Sub-timing in fluid flow and transport simulations

S. Murty Bhallamudi a,⁎, Sorab Panday b, Peter S. Huyakorn b

a Department of Civil Engineering, Indian Institute of Technology, Madras, India
b HydroGeoLogic Inc., Herndon, VA, USA

Received 11 July 2002; received in revised form 31 December 2002; accepted 27 January 2003

Abstract

A sub-time stepping method is described for computational fluid dynamics problems that utilize implicit-type time marching procedures to resolve transients. In this method, small time-step sizes are used in portions of a domain where interest and activity is high, with larger time-step sizes being applied in other locations. The sub-time step is an integral portion of a larger time step—i.e., multiple sub-time steps over a sub-timed part of a domain add up to the time interval of the full time step used over the remainder of the domain. The technique is particularly suitable for extensive simulations where large portions of a domain are temporally over-discretized. The principles underlying implementation of the implicit sub-timing procedure, the computational effort in relation to conventional implicit time-stepping methods, and an analysis of the effects of sub-timing on the matrix structure are presented. Feasibility and applicability of the implicit sub-timing method is demonstrated through three proof-of-the-concept example problems. These examples include implementation of implicit sub-timing to one-dimensional as well as two-dimensional flow and solute transport problems.

© 2003 Elsevier Science Ltd. All rights reserved.

1. Introduction

Simulations of fluid flow and/or of transport of chemicals or heat in complex, heterogeneous surface or subsurface environments, that utilize a time marching procedure, require that a time-step size be the same over all solution points of a spatially discretized domain. This can lead to an over-discretization in the temporal domain at several locations within each time step of a simulation.

The most glaring instance of this is in simulation of processes having varying time scales. For instance, coupled surface/subsurface flow simulations encounter surface flow responses that are generally much quicker than for subsurface flow, with associated smaller time-step size requirements in the surface domain. Thus for a fully coupled solution, the subsurface domain is temporally over-discretized such that resolution of the surface flow processes is maintained. For this particular case, de-coupled or linked solution schemes have evolved, that use many small time steps to resolve the surface water domain before iteratively solving for one large time step of a subsurface domain module. De-coupled schemes, however, lack robustness and are inefficient or may fail altogether when interactions between the surface and subsurface are large. HydroGeoLogic [9] has performed a fairly detailed study of de-coupled schemes for various cases and has determined that often, it is the coupling fluxes that restrict the time-step size of the subsurface simulation causing a temporal over-discretization of the subsurface. Examples of poor performance of de-coupled schemes are provided by Fairbanks et al. [5].

Temporal over-discretization in flow and transport simulations also becomes evident, for situations that require small time steps to provide temporal resolution and accuracy to activities around a plume or around the center of pumping (where the required spatial resolution is also high), while a much larger time step may have sufficed in other parts of the domain. Explicit solution schemes further suffer from Courant number stability constraints, with temporal over-discretization resulting from the smallest time step that achieves stability being applied over an entire domain. Semi-analytical eigen-value techniques have been applied to overcome the difficulties of time marching. For instance, the Laplace transform method [15] eliminates the time domain by solving a boundary value problem in Laplace space.

⁎ Corresponding author.
E-mail address: spanday@hgl.com (S.M. Bhallamudi).
Semi-analytical methods, however, are applicable only to linear flow solutions or linear transport situations in steady-state flow fields, and are prone to round-off errors.

Non-linearities in flow or transport processes also lead to temporal over-discretization in many practical situations. For instance, solution to Richards’s equation for unsaturated flow is challenging for dry, low capillary soils, and iterative implicit schemes that resolve the non-linearity have difficulty converging unless the time-step size is small. In practical situations, convergence difficulties occur only in a very small portion of a simulation domain with an associated small time-step size requirement, while the remaining nodes are able to accept much larger time-step sizes. Additionally, the situation may be adaptive depending on the current state of the system, with the location of convergence difficulties changing in time. Notwithstanding such complexities which may be handled heuristically, temporal over-discretization in non-linear systems may significantly increase the computational burden of several simulation scenarios. VanderKwaak [16] has experimented with adaptive explicit-implicit methods to reduce the computational burden of implicit or Crank–Nicolson time-stepping schemes. Portions of a simulation domain could adaptively be made explicit (if they were deemed to be temporally over-discretized) to reduce the size of the matrix containing the implicit set of equations that is solved at each time step. If certain criteria were exceeded at any location within an iteration, an explicit node is turned implicit during the subsequent iteration (and vice versa). He found slight to moderate enhancements in simulation speed using this procedure, which is simple to implement and holds considerable promise. Temporal over-discretization is particularly acute in multi-phase or non-isothermal system simulations wherein NAPL contaminant movement or large thermal flows (with associated non-linear difficulties) occur only in small regions of a domain. Adaptive implicit schemes [6] help alleviate the overall computational burden by treating the saturation variables explicitly in inert portions of the domain, to reduce the size of the matrix that is solved. These techniques, however, do not directly tackle the issue of temporal over-discretization.

Temporal over-discretization also prevails in front tracking situations and for cases where a boundary is rapidly varying in a small portion of the solution space. In the former case, adaptive sub-gridding techniques have been developed to resolve spatial gradients at the front, however, a finer grid results in higher Courant numbers which, when resolved, result in temporal over-discretization in the remainder of the domain. A rapidly varying boundary condition (like a rainfall event) requires small time steps to resolve the hydrograph of the event, causing temporal over-discretization in the remaining portions of the simulated domain where the flashiness of the event is not felt. Several other instances also exist in practical computational fluid dynamics situations, whereby temporal over-discretization plays a major role in increasing the computational burden. For large-scale simulations of complex systems, it is therefore desirable to alleviate this burden.

