EVALUATION OF THE MATRIX EXPONENTIAL FOR USE IN GROUND-WATER-FLOW AND SOLUTE-TRANSPORT SIMULATIONS: THEORETICAL FRAMEWORK

By Amjad M. J. Umari and Steven M. Gorelick

U.S. GEOLOGICAL SURVEY

Water-Resources Investigations Report 86-4096

Albuquerque, New Mexico

1986

UNITED STATES DEPARTMENT OF THE INTERIOR

DONALD PAUL HODEL, Secretary

GEOLOGICAL SURVEY

Dallas L. Peck, Director

For additional information
write to:

District Chief
U.S. Geological Survey
Water Resources Division
505 Marquette NW, Room 720
Albuquerque, New Mexico 87102

Copies of this report can
be purchased from:

Open-File Services Section
Branch of Distribution
U.S. Geological Survey, MS 306
Box 25425, Denver Federal Center
Denver, Colorado 80225
(303) 236-7476

CONTENTS

ILLUSTRATIONS

TABLES

EVALUATION OF THE MATRIX EXPONENTIAL FOR USE IN GROUND-WATER-FLOW AND SOLUTE-TRANSPORT SIMULATIONS: THEORETICAL FRAMEWORK

By Amjad M. J. Umari and Steven M. Gorelick

ABSTRACT

It is possible to obtain analytic solutions to the ground-water-flow and solute-transport equations if space variables are discretized but time is left continuous. From these solutions, hydraulic-head and concentration fields for any future time can be obtained without "marching" through intermediate time steps. This analytical approach involves matrix exponentiation and is referred to here as the Matrix Exponential Time Advancement (META) method. Two algorithms are presented for the META method, one for symmetric and the other for non-symmetric exponent matrices. A numerical accuracy indicator, referred to as the matrix condition number, is defined and is used to determine the maximum number of significant figures that may be lost in the META method computations.

The relative computational and storage requirements of the META method with respect to the time-marching method are shown to increase with the number of nodes in the discretized problem. The potential greater accuracy of the META method and the associated greater reliability through use of the matrix condition number have to be weighed against this approach's increased relative computational and storage requirements as the number of nodes becomes large. For a particular number of nodes, the META method may be computationally more efficient than the time-marching method, depending on the size of time steps used in the latter. A numerical example is given to illustrate application of the META method to a sample ground-water-flow problem.

INTRODUCTION

The standard approach to obtain transient solutions for the ground-water-flow and solute-transport equations is to discretize space and time variables and then advance the solution one step at a time. An alternate approach to this time-marching method is to discretize space variables but leave the time variable continuous in the flow and solute-transport equations. It is then possible to use matrix exponentiation to obtain analytic solutions to these spatially discretized equations. These analytic solutions give hydraulic-head and solute-concentration fields for any future time without the need to "march" through intermediate times.

This approach has been applied to cases where hydraulic-head and solute-concentration fields are needed either explicitly (as in direct problems) or implicitly (as in optimization problems) for a future time, but not for intermediate times. Kuiper (1973), for example, used the matrix exponential, allied with finite-element spatial discretization, to obtain a direct solution to a transient-flow problem. Willis (1979) used the matrix exponential to obtain analytic response equations to use as constraints in a ground-water-quality management model.

It is the objective of this report to investigate the accuracy and reliability of this analytic approach based on matrix exponentiation and to compare its computational and storage requirements to those of the conventional time-marching method. This project was funded by the U.S. Geological Survey, Office of Hazardous Waste Hydrology. The authors are grateful to Dr. Cleve Moler, INTEL Scientific Computers, for valuable discussions regarding the mathematics of the report.

Problem Formulation

After spatial discretization has been carried out, but before the time variable is discretized, the ground-water-flow and solute-transport equations can be written in the form:

$$[A]\frac{d\underline{h}}{dt} + [B]\underline{h} + \underline{F} = \underline{0} \qquad (1)$$

$$[E]\frac{d\underline{c}}{dt} + [G]\underline{c} + \underline{Q} = \underline{0} \qquad (2)$$

Equations 1 and 2 may be obtained from the original flow and solute-transport partial differential equations by either finite differences or the Galerkin method (see "Supplemental information"). In the above equations, [A], [B], [E], and [G] are square coefficient matrices of order n (the number of nodes in the spatial discretization grid), whereas $\underline{F}$ and $\underline{Q}$ are column vectors of order n representing boundary conditions and mass/solute sources or sinks. $\underline{h}$ and $\underline{c}$ are vectors of nodal hydraulic heads and concentrations, respectively. The coefficient matrix [G] depends on flow velocities, which depend on the vector of hydraulic heads, $\underline{h}$, as determined by equation 1.

To obtain $\underline{h}(t)$ or $\underline{c}(t)$ for a future time t, the time derivatives in equations 1 and 2 may be written in the form:

$$\left.\begin{aligned}\frac{d\underline{h}}{dt} &= \frac{\underline{h}^{k+1} - \underline{h}^{k}}{\Delta t}\\ \frac{d\underline{c}}{dt} &= \frac{\underline{c}^{k+1} - \underline{c}^{k}}{\Delta t}\end{aligned}\right\} \qquad (3)$$

where the superscript indicates the time step. This formulation, or another version of it, can be used to obtain $\underline{h}(t)$ and $\underline{c}(t)$ from equations 1 and 2 for any time in the future by simply "marching" through time and setting k = 1, 2, 3, ...(see "Supplemental information").

An alternative approach for obtaining $\underline{h}(t)$ or $\underline{c}(t)$ is to use the analytic solutions for equations 1 and 2 (Bellman, 1960), which, after simplifying, can be written as:

$$\underline{h}(t) = e^{-[M]t}\left(\underline{h}_o + [B]^{-1} \underline{F} \right) - [B]^{-1} \underline{F} \qquad (4)$$

$$\underline{c}(t) = e^{-[N]t}\left(\underline{c}_o + [G]^{-1} \underline{Q} \right) - [G]^{-1} \underline{Q} \qquad (5)$$

where $[M] = [A]^{-1}[B]$, $[N] = [E]^{-1}[G]$, and $\underline{h}_o$ and $\underline{c}_o$ are the initial condition vectors. The implicit requirement that $\underline{F}$ and $\underline{Q}$ be constant in time can be overcome by superposing the solutions obtained from each pumping period. The other requirement, that of a constant [G], restricts equation 5 to a steady flow pattern.

To satisfactorily compute $\underline{h}(t)$ and $\underline{c}(t)$ from the vector-matrix equations 4 and 5, an accurate and reliable method has to be employed to evaluate the matrix-exponential terms $e^{-[M]t}$ and $e^{-[N]t}$. The method of obtaining $\underline{h}(t)$ and $\underline{c}(t)$ by use of equations 4 and 5 is referred to here as the Matrix Exponential Time Advancement method, or the META method.

