)$ to $Q_i(t+\Delta t)$ is therefore $[\mathbf{M} - \mathbf{I}](i,j) \cdot Q_j(t+\Delta t)$. Hence, $[\mathbf{M}](i,j)$ can be used to study both the contribution from reach j to a different reach i based on previous flows using $b_j(t)$ and during the update time step using $Q_j(t+\Delta t)$.

[82] **Acknowledgments.** This work was supported by the Interdisciplinary Science Project NNX11AE42G of the U.S. National Aeronautics and Space Administration and by the University of California Office of the President Multicampus Research Programs and Initiatives which are both gratefully acknowledged. The practical application in this study was made possible using the following freely available data: river network information from the National Hydrography Dataset Plus, estimates of runoff from phase 2 of the North American Land Data Assimilation System, and gauge observations from the U.S. Geological Survey National Water Information

System. The authors also wish to thank the PETSc developers, especially Victor Eijkhout for recommending the investigation of the magnitude of subdiagonal elements of the linear system matrix, Jed Brown for input concerning the precision that can be expected from computations of linear system solvers in PETSc, and Matthew Knepley for mentioning the importance of the condition number. Gilbert Strang deserves appreciation because his Fall 2008 Lecture 14-MIT 18.085 shared freely through the Massachusetts Institute of Technology Open Course Ware was particularly helpful to the authors. Finally, the authors are thankful to the three anonymous reviewers, to the Associate Editor, and to Graham Sander, the Editor, for their valuable comments and suggestions that helped improve the original version of this manuscript.

References

Balay, S., W. D. Gropp, L. C. McInnes, and B. F. Smith (1997), *Efficient management of parallelism in object oriented numerical software libraries, in Modern Software Tools in Scientific Computing*, edited by E. Arge, A. M. Bruaset, and H. P. Langtangen, pp. 163–202, Birkhauser Press, Boston, Mass.

Balay, S., K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang (2010), PETSc users manual (revision 3.1), Rep. ANL-95/11, pp. 1–189, Argonne Natl. Lab. [Available at http://www.mcs.anl.gov/petsc/petsc-3.1/docs/manual.pdf.]

Balay, S., J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang (2012), PETSc web page, Rep. [Available at http://www.mcs.anl.gov/petsc.]

Cunge, J. A. (1969), On the subject of a flood propagation computation method (Muskingum method), *J. Hydraul. Res.*, *7*(2), 205–230.

David, C. H., F. Habets, D. R. Maidment, and Z.-L. Yang (2011a), RAPID applied to the SIM-France model, *Hydrol. Processes*, *25*(22), 3412–3425.

David, C. H., D. R. Maidment, G.-Y. Niu, Z.-L. Yang, F. Habets, and V. Eijkhout (2011b), River network routing on the NHDPlus dataset, *J. Hydrometeorol.*, *12*(5), 913–934.

David, C. H., Z.-L. Yang, and S. Hong (2013), Regional-scale river flow modeling using off-the-shelf runoff products, thousands of mapped rivers and hundreds of stream flow gauges, *Environ. Modell. Softw.*, *42*, 116–132, ISSN 1364-8152, doi: 10.1016/j.envsoft.2012.12.011.

Goldberg, D. (1991), What every computer scientist should know about floating-point arithmetic, *Comput. Surv.*, *23*(1), 5–48.

Hernandez, V., J. E. Roman, and V. Vidal (2005), SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems, *ACM Trans. Math. Softw.*, *31*(3), 351–362.

Higham, N. J. (1990), How accurate is gaussian elimination?, in *Numerical Analysis 1989, Proceedings of the 13th Dundee Conference*, edited by D. F. Griffiths and G. A. Watson, pp. 137–154, Longman Scientific and Technical, Essex, U. K.

Larson, J. W., A. P. Craig, J. B. Drake, D. J. I. Erickson, M. Branstetter, and M. W. Ham (2007), A massively parallel dynamical core for continental- to global-scale river transport, paper presented at the International Congress and Modelling and Simulation (ModSim 2007).

Li, T., G. Wang, and J. Chen (2010), A modified binary tree codification of drainage networks to support complex hydrological models, *Comput. Geosci.*, *36*(11), 1427–1435.

Li, T., G. Wang, J. Chen, and H. Wang (2011), Dynamic parallelization of hydrological model simulations, *Environ. Modell. Softw.*, *26*(12), 1736–1746.

Liang, X., D. P. Lettenmaier, E. F. Wood, and S. J. Burges (1994), A simple hydrologically based model of land-surface water and energy fluxes for general-circulation models, *J. Geophys. Res.—Atmos.*, *99*(D7), 14,415–14,428.

McCarthy, G. T. (1938), The unit hydrograph and flood routing, paper presented at Conference of the North Atlantic Division, U.S. Engineer Department, New London, Conn., 24 June.

McInnes, L. C., J. Moré, T. Munson, and J. Sarich (2011), TAO user manual (revision 1.10.1), Rep. ANL/MCS-TM-242, pp. 1–64, Math. and Comput. Sci. Div., Argonne Natl. Lab. [Available at http://www.mcs.anl.gov/tao.].

National Environment Research Council (1975), *Flood routing studies*, Flood Stud. Rep., London.

O'Donnell, T. (1985), A direct three-parameter Muskingum procedure incorporating lateral inflow, *Hydrol. Sci. J.*, *30*(4), 479–496.

Orlandini, S., and R. Rosso (1998), Parameterization of stream channel geometry in the distributed modeling of catchment dynamics, *Water Resour. Res.*, *34*(8), 1971–1985.

Ponce, V. M. (1986), Diffusion wave modeling of catchment dynamics, *J. Hydraul. Eng.—ASCE*, *112*(8), 716–727.

Turing, A. M. (1948), Rounding-off errors in matrix processes, *Q. J. Mech. Appl. Math.*, *1*(1), 287–308.

U.S. Environmental Protection Agency and U.S. Geological Survey (2010), NHDPlus user guide, *Rep*. [Available at http://www.horizon-systems.com//NHDPlus/NHDPlusV1/documentation/NHDPLUSV1_UserGuide.pdf.]

Veitzer, S. A., and V. K. Gupta (2001), Statistical self-similarity of width function maxima with implications to floods, *Adv. Water Resour.*, *24*(9–10), 955–965.

Vivoni, E. R., G. Mascaro, S. Mniszewski, P. Fasel, E. P. Springer, V. Y. Ivanov, and R. L. Bras (2011), Real-world hydrologic assessment of a fully-distributed hydrological model in a parallel computing environment, *J. Hydrol.*, *409*(1–2), 483–496.

von Bloh, W., S. Rost, D. Gerten, and W. Lucht (2010), Efficient parallelization of a dynamic global vegetation model with river routing, *Environ. Modell. Softw.*, *25*(6), 685–690.

Wilkinson, J. H. (1963), *Rounding Errors in Algebraic Processes*, 167 pp., Prentice-Hall, Englewood Cliffs, N. J.

Xia, Y., et al. (2012a), Continental-scale water and energy flux analysis and validation for the North American Land Data Assimilation System project phase 2 (NLDAS-2): 2. Validation of model-simulated streamflow, *J. Geophys. Res.*, *117*, D03110, doi:10.1029/2011JD016051.

Xia, Y., et al. (2012b), Continental-scale water and energy flux analysis and validation for the North American Land Data Assimilation System project phase 2 (NLDAS-2): 1. Intercomparison and application of model products, *J. Geophys. Res.*, *117*, D03109, doi:10.1029/2011JD016048.