The use of sub-timing can be helpful in reducing the computational burdens of temporal over-discretization. It is not unlike the concept of sub-gridding in spatial discretization (for example, see [4]), whereby nested grids are used to provide spatial resolution in certain regions of a simulation domain, with larger grid blocks in other locations where “spatial over-discretization can be avoided”. Similarly, in sub-timing, small time-step sizes are used in portions of a domain where interest and activity is high, with larger time-step sizes being applied in other locations to avoid temporal over-discretization. Like a sub-grid, which is an integral portion of a larger grid block, a sub-time step is an integral portion of a larger time step—i.e., multiple sub-time steps over a sub-timed part of a domain (of possibly differing time intervals), add up to the time interval of the full time step used over the remainder of the domain. Again like in sub-grids, sub-timing may be performed at several different locations in a domain, with sub-time-step sizes and number of intervals being possibly different among the various locations.

For explicit time discretization schemes, a method of sub-timing has been presented by Singh and Bhallamudi [11,12]. The method is implemented along with a sub-gridding technique for capturing sharp fronts in one and two dimensions, with high order Essentially Non-Oscillatory (ENO) methodologies for advancing the solution in space and time. For a domain discretized into separate coarse and fine-grid locations, computations proceed from an upstream boundary for all coarse grid-blocks using a time-step size ($\Delta t_1$) that satisfies the coarse grid-block stability constraint. Once all computations in a contiguous coarsely discretized region are complete, the nodes within the finely discretized region near the front are solved using a time step size ($\Delta t_2$), that matches the fine grid-blocks stability constraints (note that $\Delta t_2 = M\Delta t_1$ where $M$ is an integer). Upstream boundaries for flux and/or head for the finely discretized domain are obtained from the adjacent coarse grid domain, with values at sub-times obtained by linear interpolation in time of the coarse grid values. An order of magnitude speedup was demonstrated over traditional explicit finite-difference schemes, for one- and two-dimensional problems. Explicit sub-timing schemes have also been discussed for structural dynamics problems by Smolinski et al. [14] Gravouil and Combescure [8] and Klisinski [10], and for first order diffusion problems by Belytschko and Liu [3] and Smolinski et al. [13].

This paper presents a sub-timing methodology for implicit-type time-stepping schemes (i.e., schemes that
solve a system of matrix equations at all iterations or time steps of a simulation. The methodology is presented in the framework of a semi-implicit time discretization scheme, and can be adapted to all implicit-type schemes that perform a time-integration on a set of spatially related matrix equations. The computational effort of sub-timing is then analyzed in relation to conventional implicit time-stepping methods, followed by a brief analysis of the effects of sub-timing on matrix structure and diagonality characteristics. Proof-of-the-concept example problems finally demonstrate feasibility and applicability of the implicit sub-timing method.

2. Methodology

The continuity equation for fluid flow or for transport of contaminant or heat may be written for a semi-implicit time discretization scheme in a general form as

\[
\begin{align*}
\frac{S_i}{\Delta t} (h_i^{t+\Delta t} - h_i^t) &= \theta \sum_{j}^{N_j} (\text{Flux})_{ij}^{t+\Delta t} \\
+ (1-\theta) \sum_{j}^{N_j} (\text{Flux})_{ij}^t &\pm Q_{ij}^{t+\Delta t} \\
\end{align*}
\]

where \(i\) is the nodal index of a spatially discretized system \((i = 1, \ldots, N)\), \(S_i\) is the storage capacity of node \(i\), \(\Delta t\) is the time-step size, \(h_i^{t+\Delta t}\) is the variable of solution (fluid potential or head for a flow simulation; concentration for a contaminant transport simulation; or temperature for a thermal simulation), \(h_i^t\) is the known value of the variable at time \(t\), and \(\theta\) is a time weighting factor \((\theta = 1\) for a fully implicit scheme, \(\theta = 0.5\) for a Crank–Nicolson scheme, and \(\theta = 0\) for a fully explicit scheme). The term \((\text{Flux})_{ij}\) is the net flux into cell \(i\) from its neighbor \(j\), and \(N_j\) is the total number of neighbors. The superscript \(t + \Delta t\) indicates the evaluation of term at time level \(t + \Delta t\), and superscript \(t\) indicates the evaluation of term at time level \(t\). The flux at any time level depends on the variable \(h\), at cell \(i\) and at its neighbor \(j\) at that time level. \(Q_{ij}\) is an external source or sink which is evaluated fully implicitly at node \(j\). The flux into node \(i\) is typically represented in one, two, or three dimensions using a finite-element, finite-difference, or finite-volume spatial discretization of the governing equation, and is generally a function of \(h\) for node \(i\) and its neighbors. For instance, Darcy’s law governs saturated or unsaturated porous medium flow where fluid flux is a function of the gradient in potential between nodes \(i\) and \(j\). Contaminant fluxes including advection and Fickian dispersion are governed by nodal concentrations and concentration gradients at node \(i\) and its adjacent nodes, as are heat fluxes on temperature and temperature gradients. For an appropriate quantification of flux and after treatment of possible non-linearities, Eq. (1) forms a system of \(N\) equations in the \(N\) unknowns, \(h_i^{t+\Delta t}\), \(i = 1, \ldots, N\). In a traditional time marching procedure, a matrix system of \(N\) equations is inverted for a linear system (or in an iterative linearized procedure for a non-linear set of equations) to ultimately obtain solution for the unknown vector \(h_i^{t+\Delta t}\).