The matrix exponential $e^{[P]}$ of an n x n matrix [P] is an n x n matrix defined by the convergent Taylor power series:

$$e^{[P]} = [I] + [P] + \frac{1}{2!}[P]^2 + \frac{1}{3!}[P]^3 + \cdots \qquad (6)$$

where [I] is the identity matrix. Equation 6 should be considered only as a definition for the matrix exponential not as a process to compute $e^{[P]}$.

Purpose and Approach

The purpose of this report is to: (1) Present two reliable algorithms for computing the matrix-exponential terms of equations 4 and 5; and (2) through presentation of these two algorithms, examine the reliability and

accuracy of the META method, and compare its computational efficiency and computer-storage requirements with those of the conventional time-marching approach. One of these algorithms is specifically suitable for computing the matrix-exponential term in equation 4, whereas the other is specifically suitable for computing the matrix-exponential term in equation 5. Equations 4 and 5 require different algorithms due to the different structures of their coefficient matrices [M] and [N].

A MATRIX-EXPONENTIAL ALGORITHM FOR THE GROUND-WATER-FLOW EQUATION

Rather than use the spatially discretized flow equation in the form of equation 1 and its analytic solution as given by equation 4 where the coefficient matrix [M] to be exponentiated is generally not symmetric, equation 1 is transformed into an equivalent one for which the analytic solution involves the exponential of a symmetric matrix. This is done because the process of exponentiating a matrix is facilitated and strengthened when the matrix is symmetric. The steps of the proposed algorithm, which involves the above transformation, are presented below.

Algorithm Steps

(a) Assume that [A] is diagonal and positive and that [B] is symmetric (as would result from a finite-difference spatial approximation for example). Transform the system of first order, ordinary differential equations represented by the vector-matrix equation 1 into an equivalent system by defining a matrix [U] such that $[U]^2 = [A]$. Then:

$$\frac{d\underline{z}}{dt} + [R]\underline{z} + \underline{S} = \underline{0} \qquad (7)$$

where $[R] = [U]^{-1}[B][U]^{-1}$ and is symmetric, $\underline{z} = [U]\underline{h}$ and is the new dependent variable, and $\underline{S} = [U]^{-1}\underline{F}$. The analytic solution of equation 7

comparable to equation 4 is given by:

$$\underline{z}(t) = e^{-[R]t} (\underline{z}_o + [R]^{-1} \underline{S}) - [R]^{-1} \underline{S} \quad (8)$$

(b) Because [R] is symmetric, a specialized EISPACK (Eigensystem Package) subroutine, namely BANDR (Garbow and others, 1977), can be efficiently used to decompose [R] as follows:

$$[R] = [V] [D] [V]^{-1} \quad (9)$$

where [V] is a matrix whose columns are the eigenvectors of [R], and [D] is a diagonal matrix with diagonal elements being the real eigenvalues of [R].

(c) Once [R] is decomposed in the form of equation 9 the following can be written (Van Loan, 1975; Moler and Van Loan, 1978):

$$e^{-[R]t} = [V] e^{-[D]t} [V]^{-1} \quad (10)$$

where $e^{-[D]t}$ is a diagonal matrix with the components $e^{-\lambda_i t}$ ($i = 1, 2, \ldots, n$). The quantities λ_i are eigenvalues of [R] and constitute the diagonal of [D]. Therefore, $e^{-[R]t}$ is computed from equation 10 and substituted into equation 8 to obtain $\underline{z}(t)$.

(d) The inverse of the transformation employed in step (a) is used to obtain $\underline{h}(t)$ from $\underline{z}(t)$.

The transformation described under (a) can also be employed for the more general case of a symmetric and positive [A] as would result from a finite-element spatial discretization. In this case $[U] = [A]^{\frac{1}{2}}$ is given by:

$$[U] = [V'] [D']^{\frac{1}{2}} [V']^T \quad (11)$$

where [V'] and [D'] are the matrix of eigenvectors and the matrix of eigenvectors of [A].

Decomposition of [R] by BANDR

In addition to being symmetric, [R] of equation 7 is banded because [B] of equation 1, which is used in constructing [R], is banded. Because [R] is symmetric and banded, it can be stored efficiently by computers, and due to its special structure can be decomposed efficiently, as indicated in equation 9, by EISPACK's subroutine BANDR (Garbow and others, 1977). This subroutine is written specifically to obtain eigenvalues and eigenvectors of banded, symmetric, real matrices.

An important advantage of [R] being symmetric, beyond computer-storage economy and computational efficiency, is that it has an orthogonal set of eigenvectors (the columns of [V]) and real eigenvalues (the diagonal elements of [D]). Because the columns of [V] are orthogonal, [V] is an orthogonal matrix and, therefore, its transpose $[V]^T$ is equal to its inverse. But because $[V]^T$ is easy to compute, there is a computational advantage to using it in equation 9 instead of $[V]^{-1}$, which is costly to compute. A further advantage of orthogonality of the eigenvectors is that it leads to maximum accuracy when equation 10 is used to compute the matrix exponential $e^{-[R]t}$ and consequently to maximum accuracy in computing $\underline{h}(t)$ (see "Accuracy, reliability, and the matrix condition number" beginning on page 14).

A MATRIX-EXPONENTIAL ALGORITHM FOR THE SOLUTE-TRANSPORT EQUATION

An overview of the algorithm (Moler and Van Loan, 1978) is presented below. (Details of steps a and b are shown later in this section.)

(a) Matrix [N] of equation 5 is decomposed--primarily through use of EISPACK'S subroutine HQR2 (Smith and others, 1974)--as follows:

$$[N] = [V_1]\ [D_1]\ [V_1]^{-1} \tag{12}$$

In equation 12, $[V_1]$ is a matrix whose columns are the eigenvectors of [N], and $[D_1]$ is a quasi-diagonal matrix of eigenvalues. $[D_1]$ is not quite diagonal like its counterpart [D] in the ground-water-flow algorithm (see equation 9), because, unlike [R], [N] is not symmetric and may have complex as well as real eigenvalues (this is discussed further below). The two steps of (1) forming [N] from [E] and [G], and (2) obtaining the eigensystem of [N] may be combined into one step. This is done by use of the QZ algorithm (Golub and Van Loan, 1983), which uses [E] and [G] directly to obtain the eigensystem of $[N] = [E]^{-1}[G]$ without explicitly forming [N].

(b) Based on the decomposition represented by equation 12, the following can be written:

$$e^{-[N]t} = [V_1]\ e^{-[D_1]t}\ [V_1]^{-1} \tag{13}$$

where $e^{-[D_1]t}$ is not as simple to compute as its counterpart $e^{-[D]t}$ in the ground-water-flow algorithm because $[D_1]$ is quasi-diagonal rather than diagonal.

(c) Once $e^{-[N]t}$ is computed using equation 13, it can be substituted into equation 5 to obtain $\underline{c}(t)$.

Decomposition of [N] by HQR2

Given the non-symmetric, real matrix [N], EISPACK'S subroutine HQR2 and its associated routines ORTRAN and ORTHES may be modified to produce the orthogonal matrix $[Q_1]$, the upper quasi-triangular matrix $[R_1]$, and the quasi-diagonal matrix $[D_1]$, such that:

$$[N] = [V_1]\ [D_1]\ [V_1]^{-1} \tag{12}$$

where

$$[V_1] = [Q_1]\ [R_1] \tag{14}$$

The structure of upper quasi-triangular $[R_1]$ and quasi-diagonal $[D_1]$ for a 4 x 4 case is shown in figure 1.

$$[R_1] = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & m & \ell & 7 \\ 0 & -\ell & m & 1 \\ 0 & 0 & 0 & 2 \end{bmatrix} \qquad [D_1] = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & b & c & 0 \\ 0 & -c & b & 0 \\ 0 & 0 & 0 & \lambda_4 \end{bmatrix}$$

Figure 1.--Structure of an upper quasi-triangular matrix $[R_1]$ and a quasi-diagonal matrix $[D_1]$.

$[D_1]$ has scalar (1 x 1) elements along its diagonal that are the real eigenvalues of [N] (like λ_1 and λ_4 in figure 1), and 2 x 2 blocks corresponding to complex conjugate pairs of eigenvalues of [N] (like the submatrix $\begin{bmatrix} b & c \\ -c & b \end{bmatrix}$ in figure 1). The complex conjugate pair of eigenvalues of [N] associated with the 2 x 2 submatrix $\begin{bmatrix} b & c \\ -c & b \end{bmatrix}$ are b + ci and b - ci, where $i = \sqrt{-1}$. (b + ci and b - ci are not only eigenvalues of [N] but also of the submatrix $\begin{bmatrix} b & c \\ -c & b \end{bmatrix}$.)

Similarly, $[R_1]$ would be upper triangular if it were not for 2 x 2 blocks along its diagonal (like the 2 x 2 submatrix $\begin{bmatrix} m & \ell \\ -\ell & m \end{bmatrix}$ in figure 1) that protrude into the lower triangle of the matrix making it not quite upper triangular but upper quasi-triangular. Associated with the 2 x 2 submatrix $\begin{bmatrix} m & \ell \\ -\ell & m \end{bmatrix}$ are the complex conjugate pair of eigenvectors $\begin{Bmatrix} m + i\ell \\ -\ell + im \end{Bmatrix}$ and $\begin{Bmatrix} m - i\ell \\ -\ell - im \end{Bmatrix}$ which are the eigenvectors of $\begin{bmatrix} b & c \\ -c & b \end{bmatrix}$, the 2 x 2 submatrix along the diagonal of $[D_1]$. By employing the definition relating an eigenvalue λ and an eigenvector $\underline{v}$ of a matrix [w], namely $[w]\underline{v} = \lambda\underline{v}$, the following can be written:

$$\begin{bmatrix} b & c \\ -c & b \end{bmatrix} \begin{Bmatrix} m + i\ell \\ -\ell + im \end{Bmatrix} = (b + ci) \begin{Bmatrix} m + i\ell \\ -\ell + im \end{Bmatrix} \tag{15}$$

and

$$\begin{bmatrix} b & c \\ -c & b \end{bmatrix} \begin{Bmatrix} m - i\ell \\ -\ell - im \end{Bmatrix} = (b - ci) \begin{Bmatrix} m - i\ell \\ -\ell - im \end{Bmatrix} \tag{16}$$

This ends the discussion on the content and structure of $[D_1]$ and $[R_1]$.

To complete the decomposition of [N] as presented by equations 12 and 14, $[V_1]^{-1}$ has to be computed. From equation 14, it can be seen that:

$$[V_1]^{-1} = [R_1]^{-1} [Q_1]^{-1} \tag{17}$$

But because $[Q_1]$, which HQR2 may be modified to produce, is orthogonal, its transpose is equal to its inverse. Therefore, the decomposition of [N] as presented by equations 12 and 14 can be restated as follows:

$$[N] = [V_1]\ [D_1]\ [V_1]^{-1} \tag{12}$$

where

$$[V_1] = [Q_1]\ [R_1] \tag{14}$$

and

$$[V_1]^{-1} = [R_1]^{-1}\ [Q_1]^T \tag{18}$$

In equation 18, the computation of $[Q_1]^T$ is simple (the rows of $[Q_1]$ become columns of $[Q_1]^T$), and the computation of $[R_1]^{-1}$ is simplified due to the upper quasi-triangular structure of $[R_1]$.

Computation of $e^{-[D_1]t}$ and $e^{-[N]t}$

Now, $e^{-[N]t}$ can be expressed in terms of the components of [N] by equation 13. As previously mentioned, $[D_1]$ is quasi-diagonal due to the asymmetry of [N]. Computing $e^{-[D_1]t}$, therefore, involves exponentiation of the 2 x 2 blocks along the diagonal of $[D_1]$, as illustrated in figure 2, that is:

$$[s] = \begin{bmatrix} s_1 & s_2 \\ s_3 & s_4 \end{bmatrix} = e^{-\begin{bmatrix} b & c \\ -c & b \end{bmatrix} t} = e^{-[T]t} \tag{19}$$

$$[D_1] = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & b & c & 0 \\ 0 & -c & b & 0 \\ 0 & 0 & 0 & \lambda_4 \end{bmatrix} \qquad e^{-[D_1]t} = \begin{bmatrix} e^{-\lambda_1 t} & 0 & 0 & 0 \\ 0 & s_1 & s_2 & 0 \\ 0 & s_3 & s_4 & 0 \\ 0 & 0 & 0 & e^{-\lambda_4 t} \end{bmatrix}$$

$$[s] = \begin{bmatrix} s_1 & s_2 \\ s_3 & s_4 \end{bmatrix} = e^{-\begin{bmatrix} b & c \\ -c & b \end{bmatrix} t} = e^{-[T]t}$$

Figure 2.--Exponentiation of quasi-diagonal matrix for 4 x 4 case.