Sub-timing is a procedure for providing sub-levels of time discretization in certain portions of a simulation domain as exemplified in Fig. 1. For this example, one time step of a simulation for the entire domain \((\Delta t)\) is subdivided into multiple \((M = 5)\) pre-determined time-step intervals for a portion of the domain (nodes 3–5) where sub-timing was deemed necessary (either a priori, or via adaptive schemes through the course of a simulation). For any sub-timed cell, “s”, temporal discretization equations can be written for each of the \(m = 1, \ldots, M\) sub-time steps of a solution in a general form as

\[
\begin{align*}
\frac{M_s S_i}{\Delta t} (h_i^{s+m\Delta t/M_s} - h_i^{s+(m-1)\Delta t/M_s}) \\
&= \theta \left( \sum_{j}^{N_j} (\text{Flux})_{ij}^{s+m\Delta t/M_s} \right) \\
+ (1-\theta) \left( \sum_{j}^{N_j} (\text{Flux})_{ij}^{s+(m-1)\Delta t/M_s} \right) &\pm Q_{ij}^{s+m\Delta t/M_s} \\
\end{align*}
\]

where \(h_i^{s+m\Delta t/M_s}\) is the value of the variable of solution at sub-time level \(m\), \((\text{Flux})_{ij}^{s+m\Delta t/M_s}\) is the flux into cell “s” from neighbor \(j\) at sub-time level \(m\), and the source/sink term, written fully implicitly, may vary for node “s” at the sub-time level. For the sake of simplicity, all sub-step sizes are of equal duration in Eq. (2), with a value of \(\Delta t_s = \Delta t/M_s\). Eq. (1) is still applicable for temporal discretization of the full time-step size in the remainder of the domain (nodes 1, 2, 6, 7, etc. in Fig. 1). Consequently, for each sub-timed node within a domain, there

![Fig. 1. Temporal discretization of a domain using sub-time steps.](image-url)
expressed in terms of the variables of solution, the equations. Hence, Eq. (2) is kept consistent with Eq. (1) in its semi-implicit scheme by maintaining the same implicitness factor, \( \theta \), as that of Eq. (1).

The flux term, \( \langle \text{Flux} \rangle_{ij}^{t+\Delta t/M_s} \), at the \( m \)th sub-time interval in Eq. (2), between two sub-timed nodes (e.g., between nodes 3 and 4 or between 4 and 5 of Fig. 1), is expressed in terms of the variables of solution, \( h^{t+m\Delta t/M_s} \), at cell “s” and its \( j \)th (sub-timed) neighbor. The flux term, \( \langle \text{Flux} \rangle_{ij}^{t+m\Delta t/M_s} \), at the \( m \)th sub-time interval, between a sub-timed node “s” and a fully time-stepped node “j” (for example, between nodes 2 and 3 or between 5 and 6 in Fig. 1), is computed without interpolation, with the same implicitness factor as for the rest of the equations. Hence, the flux value of the fully time-stepped level at the same implicitness level for factor \( \theta \).

\[
\langle \text{Flux} \rangle_{ij}^{t+\Delta t/M_s} = \theta \langle \text{Flux} \rangle_{ij}^{t+\Delta t} + (1 - \theta) \langle \text{Flux} \rangle_{ij}^{t}
\]

(3)

is used as the compatibility condition for flux across the junction of a fully time stepped, and a sub-timed domain. Note that this method can be applied recursively, for a recursively sub-timed domain.

The “non-interpolated” interface flux between a sub-timed node and a fully time-stepped node of Eq. (3) applies to all situations of semi-implicitness. This is shown by equating the volume fluxed through the right face of cell 2 (see Fig. 1) over the time step \( t \), to the total volume fluxed through all the sub-time levels at the interface of cells 2 and 3. Thus, referring to the example of Fig. 1, we have

\[
[\theta(\text{Flux})^{t+\Delta t} + (1 - \theta)(\text{Flux})^t] \Delta t
\]

\[
= [(1 - \theta)(\text{Flux})^t + \theta(\text{Flux})^{t+\Delta t/5}] \Delta t/5
\]

\[
+ [(1 - \theta)(\text{Flux})^{t+\Delta t/5} + \theta(\text{Flux})^{t+2\Delta t/5}] \Delta t/5
\]

\[
+ [(1 - \theta)(\text{Flux})^{t+2\Delta t/5} + \theta(\text{Flux})^{t+3\Delta t/5}] \Delta t/5
\]

\[
+ [(1 - \theta)(\text{Flux})^{t+3\Delta t/5} + \theta(\text{Flux})^{t+4\Delta t/5}] \Delta t/5
\]

\[
+ [(1 - \theta)(\text{Flux})^{t+4\Delta t/5} + \theta(\text{Flux})^{t+5\Delta t/5}] \Delta t/5
\]

(4)

which, upon simplification of the right-hand side, and rearranging gives

\[
4[\theta(\text{Flux})^{t+\Delta t} + (1 - \theta)(\text{Flux})^t] = (\text{Flux})^{t+\Delta t/5} + (\text{Flux})^{t+2\Delta t/5} + (\text{Flux})^{t+3\Delta t/5}
\]

\[
+ (\text{Flux})^{t+4\Delta t/5}
\]

(5)

The condition of Eq. (5) is satisfied for all values of \( \theta \), only when

\[
[\theta(\text{Flux})^{t+\Delta t} + (1 - \theta)(\text{Flux})^t] = (\text{Flux})^{t+5\Delta t/5}
\]

\[
= (\text{Flux})^{t+2\Delta t/5} = (\text{Flux})^{t+3\Delta t/5} = (\text{Flux})^{t+4\Delta t/5}
\]

(6)

i.e., the flux values used at the sub-time levels are the same as the flux value of the fully time-stepped level at the same implicitness level for factor \( \theta \). For use of a linear interpolation for fluxes between a sub-timed node and a fully time-stepped node, the condition of Eq. (5) is satisfied only for \( \theta = 0.5 \).