Because of the specific structure and size of [T], the following analytic solution (Cleve Moler, INTEL Scientific Computers, written commun., 1985) may be derived:

$$[s] = e^{-[T]t} = \begin{bmatrix} e^{-bt}\,\mathrm{Cos}\,(-ct) & e^{-bt}\,\mathrm{Sin}\,(-ct) \\ -e^{-bt}\,\mathrm{Sin}\,(-ct) & e^{-bt}\,\mathrm{Cos}\,(-ct) \end{bmatrix} \tag{20}$$

Substituting $e^{-[T]t}$ as computed above for $[s] = \begin{bmatrix} s_1 & s_2 \\ s_3 & s_4 \end{bmatrix}$ along the diagonal of $e^{-[D_1]t}$ (as illustrated in fig. 2), and doing the same thing for other locations along the diagonal of $e^{-[D_1]t}$ corresponding to 2 x 2 blocks along the diagonal of $[D_1]$, $e^{-[D_1]t}$ will be determined.

Having determined $e^{-[D_1]t}$, it is substituted into equation 13:

$$e^{-[N]t} = [V_1]\ e^{-[D_1]t}\ [V_1]^{-1} \tag{13}$$

where

$$[V_1] = [Q_1]\ [R_1] \tag{14}$$

and

$$[V_1]^{-1} = [R_1]^{-1}\ [Q_1]^T \tag{18}$$

to obtain $e^{-[N]t}$. If equation 13 is written as: $[X]\ [V_1] = [S_1]$ where $[X] = e^{-[N]t}$ and $[S_1] = [V_1]\ e^{-[D_1]t}$, this system can be solved directly for the unknown $[X] = e^{-[N]t}$ without explicitly inverting $[V_1]$.

An alternative to the exponentiation of the 2 x 2 matrices along the diagonal of $[D_1]$ is to use complex arithmetic, making the entries along the diagonal of $[D_1]$ complex scalar entries (Hwang and others, 1984). An advantage of this approach is that an equation similar to equation 2 can be separated into n equations one for each $c_i(t)$, and only the ones for which $c_i(t)$ is desired are solved. A disadvantage of the approach is the costly complex arithmetic.

ACCURACY, RELIABILITY, AND THE MATRIX CONDITION NUMBER

If the matrix to be exponentiated is symmetric like [R] of equation 8, its eigenvectors can be chosen to be orthogonal (the ultimate in linear independence) as noted under "Decomposition of [R] by BANDR." This "well conditioned" eigensystem contributes to maximum accuracy in computing $e^{-[R]t}$ by equation 10 and consequently to maximum accuracy in computing $\underline{z}(t)$ by equation 8. This accuracy dependence on the "condition" of the exponentiated matrix's eigensystem can be quantified and made precise by definition of the "condition number" of the matrix of eigenvectors:

$$\text{Cond}(V) = \left\|V\right\| \cdot \left\|V^{-1}\right\| \tag{21}$$

This number can be estimated without inverting [V] (Dongarra and others, 1979). If expressed as a power of 10 (for example, 10^0 for the present symmetric exponent matrix [R]), the condition number of the eigensystem indicates through its exponent (zero in this case), the maximum number of significant figures that may be lost in the digital floating-point computation of $e^{-[R]t}$ and $\underline{z}(t)$ by equations 10 and 8.

The condition number can be thought of as the "degree of defectiveness" of a particular matrix. In pure mathematics, there is either a defective matrix (one with an incomplete set of linearly independent eigenvectors, that is, a defective eigensystem), or a non-defective matrix (one with a complete set of linearly independent eigenvectors, that is, a non-defective eigensystem). In computational mathematics, the realm of the present analysis, there are matrices in the gray area between the defective and non-defective extremes, which are referred to as "nearly defective" matrices, and the condition number as defined by equation 21 can be used as an indicator of the location of a particular matrix in this continuum. If Cond(V) is expressed as a power of 10, say 10^n, it denotes a degree of defectiveness of order n.

When the matrix to be exponentiated is symmetric (like [R] of equation 8) and consequently has an orthogonal system of eigenvectors, the condition number of its matrix of eigenvectors is 10^0 (indicating a degree of defectiveness of order 0, non-defective matrix), and no significant figures

are lost in computing $\underline{h}(t)$. Furthermore, because the matrix of eigenvectors of [R], namely [V], is orthogonal (the ultimate in linear independence), it is also invertible (non-singular), and so $[V]^{-1}$ can be evaluated and computations represented by equation 10 to obtain $e^{-[R]t}$ become feasible.

If the matrix to be exponentiated is not symmetric, like [N] in equation 5, the eigensystem may not be well conditioned, which causes the matrix to be nearly defective. The condition number of its matrix of eigenvectors, $[V_1]$ of equation 12, should then be computed to establish the degree of defectiveness of [N] and the possible loss of accuracy resulting from it. $\text{Cond}(V_1)$ may, for example, be 10^4 indicating that [N] has a degree of defectiveness of order 4. This results in a maximum possible loss of four significant figures in floating-point computations when $e^{-[N]t}$ is computed by equation 13 and $\underline{c}(t)$ by equation 5. This may be acceptable if double-precision computations are used, which provide sixteen significant digits of accuracy, because losing four leaves twelve significant figures. If single precision is used, which only provides eight significant figures of accuracy, losing four of them may be intolerable. The potential problem of accuracy loss (when the matrix to be exponentiated is not symmetric) is an insidious one because, although the matrix [N] may be nearly defective, $[V_1]$ can be invertible, so calculation of $e^{-[N]t}$ according to equation 13 and $\underline{c}(t)$ according to equation 5 become feasible but will not give any indication that a loss of accuracy may have taken place.

A computer model that uses matrix exponentiation of a non-symmetric matrix, whether it is through implementation of the algorithm presented under "A matrix-exponential algorithm for the solute-transport equation" or any other algorithm, needs to be provided with a flag to be triggered whenever the matrix condition number indicates the possibility of excessive loss of accuracy.

COMPUTATIONAL AND STORAGE REQUIREMENTS OF THE META METHOD

The computational requirements of a process can be quantified by indicating the number (in order of magnitude only) of floating point operations (or "flops") needed by the process, to which cost of the process is proportional:

$$[\text{Process I}] = f(n_1, n_2, \ldots) \tag{22}$$

This expression means that the number of flops needed by Process I is proportional to some function $f(n_1, n_2, \ldots)$, where n_1, $n_2, \ldots$ are process parameters.

The computational requirements of the time-marching method, which will be the basis for evaluating the relative requirements of the META method, will now be obtained. For flow, the time-marching approach involves repeated solution of a system of linear algebraic equations of the form of equation A5 (developed in "Supplemental information"):

$$[AA] \quad \underline{h}^{k+1} = \underline{FF} \tag{A5}$$

The matrix [AA] is given in terms of the coefficient matrices of the spatially discretized flow equation 1 as:

$$[AA] = \frac{1}{\Delta t}[A] + [B] \tag{A6}$$

For solute transport, the equations corresponding to equations A5 and A6 would be:

$$[EE] \quad \underline{c}^{k+1} = \underline{QQ} \tag{23}$$

$$[EE] = \frac{1}{\Delta t}[E] + [G] \tag{24}$$

where [E] and [G] are coefficient matrices of the spatially discretized

solute-transport equation 2.

Because [A], [B], [E], and [G] are banded, it follows that [AA] and [EE] as defined by equations A6 (developed in "Supplemental information") and 24 are also banded. Assuming that the band width of [AA] and [EE] is d, the amount of work involved in solving equations A5 or 23 once, when the time-marching (TM) method is used, is proportional to d^2n where n is the number of nodes in the spatial discretization grid. Using the notation of equation 22, the following can be written:

$$[TM] = d^2n \text{ for one time step} \tag{25}$$

This estimate does not take into consideration some efficiency measures that may be incorporated into a time-marching algorithm. One such measure is use of iterative, rather than direct, techniques for solving linear systems. Another is use of a single matrix factorization for a sequence of equal time steps when direct solution techniques are used.

To simplify the following discussion of the computational requirements of the META method relative to those of the time-marching approach, the discussion is restricted to square, two-dimensional, physical grids with $\sqrt{n}$ by $\sqrt{n}$ nodes. This implies that if the nodes are numbered sequentially parallel to either of the coordinate axes of the grid, the band width of the resulting coefficient matrices, when spatial discretization is carried out, is given by:

$$d = 2\sqrt{n} + 1 \tag{26}$$

Computational requirements of the algorithm presented under "A matrix-exponential algorithm for the ground-water-flow equation," and referred to here as the META-1 method, are obtained by considering the component processes that involve a large number of flops. The major component processes (and the

associated number of flops) for the META-1 method are those involving computation of:

(1) [V] of equation 9 by BANDR, dn^2;

(2) [D] of equation 9 by BANDR, d^2n;

(3) $[V]e^{-[D]t}$ of equation 10 by matrix multiplication where $e^{-[D]t}$ is diagonal, n^2;

(4) $[V]e^{-[D]t}[V]^{-1}$ of equation 10 by matrix multiplication, n^3;

(5) $[R]^{-1}\ \underline{S}$ of equation 8 by solving the banded system of equations $[R]\underline{x} = \underline{S}$ for $\underline{x} = [R]^{-1}\ \underline{S}$, d^2n; and

(6) $e^{-[R]t}\ (\ z_o + [R]^{-1}\ \underline{S}\ )$ of equation 8 by vector-matrix multiplication, n^2.

Summing up the number of flops required for the above six major component processes, and using the notation of equation 22, the following can be written:

$$[\text{META-1}] = 2d^2n + (2+d)n^2 + n^3 \tag{27}$$

Similarly, the major component processes (and the associated number of flops) for the META-2 method as presented under "A matrix-exponential algorithm for the solute-transport equation," are those involving the computation of:

(1) $[N] = [E]^{-1}\ [G]$ for use in equation 5 by solving the n banded systems of equations $[E]\underline{x}_1, = \underline{G}_1$, $[E]\underline{x}_2 = \underline{G}_2, \ldots$ where $\underline{G}_1, \underline{G}_2, \ldots$ are the columns of [G], and $\underline{x}_1, \underline{x}_2, \ldots$ will form the columns of [N], $d^2n + n(2dn)$;

(2) $[V_1]$, $[V_1]^{-1}$, and $[D_1]$ by HQR2, as indicated by equations 12, 14, and 18, n^3;

(3) $[V_1]e^{-[D_1]t}$ of equation 13 by matrix multiplication where $e^{-[D_1]t}$ is almost diagonal, n^2;

(4) $[V_1]e^{-[D_1]t}[V_1]^{-1}$ of equation 13 by matrix multiplication, n^3;

(5) $[G]^{-1}\underline{Q}$ of equation 5 by solving the banded system of equations $[G]\underline{x} = \underline{Q}$ for $\underline{x} = [G]^{-1}\underline{Q}$, d^2n; and

(6) $e^{-[N]t}(\underline{C}_o + [G]^{-1}\underline{Q})$ of equation 5 by vector-matrix multiplication, n^2.

Summing up the number of flops required for the above six major component processes, and using the notation of equation 22, the following can be written:

$$[\text{META-2}] = 2d^2n + 2(1+d)n^2 + 2n^3 \tag{28}$$

The relative computational requirements of the META-1 and the META-2 methods with respect to the time-marching (TM) method are:

$$K(\text{META-1}) = [\text{META-1}]/[\text{TM}] \tag{29}$$

$$K(\text{META-2}) = [\text{META-2}]/[\text{TM}] \tag{30}$$

where the computational requirements [META-1], [META-2], and [TM] are given by equations 27, 28, and 25, respectively. Because equation 25 gives the computational requirements for only one time step of the time-marching method, K(META-1) as given by equation 29 represents the "break-even" point for the META-1 method, or the number of time steps of the time-marching method at which its computational requirements (or cost) "break even" with those of the META-1 method. Similarly, K(META-2) represents the number of time steps of the time-marching method at which its computational requirements break even with those of the META-2 method.

If the expressions for [TM], [META-1], and [META-2] (equations 25, 27, 28) are substituted into the definitions for K(META-1) and K(META-2) as given by equations 29 and 30, and because $d \simeq 2\sqrt{n}$ (equation 26), the following can be written:

$$K(\text{META-1}) = 2\tfrac{1}{2} + \tfrac{1}{2}\sqrt{n} + n/4 \qquad (31)$$

$$K(\text{META-2}) = 2\tfrac{1}{2} + \sqrt{n} + n/2 \qquad (32)$$

Typical values based on equations 31 and 32 are K(META-1) = $32\frac{1}{2}$, K(META-2) = $62\frac{1}{2}$ for n = 100; and K(META-1) = $74\frac{1}{2}$, K(META-2) = $146\frac{1}{2}$ for n = 256.

It is seen from equations 31 and 32 that as the number of nodes (n) in the spatially discretized grid increases, it takes a proportionally increasing number of steps of the time-marching method for its cost to break even with that of the META-1 and META-2 methods. So equations 31 and 32 indicate that from a computational-efficiency standpoint, the META method (in general) becomes less attractive--compared to the time-marching method--as the size of the discretized problem, namely n, increases. For a specific size n, the META method may be computationally more efficient than the time-marching method depending on the size of the time steps used for the latter, as discussed in the following paragraph.

From K(META-1) and K(META-2) given by equations 31 and 32 for a particular grid size n, and the size of the time steps used in the time-marching method, the actual simulation time at which the break-even point is reached (the actual break-even simulation time) can be obtained. Although K(META-1) and K(META-2) may appear large for a particular n, the size of the time steps used in the time-marching method may be small enough (due to numerical stability restrictions) that the actual break-even simulation time may not be large. If that is the case, and the actual break-even simulation time is smaller than the required simulation time for the particular application, the META-1 and META-2 methods would be computationally more efficient than the time-marching approach for the grid size n in question.