For an appropriate quantification of the flux (advection plus dispersion terms for transport or a diffusion-type term for flow), Eqs. (1) and (2) completely define the system of \((N - N_s) + \sum_{M_s}^{M_f} M_s\) equations for a temporally sub-discretized domain containing \(N_s\) nodes that are sub-timed, each with its \(M_s\) sub-time levels. The corresponding unknowns are the \((N - N_s)\) values of the variable of solution at \((t + \Delta t)\) in the fully time stepped portion of a simulation domain and at the \(M_s\) sub-time levels for each of the \(N_s\) nodes that have sub-time discretization. Hence the matrix size is increased by \((\sum_{M_s}^{M_f} M_s - 1)\) for the additional unknowns that result from this scheme. However, the matrix is solved only once to advance from time \( t \) to \( t + \Delta t \), rather than \( M_s \) times for the same \( \Delta t = \Delta t/M_s \) level of accuracy at the sub-timed nodes, needed by using a traditional implicit time-stepping scheme.

3. Analysis of computational effort

An analysis of the computational efficiency of the sub-timing method as compared to traditional time marching schemes for semi-implicit formulations is provided here to estimate the possible computational savings achieved by its application. The analysis here is based on the tradeoffs between having a larger matrix to solve for the sub-timing scheme, versus having to solve a smaller matrix more number of times to achieve the same level of time-discretization at the sub-timed nodes, using a traditional time marching procedure. The cost per time step, \( C_s \), associated with a traditional time-stepping scheme is dependent on the size of the matrix being solved and can be written as

\[
C_s = K(N)^x
\]

(7)

where \( K \) is a coefficient and \( x \) is an exponent, which depends among other things, on the nodal connectivity and type of matrix solver used to invert the matrix of \( N \) equations. In using a sub-timing method, it can be assumed that fine time-steps are necessary only on the \( N_s \) sub-timed cells (all with equal \( M_s \) sub-time intervals for this analysis which may be considered an extreme case for differently sub-timed nodes) with the full time step, \( \Delta t \), being sufficient for the rest of the domain. Traditional time stepping will need to solve its system \( M \) times
to achieve the same accuracy, giving an associated cost, \( C_f \), as \( M \) times the unit cost. Hence,

\[ C_f = K(N)^\gamma M \]  

(8)

The cost \( C_s \) associated with one full time step of the sub-timing method can be written (neglecting the effect of expanded row connectivity of sub-timed nodes) as

\[ C_s = K[N - N_s + MN_s]^\gamma \]  

(9)

The sub-timing scheme is computationally worth the effort if its cost is less than the cost of achieving the same temporal accuracy using traditional semi-implicit time-stepping schemes (i.e., when \( C_s < C_f \)). Alternately, a computational efficiency factor may be defined as \( \phi = C_f/C_s \), which should be greater than unity for sub-timing to be a less costly solution. Thus,

\[ \phi = \frac{N^\gamma M}{(N - N_s + MN_s)^\gamma} > 1 \]  

(10)

Or upon rearranging,

\[ M^{1/\gamma} - (M - 1)N_s/N > 1 \]  

(11)

For an “ideal” matrix inversion scheme, wherein the cost is linearly proportional to matrix size (i.e., \( x = 1 \)), the above inequality is always satisfied and it would always be worth pursuing a sub-timing scheme to avoid temporal over-discretization. On the other hand, if \( x = 2 \) and if \( M = 4 \), computational savings are achieved only if \( N_s/N < 1/3 \) (i.e., if time nesting is done for less than a third of the total simulation nodes). Current iterative sparse matrix solvers have \( x \) ranging typically between 1.0 and some value around 1.5 (without accounting for overhead of setup), making sub-timing a competitive technique for achieving computational savings for large complex situations. It is again noted here that the values of \( K \) and \( x \) in Eq. (8) may be different from that in Eq. (9), depending on the matrix solver’s characteristics, due to the altered connectivity structure for the sub-timed nodes. Further, the above conclusions apply to a linear system. For a non-linear system, the sub-timing method may alter the number of iterations required for convergence during a time-step computation, and hence the computational cost discussed here may not be appropriate. This is further detailed in the section on illustrative examples.

4. Effect of sub-timing on matrix characteristics

Robust and efficient matrix inversion schemes are available for solving large sparse matrices that are generated from implicit-type time marching schemes for solution to flow or transport problems (see [1]). The implications of the sub-timing method on application of these matrix solvers are explored here, since the different solvers may have differing convergence behavior due to the sensitivity of different pre-conditioning and acceleration schemes to the issues discussed.

The matrix that is generated for a traditional implicit-type time marching procedure for flow and transport simulations, has a sparse structure with number of rows equal to the number of nodes, \( N_s \), and number of non-zero columns limited to the maximum number of neighbors plus one itself (i.e., number of non-zero columns equals \( N_f + 1 \)). The sub-timing procedure adds \( \sum N_s (M_s - 1) \) rows to the matrix for the \( m = 2, \ldots, M_s \) additional unknowns of solution at each sub-timed node “s”. The equation for \( m = 1 \) at a sub-timed node can replace Eq. (1) in the respective sub-timed node’s row since its connectivity is the same as for Eq. (1). For the additional rows, however, the number of non-zero columns doubles to \( 2N_f + 2 \) to accommodate the fluxes from the various neighboring cells, that are expressed in terms of the variables of solution at two adjacent unknown sub-time levels \( m \) and \( m - 1 \) (see Eq. (2)). Variable matrix connectivities are readily accommodated by unstructured matrix storage schemes, however, sub-timing can be wasteful of array space in structured matrix storage schemes, unless special considerations are made for the sub-timed nodes. One method of using sub-timing with structured matrix storage schemes would be to place all additional matrix rows (of connectivity \( 2N_f + 2 \) for the \( m = 2, \ldots, M_s \) sub-time intervals) grouped together and appended below the first \( N_s \) rows (of connectivity \( N_f + 1 \), with appropriate maximum connectivities provided to each section of a matrix containing two connectivity structures. However, for a fully implicit time discretization, (\( \theta = 1 \), the fluxes in Eq. (2) are expressed only at the \( m \)th time interval, causing an increase in matrix connectivity by only one, for the additional sub-timed rows, to accommodate the left-hand side of Eq. (2). Thus, the maximum matrix connectivity for a fully implicit sub-time-stepping scheme is \( N_f + 2 \). This extra column is easy to accommodate even in structured matrix storage schemes without overly excess storage allocation.