The terms n/4 of equation 31 and n/2 of equation 32 are the major contributors to the increase in relative computational requirements (or diminishing relative efficiency) of the META method with increasing n. The first term, n/4 of equation 31, reflects work associated with performing the full matrix multiplication of $[V]e^{-[D]t}[V]^{-1}$ in the META-1 method, which is the cause of the n^3 term in equation 27. The second term, n/2 of equation 32, reflects work in the META-2 method of: (1) Computing the eigenvalues and eigenvectors of [N] by EISPACK'S HQR2 as indicated by equation 12, which contributes an n^3 to the $2n^3$ term of equation 28; and (2) performing the full matrix multiplication of $[V_1]e^{-[D_1]t}[V_1]^{-1}$, which contributes the other n^3 to the $2n^3$ term of equation 28.

Matrix multiplication of full matrices and eigensystem computations by EISPACK'S HQR2 are the two major reasons for the increasing relative computational requirements (or diminishing relative efficiency) of the META method with increasing n. The first of these processes, that of matrix multiplication of full matrices, can be improved in efficiency by use of array processors: electronic hardware equipment that can be made to interface with whatever machine is being used to do the computations.

As for computation of the eigensystem for [N] of equation 5 by EISPACK'S HQR2 in the META-2 method, the core of the inefficiency is that HQR2 considers the matrix for which it seeks to compute eigenvalues and eigenvectors as a general one and does not take advantage of structural features of the matrix (such as the bandedness in [N]). If EISPACK, or any other eigensystem package, acquires the capability of taking advantage of matrix bandedness--as it is now possible to take advantage of banded symmetry through use of EISPACK'S BANDR -- computational requirements of obtaining the eigensystem of [N] in the META-2 method will be greatly reduced. Also, utilization of the QZ algorithm, which uses [E] and [G] to obtain the eigensystem of $[N] = [E]^{-1}[G]$ without explicitly forming [N], would make the META-2 algorithm more efficient.

For minimum storage requirements, the following number of locations are needed for the time-marching (TM), META-1, and META-2 methods:

$$\text{TM: } (3+2d)n \text{ locations} \tag{33}$$

$$\text{META-1: } (6\tfrac{1}{2} + 1\tfrac{1}{2}d)n + 2n^2 \text{ locations} \tag{34}$$

$$\text{META-2: } (7\tfrac{1}{2} + 1\tfrac{1}{2}d)n + 3n^2 \text{ locations} \tag{35}$$

Expression 33 assumes use of direct solution techniques. If iterative techniques are used, storage requirements of the time-marching approach would be less than indicated.

Use of high-speed scientific computers, such as the CDC 6600, CDC 7600, STAR-100, CYBER-203, and CRAY-1, would reduce the absolute computing time for the META method to such an extent that considerations regarding its relative computational requirements (or relative computing time) with respect to the time-marching method become irrelevant. Also, such computers have very large memories and can easily handle the greater storage requirements of the META method indicated by equations 34 and 35. Choice between the two methods might then depend on superior accuracy as discussed below.

As shown under "Accuracy, reliability, and the matrix condition number," the accuracy in temporal computations associated with the META method can be quantified by use of the condition number of the matrix of eigenvectors. With it, the maximum number of significant figures that may be lost in the computations can be predicted. For a symmetric exponent matrix (as is the case in the algorithm presented under "A matrix-exponential algorithm for the ground-water-flow equation"), the matrix of eigenvectors is orthogonal, leading to maximum possible accuracy and no loss in significant figures. There is no comparable process by which the accuracy in temporal computations associated with the time-marching method can be quantified, given a particular time discretization scheme and stepping strategy.

NUMERICAL EXAMPLE

In this section, the procedures presented in this report are applied to a ground-water-flow problem where the spatial discretization is done by the finite-element method (Galerkin Formulation). These procedures have been also applied by the authors to a solute-transport problem (Umari and Gorelick, 1986).

The areal layout of the ground-water-flow problem, modified from Willis and Newman (1977) and shown in figure 3, has a well field of five wells (circles) adjacent to a river (two lines connecting three triangles at top of figure). The aquifer from which the wells are pumping is semi-confined. The physical domain is discretized into twenty triangular elements and sixteen nodal points. Every element has a three-component row vector associated with it; the first component indicates the element number, the second the property zone, and the third the recharge zone. For example (19, 4, 2) indicates that element 19 is in property zone 4 and recharge zone 2. The characteristics of the four property zones defined for this problem are presented in table 1. Note that no units are shown; any consistent set of units (hereafter referred to as "dimensionless") is acceptable for the parameters and variables of this problem.

Table 1.--Dimensionless x conductivity (K_x), y conductivity (K_y), and storage coefficient (S) for the four property zones

Zone	K_x	K_y	S
1	1.45	0.36	0.01
2	144.80	36.07	0.01
3	0.14	0.04	0.01
4	72.39	9.02	0.01

EXPLANATION

2 NODE NUMBER

▲ CONSTANT-HEAD RIVER NODE

● WELL NODE

(19,4,2) ROW VECTOR (ELEMENT NUMBER, PROPERTY ZONE, RECHARGE ZONES)

NOTE. ALL THE DISTANCES ARE DIMENSIONLESS.

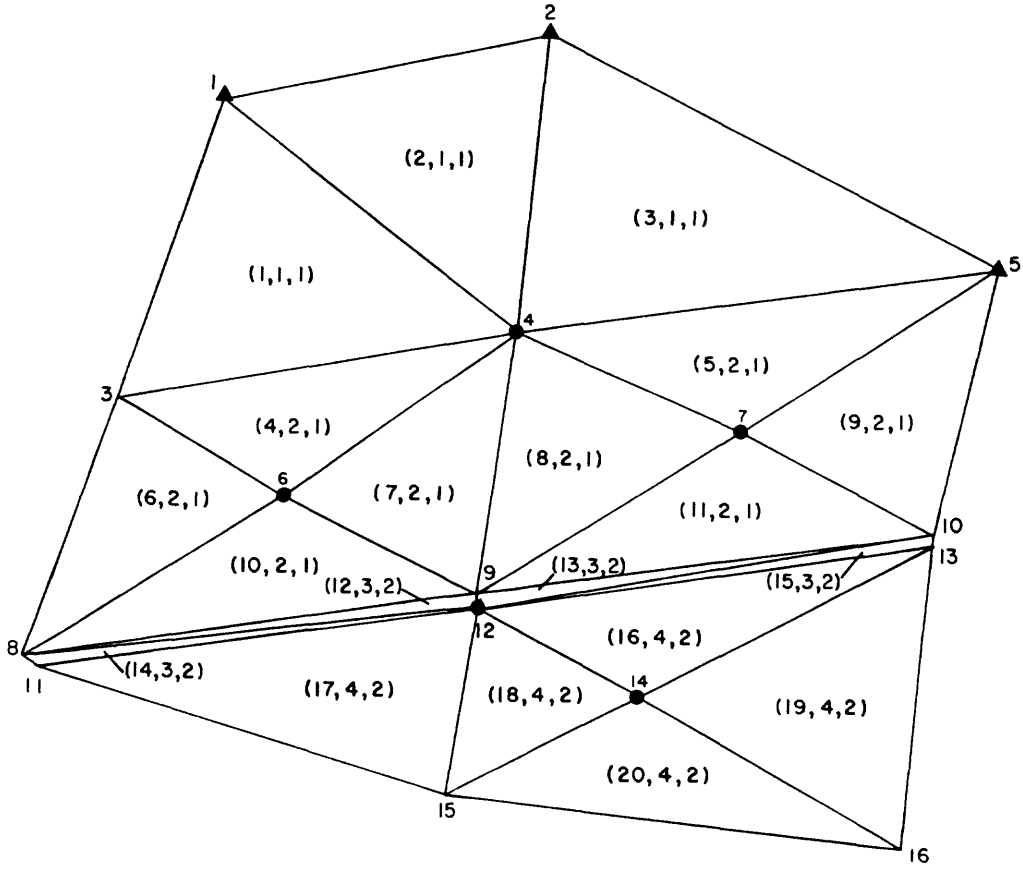

Figure 3.--Physical layout of the numerical example.

Recharge zone 1 represents the leaky part of the semi-confined aquifer, which is overlain by a confining bed for which the dimensionless vertical hydraulic conductivity divided by the thickness ($K_z/\Delta z$) is 0.0000223 and in which the hydraulic head is held constant at 60.96. Recharge zone 2 represents the totally confined part of the semi-confined aquifer.

The five wells are located at nodes 4, 6, 7, 12, and 14 below the river, which is represented by line segments connecting nodes 1, 2, and 5 and forms the upper boundary of the grid. The hydraulic head at the river nodes is held at the constant value of 60.96, which is also the initial value for heads at all the nodes in the grid. Apart from these constant-head river nodes, heads at all other nodes are allowed to change. The five wells are pumped at the same constant rate of 60.00 dimensionless units. It is desired to obtain the piezometric surfaces at some specific future time by the conventional time-marching approach and the META method and to compare these surfaces.

The spatial discretization of the flow equation to obtain equation 1 for use of both methods is done by using the triangular-finite-element computer code of Wiggert (1974). For the time-marching approach, the Crank-Nicolson approximation of the time derivative (rather than the simpler forward-difference approximation represented by the set of equations 3) is used, based on Wiggert's program.

An arbitrary time of 14.25 is chosen to compare results of the two methods. The results are presented in table 2. As shown in table 2, results obtained by the two methods are practically identical.

For the time-marching solution, 29 time steps were used, the first of length 0.25 and the remaining of length 0.50 dimensionless time units. The computations, which were performed on an IBM 370/168, took 1 second of Central Processing Unit (CPU) time, 1 second of Input/Output (I/O) time, and required 80 kilobytes (80K) of memory for an execution charge of $0.34. For the META method, the CPU time was 3 seconds, the I/O time 3 seconds, and the memory requirement 170K for an execution charge of $1.63. The cost differential is actually less severe because the time-marching program was already compiled at the beginning of the execution sessions, whereas the META program was not.

Table 2.--The piezometric surface at time 14.25 as obtained by the conventional time-marching approach and the META method

Node	Time-marching approach	META algorithm	Boundary conditions
1	60.96	60.9600	Dirichlet
2	60.96	60.9600	Dirichlet
3	60.96	60.9583	
4	60.84	60.8495	
5	60.96	60.9600	Dirichlet
6	60.82	60.8235	
7	60.81	60.8136	
8	60.97	60.9767	
9	60.97	60.9726	
10	60.93	60.9367	
11	61.01	61.0104	
12	60.56	60.5640	
13	61.12	61.1187	
14	60.66	60.6643	
15	60.90	60.8956	
16	60.97	60.9674	

SUMMARY AND CONCLUSIONS

Two algorithms to obtain analytic solutions for $\underline{h}(t)$ and $\underline{c}(t)$ of the spatially discretized flow and solute-transport equations 1 and 2 have been presented. These analytic solutions involve obtaining the exponential of a matrix, which in both algorithms is done by manipulating the eigensystem of the matrix to be exponentiated.

The first algorithm, especially suited for the flow problem when spatial discretization has been carried out by the finite-difference method, employs a transformation that makes the matrix to be exponentiated symmetric. This symmetry leads to an orthogonal system of eigenvectors and maximum accuracy. Because the matrix to be exponentiated is banded and symmetric, EISPACK'S efficient BANDR routine can be used to obtain its eigensystem. Because of symmetry, the eigenvalues are all real, thereby simplifying the matrix-exponentiation process.

The second algorithm, especially suited for exponentiating non-symmetric matrices (like the ones that arise in the solute-transport problem), obtains the necessary matrix decomposition (which involves the eigensystem) by use of EISPACK'S HQR2 routine. Due to non-symmetry of the matrix to be exponentiated, some eigenvalues may be complex, which makes it necessary to exponentiate 2 x 2 matrices that represent conjugate pairs of these complex eigenvalues. The exponentiation of these 2 x 2 matrices is done by using an analytic solution derived especially for them.

The accuracy of the method of obtaining analytic solutions for $\underline{h}(t)$ and $\underline{c}(t)$ of equations 1 and 2 through the process of exponentiating a matrix is quantified by use of the matrix condition number. This number indicates the maximum number of significant figures that can be lost in the computations. Use of the matrix condition number gives the META method a high degree of reliability in that the user can ascertain ahead of time the degree of accuracy in temporal computations associated with the final solution for $\underline{h}(t)$ and $\underline{c}(t)$. There is no counterpart to the matrix condition number in the conventional time-marching method.

The relative computational and storage requirements of the META method, with respect to the time-marching method, were found to increase with the number of nodes in the spatial-discretization grid. It is suggested that the potential higher accuracy of the META method and its reliability acquired through use of the matrix condition number should be weighed against its increasing relative computational and storage requirements compared to the time-marching method when a choice between the two methods is made. For a particular grid size, the META method may be computationally more efficient than the time-marching method, depending on the size of the time steps used in the latter.

A numerical example is given in which a ground-water-flow problem is spatially discretized by the finite-element method. The solution for $\underline{h}(t)$ obtained by the META method and that obtained by the time-marching method are compared for a specific future time t. The two solutions are found to be practically identical.