Setup of the additional rows of the matrix for the sub-timing scheme incurs an overhead dependent on the matrix storage scheme as well as on the pre-conditioning used. Thus, if a sub-timing scheme is fixed (for instance, if it was pre-determined that all surface water nodes would use 10 sub-time steps per groundwater node time-step in a coupled surface–subsurface flow model), one can assign the additional matrix rows and connectivities, and set up appropriate workspaces, prior to time-looping within a simulation. If, however, the sub-timing were to be adaptive (in either the location of sub-timing, or the number of sub-time intervals), the overhead for setup of connectivities for the additional rows at every time step should be taken into consideration before using the procedure.
The sub-timing equation (2) can affect the diagonality characteristics of the matrix system of equations for the most general case. The additional rows in the matrix, for solution to the $m = 2, \ldots, M$ sub-time-step values of the variable $h$ at each node, incorporate off-diagonal terms whose absolute sum may be larger than the magnitude of the diagonal term due to a semi-implicit formulation (caused by the second term on the right-hand side of Eq. (2)). It should be noted that for sub-timing, the spatial connectivity is not altered—only that solution is required for multiple equations in multiple unknowns (the multiple sub-time steps) at a sub-timed node. Therefore, mixed point/block solution schemes may also be explored further for sub-time stepped solutions. For the case of a fully implicit time discretization, the second term on the right-hand side of Eq. (2) is zero, therefore, if the storage capacity, $S_s$, is constant in time, the sum of the off-diagonal terms is equal to the magnitude of the diagonal term for the additional rows of the matrix.

Finally, symmetry considerations are not altered by the sub-time stepping approach. Thus, if the flux term is symmetric, Eq. (2) does not change that, and special matrix storage and solution schemes used for symmetric matrices may also be used with the sub-timing procedure.

5. Illustrative examples

The sub-time stepping methodology may typically be used in regions where nested grids are also applied. The use of sub-timing with nested grids is a straightforward procedure, with nesting formulas providing the flux term (in Eq. (2)) between spatially nested and un-nested portions of a simulation domain. de Marsily [4] provides equations for flow across nested squares (at one level of nesting) which may readily be generalized to include nested rectangles, and multiple levels of spatial nesting.

Three proof-of-the-concept example problems are provided here, to validate the sub-time stepping methodology and demonstrate its application. The first two situations exemplify the methodology for flow problems. These problems do not incorporate sub-gridding in space, because they are one-dimensional illustrations for clarity and simplicity. The third example illustrates application of sub-time stepping for a contaminant transport simulation, which also includes spatial sub-gridding. The codes written to solve these examples use a dense LU decomposition scheme due to its lack of complexity in implementing the concepts of sub-timing.

5.1. Example 1. One-dimensional surface flow with a rapidly varying upstream boundary condition

This example problem depicts the use of sub-time stepping in obtaining accurate and efficient solutions to problems with rapidly varying boundary conditions. For the sake of simplicity, one-dimensional unsteady routing equations with zero-inertia assumption are solved for this demonstration. These equations for flow in a wide rectangular open-channel are given by

$$\frac{\partial H}{\partial t} + \frac{\partial q}{\partial x} = 0$$  \hspace{1cm} (12)

with the flux expressed as

$$q = \text{sign}\left(\frac{\partial H}{\partial x}\right) \left(\frac{h}{n}\right)^{1/3} \left(\frac{\partial H}{\partial x}\right)^{1/2}$$  \hspace{1cm} (13)

where $H$ is the water level above the datum, $q$ is the flux per unit width, $h$ is the flow depth, $n$ is the Manning roughness coefficient, $x$ is the distance along the channel, and $t$ is the time. Flow depth, $h$, is obtained from the water level, $H$, using the following equation:

$$h = H - Z$$  \hspace{1cm} (14)

where $Z$ is the elevation of the channel bottom. Eqs. (13) and (14) are substituted into Eq. (12) which is solved for the flow system.

Fig. 2 depicts the problem geometry, conceptualization, and simulation parameters for this study. Total length of the domain is equal to 600.0 km, with a uniform channel slope of 0.0002, and roughness coefficient of 0.11. Initial uniform flow depth through the domain is equal to 2.5 m, with the corresponding initial flow rate calculated using Eq. (13). The upstream boundary consists of prescribing a time-varying flux at the first node according to Log Pearson Type-III hydrograph as follows:

$$q_u = q_0 + (q_p - q_0)e^{-\left(\frac{t - t_p}{t_p - t_0}\right)^{\frac{1}{n}}} \left(\frac{t}{t_p}\right)^{\frac{1}{n}}$$  \hspace{1cm} (15)

where $q_u$ is the upstream boundary flux per unit width as a function of time, $q_0$ is initial flow rate, $q_p$ is the peak flow rate ($= 20$ m$^2$/s), $t_p$ is the time to peak flow ($= 6$ h), and $t_0/t_p = 0.95$. $t_0$ = time to centroid of the hydrograph. A normal depth boundary condition is placed at the downstream end of the simulated channel.

![Flow Surface at time t](image)

Flow Surface at time t

Initial Flow surface

U/S boundary

Log Pearson

Hydrograph

Channel bed

Bed slope = 0.0002

Channel length = 600 km

Uniform roughness

Manning $s$ n = 0.11

D/S boundary

Normal flow

Fig. 2. Description of flow problem for Example 1.
Numerical solution up to 200 h is obtained by discretizing the channel into 200 reaches each of length \( \Delta x = 3000 \) m and using fully implicit time-stepping scheme \((\theta = 1)\). Three different simulations are performed to evaluate the accuracy and speed of the sub-timing procedure:

1. using a traditional time marching procedure with \( \Delta t = 4 \) h;
2. using a traditional time marching procedure with \( \Delta t = 0.5 \) h; and
3. using a sub-timing approach with eight sub-time steps of size \( \Delta t = 0.5 \) h applied to nodes 1–10, and a large time-step size of \( \Delta t = 4 \) h applied at all other nodes.