REFERENCES

Bear, Jacob, 1972, Dynamics of fluids in porous media: New York, American Elsevier, 764 p.

Bellman, R. E., 1960, Introduction to matrix analysis: New York, McGraw-Hill, 328 p.

Dongarra, J. J., Moler, C. B., Bunch, J. R., and Stewart, G. W., 1979, Linpack, user's guide: Society of Industrial and Applied Mathematics, 356 p.

Faddeev, D. K., and Faddeeva, V. N., translated by Williams, R. C., 1963, Computational methods of linear algebra: San Francisco, W. H. Freeman, 621 p.

Garbow, B. S., Boyle, J. M., Dongarra, J. J., and Moler, C. B., 1977, Matrix eigensystem routines - EISPACK guide extension: New York, Springer-Verlag Lecture Notes in Computer Science 51, 343 p.

Golub, G. H., and Van Loan, Charles, 1983, Matrix computations: Baltimore, Johns Hopkins University Press, 476 p.

Hwang, J. C., Cho, W. C., and Yeh, G. T., 1984, An eigenvalue solution continuous in time to the spatially discretized solute transport equation in steady groundwater flow: Water Resources Research, v. 20, no. 11, p. 1725-1732.

Jacob, C. E., 1950, Flow of groundwater, in Rouse, Hunter, ed., Engineering hydraulics: New York, John Wiley, p. 321-386.

Kuiper, L. K., 1973, Analytic solution of spatially discretized groundwater flow equations: Water Resources Research, v. 9, no. 4, p. 1094-1097.

Moler, Cleve, and Van Loan, Charles, 1978, Nineteen dubious ways to compute the exponential of a matrix: Society of Industrial and Applied Mathematics (SIAM) Review, v. 20, no. 4, p. 801-836.

Smith, B. T., Boyle, J. M., Garbow, B. S., Ikebe, Y., Klema, V. C., and Moler, C. B., 1974, Matrix eigensystem routines - EISPACK guide: New York, Springer-Verlag Lecture Notes in Computer Science 6, 387 p.

Umari, A. M. J., and Gorelick, S. M., 1986, The problem of complex eigensystems in the semianalytical solution for advancement of time in solute transport simulations: A new method using real arithmetic: Water Resources Research, v. 22, no. 7, p. 1149-1154.

Van Loan, Charles, 1975, A study of the matrix exponential: Manchester (England), University of Manchester, Department of Mathematics Report No. 10, 52 p.

REFERENCES - Concluded

Wiggert, D. C., 1974, Two dimensional finite element modeling of transient flow in regional aquifer systems: Lansing, Michigan, Michigan State University, Institute of Water Research Technical Report No. 41, 102 p.

Willis, Robert, 1979, A planning model for the management of groundwater quality: Water Resources Research, v. 15, no. 6, p. 1305-1312.

Willis, Robert, and Newman, B. A., 1977, Management model for ground-water development: American Society of Civil Engineers, Journal of the Water Resources Planning and Management Division, v. 103, no. WRI, p. 159-171.

SUPPLEMENTAL INFORMATION

Flow and Solute-Transport Equations

Based on the principle of conservation of mass, Jacob (1950) derived an equation for ground-water flow, which, for two dimensional (areal) problems, can be written as follows:

$$\nabla \cdot (\underline{\underline{T}}\ \nabla h) \pm P = S \frac{\partial h}{\partial t} \qquad \text{(A1)}$$

where ∇ is the del operator ($\nabla = \frac{\partial}{\partial x}\ \underline{i} + \frac{\partial}{\partial y}\ \underline{j}$, where $\underline{i}$ and $\underline{j}$ are the unit vectors in the x and y directions);

"$\cdot$" indicates a dot product;

$\underline{\underline{T}}$ is the transmissivity tensor, L^2/T;

h is the hydraulic head, L^1;

+P is the total recharge rate while (-P) would be the total pumping rate, L^3/T; and

S is the storage coefficient (confined aquifer) or the specific yield (unconfined aquifer), L^0.

Equation A1, which was derived for a confined aquifer, can be also used for an unconfined aquifer, if the assumption is made that drawdown in the hydraulic head is small relative to the saturated thickness of the aquifer. This assumption would be valid for a regional analysis but not for near-well studies.

Similarly, based on the principle of conservation of a certain constituent's mass, the two-dimensional (areal) solute-transport equation for a conservative constituent can be written as follows (after Bear, 1972):

$$\nabla \cdot (\underline{\underline{D}}\ \nabla c) - \nabla \cdot (c\ \underline{q}) + Q = \frac{\partial c}{\partial t} \qquad \text{(A2)}$$

where $\underline{\underline{D}}$ is the hydrodynamic dispersion tensor, L^2/T;

c is the dissolved concentration of the constituent, M/L^3;

$\underline{q}$ is the flux of ground-water flow as given by Darcy's law, L/T; and

Q is the mass source rate through injection, M/T.

$\underline{q}$ is related to the gradient of the hydraulic head, ∇h, by Darcy's equation:

$$\underline{q} = -\underline{\underline{k}} \nabla h \qquad \text{(A3)}$$

where $\underline{\underline{k}}$ is the hydraulic conductivity tensor.

Time-Marching Approach

Substituting the finite difference approximation given by equation 3 for $\frac{d\underline{h}}{dt}$ in equation 1, one can write:

$$[A] \left\{ \frac{\underline{h}^{k+1} - \underline{h}^{k}}{\Delta t} \right\} + [B]\, \underline{h}^{k+1} + \underline{F} = \underline{0} \qquad \text{(A4)}$$

Using $\underline{h}^{k+1}$ (rather than $\underline{h}^{k}$) for $\underline{h}$ in the term $[B]\underline{h}$ makes this an implicit approach, in which a set of simultaneous linear equations has to be solved at each time step. Equation A4 can be re-written as:

$$[AA]\ \underline{h}^{k+1} = \underline{FF} \qquad \text{(A5)}$$

where

$$[AA] = \frac{1}{\Delta t}[A] + [B] \qquad \text{(A6)}$$

and

$$\underline{FF} = \frac{1}{\Delta t} [A] \underline{h}^k - \underline{F} \qquad (A7)$$

Solution of equation A5 at each time step produces a sequence of solutions $\underline{h}^1$, $\underline{h}^2$, ... corresponding to the sequence of time steps employed. The process stops when the desired time level has been reached.

The process of marching through time using the spatially discretized solute-transport equation 2 and obtaining the sequence of solutions $\underline{c}^2$, $\underline{c}^3$, $\underline{c}^4$,..., is exactly analogous to that presented above for the flow equation, and will not be presented here.

☆U.S. GOVERNMENT PRINTING OFFICE:1986—672-628 / 45300