The time stepping in each of the above cases is applied uniformly to the marching procedure throughout the simulation. Figs. 3 and 4 present the flow depth variation with time at \( x = 9.0 \) km, and at \( x = 150.0 \) km, respectively, obtained using the three alternative simulations. Numerical solution obtained using the traditional method and a small time-step value (\( \Delta t = 0.5 \) h) is considered as the reference solution in the absence of an analytical solution for this problem. It can be seen from Figs. 3 and 4 that numerical results obtained using the sub-timing approach and a large time step, \( \Delta t = 4 \) h match very well with the reference solution. However, the solution obtained using the traditional method and \( \Delta t = 4 \) h is noted to be incorrect. The source of this error lies in the large computational time-step size, \( \Delta t \), at the upstream boundary, which is insufficient to capture the rapid boundary response as shown by the inflow hydrograph of Fig. 5. The computed peak flow in this case is equal to 8 m\(^2\)/s (occurring at 8.0 h) as compared to the reference value of 20 m\(^2\)/s (occurring at 6.0 h). The sub-timing approach, however, makes it possible to implement the specified, time-varying upstream flux boundary conditions exactly (see Fig. 5) even though a larger time-step value of 4.0 h is taken elsewhere.

The solution obtained using the sub-timing approach matches excellently with the reference solution at \( x = 9.0 \) km (Fig. 3). However, there is some discrepancy between the sub-timing results and the reference solution at \( x = 150 \) km. This is because sub-timing is applied only for a distance of 27.0 km from the upstream end and the effects of the larger time-step value used beyond 27.0 km are felt at 150 km. It may be noted, however, that the sub-timing solution is much superior to the traditional method with large time step, even at \( x = 150 \) km (Fig. 4) since it implements the upstream boundary.
appropriately, and maintains the incoming mass. Sources of error for the numerical solution are: (i) incorrect simulation of the upstream boundary condition, (ii) discretization error due to a large time-step value, and (iii) errors occurring at the interface of a sub-timed node and a regular node from the flux compatibility condition of Eq. (6). In the present example, the error due to the first two reasons is minimized. Error due to the last reason is inherent to the sub-timing method, and cannot be eliminated, though this error may be further reduced by use of a Crank–Nicolson scheme \((\theta = 0.5)\), and a linear interpolation of the interface fluxes between sub-timed and fully time-stepped nodes.

Theoretically, the computational efficiency of the sub-timing approach as compared to the traditional time marching approach may be calculated using Eq. (10). For this example, \(N = 201, N_s = 10, M = 8, \) and \(x = 3\) for the LU decomposition method used in the simulator adapted for these calculations, giving a theoretical efficiency of \(\phi = 3.264\). The simulations performed with CPU times of 1365 s, and 600 s as actual values of \(C_f\) and \(C_s\), respectively, giving an efficiency of \(\phi = 2.275\). The difference in the theoretical and actual efficiencies is due to the difference in the number of iterations taken for each time step computation in a non-linear problem. Total number of iterations (for the complete simulation) in the traditional method was equal to 432 (1.08 per time step) while it was equal to 78 (1.56 per time step) in the case of the sub-timing simulation. Also, it may be noted here that for a more efficient matrix solver with \(x = 2.0\), the theoretical efficiency for sub-timing for this problem is calculated as \(F = 4.4\), thus underscoring the need for an efficient iterative sparse matrix solver in obtaining larger speedups using the sub-timing procedure. It should further be noted that for one-dimensional example problems, the traditional approach may use an efficient tri-diagonal solution scheme, while the tri-diagonal structure is destroyed by the sub-timing approach—however, this example is provided to demonstrate simply and succinctly, the applicability and proof-of-concept of the sub-timing approach, which may not be beneficial for simple systems.

5.2. Example 2. One-dimensional surface flow with convergence difficulties in portions of the simulated domain

This example problem depicts the use of sub-timing in obtaining efficient solutions to highly non-linear situations in certain portions of a simulation domain. The one-dimensional zero-inertia routing equations (12)–(14) are again solved for this demonstration. Fig. 6 depicts the problem geometry, conceptualization, and simulation parameters. Total length of the domain is equal to 20.0 km, with a uniform channel slope of 0.0001. The computational domain is divided into 200 reaches, each of 200 m length. The initial flow depth at the upstream end, \(h_0\), equals 1.0 m. Unlike in the previous example problem, a parameter non-linearity is introduced in the present problem. A Manning roughness value equal to 0.15 is specified for all the computational reaches except for every fifth reach. The Manning roughness value = 0.012 is specified for every fifth reach of the domain. Initial non-uniform flow depth variation along the channel is first obtained using Eq. (13) for a specified initial flux of \(q = q_0\) (= 0.01 m²/s). The upstream boundary is varied during the simulation according to a triangular hydrograph shape. Peak flux at the upstream section equals 9.0 m²/s, which occurs at 2.0 h. The flux reduces to the initial value at \(t = 8\) h, and remains at this value thereafter. A normal depth boundary condition is placed at the downstream end of the simulated channel.

Numerical computations are performed up to 3 h using a fully implicit weighting factor of \(\theta = 1\). Three different simulations are performed to assess the use of sub-timing in resolving non-linearities more efficiently. The simulations include:

1. using a traditional time marching procedure with \(\Delta t = 450\) s,
2. using a traditional time marching procedure with \(\Delta t = 150\) s; and
3. using a sub-timing approach with three sub-time steps of size \(\Delta t = 150\) s applied to nodes 1–12, and a large time-step size of \(\Delta t = 450\) s applied to the rest of the domain.

Cumulative number of iterations versus time is plotted for all three runs in Fig. 7. Cumulative number of iterations indicates how the non-linearity is being resolved during each time step of computation. For the first simulation using a traditional time marching procedure with \(\Delta t = 450\) s, total number of iterations for a 3 h simulation period is equal to 219, while it is equal to
only 78 when sub-timing is used. Total number of iterations for the 3 h simulation period is equal to 186 with traditional time-stepping and a smaller $\Delta t$ of 150 s. Fig. 8 shows the flow depth variation in time at node 3 for the second and third simulations. It is noted that the sub-timing scheme produces essentially the same results as the traditional time-stepping approach. However, the sub-timing approach performs 1.7 times faster than the traditional time-stepping approach, since it enables the use of larger time-step values in most of the domain, to obtain a solution to the non-linear problem. Theoretically, the computational efficiency for this problem can be obtained from Eq. (10) to be $\phi = 2.14$ for a matrix solution scheme with $x = 3$. Note that the predetermined sub-timed nodes in this example are chosen from analysing the non-convergence behavior of the simulation that uses a traditional time marching approach with a large time-step size of $\Delta t = 450$ s. This may be done adaptively in a complex simulation, and the sub-timed nodes may be introduced into the simulation only when convergence difficulties arise.

5.3. Example 3. Two-dimensional solute transport with a rapidly varying upstream boundary condition

Like in Example 1, this example problem depicts the use of sub-time stepping in obtaining accurate and efficient solutions to problems with rapidly varying boundary conditions, but for the case of two-dimensional problems. For the sake of simplicity, two-dimensional solute transport is considered in a uniform flow field. A finite-length strip source is located asymmetrically along the $y$-axis at $x = 0$ in a steady unidirectional seepage velocity field as shown in Fig. 9. The concentration at the strip source is a given function of time. It is required to compute the temporal and spatial variation of concentration in the domain. The governing equation for the above solute transport problem under saturated groundwater flow conditions is given by [7]

$$R_d \frac{\partial C}{\partial t} = \nabla \cdot (D_h \nabla C) - \nabla \cdot (\vec{V} C) - \lambda R_d C \quad (16)$$

where $C$ is the solute concentration, $V$ = pore water velocity vector, $R_d$ = retardation factor, $\lambda$ = first order decay coefficient, $D_h$ = hydrodynamic dispersion tensor and $t$ = time. The elements of the dispersion tensor $D_{xx}$, $D_{yy}$, and $D_{xy} (= D_{yx})$ are generally functions of velocity and molecular diffusion. In the present study, the case of a homogeneous and isotropic medium under unidirectional groundwater flow with two-dimensional dispersion is considered. This means the dispersion coefficients $D_{xx} = D_{yy} = 0$. Also, the decay coefficient $\lambda = 0$, and the retardation factor $R_d = 1$, for a non-reactive solute transport.

Eq. (16) is solved for specified time-varying concentration along the strip source shown in Fig. 9. The

![Fig. 7. Cumulative number of iterations for alternative simulations for Example 2.](image1)

![Fig. 7. Cumulative number of iterations for alternative simulations for Example 2.](image2)

![Fig. 8. Variation of flow depth with time at node 3 for Example 2.](image3)

![Fig. 8. Variation of flow depth with time at node 3 for Example 2.](image4)

![Fig. 9. Description of transport problem for Example 3.](image5)
lateral boundaries are no flow boundaries while the downstream boundary condition is a far field boundary condition. In the present work, the numerical solution is obtained by discretizing the computational domain into a total of 587 rectangular, unstructured finite-volume cells as shown in Fig. 10. The cell size is non-uniform. Also, note that the above spatial discretization uses the concept of sub-gridding as explained by de Marsily [4], to get better resolution in areas right ahead of the strip source. Flux computations for applying the implicit finite volume scheme to this sub-grid cell arrangement follows the method detailed by de Marsily [4], with further generalization for rectangular (non-square) grids.

5.4. Case I. Validation

A computer code is developed for solving the two-dimensional solute transport equation (Eq. (16)) using both the sub-gridding and the sub-timing concepts. The computer code is first validated, by comparing the numerical results with analytical solutions. Fig. 9 depicts the problem geometry and conceptualization for this study. The rectangular domain is 50 m in the x-direction and 30 m in the y-direction. The geometrical parameters for the source (see Fig. 9) are $B_1 = 5$ m, $B_2 = 10$ m, and $B_3 = 15$ m. The uniform pore velocity, $u$, is 0.1 m/day. The longitudinal and transverse dispersivities, $D_{xx}$ and $D_{yy}$, are 1.0 and 0.1 m$^2$/day, respectively. The initial concentration is given by $C(x,y,0) = 0$. The boundary condition at $x = 0$, $t > 0$ is given by

$$C(0,y,t) = \begin{cases} 
0 & 0 < y < B_1 \\
1.0 & B_1 < y < B_1 + B_2 \\
0 & B_1 + B_2 < y < 35 \text{ m}
\end{cases}$$

(17)

As mentioned earlier, the computational domain is discretized using the spatial grid shown in Fig. 10. Numerical solution up to 100 days is obtained using the Crank–Nicolson time-stepping scheme ($\theta = 0.5$). The time-step size $\Delta t$ is taken as 1 day for all cells except those which are sub-timed. Sub-timing with four sub-time levels is applied to a total of 30 cells, which are shown shaded in Fig. 10. Fig. 11 presents the longitudinal concentration distribution at $t = 100$ days as a function of $x$ for $y = 10$ m. Fig. 12 presents the lateral concentration distribution for $x = 5$ m at $t = 100$ days. The analytical results shown in Figs. 11 and 12 are taken from [2]. As can be seen from these figures, the proposed numerical method gives accurate solutions.

5.5. Case II. Applicability of sub-timing

The example problem in this case depicts the use of sub-time stepping in obtaining accurate and efficient solutions to problems with rapidly varying boundary...
conditions. The problem geometry and conceptualization for this case are same as those described for the example in Case I (see Fig. 9). However, the uniform pore velocity, \( u \), is 0.25 m/day, and the longitudinal and transverse dispersivities, \( D_x \) and \( D_y \), are 2.0 and 0.2 m\(^2\)/day, respectively. The initial concentration is given by \( C(x, y, 0) = 0 \). The boundary condition at \( x = 0, t > 0 \) is given by

\[
C(0, y, t) = \begin{cases} 
0 & \text{as shown in Fig. 13} \\
0 < y < B1 & \text{as shown in Fig. 13} \\
B1 < y < B1 + B2 & \text{as shown in Fig. 13} \\
B1 + B2 < y < 35 \text{ m} & \text{as shown in Fig. 13}
\end{cases}
\]

(18)

The computational domain is discretized using the spatial grid shown in Fig. 14. Numerical solution up to 30 days is obtained using the Crank–Nicolson time-stepping scheme \((\theta = 0.5)\). Three different simulations are performed to evaluate the accuracy and speed of the sub-timing procedure:

1. using a traditional time marching procedure with \( \Delta t = 2 \) days;
2. using a traditional time marching procedure with \( \Delta t = 0.5 \) days; and
3. using a sub-timing approach with four sub-time steps of size \( \Delta t = 0.5 \) days applied to cells shown shaded in Fig. 14 and a large time-step size of \( \Delta t = 2 \) days applied to all the other cells.

The time stepping in each of the above cases is applied uniformly through out the simulation. Fig. 15 presents the concentration variation with time at \((x, y) = (6.5 \text{ m}, 9.5 \text{ m})\), obtained using the three alternative ways. Numerical solution obtained using the traditional method and a small time-step value \((\Delta t = 0.5 \text{ days})\) is considered as the reference solution in the absence of an analytical solution for this problem. It can be seen from Fig. 15 that numerical results obtained using the sub-timing approach and a large time-step size of \( \Delta t = 2 \) days, match very well with the reference solution. However, the solution obtained using the traditional temporal discretization method and \( \Delta t = 2 \) days is noted to be incorrect. The source of this error obviously lies in the large computational time-step size, \( \Delta t \), at the upstream boundary, which is insufficient to capture the rapid boundary response of the inflow concentration variation of Fig. 13. The sub-timing approach, however, makes it possible to implement the specified, time-varying upstream concentration boundary condition exactly even though a larger time step value of 2.0 days is taken elsewhere.

In this example, the execution time for the traditional approach with small time step of 0.5 days is equal to 135 s, while it is equal to 120 s in the case of the sub-timing approach.
approach. Therefore, it may appear that not much is gained by the way of application of sub-timing. However, it may be noted that the governing equation in the present case is linear, and also the Jacobian of the system does not vary with time because of the steady flow field. Therefore, LU decomposition of the Jacobian is performed up front before starting the time-marching computations, and only right-hand-side assembly and back substitutions are performed during each time step. The sub-timing approach results in a Jacobian which is larger in size, requiring extra effort while performing LU decomposition. This extra effort is compensated for during the time marching computations with the sub-timing approach which uses lesser number of time steps compared to the traditional time discretization approach. This is precisely what is observed when computations are performed for longer simulation periods. For a simulation period of 365 days, the traditional approach takes 353 s while the sub-timing approach takes only 210 s, giving a speed up factor of 1.65, thus demonstrating the utility of the sub-timing approach. It should be stressed again, that the above measures of computational efficiency are provided as a proof-of-concept, and to demonstrate that actual efficiencies are typically lower than the theoretical considerations provided earlier. Different sparse matrix solvers have different characteristics, and their convergence properties would differ depending on diagonality and other considerations, therefore, it is not the intent here to analyze their efficiencies.

6. Conclusions

This study discusses a sub-time stepping method for computational fluid dynamics problems that utilize implicit formulations to obtain transient solutions. The proposed technique is particularly suitable for extensive simulations where portions of a domain are temporally over-discretized by conventional implicit time-stepping approaches. Such temporal over-discretization occurs in problems where activity and interest is high in only a small portion of the domain, for example regions close to boundary, regions around shocks etc. Also, non-linearities in flow or transport processes may lead to temporal over-discretization in many practical situations.

The principles underlying implementation of the sub-timing procedure, and the computational effort in relation to conventional time-stepping methods are presented. It is shown that the speed up obtained by the sub-timing procedure as compared to a conventional method depends on, among other things, (i) the percentage of grid points where sub-timing is applied, and (ii) the matrix solver used for solving the simultaneous algebraic equations resulting from the discretization. The need for a computationally efficient and robust iterative sparse matrix solver along with the sub-timing approach is further demonstrated by an analysis of the effects of sub-timing on matrix structure. The ‘‘detrimental’’ effects of sub-timing on the matrix characteristics are minimized for a fully implicit time discretization.

The feasibility, applicability and accuracy of the implicit sub-timing method is demonstrated through three proof-of-the-concept example problems. These examples include implementation of sub-timing method for solving (i) the one-dimensional zero-inertia unsteady surface water flow equation, and (ii) the two-dimensional solute transport equation. The one-dimensional examples demonstrate the application of sub-timing to avoid temporal over discretization in a conventional method, arising from rapidly varying boundary conditions, and non-linearities in the problem. The two-dimensional example of solute transport demonstrates the application of sub-timing along with the sub-gridding approach for the advective-dispersive equation. It is further shown that the sub-timing method is faster than the conventional method in the above examples. A combination of implicit sub-timing with other techniques such as adaptive implicit schemes, implicit–explicit combination methods, nested grids, and efficient iterative sparse matrix solvers may provide even higher computational efficiencies necessary to analyze large complex systems, even on today’s fastest computers.

References


